Factors Influencing Students' Help-Seeking Behavior while Programming with Human and Computer Tutors

Thomas W. Price
North Carolina State University
Raleigh, NC 27606
twprice@ncsu.edu

Zhongxiu Liu
North Carolina State University
Raleigh, NC 27606
zliu24@ncsu.edu

Veronica Cateté
North Carolina State University
Raleigh, NC 27606
vmcatete@ncsu.edu

Tiffany Barnes
North Carolina State University
Raleigh, NC 27606
tmbarnes@ncsu.edu

ABSTRACT

When novice students encounter difficulty when learning to program, some can seek help from instructors or teaching assistants. This one-on-one tutoring is highly effective at fostering learning, but busy instructors and large class sizes can make expert help a scarce resource. Increasingly, programming environments attempt to imitate this human support by providing students with hints and feedback. In order to design effective, computer-based help, it is important to understand how and why students seek and avoid help when programming, and how this process differs when the help is provided by a human or a computer. We explore these questions through a qualitative analysis of 15 students' interviews, in which they reflect on solving two programming problems with human and computer help. We discuss implications for help design and present hypotheses on students' help-seeking behavior.

1 INTRODUCTION AND BACKGROUND

Programming is a challenging skill to learn, particularly for novices. Getting "stuck" on an assignment can place an emotional toll on students [19] and may result in the student giving up. Ideally in these situations, a student will be able to seek help from a professor or teaching assistant and receive one-on-one, expert tutoring, arguably the most effective form of instruction [6]. However, experts – or even knowledgeable peers – are not always available or accessible, especially in overcrowded classrooms, and they may be absent altogether in informal learning settings. To address this, researchers and educators are increasingly augmenting programming environments with help features to aid students who are stuck when no human is available.

ICER'17, August 18-20, 2017, Tacoma, WA, USA.

978-1-4503-4968-0/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3105726.3106179

There are many forms of embedded help found in programming environments, including enhanced error messages [13], program visualization tools [38] and debugging aides [22]. Particularly promising are Intelligent Tutoring Systems (ITSs), which attempt to play the role of the human tutor [42]. In the domain of computing, ITSs can offer intelligent, contextual help through hints (e.g. [32, 34]), feedback (e.g. [16]) and curated examples (e.g. [43]). Working with an ITS can improve computing students' performance and decrease their programming time on problems, both when help is available and afterwards, outside of the tutor [10], leading to claims that ITSs can match the learning gains of human tutors [11].

However, researchers have noted that students' use of help features in ITSs is far from ideal, with students avoiding help when they need it, or abusing help to expediently solve the problem [1, 4], behaviors negatively correlated with learning [35]. ITSs for learning programming are not immune to these problems, with studies showing evidence of help avoidance and abuse [32, 33]. This leads us to ask, *what do we know about how and why students seek programming help from computer tutors – or from human tutors*? With a few exceptions [21, 40], little work has investigated help-seeking in the domain of computing. Existing models of help-seeking focus on general classroom learning [29, 30] or on *desired* help-seeking behavior in an ITS [2]. However, educational psychology results from other domains do not always apply to computing [27, 28], and we are interested in how and why students *actually* seek help, not just how they *should*. In this work, we investigate factors that influence novice programmers' help-seeking behavior and how this differs with human and computer tutors. We present a qualitative analysis of 15 students' interviews, in which they reflect on solving two programming problems: one with a human tutor and one with intelligent, computer-based help. We discuss design implications and hypotheses that arise from these results.

1.1 Help-Seeking in the Classroom

Help-seeking can be defined as "the ability to solicit help when needed from a teacher, peer, textbook" or other information source [2]. Most models of how students seek help are rooted in original work by Nelson-Le Gall [29], who proposed a cognitive and behavioral model of help-seeking in school-aged children, consisting of five components: 1) Awareness of the need for help, 2) Decision to seek help, 3) Identification of potential helpers, 4) Employment of strategies to elicit help, 5) Reactions to help-seeking attempt(s). The

model was intended as a heuristic aid, rather than a formal theory, and focuses on *instrumental* help-seeking, in which the child is seeking only enough help to allow them to complete the problem on their own. Newman [30] expanded this model, in part by presenting step (2) as a comparison between a learner's self-efficacy level (SEL), a dynamic self-assessment of confidence and ability for a given problem or step, and their confidence tolerance level (CTL), the threshold of challenge and risk the student is willing to accept, inherent in the student's beliefs. Both authors stress that *dependence* (evidenced by requests for help) and *independence* are not mutually exclusive ends of a spectrum, but rather that students can use instrumental, adaptive help-seeking as a self-regulatory skill that maintains independent work [29, 30].

Researchers have also identified key factors that influence students' decision to seek or avoid help in the classroom. Students may avoid help for practical reasons: help may be inaccessible or inconvenient, and the use of help may be prohibited or go against social norms of the classroom [37]. Students may avoid help because of concerns about the help-giver's competence (e.g. if it is a peer), a desire for independence [41], or because of a perceived threat to competence [7]. Lower-performing students and students with lower self-esteem are more likely to feel a threat to competence by seeking help, and this threat can manifest in help avoidance or seeking expedient help (with the goal of finishing the problem), rather than instrumental help to foster learning [17]. Students with a *performance* achievement-goal orientation, focused on others' perception of their relative competence, are less likely to seek help than those with a *mastery* orientation, focused on learning and self-improvement. Help-seeking in the classroom is also a social experience, and students with a social status-goal orientation, focused on social visibility and prestige, may feel increased social costs of help-seeking and therefore avoid help [36, 37].

1.2 Help-Seeking with Computer-Based Help

While most research has focused on how students seek help in the classroom, a more recent effort has been made to understand how this work translates to the context of intelligent, computer-based help. Aleven and Koedinger [1] noted that help-usage in their PACT Geometry Tutor ITS was far from ideal. Students in the tutor requested help on only 22-29% of steps, and when they did request help, students skipped through all the levels of help to get to a "bottom-out" hint that explained the answer on 82-89% of steps. The authors addressed this by formulating their own model of ideal help-seeking behavior in a Cognitive Tutor [2]. This model drew on the previous models of Nelson-Le Gall and Newman, but situated itself in the context of step-by-step problem solving with on-demand computer help. Additionally, theirs was a model of *ideal* help-seeking, which noted where students might deviate from this model as "help-seeking errors." They used this model as the basis for the Help Tutor, a cognitive tutor which teaches the metacognitive skill of help-seeking as students solve problems in the Geometry Tutor. In an empirical evaluation, the authors found that help-seeking errors negatively correlated with domain learning, that the Help Tutor reduced the incidence of some of these errors, but that it had no impact on domain learning [35]. Mercier and Frederiksen [26] attempted to generalized the model put forward by Aleven et al.

