A practical model of student engagement while programming

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

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

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

Abstract

We consider the question of how to predict whether a student is on or off task while working on a computer programming assignment using elapsed time since the last keystroke as the single independent variable. In this paper we report results of an empirical study in which we intermittently prompted CS1 students working on a programming assignment to self-report whether they were engaged in the assignment at that moment. Our regression model derived from the results of the study shows power-law decay in the engagement rate of students with increasing time of keyboard inactivity ranging from a nearly 80% engagement rate after 45 seconds to 30% after 32 minutes of inactivity. We find that students remain engaged in programming for a median of about 8 minutes before going off task, and when they do go off task, they most often return after 1 to 4 minutes of disengagement. Our model has application in estimating the amount of engaged time students take to complete programming assignments, identifying students in need of intervention, and understanding the effects of different engagement behaviors.

CCS Concepts

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

Keywords

CS1, Keystrokes, Engagement, Vigilance

ACM Reference Format:
John Edwards, Kaden Hart, and Christopher M. Warren. 2022. A practical model of student engagement while programming. In *The 53rd ACM Technical Symposium on Computer Science Education (SIGCSE '22), March 2–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366863

1 Introduction

How long does it take a student to complete a computer programming assignment? Instructors get answers from students that, anecdotally, often seem inflated. Knowing how long students actually take to complete assignments is useful in assignment and instruction design. Also of interest to practitioners, as well as to researchers, is the question of how long students remain on task before becoming disengaged. Other questions include how long students remain off task before returning, whether students get distracted easily, and whether an instructor can count on students engaging deeply with the code. Answering these questions is difficult. In their survey of Computing Education Research (CER) literature, Margulieux et al. state: "Measuring time was not common. Perhaps it should become more common to provide valuable insight into the learning process." [19]. The study reported in this paper attempts to advance our community's ability to make these various measurements of time.

Computer Science is a unique area of study because much of what students are doing to complete assignments is measurable by capturing keystrokes. As a result, some researchers have used keystroke counts to measure effort on assignments [6]. But since the total number of keystrokes as a measure of effort has little practical meaning to students and instructors, measuring effort in terms of time is still important. Time can be estimated from keystrokes: keystroke logs generally include a millisecond-resolution timestamp, and so latencies (elapsed time between keystrokes) can be summed together to estimate total time taken. But the problem is that the keystroke logs do not indicate when students become disengaged. A 12-hour latency between keystrokes more than likely indicates that the student has disengaged from the assignment and is taking a break, but what about a 10-minute latency or a 30-second latency? How do we know which latencies represent time working on the assignments and which do not?

Researchers have not come up with a good answer to this question. Various latency thresholds for stopping typing or becoming disengaged have been suggested [5, 6, 16, 28], but in every case the authors have acknowledged that, to at least some degree, the thresholds are arbitrary. Of course, we can never know, just from a separation of two keystrokes in time, whether a person is engaged in or disengaged from a task. However, we suggest that a statistical model could allow us to make useful estimates.

In this paper we present just such a model. Our model is built on data obtained from student self reports of engagement while working on a programming assignment. At intermittent times, a student is shown a window that asks if they are engaged or not. With careful sampling of times to prompt the student, we compile responses that we then fit a regression curve to. Using this model we can statistically estimate the probability that a student is engaged in their assignment using elapsed time since their last keystroke as the only independent variable.

Our objective in this study is to *determine an empirically based regression model for student engagement* using keystroke latency as the independent variable and with probability of engagement

SIGCSE '22, March 2–5, 2022, Providence, RI, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366863

as the dependent variable. We use this model to answer two research questions: [RQ1] *How long do CS1 students stay on task while programming?* and [RQ2] *How long do students remain disengaged?*

In this paper we explore related work (Section 2); explain how our data were collected (Section 3); present results, including the regression model and answers to research questions (Section 4); discuss results (Section 5); and conclude with implications of our model and various avenues for follow-on work (Section 6).

