Admitting you have a problem is the first step: Modeling when and why students seek help in programming assignments.

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

ABSTRACT

In computer science education timely help seeking during large programming projects is essential for student success. Help-seeking in typical courses happens in office hours and through online forums. In this research, we analyze students coding activities and help requests to understand the interaction between these activities. We collected student's help requests during coding assignments on two different platforms in a CS2 course, and categorized those requests into eight categories (including implementation, addressing test failures, general debugging, etc.). Then we analyzed the proportion of each type of requests and how they changed over time. We also collected student's coding status (including what part of the code changed and the frequency of commits) before they seek help to investigate if students share a similar code change behavior leading to certain type of help requests.

Keywords

Help Seeking, Categorization, Data Mining, CS2, Computer Science Education

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 formulating requests [1, 16, 15]. Help seeking is essential for learners to develop a better understanding of assignments or class content that they do not understand [15]. Effective help-seeking is a key part of learning and is associated with a capacity for self-regulated learning through monitoring and goal-setting among other aspects [22, 24]. Motivation in help-seeking has been analyzed by a number of authors [19, 4, 25, 5, e.g] . These researchers have noted 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. [17, 18, 21]) has included analyses of the interaction between help-seeking and instructor feedback. However, most of this prior research focused only on students' help seeking behaviours in isolation or in the context of relatively focused problem solving. It has not typically exampled that help-seeking responds to prior problem-solving, and how these help requests affect their subsequent work.

In this study, our goal is to explore the motivations, help-seeking, and problem solving actions of computer science students seeking help for their 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 [2]. 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 combined different types of student help seeking behaviours including attending office hour and posting on a public class forum. We analyze how students change their code when trying to receive help from others. Thus, in this work, we answer 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: What types of coding behaviors do students engage in before seeking help?

In order to address these questions we began by identifying common help requests and coding behaviors. We manually

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

https://doi.org/10.5281/zenodo.6853173

labeled help seeking requests from two different platform from a blended course. We then analyzed how these different types of help seeking requests changed during the coding project lifespan. Finally, we compared students' coding changes based on the commit frequency and the number of their passed test cases before and after they seek help.

2. RELATED WORK

2.1 Help Seeking Behaviours

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 [23]. This regulation and essential help-seeking is driven in part by students personal motivations [13, 14, 12], and by their general attitude toward learning [19]. In general students who have better performance in an educational environment tend to have better metecognitive skills and tend to seek help more frequently [10]. Ryan et al. [19], 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 behaviour. 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 behaviour. There finding indicates social competence had an indirect effect on avoidance of help seeking. And the results illustrate the importance of linking cognitive and social characteristics of students to provide a full understanding of teenager help seeking.

Help seeking is an important factor to success in learning programming. Bumbacher et al. [3], 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 got help. Marwan et al. [11] focused on analyzing and classifying students' help-seeking behaviors. Based upon an analysis of student-system logfiles they proposed a taxonomy of unproductive help seeking behaviour in a programming environment. They then used these findings to design a hint interface that scaffolded appropriate help-seeking. Students using their platform were ultimately less than half as likely to produce unproductive help-seeking and thus improved overall.

2.2 Student Help Seeking Behaviour Analysis

Unlike in-person environments online help-seeking is far more open and produces a higher volume of data [9]. Prior studies of online learning have typically shown a positive relationship between help seeking behaviours 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. [8] 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. [7] 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. The results revealed that factors that significantly affect student use of office hour vary with one exception: usefulness of instructional staff feedback. Thus, in this study, they suggest instructors provide more efficient feedback to solve students problems to encourage students to engage in office hour.

3. DATASET

3.1 Course Background

The data for this work originates from the Fall 2020 offering of a 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 courses 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 a first step in their course projects the students are 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 allows 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, 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, while the final milestone defines the requirements to receive full credit.

- **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).

