Satori: Open-source Course Management System
(Technical Paper)

The Social-Technical Aspects of Queues in Office Hours
(STS Paper)

A Thesis Prospectus Submitted to the

Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements of the Degree
Bachelor of Science, School of Engineering

Winston Liu
Fall, 2019

Technical Project Team Members
Disha Jain
Andrew Lewis
Winston Liu
Austin Sullivan

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signed: Winston Liu

Approved: S. Travis Elliott Date: 12/5/19
S. Travis Elliott, Department of Engineering and Society

Approved: Aaron Bloomfield Date: 11-26-19
Aaron Bloomfield, Department of Computer Science

Introduction

The University of Virginia requires that "all Instructors teaching courses for academic credit...are expected to maintain scheduled office hours that are sufficient to accommodate students who want consultation" (PROV-008), a policy similar to other institutions across the world. Yet when faced with rising class enrollment, sometimes exceeding 1,200 students in a single course (NBCNews), it becomes increasingly infeasible for instructors to hold enough office hours to satisfy students' needs. In courses with significant "hands-on" assignments, as is typical of Computer Science courses, instructors have turned to hiring significant numbers of undergraduate students as undergraduate teaching assistants (TAs) who hold their own office hours. Even with enough TAs to constitute more than an estimated 80% of the entire University of Virginia undergraduate TA population and a TA to student ratio of 1:12 in some courses, the Computer Science department still struggles to provide adequate help to every student who attends office hours. In an attempt to alleviate these pressures and provide a more organized structure to office hours, some courses have turned to requiring students to "queue" for help (Basu), in a similar vein to how one waits in line to enjoy a ride at an amusement park. The implementation of these queues has surfaced multiple social aspects of how students should move through queues, which can be examined using Social Construction of Technology.

Technical Problem Statement

For my capstone project, I am working in a team of four to redesign the course management system for CS2150, an integral part of the UVA Computer Science curriculum, and make it an attractive option for other professors, both within UVA and at other universities as well. As part of this redesign, we have identified four key aspects to focus on — assignment

submission, grading, office hours, and support requests. Each area presents their own significant technical, as well as social, challenges.

The existing system, named “Course Tools”, is written in PHP and originates from the early 2000s. It primarily functions by checking a user’s Netbadge credentials against an internal database and appending output to the final webpage to be served as appropriate. This tends to be inefficient for any user of the system who is not a student, as they then have a superfluous “Student” tab in the interface that is not relevant to the functions they need to perform. Students receive almost no feedback upon submitting an assignment, a far cry from most submission systems for programming assignments, which provide detailed test output. This is partly due to the fact that behind the scenes, grading is done almost entirely manually, with execution of the submission only occurring when a TA opens a tab containing that submission. The queue was not designed for the current volume of enrolled students, leading to sluggish response times during busy office hours, a problem not isolated to just this particular queue: at Stanford, “the site they were using was crashing from too many users” (Park). Its mobile interface, tacked on years after the system was originally created, is just as clunky as the desktop version and remains difficult to use.

Our redesign, tentatively named “Satori”, starts from scratch using Django, a popular Python web framework that integrates closely with databases. The project is split up into four main apps — core, assignments, queue, and tickets. For each app, we aim to create a scalable system that features an intuitive, easy-to-use interface. Core houses the main shared components of Satori, such as how we represent and display users and their roles, courses, comments, and permissions. Assignments handles displaying of assignments, assignment submission, grading, and a gradebook. Queue is dedicated exclusively towards the queue system of office hours.

Finally, tickets allow students to contact professors directly for help and make them aware of unforeseen circumstances that may require professor intervention. Our technology stack is currently composed of the aforementioned Django to power our website, gunicorn to serve Satori to users, MySQL to create the database that Satori uses, Python 3.8 to run both Django and gunicorn, Docker in which to run Django, gunicorn, MySQL, and Python, as well as to provide containers for compiling and running assignment submissions, and Bootstrap to style the frontend.

Social-Technical Aspects

Many stakeholders are present in a course management system, including students, teaching assistants, instructors, and administrators, making Social Construction of Technology a valuable approach to help understand the complex social issues of using queues to handle office hours. As a teaching assistant for CS1110: Introduction to Programming and CS2150: Program & Data Representation at the University of Virginia, both of which use queues, I am in the fortunate position of being able to interact directly with three of the major stakeholders — students, fellow teaching assistants, and the course instructors — and evaluate how queues affect them, while also being able to comment on administrators' reactions to queues.

In general, queues are beneficial to all stakeholders involved. Through a standardized process, students are able to have clear expectations for how and when they will receive help. Teaching assistants can form one-on-one working relationships with students and provide focused attention, whereas instructors are able to view aggregate data for how queues and teaching assistants are performing, as well as identify which students may need more personalized attention. Through a queue system, office hours may also be held in larger common

spaces co-located with other office hours (MacWilliam and Malan), reducing pressure on room reservations and access issues, allowing administrators to focus on other tasks.

Even today, with the advent of technology and a discipline that aims to utilize this new field of computer science to its fullest extent, many instructors still prefer to run their personal office hours using manual queues, wherein students line up outside the instructor's office to receive help. "Fifteen years ago, students signed up for such help by writing their name on a sheet of paper...eventually, that sheet of paper evolved into a whiteboard, but the process today remains largely the same. It works, and it is simple" (Malan). However, queues can be re-imagined by using the new tools afforded to us in the Information Age. Queue information can be enriched with students' data from all aspects of the course, paving the way for more complex algorithms that streamline student-TA interactions and provide shorter wait times.

