What Help Do Students Seek in TA Office Hours?

Yanyan Ren
Computer Science Department
Brown University
Providence, Rhode Island, USA
yren17@cs.brown.edu

Shriram Krishnamurthi
Computer Science Department
Brown University
Providence, Rhode Island, USA
sk@cs.brown.edu

Kathi Fisler
Computer Science Department
Brown University
Providence, Rhode Island, USA
kfisler@cs.brown.edu

ABSTRACT

In many universities, Teaching Assistants (TAs) are an important part of students' educational experience. This is especially true in early courses, where students may suffer from inexperience and anxiety, and find fellow students more accessible than professors.

Despite its importance, this learning channel has not been studied very much. Part of the difficulty lies in how to meaningfully evaluate it. Any intervention needs to be both unintrusive and lightweight, and yet yield useful data. As a result, to many faculty and researchers, TA office hours remain fairly opaque.

This paper presents one approach to studying the technical component (but not the social dynamics) of TA office hours. We use a program-design methodology as a device to help track what students are asking about in hours, using a simple survey-based method to gather data. Data from TAs effectively summarize students' questions. In addition, contrasting data from both TAs and students provides insight into students' progress on program design help-seeking over the course of the semester.

KEYWORDS

Teaching Assistants, student self-assessment, program design method

ACM Reference Format:
Yanyan Ren, Shriram Krishnamurthi, and Kathi Fisler. 2019. What Help Do Students Seek in TA Office Hours?. In *International Computing Education Research Conference (ICER '19), August 12–14, 2019, Toronto, Ontario, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3291279.3339418

1 INTRODUCTION

Teaching Assistants (TAs) who hold office hours (where they answer student questions in person) are a valuable part of the learning experience. They enable a course to provide many more contact hours than an instructor alone can provide. Because TAs are more of a peer group than instructors, students may find them less intimidating to approach. They save students the embarrassment of having to admit their weaknesses to the people who will be assigning grades. They are also likely to know more about the experience of doing the same tasks than the instructors, and hence be more empathetic. Finally, they offer a scalable source of help for students.

In this paper we focus on TA office hours (henceforth just "TA hours") in programming-rich introductory undergraduate collegiate courses (details are in § 5.1). What happens in such courses? Some conversations are of a personal nature (e.g., help navigating the university, dorm life, etc.), which we do not focus on in this paper. The bulk, however, are technical questions that range from understanding a problem statement to checking proposed solutions. Thus, TA hours are a vital educational component, where students not only get help with their proximate problem but hopefully also develop better problem-solving skills. But what do students need help with, and how much progress do they make?

Developing a meaningful instrument is not straightforward. One natural approach would be to record all student-TA interactions, and analyze the speech and video later. We reject this because we believe knowing they are being recorded would make students far less less likely to ask (what they consider) "dumb" questions, or questions of a more personal nature (e.g., how to navigate the system as an underrepresented student), or questions about the instructors—it would be like taping a confessional. Surveillance would destroy the friendly, peer nature of hours.

As an alternative, we considered having TAs write a summary at the end of their interaction with the student. This way, students wouldn't feel the pressure of being judged. However, this also has two problems. First, it doesn't help us understand the use of hours from the student's perspective. Second, writing such a summary takes time, and with students already suffering from long TA lines, this would either cause an unconscionable delay (materially hurting the educational experience, and reducing ecological validity) or simply nudge TAs to write sloppy or skimpy documents.

In short, any method to study TA hours should have two characteristics. First, it should be as unintrusive as possible. Second, it should be lightweight, not requiring too much effort for either TAs or students. Ideally, it should merge seamlessly with processes already being used for TA hour management. It would be nice if the process also meshed with the course's pedagogic methods.

In this paper we present a method that meets these characteristics. Our method, which we introduce in § 4, centers around a specific program design methodology called the Design Recipe (DR). We present the methodology and the theoretical framing that inspires it (§ 2), and also situate this work amongst other efforts to analyze TA hours (§ 3). We report (§ 6) on a study of using this method in an introductory class (§ 5). Concretely, we focus on the following research questions:

RQ 1. *Is the* DR *useful for tracking what students are asking about in* TA *hours?*

RQ 2. *Do students ask for help with the right steps in the* DR*, and how does that evolve over time?*

(1) **Data Definitions** Understand the information provided by the problem and form data definitions to represent it as data.
(2) **Signature, Purpose Statement, Header** State a name, purpose, and type—in terms of what it consumes and produces—for the function (or program) being defined.
(3) **Functional Examples** Work through examples that illustrate the function's purpose.
(4) **Function Template** Translate the data definitions into an outline of the function.
(5) **Function Definition** Fill in the gaps in the function template, exploiting the purpose statement and the examples.
(6) **Testing** Make sure the function passes the tests, and construct additional tests as needed based on the implementation details.

Figure 1: The Design Recipe (DR) [7], paraphrased

The contributions of this work are (1) our process for using a design methodology to track what happens in TA hours, (2) our assessment of the DR as the specific methodology in such a process, and (3) evidence of how students' help-seeking evolves over the course of a semester.

2 THEORETICAL FRAMEWORK

Our work is based on the Design Recipe (DR) introduced in *How to Design Programs* (HTDP) [7]. The DR suggests to students that they break down their problem-solving into six steps, which are summarized in fig. 1. Though HTDP uses Racket, the DR is not language-specific and has been used for a variety of languages (including a book expositing its use for Java [8]).

