3-3-2023

Accurate Estimation of Time-on-Task While Programming

Kaden Hart
Utah State University, kaden.hart@usu.edu

Christopher M. Warren
Utah State University, chris.warren@usu.edu

John Edwards
Utah State University, john.edwards@usu.edu

Recommended Citation

Kaden Hart, Christopher M. Warren, and John Edwards. 2023. Accurate Estimation of Time-on-Task While Programming. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569804

Accurate Estimation of Time-on-Task While Programming

Kaden Hart
Utah State University
Logan, Utah
kaden.hart@usu.edu

Christopher M. Warren
Utah State University
Logan, Utah
chris.warren@usu.edu

John Edwards
Utah State University
Logan, Utah
john.edwards@usu.edu

Abstract

In a recent study, students were periodically prompted to self-report engagement while working on computer programming assignments in a CS1 course. A regression model predicting time-on-task was proposed. While it was a significant improvement over ad-hoc estimation techniques, the study nevertheless suffered from lack of error analysis, lack of comparison with existing methods, subtle complications in prompting students, and small sample size. In this paper we report results from a study with an increased number of student participants and modified prompting scheme intended to better capture natural student behavior. Furthermore, we perform a cross-validation analysis on our refined regression model and present the resulting error bounds. We compare with threshold approaches and find that, in at least one context, a simple 5-minute threshold of inactivity is a reasonable estimate for whether a student is on-task or not. We show that our approach to modeling student engagement while programming is robust and suitable for identification of students in need of intervention, understanding engagement behavior, and estimating time taken on programming assignments.

CCS Concepts

• **Social and professional topics** → CS1.

Keywords

CS1, Keystrokes, Engagement, Vigilance

ACM Reference Format:
Kaden Hart, Christopher M. Warren, and John Edwards. 2023. Accurate Estimation of Time-on-Task While Programming. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569804

1 INTRODUCTION

Time-on-task is an important topic in computing education research (CEdR) that is surprisingly under-studied [6]. With few exceptions, researchers choose arbitrary thresholds to determine if a break in activity indicates student disengagement or not, resulting in a hodge podge of cutoff values, of which almost none are empirically determined. Such haphazard methodologies are a concern, especially when considering the wide-ranging importance of estimating the amount of time a student (or any programmer, for that matter) spends on a task: time-on-task can be used to determine difficulty of concepts or assignments, identify struggling students, give students an idea of how they're performing relative to their peers, study attention and engagement patterns, and compare contexts.

A recent study [1] resulted in a first-of-its-kind model that predicts time-on-task using keystroke logs. The innovation was both in the model itself, a regression model instead of one that is threshold-based, and in the methodology, the model being derived from empirical data. Despite methodological shortcomings, the study resulted in a viable estimation approach backed with material justification. Our objective in this paper is to present results of two studies that address methodological issues acknowledged in the original study and that extend the work, including comparison with the seemingly ubiquitous threshold approaches.

The study by Edwards et al. [1] showed how a statistical model could be built to determine the probability of student engagement by intermittently asking students if they were working on their computer programming assignment or not. A plugin was installed in students' coding environments that would record their keystrokes and when a predetermined amount of inactivity was detected, showed students a window that asked if they were engaged or not. Researchers built a regression model which could then be used to find the probability that a student was engaged given a latency between keystrokes.

The statistical model of engagement worked by querying students at randomly selected times from a predetermined set of eight break lengths: 45 seconds, and 1, 1.5, 2.5, 4.5, 8.5, 16.5, and 32.5 minutes. A shortcoming of this approach is that students may have been on what would have been say, a 10 minute break, but after noticing an engagement query at 8 minutes, they decided to return to their task. This side-effect would artificially increase engagement rates. A second shortcoming of this approach is that the yes or no answers did not consider breaks where students were partially engaged; the responses had to be all or nothing. While the proposed model was based on empirical data, the lack of any error analysis compromised confidence in its use. Nor was there any analysis of existing threshold methods – despite the fact that thresholds are generally arbitrary, some statistical justification should be made before discontinuing their use.

