You asked, now what? Modeling Students' Help-Seeking and Coding actions from Request to Resolution

Zhikai Gao
North Carolina State University
zgao9@ncsu.edu

Bradley Erickson
North Carolina State University
bericks@ncsu.edu

Yiqiao Xu
North Carolina State University
yxu35@ncsu.edu

Collin Lynch
North Carolina State University
cflynch@ncsu.edu

Sarah Heckman
North Carolina State University
sarah_heckman@ncsu.edu

Tiffany Barnes
North Carolina State University
tmbarnes@ncsu.edu

Demand for education in Computer Science has increased markedly in recent years. With increased demand has come to an increased need for student support, especially for courses with large programming projects. Instructors commonly provide online post forums or office hours to address this massive demand for help requests. Identifying what types of questions students are asking in those interactions and what triggers their help requests can in turn assist instructors in better managing limited help-providing resources. In this study, we aim to explore students' help-seeking actions from the two separate approaches we mentioned before and investigate their coding actions before help requests to understand better what motivates students to seek help in programming projects. We collected students' help request data and commit logs from two Fall offerings of a CS2 course. In our analysis, we first believe that different types of questions should be related to different behavioral patterns. Therefore, we first categorized students' help requests based on their content (e.g., Implementation, General Debugging, or Addressing Teaching Staff (TS) Test Failures). We found that General Debugging is the most frequently asked question. Then we analyzed how the popularity of each type of request changed over time. Our results suggest that implementation is more popular in the early stage of the project cycle, and it changes to General Debugging and Addressing TS Failures in the later stage. We also calculated the accuracy of students' commit frequency one hour before their help requests; the results show that before Implementation requests, the commit frequency is significantly lower, and before TS failure requests, the frequency is significantly higher. Moreover, we checked before any help request whether students changed their source code or test code. The results show implementation requests related to higher chances of source code changes and coverage questions related to more test code changes. Moreover, we use a Markov Chain model to show students' action sequences before, during, and after the requests. And finally, we explored students' progress after the office hours interaction and found that over half of the students improved the correctness of their code after 20 minutes of their office hours interaction addressing TS failures ends.

Keywords: computer science education, help-seeking, self-regulation, blended learning, markov models

1. INTRODUCTION

Help-seeking is a complex cognitive skill involving metacognition and self-evaluation to identify the need for assistance, identification of problems for support, and formulate requests (Aleven et al., 2006; Newman and Schwager, 1995; Newman, 2008). Effective help-seeking skills are essential for learners to develop a better understanding of assignments or class content that they do not understand (Newman, 2008). It is thus a key part of learning and is associated with a strong capacity for self-regulated learning through monitoring and goal-setting, among other aspects (Zimmerman, 2000; Zimmerman, 1989). Students' motivation in help-seeking has been analyzed by a number of authors (Ryan and Pintrich, 1997; Cheong et al., 2004; Zusho et al., 2007; Gao et al., 2022; Oberman, 2002, e.g.). These researchers have found that learners who felt comfortable and skillful in relating to others were more likely to ask for help. Further studies of specific help-seeking behaviors (e.g.(Roll et al., 2011; Roll et al., 2006; Vaessen et al., 2014)) have included analyses of the interaction between help-seeking and instructor feedback.

This work, however, has focused primarily on students' help-seeking behaviors in isolation or in the context of relatively focused problem-solving. It has not typically considered how help-seeking responds to prior problem-solving or how these help requests affect students' subsequent work. We aim to address that in this work by exploring the motivations, help-seeking, and problem-solving actions of computer science students as they work on programming projects. We chose to focus on coding for two reasons. First, computer science skills have become an increasingly important domain at all levels with increased demand at all grade levels and coding is an essential part of that (Beaubouef and Mason, 2005). Second, programming is a complex task that involves students in long-term complex problem-solving, which offers multiple opportunities for help-seeking and for a complex interrelationship between problem-solving, assistance, and outcomes. In contrast to prior research, we examined different types of help-seeking behaviors, including attending office hours and posting on a public class forum. We further analyze how students change their code when trying to receive help from others. This focus yields the following research questions.

- RQ1: What types of help are students seeking in online forums and office hour settings, and how do they differ?
- RQ2: How does the frequency of help-seeking change over the course of students' complex assignments, and can we use the project stages or sub-goals to predict this help-seeking?
- RQ3: How does student behavior change during different periods of the help-seeking process?

  RQ3.1 How often do students make commit before they seek help?

  RQ3.2 What part of the code do students change before they seek help?

  RQ3.3 How can we model and analyze students' action sequences during office hours' lifespan?

  RQ3.4 Do students make successful progress after office hours completes, and how to evaluate it?

Addressing these questions will not only enhance our understanding of students' help usage and behaviors. They will improve our ability to triage incoming help requests, separating students who need immediate help from those that can benefit from productive struggle or who have already solved their problems. This in turn, will allow instructors to better target scarce help resources, and it will support the development of adaptive platforms and tutors for help-seeking that can provide personalized support.

In order to address these questions, we began by identifying common help requests and coding behaviors. We manually labeled help-seeking requests from two different platforms used in a blended CS course. We then analyzed how these different types of help-seeking requests changed during the coding project lifespan. Finally, we compared students' code changes based on the timing of their commits before, during, and after they seek help; specifically, we investigated the commit frequency and location of the code change before they seek help in office hours or piazza.

As an extension from our previous publication (Gao et al., 2022), this paper adds an extra semester's data to the analysis, expands the original focus on what happens before help requests into all three phrases (before, during, and after) of help-seeking actions, and also proposed a potential method for evaluating the effectiveness of office hours interaction after it completed.

2. Related Work

As prior researchers have shown, self-regulation of learning through modulation of affective, cognitive, and behavioral processes is an essential component of learning across domains (Zimmerman, 1994). This regulation and essential help-seeking are driven in part by students' personal motivations (Newman, 1990; Newman, 1991; Newman, 1994), and by their general attitude toward learning (Ryan and Pintrich, 1997). In general, students who have better performance in an educational environment tend to have better metacognitive skills and tend to seek help more frequently (Karlsson et al., 2012). Ryan et al. (Ryan and Pintrich, 1997), for example, investigated motivational influences on help-seeking behavior in math classrooms, focusing on early teenagers' perception of the benefits and threats associated with such behavior. They designed a survey for 203 seventh- and eighth-graders on perceptions of social and cognitive competence, achievement goals, attitudes, and avoidance of and adaptive help-seeking behavior. Their findings indicate that social competence had an indirect effect on the avoidance of help-seeking. And the results illustrate the importance of linking the cognitive and social characteristics of students to provide a full understanding of teenagers' help-seeking behaviors.

Effective help-seeking is an important factor for successful CS learning. Bumbacher et al. (2013), for example, developed novel models to predict student learning gains based upon their semantic and structural features of their coding submissions. They found that these code features extracted from a single assignment can be used to predict whether or not students get help. Marwan et al. (Marwan et al., 2020) focused on analyzing and classifying students' help-seeking behaviors. Based upon an analysis of student-system log files, they proposed a taxonomy of unproductive help-seeking behavior in a programming environment. They then used these findings to design a hint interface that scaffolds appropriate help-seeking. Students using their platform were ultimately less than half as likely to produce unproductive help-seeking and thus improved overall.

While help-seeking is an important part of learning and self-regulation (Won et al., 2021), individual students' willingness to seek help and the types of help that they seek vary. An analy-

sis by Oberman (2002), for example, found that high-school students who were more motivated were more inclined to engage in instrumental help-seeking, while unmotivated students were more inclined to engage in executive help-seeking if they sought help at all. *Instrumental help seeking* occurs when a student is only seeking the support necessary to move them beyond a block but otherwise intends to solve the problem on their own, while *executive help-seeking* occurs when a student is looking for someone else to solve the problem for them, even in cases where they could do it on their own (s. Nelson-Le Gall, 1981). We would consider the former productive and the latter unproductive for learning. Oberman also found that help-seeking was also affected by general self-efficacy and motivation towards learning or achievement. This is consistent with other work by Zahn et al. (2022); in our target class help usage is not uniform across courses, with some students making little use of office hours and others soaking up the time. Students' willingness to do so is in turn, connected to their individual motivation and self-perceptions. In the present work, we do not focus on these motivational variables or on students who engage in help avoidance. However, the work that we are doing will inform future efforts to address help avoidance and executive help-seeking.