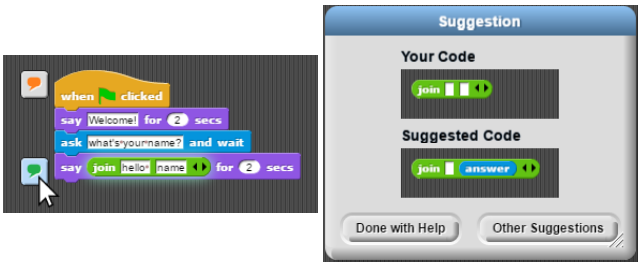

Figure 1: iSnap annotates code with hint buttons (left). When clicked, they suggest a change to the code (right).

to create a cognitive model of help-seeking in interactive learning environments, which added steps for diagnosing the problem and comprehending help.

Others have explored student factors that impact help-seeking with computer-based help [4]. Wood and Wood [45] found that students with lower prior knowledge were more likely to seek on-demand help and benefit from it in the Quadratic math learning environment. This contrasts with classroom research on help-seeking, which shows that lower-achievers (or those who perceive themselves to be) seek less help [37]. In a later study, Wood used a pretest to select challenging problems for each student [44]. In this case, prior knowledge did not predict help-seeking, suggesting that the original effect was due to subjective problem difficulty, rather than prior knowledge alone. Bartholomé et al. [5] studied pairs of students working in a "plant identification" learning environment with on-demand help and found that pairs with mixed (high and low) prior knowledge were more likely to seek help and less likely to produce errors, but they found no effect of pairs' "motivational orientation" or interest. Vaessen et al. [40] studied help-seeking in the context of a *computing* learning environment for Haskell with on-demand help. They clustered students' attempts at short programming problems based on log data to identify help-seeking strategies. They identified 5 strategies, and noted that students' achievement-goal orientations were predictive of 3 of the strategies. Research into help-seeking in ITSs appears to agree with some, but not all, results on classroom help-seeking, leaving open research questions and prompting calls for further research [3, 40].

2 ISNAP

This study investigates students' help-seeking behavior with human and computer tutors. For our computer tutor, we selected iSnap [32], an extension of Snap*!* [15], a block-based programming environment for novices. iSnap adds on-demand hints, which offer next-step suggestions to students on how to progress their code towards a solution. Unlike compiler messages, the hints are semantic, not syntactic (Snap*!* does not have syntax errors); they concern the structure of code and how closely it matches a correct solution. Hints are contextual, generated based on the student's current code and how it compares to previous students' solutions [31].

A student requests help in iSnap by clicking a button, which annotates their code with one or more hint buttons (see Figure 1). Hovering over a button highlights the code to which the hint applies. When a hint button is clicked, all buttons disappear and the

student is shown a next-step hint window. The hint generally suggests a single edit to the student's code, such as inserting, deleting or reordering a code element. The hints are presented visually, demonstrating the edit to make. These are similar to "bottom-out" hints, in that they tell the student exactly what to do, but they do omit some details, such as variable names and literal values. Students can continue to work with the hint window visible, or they can dismiss it to search for another hint that suits their needs.

3 METHOD

This work was motivated by two primary research questions:

RQ1 Why do novices seek and avoid help when programming?

RQ2 How is the process of help-seeking different when novices are working with human help and computer-based help?

We focus on novice programmers because that is the target population for many programming ITSs [16, 34], including iSnap. The analysis presented in this paper is part of a larger study to investigate help-seeking and help use in computing, the procedures of which are explained below. This analysis represents an initial attempt at understanding our data and formulating hypotheses that lay the groundwork to answer these research questions.

3.1 Population

We recruited 15 undergraduate students at a large U.S. research university to participate in our study. We recruited students who had completed or enrolled in an introductory programming course (AP CS, Java, Matlab or Python), but not a more advanced course. We made short, in-class announcements about the study, and we accepted the first 15 participants to volunteer. To encourage less eager programmers to participate, we compensated participants with a $20 gift card. Our participants included 4 females and 11 males, with most students identifying as White (11) or Asian (2), one as Hispanic/Latino and one as Other. Participants' majors were primarily CS (5) or other engineering fields (8), and 9 had taken or were enrolled in the introductory Java course for CS majors. None reported having used Snap*!* or Scratch more than a few times.

3.2 Procedure

Each participant completed the study[1] individually in a single session. Participants completed two programming assignments, one with a human tutor available and one with iSnap help available, and then participated in a semi-structured interview, which is the focus of our analysis. Participants each attended an individual research session with two researchers: an experimenter and a tutor. In this study, both tutors were female. The participant began by completing a short logic puzzle for 5 minutes to practice thinking aloud, with the tutor available for help if needed. The tutor then led the participant through a short tutorial on the core Snap*!* programming concepts the participant would need for approximately 20 minutes. To reduce bias, no indication was given whether the research team was involved with the development of Snap*!*, and the whole system, including the iSnap help features, was simply referred to as Snap*!*. The participants then worked on two programming tasks, each for up to 30 minutes. Students were prompted to think aloud while working on these tasks. Before starting, the participant was asked to read the assignment instructions out loud and to ask the tutor any questions they had *just on the instructions*.

For the first task, the participant was encouraged to ask the tutor for help if they needed it, or to work independently if not. For the second task, the tutor left the room, and the participant was similarly encouraged to ask iSnap for help if they needed it. Before starting, participants were shown a 3 minute video on the help features of iSnap. In both cases, if the student appeared to be struggling with the same issue for at least 45 seconds, the tutor (in the first task) or experimenter (in the second task) intervened by saying, "remember, you can ask me/snap for help." Assignments were selected from the first two units of the Beauty and Joy of Computing curriculum (bjc.edc.org) [15] to be appropriate for novice Snap*!* programmers. In the Guessing Game, the program should greet the player by name and then repeatedly ask them to guess a stored random number, telling them if it is too high, too low or correct. In the Brick Wall, students create custom blocks (procedures) to draw a brick wall with an arbitrary number of alternating rows. The order of the assignments was randomized, with 10 receiving the Guessing Game first with human tutor help. Because iSnap was not designed to answer questions about the Snap*!* programming interface itself, we ordered human tutor help first for all participants so that they had an opportunity to ask the tutor any additional interface questions during their first assignment.