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

After achieving each of the milestones, the students receive feedback on their code based upon 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 purposes of our analyses we defined four project stages based upon 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 options for help, general questions could be posted to Piazza an online question and answer forum that was monitored by the instructors and took answers from other students; and office hours which were accessed via a help request system called My Digital Hand.

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 a category for their question. These categories correspond to logistics or each assignment, lab, or project. 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 remove the teaching staff posts and filter posts down to only the second part of each project. We focus our attention only on the initial posting and remove all replies.

Office Hours During teaching staff office hours, students request help by filing a help ticket using My Digital Hand (MDH) an online support system [20]. Students seeking help through MDH fill out a form listing the tasks they are working on, what question they have or problem 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 upon their questions and may help them individually or in small groups. After the interaction is complete the teachers will close the ticket and both they and the student can enter followup information to describe the advice given, rate the outcome, and any potential followups. 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 remove all tickets that were not related to the second part of the projects. Next, we remove any follow-up tickets or tickets that were re-opened with the same content.

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

4.1 RQ1: Categorization of Help Requests

In developing our analysis we held the initial expectation that students' coding behaviors would differ based upon the type of help that they required. Thus, we began by classifying the students' help requests based upon 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 authoring 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 to 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 acheive 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 TAs 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 0.72 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 upon whether or not students showed evidence of code execution.

4.2 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 students 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 students' 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.

4.3 RQ3: Pre-help code analysis

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 1 hour before their pre-help commit. We believe that a higher volume of pre-help commits indicate that the students are unlikely to be working on higher level implementation and are more focused on small-scale debugging or test passing.

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 hypotheses:

Category	MDH	Piazza	Overall
Implementation	334	281	615
General Debugging	392	316	708
TS test failures	127	307	434
Self-written test failures	30	90	120
General test failures	103	64	167
Improving test coverage	21	31	52
static analysis notifications	9	38	47
Others/Unclear	314	75	389
Total	1330	1202	2532

Table 1: Categorization Results

- Hypothesis 3: Before an Implementation request, student commit frequency is low and before a TS test failure request, the frequency is high.
- Hypothesis 4: Before an Implementation request, students are making changes to their source code and before a Coverage request, students are making changes to their test code.

5. RESULTS

5.1 RQ1

Table 1 lists the results of our final request categorization. In both of the help platform, the most popular type of request is General Debugging; We identified 392 (29.4%) MDH requests and 316 (26.3%) Piazza requests in this category. For office hours request, 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 student 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 [6]. In piazza, since students more reliant on the description to get help, the amount of requests categorized as Others/Unclear is only 75 (6.2%).

5.2 RQ2

5.2.1 MDH Requests Timeline Analysis

Figure 1 shows how the student's help request types changed over time during office hours. 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.

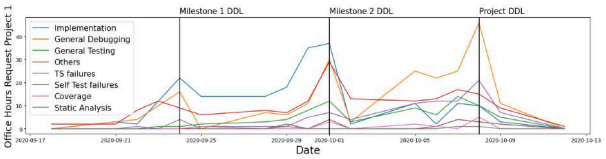

Figure 1: Project 1 Office Hours request timeline for each type.

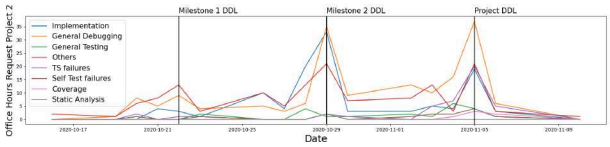

Figure 2: Project 2 Office Hours request timeline for each type.

Similarly, Figure 2 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 increase 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 3 and 4 show the number of help requests student made on Piazza each day during the two projects. The three vertical lines represent the milestones and final deadline dates.

Figure 3 shows the number of requests in Piazza that students made during project 1. We observed that the General Debugging category reaches the peak at all deadlines; the Implementation category is prominent during the first few stages;and the TS Test Failure category increases after the first milestone deadline and reaches its peak during the final deadline.