This paper has five specific contributions to this line of inquiry:

- C1 Reactive queries prompt students only after they have returned to interaction with their coding environment. Reactively querying students in this manner allows us to observe the natural duration of a student's inactivity without changing their behavior during the break.
- C2 Queries allow students to respond with what percent of their break they were engaged instead of just whether they were engaged or not.

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada

ACM ISBN 978-1-4503-9431-4/23/03.
https://doi.org/10.1145/3545945.3569804

C3 Cross-validation error analysis shows us how well our methods model student engagement. We find that the model is robust and can be relied on to estimate important measures like time spent on assignments.
C4 We include a detailed comparison with threshold approaches and, somewhat unexpectedly, empirical validation of a 5-minute threshold in one context.
C5 We conducted our study with 92 participants, more than six times the 15 participants in the original study.

We find that, even given the flaws of the original study, the regression model was reasonably robust and accurate, though our refined model has tighter error bounds. We also find that, in the context of our CS1 course, a 5-minute threshold for determining engagement is surprisingly reasonable, easing implementation for educators who prefer something simpler than the regression model.

2 Background

2.1 Vigilance and Engagement

Central to our work are the psychological topics of vigilance and engagement. Vigilance refers to a person's capacity to maintain attention on a task over time. Maintaining attention gets more difficult as time goes on and mental resources are depleted, a decline termed the "vigilance decrement" [2, 12, 16, 19]. Investing physical, cognitive, and emotional energy on a task is task engagement [4, 15]. The recent regression model showed how longer periods of inactivity in students' programming assignments yielded lower engagement rates [1]. Our engagement model shows the vigilance decrement as a function of an inactive period of time.

2.2 Time-On-Task

Time-on-task is the amount of time a student spent working on an assignment, and has been studied since the 1970's [6]. Leinonen et. al. [8] noted that time-on-task is one of the most important metrics that contribute to learning and achievement. Leinonen et. al. [8] also found that fine-grained metrics performed better than course grained metrics; for example Munson's [13] method of calculating time-on-task by subtracting the time from a student's first compilation event from the last may not work as well as Toll et. al.'s [20] method of using a combination of keystrokes, mouse movements, and other events. Toll et. al. were able to use their additional levels of granularity to estimate different activities as well as time-on-task, such as periods of time with mouse movement but not keystrokes suggests reading or navigating, but not editing.

A difficulty with time-on-task is the difference between time taken, and time spent engaged [6]. Judd found that given a 15 minute study period, students only spent 10 minutes engaged [3]. As technology continues to be more accessible, distractions become easier and students may spend even more time disengaged. Kovanovic et. al. argue that "time-on-task estimation, its issues, limits, and reliability challenges warrant further consideration." [6].

2.3 Thresholds

Thresholds are a popular way to make time-on-task estimates. In keystroke data, thresholds work by including any latencies less than the threshold in the time-on-task estimate, and ignoring any latencies longer than the threshold. Using thresholds to determine time-on-task estimates from keystroke data is a common practice; however, what threshold is best is not straightforward, and may be context dependant [8]. Kovanovic et. al. noted that many researchers use a threshold and do not address the consequences of that threshold [6]. Nevertheless, these approaches are as common as their threshold values are haphazard. Suggested threshold values range from 60 seconds [18] to 3 minutes [10] to 5 minutes [7] to 30 minutes [14] to one hour [5], and none of them were determined from empirical data. Indeed, authors acknowledge this fact using language such as "reasonable picture" [18], "assuming" [14], and "would seem reasonable" [9].

Only one work has proposed a threshold even loosely based on empirical data. Leinonen et al. [9] produced a curve showing the number of work sessions given different threshold values and use an argument somewhat akin to the elbow method to support a 10-minute threshold, though they nevertheless acknowledge significant uncertainty.

2.4 Empirically Determined Model of Engagement

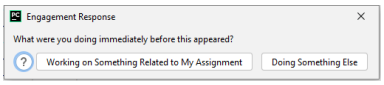
In a recent study, Edwards et al. installed software on students' PyCharm IDEs that would periodically ask students if they were working on their assignments when inactivity was detected [1]. Installing software directly on student computers allowed data to be collected in a student's natural working environment and recorded behaviors as they happened. To query students, after each keystroke a timer was started to count down from one of a set of eight predetermined query times and if the timer reached zero before the student pressed another key, the student was immediately prompted to answer if they were working on their assignment or not.