After completing both assignments, the experimenter conducted an interview for 10-15 minutes, and the tutor remained out of the room. The experimenter used a semi-structured interview protocol to investigate the student's perceptions of the tutor's help, iSnap's help, comparisons between the two and any suggestions the participant had for iSnap. The questions explored why the student did or did not seek help, the quality of the help and whether the student trusted the help for both assignments. The experimenter also followed up on interesting points and inquired about specific behaviors observed during the programming exercises using notes. While future work will explore the various types of data collected (think aloud protocols, survey responses, finished programs), we focus in this work on the interview data, as it provides a good starting point for understanding participants' experiences.

3.3 Qualitative Analysis

We investigated our research questions using a qualitative analysis, based in Grounded Theory (GT). GT is a set of techniques for building theory that is "discovered, developed and provisionally verified through systematic data collection and analysis" [39, pp. 23]. These techniques, discussed in the next section, were appropriate because of the qualitative nature of our data, the open-endedness of our research questions and the lack of existing, satisfying theory addressing help-seeking in programming. GT has also been applied in CS Education research in a number of contexts [14, 19, 20, 23, 24]. We focused on the variants of GT detailed by Strauss and Corbin [12, 39] and Charmaz [8, 9], and drew on the detailed work of Kinnunen and Simon [18] to apply GT to CS Education.

GT is more than just a set of analysis techniques; it is an iterative approach to "gathering, synthesizing, analyzing and conceptualizing qualitative data" [8, pp. 82]. As in previous work [18], we

[1]Full study materials available at: http://go.ncsu.edu/icer2017-materials

primarily applied these techniques to data that had already been collected, so we recognize that our work borrows from GT but does not embody it. The analysis we present is an initial attempt to understand our data and research questions. Due to the limitations of our sample size and methods, our goal is not to definitively answer these questions, but rather to describe and interpret the phenomena we observed, discuss how our results relate to existing theory, and generate hypotheses. This is an expected first step in qualitative research, which can take many iterations to generate a full theory.

3.3.1 Coding. We began by performing line-by-line, open coding [8] on transcriptions of the interviews we collected. We started with two intentionally selected interviews, from P12 and P15, which the experimenters identified as containing rich content that directly addressed our research questions. The four researchers that conducted the study served as coders. They coded the interviews independently, with two coding each of the two chosen examples, generating a total of 191 independent codes. The researchers met to discuss these codes and collectively sorted them (written on cards) into conceptual groups. Each group of similar codes was refined and named to produce an initial set of 36 *focused codes*. Examples include "trust" (a participant's trust, or lack of trust, in the tutor or provided help) and "independence" (a participant's beliefs about independence and actions taken based on these beliefs).

One researcher compiled these focused codes into a codebook and wrote brief descriptions for the other researchers to reference. The codes were not tutor-specific (e.g. "trust in *human*" vs "trust in *iSnap*") and could be applied in reference to either the human or computer tutor. The referent ("human" or "iSnap") was noted when the code was applied to a segment. Three of the original researchers then performed open coding on the full set of 15 interviews, during which the codes were further refined for a final set of 43 focused codes. They coded a total of 713 segments (over 30,000 words), with an average 1.8 codes per segment. Since the primary purpose of coding was to help us organize and understand our data, not to quantify and reduce it, each interview had one primary coder, and we therefore did not calculate inter-rater reliability. Instead, as in previous qualitative studies, we worked to establish rigor by discussing and refining codes throughout the analysis [18, 24].

Near the end of coding, the researchers again physically sorted the codes, this time into broader conceptual *categories*. Drawing on the paradigm model of GT [39], we identified a central phenomenon of interest, "Help-seeking Behavior," and related other categories to that phenomenon as causal conditions, context, and consequences. For this paper, we narrowed our focus to factors *influencing* help-seeking behavior (rather than the help-seeking itself or its outcomes), which included 3 of the 8 categories we identified. We compiled, summarized and discussed the interview segments coded with each code in these categories, which we present in the next section. We applied constant comparison [39] throughout the process, writing extensive memos that contrasted segments and explored relationships among the codes and the categories.

4 RESULTS

Our results focus on three categories of codes and their relationship to students' help-seeking behavior. We briefly explain these categories and then discuss the most prominent codes from each:

Inputs are static factors that exist outside of the problem solving session, such as a student's previous experiences, the student's existing knowledge and beliefs, and attributes of the programming assignment. The tutor (or ITS designer) has no direct control over these factors, and they are unlikely to change in the course of problem solving. While they may not influence help-seeking behavior directly, Inputs set a baseline for the Student Mindset.

The **Student Mindset** is how a student interprets their relationship to the programming problem, the tutor and the help the tutor offers. Examples include perceptions of the tutor's trustworthiness, the accessibility of help and how stuck the student is. Unlike Inputs, the mindset is dynamic and can change throughout problem solving, especially but not exclusively when the student seeks help. A student's help-seeking behavior is a response to this mindset.

Attributes of Help are qualities of a tutor's help which determine its effectiveness and perceived value, such as specificity, interpretability and utility. Unlike codes in the Student Mindset category, these attributes can only be experienced by the student after seeking help from the tutor. They affect the outcome of receiving help, possibly leading to changes in the Student Mindset.

4.1 Inputs

4.1.1 Previous Experiences. Previous Experiences encompass students' past encounters with human or computer help and how these experiences shape their behavior during problem solving. Participants related the human tutor to past experiences with TAs and professors. When asked what it was like working with the human tutor, one participant said it was "*just like office hour[s]... I don't know how to do my homework, and I ask the TA for help*" [P2]. Another participant expressed how their previous experiences in the classroom made help-seeking unfamiliar: "*I'm also just not used to asking for help in [programming] classes*" [P15]. Participants discussed avoiding human help during the study due to negative past experiences receiving programming help: "*I've definitely experienced before where I had my- the design of my program in mind for a project... and then someone would be like, "oh no do it this way"... And I'm not saying [the tutor] did this. I'm just saying this is my experience that made me quick to dismiss her at first.*" [P11]. P11 rarely asked for help from the human tutor, even after explicit reminders. These previous experiences, appear to strongly influence students' later help-seeking behaviors.

Participants attempted to relate iSnap to previous experiences with computer help, compilers and IDEs. One participant noted how iSnap defied those expectations for the better: "*in Python or in Matlab you don't have little help buttons like that... I was expecting syntactic help... but [iSnap] gives you pretty exact advice... that was pretty cool* " [P15]. Another wished iSnap had behaved more aggressively like a compiler: "*in eclipse or any other IDE [if] you enter a... statement where you... have an argument that isn't compatible it will immediately flag that as red*" [P5]. Other participants explicitly linked previous experiences with computer help to their help-seeking behaviors. One participant's confidence rejecting one of iSnap's suggestions was "*not based on... my experience in this [study] but rather what I felt in my experience prior*" [P5]. Another participant never used iSnap's help, and also noted "*I don't like using Java help*" [P7]. We hypothesize that prior experiences

played an important role in shaping participants' expectations of the computer tutor, as discussed in Section 4.1.2.

