Computer Science at UC Berkeley: The Consequences and Considerations of Running Courses at Scale

Alex Schedel
Joshua Hug, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-177
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-177.html

May 15, 2023

Computer Science at UC Berkeley:
The Consequences and Considerations of Running Courses at Scale

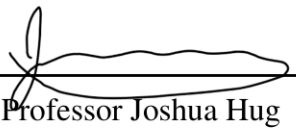
by Alexander Schedel

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:



Professor Joshua Hug
Research Advisor

May 12, 2023

* * * * * * *

Professor John DeNero
Second Reader

May 12, 2023

Acknowledgements

To compile a list of all those to whom I am indebted would be impossible, and even if such a list were to exist words would fail to express my thanks to every person who occupied a place on it. Despite this, I would like to attempt a few modest thanks. I would like to thank Professor Paul N. Hilfinger, whose iteration of CS61B in the Fall of 2018 served as the greatest educational experience of my life. Were it not for Professor Hilfinger and that class, I doubt this report would exist, and I doubt that I would have ever truly understood my place in computer science. I also wish to thank my advisor, Professor Josh Hug, for always understanding my creative impulses as both an educator and a student. Thank you for helping me to understand what it means to be an artist in STEM. Perhaps most importantly however, I wish to express my deepest gratitude to CS61B, the class in which I have spent 12 semesters as a student, intern, uGSI, and finally GSI, and everyone who has been a part of it, it is all of you who have many my time at UC Berkeley worthwhile.

As for this report itself, I wish to extend a special thanks to the TAs who have helped and allowed me to collect information on their courses for the analysis below. Thank you to Anton Zabreyko and Meshan Khosla of CS61B, Tianchen Liu of CS170, Kanu Grover of Data100, Laryn Qi of CS61A, and Anjali Thakrar of CS184/284a.

Contents

List of Figures

List of Tables

Abstract

The demand for a degree in computer science has exploded exponentially in recent decades, creating massive pressure on universities to scale up the size of their CS programs. The efforts of the UC Berkeley Electrical Engineering and Computer Science (EECS) Department to facilitate this expansion while maintaining academic quality stand out as a unique insight into the considerations and consequences of developing and maintaining a program at scale. This report details three aspects of course structure which require fundamental overhauls in order to be run at scale: systems for extensions and late work, mechanisms for running office hours, and techniques for proctoring and facilitating exams. Each of these subsections contain a number of considerations and case studies into the efforts of specific courses to facilitate themselves at scale in an effective manner. As a whole, this report serves as both a record and a guide for instructors of university courses, at UC Berkeley or otherwise, on the basics of running courses at scale.

Introduction

Degrees in computer science are more popular than ever before. Not only is such a degree a gateway to high paying careers and vast employment opportunities, it also affords one a position at the forefront of an increasingly technologized world. As such, it should come as no surprise that demand for Computer Science degrees has too skyrocketed in recent decades. Despite this, many university level CS programs have not drastically increased enrollment in this period, instead opting to make their programs more selective and simply deny entry to more and more applicants each year. One notable exception to this trend is UC Berkeley, which has consistently scaled up the size of its CS program over the past two decades to become one of the largest at any University. This means more students, bigger class sizes, more supporting personnel, and a number of other largely ad hoc and informal techniques to operate the program at an increasing scale.

The report will detail many of the techniques that courses within the program have experimented with and embraced in order to service courses at scale. In particular, many of the student facing policies of CS61B, one of the largest Computer Science classes at UC Berkeley, will be examined alongside other courses within the Berkeley EECS department. This report is intended as a dialogue, both between courses within UC Berkeley which are currently grappling with the issues of running programs at scale, as well as departments at other universities, who may be facing similar issues in either upsizing the scale of their own CS programs or looking to expand in the future.

Running Courses at Scale

1.1 Extension Mechanisms and Late Work

One of the great banes of any professor is late work. It clogs up courses and requires resources be spent resolving grades and details of assignments which are ostensibly no longer relevant due to the passage of time. For a conventionally sized university class, late work often represents an inconvenience which might require a bit of extra attention and bookkeeping on the part of the instructor(s). For a course at scale however, lateness in turning assignments proves to be a massive logistical challenge to surmount, as potentially hundreds of students may require extensions and special exemptions. This section of the report will examine some of the techniques employed in recent years in UC Berkeley EECS Courses to deal with late work and assignments, as well as their pedagogical implications and popular reception. These techniques range from relatively simple one size fits all policies to more fine grain individual specific policies.

CS170 Spring 2022 - No Extensions

Description and Motivation

Perhaps the most obvious and simple solution to the issue of late work is simply to not accept it at all. Under this policy, a deadline for each assignment will be set, after which point no submissions will be accepted, even in the face of extreme circumstances. The appeal of such a system is its simplicity and ease of implementation. It requires no overhead in managing or allotting of extensions, no mechanism for adjusting or retabulating grades after the fact, and keeps students on pace with the course, as they have little incentive to continue work on a particular assignment once its deadline passes. Additionally, as courses at scale often operate with dozens of individuals on each course staff, there is no training required in running this system.

In addition to this simplicity of implementation, there are pedagogical justifications for not accepting late work as well. The primary one is that it keeps students moving through the course at a controlled pace. A major issue with extensions in general is that they often create a situation wherein students have many assignments running concurrently, creating a domino effect where an extension on one assignment results in the student having less time to work on another assignment, which makes them need an extension for that other assignment as well.

CS170 is an EECS Course that employs such a policy while also allowing students some homework drops in order to alleviate pressure. The policy reads in full:

"We accept absolutely no submissions after 11:59pm, even after technical issues or emergencies. No exceptions... At the end of the semester, we will automatically drop your two lowest homeworks (including any you may have received a 0 on). [3]

An additional and relevant policy in the class is that any homework which receives a 90% or higher will be awarded full credit, giving students leeway in getting a full score on course assignments.

Results

Assignment	Number of Submissions	Percent Unsubmitted*	Percent Submitted Scoring <90%	Near Midterm?
Homework 1	563	1.4	71.9	No
Homework 2	544	4.7	37.9	No
Homework 3	500	12.4	63.4	No
Homework 4	526	7.9	69.0	Yes
Homework 5	553	2.0	43.6	No
Homework 6	470	13.3	79.4	No
Homework 7	488	10.0	35.9	No
Homework 8	516	4.8	57.6	No
Homework 9	512	5.5	50.1	Yes
Homework 10	516	4.8	58.1	No
Homework 11	478	11.8	64.0	No
Homework 12	502	7.4	61.8	No

Table 1.1: Submission patterns of CS170 students.

As students can drop courses throughout the semester, the total number of students at any point in the course was estimated as the number of people who took midterm 1 (571) for homeworks 1 - 4 and the number of people who took midterm 2 and the final (both 542) for all other homeworks.

As is to be expected with this system, students seldom submit work late, instead opting to simply use their homework drops if needed.

The table provides insight into how students actually use their late drops. Students tend to drop the hardest homeworks. Many of the homeworks with the highest percentage of students who scored below 90% also have the highest rate of no submissions. This behavior maximizes the perceived "usefulness" of homework drops, but comes at an expense. The irony of this type of drop policy is that it encourages students to put the least amount of effort into the hardest assignments. This means that if a course staff is using a similar system and is aware that a particular assignment is disproportionately challenging, it is best to make the assignment easier or split it into multiple parts, otherwise students are unlikely to give it their best effort. That said, even if a course staff is able to balance all the homeworks, the system ultimately encourages students to not do at least some of their work. For example, if a student gets full credit on all homeworks in a semester up until the end, there is no grade based reason for them to attempt the last two homeworks because they will be worth nothing due to the lateness policy.

A potential issue with this system is that it forces students to gamble on their situation due to the difficulty of estimating the ramifications of not attempting a particular assignment. On the one hand, if a student is confident that they will be able to score a 90% or greater on all assignments, then that drop saves them some time and effort, with the caveat that they might have to do additional studying later to learn the homework content they missed out on. On the other hand however, if the student is not confident that they will be able to score a 90% on their other assignments, then a 0% received for not attempting a homework will be dropped instead of another low grade they have, ultimately hurting their final grade.

There is a perceptible decline in submissions of the last two homeworks. This implies that many students who were able to achieve perfect or near perfect scores on all the previous homeworks simply did not attempt the last two. This is another potential issue with giving students a finite resource which they can expend to ease the workload of a class, at the end of the semester they will use whatever they have left all at once, meaning that students overall are less likely to have a good understanding of the last few assignments in a course.

Another note is that homeworks which were due around midterms do not seem to differ substantially in turn in rate compared with those that were not. Although it might stand to reason that students are more likely to take a drop deliberately before a midterm in order to allow themselves more time to study, this does not seem to have happened in CS170. A likely reason for this is that the drop eligibility rate for every assignment is very high.

Implementation and Staff Results

As alluded to above, much of the appeal of this system is that it can be implemented basically automatically. No extensions means no staff having to deal with accounting for lateness or potentially different deadlines for different portions of the class. The system scales extremely well with the number of students and virtually no additional overhead is required as the class expands

Student Perception

The main note regarding student perception of this policy is understanding how students conceptualize the homework drops. Although from a staff perspective, homework drops are often thought of as a way to deal with emergency situations, students typically think of them as a kind of currency. The homework drops act as “get out of jail free” cards when it comes to assignments, and therefore should be “spent” in a way as to maximize their “value”. Additionally, when students are in actual emergency situations such as sudden hospitalization, it is common for them to ask for more drops or extensions, implying that students see drops as something they should always be able to use consciously and by choice, rather than something meant to help them in situations they cannot control.