Edwards et al. built rates of engagement from student responses for each query time, and then fit a generalized logistic curve to yield an engagement function for the probability of engagement given time since last keystroke. With this function, time-on-task for an assignment could be estimated. To estimate time-on-task for an assignment, for each latency between keystrokes in a student's assignment, if a randomly generated number is smaller than the probability given by the engagement function for that latency, add that latency to the time a student spent engaged with their assignment. This method to estimate time-one-task was novel because the probability of engagement was data driven and not arbitrary.

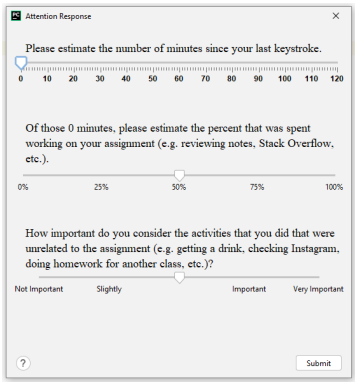
With their model they discovered many previously unknown student behaviors. They found that after about 3 1/2 minutes of inactivity, half of the students had disengaged from their assignments. Students on average worked for 8.5 minutes on their assignments before disengaging and half of the time returned to work after between 1 to 4 minutes.

3 METHODS

This paper reports on two studies. In the first, which we call the Reactive study, CS1 students were periodically prompted while working on their computer programming assignments to self-report if they were working on their assignment at the moment they were prompted (Figure 1a). In the second, which we call the Slider



(a) Reactive study



(b) Slider study

Figure 1: Screenshots of the prompts used in the studies.

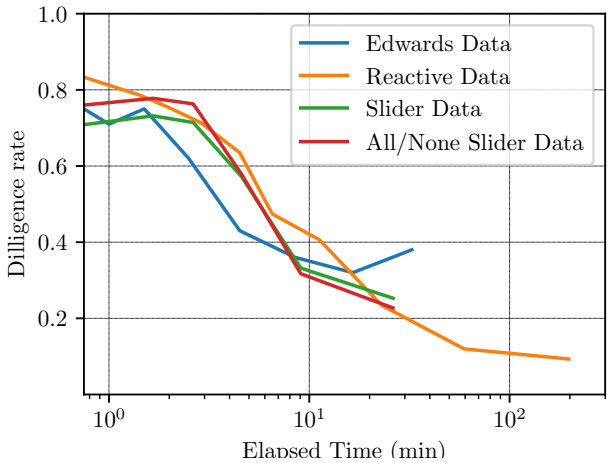


Figure 2: Comparison of Edwards et al.'s [1], our Reactive, our Slider, and our all/none Slider data (section 4). Elapsed time uses a log scale.

study, CS1 students were similarly prompted but were given the ability to report how long ago they thought their last keystroke was, what percentage of that reported time was spent working on their assignment, and how important any non-assignment related thing they did during that reported time was (Figure 1b).

3.1 Prompts

PyCharm is the IDE required for students to use in the CS1 classes we investigated. A plugin was developed for PyCharm that participating students installed that logged their keystrokes and prompted them to answer questions about their engagement. After a random period of inactivity (see Section 3.2), students were shown a window that prompted them to respond. In the Reactive study, the window asked "What were you doing immediately before this appeared?" Students would then choose one of the two responses to continue their work: "Working on something related to my assignment" or "Doing something else." In the Slider study, the window gave three sliders to respond and asked three questions: "Please estimate the number of minutes since your last keystroke", "Of those X minutes, please estimate the percent that was spent working on your assignment", and "How important do you consider the activities that you did that were unrelated to the assignment?" Responses were recorded within the same log as the keystrokes. Data was collected in a student's normal working environment rather than in a lab setting which could change their behavior.

In the Reactive study, 96% of responses occurred within 10 seconds of the prompt appearing, and 99% occurred within 60 seconds. In the Slider study, 58% of responses occurred within 10 seconds of the prompt appearing, and 99% occurred within 60 seconds. We do not consider the time a student took to respond to a prompt important because the prompt was asking what they had been doing during the break that they had just finished, and not what they were doing after being prompted. Response time to our prompts is likely more indicative of student familiarity with the prompts than engagement; the first time a student saw a prompt they may take more time to understand what it was asking them to do. In the Slider study, students may have taken longer because of the additional work of adjusting the sliders.