Unlike in-person learning environments, online help-seeking is far more open and produces a higher volume of data (Karabenick, 2011). Prior studies of online learning have typically shown a positive relationship between help-seeking behaviors and academic performance. In-person or small-group online office hours are an important venue for support in traditional and blended courses. Guerrero et al. (Guerrero and Rod, 2013) noted that students fail to take advantage of office hours when they are available despite the fact that the use of office hours correlates with performance. Griffin et al. (Griffin et al., 2014) extended this work by working to identify distinct factors that influence students' use of office hours. To that end, they developed a survey with 625 valid responses from undergraduate students at a large public university. Their results revealed that factors that significantly affect student use of office hours vary with one exception: the usefulness of instructional staff feedback. Thus, in this study, they suggest that instructors provide more efficient feedback to solve students' problems to encourage students to engage in office hours.

In summary, prior studies have provided different perspectives on how students engage in help-seeking. This work, however, has typically focused on analyzing individual behaviors or motivations in isolation from the students' activities before, while waiting for, and during the interaction. This is why we have focused on integrating our model of help-seeking with problem-solving actions and analyzing how those two streams of activity interact and affect one another. Our three research questions are built around this integration.

3. Dataset

3.1. Course Background

The data for this work originates from the Fall 2020 (F20) and the Fall 2021 (F21) offerings of an undergraduate CS2 course at a research-intensive university in the Southeast United States. Students in the course complete two projects, each worth 22% of their overall grade. Each project consists of two parts: 1) designing the system and creating a testing plan and 2) implementing the teaching staff (TS) UML diagram. Our focus will be on the second part of each project.

The course initially offered both online and in-person sections; however, due to the COVID-19 pandemic, all undergraduate course offerings covered in our dataset were moved online a

few weeks into the semester. The overall structure of the course as well as the task deadlines remained the same with interactions, including office hours moving to individual web meetings but retaining their overall structure.

As the first step in their course projects, the students were required to develop UML diagrams for the problem task. However, for the later coding stage, the students must all follow the instructor-provided UML diagram. This allowed them to fit a shared model for testing and evaluation. Students manage their repository using a Github[1] enterprise server. Whenever they push code, Jenkins[2], a continuous integration system, runs an Ant-driven build. Each build compiles the code and runs static analysis tools (Checkstyle[3], PMD[4], SpotBugs[5]), compiles the teaching staff test cases against the code to ensure it abides by the provided UML diagram, runs student-written tests to check for coverage metrics, runs teaching staff tests and finally provides feedback to the student based on their status. For the purposes of our analysis, we recorded the state of their code on each commit along with a record of the unit test results.

Over the course of the project, there are two intermediate milestones, or Process Points, for students to follow. Achieving the first two milestones provide a fraction of their project grade and unlock the test for the next milestone, while the final milestone defines the requirements to receive full credit. Students have one week for each stage of the project.

- **Milestone 1 - Process Points 1:** students complete a compiling skeleton, at least one test case, and fully Javadoc (i.e. no Checkstyle notifications) their code.
- **Milestone 2 - Process Points 2:** students achieve 60% statement coverage on their self-written tests.
- **Milestone 3 - Done:** students achieve 80% statement coverage, have no static analysis notifications, and all tests are passing (both teaching staff and student written).

After achieving each of the milestones, the students receive feedback on their code based on the shared tests. At the start, students only receive information about their compiling status and Checkstyle notifications related to developing Javadocs for their code. Once a student completes Milestone 1, they begin to see feedback about the remaining static analysis notifications. For completing Milestone 2, they begin seeing feedback regarding teaching staff test case failures. This feedback includes the number of tests passing or failing and for the failing tests, it provides a hint to reproduce locally. For the purpose of our analysis, we defined four project stages based on these milestones. At the beginning of the project, students are in stage 0; they then move to stage 1 after completing Milestone 1, and so on.

3.2. Help Seeking

Students in this course had two primary options to receive help, general questions could be posted to Piazza as online questions, and office hours which were accessed via a help request system called My Digital Hand. Direct emails and unofficial channels were actively discouraged. Table 1 offers general information about the help requests. These two alternatives are available

[1] https://github.com/
[2] https://www.jenkins.io/
[3] https://checkstyle.sourceforge.io/
[4] https://pmd.github.io/
[5] https://spotbugs.github.io/

Table 1: Help requests general information.

	Fall 20	Fall 21
Total students	303	405
Total Office Hours requests	1,330	800
Total canceled Office Hours requests	335	175
AVG Interaction Time	8.78 mins	8.23 mins
Total Piazza requests	1,202	838

in parallel, and it is possible that while students waiting for help on one platform, they post the same request on the other in order to increase the chance they can get it resolved early.

Piazza: At any time during the semester, students are able to post questions or comments on Piazza, an online forum for the class. When posting, students are required to select the assignment that they are seeking help on, if any (lab, project, or logistics issue). Students are able to make their posts private to instructors, but they are encouraged to post publicly, so other students are able to answer questions or receive similar help. The teaching staff also uses Piazza as a place to post updates or announcements. We removed the teaching staff posts and filtered posts down to only the second part of each project. We focused our attention only on the initial posting and removed all replies.

Office Hours During office hours, students were instructed to request help by filing a help ticket using My Digital Hand (MDH), an online support system (Smith et al., 2017). Students seeking help through MDH fill out a form listing the tasks they are working on, what questions they have or problems they are struggling with, and what steps they have taken thus far. The students are then placed in a queue which is monitored by the teaching staff. The teachers prioritize students to help based on their prior help-seeking, the content of their questions, and other factors. During in-person and online office hours, they may also opt to help students with similar questions in small groups where possible rather than individually. It is apparent that the instructors tend to favor students who have not received help previously. After the interaction is complete, the teachers will close the ticket, and both they and the student can enter follow-up information to describe the advice given, rate the outcome, and any potential follow-ups. Students who require more assistance may have their tickets re-opened or, more commonly, make a new ticket with additional questions. In our analysis, we first removed all tickets that were not related to the second part of the project. Next, we remove any follow-up tickets or tickets that were re-opened with the same content in order to avoid counting the same question twice.

3.3. COMMIT MINING

After the semester concluded, we ran the BUILDDATACOLLECTOR[6]. This tool iterates over each commit from each repository and runs the Jenkins build. The output files are mined for relevant data. The data includes commit metadata, static analysis notifications, information about all test cases, code counts, and coverage metrics. The data are stored in a SQL database.

[6] https://github.com/SOS-CER/BuildDataCollector

4. METHODOLOGY

After we collected the data, we first cleaned the office hours data and corrected some anomaly timestamps. Then, to address RQ1, we manually categorized the office hours requests and piazza requests and validated the tagging progress through Fleiss' Kappa agreement; To answer RQ2, we used calculated the frequency of each category of help request day by day. Finally, in RQ3, we perform statistical tests to discover, before the request is raised, if there are any significant differences in students' commit frequency and code change location when they request different types of help; we also examine students' action sequences by building a Markov model.

4.1. DATA PRE-PROCESSING

Before we start the analysis, we have some concerns about the large number of help requests that were completed within 1 minute (over 10%) in the MDH dataset. According to the instructor, two general misuses of the MDH system might cause the abnormal spike of extremely short interactions. First, when the teacher starts interacting with a student sometimes, they forget to open the associated ticket on MDH, and they usually notice their mistake towards the end of the interaction, which results in opening and closing the ticket in the system before moving to the next student. And according to the teaching staff's report, this mistake has happened relatively frequently since office hours moved online due to working with both MDH and Zoom. Second, there is no way to cancel requests from a student who has been called to receive help. If the student does not respond to a call to receive help, the teaching staff member may reopen and then cancel the ticket leading to a short interaction time.

Our first step in dealing with the short interaction data was to identify cases when the ticket was simply unopened when the interaction began. Only the start time would be incorrect in this case. To approximate the correct start time, we traced back to the teacher's very last action right before they started the ticket on MDH. Our method assumes the time between the teacher opening the ticket and the teacher's last action was incorrect if the time difference was too long, and we adjusted it accordingly rather than removing the ticket without adjusting the time.

