Pausing While Programming: Insights From Keystroke Analysis

Raj Shrestha
raj.shrestha@usu.edu
Utah State University
Logan, Utah

Juho Leinonen
juho.2.leinonen@aalto.fi
Aalto University
Espoo, Finland

Albina Zavgorodniaia
albina.zavgorodniaia@aalto.fi
Aalto University
Espoo, Finland

Arto Hellas
arto.hellas@aalto.fi
Aalto University
Espoo, Finland

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

ABSTRACT

Pauses in typing are generally considered to indicate cognitive processing and so are of interest in educational contexts. While much prior work has looked at typing behavior of Computer Science students, this paper presents results of a study specifically on the pausing behavior of students in Introductory Computer Programming. We investigate the frequency of pauses of different lengths, what last actions students take before pausing, and whether there is a correlation between pause length and performance in the course. We find evidence that frequency of pauses of all lengths is negatively correlated with performance, and that, while some keystrokes initiate pauses consistently across pause lengths, other keystrokes more commonly initiate short or long pauses. Clustering analysis discovers two groups of students, one that takes relatively fewer mid-to-long pauses and performs better on exams than the other.

CCS CONCEPTS

• **Social and professional topics** → *Computing education.*

KEYWORDS

pauses, pausing, breaks, keystroke data, digraphs, programming process data

ACM Reference Format:
Raj Shrestha, Juho Leinonen, Albina Zavgorodniaia, Arto Hellas, and John Edwards. 2022. Pausing While Programming: Insights From Keystroke Analysis. In *44nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3510456.3514146

1 INTRODUCTION

Pausing during work is a natural behaviour for a person which allows them to reflect on their task, plan what they are going to do next, revise, or take a rest. Pauses, however, can also be initiated by distraction and lead to hindering one's working process. In this paper we aim to study pauses that students take while doing programming projects in Introductory Computer Programming (CS1). In a typical CS1 course instructors and graders look at the final program a student has produced for assessment, but there is no indication from the code as to how the student wrote it. It is possible that information on the number and types of pauses students take could be mined to shed more light on processes that underlie programming in CS1.

Since pauses can be products of various activities (e.g., thinking, disengaged), we investigate whether pause length may hold insights into these activities. Our study features four types of pauses. *Micro* pauses (2-15 seconds) which may indicate the student is thinking about the code on a low level or "locally" (e.g. syntax). *Short* pauses (15 seconds to 180 seconds) may indicate that the student is involved in a higher-level process such as planning or revision. *Mid* pauses (3-10 minutes) may indicate that the student is disengaged or that they are going to an outside resource for help (e.g. YouTube, Stack Overflow, or course materials). Finally, *long* pauses (greater than 10 minutes) may indicate disengagement from the task.

In this paper, we look at relative number of pauses over the course and correlate with outcomes (exam score). That is, if a student takes more or fewer pauses relative to their total number of keystrokes, could it suggest their better or worse course performance? Many pauses that are very small may indicate that the student is planning their typing carefully rather than writing without a clear direction and may have a positive correlation with performance. Many medium-sized pauses may indicate the same thing, but the measurement may be confounded by students who are easily distracted. Many long pauses may indicate distraction.

The research questions we investigate in this paper are:

- RQ1 Is there a correlation between the relative number of pauses a student takes and their performance (exam score)?
- RQ2 What groups of students exist when clustering on pausing behavior?
- RQ3 What events initiate a pause and how does this correlate with the performance of the student?

We seek to answer these research questions using analysis of keystroke data collected in two CS1 courses at different universities on different continents. The closest matches to our work come from two different research streams. One of the research streams has studied syntax errors and identified pauses or breaks when correcting such errors (e.g. [2, 10, 16, 30, 59]). The other research stream has focused on the analysis of keystroke data, which has been shown to be effective at gaining insights into student behavior [26, 32, 37]

ICSE-SEET '22, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9225-9/22/05.
https://doi.org/10.1145/3510456.3514146

as well as predicting student outcomes [19, 40, 55]. However, little work exists at the intersection of these research streams, where our work lies. The novelty of our analysis is that we are looking less at typing behavior and more at pausing behavior, which might indicate more or less of a student's cognitive processing, examining of external resources, or disengagement.

In this paper, we report on several findings: those students who pause more often generally show worse performance in the course; students who take more *shorter* pauses perform better than students who take more *longer* pauses; *mid* pauses have the strongest negative correlation with exam scores; specific events that precede pauses have a more evident correlation with performance and thus allow conjecture about the underlying processes.