The DR is grounded in multiple theoretical foundations. At a high level, its steps provide a form of scaffolding [3] designed to lead a student from a prose-based problem statement to a working program. The scaffolding steps ask students to produce intermediate artifacts (signature/purpose, examples, code template) that capture the problem at multiple levels of detail and abstraction. The progression from data definitions to examples to code move the student through different representations of the problem, providing a form of concreteness fading [11] as students progress towards a symbolic-form solution to a problem.

Completed sequences of DR steps form worked examples [21] that students can leverage when considering new problems. A student might refer to a DR example when writing a new program on an already-studied datatype definition: this would focus on the examples, templates, and code features of the example. When asked to work with a new datatype, the DR suggests higher-level steps that a student can follow to make progress on the problem.

Templates are a form of program schema [16, 20] that students can recall and reuse in constructing solutions to new problems. The LISP Tutor [1] builds on a theory that students can recognize and adapt solutions to recursive problems, though without an explicit step of articulating the template independently from the code. The template, in contrast, provides an explicit scaffold that handles traversing an entire data structure as part of implementing a solution to a specific problem.

Several papers have begun to explore the impact of the DR on students in different contexts. Fisler and colleagues on multiple projects [9, 10] showed that HTDP-trained students made more progress and fewer programming errors than students trained in more conventional curricula. Schanzer et al. [17, 18] have found improvements in middle- and high-school students' abilities to solve algebra word problems after working with a version of the DR.

The DR offers a valuable framework for our research. Its steps provide a vocabulary for students to express the help they seek and for TAs to indicate the help they offered. Because of its diagnostic properties, this vocabulary potentially has a high likelihood of being useful for students during TA hours.

3 RELATED WORK

Previous studies on TAs in CS courses have analyzed TA contributions to student success in introductory courses [24], training and supporting undergraduate TAs [14, 22], and improving accessibility and effectiveness of TA hours [4]. However, only a few (discussed below) have examined help-seeking activities and student-TA interactions. The broader education literature also studies TA development and TA-initiated interactions, but these are not relevant to our study.

Patitsas & Belleville [15] asked students to evaluate each TA on five criteria (paraphrased): preparation, helpfulness, consideration, understandability, and effectiveness. As they acknowledge, this provides only a very coarse view of the TA hours. Since the survey was handed out at the end of the semester, the student may not remember every encounter with a certain TA. A TA's performance may also vary throughout the semester. Finally, it evaluates the TAs but provides little insight into the use of hours.

Vellukunnel et al. [23] analyzed students' posts on Piazza, an online forum where students post questions and TAs answer. They labeled questions using literature on question-asking and learning theories. Their classification scheme has four types: Active, Constructive, Logistical, and Content-Clarification. Constructive questions "reflect students' reasoning or attempts to construct a solution to the problem", and comprise the largest portion of the total questions asked. The authors found that Constructive problem-solving activities are positively correlated with students' course grades. However, their classification does not give detailed insight into this category, which is precisely what the DR elaborates on.

Smith et. al. developed and analyzed data from My Digital Hand, a one-to-one peer teaching tracking tool [19]. The authors focused on service analytics. When signing up, students were prompted to describe their problem, but the authors offer only a brief comment on the kind of questions students asked.

While these studies provide different ways to evaluate TA hours, none of them has asked students to use an existing problem solving strategy like the DR to structure their questions before interacting with the TAs. We believe that there are several potential benefits to involving students' self-evaluation. Instead of hearing about the DR in lecture *passively*, students get to *actively* apply their existing

(1) DR step 1, from problem analysis to data definitions
(2) DR step 2, signature, purpose statement, header
(3) DR step 3, examples
(4) DR step 4, template
(5) DR step 5, function body
(6) DR step 6, test cases
(7) Grading of a past homework
(8) Understanding a past homework
(9) Programming language
(10) Complexity analysis for this homework
(11) General question on complexity analysis
(12) Tools
(13) Non-course issues
(14) Other

Figure 2: Major part of TA form (additional parts described in § 4.1)

knowledge of the DR every time they encounter a programming problem, which is a more effective kind of learning activity [5]. Letting students categorize their problem involves metacognition [6], which is an important aspect of programming problem solving [12]. Because the DR is a linear process that eventually leads to solutions, with checks built-in, it corresponds to three steps (Planning, Process Monitoring, and Comprehension Monitoring) in Loska & Ko's description of five types of self-regulation activities in the context of programming [13]. Their study found that self-regulation is critical to students' success in introductory programming classes.

4 THE EVALUATION TOOL

Driven by our operating constraints (§ 1) and guiding theory (§ 2), we constructed a tool to assess the use of TA hours.

Our goal is to learn *what happened* at hours. That requires two things: looking back at the interaction while it's still fresh in memory (*happened*), and expertise to determine how the time was used (*what*). We take as axiomatic that TAs are in a position to judge how the hours were used, which we discuss further in § 5.1.

4.1 A Form for TAs

Our central premise is that the DR provides a strong framework for assessing what happens at hours. While it cannot indicate exactly *which* questions were asked, it can potentially describe what the questions were *about*: namely, the steps of the DR provide a "coding manual" to bin the questions. The question remains whether these are useful bins (§ 6).

To this end, we created an "exit survey" that a TA fills in after every student meeting. TAs didn't design the form, but they had used the DR previously as students and as graders, and provided feedback on the form interface and distribution process (§ 4.4). Of course, the DR steps are not sufficient on their own. Students may have many other questions, ranging from programming language difficulties to big-O complexity to issues with past homeworks to seeking peer advice, and more. Therefore, our full rubric is much richer, and shown in fig. 2. (This represents the final state of the form. Not all options were present from the beginning: for instance, the options about past homeworks were added after the first homework, while complexity questions were added after the topic (big-O time analysis) was introduced in class and in homeworks. Nevertheless, all the DR questions, and the other items, were present throughout.)