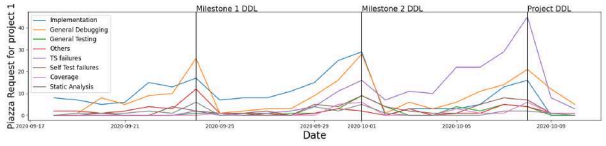

Figure 3: Project 1 Piazza request timeline for each type.

Figure 4 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 Project deadline; and the Implementation category increase before each deadline, but overall decreases after each subsequent deadline.

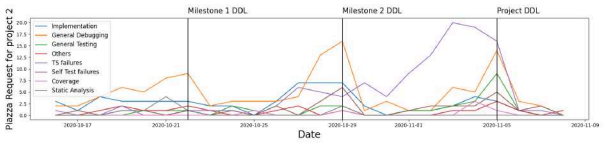

Figure 4: Project 2 Piazza request timeline for each type.

	Implementation	Coverage	Overall
src change	79.65%	60.78%	73.25%
test change	64.25%	84.31%	70.12%

Table 2: Percentage of pre-help commits with source or test code changes

5.3 RQ3

5.3.1 *Frequency of pre-help commits*

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, 4.6 commits and 2.9 commits respectively (p-value < 0.0001). 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, 2.4 and 3.4 commits per hour respectively (p-value < 0.0001).

5.3.2 *Code changes of pre-help commits*

For the Implementation pre-help commits, 490 (79.65%) of those commits contains source code changes. Our chi-square test proves that Implementation requests have a significant higher number of pre-help commits with source code change (p-value < 0.05). For the Coverage pre-help commits, we found 43 (84.31%) of those commits contains test code changes. Similarly, we proved that this high number of commits with test code changes is significant than the rest of commits (p-value < 0.01).

6. DISCUSSION AND CONCLUSIONS

Our goal in this work was to investigate what kinds of requests students make when they seek help and what precursors exist in their code state before such requests are made.

In addressing RQ1, we categorized the help requests based on student's purpose. Our results shows 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 request each day and observed how it changed during different stages of the

project. We proved both of our hypothesis. 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 of 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 the pre-help commit. We firstly looked at how frequently students are making commits 1 hour before seeking help. Our analysis shows that the frequency of committing before an Implementation request is significantly lower and for TS Test Failures, it's significantly higher. 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.

Our current results shows student's coding behavior is quite predictable and it does change when they need different types of help. Instructor 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 a single offering. Additional analysis of future semesters and other courses would enhance our understanding of how stable these behaviors are across cohorts and projects. 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 requests. 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.

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 link of 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 student's behavior, and support a much effective help management for the course.

Besides the pre-help commits, we also plan to analyze changes that the students make to their code during and after help requests to get a better assessment of how they process and implement guidance. This work has long-term potential to guide automated feedback and to assist in general triage for help responses. For example, after we figure out an accurate way to measure the success of commits, we can combine it with this work and evaluate the effectiveness of each help-seeking interaction by monitoring when student make an successful commit after the help. The instructor can use this information to quickly identify which students need further guidance or analyze why the help is ineffective.

8. ACKNOWLEDGEMENTS

This material is based upon work supported by NSF under grants #1821475 "Concert: Coordinating Educational Interactions for Student Engagement" Collin F. Lynch, Tiffany Barnes, and Sarah Heckman (Co-PIs).

References

[1] Vincent Aleven et al. "Toward meta-cognitive tutoring: A model of help seeking with a Cognitive Tutor". In: *International Journal of Artificial Intelligence in Education* 16.2 (2006), pp. 101–128.

[2] Theresa Beaubouef and John Mason. "Why the high attrition rate for computer science students: some thoughts and observations". In: *ACM SIGCSE Bulletin* 37.2 (2005), pp. 103–106.

[3] Engin Bumbacher et al. "Student coding styles as predictors of help-seeking behavior". In: *International Conference on Artificial Intelligence in Education*. Springer. 2013, pp. 856–859.

[4] Yuk Fai Cheong, Frank Pajares, and Paul S Oberman. "Motivation and academic help-seeking in high school computer science". In: *Computer Science Education* 14.1 (2004), pp. 3–19.