2 RELATED WORK

2.1 Pausing behavior

In the research literature, pauses are prevalently discussed in relation to language production – written or oral speech/narration [13, 21, 49], language translation [33] and editing [43]. Relying on cognitive psychology, researchers associate pauses with cognitive processing of various types [43]. For example, in writing, it is thought that pauses at higher-level text units (e.g. between sentences) are likely to be conditioned by higher-level subprocesses, such as planning and organization of the content, whereas pauses at lower-level units (e.g. between and inside words) – by lower-level subprocesses, such as morphology encoding and retrieval of items from one's memory [54].

A pause is also considered to signify cognitive effort imposed by language production mental processes [33, 34]. Butterworth [11] hypothesised that the more cognitive operations are needed for output production, the more pauses arise. Damian and Stadthagen-Gonzalez [14] and Révész et al. [48] argued that the length of a pause taken before a textual unit reflects the mental effort made with respect to production of this forthcoming unit. Reflecting on pausing in post-editing, O'Brien [43] concluded that pausing patterns do, to an extent, indicate cognitive processing. However, they are ultimately subject to individual differences.

Pausing has also gained attention in the medical training domain. Lee et al. [35] studied pauses and their relation to cognitive load. Students had to complete a medical game that simulated emergency medicine under two conditions: pause-available and pause-unavailable. In the study, pauses of two types were identified: reflection and relaxation. The first type is argued to enhance task-related cognitive processes and therefore increase mental effort (or cognitive load). The second type reflects the opposite process when the load lowers due to the resting state.

That being said, pauses during problem-solving can signify not only ongoing mental work but a suspension of it caused by various things. Gould [23] defines three types of interruptions: those that are relevant to the task and reinforce processes in the working memory, those that are relevant to the task and interrupt processes in the working memory, and those that are not relevant to the task. The author states that how these interruptions affect the following resumption and productivity depends on "contextual factors at the moment of interruption". Borst et al. [9] also relate the length of interruption to the time of subsequent resumption and number of possible errors in a task.

2.2 Pausing behavior in CER

Pausing behavior has been studied in Computing Education Research (CER) both directly and indirectly in the context of computer programming. Similar to written language, where pauses between and within sentences are likely conditioned by different subprocesses [54], code writing has its own milestones and units of different level of complexity. When considering the mental effort needed to write code, one stream of research has focused on identifying and discussing plans and schemas for programming [15, 51]. It has been suggested that programmers who know the solution to a problem write their solution in a linear manner, while solving a new problem is done using means-ends analysis with the use of existing related schemas [15, 51]. Over time and through practice, accumulation and evolution of schemas allow programmers to solve problems more fluently, and also to learn to solve new problems with more ease [51, 58].

As discussed in Section 2.1, pauses can signify cognitive effort and are a natural part of the learning process. In programming however, an additional contributor to pauses, especially for novice programmers, are syntactic and semantic errors related to writing computer programs with the chosen programming language. These errors may be highlighted by the programming environment in use—similar to a word processing engine that shows spelling errors—as programming environments often highlight errors in program code, but they may be also visible through specific actions such as compiling the source code. These errors have been discussed especially in the context of Java programming, where researchers have studied the frequency of different types of errors [2, 10, 16, 30, 59] and the amount of time that it takes to fix such errors [3, 16]. Denny et al. [16] and Altamdri and Brown [2], for example, have noticed that there are significant differences in the time that it takes to fix specific errors, and that over time students learn to avoid specific errors [2]. At the same time, the granularity of the data used in the analysis has an influence on the observed errors [59] – different data granularity will lead to different observed syntax errors. In practice, collecting typing data with timestamps can provide more insight into the programming process over snapshot or submission data [60].

When considering syntax errors, pauses, and performance, the ability to fix syntax errors between successive compilation events has been linked with students' performance in programming [31], although it is unclear what the underlying factors that contribute to the observation are [46]. Including an estimate on the amount of time that individual students spend on fixing specific errors can increase the predictive power of such models [1, 64], highlighting the effect of time (or pause duration) on the learning process.