The form allows TAs to select multiple entries. "Other" lets TAs record discussions that did not fall within this rubric; we regularly reviewed these to see whether we should add items to the form (which is how some other entries came about).

It is also important to collect the right information about these categories. Sometimes a student may come in completely lost and with no idea on how to proceed; in other cases a student may think they have solved a step correctly, and may merely want the TA to check their work. To lump these two extremes in the same bin would lose too much information and confound effective analysis. Instead, corresponding to each box, the TAs were given a three-point (multiple-choice) *classification* scale to capture the kind of help the student needed for that entry:

- instruction ("how do I do ...")
- clarification ("what is expected for ...")
- verification ("could you check my work on ...")

(There was also "Other", which was virtually never used and hence is ignored here.) The last question in the TA exit survey asks how the student is performing on this assignment, and the TA selects a score on the scale of 1 to 7, representing totally lost to perfect. Later analysis showed a low [2] coefficient of variation (barely touching 30%). Therefore, we do not discuss this further in the paper.

TAs reported that once they had gotten used to the form (which took a couple of weeks), it took them less than 20 seconds per student, which confirms that this is a lightweight intervention.

We consider this form to be a key contribution of this paper. It provides a preliminary attempt at an instrument for the instructors to learn what happens in TA hours, and the findings of this paper show that it has utility and provides insight. Of course, the form does suffer from threats of validity and especially generalizability, which we discuss in § 7.

4.2 The Student Form

In principle, the TA form provides all the information needed for RQ 1; it also lets us get some sense of how students are progressing. However, it does not help us understand how the student's own self-assessment progressed (RQ 2), nor does it force reflection.

We gave students a variant of the TA form. Specifically, we used the same questions as in fig. 2. However, students were not given the three-point classification scale; instead, they simply checked off all items they wanted to discuss.

We considered asking students to fill in these same classifiers, but decided not to for two reasons. First, we wanted to minimize the time spent filling out the form, and could not be sure how much reflection it would take for students to fill it out (especially if they feared doing so inaccurately). Second, we were greatly concerned about the effect on student morale—and hence, ultimately, their self-efficacy—if they had to fill out the weaker classifications. Since we could not guarantee it would not have an impact, we chose to not ask for this information to avoid harm.

How did we get students to fill out this form? Fortunately, the classes at this institution already have an electronic queueing system (which automatically records their identity) for students to get time with TAs. We leveraged this, replacing the existing sign-up system with this form, which also automatically gathers their identity. Since students were not required to check any boxes, in principle signing up would take no more effort. Because the form was implemented as a Google Form (inside a university system with student data protection), we could not determine how long students spent filling the form, though we also did not receive any complaints about it.

Of course, the time spent includes not only the act of filling the form, but also thinking about what to fill in. If students were already using the DR this time should have been minimal, while if they were not, it may have taken some effort. Under our theory of instruction, however, this was time we would consider productively spent and one that might enhance the quality of their TA meeting.

4.3 Terminology

Each form response is called an *entry*. A student entry can be complemented by a TA entry or stand alone. Each item corresponding to a check box is called an *option*.

4.4 The Process

Students fill out their form when they sign up for a slot. The act of submitting it creates a new entry in a dashboard for the TAs, who then call on the next student in the queue. When that student is done, the TA clicks on the student's entry in the dashboard. This brings up the TA form, *pre-populated* with the student's entries; the TA must still enter at least their classification alongside.

The reason for pre-populating it was to reduce time: if the TA agreed with the student, it would take less time to finish the form. Of course, a lazy TA could simply choose the same options that the student did. However, this would still require conscious action to pick the classification. Laziness would also manifest as no difference between the student and TA; the data (§ 6.2) show this was not the case.

The form is filled by the TA after the student has left, and the result is not shared with the student, so that the student does not feel the weight of judgment. In principle, a TA could click on the student's entry at any time, including before the student's session began, thereby seeing the student's choices. However, we requested TAs to not do this, and they did not argue that this restriction was onerous. Furthermore, they confirmed that they usually followed this directive.

4.5 Variance Amongst TAs

We use TA entries to reflect "the actual question that student should have asked". (We trust the TAs' judgement for reasons discussed in § 5.1 and § 7.) However, this doesn't address the issue of individual TA bias: maybe some TA always thinks that the students are doing worse than they actually are, or vice versa.

Each of the 12 TAs held a two-hour help session every week. Students were free to go to as many hours and ask as many questions as they wanted. This approach makes comparisons across students and the general trends observed on the class level more valid, since student are not tied to any particular TA. But this also makes variation within each student harder to interpret, since we cannot discern between TA judgment and true student progress.

The course staff put effort into reducing variation among TAs. When grading homework, 10% of the submissions were graded by two TAs. Disagreements resulted in discussion until reaching an agreement. In effect, the goal was to increase inter-coder reliability amongst the TA staff. This conformity would, hopefully, also carry over to assessing hours.

To check whether we have any consistently biased TAs, we assigned a distance score for each student-TA answer pair (see § 6.2.2 for details). We first calculated the average of distance score of each TA, and then compared these average scores against each other. No outlier was found, based on 1.5 Interquartile Range Rule for Outliers.

5 DATA FROM TA HOURS

We first provide raw data about the TA hours that we studied. The analysis follows in § 6.

5.1 Course Context

The study was conducted a highly selective private university in the USA. The course is an accelerated introductory course that compresses much of the first year's content (programming through lists, trees, and graphs; graph algorithms up to Dijkstra's and minimum spanning trees; big-O analysis; and a brief introduction to other programming techniques such as laziness) into a semester. We discuss individual assignments as relevant later in the paper. 67 students, almost all first year, finished. 48 earned an A (highest), 12 a B, 6 a C (lowest passing grade), and one failed.