There are four different types of situations right before any interaction begins, listed in Figure 1. In these examples, we consider two students which are labeled A and B. In the first situation, B makes a request after A completes an interaction when the queue is empty. In this situation, the teacher usually responds and opens the ticket very quickly after B made the request, so the teacher's response time (TR time) is usually within 10 seconds. Similarly, in situation 2, A completes after B requests help, so it depicts that when finished an interaction with A, the queue is not empty. And the teacher needs to help students one by one until the queue is empty or the shift is over. The TR time here represents the break between each pair of adjacent interactions, which is also usually within 30 seconds. Based on the teaching staff's experience, we believe that the TR time for both situation 1 and 2 should not be over 3 minutes. For situation 3, the teacher calls student B to come to the zoom room before they start the interaction; in this situation, the TR time also depends on how fast the student responds to the call. Still, the teaching staff reports that they will not keep waiting for the same students for more than 5 minutes; if the student did not show up in 5 minutes, they will call the next student instead. Situation 4 shows the cases when the teacher calls another student right before the interaction starts; again, if the TR time in this situation is more than 5 minutes, the teacher should call another student C in the queue. So, in summary, the TR time should not exceed 5 minutes if the last action before the start is "call"; otherwise, it should not exceed 3 minutes. We apply this

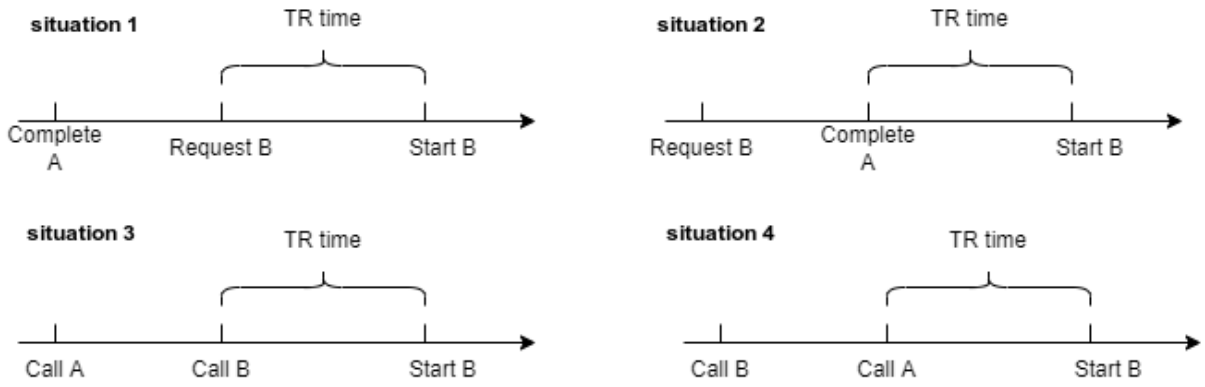


Figure 1: Four types of the last action before an interaction start.

rule to identify if any request has an abnormal start time and correct them by moving the start time early until it satisfies the rule.

The other issue of reopened tickets that should be canceled was quickly resolved after fixing the start time inaccuracies; we believe that an actual help interaction cannot be under 10 seconds, so we simply filter out all tickets that take less than 10 seconds and mark them as canceled, with a cancel time equal to their completion time.

4.2. RQ1: Categorization of Help Requests

In developing our analysis, we held the initial expectation that students' coding behaviors would differ based on the type of help that they required. Thus, we began by classifying the students' help requests based on their question content. In general, students begin the project by seeking to understand the overall functionality of the system and to build the general structure. They typically turn to author their own test cases based upon the functional goals with the goal of unlocking the instructor test cases on which their grade is partially based. Once these are unlocked, students frequently use the test cases to drive their development process, as intended, so they often focus their questions around those tests. Across all of these stages, they also face challenges with basic implementation errors and general static analysis notifications. Our goal was to separate students who were seeking support with basic development tasks (e.g. debugging) or with specific test cases from those who were seeking to address deeper comprehension questions or general notifications.

We therefore classified student help requests as follows:

- **General debugging and addressing issues:** students indicate they are receiving an error (i.e. null pointer exception) or describe unexpected behavior in their code.
- **Implementation and understanding:** students ask about how to implement some portion of the project or ask for clarification.
- **Improving test coverage:** students ask about how to improve their code coverage to achieve the 60% or 80% threshold.
- **Addressing TS test failures:** students indicate they are failing specific teaching staff test cases.

- **Addressing student-written test failures:** students indicate they are failing specific tests they wrote.
- **Addressing general test failures:** students indicate they are struggling with testing, but do not specify which type of test.
- **Addressing static analysis notifications:** students ask about Spotbugs, Checkstyle, or PMD notifications they receive.
- **Others or unclear:** students ask about unrelated topic (i.e. coding environment setup, documentation) or they are unclear with the type of help they need (i.e. a method name without explanation).

Manual Annotation To support our analysis, we engaged three experienced previous teaching assistants to manually tag the student questions across the two help contexts. We began by developing a shared coding process using a subset of Piazza and MDH posts. An initial round of grading yielded a Fleiss' Kappa agreement of 0.73 for MDH data and 0.69 for Piazza data. After one round of iterative evaluation and agreement, we achieved a final agreement of 0.81 for MDH data and 0.79 for Piazza. The primary area of disagreement in the Piazza data lies between questions in the Implementation and General Debugging categories. We therefore opted to perform additional segmentation based on whether or not students showed evidence of code execution.

4.3. RQ2: Timeline analysis

After the categorization was complete, we examined the frequency of the students' help requests by type and the change in those frequencies over time. Our hypotheses in this analysis were:

- Hypothesis 1: At the early stages of the project (before the Milestone 2 deadline), students mainly ask Implementation questions.
- Hypothesis 2: After achieving Milestone 2, students mainly asked about General Debugging and TS Test Failure questions.

We believe that consistent with instructor guidance, the student will try to understand the requirements of the project and how to implement each function first. Moreover, since Milestone 2 requires students to achieve 60% test coverage, it indicates that students need to finish most of the implementation to develop corresponding tests and reach Milestone 2. Therefore, we believe that prior to the Milestone 2 deadline, the most frequently asked questions should be Implementation related.

Moreover, after Milestone 2, students unlocked the TS test cases and also needed to develop more self-written tests to reach 80% code coverage. We believe that during this process, the student's main goal will be to correct their implementation based on the feedback of error messages and testing failures. Therefore, later in the projects, questions related to General Debugging and TS Test Failures should be more prevalent.

4.4. RQ3: Coding action analysis

In RQ3, the key question is to examine students' coding behaviors before, during, and after they seek help. For the before period, it is clear for both piazza or office hours, since we have recorded the time when students fill the ticket for office hours request on MDH and the time when student post their question on piazza. In RQ3.1 and RQ3.2, we therefore calculate the frequency of student's commit action and the location of the code change before they fill the ticket on MDH or post the question on piazza. However, there is no during period and after period for help requests on piazza, because there is no way to track when students see the reply of their post, and whether the reply is a proper solution, instead of asking for more detail on the question; while for office hours request, the during period is when students and teacher are actually talking, and the after period is after the interaction end and teacher close the ticket on MDH. Therefore, in RQ3.3, we only did action sequence analysis based on office hours lifespan; in RQ3.4, we also only focus on whether students making progress after the office hours interaction ends.

4.4.1. Code frequency

In order to analyze the precursors of students' help requests, we analyzed the state of the students' code on the last commit before they post a question to Piazza or file a ticket on MDH. We define these as the pre-help commit states. In analyzing these precursors, we began by analyzing how frequently students made commits within one hour before they seek help (request and office hours or post a piazza question). Because making commits can give students more feedback from Jenkins to figure out the problems within their code, we believe that a higher volume of those commits indicates that the students are unlikely to be working on higher-level implementation and are more focused on small-scale debugging or test passing. In other words, we have the following hypothesis:

- Hypothesis 3: Before an Implementation request, student commit frequency is low and before a TS test failure request, the frequency is high.

In order to test our theory, we counted the commits before one hour of each office hours request, calculate and reported the average for each type, then test the difference among different categories.

4.4.2. Code change location

In addition to the commit frequency, we also examined where code changes were made prior to help requests. Prior to requests for implementation help, we imagine that most commits include changes to the core functionality, while help requests for coverage focus on changes to the student-written unit tests. This leads to the following hypothesis:

- Hypothesis 4: Before an Implementation request, students are making changes to their source code and before a Coverage request, students make changes to their test code.