CS184/284a Spring 2022 - Allotted Slip / Late Days

Description and Motivation

Another common approach to dealing with lateness is a slip / late day policy. Under such a policy, courses grant a specific number of slip days to students at the beginning of the semester. The student can then use these slip days on various assignments in order to extend the deadline for themselves. This creates a finite and guaranteed amount of extension time which can be used by students based on how it works best for them. Additionally, unlike the drop system, slip days are much more transparently used at the student's discretion, meaning that it is expected they will take slip days in non-emergency situations. The motivation for such a system is to provide a limited amount of flexibility for students and to provide better granularity in extension policy.

One class that employed such a policy of CS184/284a, Computer Graphics in Spring 2022. The class's slip day policy reads as follows:

"Each student has five late days for the semester. Late days apply to regular programming assignments only and not the final project. You can extend a programming assignment deadline by 24 hours using one late day. If you do not have remaining late days, late hand-ins will incur a 1 point penalty per day. Late days are meant to account for submission issues and other unforeseen circumstances." [2]

Results

Assignment	1 Slip Day	2 Slip Days	3 Slip Days	4 Slip Days	5 Slip Days	Total Slip Days Used	% Late Subs
Project 1	45	26	9	6	6	178	30.1%
Project 2	47	15	7	3	0	110	24.1%
Project 3-1	31	40	80	22	10	489	62.2%
Project 3-2	45	9	2	0	0	69	19.2%
Project 4	42	76	56	27	4	490	46.1%

Table 1.2: Submission patterns of CS184/284a students. [5]

There were approximately 300 students in CS184/284a in Spring 2022, and a grand total of 1336 slip days were used. This means that the average student used 4.45 of their 5 slip days, quite close to the allotted limit. This acts as evidence that, much like the homework drops of CS170, students seek to use as many slip days as they can by the end of the semester.

In terms of how many slip days were used on a particular project, projects 1, 2, and 3-2 demonstrate a trend wherein students are less likely to use slip days the farther from the dead-

line they get. This shows that students ration slip days and make an effort to use as few of them as possible until the end of the semester.

Project 3-1, often considered to be the most challenging project of the class, is a notable exception in this trend. This project had the highest rate of late submissions of any project, with the vast majority of the class using some amount of their slip days on it. Additionally, peak slip day usage occurred on day 3 and then sharply dropped thereafter. It is likely that many students felt they needed to take a longer extension due to the difficulty of the project, and also that they could afford to spend 3 of their 5 slip days because they had gone through the first two projects using few if any at all.

Project 4 is also worth examining. More slip days were used on this project than any other, amounting to a number quite close to the total number of slip days used in project 3-1. However, 62.2% of students used at least one slip day on project 3-1, whereas only 46.1% of students used at least one slip day on project 4. This means that although a similar number of slip days were used across the two projects, the average length of extension was much longer for project 4 than for project 3-1. This, combined with the fact that most students used all or nearly all their slip days by the end of the semester, shows that, much like in the CS170 case study above, when given a finite resource, students will spend anything they have left at the end of the semester.

Implementation and Staff Results

Much like the CS170 policy, this policy is relatively simple from a staff perspective. There is some additional overhead in managing the number of slip days that students have, but overwhelmingly there is little need for staff to personally intervene or handle exceptions on a case by case basis.

Student Perception

As with the homework drop system, slip days are typically conceived of as a sort of currency by students. Students typically ration slip days as best they can until the final project, and they dump whatever remains at the end of the semester.

CS61B Fall 2021 - Lateness by Hour

Description and Motivation

Much like the CS184/284a policy outlined above, the policy of CS61B in Spring 2022 was based on time, however there are some notable differences. The first is that rather than a slip day system which worked on a daily basis, this system was hourly. Additionally, the system had an active penalty that was applied per hour. In full the policy reads:

"For the first 24 hours, we will deduct 5/24 percent of the maximum score (not including extra credit) for each hour it is late, rounded off in some unspecified fashion. After that, the penalty rises to 5/12 percent for each additional hour late. This means that you will lose roughly 5% of the points for a project for the first day late and 10% for each subsequent day (though tracking is by hours late, not days). We will not accept projects submitted more than 72 hours late."

The reasoning for this approach is further outlined within the course policy page itself:

- Lenient late policies tended to encourage people to put off working on projects until it was too late to finish them in time.
- The habit of assuming that deadlines are optional is probably not a good one to carry into the "real world."
- Students would work on a project for many days after the deadline, falling farther behind on other work, when the wiser course would have been to hand in what they had and cut their losses.

In this version of CS61B, 50% of the grade was determined by the score on 4 large programming projects. [11]

Results

Assignment	Within 24 Hours	Within 48 Hours	Within 72 Hours	Late / Total Submissions	% Late Subs
Project 0	42	29	24	95 / 945	10.1%
Project 1	183	114	62	359 / 921	39.0%
Project 2	94	41	54	189 / 865	21.8%
Project 3	93	51	89	223 / 742	31.4%

Table 1.3: Submission patterns of CS61B students.

Note that only a student's final submission is recorded, meaning that it is quite likely that those submitting on day three had also been submitting on days one and two. Additionally, This chart only indicates that students made a late submission, not that it improved their grade. A student who submitted on day three and received no credit for any submission would still be counted here as submitting on day three.

The flowing charts show how many submissions were made per hour for each assignment:

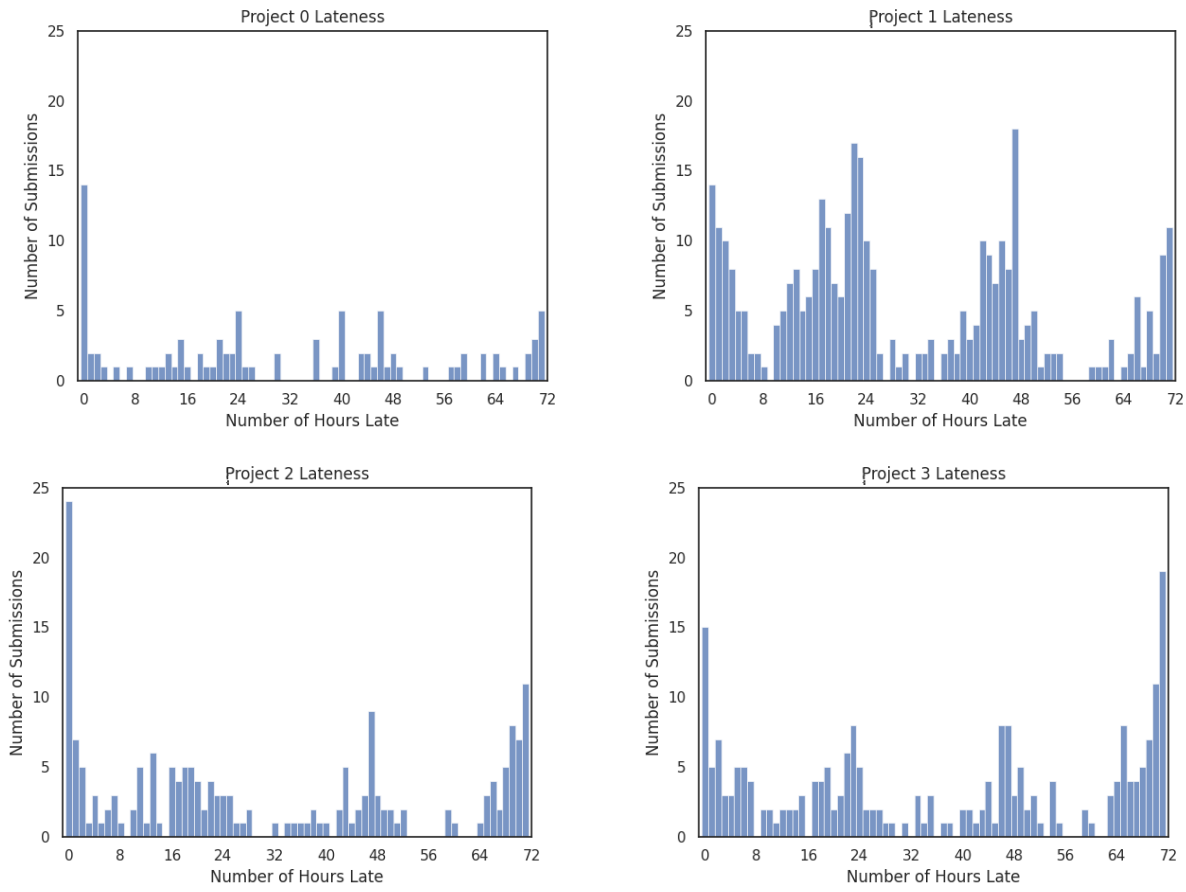


Figure 1.1: Lateness by assignment.

There was a grand total of 875 late submissions over the semester made by 525 unique individuals in a class of roughly 900 students. This means that although across the whole class the average student submitted about one assignment late, students who submitted at least one late assignment submitted about 1.6 late assignments. In other words, students who submit one assignment late are quite likely to submit another one late as well.