3.2 When to Prompt

Both of our studies used the same method to determine when to prompt a student: after each keystroke, apply a 70% chance that the keystroke will qualify to show a prompt. If the keystroke qualifies then use the amount of time since the previous keystroke to find the probability that a prompt should be shown using the Cumulative Density Function (CDF) used by Edwards et al. [1]. Then randomly generate a number, and if it is lower than the probability, query the student. Informally, after a student does not type for a while they may be prompted after their next keystroke.

3.3 Participants

Students in a mid-sized university in the United States were given the opportunity to participate in our study. Students were not offered any compensation. In the Reactive study, 62 students participated in our study across 8 assignments. In the Slider study, 30 students participated across 10 assignments. This represents a six-fold increase in the number of participants over Edwards et al.'s study. Both the Reactive and the Slider studies were conducted under review and approval of our university's ethics review board.

4 RESULTS

In the Reactive study, 365 assignment submissions were collected from our 62 participants. On task, engaged, responses to the queries totaled 970. Off task, disengaged responses to the queries totaled 951. In the Slider study, 271 assignment submissions were collected from our 30 participants. The estimated break duration slider defaulted to 0 minutes. If the student did not change this slider, the response was discarded. We collected 883 responses and discarded over half, leaving 396 usable responses.

When comparing engagement rates between our Reactive study and Edwards et al.'s [1] we found a higher rate of engagement in general. See Figure 2. A possible reason for the higher engagement rate is student misunderstanding because of our revised prompting methodology. In the original study [1], students would be prompted at the query time, and may not see or respond to the query very quickly. In our Reactive study the prompt was only shown after the student began typing again, which means the student would be interrupted immediately after pressing a key and see the prompt as soon as it was shown. An interrupted student may feel that since they had just typed a key that they were obviously on task. Our prompts asked what students were doing immediately before they saw the prompt, and students may have responded with what they had literally just done – pressed a key – and not with what we were interested in – what they did during their break. Our second study, Slider, asked students to estimate the percentage of time they were on task. When students were given the chance to estimate how long their break was, they may have understood our intent better. The slider study data shows a lower engagement overall, but shows more noise due to a smaller sample size.

4.1 Model of Engagement Rate

A difference between our model and prior work is the continuous elapsed time variable of queries instead of a discrete set of query times. A continuous elapsed time variable gives us many options to use when determining the vigilance decrement curve (Figure 2): time scaling bins (e.g 1 minute, 2 minute, 4 minute), rolling averages, or equisized bins. Time scaling bins defined arbitrary times to bin, and led to bins with wildly varying observation counts. Rolling averages introduced noise when used with binary responses. Ten equisized bins were used for the Reactive study because they limited noise and provided equal support for each bin. Six equisized bins were used for the Slider study because of the reduced sample size. Each query response fits in the nearest bin that has a smaller or equal elapsed time than the query.

Conceptually, as elapsed time increases, the engagement rate should start at 100% and go to 0%. That is, after zero seconds from the last keystroke, the student would have a 100% probability of being on task, and after infinite time, the probability of engagement would be 0%. Student responses may be biased towards higher engagement rates as discussed in Section 5.2. Edwards et al. suggested an adjustment for bias and used a generalized logistic curve to fit the corrected data because it has a higher and lower asymptote. We also use the generalized logistic function, but instead of correcting the data we define the top asymptote as 1.0 and the bottom asymptote as 0.0. We then fit the curve to our sampled data. See Figure 3. The generalized logistic function fit to our data is

$$y(x) = \frac{1}{(1 + Qe^{-B(x-M)})^{1/v}} \quad (1)$$

with $Q = 6604, B = -4.99, M = 0.01, v = 58.32$, and x equal to the keystroke latency in minutes. With this model we can probabilistically classify a latency as engaged or disengaged.