4.1.2 Expectations of the Tutor. Expectations of the Tutor refers to a student's preconceptions of how a human or computer will provide help and how these expectations are confirmed or defied. Students had generally high expectations of the human tutor's knowledge and experience, assuming "*she's an expert*" [P1] and "*she understands it*" [P13]. These expectations were largely met during the study: "*she had the answer right away*" [P10].

In contrast, participants' expectations for iSnap were quite low. Many noted the fact that the help came from a computer as a specific source of low expectation, saying "*it's just a computer*" [P12], "*listening to a computer... is hard*" [P3], and more exact, perceptive help is "*not really something you should expect from little computer help buttons*" [P15]. For many participants, these low expectations produced a satisfactory experience, even for some who rarely used the help or described it as being generally unhelpful. iSnap was "*good enough*" [P1], and it "*does everything well enough*" [P4]. For others, however, these low expectations meant never using the help in the first place, as was the case for one who saw iSnap as helping with the "*basics of programming*" [P7]. As noted in Section 4.1.1, iSnap defied these low expectations in some cases.

4.1.3 Independence. Independence refers to students' beliefs and preferences about working without help. When asked what discouraged them from asking for help, nearly all participants expressed some form of the sentiment that they wanted to "*figure it out myself*" [P12], often using very similar language. They cited the belief that independent work leads to self-improvement or learning: "*it's more beneficial for me to just figure it out on my own*" [P11], or that asking for help lowers self-efficacy: "*if they ask for help, they feel like they can't do it themselves*" [P13]. One was willing to "*keep trying myself until I'm too frustrated to keep trying*" [P10], and another expressed an unwillingness to ask for help on math errors, "*something I should be responsible for*" [P5]. Two students mentioned that programming *in particular* may lend itself to independent work because "*you can just kinda puzzle your way through it and figure it out on your own*" [P15]. Some of these responses may reflect a selection bias in our participants, who willingly participated in a programming study and may be more motivated learners.

While participants also cited a desire to "*figure this out myself*" [P13] as a reason not to use computer-based help, it may not trigger the same threat to independence that human help does: "*I'd probably click on the little computer help things quicker than going to an actual tutor just because I'm not asking someone else for help [laughs] per say*" [P6]. Another participant got deep into debugging a problem without the human tutor's help, but said "*maybe if help indicators had been turned on for that program I would have gone back*" [P5]. However, independence may also lead students to invest less effort in understanding hints: "*I don't think I really gave it a good enough chance... I didn't actually over-analyze the structure [of the hint]... because I wanted to figure it out myself*" [P10].

4.2 Student Mindset

4.2.1 Trust. Trust refers to a student's confidence in the tutor's ability and the quality of their help. Participants clearly trusted the human tutor in our study, citing assumptions about the tutor's experience (see Section 4.1.2), the fact that the tutor was "*encouraging... friendly and approachable*" [P11] and provided affective support, the perceived quality of the advice itself, which "*made sense*" [P10], and the perceptiveness of the tutor (see Section 4.2.4). Trust did not always lead to help-seeking: "*I... trust that she has the answer, but I felt it did not really motivate me to ask [for help]*" [P1], indicating that it may be a necessary but not sufficient condition for help-seeking. Two participants' negative previous experiences with human tutors did lead to initial mistrust and few help requests (see Section 4.1.1), but when asked directly the same students still described the study's tutor as "*definitely*" [P11] trustworthy, again indicating that trust alone does not lead to help-seeking.

Participants generally expressed Trust in iSnap, though not as unanimously. When asked whether they trusted iSnap's help, one participant expressed wariness because a computer could not adapt to the "*many different ways that you can go about [solving the problem]... [iSnap] can't know everything that I've done*" [P12]. One trusted iSnap, but "*a little less than an actual tutor*" [P6]. Others expressed confidence that iSnap had more experience than they did with Snap*!* programming "*because its... Snap itself that gives me advice*" [P8]. Another found the directness of the advice trustworthy: "*of course I trust it because... it just tells you... what needs to be done*" [P7]. Trust may be less important if students feel confident evaluating help quality: "*if you're a programmer you should probably know... what advice is good and what advice is bad*" [P15].

There was a small breach of procedure for P1, in which the tutor remained in the room during the second programming exercise. After being prompted to use iSnap's help, P1 asked the tutor how the help was generated, and she explained. This changed P1's trust of iSnap: "*I only trusted... it after I asked [the tutor]... where is it getting this information and she said it's based on previous students*" [P1]. This may suggest that the human tutor "lent" iSnap some of her trustworthiness by explaining it, or that simply understanding the help mechanism makes it more trustworthy.

4.2.2 Stuck. A student is Stuck when they are no longer making progress on the problem. Many students noted this as a primary motivator for seeking help. This was true for both human help: "*whenever I got stuck and like literally didn't know what the next step was... then I was like, okay I need to ask for help*" [P15], as well as computer-based help: "*I was motivated to click on the hint bubbles when I couldn't figure it out on my own*" [P11]. For iSnap in particular, participants noted that the presence of the help was reassuring when stuck: "*it was kind of a good feeling that you knew you had that little bit that could help you if you were stuck*" [P12]. Some participants also expressed reluctance to ask for help, even when stuck, and in retrospect recognized this as a harmful behavior that "*comes back and bites me because... I really needed to ask a question, but I didn't*" [P13]. Participants described Stuck as a state one enters for a duration with degrees of intensity, rather than a single incident: "*sitting there thinking too long*" [P13] or "*just sitting there stuck*" [P15]. When this reaches a threshold, it can prompt help-seeking: "*then I was like okay, I need to ask for help*" [P15].

While participants were generally positive about the human tutor's role in getting them unstuck, they had mixed opinions on iSnap's ability to do so. Some mentioned its ability to nudge them

in the right direction, "*if you're stuck, kinda getting me on the right path*" [P12], or to help them solve the problem without wasting time: "*this kind of help is better on... programming efficiency*" [P15]. Two mentioned that iSnap should intervene in these situations: "*if you're having issues too many times, then maybe it can provide some suggestions*" [P13], while others thought this could be disruptive. Some participants noted that iSnap was useful even when they weren't stuck, as a means of "*confirming that I was doing the right thing*" [P15]. Others found that iSnap "*wasn't really helping*" [P3] them get unstuck, due to difficulty interpreting the hints or iSnap's lack of perceptiveness.