4.4.3. Action sequence analysis

We also try to model the student's action sequences through the use of a Markov Chain (Geyer, 1992). Specifically, there are 12 types of actions we focus on, four types of office hours (OH)

actions (including requesting, starting, ending, or canceling the interaction), four types of piazza post actions, and four different commit actions based on the different periods. Clearly, how one action could transmit to another is static. For instance, after requesting help, students can only transmit to either making a commit, posting questions on piazza, starting the interaction, or canceling the help request altogether. Therefore, we find all transactions that are possible in the office hours action sequence, and then calculate the probability of each transaction through our dataset and derive the Markov Chain.

For each type of request, we generate an additional Markov Chain to compare with the overall Markov Chain. We do not generate the Markov Chain for addressing static analysis, improving test coverage, addressing student-written test failures, and addressing general test failures due to the very limited number of requests. We also did not include the other/unclear category because it lacks a central theme to the OH requests. This leaves us with generating a Markov Chain for overall, general debugging and addressing errors, implementation, and understanding/addressing teaching staff test failures.

After generating the Markov Chains for each request type, we compare them with the overall chain and test the significance of their difference in each transaction using the Chi-square test (Tallarida and Murray, 1987). A significant result for a given transaction (p-value less than 0.05) would indicate that the transaction is statistically different from the overall Markov Chain. This could tell us if there is any behavioral difference between different types of help requests.

4.4.4. Post-help commit success

In RQ3.4, in order to evaluate students' progress after the office hours interaction completes, we tracked how much time it takes for the students to submit a commit with higher correctness. In these projects, the easiest way to measure the correctness of their solution is to compare the number of failed TS test cases. If the student solved their problem, the number of failed TS test cases should drop. However, this measure only works when students are asking questions about TS test failures. When they are asking how to implement something, the TS test failure is not a good indicator of a student's progress. So specifically, for each completed TS test failure office hours interaction, we first record the number of TS test failures before the interaction starts, track their next commit with a lower number of TS test failures, and calculate the time difference between such commit and the completion of interaction. Be aware that this time difference could be negative since students might already solve their failure during the interaction and submit to Jenkins. We call this time difference as Improved Failure Time for the rest of this paper. A lower number of Improved Failure Time would indicate the students fully address the failure really fast after the interaction, and it suggests the interaction is more successful. Also, we use the last commit before the interaction starts as the benchmark, not the one before the student file the request. We know there are some office hours requests with hours of wait time and students are continuing to work on the problem, and they might make some progress during that wait time; when students come to the office hours, the whole question they actually asked is completed based on their status before the interaction begins, not before they request the help, so it is only fair to use the commit before the interaction starts as a benchmark for measuring the success of the office hours interaction.

Table 2: Categorization Results.

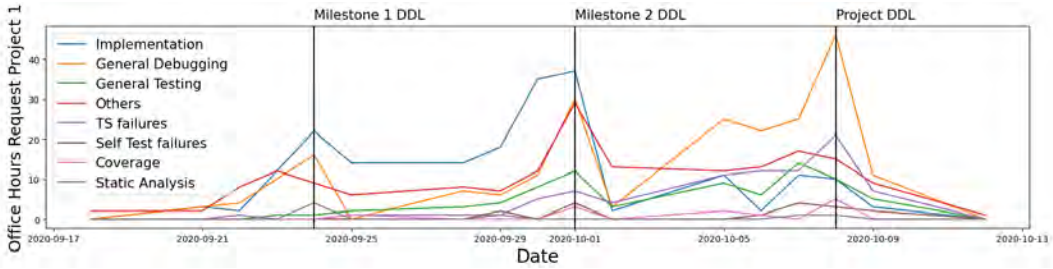
	F20		F21	
Category	MDH	Piazza	MDH	Piazza
Implementation	334	281	222	137
General Debugging	392	316	263	315
TS test failures	127	307	124	167
Self-written test failures	30	90	24	84
General test failures	103	64	38	31
Improving test coverage	21	31	20	27
Static analysis notifications	9	38	7	19
Others/Unclear	314	75	102	58
Total	1330	1202	800	838

5. Results

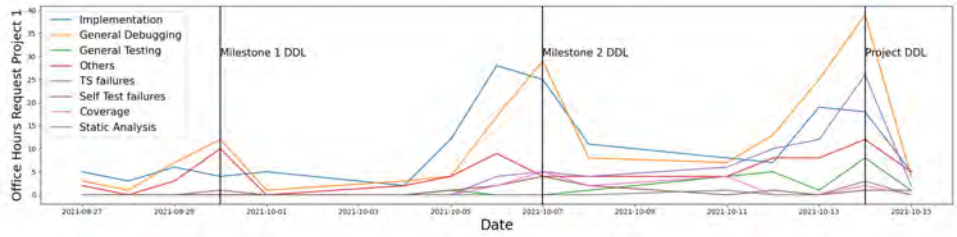
5.1. RQ1

Table 2 lists the results of our final request categorization. In both of the help platforms, the most popular type of request is General Debugging; For Fall 20, We identified 392 (29.4%) MDH requests and 316 (26.3%) Piazza requests in this category; for office hours requests, the next popular category is Implementation (25.1%), while for Piazza, there are more requests on Addressing TS Test Failures (25.5%) than Implementation (23.3%). The remaining categories are relatively uncommon with a frequency lower than 10% on each platform, except for the Others/Unclear category. We found a large amount of MDH requests containing very vague descriptions, which were categorized as Other/Unclear (23.6%). This is caused by the nature of the office hours process, the description we collected from MDH does not ask students to give very detailed information and students do not rely on it for getting the actual help; they would prefer to briefly describe their problem on MDH and elaborate on the detail orally when meeting with the teaching staff. This also matches with Gao's founding on the usage of MDH (Gao et al., 2021). In piazza, since students are more reliant on the description to get help, the amount of requests categorized as Others/Unclear is only 75 (6.2%).

Similar results we observed from Fall 21, the most popular category is General Debugging with 263 (32.9%) requests for MDH and 315 (37.6%) for piazza. Then for MDH, the implementation is the second most popular request (27.7%), followed by TS test failures (15.6%); while for piazza, TS failures are more popular (19.9%) than Implementation (16.3%). The results are very consistent between two semesters. The only difference between the two semesters is that the F21 has significantly fewer requests categorized as Others/Unclear. Our theory of this difference is that the F21 has much fewer requests overall, which means when the office hours resources are more sufficient, the students actually become more patient with the questions and resulting in less unambiguous descriptions.



(a) F20



(b) F21

Figure 2: Project 1 Office Hours request timeline for each type.

5.2. RQ2

5.2.1. MDH Requests Timeline Analysis

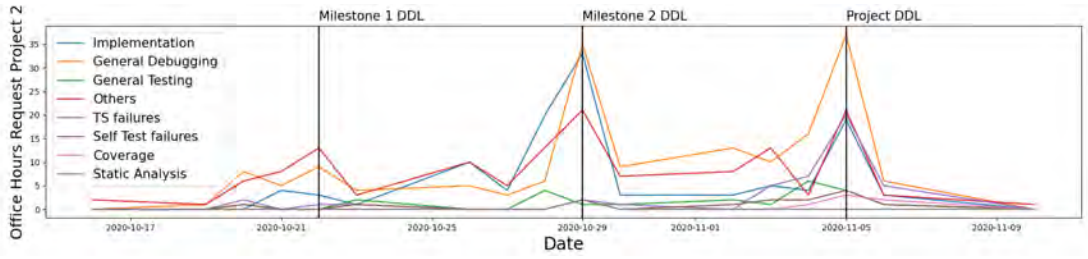
Figure 2 shows how the student's help request types changed over time during office hours. Results shows the same patterns between two semesters. Prior to the Milestone 1 deadline, students mainly asked the Implementation and General Debugging questions; during the next stage, before the Milestone 2 deadline, Implementation questions are dominating the help requests and maintain a very high number every single day. Then, after the Milestone 2 deadline, the amount of Implementation questions suddenly drop to less than 10 each day; while we witness a great increase in both the General Debugging and TS Test Failures, especially the General Debugging.

Similarly, Figure 3 provides the project 2 office hour requests. This also contains a high number of Implementation questions leading up to the Milestone 2 deadline, while after the deadline, Implementation instantly becomes less common. However before Milestone 1, the General Debugging is also one of the dominating categories, with roughly the same amount as Implementation. After Milestone 2, General Debugging stays as the most common request while the amount of TS Test Failure requests increased similar to the first project.