2 Related work

2.1 Vigilance and engagement

Two inter-related psychological constructs are relevant when considering student work habits while completing programming assignments: vigilance and engagement. Vigilance is defined as the ability to sustain attention to a task over a period of time, and the vigilance decrement describes how vigilance declines over time [8, 18, 26, 33]. Task engagement refers to the investment of physical, cognitive and emotional energy into a task [12, 24]. Though tasks used in vigilance studies typically involve passively monitoring a predominantly constant stimulus for change (e.g., a night security officer monitoring surveillance video of empty hallways), explanations of the vigilance decrement may have broader applicability. For example, maintaining attention to a task gets inherently more difficult over time as the mental resources required for the task become depleted [33], but at the same time, the effort required to maintain attention on a task is lessened by more engaging (interesting) tasks. All else being equal, more difficult tasks are associated with a greater vigilance decrement [10], and all else being equal, more engaging tasks are associated with a lower vigilance decrement [3]. This suggests that though the vigilance decrement is inevitable for any task, it can be mitigated by promoting task engagement, and in turn, task engagement can be inferred from the steepness of the vigilance decrement. In our study, we deviate from typical investigations of the vigilance decrement. Rather than examining how task performance decreases as a function of time on task, we examine how the probability of a student being on task decreases with time.

2.2 Engagement in CS education

Prior work in student engagement has largely been at the scale of the semester. For example, the National Student Survey of Engagement [1] asks questions like, "During the current school year, about how often have you asked questions or contributed to course discussions in other ways?" Other, similar surveys are used to gauge quality of education, in terms of engagement, from student reflection [32]. In studies of engagement in Computer Science Education, engagement generally isn't explicitly defined, and it is measured using indirect observation. For example, Ibanez et al. [11] study the effect of gamification on engagement. They measure engagement by whether students continue working on assignments after completing the requirements, as well as student reports of which assignments students preferred. Another example is the report of an ITiCSE working group on the effect of visualization on student engagement [23] where measurement of engagement is based on surveys and academic outcomes. The authors do note that time on task could be an instrument for evaluating engagement, though nothing more is said about it. Other studies are similar [31]. The majority of the literature seems to study engagement as defined by Axelson and Flick: "The phrase 'student engagement' has come to refer to how involved or interested students appear to be in their learning and how connected they are to their classes, their institutions, and each other" [2].

In our study we explore a different type of engagement. A good theoretical definition of engagement for the purposes of our study is given by Kahn, who defined it, in the context of work roles, as "the harnessing of organization members' selves to their work roles; in engagement, people employ and express themselves physically, cognitively, and emotionally during role performances" [12]. We formalize this definition for our study to be: "a student is engaged at a given moment if, when asked if they are on task, their honest and self-aware answer would be 'yes.'"

Some CER work has been done looking at activity while programming. Munson calculated time taken on a programming assignment simply by subtracting the time of the first compilation event from the last [21]. Toll et al. used a combination of keystrokes, mouse movements, and other events to estimate student activity at different levels of granularity [35] and used different events to estimate different activities, e.g., using mouse movements but not keystrokes suggests reading or navigating activity but not editing. Other works simply sum elapsed time between keystroke, compile, or other events as long as the elapsed time doesn't exceed a threshold beyond which students are considered disengaged. The need for a meaningful model of engagement is graphically demonstrated in the lack of agreement between these thresholds, which range from 60 seconds [28] to 3 minutes [15] to 30 minutes [22] to one hour [13].

2.3 Predicting emotion from keystrokes

A number of studies have looked at predicting emotional states, including engagement, using keystrokes and video. Bixler and D'Mello [4] asked participants to write natural language essays for 10 minutes. After the conclusion of the writing, participants were shown a video of their face along with corresponding screen capture video and were asked to make judgments as to their affective state at 15 second intervals. Participants were instructed to select one or more relevant affective states among 15 choices, including engagement, boredom, surprise, and happiness. Keystroke features were used to predict the student's affective state. Only 35% of responses included "engaged" as the affective state indicating that students may have used "interested" or "excited about" as synonyms for "engaged".

Epp et al. [7] recorded keystrokes while participants did everyday tasks on their personal windows computers at home. Every so often participants were prompted to answer 15 questions on their emotional state using the Likert scale. Engagement was not included in the list of emotional states. Kołakowska [14] used a controlled study to predict, using keystroke features, stress among students. They had two groups write a computer program, where only the test group had a time limit.