4.2 Time-On-Task

Time-on-task for an assignment is determined using the following method: for every keystroke, add the latency to the assignment's time-on-task if the latency is less than 45 seconds (the lower bound of our prompts) or if a randomly generated number between 0 and 1 is lower than the probability given by Equation 1. Using this method with the 365 submissions we collected in the Reactive study, we found similar statistics as Edwards et al.[1].

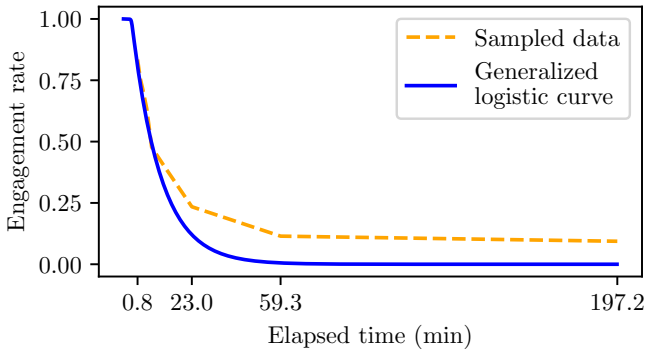


Figure 3: Generalized logistic curve. The curve is fit using the Levenberg-Marquardt algorithm [11]. Sampled data is from the Reactive study.

In the Reactive study, in which students could give only a yes or no response as to whether they were on task, we found that participants had a median engagement time of 9.6 minutes, or 209 keystrokes, before becoming disengaged. Our study shows an increase from 8.5 minutes of engagement, or 163 keystrokes, found in Edwards et al.'s study, possibly due to students only being prompted after they begin typing after a break. Our study found the same statistic as Edwards et al.'s study that most breaks last 1 to 4 minutes. An interesting statistic found in Edwards et al.'s study, as well as ours, was that on average, students take breaks over 20 times during the course of a programming assignment. This pattern of working and taking short breaks is an interesting phenomenon, and what students do during such short breaks is an important question for future work.

4.3 Partially On-Task Responses From Slider Study

In the Slider study, we found that some breaks that are mixed i.e. students were on task for part of the time; however, most responses – 65% – were either 0% or 100% engaged, which we call an all/none response. Figure 2 shows the difference between using all responses compared to using just the all/none responses. The default value for the percentage of on time-on-task was 50%. Of the 396 responses in the Slider study, 11% left the slider at the default value. Keepring their responses had the effect of smoothing the engagement rate curve towards 50% probability of being on task (Figure 2). As we show in Section 4.4, removing all responses that weren't all/none, resulted in engagement curves within error bounds of using all of the Slider study data.

4.4 Error Bounds

Determining how well an individual's engagement is modeled cannot be done directly with keystroke data. We instead determine how well different groups are modeled using a method known as cross validation. Cross validation is commonly used in machine

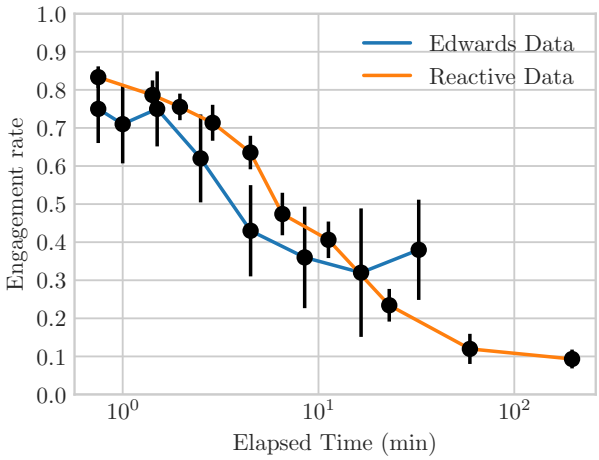


Figure 4: Comparison of the standard deviation of error between Reactive study and Edwards et al.'s data. The Reactive study results in smaller error.

learning where data is divided into two groups, usually called the training and test groups. A model is built on the training set and tested on the test set. Data division and model building is done multiple times and an error estimate can be determined either by how much the model changes or by how well the model fits the test data across trials. For our purposes we randomly removed half the students from the data and then determined how well that half of the students fit the entire data set to measure the error. Repeating this many times showed how sensitive the engagement rates are to individual differences in behavior. After repeating this process 1,000 times on our Reactive data and Edwards et al.'s [1] data, we were able to see that our Reactive data had significantly less error. See Figure 4. We collected 1,921 responses where Edwards et al. collected 424; we suggest our larger sample size accounts for the improved error bounds.