This policy is designed to encourage students to catch up quickly when it comes to late work, as the longer one waits, the more points will be marked off of each project. This seems to have

been a successful motivator in regard to projects 0 and 1, both of which saw consistent declines in submissions each day after the deadline. However, the policy seems more mixed in regard to projects 2 and 3, both of which saw a spike in submissions within the last day of the lateness window. This likely indicates that many students who were not able to complete the project in full within the lateness window simply made a final hail mary attempt to recover some points on the final day.

Implementation and Staff Results

From a staff perspective, there are a number of advantages to this system. One is that it gives students flexibility to complete the project late, while still guaranteeing a firm deadline for late work. Unlike a slip day system, there is always an absolute deadline on which everyone will no longer be eligible to submit. This means that course staff will not have to worry about being prepared to help students well after deadlines, and can also plan the course schedule to make sure students do not end up overloaded by extending a project into the work time for the next project. Additionally, there is also a degree of predictability within the lateness window. For all assignments, the day after the deadline is the most popular day to submit late. This pattern allows a course to taper access to resources and shift focus onto another assignment over time naturally as fewer and fewer students will ask questions as the lateness window goes on.

Another staff side advantage is that the system is quite simple to implement and understand. The exact number 5/24 points off per hour (meant to ensure that each 24 hour period late resulted in a 5% deduction) could sometimes be difficult for students to fully grasp, but the general idea of a constant reduction by hour over three days was easily understandable.

Student Perception

Although this system is relatively simple to implement and understand, it does come with some major qualitative downsides, the most notable of which can be gleaned from the figure below, which combines data from all four projects.

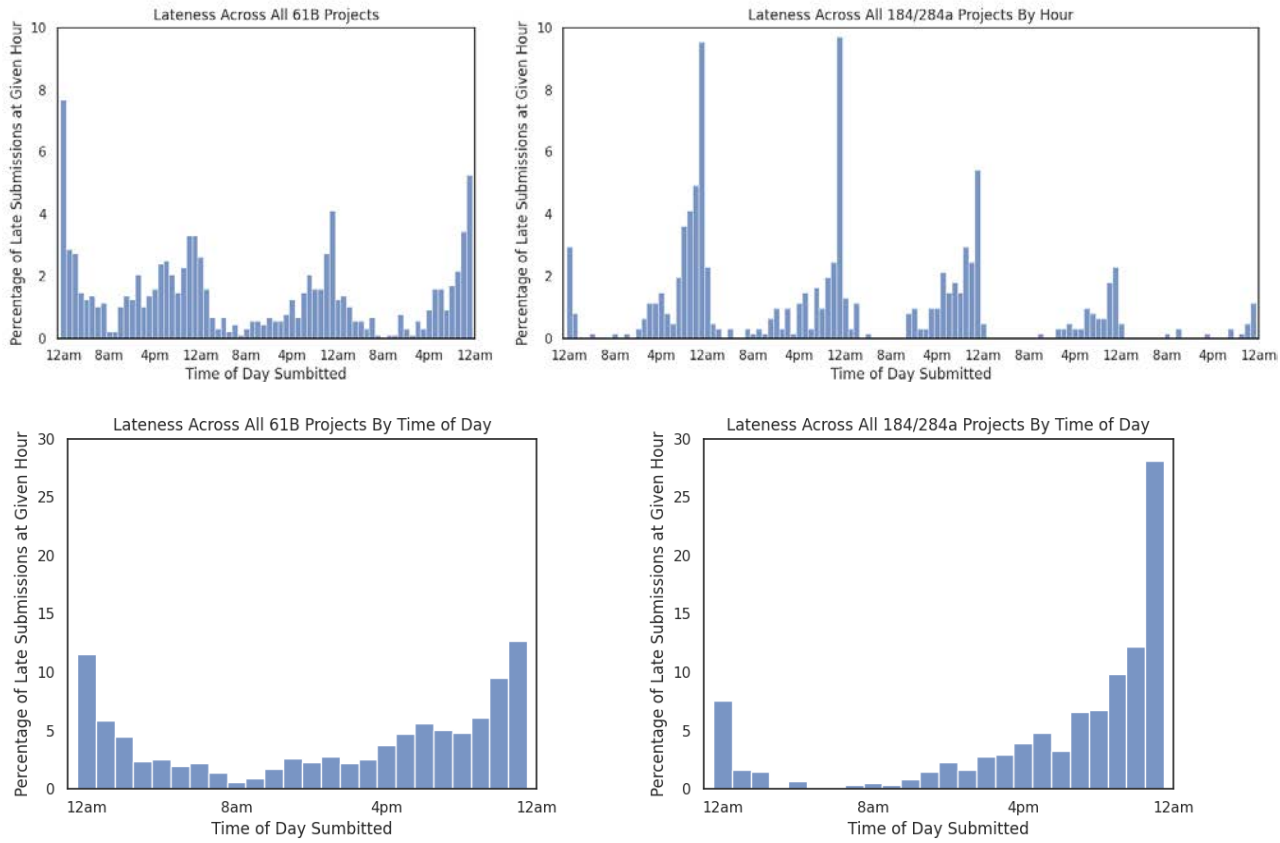


Figure 1.2: Times of day and on what days late assignments were submitted in CS61B and CS184/284a.

Deadlines in 61B were always at 11:59pm, meaning that the hourly nature of the system encouraged students to work unusual hours and late into the night to maximize their grades. Of all the extensions submitted on the first day of each project, 41.5% of them were submitted between 12am and 8am. 19.2% were submitted between 2am and 8am. In fact, the period from 8am to 4pm, what most would likely consider to be reasonable working hours, is actually the least popular time to submit. This, combined with the popularity of late night submissions, implies that students often stayed up late to work on the their assignments rather than simply sleep at night and continue in the morning to minimize their lateness penalty

Further evidence that this policy encouraged disruptive sleep patterns can be seen when comparing submission times in CS61B to those in CS184/284a. Recall that CS184/284a used a slip day policy that had the granularity of a day rather than an hour. In general, students rarely submit assignments late at night in CS184/284a, whereas they often do in CS61B. This demon-

strates that the hourly lateness policy encourages students to stay up late and finish projects in a way that a daily system simply does not.

It will likely come as no surprise that the hourly lateness policy was divisive among students. The sleeplessness encouraged by the policy disproportionately affects those already behind and most likely to require additional time to complete assignments. Additionally, the policy enters students into a game theory situation where they have to decide whether is it better to stay up late and produce likely inferior quality work due to lack of sleep, or to simply sleep, wake up the next morning fresh, and finish the project, albeit with a slight penalty due to taking the rest. Considering that the chart above shows that the least popular time to submit was 8am - 4pm each day, it seems students overwhelmingly favored the former option. In this way, the act of sleeping and resting itself almost becomes a sort of gamble to a student, causing a great deal of physical and mental strain on the student body in days after large deadlines. Many students conceptualized the sleep deprivation caused by the system as a kind of "punishment" for not completing work on time. This was further exacerbated by the fact that many members of course staff, who during normal hours would be available to answer and help with student questions online, were asleep in the immediate wake of the deadline.

CS61B Spring 2023 - Autoextensions

Description and Motivation

The above policies are largely motivated by the fact that they issue a simple blanket policy that applies to all students and requires little upkeep on the part of instructors and course staff. However, it can still be desirable to issue extensions on a case by case basis or more generally only as students need them. The autoextension policy of CS61B Spring 2023 attempted to find a balance between these two lines of thought. The policy reads:

"There are two types of extensions:

- Short extension: Extensions submitted for less than 24 hours will be instantly processed by an automatic system,
- Long extension: Extensions submitted for 24-72 hours will be manually processed by GSIs and will receive a reply within 1 business day. Extensions beyond 72 cannot be granted using the Beacon extension system.

After week 5, the course content builds on itself in a less time-critical manner, so starting in week 6, the thresholds for "short" and "long" extensions will be increased to 72 for short extensions and 120 hours for long extensions." [9]

This policy requires students to manually request extensions if they feel they need them but is mostly automated for easy deployment on the part of the course staff. Students could request as many extensions as they wished on all but a few selected assignments. Additionally, the policy changes throughout the semester, becoming more generous later on as assignments become fewer and slightly larger.

There are several motivations for this policy. A primary one is creating a flexible system with a high degree of automation. In past semesters, CS61B has had issues with huge amounts of extension requests which could be alleviated, at least in part, by autoextensions. The policy was also an attempt to avoid the issue of making extension resources act as a "currency", as happened with the slip days system described earlier. Another final was to give students themselves a high degree of flexibility in completing assignments. Extensions do not get students out of doing work, instead they only force them to complete that work later, therefore part of the idea behind the system was that students would try their best to complete their work on time and only take extensions if they needed them.