Work has been done to predict affect based on both keystrokes and video. In the work of Tiam-Lee and Sumi [34], students were

observed in a lab setting rather than at home. Similar to Bixler and D'Mello [4], this was based on self-report as students watch themselves programming and declare affective state among the choices of engaged, confused, frustrated, bored, and neutral. Students were given definitions of each state, with engaged being defined as "you are immersed in the activity and enjoying it." Participants were significantly more likely to be modifying the code while they were engaged.

3 Methods

We conducted a study in which we asked CS1 students to self-report whether they were engaged in working on the programming assignment at the moment they are prompted.

3.1 Prompts

Participating students were asked to install a plugin to the PyCharm IDE and to complete their Python programming assignments using PyCharm. The PyCharm plugin both collects keystroke logs and prompts students to self-report engagement as follows: after varying periods of inactivity, or "query times," of .75, 1, 1.5, 2.5, 4.5, 8.5, 16.5, and 32.5 minutes, the plugin shows a window to the student that asks the question: "What were you doing immediately before this appeared?" The window has two button responses, "Working on something related to my assignment" and "Doing something else." The prompt window is modal, so the student is required to respond before continuing work on their assignment. The student's response is logged in the keystroke log file. The prompt is intended to be as non-intrusive as possible, with between 2 and 5 prompts per assignment. Our approach is similar to the Experience Sampling Method [9] in that we collect samples *in situ*, with participants in their own environment working on their own time rather than in a lab setting or being asked about engagement after the fact.

Students may not respond to the prompt window immediately, especially if they are using another computer application, on their phone, or away from their computer when the prompt appears. We considered an optimization of closing the prompt and marking the student off task if the student didn't respond within a few seconds. This idea was discarded, however, because students may still be engaged even if they are not working in the programming IDE – they may be searching for help on the internet or consulting the textbook.

3.2 When to prompt

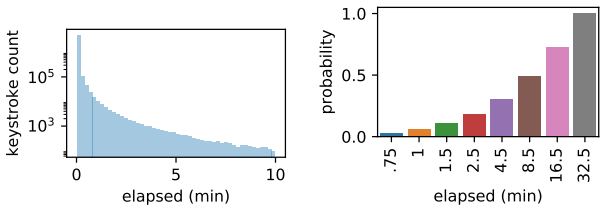
Our approach to determine when to show the query window is as follows: after each keystroke, use probability 0.7 to determine if the keystroke will start a query timer. If the keystroke is chosen to start a timer, determine the amount of time for the timer using a Cumulative Density Function (CDF). If the timer expires before another keystroke is executed then we prompt the student. See Algorithm 1.

We desire to prompt evenly between the different query times, e.g., we wish to prompt students after 32 minutes of inactivity roughly the same number of times that we prompt students after 45 seconds of inactivity. Sampling query times uniformly would result in far more prompts after short times of inactivity. The reason is because of the distribution of keystroke latencies (the periods of time between pairs of keystrokes): the large majority of keystrokes are executed within a couple of seconds of the preceding keystroke (Figure 1a). If we start a timer for two seconds, there's a good chance it will complete but if we start a timer for 32 minutes, there is very little probability of its completion before another key is pressed. Thus, to achieve uniform sampling of times at which students are prompted, we must allocate far more keystroke timers to larger query times than to smaller.

Algorithm 1 Algorithm for determining when to prompt

```
function onKeystroke:
  cancel active timer
  if rand() < 0.7
    time = CDF(rand())
    timer(time, callback)

function callback:
  show prompt window
```



(a) (b)

Figure 1: (a) Histogram of elapsed time between keystrokes. Note that the y axis is on the log scale. (b) Sampling function used to determine elapsed time to set the timer for. The goal is to get a distribution of prompts that is uniform across elapsed times.

To calculate a sampling function for the different query times we start with a histogram of keystroke latencies calculated from keystroke data from a previous study (citation elided for double-blind review). See Figure 1a. We crop the histogram at a minimum of 30 seconds and then bin into query times. We then compute the cumulative sum in reverse and normalize it. We then take the multiplicative inverse of the probability mass function, normalize, and once more take the cumulative sum to compute the Cumulative Density Function (CDF) that we use to sample times for the prompt timer. The CDF probabilities are given in Figure 1b and Table 1. We simulated times at which students are prompted using our existing keystroke data and achieved a nearly uniform distribution.