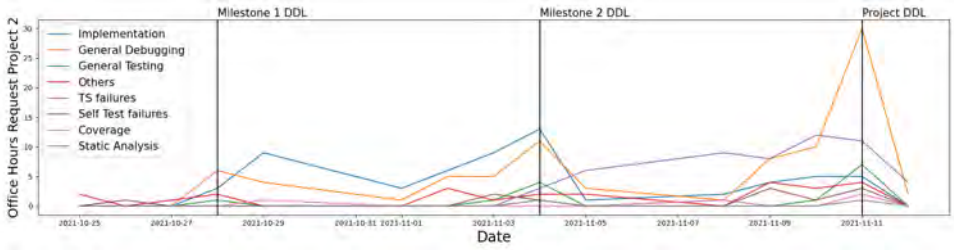
Furthermore, both project 1 and project 2 have the most requests at the exact date of each deadline. The project deadlines have more requests than the Milestone 2 deadlines, which have more requests than the Milestone 1 deadlines. This observation shows that most students are completing their work on the deadline date and require more help.

5.2.2. Piazza Requests Timeline Analysis

Figures 4 and 5 show the number of help requests students made on Piazza each day during the two projects. The three vertical lines represent the milestones and final deadline dates.



(a) F20



(b) F21

Figure 3: Project 2 Office Hours request timeline for each type.

Figure 4 shows the number of requests in Piazza that students made during project 1. We observed that the General Debugging category reaches its peak at all deadlines; the Implementation category is prominent during the first few stages; also the TS Test Failure category increases after the first milestone deadline and reaches its peak during the final deadline.

Figure 5 shows the number of requests in Piazza that students made during project 2. We observed that the General Debugging category always peaks at each deadline; the General Testing and Self-Testing categories peak at the project deadline; the TS Test Failure category raises rapidly from the Milestone 2 deadline to the Project deadline; and the Implementation category increases before each deadline, but overall decreases after each subsequent deadline.

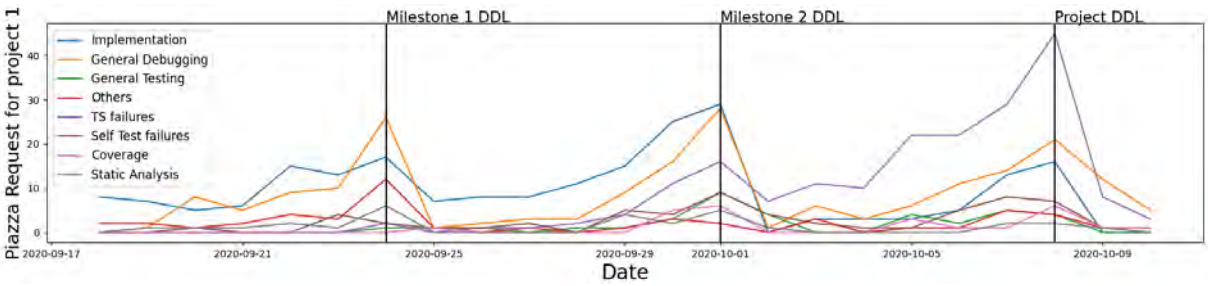
5.3. RQ3

5.3.1. Commit Frequency before help request

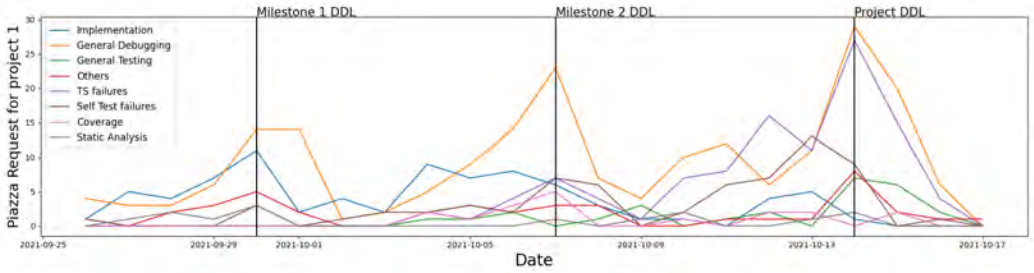
Figure 7 and Figure 6 show the results for RQ3.1. When comparing the frequency of commits, we find that students who ask about TS Test Failures are on average making more commits in the last hour than students making other help-seeking requests: In F21, 4.3 commits and 2.7 commits respectively ($p - value < 0.01$); while for F20, 4.6 commits and 2.9 commits respectively. Similarly, we see that students who seek help on Implementation requests are on average making fewer commits in the last hour than students who make any other request, 2.2 and 3.3 commits per hour in Fall 20, and 2.9 commits and 3.4 commits in Fall 21 respectively ($p - value < 0.01$).

5.3.2. Code changes of pre-help commits

As for the code change analysis results, we also found the results are identical between two semesters. For the Implementation pre-help commits, around 80% of those commits con-

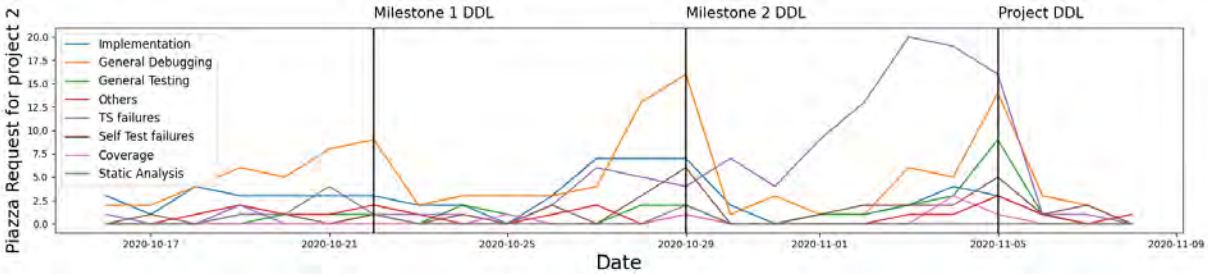


(a) F20

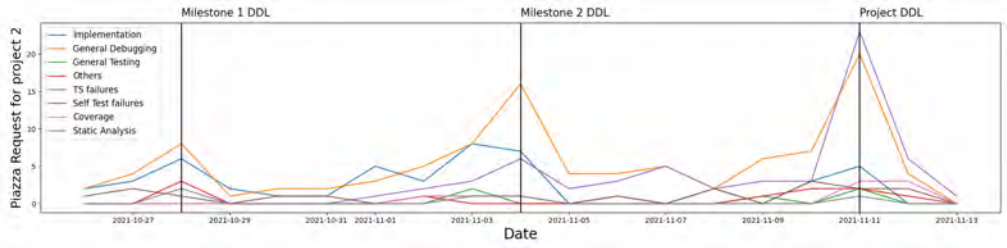


(b) F21

Figure 4: Project 1 Piazza request timeline for each type.



(a) F20



(b) F21

Figure 5: Project 2 Piazza request timeline for each type.

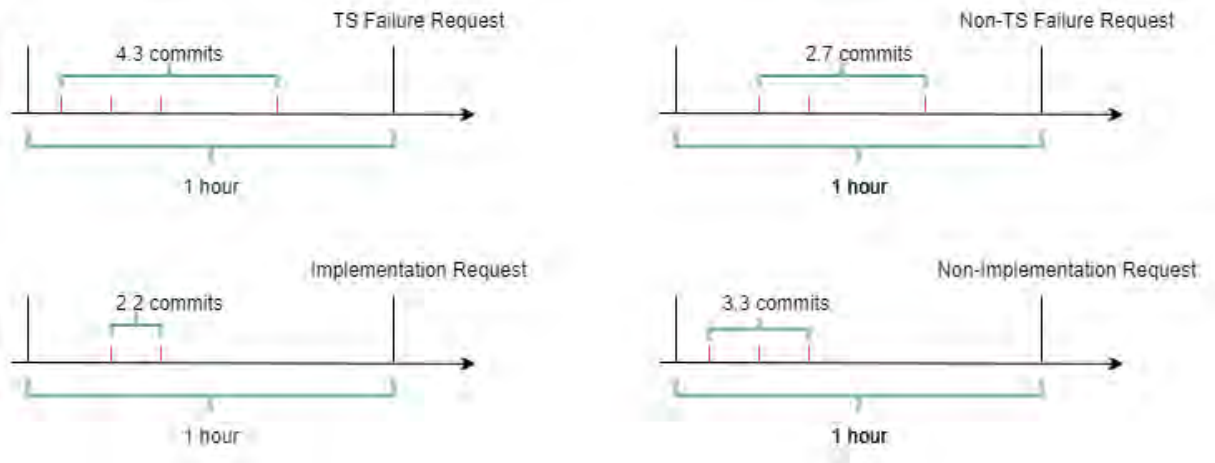


Figure 6: Commit frequency before help request in Fall 21.