4.2.3 Accessibility and Salience. Accessibility refers to the perceived availability of the tutor and their help, and how convenient it is to access help[2]. Salience is often intertwined with Accessibility and refers to how noticeable the help is, or how present it is in the minds of students. Participants spoke of Accessibility and Salience almost exclusively in regards to iSnap, which had a "*convenient feature to ask for help*" [P14]. The help buttons were considered a "*low risk, high reward*" [P4] option because they "*were there, and it's really quick and easy just to click on one and see what it says*" [P12]. Many noted that iSnap's help buttons "*were kind of there*" [P4], subtly pervasive "*from the very start*" [P15], though one did recall forgetting about iSnap until prompted by the experimenter. While salient, the buttons were also appreciated for their subtlety; they were "*non-intrusive and non-intrusive; I think is a very good way of having it*" [P4]. iSnap's Accessibility and Salience were important factors in enabling help-seeking for some, especially when stakes were otherwise low: "*I figured I might as well at least see what it's trying to suggest*" [P4]. Participants noted for both tutors that they sometimes asked for help because the experimenter or tutor prompted them to (as described in Section 3.2), indicating that the help was not salient: "*I was motivated to click on the hint bubbles... when you reminded me I could have help if I wanted*" [P11].

Our human tutor's help was not described as *inaccessible*; however, when asked about useful aspects of the human tutor, participants rarely mentioned Accessibility or Salience. One participant did say that "*it wasn't really an issue having to ask [the tutor]*" [P9]. However, another participant did describe *previous experiences* with professors' inaccessibility, having "*to wait on... the professor to come help you*" [P15], and others mentioned that iSnap would be useful when humans were unavailable: "*the little automated stuff is helpful... especially if I'm doing something in my room*" [P6]. One noted that the inaccessibility of human help can be an intentional aspect of coursework: "*I wouldn't be allowed to access [help] because it's a project for school*" [P10].

4.2.4 Perceptiveness. Perceptiveness refers to the student's belief that a tutor is aware of the student's code, current objectives and possible mistakes, and can provide help based on this knowledge. There was consensus that the human tutor was highly perceptive. The human tutor knows "*what you're trying to do*" [P14], "*what your goal is*" [P7] and "*where I'm going wrong*" [P1]. The human was also perceived to understand the context of the programming assignment. Participants did not directly link the human tutor's perceptiveness to help-seeking, but it was cited as a reason to trust the tutor: "*[her help] was very reliable, since she's right here can see exactly what I have and what I've done*" [P12].

There was a general sentiment that iSnap was not perceptive. iSnap "*only can respond to what I'm putting on [the screen]*" [P11], and so it misses important visual and audio queues that the human would pick up, especially since the participant is thinking out loud. It was perceived as being unresponsive to changes in student code: "*It doesn't know when you make an error*" [P7] and it "*couldn't anticipate what I wanted so it couldn't really help*" [P14]. When asked what discouraged them from using iSnap, one participant explained, "*it doesn't know what I want to do so it can't tell me what to... do*" [P14]. Another responded that in the instructional video for iSnap, the speaker advised students to use their best judgement when receiving help, noting that one suggestion in the demo was not useful in this situation. This possibility for error made the participant "*feel like the computer doesn't understand*" [P1]. It is clear that a lack of perceptiveness directly dissuaded students from using iSnap. One student did describe iSnap as perceptive, "*I realize it knows what what I'm trying to get at*" [P3], but thought iSnap was unable to translate this knowledge into an interpretable hint. Another, who had a more positive experience with iSnap, experienced some perceptiveness: "*it looks at your code and then it tells what you should probably do*" [P15].

4.3 Attributes of Help

4.3.1 Specificity. Specificity refers to how precise a tutor's help was, whether it consisted of directly actionable suggestions or a higher-level outline. Participants appreciated the human tutor's ability to provide *both* exact advice: "*it was more precise... she would be able to say exactly what I needed*" [P12], and high-level advice: "*tutors have always been helpful... they don't just give you the answers... they actually try to walk you through how to find the answer*" [P6]. The former was appreciated for its efficiency and helpfulness, while the latter was appreciated because it allowed participants to retain a sense of independence and agency (see Section 4.1.3). As one participant noted, "*instead of straight-up just telling me where the commands were, she provided a path for me to think... I think that was helpful*" [P8]. While participants did not directly link Specificity to help-seeking or avoidance, it was tied to the perceived "helpfulness" of the tutor's advice.

Participants also perceived iSnap as providing a variety of types of help. Participants noted appreciating high-level advice: "*it was really good to get the outline of like 'oh this is what needs to be done, not what I have'*" [P12], as well as "*pretty exact advice*" [P15]. Others perceived the help to be so low-level that it was too basic, "*more syntactic than anything else*" [P11]. Participants who never used iSnap's help had mixed expectations of how specific it would be, inferring either that it helps with low-level things "*like the formatting*" [P7] or high-level things like the "*skeleton of the program*" [P9]. Since all participants viewed the same instructional video on iSnap's help, these diverse opinions may have been reflections of expectations based on previous experiences with computer help.

4.3.2 Interpretability. Interpretability refers to how easily the help provided by the tutor can be understood and applied by the student. Participants rarely discussed the interpretability of the human

[2]We are not referring here to accessibility as a system's ability to support users regardless of physical or mental ability

tutor, but when they did their comments were positive and terse. The tutor's help simply "*made sense*" [P10], and this influenced participants' trust of the tutor: "*I would trust in her opinion... whenever I followed what she did it made sense*" [P15]. Participants were much more critical of iSnap's interpretability: "*half the things you would click on didn't make sense for the program*" [P10]. A common reason was that iSnap offered no explanation for its suggestions: "*I found it unhelpful, mostly because I wasn't quite sure why it was offering the help that it was offering. It was just like, here's a suggestion- but why?*" [P4]. One participant followed the advice but found the results difficult to interpret: "*I did that and nothing different happened*" [P1]. Others chose not to follow the advice because of difficulty interpreting: "*I saw it, I just wasn't quite sure how to go about implementing it*" [P12].

While some students found the hints too difficult to interpret, others found the need for active interpretation to be worth the time it took: "*I would kind of have to interpret and figure out what it meant... If I was completely lost it would be a really good idea to figure out what it was*" [P12]. Participants recalled that hints that did not make sense at first could become more clear in retrospect after progressing further in the program: "*[it] gave me a hint as to where I need to go later, even though I didn't think it was applicable when I first saw it*" [P15]. One participant even realized, in the middle of explaining why a hint did not make sense, that its suggestion was in fact applicable but hard to understand in the moment: "*they wanted me to change it to an equals sign... um come to think of it I should have set that equal to something... it just wasn't like saying things clearly*" [P14]. Sometimes this interpretation can lead to a student making an unintended discovery. One participant added a custom length parameter to their "draw brick" block, a more advanced feature, based off of a misinterpreted, unrelated hint.