3.3 Participants

Participants in this study are students in a CS1 course from a mid-sized university in the United States. Students were given the opportunity to opt in or out with no benefit or penalty. Of the 36 students that opted in to our study, 15 installed the PyCharm plugin. Data was collected over the course of 6 programming assignments. The

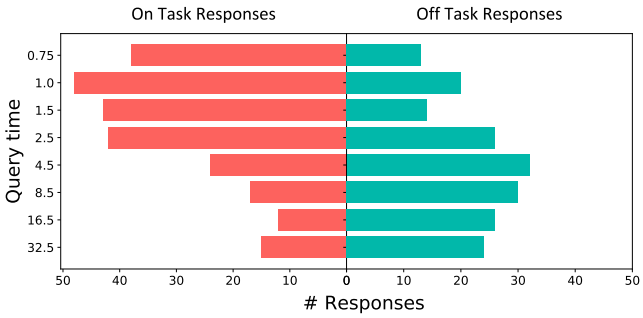


Figure 2: Distribution of responses across query times.

Elapsed (min)	CDF	Off	On	N	Engagement rate
0.75	0.03	13	38	51	0.75
1.0	0.06	20	48	68	0.71
1.5	0.11	14	43	57	0.75
2.5	0.18	26	42	68	0.62
4.5	0.31	32	24	56	0.43
8.5	0.49	30	17	47	0.36
16.5	0.72	26	12	38	0.32
32.5	1.00	24	15	39	0.38

Table 1: Student responses to the engagement query. *Elapsed* is the number of minutes elapsed since the last keystroke at the time of the query. *CDF* are the values of the CDF sampling function. *Off*/*On* are the number of off task/on task responses and *N* is the total number of responses. *Engagement rate* is *On*/*N*.

study was conducted under IRB protocol (ID elided for double-blind review).

4 Results

We collected 81 submissions from the 15 participants, for an average of 5.4 submissions per student. The distribution of responses to the engagement query across inactivity times is shown in Figure 2 and Table 1. In the table we see that the prompts were reasonably well spread across query times from 38 prompts after 16.5 minutes to 68 prompts at each of 1 and 2.5 minutes. Of the 424 total responses to the engagement query, 239 reported being on task and 185 reported being off task. Students were queried, on average, about 5 times per assignment.

4.1 Model of engagement rate

Table 1 reports engagement rate for each query time. Engagement rate is the ratio of on task responses to engagement events. Figure 3 shows a plot of the sample data along with the least squares best fit power-law curve using the Levenberg-Marquardt algorithm [17]. The curve is given as

$$P(t) = 0.73t^{-0.25} \quad (1)$$

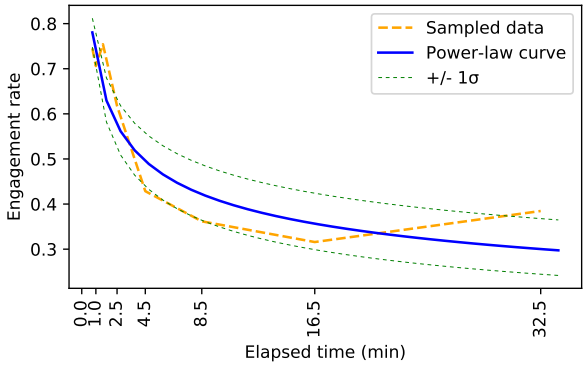
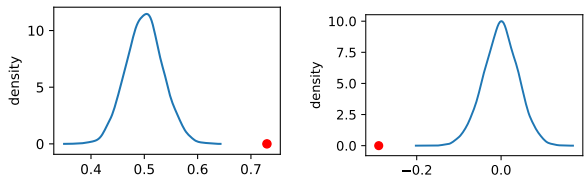


Figure 3: Engagement rate from 45 seconds to 32.5 minutes of inactivity. The curve from the sampled data and the power-law curve, with curves at $\pm 1\sigma$, are shown.

where t is elapsed time since the last keystroke and $P(t)$ is the probability that the student is engaged. Standard deviations are 0.039 for the coefficient a and 0.043 for the exponent b. The 50% threshold, according to the model curve, is at 3:41. In other words, according to this model, after about 3 1/2 minutes of inactivity we can expect about half of the students to have directed their engagement away from the programming assignment.