Students are more likely to be on task with very short breaks and more likely to be off task with very long breaks. At engagement rates near 50% is where we expect to see the most error. Standard deviations show where most student's engagement rates will be and an inspection of the size of the standard deviation indicates that error increases as the engagement rate nears 50%, and decreases as engagement approaches 100% or 0%. See Figure 4. Most students' behavior will be within 10% of the sampled data. On occasion a student may exhibit outlier behavior, but this is to be expected with human behavior.

4.5 Thresholds Vs. Regression Model

Thresholds, i.e., "a student is on task if their inactivity is below t seconds." are common in the time-on-task estimation literature. Figure 5 shows how much student time-on-task predictions change when using different thresholds. The chosen threshold can greatly affect the time on task estimate, emphasizing Kovanoic et. al.'s call for "more caution when using time-on-task measures" [6]. If a researcher chooses a short threshold, they will find significantly lower time-on-task estimations than if they used a longer threshold. What

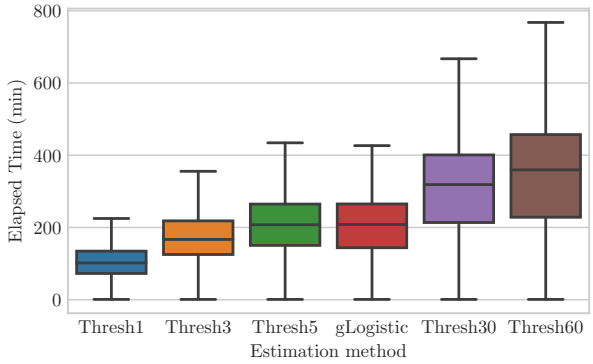


Figure 5: Time-on-task estimates for Reactive data using different estimation methods. Thresh1 uses a 1-minute threshold, Thresh3 uses a 3-minute threshold, etc. gLogistic uses the Reactive model described in section 4.2.

surprised us the most was the similarity between our model and the 5-minute threshold. We found that our model and the 5-minute threshold predict similar time on task with no significant difference. We suggest that educators consider using the 5-minute threshold for time-on-task estimates instead of our probabilistic model because of its simplicity. For research, our model is more suitable as it provides probabilities that students were engaged during a break, giving greater flexibility for analysis. We do caution that our model was built with data from CS1 classes at our university, and we do not know how well our model or the 5-minute threshold generalize to other contexts.

5 DISCUSSION

5.1 Sampling

We used the same CDF as the Edwards et al. [1] study to achieve uniform sampling across time. With our error bounds in the Reactive study we can also show that individual behaviors are most different when the average engagement rate approaches 50%. We discovered that data when engagement rates are nearing 100% or 0% are not as varied; the error is small, showing most students are behaving similarly.

5.2 Bias of Responses

In the Edwards et al. [1] study an adjustment to the data was suggested due to the non-zero asymptote found in their data, which should be zero. In our Reactive study, we similarly found a lower asymptote of around 10%. Rather than correct the data, we fixed a lower asymptote of zero to the regression curve and then fit it to the data. We argue that this adjustment is appropriate because, conceptually, the lower asymptote should be zero, and so student responses are not completely accurate. We consider two cases in which student responses are likely not accurate: (1) students may not answer honestly or (2) students may have only worked a short time just before they answered the query. First, students answering dishonestly should not come as a surprise; some students may want to appear diligent and hardworking, and answer in a way that makes them seem that way. For example, in the Reactive study, we

found 15 students who responded on task to a prompt after a break over an hour, including 3 that never once responded as being off task after an hour. Second, students may have been disengaged for most of their break, but before they began typing they re-engaged. When seeing the question, "What were you doing immediately before this appeared?", the answer is obvious: they had just done something for their assignment – they pressed a key! The goal of our prompt is to discover if they were engaged during the break, and these students may have misunderstood. For example, in the Slider study, when given the ability to respond with what ratio of their break was spent engaged or not, some students still responded as 100% engaged even after very long periods of time.