5 DISCUSSION

We now return to our research questions. To address RQ1, we identified key factors which influenced participants' willingness to seek help. Many of these correspond to factors identified in previous research, such the importance of Trust in a tutor's ability [41] or the Accessibility of help [37]. Independence, and the threat that help can present to it, are also well established reasons for help avoidance [7, 17]. Students' experiences of being Stuck leading to help-seeking fit nicely into Newman's dynamic self-efficacy level (SEL), which can prompt help-seeking if it falls below a threshold [30]. However, our results place more emphasis than existing work on students' Previous Experiences and Expectations, as well as their beliefs about the tutor's qualities, such as Perceptiveness. Our results show these factors have direct and indirect influence on help-seeking behavior. Additionally, while previous work mentions "evaluating help" as an important step in help-seeking [26, 29, 30], our results explore specific attributes of help, such as Specificity and Interpretability, that may be particularly relevant when evaluating the quality of computer-based help.

Our results also speak to how the domain of programming specifically impacts students' help-seeking behavior. Programming students are more likely to have had Previous Experiences with computer-based help (e.g. compilers and documentation) that shape their Expectations. Programming problems have many valid solutions and require students to make design choices, which may make it difficult to Trust help that contradicts those choices: "*that way would work but my way will also work too*" [P11]. That same open-endedness makes clear explanations particularly important when offering help, and participants noted that iSnap's lack of explanation hindered its Interpretability. Some participants believed that one can "*puzzle your way through*" [P15] programming problems with enough time and effort, which justified their decision to continue working Independently. Each of these attributes of programming present specific obstacles that should be addressed to provide effective, computer-based help in this domain.

5.1 Differences and Design Implications

In this section, we address RQ2, which deals with differences in how students experience help-seeking factors with human and computer tutors. We focus on how these differences have design implications for programming ITSs. While not every student needs help, many students do avoid help when they need it [4, 33], so it is important that ITSs are designed to facilitate help-seeking. Our methods and analysis were not intended as an evaluation of iSnap; rather, we use students' experiences with iSnap to highlight the challenges and opportunities that arise when providing computer-based, on-demand help to novice programmers.

Our study highlights some key challenges faced specifically by computer-based help. Unlike with human help, few students have experienced intelligent computer tutoring, which can lead to very low expectations for computer tutors. These expectations will shape how students interact with help, whether or not their preconceptions are accurate. Designers may be able to address this by explaining how their help system works, how it can be used effectively and how it differs from other help technologies (e.g. compilers, documentation). iSnap, for example, could explicitly state that its hints contain advice on achieving a correct solution based on previous students' work, which P1 noted as important in changing their perceptions and trust. Compared to human tutors, participants described trusting a computer as difficult, in part because of preconceptions about its inherent limitations. This was closely tied to a feeling that the computer was not, or could not be, perceptive or adaptive in the way that a human could. For help systems that rely on intelligent algorithms, it may be possible to mitigate this by subtly displaying what the help system *does* perceive and believe about the student's current state. iSnap could accomplish this by displaying the assignment objective it believes the student is currently pursuing, along with hints for this objective. In short, transparency and explanations of the computer's behavior may help to avoid confusion and unfounded mistrust.

Participants also noted some key advantages of computer-based help, describing it as efficient, salient and accessible. iSnap achieved efficiency in part by keeping help light-weight and visual, avoiding the need for "*3 paragraphs for each [hint]*" [P4], which one participant cautioned against. However, the simplicity of the helpfis presentation may have also led students to believe that the help was not very sophisticated (i.e. perceptive/intelligent). iSnap's accessibility and salience were enhanced by embedding the help buttons directly into the code, with help constantly visible and available.

iSnap also seemed to enable some students to seek help but still feel independent. This may be because computer-based help is private and does not risk impacting how others perceive one's competence or social status, as human help does [37]. Just as students may be more willing to seek help from peers than teachers [41], they may be even more willing to seek help from computers. Designers should craft help which minimizes the threat to students' independence and avoids unnecessary intervention, e.g. like "*Microsoft Word and Clippy... popping up*" [P4]. iSnap addressed this in part with the subtlety of its hints, but it could go further by adapting the quantity and specificity of hints to a student's ability level as other ITSs do [25, 45]. Overall, our results suggest that computer tutors are not simply a "worse" version of human tutors; rather, they offer distinct advantages and disadvantages, and may fill an important niche for students unable or unwilling to seek human help.

5.2 Hypotheses

In addition to to addressing our research questions, our results also prompted us to propose new hypotheses about how and why students seek programming help from humans and computers:

H1: The same factors shape students' help-seeking process with both human and computer tutors. We found that few of our initial line-by-line codes directly referenced the human or computer, and when they did, they were often pairs (e.g "human sees what I'm doing"; "iSnap doesn't know what I'm doing"). We therefore selected focused codes that were independent of the tutor itself (e.g. "Perceptiveness"). Our results suggest that the same factors (e.g., Independence, Trust, Accessibility) were relevant with human and computer tutors, but they may matter more for one type of tutor. Our presentation of results reflects this, as the description of each factor includes references to both human and computer tutors. For some factors, one type of tutor was referenced more frequently, perhaps indicating increased importance.

H2: Help-seeking is not simply a triggered response to difficulty; it can be a problem solving strategy. It is clear from our results that being Stuck, on its own, did not always lead to help-seeking, in part due to the participants' desire for Independence. However, participants also reported asking for help for reasons besides perceived difficulty, such as the Salience of help ("*buttons were kind of there*" [P4]), a desire to check one's work ("*confirming that I was doing the right thing*" [P15]), or expediency ("*I just want to get it right quickly*" [P7]). For these students, help was not simply a means of overcoming difficulty, but rather a tool to use in problem solving. We hypothesize that the extent to which students view help-seeking as a viable strategy is the product of a constant negotiation of many factors, which we call the Student Mindset. This contrasts with existing models of help-seeking, in which a help request always arises from a student's recognition of difficulty [26, 29, 30]. We believe that difficulty is a key factor in prompting a student to seek help, but it is neither necessary nor sufficient.

5.3 Limitations

This qualitative analysis of a small dataset has inherent limitations, and our results therefore lend themselves to recommendations and hypotheses rather than definite conclusions. This work is a necessary and expected first step in a larger research effort to understand help-seeking in computing. Our analysis required thoughtfulness and creativity on the part of the researchers, which means it is also a reflection of the researchers and their biases [39]. We attempted to minimize this by recognizing our biases, discussing results among the coders, and consistently supporting our claims with quotations.