The data are consistent with the hypothesis that engagement rate decreases with increasing times of inactivity. We tested the hypothesis as follows: we fit a power-law curve to the data (Figure 3). We then, using a uniform distribution, randomly assigned *Off* and *On* responses to the sampled data and fit a new curve to the randomized data. We performed this randomization 10,000 times and fit a power-law curve $\hat{P}(t) = \hat{a}t^{\hat{b}}$ to each. Across the model curves, the constant $\hat{a}$, as expected, has a mean of 0.50 and is normal (D'Agostino-Pearson statistic=0.69, p=0.71). Similarly, the exponent $\hat{b}$ has a mean of 0.00 and, because it is a nonlinear term, is not normally distributed (D'Agostino-Pearson statistic=15.58, p=0.0004). See Figure 4. Even after 10,000 random reassignments, our model constant a and exponent b that were fit on the sample data are both outside the range of $\hat{a}$ and $\hat{b}$, suggesting that we can reject the null hypothesis that on task and off task responses from students are randomly distributed relative to time of inactivity.

Equation (1) is a regression model for probability of engagement that is easily implementable, even *in situ*, using only a single independent variable, that of time since the last keystroke. We emphasize that our model is one of regression, not classification. Rather than predicting if a student is on or off task, it predicts the *probability* that the student is on task. We use this approach for two reasons. The first is that the model itself lends insight into student behavior. For example, we can now say things like, "according to the model, if a student hasn't pressed a key in 2 minutes then they are 40% likely to be disengaged." The other reason to use regression rather than classification is to avoid firm separating hypersurfaces and allow randomness. If we used, say, logistic regression, then every pause of 45 seconds would be classified as on task and every pause of 4 minutes would be off task. As our model is designed, we



(a) Constant coefficient $\hat{a}$ (b) Exponent $\hat{b}$

Figure 4: Distribution of the power-law curve coefficient and exponent given 10,000 randomizations of student responses. Values from the empirically sampled data are marked as red points on the charts at $a = 0.73$ and $b = -0.25$.

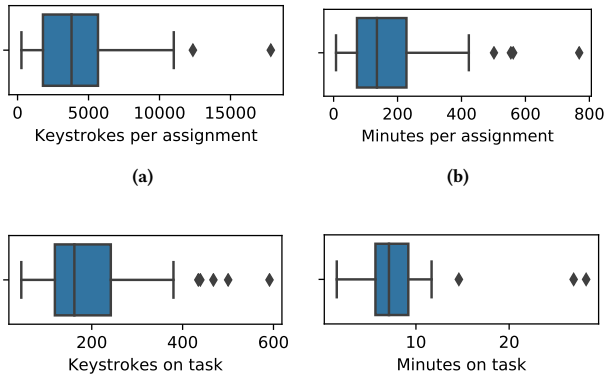


Figure 5: Statistics on length of assignment and length on task, as measured in keystrokes and minutes.

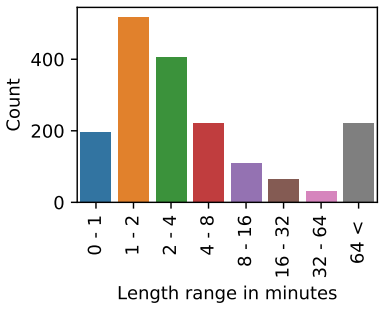


Figure 6: Distribution of length of time students remain off task after disengaging from programming.

can use the probabilities to randomly determine engagement. That way, 80% of 45-second pauses will be predicted to be on task and 20% will be predicted to be off.