Results

Assignment	Extens / Subs	% Extens	Avg Short Exten Len in Days	Avg Short + Long Exten Len in Days	Avg Total Exten Len in Days	Avg Sub Lateness in Days	Median Exten Len in Days
HW 0A	64 / 1738	3.7%	0.87	1.54	5.12	2.44	2
HW 0B	440 / 1680	26.2%	0.92	1.03	1.78	1.40	1
Lab 1	91 / 1795	5.1%	0.92	1.28	5.65	2.60	1
Lab 2	299 / 1685	17.7%	0.95	1.29	2.88	1.64	1
Proj 0	660 / 1616	40.8%	1.00	1.40	8.59	1.37	1
Lab 3	416 / 1597	26.0%	1.00	1.13	2.17	1.50	1
Proj 1A	628 / 1631	38.5%	1.00	1.06	1.87	1.82	1
Proj 1B	628 / 1602	39.2%	1.00	1.06	1.46	2.08	1
Lab 5	345 / 1540	22.4%	1.00	1.04	1.97	1.62	1
Proj 1C	611 / 1552	39.4%	1.00	1.04	1.67	1.75	1
HW 2	592 / 1522	38.9%	1.96	1.97	2.57	2.24	2
Lab 7	541 / 1520	35.6%	2.18	2.20	2.76	2.35	3
Proj 2A	892 / 1581	56.4%	1.58	2.71	3.33	3.26	3
Lab 8	833 / 1464	56.9%	1.55	2.56	3.05	2.84	3
Proj 2B	878 / 1581*	55.5%	1.66	2.29	6.96	6.28	8
Lab 12	455 / 1421	32.0%	2.58	2.56	2.87	2.81	3
Proj 3.1	760 / 1581*	48.1%	2.79	2.84	3.01	3.20	3
Lab 13	548 / 1458	37.6%	2.65	2.66	2.83	2.72	3

Table 1.4: Submission patterns of CS61B Spring 2023 students. Assignments that involve little or no code, as well as those ineligible for extensions have been omitted. Unhighlighted rows show the first part of the semester with shorter extensions, highlighted rows show the second part of the semester with longer short extensions. All assignments are listed chronologically by due date.
$*$ imputed values

10617 unique extensions were submitted over the course of the semester, with the average student receiving 6.64 extensions of some length. The system was extremely widely used. Of approximately 1600 students, 1549 unique students took at least one extension, meaning this system was used at a higher rate and more widely than others cited in this report. The average number of extensions received by a student who received at least one was 6.85. For a number of assignments, including Project 2A, Lab8, and Project2B, *most* students took an extension of some length.

The lengths of granted extensions are also notable. Unsurprisingly, once the course switched from defining a short autoextension as one day to three, the average length of extensions dras-

tically increased. However, there is only one assignment in the second half of the class where the median request was two days; all the other medians, with the outlier of Project2B, are three days long. This implies that students quickly got in the habit of simply requesting the maximum extension length and taking that amount of time regardless of their own estimates of how long they might need.

Implementation and Staff Results

The system turned out to be more work on the staff than was originally intended. First of all, it made the logistics of coordinating assignment release times and due dates trickier. Because so many students would take extensions, the staff had to be weary of deadlines becoming stacked between the original deadline of the assignment and the late autoextension deadline. The blurring of deadlines also placed a burden on the staff to be prepared to help with recently released assignments as well as older assignments which had already come due. At several points in the semester, there were so many students with extensions that the staff would deliberately overstaff office hours that took place in the days *immediately after* the due date in anticipation of high demand. The policy also sparked pedagogical debate over whether or not a student who wanted help with an already due assignment should receive the same level of attention as one who was working on a not yet due assignment.

Student Perception

Thanks to the official course policy as quoted above, it was always public knowledge to students that short extensions would be automatically granted. As the semester went on and more and more students used and became familiar with the extension system, their perception of deadlines in CS61B changed. Students stopped perceiving the actual deadline of assignments as fixed and began to think of the late deadline as the "real" one. As evidence of this, the busiest and highest demand office hours always happened *after* the assignment deadline. This system presents a contrast with the previously noted slip day and homework drop systems. Those systems allotted students a finite resource which came to be thought of as a currency, which students attempted to "spend" wisely. Because extensions in CS61B were allowed on virtually every assignment however, there was no incentive to be conservative with them, and many students began requesting extensions on every assignment. In the above table, a larger and larger percentage of students requested extensions as the semester went on. Of the 1549 unique students who requested extensions, 919 (59.3 %) of them requested more than 10; 452 (29.2%) requested more than 10.

Additionally, students also became frustrated with some of the limitations of the extension system, especially early in the semester when only one day extensions were automatically granted. Because most CS61B deadlines fell on a Friday, that meant that everyone in the class could get an extension until Saturday with no questions asked. This was useful to students who simply did not have time during the week to do an assignment, however it did little to help many who were most behind in the class. Because there was no synchronous staff support available

over the weekend, those who relied on that synchronous help often made little progress even with the extra day. When the three day autoextensions began six weeks into the semester, it became common for students to flood office hours Monday for assignments due Friday as this was the only day of the extension period wherein they could get synchronous help with assignments.

CS61B Fall 2020 - Amnesty

Description and Motivation

One of the more eclectic approaches to deal with late work can be seen in Fall 2020 CS61B's amnesty policy. The policy reads:

"In addition to slip days, during RRR week a week long reading period immediately before finals week, there will also be an amnesty period where students may resubmit up to two assignments without penalty ... We will not grant amnesty for more than two assignments. "

This means that at the end of the semester, all students were allowed to resubmit two assignments for a potentially better score. In addition, since this policy was made known at the beginning of the semester, students could complete the assignment they wished to claim amnesty on at any point, making the system equivalent to allowing unlimited (at least until the end of the semester) slip days on two particular assignments chosen by the students. That said, the actual time of submission was limited to a one week window at the end of the semester, leading to that week being perceived as a "new deadline". It is also worth noting that although this semester featured four distinct projects, those projects were further broken down into subparts, amnesty being applied per subpart of a project. This means that if a student claimed amnesty on project 2A and project 2B, that would count as two amnesty submissions. This was used in conjunction with a more conventional slip day system.

The rationale behind this system was largely built on the notion that although extensions can help students who are behind in the course by giving them more time to complete assignments, they can also be harmful because extending an old assignment can get in the way of the current assignments and overload students already behind in the class. Additionally, this policy is meant to aid students who improve over time. Perhaps an assignment which proved to be too much for them at the start of the semester would be more manageable by the end of it.

Results

Projects:

Assignment	Amnesty / Total Submissions	% Amnesty Submissions
Project 0	7 / 1168	0.6%
Project 1A	32 / 1110	2.9%
Project 1B	17 / 1058	1.6%
Project 2A	164 / 1044	15.7%
Project 2B	244 / 1012	24.1%
Project 2C	57 / 989	5.8%
Project 2D	155 / 930	16.7%
Project 3A	44 / 1018	4.3%
Project 3B*	0 / 1002	0.0%

Extra Credit:

Assignment	Amnesty / Total Submissions	% Amnesty Submissions
Project 0 Gold	89 / 192	46.4%
Project 1 Gold	125 / 516	24.2%
Project 2D Gold	144 / 391	36.8%

Table 1.5: Number and ratio of assignments used on projects as amnesty submissions over the semester. [10]
∗ Project not eligible for amnesty

1200 amnesty requests were made in total by the end of the semester. Around that time, there were about 1000 students in the class. This means that the average student used 1.2 of their 2 allotted amnesty requests, indicating that though the policy was widely used, it was not utilized quite to the degree of the slip day or homework drop systems outlined above However, of the 1200 requests, 663 of them were made by unique students. This indicates that nearly all students who used at least one amnesty submission used both their submissions and at about 66% of the student population took advantage of the policy.

Whether or not this high usage is a good thing is largely determined by the pedagogical perspective of the course staff. It is quite likely that, for whatever reason, students in a class will not be able to complete a given assignment. If the course staff views it as valuable to complete all the assignments in a class, even if that completion comes at a later date, then the amnesty policy appears to be quite successful. Additionally, the amnesty framework attempts to address the domino effect outlined above. Students are given a huge amount of time to make up assignments, so it matters less if they have other assignments to do immediately after the deadline

of the amnestied assignment. The amnesty policy leaves some room for students to choose a schedule that works for them without taking the approach of total autonomy.

That noted, there are other pedagogical perspectives under which this system may be viewed as problematic, in particular, if every course assignment builds on the previous one and there is a "narrative arc" to the class. Under this framework, amnesty is counterproductive in that it encourages students to abandon difficult assignments with the opportunity of completing them later. If the lessons of that particular assignment are fundamental to completing the next one, then it may be better for the student to attempt the first assignment, even if they are ultimately unable to finish it, as otherwise the second assignment will be more difficult for them. Amnesty can result, especially if taken advantage of early in the semester, in a student progressively falling farther and farther behind. Within the EECS Courses specifically, CS164, a compilers course, stands out as an example wherein amnesty could potentially run into this issue. The Fall offerings of this class are structured in an iterative fashion, wherein students effectively continuously build a single compiler over the course of a semester. Missing an assignment here could potentially not only hinder a student's progress, but also prevent them from implementing later parts of the compiler.

An additional perspective on this system can be seen in what kinds of assignments students regularly choose to claim amnesty on. Notice in the charts above that extra credit assignments saw a far larger proportion of amnesty submissions than conventional projects. To make that even more clear, consider the chart below:

Type of Assignment Amnesty Requested for	Submission Numbers	Percentage of Total
Required	892	74.3%
Extra Credit	308	25.7%

Table 1.6: Amnesty submissions used for regular assignments versus those used for extra credit.