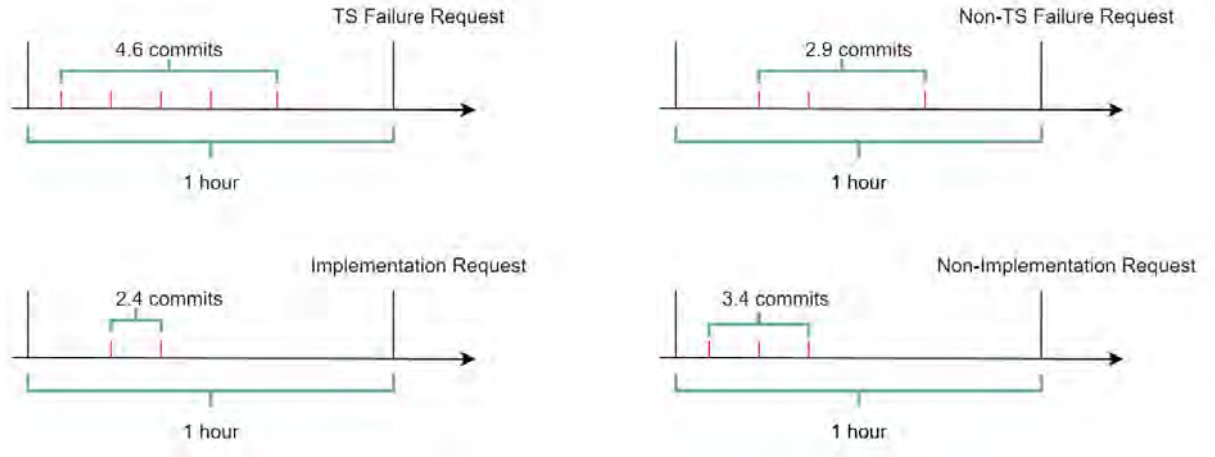


Figure 7: Commit frequency before help request in Fall 20.

Table 3: Percentage of pre-help commits with source or test code changes.

	F20			F21		
	Implementation	Coverage	Overall	Implementation	Coverage	Overall
src change	79.65%	60.78%	73.25%	81.37%	62.38%	74.12%
test change	64.25%	84.31%	70.12%	62.55%	89.85%	72.38%

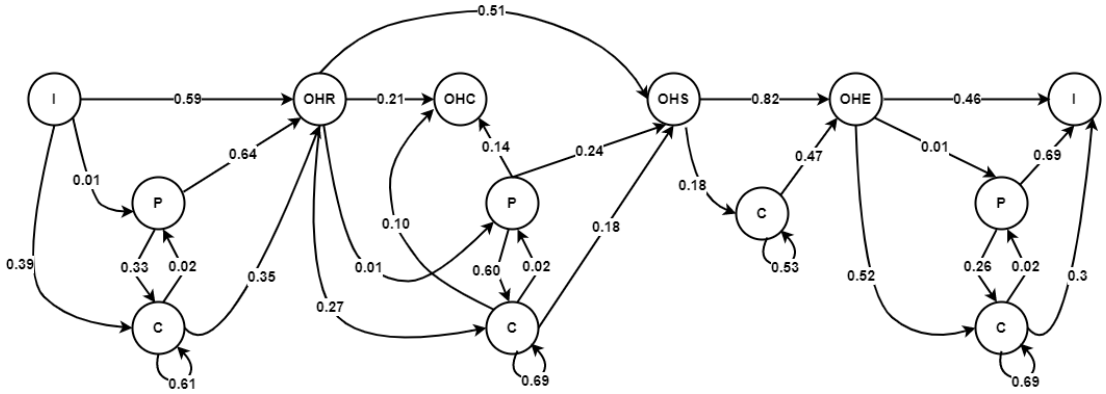


Figure 8: Overall action sequence Markov Chain results.

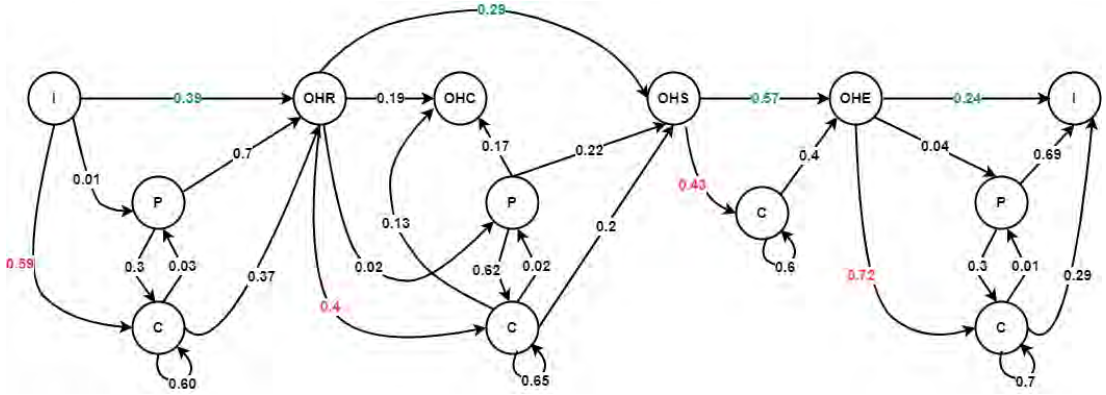


Figure 9: Action sequence Markov Chain results for TS failures request.

tain source code changes. Our chi-square test proves that Implementation requests have a significantly higher number of pre-help commits with source code change in both semesters ($p - value < 0.05$). For the Coverage pre-help commits, we found 84% - 90% of those commits contain test code changes. Similarly, we proved that compared with the rest of the commits, this number of commits with test code changes is significantly higher in both semesters ($p - value < 0.01$). Clearly, these results prove the correctness of our hypothesis 4.

5.3.3. Action Sequence

Figure 8 shows the resulting Markov Chain model for students' coding and help-seeking actions. In the figure, *I* stands for inactivity, which means students have no action in the 1-hour window before the office hours request or after it ends; *OHR*,*OHC*,*OHS*,*OHE* are office hours actions and represent the request, cancel, start, and end of the office hours interaction; *P* means students making a piazza post and *C* means students making a commit.

Before students made an office hours request, most of them (59%) did not make any commit or piazza post in one hour; 39% of them made some commits while only 1% made a piazza request during that time. And after making a piazza post during that time, most students (64%) did not make any commit before they request help in office hours; for those students, they post a question on piazza and wait for help without any commit, then they are very impatient with

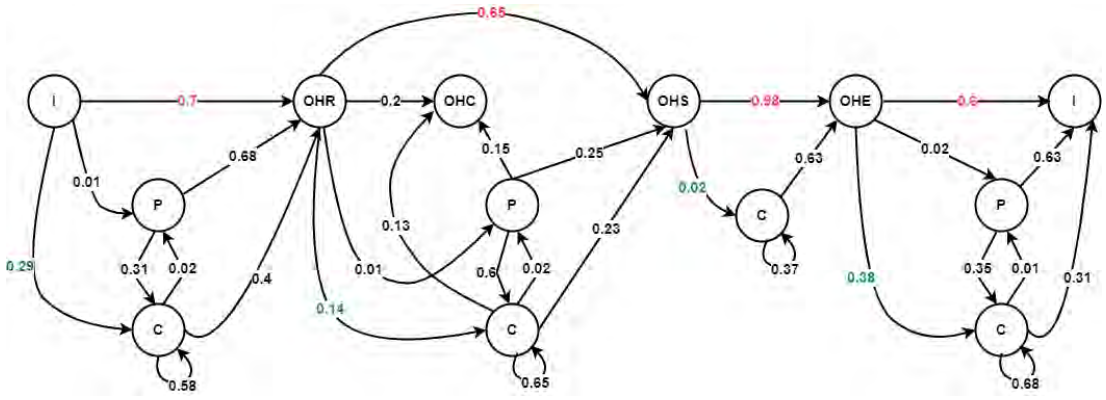


Figure 10: Action sequence Markov Chain results for Implementation request.

the waiting and ask for help again on the office hours. This type of behavior could result in wasting limited support staff because the questions asked on multiple platforms likely cover the same ground, thus forcing them to resolve each student's question twice with no additional help for the student. Therefore, the relative rarity of piazza posts while waiting for office hours help, from our perspective, may actually be good for the time spent as it suggests that the help-seeking resources are not wasted on the same question from the same student.