4.2 Length of engagement

Using our model we are able to analyze how long students are engaged in their assignments before going off task. We gather all keystrokes from a student submission and compute digraph latencies, or elapsed time between keystrokes. We then use our model to determine the probability that, given the latency, the student remained engaged following a keystroke. Because of considerations discussed in Section 5, we assign a probability of 1 if the latency is less than 45 seconds and 0 if greater than 2 hours. Once probabilities are assigned, we assign the attribute *onTask* to keystrokes, with a value of *true* assigned if a random number is less than the probability, and *false* otherwise. After assigning the *onTask* attribute to all keystrokes in all submissions we can compute various statistics. See Figure 5. In our data, the median number of keystrokes for an assignment is 3803. With our model we can compute the number of minutes students worked on their assignments more accurately than just adding up all the latencies: by excluding latencies for keystrokes that were modeled as off task we get a median of 184 minutes, or just over 3 hours, of actual (though not necessarily continuous) time on task for each assignment.

The most interesting statistic, and answer to research question RQ1, is the median length of time students remain engaged before going off task, which is 8.5 minutes, or about 163 keystrokes. This means that students generally work on their programming assignments for only 8-9 minutes before becoming distracted or otherwise disengaged. This means that over the course of the 3 hours of programming, students go off task over 20 times. In Figure 6 we see how long students are off task, answering research question RQ2. Among the lengths shown in the figure (binned in powers of two), the most common length of time for a student to be off task is between one and two minutes and 51% of students remain off task for between 1 and 4 minutes.

5 Discussion

5.1 Sampling bounds

We sampled student engagement after at least 45 seconds of inactivity. In order to achieve our uniform sampling rate (see Figure 2) we are required to sample far more of the long inactivity rates than the short (Section 3.2), and so adding shorter inactivity rate sampling would require a much larger number of keystrokes to sample from. Our participant pool of 15 students didn't admit a large enough number of keystrokes to sample at times lower than 45 seconds. However, we expected most of the range of probabilities to be covered between 45 seconds and 32 minutes, and this was indeed the case, with 45 seconds sampling at about 80% engagement rate down to about 30% engagement rate at 32 minutes. Sampling at lower than 45 seconds should give insight into behaviors near 100% engagement rate, though we expect that sampling at higher than 32 minutes may not add much insight as the behavior appears asymptotic beyond 32 minutes.

5.2 Adjustment for bias

There are reasons to consider our model an upper bound of engagement. The first reason is student bias toward reporting as being

engaged, even if they aren't. This could be caused by students answering dishonestly and also by lack of self-awareness resulting in the likelihood that more students were disengaged than reported. Another reason to consider the model an upper bound is participant reactivity, discussed in Section 5.3. Because students knew they were being observed they may have stayed on task more than they normally would have. As discussed in Section 5.3, we expect this effect to be somewhat low. Note that answer bias and reactivity remain even with large sample sizes, so the model would need to be corrected for these effects, if possible, no matter the sample size. We propose a possible correction based on the observation that the sampled data appears to have a non-zero lower asymptote of about 30% in our data (see Figure 3) when the engagement rate should actually approach zero with increasing time. In sampling longer inactivity times, some percentage of students may still respond as being on task even if they are not. We suggest the following correction: if the asymptote is at, say, 30% then we could say that $f = 0.3$, or 30% of students will always say that they are engaged. We can correct the sampled engagement rate $\hat{R}$ for a given query time t as

$$R(t) = \frac{\hat{R}(t) - f}{1 - f} \tag{2}$$

where $R(t)$ is the corrected engagement rate. Applying the correction to our model yields the curve in Figure 7. After correction, the asymptote correctly appears to be near zero while behavior at small times of keyboard inactivity is close to the original data.

5.3 Hawthorne effect

An additional consideration in this research is the phenomenon of participant "reactivity," or the Hawthorne effect. Studies have shown that research subjects change their behavior when they know they are being observed [20, 25, 27, 30]. The Hawthorne effect may be exhibited in our study through higher rates of engagement. Note that this is different from students answering that they were on task when they were actually off task. The Hawthorne effect indicates that students may actually have been on task more than they would have had they not been observed. However, the Hawthorne effect is fleeting. One cannot indefinitely improve productivity merely by letting the subjects know they are being watched. An advantage of the sampling approach used in our study is that it is less intrusive than other approaches – students are not required to be in a lab, they use their own computers, they work on their own schedule, and they aren't required to be recorded by a camera, thus lessening the effects of the Hawthorne effect.