Although it is the case that far more requests were submitted for regular assignments than extra credit ones, the fact that over a quarter of amnesty submissions were for extra credit is still worth noting. This indicates that many students did not strictly "need" amnesty submissions in the sense that they failed to complete an assignment earlier in the semester, but that many who had already completed all the assignments took advantage of amnesty to get extra credit. In other words, those who were already ahead in the class were able to further their status as top students by using amnesty to get even more ahead, while those who were behind were merely afforded the opportunity to catch up. Depending on how much extra credit is available within a given class, and if the class is curved or not, this means that amnesty can actually result in a system that rewards those ahead in a class without offering much material gain to those who are struggling.

Another, and perhaps more concerning feature of this system is how it encourages students to spend time doing extra work which does not directly impact their grade. In the particular

semester of CS61B in which amnesty was deployed, projects were worth 33.4% of the total grade, while extra credit was worth only up to 2.27% of the total grade. This means that extra credit is quite unlikely to make any difference in a student's grade and that the main projects are vastly more important, and yet over a quarter of the amnesty submissions are made in attempt to get at this 2.27% rather than the 33.4%. This shows that students feel obligated to do extra credit assignments, even if there is little material benefit, simply because if they do not they are missing out on "free" points. Referring once more to the notion of pedagogical goals, it is clear that extra credit often becomes seemingly mandatory for students, meaning more work on students who can afford to do it and more stress on those who have to forgo the extra credit to use their amnesty on other assignments.

Implementation and Staff Results

Unlike some of the previously mentioned systems, amnesty can be fairly complex to implement and thus the additional workload it places on the course staff is non-trivial. Notably, there must be a mechanism for allowing students to resubmit, but only if they are using one of their two amnesty submissions. Even in CS61B, where the ability to delineate these sorts of assignments is mostly automated, this turned out to be a large amount of work. In the absolute worst case, this would require the course staff combing through submissions and picking out ones that individual students had selected to be tagged for amnesty.

Additionally, although part of the goal of amnesty was to allow students to come back to assignments after their programming skills had improved, a large proportion of students still sought assistance from the course staff on online forums and in office hours. In a typical class, course staff prepare for assignments as they are released to students, meaning a member of course staff might prepare, for example, homework 3, days before it would be released to students and wait to prepare homework 4 until it too was near release. This means that a course staff only has to be prepared to answer questions on a small subset of assignments at any given time. Amnesty breaks this assumption by encouraging students to ask questions on assignments due in months past. Within CS61B, many members of course staff found amnesty a trying policy not because of the difficulty of its backend implementation, but because of the difficulty in preparing to respond to student questions on, in effect, any assignment from the whole semester.

Student Perception

Another important takeaway is how students themselves viewed the amnesty policy, regardless of whether or not it helped them numerically. Courses at scale have a huge amount of inertia to overcome when implementing policy, meaning that the more complex a policy, the more likely it is that individuals will ignore it or otherwise misunderstand it.

The general consensus was that the amnesty policy was complex not only from the staff perspective but from the student perspective as well. In a survey conducted at the end of the semester, many students noted that the amnesty policy was unclear to them or that they did

not even know about it until late in the year. This is especially problematic as a large part of the appeal of the amnesty system is that students can take advantage of it to leave assignments incomplete and come back to them later. If a student is not aware of the policy until the end of the semester, they may spend time attempting an assignment they ultimately may have just postponed instead.

Another issue was the exact timing of the policy. Though technically a student was able to complete the amnesty assignments at any time, overwhelmingly they left them until the end of the semester, at a time when they had finals and other work to worry about. Recall that one of the goals of amnesty was to allow students to come back to assignments when they had more time, but if that time happens to be finals week, it can result in even more interruption and distraction from ostensibly more important assignments. A possible variant of amnesty could perhaps allow for submission at any time for the rest of the semester, instead of just one week at the end, though it remains unclear if students would be any more motivated to complete assignments early in this case.

1.2 Office Hours

Another important element of any class, especially one that involves a great deal of mathematics as well as logical thinking, is office hours (OH). In a conventional course, an instructor will typically hold two or so OH a week. Occasionally, graduate students working for a particular course may also have OH at a set time or by appointment. It is the experience in the EECS Department that demand for office hours increases more quickly with course size than any other component of EECS Courses, necessitating a large number of highly organized OH. Though there are some obvious reasons for this, such as more students equating to higher demand, there are also more subtle reasons. Increased accessibility to courses creates a more diverse base of students, one which likely has less exposure to CS material than would otherwise be the case, and thus may require more OH. This section of the report will detail some basic considerations for running OH of courses at scale, as well as some of the novel techniques developed by EECS Courses in dealing with this increased load.

Considerations

Room Size

A fundamental yet deceptively easy to overlook aspect of OH is that it should be hosted in a room large enough to accommodate all those who wish entry. In a conventionally scaled course OH are typically hosted in, as the name implies, an office. Unfortunately, this arrangement is untenable for courses which may expect dozens, and potentially even hundreds, of students to seek help simultaneously. Getting a large room for OH is often easier said than done and may require intervention at a university level or otherwise be beyond the scope of an instructor to request, however it is always best for courses to try to maximize their OH floorspace when possible.

Not having a large enough room creates a number of issues. Perhaps most obviously, shoving too many people into a single room can violate building codes and even amount to a health hazard. It can also make the room very hot and uncomfortable, or even facilitate the spread of contagious disease, an increasingly large concern in the post-Pandemic world. Additionally, it can present pedagogical issues for a course. It is quite likely that a student who goes to OH only to find the room completely full will simply leave, either because they are uncomfortable being in a packed room, or because looking at the sheer number of students in attendance leads them to assume that they will not get helped in the first place.

To address these issues, many EECS Courses have come up with ways of freeing up allocated rooms. For example, CS61B converts many of its allocated lab rooms into OH rooms, going so far as to cancel unpopular lab sections entirely towards this end. CS70, which often hosts OH in a range of small to large rooms, encourages students to come to specific OH hosted in the larger ones. Another solution is the use of online OH (discussed in more depth later in this section), which has an unlimited capacity.

Availability

In a conventionally scaled course there are a very limited number of OH a week, and if a student cannot make any of them, they are typically told to email an instructor to set up a separate time to attend. When there are hundreds of students in a course, this system not only becomes impractical, but also introduces equity issues by allowing a student to secure more OH simply by asking. Therefore, another important consideration of OH for courses at scale is availability, or rather, ensuring that there are enough OH times available throughout the week that effectively any student with any schedule can attend at least *some* of them. However, this comes with a trade off. Generally speaking, more hours means fewer staff members per hour, which results in students often waiting longer for help.

Online OH

Though largely a Pandemic Era innovation, online OH has become a crucial part of EECS Courses even after the return to in person classes. Hosting OH on an online platform like Zoom can be convenient, and even preferable, for many students and staff. Additionally, some students with very particular schedules or disabilities may *only* be able to attend OH virtually.

As they are conducted through distinct mediums, online and in person OH are fundamentally different and thus require special considerations. For instance, due to the convenience of online OH, many students attend even when they otherwise would not, ensuring that online OH are consistently more crowded than in person OH. In CS61B Spring 2023, the average online OH saw 21.1 tickets per hour while the average in-person OH only saw 14.1. [8] Additionally, though some students are comfortable asking each other for help in an in person setting, this comfort often vanishes in a virtual environment. It is a challenge to get students to work together or ask each other for help in online OH. This means that although students often work together and collaborate while waiting for staff help in an in person setting, they seldom do so in an online setting, which results in many students feeling that all they can do in online OH is wait for their name to get called.

An additional note is that while it can be very valuable to offer both online and in person OH, offering both at the same time can be problematic. It is common for students to go to in person OH while simultaneously queuing up online, thus allowing them to double dip on resources and potentially exploit the system.

Scope of Work

The type of work expected of students has large ramifications on optimal OH structure. For example, in problem set heavy and mathy courses, students will often be stuck on the exact same logical problem or issue, allowing staff to effectively group students together and help several at once. In contrast, primarily coding based courses often see students stuck on bugs or issues entirely unique to them, forcing staff to give more individualized help.

OH Usage by Students

Finally, although OH is a resource available to all students, there is a vast disparity in who actually uses them most often. For example in CS61B Spring 2023, though the average student submitted 10.2 tickets throughout the entire semester, only 71% of students submitted a single single ticket at all, making the average number of tickets submitted by people who submitted at least one 14.4. Additionally, some students visit office hours with extreme frequency and use many tickets as is shown in the plot below.