It is impossible to tell which responses were dishonest and which were misunderstood. In either case, at low latencies of around 45 seconds, students were probably on task anyways and did not bias the data much, requiring little correction; however, at higher latencies, around 30 minutes, inaccurate answers likely biased the data more because students were most likely not on task when they answered as being on task. We currently have no suitable way to determine how much to correct our data so we defined the lower asymptote as zero and let the model determine the correction automatically. We considered other possible corrections, but ultimately decided that any other correction we made would be asserting arbitrary opinions onto the model.

5.3 Study Fatigue

We found that over the course of the semester students interacted less with the sliders (closing the dialog without adjusting the break duration). At the beginning of the semester students interacted with 74% of the dialogs. Over the course of the semester students continually interacted with less of the dialogs shown, and at the end only interacted with 26% of the dialogs. This decline in participation may show that the sliders became either too annoying or burdensome as the course progressed. Our prompts were designed to not distract students too much, but the decline in participation may show that giving students sliders to respond to dialogs is too distracting.

5.4 Threats to Validity

As discussed in sections 5.2 and 5.3, bias of responses and study fatigue may skew our results. Our studies required students to opt-in and may suffer from self-selection bias. As noted in our discussion on generalizability, our data was drawn from one school over the course of two terms of CS1. Thus, we encourage replication in other courses and institutions. Also, students may have felt that they were being observed through logged keystrokes and queries, and exhibited different behaviors than they otherwise would, the so-called Hawthorne effect [17].

5.5 Time-On-Task Estimates

We have shown that for time-on-task estimates in the CS1 courses in our study, our regression model and the 5-minute threshold perform similarly. Replications of our study in different contexts could show how well the 5-minute threshold performs in different contexts, and to what extent it is generalizable for time-on-task estimates. Assignment length may be a context that changes student behavior – shorter assignments may encourage student to stay on task longer to "just get it done" – and changes the appropriate threshold to use.

Kovanovic et. al. called for researchers to take caution when choosing time-on-task measures, and we have given a method to help researchers choose the appropriate threshold or model for their context [6]. To help researchers gather data easier, we have published the *Time On Task* plugin to the JetBrains plugin marketplace for use in JetBrains products like PyCharm and IntelliJ. Also available is the Python code to build the model and compare thresholds at https://edwardsjohnmartin.github.io/Scripts/index.html

6 CONCLUSIONS

Measuring time-on-task of computer programming students no longer needs to be the irksome task that so often uses glossed-over methods. This study, combined with that of Edwards et al. [1], gives evidence that we can accurately and robustly measure the on-task behavior of students while programming. These studies have focused on CS1 students at a mid-sized university in the United States, but the techniques and software used to collect data can be used in any context by any researchers interested in creating models specific to their context or, better yet, finding generalized models.

A past criticism of using keystroke data for measuring time-on-task is that students are often on task even when they're not physically programming. Students may be reviewing notes, designing a solution, or searching the internet for help. This study models those behaviors as well, since students self-report engagement of any kind. The non-programming on-task behavior likely increases the error in our model, but as we've shown in this paper (see Figure 4) our regression model is reasonably robust. Among this paper's contributions, we consider the error analysis particularly important. It puts time-on-task modeling on a firm statistical footing that can only improve with refinements from future studies. As importantly, it will hopefully inspire researchers and educators to confidently use empirically based models. We caution that our model may not be the right choice, as it has only been tested in our context, but at least we have the tools to generalize.

We further caution that our model is not designed to model a single student. Because of insufficient sample size, we have not done an analysis of how the model varies across individual students. It seems intuitive that behavior will vary across students, as some students will spend more time thinking, others will tinker with the code, others will design on paper, and others will consult notes and the internet. While our model cannot resolve individual student behaviors (an important task for future work), it can give reasonable and reliable estimates in the aggregate.

We conclude by revisiting the happy finding that a 5-minute threshold performs reasonably well in our context in the aggregate. If all this paper has done has been to return back to an existing approach, one could ask what value our work adds. The answer is that we are no longer blindly using an arbitrary threshold. Rather, we can now confidently use a threshold (among the many proposed thresholds) that is based on empirical evidence.