The assignments we selected, especially the Guessing Game, were designed for novice programmers and may have been too easy to produce a genuine need for help in some of our more advanced participants, a few of whom reported having done a similar assignment in class. Participants were also new to Snap*!*, and some help requests centered on its interface, rather than programming generally, which may confound our results. Lastly, we have intentionally avoided a discussion of gender in this paper, as we felt that our dataset included too few females (4) to produce a meaningful comparison, but we acknowledge that it is also a major factor in help-seeking [4, 7].

6 CONCLUSION AND FUTURE WORK

In this work we have presented a qualitative analysis of 15 students' interviews after solving programming problems with both human and computer-based help. We identified 3 categories of factors that directly and indirectly impact students' help-seeking behavior: Inputs, Student Mindset, and Attributes of Help. Our results suggest that students have very different expectations of human and computer tutors, rooted in previous experiences. These Inputs set a baseline for how the student perceives the tutor and the role of help (the Student Mindset), which impacts their help-seeking behavior. If and when students experience help, the Attributes of Help may alter their perceptions of the tutor. We contrasted students' experiences of these factors to understand the differences in how human and computer tutors are perceived. For our participants, a human tutor seemed more trustworthy, perceptive and interpretable, while a computer tutor seemed more accessible and less threatening to a studentfis sense of independence. These initial results have important implications for how we provide help to students, especially through the design of ITSs for computing, as discussed in Section 5.1.

Future work will address aspects of the help-seeking process that we identified but did not discuss here: the ways students seek and avoid help, the types of help the tutor provides, the interface through which help is mediated, outcomes of receiving help and how the students' perceptions are updated after seeking or avoiding help. Our goal is to construct a preliminary model of help-seeking for both human and computer tutoring in the domain of programming, which can serve as a theoretical basis for designing more effective help systems. We plan to use the think-aloud and log data we collected while students programmed to help construct this model and address the hypotheses raised earlier.

7 ACKNOWLEDGEMENTS

The authors thank Jen Tsan for her invaluable help, which made this work possible. This material is based upon work supported by the National Science Foundation under grant 1623470.

REFERENCES

[1] Vincent Aleven and Kenneth R. Koedinger. 2000. Limitations of Student Control: Do Students Know When They Need Help?. In *Intelligent tutoring systems*. 292–303. DOI: http://dx.doi.org/10.1007/3-540-45108-0_33

[2] Vincent Aleven, Bruce M. McLaren, Ido Roll, and Kenneth R. Koedinger. 2006. Toward Meta-cognitive Tutoring: A Model of Help Seeking with a Cognitive Tutor. *International Journal of Artificial Intelligence in Education* 16, 2 (2006), 101–128. DOI: http://dx.doi.org/10.1.1.121.9138

[3] Vincent Aleven, Ido Roll, Bruce M. McLaren, and Kenneth R. Koedinger. 2016. Help Helps, But Only So Much: Research on Help Seeking with Intelligent Tutoring Systems. *International Journal of Artificial Intelligence in Education* 26, 1 (2016), 1–19. DOI: http://dx.doi.org/10.1007/s40593-015-0089-1

[4] Vincent Aleven, Elmar Stahl, Silke Schworm, Frank Fischer, and Raven Wallace. 2003. Help Seeking and Help Design in Interactive Learning Environments Vincent. *Review of Educational Research* 73, 3 (2003), 277–320.

[5] Tobias Bartholomé, Elmar Stahl, and Rainer Bromme. 2004. Help-Seeking in Interactive Learning Environments: Effectiveness of Help and Learner-Related Factors in a Dyadic Setting. In *Proceedings of the 6th international conference on Learning sciences*. 81–88.

[6] Benjamin S. Bloom. 1984. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* 13, 6 (1984), 4–16. DOI: http://dx.doi.org/10.3102/0013189X013006004

[7] Ruth Butler. 1998. Determinants of Help Seeking: Relations Between Perceived Reasons for Classroom Help-Avoidance and Help-Seeking Behaviors in an Experimental Context. *Journal of Educational Psychology* 90, 4 (1998), 630–643. DOI: http://dx.doi.org/10.1037/0022-0663.90.4.630

[8] Kathy Charmaz. 2008. Grounded Theory. In *Qualitative Psychology* (2 ed.), Johnathan A Smith (Ed.). SAGE Publications, Inc., 81–110.

[9] Kathy Charmaz and Linda Liska Belgrave. 2012. *Qualitative Interviewing and Grounded Theory Analysis*. SAGE Publications, Inc. 347–366 pages. DOI: http://dx.doi.org/10.4135/9781452218403

[10] Albert Corbett and John R. Anderson. 2001. Locus of Feedback Control in Computer-Based Tutoring: Impact on Learning Rate, Achievement and Attitudes. In *Proceedings of the SIGCHI Conference on Human Computer Interaction*. 245–252.

[11] Albert T. Corbett. 2001. Cognitive Computer Tutors: Solving the Two-Sigma Problem. In *Proceedings of the 8th International Conference on User Modeling*. 137–147. http://link.springer.com/chapter/10.1007/3-540-44566-8

[12] Juliet Corbin and Anselm Strauss. 2008. *Basics of Qualitative Research* (3 ed.). 379 pages. DOI: http://dx.doi.org/10.4135/9781452230153

[13] Paul Denny, Andrew Luxton-Reilly, and Dave Carpenter. 2014. Enhancing Syntax Error Messages Appears Ineffectual. In *Proceedings of the 19th ACM Conference on Innovation & Technology in Computer Science Education*. 273–278. DOI: http://dx.doi.org/10.1145/2591708.2591748

[14] Sue Fitzgerald, Beth Simon, and Lynda Thomas. 2005. Strategies that Students Use to Trace Code. *Proceedings of the 2005 International Workshop on Computing Education Research* (2005), 69–80. DOI: http://dx.doi.org/10.1145/1089786.1089793

[15] Dan Garcia, Brian Harvey, and Tiffany Barnes. 2015. The Beauty and Joy of Computing. *ACM Inroads* 6, 4 (2015), 71–79.

[16] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and L. Thomas van Binsbergen. 2016. Ask-Elle: an Adaptable Programming Tutor for Haskell Giving Automated Feedback. *International Journal of Artificial Intelligence in Education* 27, 1 (2016), 1–36.