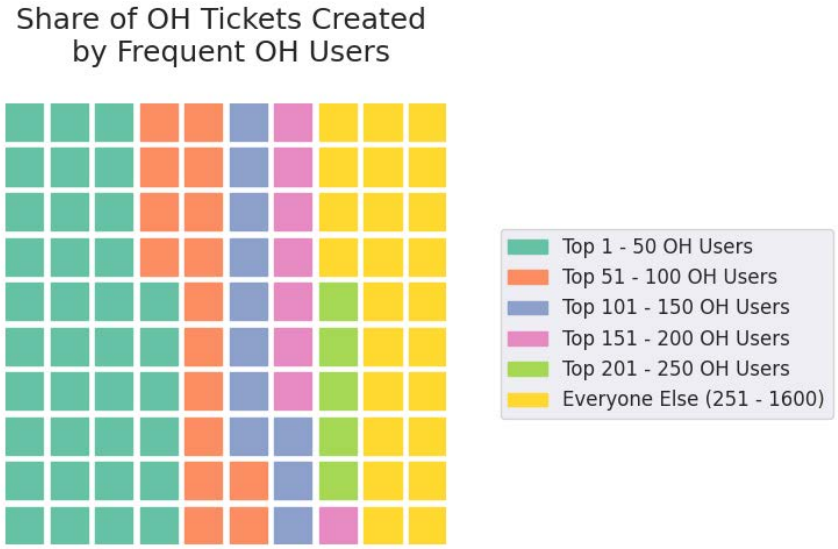


Figure 1.3: Proportions of OH Tickets made by students. [8]

Over 50% of all tickets made in OH were made by the top 100 most frequent users of OH, who collectively represent only 6.25% of the class. The top 10 most frequent users of OH alone 9.6% of the tickets. This shows that the majority of OH resources are used by a minority of students.

Informal Mechanism for Running Courses at Scale

Online Queues

With dozens or even hundreds of students pouring into OH hour after hour, it is crucial to establish a queue system to organize and triage who needs help. Virtually all EECS Courses employ online systems which allow students to queue up and be helped in the order they arrive at OH. Only the very smallest classes (for example CS164, which often sees only 2-6 students in OH at a given time) forgo an online system. It is also worth noting that those classes that forgo OH queues are often upper division courses wherein students are expected to be more independent in the first place.

Online queues have additional benefits beyond simple organization. They can also allow the staff to collect more information about the student before helping them. These queues typically have a description section where students can detail the issue they are encountering, allowing staff to pre-diagnose their problem and perhaps even send a staff member who best understands the issue at hand. Additionally, it allows staff to see if a particular issue or question is coming up very frequently among students and prioritize helping that specific issue. Historically, some courses such as CS61B and CS61C have even refused to help students until they add a descriptive note to their help ticket.

Another issue with OH for courses at scale is that due to their popularity, there might not be enough time to help every single person in every single office hour. It is not uncommon in EECS Courses for OH to end with several dozen help tickets unresolved, especially in the most popular lower division classes. Online OH queues can be closed at any time to address this issue. If the staff realizes that OH is ending soon and that there are already too many people on the queue to help everyone, they can simply stop accepting new tickets. Although this is a bandaid for the broader issue of suboptimal staffing, it at least prevents many students from being given false hope that they will be helped when the course staff is facing a deluge of other tickets.

A final perspective to note on online queues is that they allow for easy data collection. This enables data visualizations and analysis which can be used to determine trends and improve the system at a later date. Almost all the data presented in this section of this report was collected from online queues.

However, there are downsides to online queues, many of which come in the form of exploitation on the part of students. For example, if the queue is joinable online, students can join even if they are nowhere near the physical OH room. In times when the queue gets very long, such as around big project deadlines, it is common for students to first queue up at home and *only thereafter* physically depart for OH, the expectation being that by the time they arrive at in person OH, they will be higher up on the queue but still unhelped. One approach taken by some courses to fix this loophole is to require a password to join the queue which is only available in person, but this adds an extra barrier on students seeking help and does not prevent people in OH from distributing the password to others. Additionally, this fails to address the reality that this behavior is even more pronounced in online OH, where it is common for a student to put themselves on the queue while at home and then go about their day doing other things while just waiting for their ticket to be taken.

A related issue is students often use the queue as a revolving door, meaning that the second they get help, they put themselves right back on it even if they do not have a question in mind, their expectation being that by the time help comes back around to them they will have something else to ask about. This can create an over reliance on staff help and is ultimately detrimental to the students' learning. The revolving door problem can become particularly noticeable in group projects or in any class that allows student collaboration. Students in a group can stagger their placement on the queue by deliberately waiting for an opportune moment in order to ensure that by the time one staff member is done helping the group, there will only be a very small wait until another member of the group gets their ticket called and the group gets

helped again.

Other downsides of online queues are not so much a result of deliberate exploitation, but simply the fact that they give rise to certain emergent strategies which maximize the amount of staff help received. The most notable of these strategies, as well as the most perceptible from the student perspective, is the race for the top spot. When a particular OH time is popular and a lot of students want help, it is common for them to wait until the exact moment the queue is opened and then add themselves as quickly as possible so that they will be near the top of the queue. If there are dozens of people waiting, this can cause the queue to feel random and arbitrary.

Question Grouping

Often, especially in courses at scale, many students find themselves grappling with common questions. Because of this, staff often attempt to group students together who are working on similar issues in the hope that they will talk and be able to solve some of their problems together, thus reducing OH demand on staff and creating a more dynamic work environment. There exist a number of ways to do this. In an in person setting, students can be told to occupy a certain area of the room or a certain table if they are working on a particular thing. The staff can also enforce the grouping policy in an ad hoc fashion by directing students they help to talk with others who have similar issues.

However, the results of grouping have been mixed and seem to depend largely on the course structure itself. First of all, there is a general expectation that when students go to OH that they will be helped by staff, not by other students. There is also a general perception (understandably) that the staff knows more about assignments than other students do. This leads to students not wanting to work with each other because they do not want to deal with the social overhead of talking to others, or simply because they believe that other students would not be particularly helpful in solving their issue. Effectiveness is also dependent on the structure of course assignments. In programming heavy courses, grouping tends to be less effective as students often come to ask for help on specific bugs or issues with their implementation which are unique to them and would likely be met only with stares of confusion from other students. Also, as most classes have strict policies on sharing code with other students to avoid plagiarism, some are hesitant to help others for fear of being accused of cheating at a later date. This is contrasted by problem set based classes however, wherein grouping tends to be far more successful. This is because, unlike with coding, problems in a problem set rarely rely on a specific implementation. In other words, there is usually one optimal way to complete the problem and each problem is discrete and short, so it is easy for a staff member to start from the beginning each time a student or group of students needs help. Part of the downside of this assignment structure however is that it often results in one student getting the answer and then everyone else copying that answer from them with little understanding of what it actually means or how it actually works.

Surplus System and Staffing Hours

In many EECS Courses, in particular programming based courses, demand for OH is uneven throughout the semester or even throughout the week, tending to be much higher around deadlines. For this reason it is important for a course to be conscious of deadlines and staff OH appropriately and dynamically in anticipation of them. In CS61B this is achieved through a surplus OH system. TAs sign up for a certain number of "surplus" OH hours throughout the semester, which are centered around deadlines. Additionally, sometimes TAs will even be asked to forgo working certain OH times that are expected to be low demand in favor of running more surplus slots instead.

The vast majority of OH in CS61B are very full, so this means that in many ways surplus does not allow for *more* students to be helped as much as it changes *when* students are able to get helped. However, this is still important from the perspective of optics. If students see that the queue is extremely long they may become discouraged to seek help in the first place. Surplus OH helps to decrease the size of the queue when OH is at peak demand and makes OH feel more uniform.

OH Head System

Adding more staff into a given office hour is only part of the equation. How long each staff member is spending helping each ticket is also key to maintaining a high OH throughput. If OH is very full it can take a very long time for students to receive any help at all, and thus many students will try and monopolize a staff member for as long as possible once they get one. Additionally, sometimes the issue a student has is very complex and hard to solve quickly.

Time per ticket actually increases when there are more students on the queue, in part for the above reasons. This is particularly suboptimal as it means that the queue tends to run slowest when it is longest. CS61B has attempted to ameliorate this problem by introducing an "OH head" who is in charge of each office hour. This person is responsible for making sure that the appropriate staff shows up in the first place, and also that no staff member is spending too much time on a single ticket. If they are, it is the job of the OH head to approach them and instruct them to wrap things up and move on. This has greatly helped reduce wait times when the queue is longest and also allows for the staff to set a clear expectation with students that they will only help them for a certain amount of time before moving on.

Gitbugs and Asynchronous Help

Students cannot always be expected to go to OH for help and sometimes have a simple clarifying question that does not warrant going all the way to a physical location and waiting on a queue. Many of these questions may not even be particularly pressing or require synchronous help at all. For these sorts of questions, online forums and places for discussion are essential in EECS Courses. Every course will have the ability to post questions online and receive answers from staff. Although private questions are typically allowed, it is generally encouraged to put these

questions in public threads so that other students with similar issues can later look at them for help as well.

CS61B takes the use of online forums even further in an attempt to declutter office hours. If a student has a particularly bad debugging issue, especially one that could likely be solved by a staff member if only they could debug the student code on their own computer, the student will be encouraged to submit a "gitbug". The term gitbug is now a misnomer, as it does not involve git at all (though it once did), but the basic idea is that a student can explain their bug in a premade form which they post online. Later a staff member will be able to pull the student code directly to their computer in an attempt to find the issue and update the student with help asynchronously. This means that during office hours, when time is precious and potentially many students are on the queue, a staff member can tell a student to just "submit a gitbug" which they can look at later when more time is available, allowing the staff member to move on.

Project / Homework Parties and Specialized OH

OH are typically conceived of as a general purpose time for students to ask any questions they wish. These could be logistical questions about the course, information about the personal research of a given professor, programming help, problem set help, etc. However, there can be great value in having time allocated in which the staff will focus on a single assignment or project. To this end, project and homework parties are common within EECS Courses.