Students placed into the class through multiple summer tasks. The tasks were self-contained and began with basic programming (following HTDP), and were therefore open to students who had never programmed before. The grading of the placement tasks emphasized adherence to the steps of the DR, and students asked several clarifying questions about it on Piazza. A small number of students with no prior programming experience did place into the course, but most had some or extensive prior experience. However, while their background was primarily in Java, Python, Scratch, and AppInventor, the course used Pyret; the first two-thirds was purely functional, while the last third introduced and used side-effects.

All 12 TAs were undergraduates who had taken the course a year or (rarely) two earlier. They were chosen from a competitive process (fewer than a third of applicants were hired) that took into account knowledge, ability to run mock help sessions, perceived empathy, etc. We further discuss the use of undergraduates in § 7.

The course had 12 assignments on which students were permitted to seek TA help (three others were in "exam" mode). Two were done in pairs and triples; we cannot interpret student entries for these (did students come alone or in pairs, and did their partner agree?), so we exclude these in § 6. The remaining assignments, which we label P1 through P10, were out one at a time, so the timestamp effectively identifies the assignment.

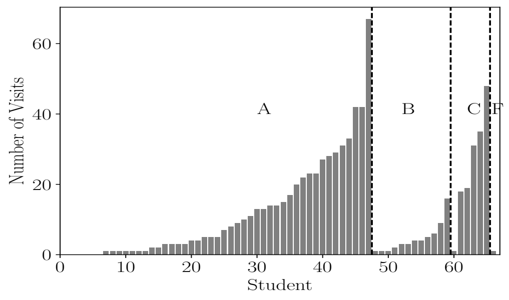


Figure 3: Total visits per student over the semester ($\mathcal{A}$)

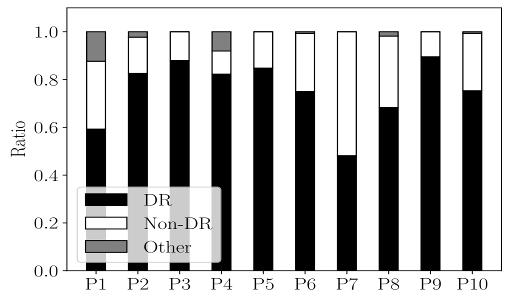


Figure 5: DR versus non-DR versus "Other" questions according to the TA forms ($\mathcal{B}$)

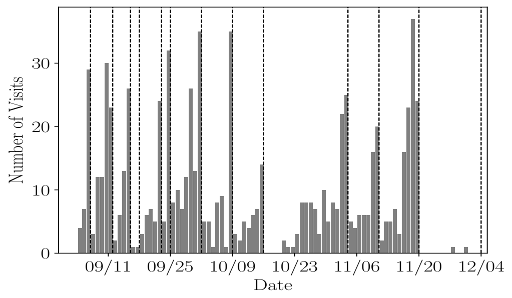


Figure 4: Number of visits on each day ($\mathcal{A}$)

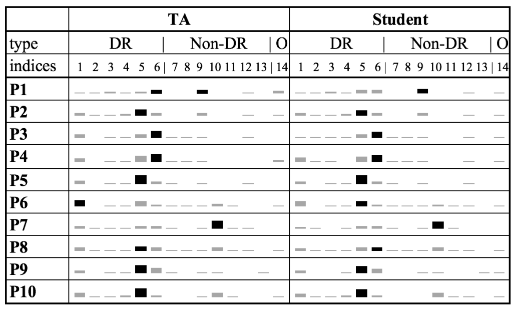
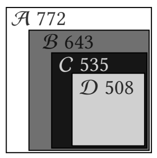


Figure 6: Ratio of options selected by TAs and students ($\mathcal{B}$). Indices are from fig. 2; O represents "Other" questions

5.2 The Scale of the Data

60 (90%) of students who completed the course visited TA hours at some point over the semester (a few more visited TAs but dropped the course very early). The 7 students who completed but never visited all earned an A. The median/mean number of visits for students who earned As and who used hours was 5/11.75; for B, 3.5/4.58; and for C, 25/25.33. Thus, even many A-earning students made good use of hours.



We obtained 772 student entries (we label this dataset $\mathcal{A}$). Limiting to the ten solo, assisted assignments, we have 727 student entries, 643 of which had a corresponding TA entry ($\mathcal{B}$). 524 of these student entries picked a DR step ($\mathcal{S}$), while 535 of TA entries did so (some TAs recorded a DR step even though the student did not) ($\mathcal{C}$). 508 paired entries ($\mathcal{D}$) both picked a DR step. The figure captions indicate which dataset was used for that figure.

Why are there missing TA entries? At times of peak student rush, TAs were eager to get to the next student. Some TAs also had form difficulties early in the semester. Most, however, are because students did not show up: they sometimes signed up for multiple slots (due to long lines) and did not use all of them.

Figure 3 is a sorted histogram of student visit count, with the dashed line binning by grade (A, B, C, fail). Figure 4 shows the number of visits on each day; dashed lines are the day an assignment was due (just before midnight). There are two gaps: Oct 17 to Oct 20 corresponds to a break when no assignment was out; Nov 21 onward represents Thanksgiving break, an assignment in "exam mode", and the course's end.

6 DATA ANALYSIS

We answer the research questions by analyzing data from the two forms: first from TAs, and later combined with student inputs.

6.1 What We Learn From TA Forms

6.1.1 Do enough entries get covered by the DR *to make it useful?* It is possible that students' questions don't fit into any DR step, and students struggle with some completely different problem areas not covered by the DR.