5.4 Model curve

We use a power-law regression curve to model engagement. One issue with this curve is that it doesn't have an upper asymptote, and so smaller times of inactivity will result in probabilities of engagement greater than 1. This is not a problem in the scope of the present work since we clamp the engagement probability to 1 at times below 45 seconds of inactivity, but for generalizability a more suitable curve should be used. A curve that may be a good candidate is the generalized logistic curve or Richards' curve [29]. We didn't use this curve for our model because of the non-zero lower asymptote in our data. However, after applying the bias correction (Equation 2) the generalized logistic curve fits nicely using Richards' equation

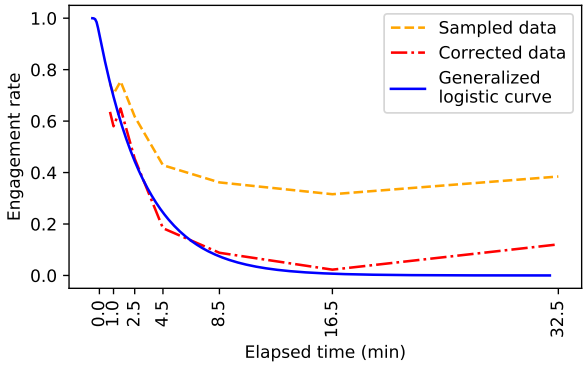


Figure 7: Engagement rate modeled with a generalized logistic function on the adjusted data. Note the upper asymptote at engagement rate of 1.0.

$$y = \frac{1}{(1 + Qe^{-B(x-M)})^{1/v}} \tag{3}$$

with $Q = 50$, $B = -17$, $M = 0.01$, and $v = 57$. See Figure 7.

5.5 Threats to validity

Participants in our study were drawn from a CS1 course taught in Spring 2021. The course was taught online because of COVID-19, so irregularities in instruction may have been present. Students opted into the study knowing that their keystrokes would be recorded, so behavior may have been influenced by the Hawthorne effect (Section 5.3), and students who did not opt into the study may exhibit different engagement behavior. Sample size should be considered when generalizing our results.

6 Conclusions

Our approach to modeling student engagement is easily implementable using recorded keystrokes and not only allows us to predict student engagement, but the model itself gives insights into students' allocation of attention while programming. We have proposed a model for analyzing how long students actually take on their assignments (useful to instructors), how long students remain engaged before going off task (useful to instructors, language designers, and researchers) and how long students are disengaged once they go off task (useful to researchers). We find that students remain on task for about 8 minutes at a time (RQ1) and return to engagement after 1-4 minutes in most cases (RQ2). These measures are useful to students who can gain improved self-awareness by seeing a model of their engagement behavior, especially when compared to that of their peers.

Predicting disengagement will allow us to not only assess student performance after the fact, but allow us to design interventions that can help students proceed when they get stuck. Simply adding a prompt when keystroke analysis suspects a student going off-task could be beneficial. And if the prompt were context-aware, specific

help could be given, such as a short practice activity if a student is stuck on a particular syntactic construct.

Our model is likely an upper bound of engagement. We expect that the median time-on-task is less than 8 minutes. While controlling for honesty, self-awareness, and the Hawthorne effect is difficult, the model itself, with its non-zero lower asymptote, can give direction for correcting for measurement error. Until such a correction can be studied in more detail, our model as given here is a reasonable first approximation.

In this paper we have presented a model for engagement as well as derived statistics, such as the median time-on-task. What is absent from our discussion is whether engagement behavior following this model is conducive to learning or not. Such a discussion is outside the scope of our work, but it is important to address when designing and implementing instructional changes based on the model. One straightforward way to determine the benefits of different engagement behaviors would be to correlate assignment and course grades with average time-on-task and similar features. Our work presented in this paper gives a model and empirical techniques that may be useful in such studies.

Our work may be generalizable to other disciplines. For example, keystroke logs of essay writing in literature courses [4] could be used to model engagement. While the parameters of the engagement curve would likely be different, the techniques for determining and interpreting the model could be the same.