These "parties" are typically not very different from regular OH, the main differences being that they are often irregular, typically occurring only prior to large deadlines, that more staff members are allocated to facilitate them, and that the scope of questions is limited to a single assignment. The real difference however, is a psychological one. Simply by advertising parties as something distinct from normal OH, staff are often much more able to get students to work with each other than they otherwise would. This is perhaps in part because everyone working on the same assignment in the party can create a powerful sense of "in it togetherness" that helps students bond and overcome the social barriers of working together. The course staff can also take advantage of this increased willingness among students to work together. CS61B often runs presentations on particular assignments or sections of assignments that students struggle with, the intention being that anyone who is interested can watch the presentation and anyone who is not can continue working as normal. These presentations are very popular, often allowing the staff to give help to dozens of students at a time.

"Conceptual only" OH are another variety of office hours occasionally offered by programming classes. The idea is to have OH centered specifically around discussing only big picture or exam questions instead of spending time on debugging or programming. Classes which have attempted this, such as CS61B, have had mixed success with the concept. It is quite common for students to still come to conceptual OH with debugging questions (as this is far and away the most common issue students attend OH for in the first place), only to end up frustrated when they are told that the staff will not help them with their code. Additionally, the popularity of conceptual OH has always been low, creating a perception that TAs would be off just running regular OH. Moreover, it can be difficult to define exactly what a "conceptual question" is. Is it

a question that doesn't involve code at all? Can it be a question about code or a programming language so long it is not specific to a course assignment? Can it be a high level question on a course assignment? The answer to any of these questions will vary based on staff opinion, course content, and course structure. All that noted, it can be a valuable resource for students, and if the course staff feels that students need more practice with the conceptual rather than programming side of a course, it may be useful to encourage students to make use of conceptual OH.

Finally, OH can be specifically tailored not to the assignment, but to the students. In Fall 2019 some EECS16A OH were hosted by the Association of Women in EE&CS (AWE). These OH were not limited to only helping women, however they were run by a women's organization and helped to provide representation within the field of EECS.

1.3 Exams and Proctoring

Exams offer a key insight into academic performance and are often the element of a course which contributes most to a student's final grade. In many traditional technical courses, exams are conducted in the very same room where lecture is normally taught in place of a day's class. In other words, "in class" midterms are the norm. Courses at scale present a series of unique challenges in this domain which must be solved in order to run and proctor exams. This section of the report will first take a look at some of the common strategies and considerations of courses at scale to facilitate exams, and then look at a few case studies of how particular EECS Courses at UC Berkeley have employed varying approaches in conducting their own exams.

Considerations

Room and Time Logistics

One of the most straightforward questions a student could ask about an exam is simply: "when and where will it be?". The answer to this question for many courses at scale however, is not so straightforward.

The first complication is that of room size. As noted in previous sections, large EECS Courses at UC Berkeley routinely discourage students from attending live lectures because the rooms in which they are taught are typically too small to hold the entire class at once. This means that come exam time, additional rooms will be needed in order to facilitate the exam for everyone. The largest EECS Courses may host exams in as many as a dozen rooms spread across the UC Berkeley campus. This means that the staff is responsible for partitioning the class into sections (often by student ID or name) and assigning those sections to take the exam in different rooms. It also requires that the course staff be large and robust enough to amply distribute itself across all these rooms to properly proctor.

Another complication is that of time. Because large EECS Courses typically do not require attendance due to a physical lack of space, it is the norm for students to take other courses taught at the same time as their EECS Courses. This means that in addition to EECS Courses needing to organize multiple rooms to take exams in, they must also take into consideration the fact that many students will not be available to take the exam during the lecture time due to a conflict with another class. In part because of this, it has become common for many EECS Courses to host exams in multiple rooms across campus at night (usually from 7pm - 9pm or 8pm - 10pm). This gives the two-fold benefit of minimizing student time conflicts as few have classes at night as well as making room logistics easier as UC Berkeley itself typically has more open rooms to lend to classes at night than it does in the middle of the day.

However, despite all attempts to minimize conflicts, the reality is that in a 1000+ person class, there will always be a number of students who simply cannot make the normal exam time for any number of reasons. To deal with this the vast majority of EECS Courses offer alternate exams. These could be earlier in the day, later in the evening, or even in the days following when

the majority take the exam. Alternates are often handled on a case by case basis by the course staff.

As a final note, many courses have aimed in the past to begin exams synchronously across all rooms. This means that the course staff communicates on exactly when everyone is to begin, or simply waits for the clock to strike an exact moment before starting. However, there are a number of issues with this approach which have led to many EECS Courses abandoning the process. One is simply the fact that starting synchronously amongst all rooms means that any one room could hold the others up. In particular, bigger rooms will likely take longer to prepare and distribute exams to students and leave smaller rooms awkwardly waiting. In Spring 2018, such a circumstance resulted in a CS70 exam scheduled for 8pm - 10pm to actually be held from 8:30pm - 10:30pm. An additional reason for not trying to sync rooms is simply the fact that so long as all the rooms start more or less at the same time, there is no way for students to pass information between rooms to other students.

Exam Corrections

No matter how hard a professor or course staff may try, it is nearly impossible to create an exam containing absolutely no errors or ambiguities. Additionally, with over a thousand students taking such an exam, it is likely that at least a few of them will ask for clarifications. In a traditional exam setting where all students were in one room, clarifications can simply be announced to everyone at once, however in a distributed environment, there arise potential concerns regarding fairness. One room should not receive a clarification that another does not, and multiple rooms should receive clarifications simultaneously. For this reason techniques have emerged amongst EECS Courses to deal with exam clarifications.

Most EECS Courses use a common shared document which is projected in all rooms at once and added should clarifications arise. This document is broadcast to all rooms simultaneously, and if a room lacks a projector or large screen, the contents of the document are copied by-hand onto a whiteboard or some other implement students can see. Exam proctors must be careful about what information they give to clarification seeking students. In CS61B, proctors are told to respond to any student question with one of only two answers: "No clarification is needed" or "Let me check and if we decide to clarify, we'll project a clarification within 10 minutes." [7]. This ensures that proctors always have time to consult other staff members before clarifying (unless of course it is clear that no clarification is needed) and never give more information to one student than is given to any other.

There are still potential issues here. Students in a room which starts later than another will get access to clarifications with more time remaining in the exam than those who started earlier. This becomes even more extreme in the case of alternate exams, wherein students taking the exam before the majority would have no clarifications to work from and those who take it later would have all clarifications available right from the start. Some potential ideas have been floated to address this, for example not projecting clarifications to all rooms immediately, but instead waiting until each room is at the same point in the exam to clarify (i.e. if one room finds a clarification 35 minutes into the exam, another room that started later will not project

that clarification until 35 minutes after starting its own exam), but this fails to solve the issue of students taking exams earlier than others. Additionally, clarifications rarely make a substantial difference in performance on the exam itself so attempts to fix this issue have largely been deemed as simply not worth the effort.

Resource Equity

It is important to consider what tools students are allowed to use during an exam. For example, many EECS Courses, in particular those with online exams, distribute PDF copies of their exam and allow students to use iPads to write directly on the PDF which they later submit. This can cause equity issues, in particular if there are figures which need copying or annotation in the exam. In Spring 2023, a notable controversy around this arose in CS184/284a, when students were asked the following question:

"Consider the following mesh. Draw in one level of Loop subdivision. Note: Please make sure your submission is clear and readable. A ruler might be useful."

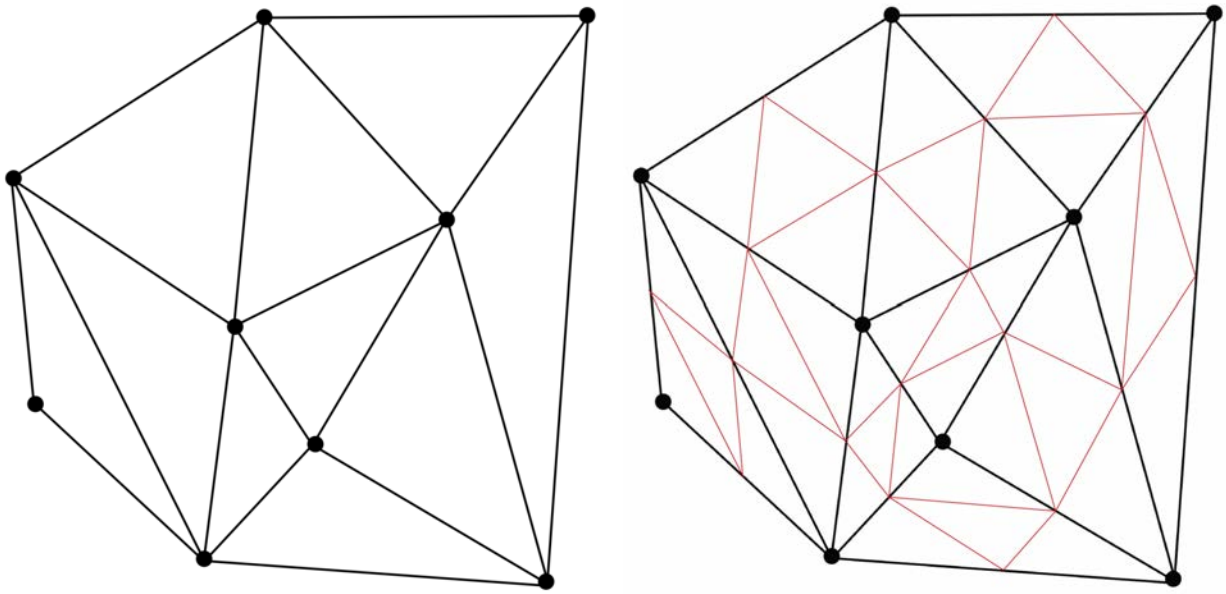


Figure 1.4: The left image is what was provided to the students, the right image is the intended solution. [4]

Because this question involved a large degree of particularly tricky by hand copying, many students complained that those with iPads were at a distinct advantage over those without. Some also complained that students who had access to a printer were too at an advantage. The

course addressed this issue on the next exam by requiring every student to put their answers in a template which was released in advance and to be printed out. This put extra work on those with iPads, but sought to standardize the exam taking process regardless of a student's access to additional technology.