After an office hours request is raised, over half (51%) of the students will keep waiting without any commit or piazza action until the interaction start; only 27% of the students made commits while waiting; this relatively low number suggests the instructor should encourage students to be more actively seeking the solution, instead of just waiting for help; also, students are rarely asking questions on piazza while waiting for help on office hours. And for students who did make commits in that period, they are actually more likely (60%) to make a commit while waiting on both approaches before they start or cancel the office hours interaction.

During the office hours interaction, 82% of the interaction does not include any commit actions. This actually suggests teaching staffs are not wasting time on checking the results and feedback from Jenkins after they provide the solution to a student's problem. We also calculate the interaction time and found the interaction are significantly longer when students made a commit while the interaction was going (14.6 minutes VS 7.12 minutes).

Finally, after the office hours interaction ends, over half of the students made commits in the following one hour. Compared to actions before they made the request, students are far more active in making commits and checking for results. These results indicate those office hours interactions actually inspire students to make changes to their problems. Our finding also shows that before, during, and after the interaction, students are rarely asking for help on piazza; and for the rare case when they do, we found most of the time the piazza request and office hours request belong to the same category, suggesting the questions on both platforms are the same.

For each type of the primary four help requests, we generated a Markov Chain and compared each chain with the overall chain by testing for any significant difference in each transaction. After generating all chains, two of the chains yielded interesting results: Addressing TS failures requests and Implementation requests. The resulting chains are in Figure 9 and 10 respectively. The green numbering indicates that the probability of that transaction is significantly lower than in the overall chain, while the red numbering means the transaction is significantly higher.

Addressing Teaching Staff Test Failures: The most significant change in the TS test failure

chain is that all four transactions from the office hour actions to the commit actions have significantly higher probabilities($p < 0.01$). This indicates that when students need help addressing TS test failures, they are more likely to continue making commits throughout all four time periods when compared to the overall Markov Chain. With the students making more commits, we notice less remaining inactive between office hour actions.

Implementation and Understanding: For the Implementation and understanding requests, we find that, for all four transactions from the office hour actions to the commit actions, the probabilities are all significantly lower ($p < 0.01$) than the overall Markov Chain. This is the opposite of what we observed with the TS test failures sequence of actions. This is not a surprising result, given that when students are asking about implementation and general understanding, they likely have no idea how to write any related code. Thus, they are probably spending more time thinking about the problem instead of coding and making any commits.

5.3.4. Interaction success

Figure 11 shows the cumulative distribution of the time difference between office hours completion and the first commit with fewer TS failures when the office hours requests are about TS failures (Improved Failure Time as we defined before). There are 9.3% of the requests have improved before the completion (Improved Failure Time less than 0). As we discussed, those requests correspond to when students changed their code with the teacher, submitted their code to Jenkins, and received feedback showing the problem had been resolved during the meeting. Also, we found that half of the students resolved their TS failure within 19.7 minutes after the office hours requests ended. The growth of this Cumulative distribution function (CDF) curve slows down after this 20 minutes mark as well. Therefore, we believe that most office hours on addressing TS failure are quite successful; students did receive the answer to resolve their problems quickly.

6. Discussion and Conclusions

Our goal in this work was to investigate what kinds of requests students make when they seek help, what precursors exist in their code state before such requests are made, and what are students' actions before, during, and after an office hours request.

In addressing RQ1, we categorized the help requests based on students' purpose. Our results show that students are mostly asking questions about General Debugging, Implementation, and Addressing TS Test Failures. For office hours, General Debugging is the most popular category followed by Implementation followed by Addressing TS Test Failures. For Piazza requests, General Debugging are also the most common type, but Addressing TS Test Failures are more common than Implementation.

In RQ2, we analyzed the amount of help requests each day and observed how it changed during different stages of the project. We proved both of our hypotheses. Before Milestone 2, Implementation is the most common request (Hypothesis 1). After Milestone 2, General debugging is the most common request. We also see an obvious increase in TS Test Failure requests after Milestone 2 (Hypothesis 2).

In RQ3, we examined students' commits immediately prior to each help request and define it as a pre-help commit. We first looked at how frequently students are making commits 1 hour before seeking help. Our analysis shows that the frequency of committing before an

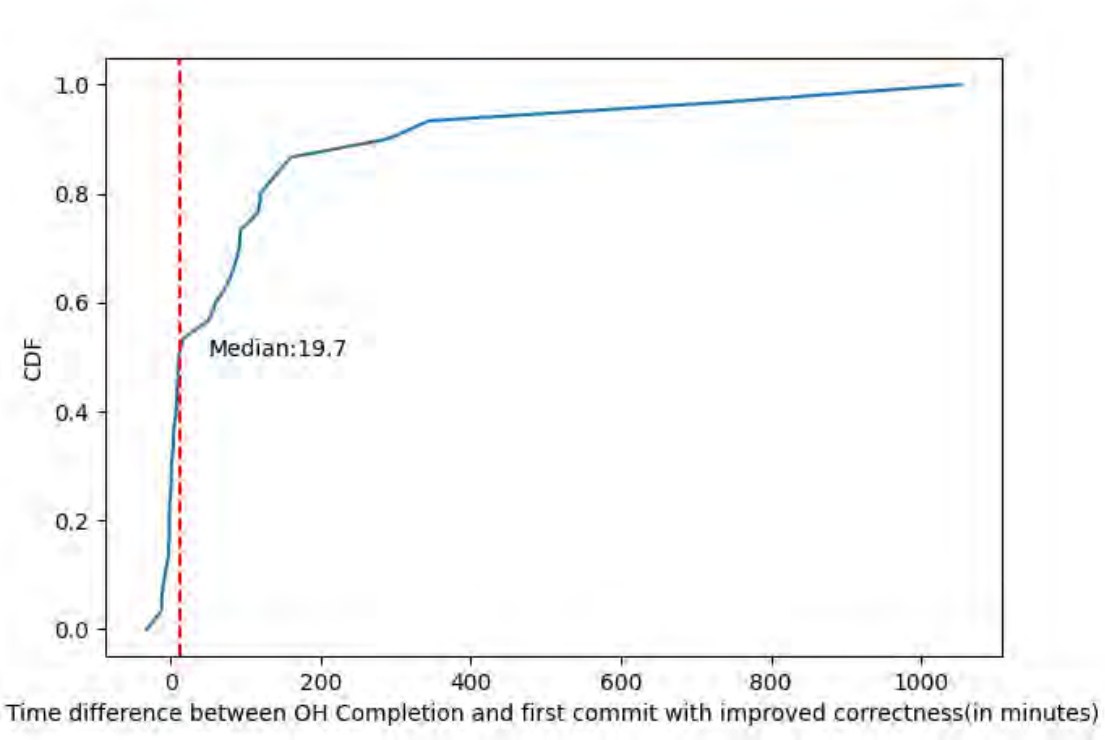


Figure 11: Results for RQ3.4.

Implementation request is significantly lower and for TS Test Failures, it's significantly higher. (Hypothesis 3). Then, we analyzed where the students change their code right before the help request. We find that most students change their source code before Implementation requests and change their test code before Coverage-related requests (Hypothesis 4). Our Markov Chain model shows the student's action sequence for different periods of office hours interaction. We find that before office hours, most students did not make any commit or piazza posts in one hour; When they are waiting in the queue, only 27% of the students made commits while waiting. During the interaction, teaching staffs are less likely to watch students check the solution by pushing their change to Jenkins. After the interaction ends, over half of the students keep making commits. Also, students very rarely make a piazza post before, during, or after they seek help in office hours. Finally, we found that when students raised questions about their TS test failures during office hours, half of them resolved some failures within 20 minutes after completing the interaction, which made us believe most office hours interactions are effective. This 20 minutes benchmark can also be used to evaluate the effectiveness of any office hours interaction.