References

[1] 2021. National Survey of Student Engagement. https://nsse.indiana.edu/nsse/survey-instruments/us-english.html (accessed 27 July, 2021).

[2] Rick D Axelson and Arend Flick. 2010. Defining student engagement. *Change: The magazine of higher learning* 43, 1 (2010), 38–43.

[3] Evelyn Barron, Leigh M Riby, Joanna Greer, and Jonathan Smallwood. 2011. Absorbed in thought: The effect of mind wandering on the processing of relevant and irrelevant events. *Psychological science* 22, 5 (2011), 596–601.

[4] Robert Bixler and Sidney D'Mello. 2013. Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits. In *Proceedings of the 2013 international conference on Intelligent user interfaces*. 225–234.

[5] Paul S Dowland and Steven M Furnell. 2004. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In *IFIP International Information Security Conference*. Springer, 275–289.

[6] John Edwards, Joseph Ditton, Dragan Trninic, Hillary Swanson, Shelsey Sullivan, and Chad Mano. 2020. Syntax exercises in CS1. In *Proceedings of the 16th Annual Conference on International Computing Education Research* (Dunedin, New Zealand) *(ICER '20)*.

[7] Clayton Epp, Michael Lippold, and Regan L Mandryk. 2011. Identifying emotional states using keystroke dynamics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 715–724.

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

[9] Joel M Hektner, Jennifer A Schmidt, and Mihaly Csikszentmihalyi. 2007. *Experience sampling method: Measuring the quality of everyday life*. Sage.

[10] William S Helton and Paul N Russell. 2011. Working memory load and the vigilance decrement. *Experimental brain research* 212, 3 (2011), 429–437.

[11] Maria-Blanca Ibanez, Angela Di-Serio, and Carlos Delgado-Kloos. 2014. Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on learning technologies* 7, 3 (2014), 291–301.

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

[13] 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.

[14] Agata Kołakowska. 2016. Towards detecting programmers' stress on the basis of keystroke dynamics. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 1621–1626.

[15] 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.

[16] Juho Leinonen, Krista Longi, Arto Klami, and Arto Vihavainen. 2016. Automatic inference of programming performance and experience from typing patterns. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 132–137.

[17] 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.

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

[19] Lauren Margulieux, Tuba Ayer Ketenci, and Adrienne Decker. 2019. Review of measurements used in computing education research and suggestions for increasing standardization. *Computer Science Education* 29, 1 (2019), 49–78.

[20] Jim McCambridge, John Witton, and Diana R Elbourne. 2014. Systematic review of the Hawthorne effect: new concepts are needed to study research participation effects. *Journal of clinical epidemiology* 67, 3 (2014), 267–277.

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

[22] 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.

[23] Thomas L Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, et al. 2002. Exploring the role of visualization and engagement in computer science education. In *Working group reports from ITiCSE on Innovation and technology in computer science education*. 131–152.

[24] 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.

[25] Ryan Olson, Lindsey Hogan, and Lindsey Santos. 2006. Illuminating the history of psychology: tips for teaching students about the Hawthorne studies. *Psychology Learning & Teaching* 5, 2 (2006), 110–118.

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

[27] 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.

[28] 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.

[29] FJ Richards. 1959. A flexible growth function for empirical use. *Journal of experimental Botany* 10, 2 (1959), 290–301.

[30] Fritz Jules Roethlisberger and William J Dickson. 2003. *Management and the Worker*. Vol. 5. Psychology press.

[31] Laura A Schindler, Gary J Burkholder, Osama A Morad, and Craig Marsh. 2017. Computer-based technology and student engagement: a critical review of the literature. *International journal of educational technology in higher education* 14, 1 (2017), 1–28.

[32] Jane Sinclair, Matthew Butler, Michael Morgan, and Sara Kalvala. 2015. Measures of student engagement in computer science. In *Proceedings of the 2015 ACM conference on innovation and technology in computer science education*. 242–247.

[33] 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.

[34] Thomas James Tiam-Lee and Kaoru Sumi. 2019. Analysis and prediction of student emotions while doing programming exercises. In *International conference on intelligent tutoring systems*. Springer, 24–33.

[35] 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.