Figure 5 shows the breakdown of DR, non-DR, and "Other" questions for each assignment. We see that DR questions are the majority (average 75.31%), except for P7, where non-DR questions account for 51.85%. P7 was an outlier where students were asked to do a new kind of complexity analysis (for lazy programming), so it is unsurprising that option 10 (complexity analysis) was the most popular option. Overall, this lets us conclude that the DR provides at least reasonable coverage of student help issues.

6.1.2 What's the per-assignment distribution of entries? In each cell of fig. 6 is a bar chart of the rate of each option selected by TAs (and by students). The dark bar in each cell represents the most popular option picked for that assignment.

From the TA columns, we see that DR steps 5 and 6 are most popular in most assignments. This is not surprising because the function body and test cases are the main "working" deliverables.

DR step 3, examples, seems to get little attention. We conjecture this is because we gave students an automated tool, Examplar [25], that checked their examples/tests for both correctness and coverage, eliminating most TA help. On assignments (P3, P4, and P9) where this tool was not provided, we see (sometimes big) spikes in DR step 6, testing (which may be conflated with examples). For P8, a complex assignment (campus tours built atop Dijkstra's algorithm using Manhattan distance), students unsurprisingly needed testing help despite the tool.

DR step 1, from problem analysis to data definitions, was also a popular DR step picked by TAs. It was not picked much at the beginning of the semester, probably because the assignments prompts were relatively easy to understand, and students didn't need to design or use complex data definitions. For assignment P6, students were required to define their own sophisticated data structure, and the most popular option was step 1 (31.94%). As the assignments became more complicated, step 1 became more popular. *Step 1's popularity shows that TAs helped students with understanding the problem, not only focusing on the end products.*

There are two assignments where the most popular option is not a DR step. In P1, where a new language was just introduced, the most popular option was unsurprisingly about the programming language (20.99%). In P7, the most popular option was complexity analysis (41.98%), as already discussed in § 6.1.1. This option is also popular in other problems whose submission required sophisticated complexity reasoning (P6, P8, and P10).

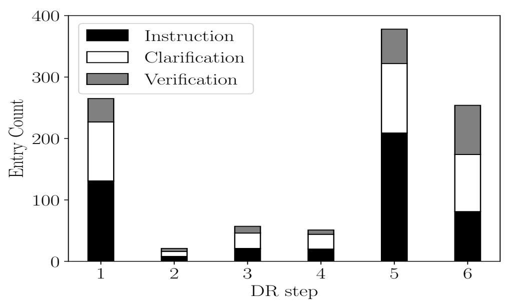


Figure 7: Number of entries and classifications of questions breakdown for each DR step (C)

6.1.3 What's the distribution of entries across DR steps? Each bar in fig. 7 represents the overall entry count for the corresponding DR step. We have discussed the low use of step 3 above. DR step 2 also gets little attention, because this is usually given in the problem statement. Likewise, DR step 4 is useful for true beginners, but the scaffolding is far less useful at this level of task and as students progress through the course.

Figure 7 also shows a breakdown of the help classification corresponding to each DR step. We conjectured that students need more higher-level help (clarification and verification) for earlier steps (step 1, 2, 3) that involves understanding the problem and planning out the solution, and lower-level help (instruction) for later steps

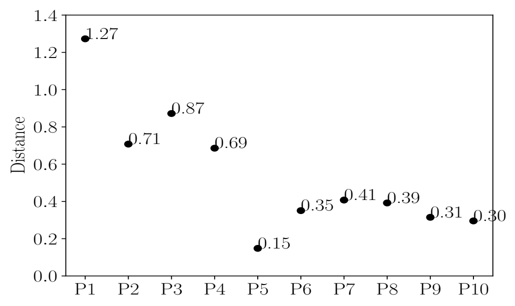


Figure 8: Average Hamming Distance between student and TA DR picks ($\mathcal{D}$)

(step 4, 5, 6) that involves the implementation details of the code and test suites. With a null hypothesis that the step (early versus late) and type of help (low- versus high-level) are independent, we ran a χ^2 test (contingency table shown below).

	High	Low
Early	183	160
Late	373	310

We obtained a p-value of 0.75, which is greater than the default alpha value of 0.05, indicating that we failed to reject the null hypothesis. This suggests that the type of help is independent of the step.

6.1.4 Is the DR useful for tracking what students are asking about in TA hours? We answer RQ 1 in the affirmative: yes, the DR is a useful metric for categorizing questions during TA hours, since a majority of the questions are about some DR step (§ 6.1.1). DR steps are at a useful if not ideal level of granularity, because the most asked about DR step reflects different features in different assignments (§ 6.1.2), and the frequency of questions about each step makes sense in the course's context (§ 6.1.3). Keeping in mind the time and effort constraints discussed previously, the DR could be refined to ask sub-questions in the popular entries (perhaps including problem-specific ones) to learn more about the help students seek.

6.2 What We Learn From Both Forms

The student form encourages students to reflect and use the DR steps, and lets us assess whether students ascribed steps more accurately as the semester progresses (RQ 2).

6.2.1 Are the data meaningful? A first question is whether students even took the form seriously: perhaps they checked boxes at random to quickly enter the TA queue. In fig. 6, the pattern in each cell in the student column is similar to that in the TA column. Thus, we can conclude that students are not picking at random, but making sensible choices that reflect different features in the assignments, based on our analysis in § 6.1.2.