Keeping the Secret

Another consideration is that, especially given alternate exam times, there is potential for a student who took the exam to leak information to other students who are taking it later. For this reason it is preferred that alternates happen *after* the main exam, minimizing the potential number of people who could be leaked to. Additionally, the most natural way to prevent leaks is by temporarily disabling student channels of communication during exam periods. Course forums are locked and course discords are shut down. This however cannot fully solve the problem as there may be private channels that the course staff does not know about through which students can communicate, as well as some public ones (such as reddit) where the staff's ability to limit the spread of midterm information is potentially limited. That noted, it is important to build in the expectation that students will not talk about the exam until everyone has completed it. During the exam itself as well as in the days leading up to it, this is always made clear to students. This has helped to reduce leaks, and it is even common to see students make revealing posts on reddit, only to then delete them after other students remind them not to share exam information.

In general though the best way to limit the potential leaking of exam information is simply to make sure that as many people take it synchronously and in person as possible. Sometimes of course this is unachievable, as was the case during the Pandemic semesters where all exams had to be conducted online.

Case Studies

CS170 Fall 2020 - Online Proctored Exams

The proctoring policy of CS170 in Fall 2020 is emblematic of many of the Pandemic era approaches taken to exams. This was a complete proctoring policy, meaning that the course attempted to proctor in a hands-on way comparable to in an in person exam. The exam was also closed book, so additional measures had to be taken to ensure that students would have difficulty accessing outside and online resources. Students were required to follow strict set up guidelines and sent a specific Zoom link by the course staff which would record them taking the exam. Additionally, though the exam was to be taken on a computer, the student had to record their physical workspace as well. There were detailed restrictions on what tabs or resources students could have open at any given time. Although it was impossible for staff to constantly observe hundreds of students all in different Zoom links constantly, they would occasionally enter the Zoom rooms to make sure that the procedure was being followed. [1]

The system was not so much designed to actually make cheating impossible as it was intended to put a "fear of God" into students. Many of the proctoring policies were in reality toothless or otherwise unenforceable. Their purpose was not to actively make cheating impossible, but to make students fear punishment if they broke the rules. The reality is that there were many ways to get around the proctoring policies. Although the exam was closed book and the student workspace was recorded, they could simply have additional resources out of view of their recording cameras. Additionally, although there was always the potential for course staff to drop in, students would only actually be proctored by a staff member for a minute or so there and therefore there was little chance they would be caught if they were breaking the rules at some point. Perhaps the greatest evidence that the proctoring policy was largely symbolic was that the recordings taken of each student for the duration of the exam were never actually viewed by course staff. There were simply too many of them, they were too long, and it would be too hard to notice cheating in them in the first place.

As classes have largely returned to the expectation of in person attendance in the post-Pandemic era, EECS Courses have overwhelmingly switched back to default in person exams because of the many of the issues posed above.

CS61B Spring 2023 - Distributed Remote Proctoring

Although in person exams have again become the norm in EECS Courses of the post-Pandemic era, there are still a number of reasons why students may wish to opt for a remote exam. Some courses, such as CS61B in Spring 2023, have run exams in a hybrid fashion to allow this practice to occur. The in person version of the exam still happens as normal and largely follows the procedures defined above, while the online version takes a different approach.

The online exams were run through a model of distributed proctoring. This means that instead of students being in an individual Zoom link and recording themselves as was the model in CS170, instead they join a common Zoom room where they cannot see each other or anyone else. A staff member will also be in the room who can see everyone. This staff member will be able to observe and proctor several students at the same time, making it actually possible to proctor everyone, unlike in the CS170 scheme described above. [12]

Additionally this technique was extended to an in person setting as well. It is common for students who have disabilities or other exam accommodations as specified by the university to take the exam in several small rooms with few people per room. This poses a logistical issue of needing a large number of staff members to proctor relatively few students. The distributed model was extended to fix this issue. Instead of allocating at least one staff member per room to observe students, a laptop was simply set up in each room with a camera facing the students and with a Zoom call running. A staff member was able to then view all small rooms through this Zoom call and proctor many at once. This system has proved effective and greatly reduced the amount of staff needed to actually proctor an exam.

CS184/284a Spring 2023 - No Proctoring

As should be clear from the above case studies, a great deal of thought and work goes into online proctoring and in the end it is still impossible to uniformly enforce many policies which would be straightforward in an in person setting. The solution that CS184 took towards this was to instead embrace the limitations of online exams by not proctoring at all.

This meant that, unlike in most EECS Courses, CS184/284a made their exam open book. All internet resources, even Learned Language Models such as chatGPT and code compiling IDE's were fair game. This meant that most of the student behavior that CS61B and CS170 sought to eliminate (students googling answers mid-exam, using code to solve problems, etc.) was no longer relevant. That said, collaboration between students was still disallowed. [6] Although the idea of allowing an open book CS exam may at first seem strange, the reality is that many EECS Courses have discovered that students are rarely able to achieve substantially better scores when online resources are allowed. [13]

Additionally, instead of attempting a synchronous exam as the previous two approaches did, CS184 instead released the exam online for one day as an assignment which, once begun, would conclude after a set amount of time. This allowed students to take the exam at any time during the day with the hope that this would minimize the need to reschedule due to time conflicts. In order to make this work, the staff did not accept correction or clarification requests of any kind during the exam, lest the aforementioned issue occur where people taking the exam at different points throughout the day would have access to more or fewer clarifications.

A primary issue with this system is that it does effectively nothing to prevent students from communicating and collaborating on the exam. What is even worse is that one student could potentially start the exam early and then distribute it to other people who would then have the rest of the day to complete it. In CS170 there was the potential for students to communicate with each other, but the strict proctoring policies intended to put the "fear of God" into students largely stopped this from happening. In CS184/284a there was no such notion and many students more quickly recognized the potential to help each other on the exam.

Bibliography

[1] CS170. *CS 170 Zoom Proctoring Policy (External)*. 2020. URL: https://docs.google.com/document/d/14W7ge0tEgwRbhoP8pmXN3cHeebXkVc8DjPqVKiQuEXw/edit#.

[2] CS170. *CS170: Algorithms*. 2020. URL: https://cs170.org/syllabus/.

[3] CS170. *CS170: Efficient Algorithms and Intractable Problems*. 2023. URL: https://cs170.org/.

[4] CS184/284a. *CS 184/284A Spring 2023 Exam 1 Solutions*. 2023. URL: https://drive.google.com/drive/u/1/folders/15ocnyUs3fJ3qD_yavbnCCQLaBB32MSj4.

[5] CS184/284a. *CS184/284a: Computer Graphics*. 2022. URL: https://cs184.eecs.berkeley.edu/sp22/policies.

[6] CS184/284a. *Exam 1 Logistics + Past Exams*. 2023. URL: https://drive.google.com/drive/u/1/folders/1bo0zr18c_QucK8ygezmvY_VOalCU7Kll.

[7] CS61B. *61B MT1 Proctoring Script*. 2023. URL: https://docs.google.com/document/d/1oucWeYJEUoRxJeV2BRVD-2LD_7y7fBHTn9EkKGIhXaQ/edit#heading=h.myfoy3uwj7o8.

[8] CS61B. *CS61B Simple Office Hours Queue*. 2023. URL: https://oh.datastructur.es/.

[9] CS61B. *CS61B: Data Structures*. 2020. URL: http://fa20.datastructur.es/about.html.

[10] CS61B. *CS61B: Data Structures*. 2020. URL: http://fa20.datastructur.es/about.html.

[11] CS61B. *CS61B: Data Structures*. 2021. URL: https://inst.eecs.berkeley.edu/~cs61b/fa21/staff.html.

[12] CS61B. *Remote Exam Proctoring (MT2) Guidelines [SP23]*. 2023. URL: https://docs.google.com/document/d/161rV2gAH1CvixazUTlLvsprOvHUr2RkVLjFL_2yZEZA/.

[13] Claudia J. Ferrante Steve G. Green and Kurt A. Heppard. *Using Open-Book Exams to Enhance Student Learning, Performance, and Motivation*. 2016. URL: https://uncw.edu/jet/articles/vol16_1/green.html.