Our current results show students' coding behavior is quite predictable and consistent among different semesters, and it does change when they need different types of help. Instructors can use our conclusion as a start and think about how to lead students to an efficient path when they encounter different types of difficulty throughout the projects.

7. Limitations and Future Work

Our study only analyzed the data from a single CS2 course in two offerings. Additional analysis of future semesters and other courses would enhance our understanding of how stable these behaviors are across cohorts and projects. Moreover, the two offerings are both Fall semesters with online office hours, the spring semesters or the in-person office hours might have completely different results.

Additionally, the quality of our analysis is limited by our post annotations. The high rate of disagreement for the Others/Unclear category was driven in part by how little information students included in their MDH request in Fall 2020. In future semesters we plan to address this problem by refining our analysis and addressing potential changes to the MDH platform to enforce more complex comments. On the other hand, the Fall 2021 semester does not show a such pattern. We also plan to extend our annotations and analysis to detect instances of executive versus instrumental help-seeking and to better understand how those affect our interactions.

Additionally, our examination of the pre-help commits was focused on both general frequencies and gestalt features. In future studies, we will conduct a more detailed analysis of the code status and align the contents of the help requests to specific code features and file locations. Finding such links between code features and request content would improve our understanding of when students seek help and potentially lead to an automotive model for detecting and predicting students who need help. Such a model would help instructors to actively and efficiently intervene in students' behavior and support a much more effective help management for the course.

References

Aleven, V., McLaren, B., Roll, I., and Koedinger, K. 2006. Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education 16,* 2, 101–128.

Beaubouef, T. and Mason, J. 2005. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin 37,* 2, 103–106.

Bumbacher, E., Sandes, A., Deutsch, A., and Blikstein, P. 2013. Student coding styles as predictors of help-seeking behavior. In *Proceeding of 16th International Conference on Artificial Intelligence in Education*, H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik, Eds. Springer, 856–859.

Cheong, Y. F., Pajares, F., and Oberman, P. S. 2004. Motivation and academic help-seeking in high school computer science. *Computer Science Education 14,* 1, 3–19.

Gao, Z., Erickson, B., Xu, Y., Lynch, C., Heckman, S., and Barnes, T. 2022. Admitting you have a problem is the first step: Modeling when and why students seek help in programming assignments. In *Proceedings of the 15th International Conference on Educational Data Mining*, A. Mitrovic and N. Bosch, Eds. International Educational Data Mining Society, Durham, United Kingdom, 508–514.

Gao, Z., Heckman, S., and Lynch, C. 2022. Who uses office hours? a comparison of in-person and virtual office hours utilization. In *Proceedings of the 53rd ACM Technical Symposium V.1 on Computer Science Education*. SIGCSE 2022. Association for Computing Machinery, New York, NY, USA, 300–306.

Gao, Z., Lynch, C., Heckman, S., and Barnes, T. 2021. Automatically classifying student help requests: A multi-year analysis. In *Proceedings of the 14th International Conference on Educational*

Data Mining, I.-H. Hsiao, S. Sahebi, F. Bouchet, and J.-J. Vie, Eds. International Educational Data Mining Society, 81–92.

GEYER, C. J. 1992. Practical Markov Chain Monte Carlo. *Statistical Science 7,* 4, 473 – 483.

GRIFFIN, W., COHEN, S. D., BERNDTSON, R., BURSON, K. M., CAMPER, K. M., CHEN, Y., AND SMITH, M. A. 2014. Starting the conversation: An exploratory study of factors that influence student office hour use. *College Teaching 62,* 3, 94–99.

GUERRERO, M. AND ROD, A. B. 2013. Engaging in office hours: A study of student-faculty interaction and academic performance. *Journal of Political Science Education 9,* 4, 403–416.

KARABENICK, S. A. 2011. Classroom and technology-supported help seeking: The need for converging research paradigms. *Learning and instruction 21,* 2, 290–296.

KARLSSON, L., KOIVULA, L., RUOKONEN, I., KAJAANI, P., ANTIKAINEN, L., AND RUISMÄKI, H. 2012. From novice to expert: Information seeking processes of university students and researchers. *Procedia-Social and Behavioral Sciences 45*, 577–587.

MARWAN, S., DOMBE, A., AND PRICE, T. W. 2020. Unproductive help-seeking in programming: What it is and how to address it. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. SIGCSE 2020. Association for Computing Machinery, 54–60.

NEWMAN, R. S. 1990. Children's help-seeking in the classroom: The role of motivational factors and attitudes. *Journal of Educational Psychology 82,* 1, 71.

NEWMAN, R. S. 1991. Goals and self-regulated learning: What motivates children to seek academic help. *Advances in Motivation and Achievement 7*, 151–183.

NEWMAN, R. S. 1994. Adaptive help seeking: A strategy of self-regulated learning. *Self-regulation of Learning and Performance: Issues and Educational Applications*, 283–301.

NEWMAN, R. S. 2008. The motivational role of adaptive help seeking in self-regulated learning. *Motivation and self-regulated learning: Theory, research, and applications*, 315–337.

NEWMAN, R. S. AND SCHWAGER, M. T. 1995. Students' help seeking during problem solving: Effects of grade, goal, and prior achievement. *American Educational Research Journal 32,* 2, 352–376.

OBERMAN, P. S. 2002. Academic help-seeking in the high school computer science classroom: Relationship to motivation, achievement, gender, and ethnicity. Ph.D. thesis, Emory University.

ROLL, I., ALEVEN, V., MCLAREN, B. M., AND KOEDINGER, K. R. 2011. Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and instruction 21,* 2, 267–280.

ROLL, I., ALEVEN, V., MCLAREN, B. M., RYU, E., D BAKER, R. S., AND KOEDINGER, K. R. 2006. The help tutor: Does metacognitive feedback improve students' help-seeking actions, skills and learning? In *International conference on intelligent tutoring systems*, M. Ikeda, K. D. Ashley, and T.-W. Chan, Eds. Springer, 360–369.

RYAN, A. M. AND PINTRICH, P. R. 1997. "should i ask for help?" the role of motivation and attitudes in adolescents' help seeking in math class. *Journal of Educational Psychology 89,* 2, 329.

S. NELSON-LE GALL. 1981. Help-seeking: An understudied problem-solving skill in children. *Developmental Review 1,* 3, 224–246.

SMITH, A. J., BOYER, K. E., FORBES, J., HECKMAN, S., AND MAYER-PATEL, K. 2017. My digital hand: A tool for scaling up one-to-one peer teaching in support of computer science learning. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '17. Association for Computing Machinery, New York, NY, USA, 549–554.

TALLARIDA, R. J. AND MURRAY, R. B. 1987. Chi-square test. In *Manual of Pharmacologic Calculations: With Computer Programs*, B. Jones, Ed. Springer New York, New York, NY, 140–142.

VAESSEN, B. E., PRINS, F. J., AND JEURING, J. 2014. University students' achievement goals and help-seeking strategies in an intelligent tutoring system. *Computers & Education 72*, 196–208.

WON, S., HENSLEY, L. C., AND WOLTERS, C. A. 2021. Brief research report: Sense of belonging and academic help-seeking as self-regulated learning. *The Journal of Experimental Education 89,* 1, 112–124.

ZAHN, M., BATTESTILLI, L., AND HECKMAN, S. 2022. Academic help seeking patterns in introductory computer science courses. In *2022 ASEE Annual Conference & Exposition*. ASEE Conferences, Minneapolis, MN. https://peer.asee.org/41526, Paper ID #37738.

ZIMMERMAN, B. J. 1989. Models of self-regulated learning and academic achievement. In *Self-regulated learning and academic achievement*, B. Zimmerman and D. Schunk, Eds. Springer, 1–25.

ZIMMERMAN, B. J. 1994. Dimensions of academic self-regulation: A conceptual framework for education. *Self-regulation of Learning and Performance: Issues and Educational Applications 1*, 33–21.

ZIMMERMAN, B. J. 2000. Attaining self-regulation: A social cognitive perspective. In *Handbook of self-regulation*, M. Boekaerts, P. R. Pintrich, and M. Zeidner, Eds. Academic Press, 13–39.

ZUSHO, A., KARABENICK, S. A., BONNEY, C. R., AND SIMS, B. C. 2007. Contextual determinants of motivation and help seeking in the college classroom. *The scholarship of teaching and learning in higher education: An evidence-based perspective*, 611–659.