6.2.2 Are students choosing correct DR steps? (RQ 2). The TA and student DR entries form an equal-length bit-vector, so it is natural to compute each pair's Hamming distance. (For most other options, it makes far less sense to ask if students chose "correctly".) Figure 8 shows the average distance for each assignment. The distance gradually decreases over the course of the semester (with the exception

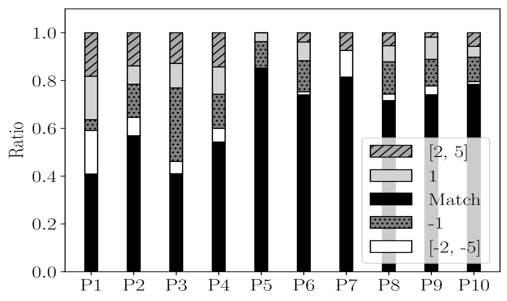


Figure 9: Directed distance between student and TA DR picks ($\mathcal{D}$)

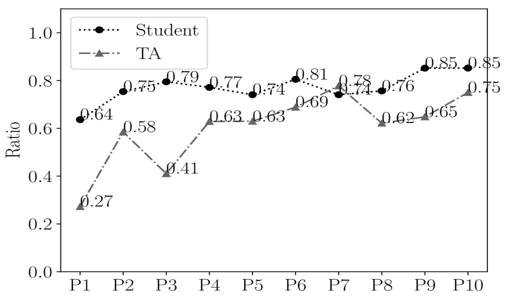


Figure 10: Rate of single step picked by students and TAs ($\mathcal{D}$)

of Fil, which was not very hard and only two days long), indicating growing student agreement with the expert TAs.

However, this distance is not truly symmetric: we care about students conforming to TAs, not the other way around. We therefore calculate a *directed* distance between TAs and students, shown in fig. 9. If the two have Hamming distance 0, we call this a *match*. Otherwise, we define the directed distance as the *latest* step on which the student and TA disagreed, where as positive number means the TA helped the student with a later stage (i.e., the student may be making good progress), while negative means the TA stopped with an earlier stage (i.e., the student is going too far ahead). The ratio of matches improves significantly: an average of 46.29% for the first three versus 74.70% for the last three. (We also checked whether the help classification correlates with the distance, but a χ^2 test gave a p-value of 0.515, suggesting no significant association.)

6.2.3 *Do students under- or over-estimate?*

The directed distance lets us ask whether certain students consistently over- or under-estimate their progress: overestimators have a negative score and underestimators a positive one. We examined all students who had five or more positive or negative scores.

There were five of these underestimators, four of them five times and one *22* times. Of those four, one had *12 negative* scores, suggesting poor accuracy in estimating the step. The one with 22 positive scores had 4 negative scores. We also found four such overestimators, two of them having done so *12* times. They had far fewer positive scores (0, 1, 2, and 5), suggesting a general tendency to overestimate. One student is, of course, in both groups.

We believe this analysis is a potentially useful diagnostic for instructors. They may wish to identify consistent under- and over-estimators and give them suitable feedback (boosting their confidence or suggesting more caution). A large number of students who are both would suggest class-level difficulty with using the DR, but we did not observe that in our population.

6.2.4 *How often do students pick just a single DR step?*

Later DR steps depend heavily on material from earlier ones, so students should complete earlier steps before moving on to later ones. Indeed, in theory, they should be working on one step at a time. In practice students may not behave this way; they may also choose multiple steps for expediency (to avoid waiting to see TAs again).

Nevertheless, we hypothesized that earlier in the course, students would tend to pick multiple steps at a time, and later pick a single step more often as their facility with (and, hopefully, perceived value of) the DR increases. We also conjectured that TAs would use a single step more often than students.

Figure 10 shows the data. We see that student selection of a single step (perhaps surprisingly) starts high. It stays high, and may have an implicit ceiling effect due to expediency. Nevertheless, their choice shows a significant positive slope: linear regression gives a slope of 0.015 (and a p-value of 0.022, which is less than the default alpha of 0.05) for students. TAs' choice also shows a significant positive slope of 0.038 (p-value is 0.011).

The TA line was, to us, even more surprising. Of the 100 times where the student picked a single step and the TA picked multiple steps, the student had requested one of the steps chosen by the TA; more than half the time, the TAs chose one of these three pairings: steps 1 & 5, 5 & 6, or 1 & 6. This suggests that the TAs are forcing students to make connections they were perhaps not seeing across different steps of the DR.

7 LIMITATIONS

There are several natural threats to generalizability. For instance, this specific form would not work for courses that do not use or emphasize the DR, or would need to be enriched if they have even more non-programming content. We also depend on TAs being both willing and conscientious. We also depend on a protocol for students to get on the TA waitlist, which may not exist at institutions with enough TAs.

By considering only TA hours, we do not account for other student help-seeking behavior, such as faculty office hours (which were used vastly less) or an on-line forum (or, even more subtly, programming tools). During this course, there were 1029 total posts to Piazza, which is substantial but not that much more than in-person use. Since Piazza did not enable us to use a similar rubric, it is non-trivial to combine the two sets of data. (An interesting question is how the lack of a rubric affects questions.) These other sources thus represent a threat to the validity of our findings. Also, not all students seek help, sometimes for logistical reasons. (This course's hours were spread through the day and week, and adapted to changing student demand. Also, the course does not have a notable commuter population.)

As noted, the course under study uses solely undergraduate TAs (UTAs). On one hand, UTAs have not had as much experience or computer science education as graduate TAs and faculty, so this represents a threat to validity. On the other hand, all our UTAs had done the same course, so they arguably have a much better understanding of the specifics of the material than graduate TAs (which is one reason the course does not employ graduate TAs). Variance-reduction measures (§ 4.5) may also help substantially. Thus, we have reason to place significant faith in their assessment. Finally, many universities have or are instituting undergraduate TA programs, especially since the number of undergraduate TAs grows roughly proportionally to the size of the undergraduate population itself—a vital need as enrollments grow rapidly in some countries. Thus, understanding this population well is important.