[17] Stuart A. Karabenick. 2004. Perceived Achievement Goal Structure and College Student Help Seeking. *Journal of Educational Psychology* 96, 3 (2004), 569–581. DOI: http://dx.doi.org/10.1037/0022-0663.96.3.569

[18] Päivi Kinnunen and Beth Simon. 2010. Building Theory about Computing Education Phenomena: A Discussion of Grounded Theory. *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (2010), 37–42. DOI: http://dx.doi.org/10.1145/1930464.1930469

[19] Päivi Kinnunen and Beth Simon. 2010. Experiencing Programming Assignments in CS1: The Emotional Toll. *Icer'10* (2010), 77–85. DOI: http://dx.doi.org/10.1145/1839594.1839609

[20] Päivi Kinnunen and Beth Simon. 2011. CS Majors' Self-Efficacy Perceptions in CS1: Results in Light of Social Cognitive Theory. *Icer'11* (2011), 19–26. DOI: http://dx.doi.org/10.1145/2016911.2016917

[21] Daniel Knox and Sally Fincher. 2013. Where Students Go for Knowledge and What They Find There. *Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13* (2013), 35. DOI: http://dx.doi.org/10.1145/2493394.2493399

[22] Andrew J. Ko and Brad A. Myers. 2004. Designing the Whyline: a Debugging Interface for Asking Questions about Program Behavior. *Focus* 6, 1 (2004), 151–158. DOI: http://dx.doi.org/10.1145/985692.985712

[23] Colleen M. Lewis, Ruth E. Anderson, and Ken Yasuhara. 2016. "I Don't Code All Day": Fitting in Computer Science When the Stereotypes Don't Fit. *Proceedings of the 2016 ACM Conference on International Computing Education Research - ICER '16* (2016), 23–32. DOI: http://dx.doi.org/10.1145/2960310.2960332

[24] Colleen M. Lewis, Ken Yasuhara, and Ruth E. Anderson. 2011. Deciding to major in Computer Science: A grounded Theory of Sudents' Self-Assessment of Ability. In *Proceedings of the Seventh International Workshop on Computing Education Research*. 3–10. DOI: http://dx.doi.org/10.1145/2016911.2016915

[25] Rosemary Luckin and Benedict Du Boulay. 1999. Ecolab: The Development and Evaluation of a Vygotskian Design Framework. *International Journal of Artificial Intelligence in Education* 10 (1999), 198–220.

[26] Julien Mercier and Carl Frederiksen. 2008. The Structure of the Help-Seeking Process in Collaboratively Using a Computer Coach in Problem-Based Learning. *Computers and Education* 51, 1 (2008), 17–33. DOI: http://dx.doi.org/10.1016/j.compedu.2007.03.004

[27] Briana B. Morrison. 2015. Computer Science Is Different! Educational Psychology Experiments Do Not Reliably Replicate in Programming Domain. In *Proceedings of the International Computing Education Research (ICER) Conference*. 267–268.

[28] Briana B Morrison, Lauren E Margulieux, and Cherry Street. 2015. Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. In *International Computing Education Research Conference (ICER)*. 21–29. DOI: http://dx.doi.org/10.1145/2787622.2787733

[29] Sharon Nelson-Le Gall. 1981. Help-seeking: An Understudied Problem-Solving Skill in Children. *Developmental Review* 1, 3 (1981), 224–246. DOI: http://dx.doi.org/10.1016/0273-2297(81)90019-8

[30] Richard S. Newman. 1994. Adaptive Help Seeking: A Strategy of Self-Regulated Learning. In *Self-regulation of learning and performance: Issues and educational applications*. 283–301.

[31] Thomas W Price, Yihuan Dong, and Tiffany Barnes. 2016. Generating Data-driven Hints for Open-ended Programming. In *Proceedings of the International Conference on Educational Data Mining*.

[32] Thomas W. Price, Yihuan Dong, and Dragan Lipovac. 2017. iSnap: Towards Intelligent Tutoring in Novice Programming Environments. In *Proceedings of the ACM Technical Symposium on Computer Science Education*.

[33] Thomas W Price, Rui Zhi, and Tiffany Barnes. 2017. Hint Generation Under Uncertainty: The Effect of Hint Quality on Help-Seeking Behavior. In *Proceedings of the International Conference on Artificial Intelligence in Education*.

[34] Kelly Rivers and Kenneth R. Koedinger. 2015. Data-Driven Hint Generation in Vast Solution Spaces: a Self-Improving Python Programming Tutor. *International Journal of Artificial Intelligence in Education* 16, 1 (2015).

[35] Ido Roll, Vincent Aleven, Bruce M. McLaren, Eunjeong Ryu, Ryan SJD Baker, and Kenneth R. Koedinger. 2006. The Help Tutor: Does Metacognitive Feedback Improve Students' Help-Seeking Actions, Skills and Learning?. In *International Conference on Intelligent Tutoring Systems*. 360–369. DOI: http://dx.doi.org/10.1007/11774303_36

[36] Allison M. Ryan and Paul R. Pintrich. 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 (1997), 329–341. DOI: http://dx.doi.org/10.1037/0022-0663.89.2.329

[37] Allison M. Ryan, Paul R. Pintrich, and Carol Midgley. 2001. Avoiding Seeking Help in the Classroom: Who and Why? *Educational Psychology Review* 13, 2 (2001), 93–114. DOI: http://dx.doi.org/10.1023/A:1009013420053

[38] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A Review of Generic Program Visualization Systems for Introductory Programming Education. *ACM Transactions on Computing Education* 13, 4 (2013), 15.1 – 15.64. DOI: http://dx.doi.org/10.1145/2490822

[39] Anselm Strauss and Juliet Corbin. 1990. Basics of Qualitative Research: Grounded Theory Procedure and Techniques. *Qualitative Sociology* 13, 1 (1990), 3–21.

[40] Bram E. Vaessen, Frans J. Prins, and Johan Jeuring. 2014. University Students' Achievement Goals and Help-Seeking Strategies in an Intelligent Tutoring System. *Computers and Education* 72 (2014), 196–208.

[41] Hans Van der Meij. 1988. Constraints on Question Asking in Classrooms. *Journal of Educational Psychology* 80, 3 (1988), 401–405. DOI: http://dx.doi.org/10.1037/0022-0663.80.3.401

[42] Kurt Vanlehn. 2006. The Behavior of Tutoring Systems. *International Journal of Artificial Intelligence in Education* 16, 3 (2006), 227–265.

[43] Gerhard Weber and Peter Brusilovsky. 2001. ELM-ART: An Adaptive Versatile System for Web-based Instruction. *International Journal of Artificial Intelligence in Education* 12, 4 (2001), 351–384. DOI: http://dx.doi.org/10.1.1.66.6245

[44] David Wood. 2001. Scaffolding, contingent tutoring and computer-supported learning. *International Journal of Artificial Intelligence in Education* 12 (2001), 280–292.

[45] H. Wood and D. Wood. 1999. Help seeking, learning and contingent tutoring. *Computers & Education* 33, 2-3 (1999), 153–169. DOI: http://dx.doi.org/10.1016/S0360-1315(99)00030-5