References

[1] John Edwards, Kaden Hart, and Christopher Warren. 2022. A practical model of student engagement while programming. In *Proceedings of the 2022 ACM SIGCSE technical symposium on computer science education*.

[2] Judith P Frankmann and Jack A Adams. 1962. Theories of vigilance. *Psychological Bulletin* 59, 4 (1962), 257.

[3] Terry Judd. 2014. Making sense of multitasking: The role of Facebook. *Computers & Education* 70 (2014), 194–202.

[4] William A Kahn. 1990. Psychological conditions of personal engagement and disengagement at work. *Academy of management journal* 33, 4 (1990), 692–724.

[5] Ayaan M Kazerouni, Stephen H Edwards, and Clifford A Shaffer. 2017. Quantifying incremental development practices and their relationship to procrastination. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. 191–199.

[6] Vitomir Kovanovic, Dragan Gašević, Shane Dawson, Srećko Joksimovic, and Ryan Baker. 2015. Does time-on-task estimation matter? Implications on validity of learning analytics findings. *Journal of Learning Analytics* 2, 3 (2015), 81–110.

[7] Antti Leinonen, Henrik Nygren, Nea Pirttinen, Arto Hellas, and Juho Leinonen. 2019. Exploring the applicability of simple syntax writing practice for learning programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 84–90.

[8] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. 2022. Time-on-Task Metrics for Predicting Performance. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*. 871–877.

[9] Juho Leinonen, Francisco Enrique Vicente Castro, Arto Hellas, et al. 2021. Fine-Grained Versus Coarse-Grained Data for Estimating Time-on-Task in Learning Programming. In *Proceedings of The 14th International Conference on Educational Data Mining (EDM 2021)*. The International Educational Data Mining Society.

[10] Juho Leinonen, Leo Leppänen, Petri Ihantola, and Arto Hellas. 2017. Comparison of time metrics in programming. In *Proceedings of the 2017 acm conference on international computing education research*. 200–208.

[11] Kenneth Levenberg. 1944. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics* 2, 2 (1944), 164–168.

[12] Norman H Mackworth. 1948. The breakdown of vigilance during prolonged visual search. *Quarterly Journal of Experimental Psychology* 1, 1 (1948), 6–21.

[13] Jonathan P Munson. 2017. Metrics for timely assessment of novice programmers. *Journal of Computing Sciences in Colleges* 32, 3 (2017), 136–148.

[14] Christian Murphy, Gail Kaiser, Kristin Loveland, and Sahar Hasan. 2009. Retina: helping students and instructors based on observed programming activities. In *Proceedings of the 40th ACM technical symposium on Computer Science Education*. 178–182.

[15] Daniel W Newton, Jeffery A LePine, Ji Koung Kim, Ned Wellman, and John T Bush. 2020. Taking engagement to task: The nature and functioning of task engagement across transitions. *Journal of Applied Psychology* 105, 1 (2020), 1.

[16] Raja Parasuraman. 1979. Memory load and event rate control sensitivity decrements in sustained attention. *Science* 205, 4409 (1979), 924–927.

[17] H Mcllvane Parsons. 1974. What Happened at Hawthorne?: New evidence suggests the Hawthorne effect resulted from operant reinforcement contingencies. *Science* 183, 4128 (1974), 922–932.

[18] Thomas W Price, Neil CC Brown, Dragan Lipovac, Tiffany Barnes, and Michael Kölling. 2016. Evaluation of a frame-based programming editor. In *Proceedings of the 2016 ACM Conference on International computing education research*. 33–42.

[19] David R Thomson, Derek Besner, and Daniel Smilek. 2015. A resource-control account of sustained attention: Evidence from mind-wandering and vigilance paradigms. *Perspectives on psychological science* 10, 1 (2015), 82–96.

[20] Daniel Toll, Tobias Olsson, Morgan Ericsson, and Anna Wingkvist. 2016. Fine-grained recording of student programming sessions to improve teaching and time estimations. In *International journal of engineering education*, Vol. 32. Tempus Publications, 1069–1077.