There is a time gap (median 21 minutes) between when a student signs up and when the TA submits their assessment, which happens when the meeting is over. (In 2.4% of the cases there was a gap of over four hours, longer than the longest contiguous TA sessions, which most probably reflects students "squatting" on a TA slot at the next available hours.) During this time they may reflect and change their question, which is not captured by our student form. This may explain some of the differences in entries between students and TAs, but the student forms still represent a meaningful construct: the student's opinion at the time they signed up for help.

There is some chance that TAs get less conscientious as the semester progresses and simply choose the same boxes as the students marked, which would explain the assessment convergence. This impacts data validity. We are not aware of a way to check for this.

8 DISCUSSION

We set out to develop insight into what kinds of help students seek at TA hours and how it evolves. We wanted to do this with a minimally-intrusive and lightweight process, and conjectured that the DR of HTDP could help.

Our main contribution is that the DR actually provides a useful initial view into student use of TA hours. For different assignments the steps used varies, in a way consistent with the assignment's content and difficult spots. Because they can "show their work" over multiple steps, students are able to ask more focused questions than just the stereotypical "My code doesn't work". Students also appear to make progress in how they use the DR over time.

By using the DR, we are effectively also assessing the book that defines it, HTDP. The DR cannot really be measured through assignment submissions, because the DR is a *dynamic process* while the submissions are *static artifacts*. Therefore, though some prior work [9, 10, 17, 18] has provided some insight into the end-result of asking students to use the DR, we know little until now about how students actually use this theory of programming education. Our secondary contribution is showing that students can identify and meaningfully use the different DR steps, and actually do so.

We believe our instrument offers a useful starting point even for courses that do not use HTDP or the DR. So long as the course's learning process can be broken down into concrete steps or areas, we believe there is real value to lightweight forms that force self-reflection before asking for help, and that let TAs assess what students actually needed help with. In our case the findings were generally positive and not disconcerting, but the same process would also have helped us identify worrisome negative trends—and in almost real time, enabling interventions as needed. Especially for courses that use some kind of systematic program design methodology, we believe an instrument analogous to ours can offer instructors significant insight into how it is actually being used.

The data from our instrument, such as the use of TA hours and especially on under- and over-estimation, may also offer insight into how individual students would themselves fare as future TAs.

Finally, other authors [10] have written with some concern about the impact of TAs on student learning: for instance, TAs may be providing information or setting goals that (perhaps inadvertently) contradict the instructor's intent. As course sizes grow and reliance on TAs grows with it, more and more student instruction will actually come from TAs. Therefore, we hope this work will inspire many more efforts to obtain a more detailed understanding of what takes place in TA hours.

9 FUTURE WORK

There are numerous interesting directions for future work to build on these results. We list just a few:

The DR does not account for all the steps in programming. Even with our classification scale, for instance, our forms mask whether students got help with syntax, run-time, or logic errors. It would be interesting to refine the DR questions to get more fine-grained information about help-seeking. We feel there are natural extensions—like additional check-boxes or small free-form text boxes—that will provide much more insight while preserving the constraints laid out in this paper. We believe there is also great value to exploring assignment-specific questions.

It would be very useful to combine information from TA hours, student help fora, programming tools, and other sources to provide a comprehensive overview of student help-seeking behavior. This would require a detailed content-coding effort to analyze actions across such a diverse set of resources with distinct structures.

In § 3 we discuss findings on the relationship between reflective processes and student success. In general it can be hard to observe students in the act of reflection, but TA hours (and other help-seeking fora) provide some visibility into this. The very use of the student form may force some reflection that does not otherwise happen, which may result in small but measurable differences in outcome. (A potential study would be to measure the impact of taking away the student form.) A broader study could analyze how students' problem-solving strategies interact with the theories of learning we have discussed in § 2. This may be possible using extensions or adaptations of the current instrument.

We have not examined any of the social and cultural issues raised in hours. In general these are few (option 13 in fig. 6), but a more fine-grained analysis may reveal these issues hidden in other steps.

ACKNOWLEDGMENTS

This work was partially supported by the US National Science Foundation. We thank the TAs in the course for adopting and using our instrument, which increased their workload. We thank Jack Wrenn for his help in brainstorming this idea. We thank our meta-reviewer for their valuable feedback.

REFERENCES

[1] John R. Anderson, Frederick G. Conrad, and Albert T. Corbett. 1989. Skill Acquisition and the LISP Tutor. *Cognitive Science* 13 (1989), 467–505. https://doi.org/10.1016/0364-0213(89)90021-9

[2] Charles E Brown. 1998. Coefficient of variation. In *Applied multivariate statistics in geohydrology and related sciences*. Springer, New York, NY, USA, 155–157. https://doi.org/10.1007/978-3-642-80328-4

[3] Jerome Bruner. 1978. On the Mechanics of Emma. In *The role of dialogue in language acquisition*, Anne Sinclair, Robert J. Jarvella, and Willem J. M. Levelt (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 241–256. https://doi.org/10.1007/978-3-642-67155-5

[4] Jennifer Campbell and Michelle Craig. 2018. Drop-In Help Centres: An Alternative to Office Hours. In *Proceedings of the 23rd Western Canadian Conference on Computing Education (WCCCE '18)*. ACM, New York, NY, USA, Article 9, 6 pages. https://doi.org/10.1145/3209635.3209642