While the previous examples are specific to syntax errors and time, little effort has been invested into looking into pauses in programming. Perhaps the closest prior work to our work is that of Leppänen et al. [41] who studied students' pausing behavior in two courses, and identified that a larger quantity of short (1-2 second) pauses was positively correlated with course outcomes, while a larger quantity of longer (over 1 minute) pauses was negatively

correlated with course outcomes. Our work builds on this by—in addition to correlation analysis—looking at pausing behavior over different contexts and also by investigating which characters precede pauses. Leppänen et al. hypothesized that one explanation for the correlation between long pauses and poorer course outcomes could be related to task switching between reading the course materials and solving the programming problems, but noted also that the pauses from writing code could be construed as instances of the student engaging in planning, reviewing, and translating the next ideas into code. Another possible hypothesis is related to differences in cognitive flexibility, i.e. the ability to fluently switch between two tasks; for example, Leinikka et al. [36] observed that students with better cognitive flexibility are faster at solving programming errors, although they did not observe links between cognitive flexibility and introductory programming course exam outcomes.

2.3 Typing and performance in programming

In CER, a multitude of data sources has been used for identification of factors and behaviors that contribute to course outcomes [25] – clicker-data [42, 47], programming process snapshots [1, 12, 31, 39, 64], background questionnaires and survey data [7, 8, 53, 57, 65], and so on, but our focus is on keystroke data collected from programming environments [27, 28].

Keystroke data, or typing data, has been used, for example, for predicting academic performance [19, 40, 55], for detecting emotional states such as stress [20, 32], and for identifying possible plagiarism [26].

Much of the analyses of typing data that relate to students' performance has focused on between-character latencies, i.e. the time that it takes for the student to type two subsequent characters. This analysis has often focused on small latencies, as pauses have been considered as noise. For example, both Leinonen et al. [40] and Edwards et al. [19] used 750 milliseconds as an upper boundary for the between-character latencies. In general, these studies have found that faster typing correlates with previous programming experience and performance in the ongoing programming course.

Not all characters are equally important, however. For example, Leinonen et al. [40] identified differences in the time that moving from 'i' to '+' took for novices and more experienced students, while differences in some other character pairs were more subtle. Similarly, Thomas et al. [55] noted that the use of control functionality (e.g. using control and C keys) in general was slower than the use of e.g. alphanumeric keys, and the use of special keys such as delete and space was also slower than alphanumeric keys. Acknowledging that some of these latencies may be also influenced by the keyboard layout, they hypothesised that some of the latencies may be influenced by the thought processes related to the ongoing problem solving [55]. Our work builds on this prior work by examining which characters precede pauses, i.e. whether all characters are equally important when analyzing pausing behavior.

3 METHODOLOGY

3.1 Context and data

Our study was conducted in two separate contexts for purposes of generalization of the results.

3.1.1 University A. University A is a mid-sized public university in the Western United States. In a 2019 CS1 course, students used a custom, web-based Python IDE called *Phanon* [17] for their programming projects. *Phanon* logged keystrokes and compile/run events. Five programming projects, one per week, were assigned to the students during the study period. Each project consisted of two parts: a text-based mathematical or logical problem, such as writing an interest calculator; and a turtle graphics-based portion requiring students to draw a picture or animation, such as a snowman. A midterm exam was administered between the fourth and fifth project. There were three sections of the course all taught by the same instructor. Projects and instruction were the same for all three sections. In-person instruction was conducted three times per week. At the beginning of the semester students were given the opportunity to opt into the study according to our Institutional Review Board protocol, and this paper uses data only from students who opted in. The course was identical for students who chose to participate in the study and those who chose not to.

Gender information on participants was not collected, but in the course participants were recruited from, 19% self-identified as women and 81% self-identified as men. No information on previous programming experience, race or ethnicity was available for this study.

3.1.2 University B. University B is a research-oriented university in Northern Europe. The data for this study was collected from a 7-week introductory programming course in the Fall of 2016. The introductory programming course is given in Java and it covers the basics of programming, including procedural programming, object-oriented programming and functional programming. During each week of the course, there was a 2-hour lecture that introduced the core concepts of the week using live coding. The emphasis in live coding was in providing examples of how programming problems were solved with the concepts learned during the week, and in helping to create a mental model of an abstraction of the internals of the computer as programs are executed (introduction of variables, changing variable values, objects, call stack in a line-by-line fashion). In addition to the lectures, 25 hours of weekly support was available in reserved computer labs with teaching assistants and faculty.