[5] Zhikai Gao, Sarah Heckman, and Collin Lynch. "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. Providence, RI, USA: Association for Computing Machinery, 2022, pp. 300–306. ISBN: 9781450390705. DOI: 10.1145/3478431.3499334. URL: https://doi.org/10.1145/3478431.3499334.

[6] Zhikai Gao et al. "Automatically Classifying Student Help Requests: A Multi-Year Analysis." In: *International Educational Data Mining Society* (2021).

[7] Whitney Griffin et al. "Starting the conversation: An exploratory study of factors that influence student office hour use". In: *College Teaching* 62.3 (2014), pp. 94–99.

[8] Mario Guerrero and Alisa Beth Rod. "Engaging in office hours: A study of student-faculty interaction and academic performance". In: *Journal of Political Science Education* 9.4 (2013), pp. 403–416.

[9] Stuart A Karabenick. "Classroom and technology-supported help seeking: The need for converging research paradigms". In: *Learning and instruction* 21.2 (2011), pp. 290–296.

[10] Liisa Karlsson et al. "From novice to expert: Information seeking processes of university students and researchers". In: *Procedia-Social and Behavioral Sciences* 45 (2012), pp. 577–587.

[11] Samiha Marwan, Anay Dombe, and Thomas W Price. "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*. 2020, pp. 54–60.

[12] Richard S Newman. "Adaptive help seeking: A strategy of self-regulated learning." In: (1994).

[13] Richard S Newman. "Children's help-seeking in the classroom: The role of motivational factors and attitudes." In: *Journal of educational psychology* 82.1 (1990), p. 71.

[14] Richard S Newman. "Goals and self-regulated learning: What motivates children to seek academic help". In: *Advances in motivation and achievement* 7 (1991), pp. 151–183.

[15] Richard S Newman. "The motivational role of adaptive help seeking in self-regulated learning". In: *Motivation and self-regulated learning: Theory, research, and applications* (2008), pp. 315–337.

[16] Richard S Newman and Mahna T Schwager. "Students' help seeking during problem solving: Effects of grade, goal, and prior achievement". In: *American Educational Research Journal* 32.2 (1995), pp. 352–376.

[17] Ido Roll et al. "Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system". In: *Learning and instruction* 21.2 (2011), pp. 267–280.

[18] Ido Roll et al. "The help tutor: Does metacognitive feedback improve students' help-seeking actions, skills and learning?" In: *International conference on intelligent tutoring systems*. Springer. 2006, pp. 360–369.

[19] Allison M Ryan and Paul R Pintrich. "" Should I ask for help?" The role of motivation and attitudes in adolescents' help seeking in math class." In: *Journal of educational psychology* 89.2 (1997), p. 329.

[20] Aaron J. Smith et al. "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. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 549–554. ISBN: 9781450346986. DOI: 10.1145/3017680.3017800. URL: https://doi-org.prox.lib.ncsu.edu/10.1145/3017680.3017800.

[21] Bram E Vaessen, Frans J Prins, and Johan Jeuring. "University students' achievement goals and help-seeking strategies in an intelligent tutoring system". In: *Computers & Education* 72 (2014), pp. 196–208.

[22] Barry J Zimmerman. "Attaining self-regulation: A social cognitive perspective". In: *Handbook of self-regulation*. Elsevier, 2000, pp. 13–39.

[23] Barry J Zimmerman. "Dimensions of academic self-regulation: A conceptual framework for education". In: *Self-regulation of learning and performance: Issues and educational applications* 1 (1994), pp. 33–21.

[24] Barry J Zimmerman. "Models of self-regulated learning and academic achievement". In: *Self-regulated learning and academic achievement*. Springer, 1989, pp. 1–25.

[25] Akane Zusho et al. "Contextual determinants of motivation and help seeking in the college classroom". In: *The scholarship of teaching and learning in higher education: An evidence-based perspective* (2007), pp. 611–659.