[5] Michelene TH Chi. 2009. Active-constructive-interactive: A conceptual framework for differentiating learning activities. *Topics in cognitive science* 1, 1 (2009), 73–105. https://doi.org/10.1111/j.1756-8765.2008.01005.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1756-8765.2008.01005.x

[6] M Fahim and E Fakhri Alamdari. 2014. Maximizing learners' metacognitive awareness in listening through metacognitive instruction: An empirical study. *International Journal of Research Studies in Education* 3, 3 (02 2014), 79–91. https://doi.org/10.5861/ijrse.2014.762

[7] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2018. *How to Design Programs* (second ed.). MIT Press, Cambridge, MA, USA. http://www.htdp.org/

[8] Matthias Felleisen, Matthew Flatt, Robert Bruce Findler, Kathryn E. Gray, Shriram Krishnamurthi, and Viera K. Proulx. 2001. *How to Design Classes: Object-Oriented Programming and Computing*. https://felleisen.org/matthias/HtDC/htdc.pdf

[9] Kathi Fisler. 2014. The Recurring Rainfall Problem. In *Proceedings of the Tenth Annual Conference on International Computing Education Research (ICER '14)*. ACM, New York, NY, USA, 35–42. https://doi.org/10.1145/2632320.2632346

[10] Kathi Fisler, Shriram Krishnamurthi, and Janet Siegmund. 2016. Modernizing Plan-Composition Studies. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 211–216. https://doi.org/10.1145/2839509.2844556

[11] Emily R. Fyfe, Nicole M. McNeil, Ji Y. Son, and Robert L. Goldstone. 2014. Concreteness Fading in Mathematics and Science Instruction: a Systematic Review. *Educational Psychology Review* 26, 1 (01 Mar 2014), 9–25. https://doi.org/10.1007/s10648-014-9249-3

[12] Dastyni Loksa. 2017. Explicitly Teaching Metacognitive and Self-Regulation Skills in Computing. In *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17)*. ACM, New York, NY, USA, 289–290. https://doi.org/10.1145/3105726.3105740

[13] Dastyni Loksa and Andrew J. Ko. 2016. The Role of Self-Regulation in Programming Problem Solving Process and Success. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, New York, NY, USA, 83–91. https://doi.org/10.1145/2960310.2960334

[14] Elizabeth Patitsas. 2012. A Case Study of Environmental Factors Influencing Teaching Assistant Job Satisfaction. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research (ICER '12)*. ACM, New York, NY, USA, 11–16. https://doi.org/10.1145/2361276.2361280

[15] Elizabeth Patitsas and Patrice Belleville. 2012. What Can We Learn from Quantitative Teaching Assistant Evaluations?. In *Proceedings of the Seventeenth Western Canadian Conference on Computing Education (WCCCE '12)*. ACM, New York, NY, USA, 36–40. https://doi.org/10.1145/2247569.2247582

[16] Peter L. Pirolli and John R. Anderson. 1985. The Role of Learning from Examples in the Acquisition of Recursive Programming Skills. *Canadian Journal of Psychology/Revue canadienne de psychologie* 39, 2 (1985), 240–272. http://dx.doi.org/10.1037/h0080061

[17] Emmanuel Schanzer, Kathi Fisler, and Shriram Krishnamurthi. 2018. Assessing Bootstrap: Algebra Students on Scaffolded and Unscaffolded Word Problems. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 8–13. https://doi.org/10.1145/3159450.3159498

[18] Emmanuel Schanzer, Kathi Fisler, Shriram Krishnamurthi, and Matthias Felleisen. 2015. Transferring Skills at Solving Word Problems from Computing to Algebra Through Bootstrap. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 616–621. https://doi.org/10.1145/2676723.2677238

[19] Aaron J. Smith, Kristy Elizabeth Boyer, Jeffrey Forbes, Sarah Heckman, and Ketan Mayer-Patel. 2017. My Digital Hand: A Tool for Scaling Up One-to-One Peer Teaching in Support of Computer Science Learning. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 549–554. https://doi.org/10.1145/3017680.3017800

[20] James C. Spohrer and Elliot Soloway. 1989. Simulating Student Programmers. In *Proceedings of the 11th International Joint Conference on Artif icial Intelligence - Volume 1 (IJCAI '89)*. Morgan Kaufmann Publishers Inc., 543–549.

[21] John Sweller. 2006. The worked example effect and human cognition. *Learning and Instruction* 16, 2 (2006), 165 – 169. https://doi.org/10.1016/j.learninstruc.2006.02.005 Recent Worked Examples Research: Managing Cognitive Load to Foster Learning and Transfer.

[22] Laura Toma and Jan Vahrenhold. 2018. Self-Efficacy, Cognitive Load, and Emotional Reactions in Collaborative Algorithms Labs - A Case Study. In *Proceedings of the 2018 ACM Conference on International Computing Education Research (ICER '18)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/3230977.3230980

[23] Mickey Vellukunnel, Philip Buffum, Kristy Elizabeth Boyer, Jeffrey Forbes, Sarah Heckman, and Ketan Mayer-Patel. 2017. Deconstructing the Discussion Forum: Student Questions and Computer Science Learning. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 603–608. https://doi.org/10.1145/3017680.3017745

[24] Brenda Cantwell Wilson and Sharon Shrock. 2001. Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors. *SIGCSE Bull.* 33, 1 (Feb. 2001), 184–188. https://doi.org/10.1145/366413.364581

[25] Jack Wrenn and Shriram Krishnamurthi. 2019. Executable Examples for Programming Problem Comprehension. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ACM, New York, NY, USA, 9. https://doi.org/10.1145/3291279.3339416