The programming assignments in the course are completed using a desktop IDE accompanied with an automated assessment plugin [61] that provides students feedback as they are working on the course assignments. Combined with an automatic assessment server, the plugin also provides functionality for sending assignments for automatic assessment. In addition to the support and assessment, the plugin collects keystroke data from the students' working process, which allows fine-grained plagiarism detection [26] and makes it possible to provide more fine-grained feedback on students' progress. Students can opt out of the data collection if they wish to do so; the data collection was conducted according to the ethical protocols of the university.

Out of the 244 students at University B included in the study, approximately 40% self-identified as women and 60% as men. No information on previous programming experience nor race or ethnicity was available for this study.

Attribute	University A	University B
Instruction	Lectures w/sections	Lectures, sessions
Language (prog.)	Python	Java
Language (inst.)	English	Finnish
Participants	231	244
Prog. Environment	Web-based	Desktop

Table 1: Summary of contexts.

Context	Students	events/ student	pauses/ student
Python (US)	231	25186 ± 11243	2183 ± 1009
Java (Europe)	244	54698 ± 25538	6774 ± 3189

Table 2: Descriptive statistics of the study.

Similar to the University A, University B had a midterm exam in the course. For the analyses conducted in this article, we focus on students' performance in the midterm examination. The contexts are summarized in Table 1.

3.2 Event and pause categories

Keystroke data was collected in both contexts. In the analysis, every keystroke of a student was categorized to an event. We consider eight event categories: (1) Alphanumeric keystroke, (2) Delete keystroke, (3) Return keystroke, (4) Spacebar keystroke, (5) Special character keystroke, (6) Tab keystroke, (7) Successful compile/run, and (8) Failed compile/run. Since the US context uses Python, in this paper we will call a compile/run event a "Run". The reasoning behind these categories is that they represent different tasks of the student: Alphanumeric events represent typing, Delete events indicate the student is preparing to make a correction, Run events represent a completion point where the student is ready to test the code, etc. The European context does not have information on tabs or the status of run events, so analyses relating to the status of run events or tabs will only use data from the US context.

For the analysis of pauses, we chose to use four types of pauses. While pausing analyses in the context of programming have been exploratory [41], research in pausing in language production varies in terms of pause thresholds. A lower bound of 1-2 seconds appears to be the most common [5, 44, 62], and thus, we adopted a 2-second lower bound for our study. Taking into account research on working (short-term) memory time capacity [50], a meaningful upper bound is at 15 seconds – pauses between 2 and 15 seconds may reflect thinking about the code on a low, "local" level, including thinking about the syntax, and could be tied to working memory. We call these pauses *micro* pauses.

Short pauses may reflect higher-level processing like planning the following code segment, setting the next sub-goal, and revising code similar to the production stage of written language or some kind of distraction. We chose 180 seconds to be the upper bound for *short* pauses.

Mid pauses, up to 10 minutes, may reflect voluntary or problem-solving related breaks. We hypothesise that students who have difficulties may consult learning materials or visit other resources in search for help or for refreshing their memory. Such a continuous pause would cause longer task resumption [9]. Finally, *long* pauses, greater than 10 minutes, are most likely to stand for task disengagement as noted by prior work [38]. We expect such pauses to take place after finished code segments or compilation.

For simplicity, we also refer to pauses initiated by a certain event type using the name of the event type. For example, a *delete* pause is a pause initiated by the student pressing the delete key; the last event before a *failed run* pause is a *failed run* event. Pauses preceded by other event types are named similarly.

Because of data availability, we use a single measure of outcome – exam score. In the US context we use the score of a exam that falls just before the last project in the study. In the European context, we use the exam score from the first out of three programming exams. The first exam was organized on the third week of the seven-week course.

3.3 Statistical tests

We report p values of all statistical significance tests, of which there are 95. We follow the American Statistical Association's recommendations to use p values as one piece of evidence of significance, to be used in context [63], though we do suggest $p < 0.05/95 \approx 0.0005 = 5e^{-4}$ as a reasonable guideline for credible p values [24]. When considering the claims in our work, we suggest taking into account the additional supports beyond single p-values. For example, claiming that *delete* pauses are negatively correlated with exam score is based on consistent negative correlation across pause types in both studied contexts. For distribution comparisons we use the t-test (as the data appears normal) with Cohen's d effect sizes and for correlation we use the Pearson r statistic.