The simplest queue is little more than a "first in, first out" system: the first student to enter the queue is guaranteed to receive help before the second student, and so on. Thusly, it is the easiest for instructors to implement in their courses and the most convenient for students and teaching assistants to use. This system tends to unfairly reward students who are able to be present when office hours start as they are able to maximize the full benefits of such a first in, first out algorithm. It also incentivizes entering the queue as many times as possible, especially when the queue grows longer during busy office hours. This creates frequent situations where students do not think critically about the issues they are facing (if any at all) before re-entering the queue for efficiency. Teaching assistants, as a result, are discouraged when seeing students rejoin the queue the moment they are finished being helped, or when a student admits that they have no immediate questions because they expected the queue to take longer, but still prevent the TA from moving on to help other students by coming up with questions so as to not waste their

spot. Faced with these limitations, instructors and administrators have come up with more complex solutions in an attempt to alleviate some of those concerns.

One type of an alternative queue is the "priority" queue, where the queue does not progress linearly but instead according to some inherent notion of a student's priority. This raises the immediate issue of how priority should be defined, a question that is left up to each course instructor. A SCOT issue is raised in that the students must use the priority defined by the instructor, even if it may not be the most beneficial to themselves. Regardless of what priority metric is chosen, it does create more work for the instructor, who must create a priority queue with those characteristics, and then closely monitor it to ensure that their algorithm is behaving as expected. One common indicator of priority is how many times a student has entered the queue, which aims to prevent students from abusing the linear mechanics of a simple queue. The more often a student joins a queue, the lower the priority becomes, and hence the longer it takes for them to receive help. Conversely, a student who rarely appears at office hours is more likely to have significant questions that they require a TA's help with when they do enter the queue. While this does encourage students to attempt to solve their problems before entering the queue, at the same time it rewards high-performing students and deprioritizes struggling students who do need to use the queue often for legitimate questions — arguably the opposite of how students should be prioritized.

Another type of priority queue allows TAs to rate each student after helping them, and the aggregate rating is then used as a proxy for how prioritized a student should be. This also seeks to reduce the number of students who do not respect their fellow students' time by joining the queue simply to reserve a spot and not because they have substantial questions. However, if students realize that their interactions with the TAs impacts how quickly they receive help, it

becomes much harder for TAs to form close relationships with the students, who feel that they are being graded and must be on their best behavior. The added layer of formality prevents TAs from acting as valuable resources, not only on the learning side but also on the personal side. Instructors commonly rely on teaching assistants, who interact at a closer level to the students, to provide deeper insight into how students are responding to the course, and this also potentially reduces the amount of unfiltered feedback TAs are able to gather.

Instead of having only one queue, priority or otherwise, there is the possibility of implementing multiple queues that operate in parallel, each for a separate purpose. For example, a priority queue could handle general cases, whereas a simple queue could be for any students who have small questions that take two minutes or less to answer. Again, this creates more overhead for the instructor, but allows more flexibility for students to get the customized help they need. At the same time, it means there are fewer teaching assistants monitoring each queue, leading to longer wait times in general.

Outside of the considerations of how to implement queues, there are also broader issues with the inclusion of queues in office hours. TAs may find themselves answering the same conceptual questions repeatedly, as there is no built-in mechanism to help multiple people at the same time when using queues. To address this, teaching assistants often resort to bypassing the queue and announcing spontaneous group lessons in an attempt to resolve multiple students' questions at once and clear up the queue. Unfortunately, while the lessons are well-attended, most students are reluctant to leave the queue due to the reasons described above. This creates extra stress on the queue, as not only are there less TAs tending to the queue while the lessons are being held, but students retain their position in the queue without consequences. Furthermore, the discoverability of queues needs to be considered. Whereas for traditional office hours,

students need only to head to the designated room at the correct time to receive help, the addition of queues requires that students know how to enter the queue. While instructors may make a best effort to explain how the queue system works at the beginning of the semester, there are inevitably students who do not pay attention or forget how to later on in the semester if they do not attend office hours earlier on. Without clear and easy discovery of how to use the queue, those students may be implicitly denied the opportunity to receive help from teaching assistants, especially when in a larger communal space where it is difficult to immediately identify the TAs.

Conclusion

While queues provide structure to otherwise-hectic office hours and enable one-on-one help, there are many design aspects that need to be addressed when deciding to go ahead with such an office hours system for a course. Proper thought must be given to all the stakeholders involved and how to balance each one's priorities. Through our capstone project, we hope to create a robust system that gives instructors the flexibility to create queues that work well for their course structure.

References

Basu, Kimaya. “Some CIS Courses Are so Overloaded That Students Wait More than an Hour for Homework Help.” The Daily Pennsylvanian, 5 Dec. 2018, www.thedp.com/article/2018/12/cis-120-office-hours-wait-time-penn-upenn-philadelphia.

MacWilliam, Tommy, and David J. Malan. “Scaling Office Hours: Managing Live Q&A in Large Courses.” Journal of Computing Sciences in Colleges, vol. 28, no. 3, Jan. 2013.

Malan, David J. “Virtualizing Office Hours in CS 50.” ACM SIGCSE Bulletin, vol. 41, no. 3, 2009, p. 303., doi:10.1145/1595496.1562969.

“Monstrous Class Sizes Unavoidable at Colleges.” NBCNews.com, NBCUniversal News Group, 24 Nov. 2007, www.nbcnews.com/id/21951104/ns/us_news-education/t/monstrous-class-sizes-unavoidable-colleges/.

Park, Elaine. “Honor Code Cases, Office Hour Queues among Challenges of Expanding CS Class Sizes.” The Stanford Daily, 29 Jan. 2018, www.stanforddaily.com/2018/01/28/honor-code-cases-office-hour-queues-among-challenges-of-expanding-cs-class-sizes/.

“PROV-008: Teaching Courses for Academic Credit.” Policy Directory, University of Virginia, 2 Aug. 2016, uvapolicy.virginia.edu/policy/prov-008.