4 RESULTS

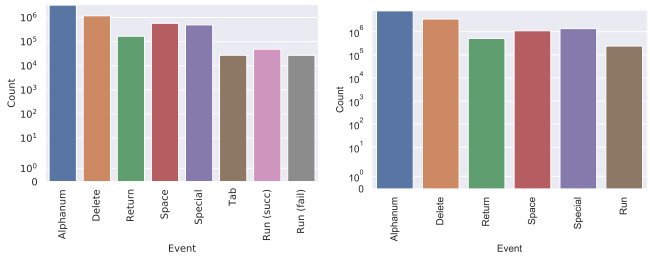
4.1 Descriptive statistics

Table 2 shows descriptive statistics of our study and Figure 1 shows the distribution of event types for each of the two contexts. *Alphanumeric* keystrokes are the most common event, with *space* (spacebar) and *special characters* also being common. *Run* events and the *tab* keystroke are less common. Both contexts use an editor that automatically indents the next line of code after a *return* key press, which likely contributes to the observed lack of tab keystrokes.

A difference between the contexts is the relative frequency of *run* events – students in the Java context run their code far more often than those in the Python context. This is likely not due to the language, but the way the courses are organized. In the Python context, students had one large assignment due each week, while Java students had tens of smaller assignments due each week. We conjecture that the smaller assignments induced the students to run/compile their code more often.

4.2 Frequency of pauses

We calculated a measure of pause frequency as $\frac{p_l}{n}$ where p_l is the number of pauses of length l and n is the total number of events. In the Python context, on average, student pause frequency



(a) US/Python **(b) European/Java**

Figure 1: Log-scale bar chart showing the total number of events for each type. (a) Number of events in the US/Python context. (b) Number of events in the European/Java context.

is 0.09 ± 0.02, meaning, on average, students execute 11 events before pausing for two seconds or more. Most pauses are *micro* pauses, which have a frequency of 0.07, followed by *short*, *mid*, and *long* pauses with frequencies of 0.02, 0.001, and 0.001, respectively.

The Java context was somewhat different: on average, student pause frequency is 0.13 ± 0.03, meaning, on average, students execute 8 events before pausing for two seconds or more. Most pauses are *micro* pauses, which have a frequency of 0.09, followed by *short*, *mid*, and *long* pauses with frequencies of 0.03, 0.002, and 0.001, respectively.

As might be expected, Figure 2 shows a negative correlation between pause frequency and exam score, meaning students who are pausing more often are performing worse on the exams. In the Python context, this correlation is consistent across *micro* ($r = -0.30, p =$ 3.16e-6), *short* ($r = -0.35, p =$ 5.57e-8), and *mid* ($r = -0.38, p =$ 3.71e-9) pause lengths, with a weaker correlation for *long* ($r = -0.18, p = 0.0061$) pauses. The Java context, in contrast, has a weaker correlation for the *micro* pause ($r = -0.11, p = 0.0654$) than for the *short* ($r = -0.20, p = 0.0013$), *mid* ($r = -0.23, p = 0.0003$), or *long* ($r = -0.22, p = 0.0006$) pauses. In general, the correlations for the Python context are stronger than for the Java context, although as seen in Figure 2, the Java context has a noticeable ceiling effect in the exam.

Figure 3 shows correlations between the number of different type of pauses that students take, i.e., whether students who are pausing for short amounts of time are also taking longer pauses. All types of pauses are at least moderately correlated with all other types of pauses (see Figure 4). Interestingly, the correlations weaken as the pause lengths grow for the Python context, while the Java context shows a strong correlation for the *mid*/*long* pause pair.

4.3 Student types

To characterize students, we represent each student using a vector that contains the relative proportions of each pause type. For example, a student represented with a vector $[0.80, 0.15, 0.03, 0.02]$ has 80% *micro* pauses, 15% *short* pauses, 3% *mid* pauses, and 2% *long* pauses. Since the vector is a partition of unity, the feature vector has only three degrees of freedom, though, for clarity, we represent it here with all four coordinates.

To identify student types based on the vector representations, we use k-means clustering to cluster students into student types. Using the elbow method (visually finding the "elbow" of a line chart of number of clusters against explained variance [56]) to identify a good number of clusters, we chose $k = 2$ for interpretability, though, as we will see in Section 5.2, the choice of k is not particularly important in this case.

We see in Table 3 and Figure 5 that one group of students in each context took relatively more *short*, *mid*, and *long* pauses than the other group, although in the Java context, the difference is less pronounced. We call the clusters the *longer pause* and *shorter pause* groups, respectively. When examining the groups and exam scores, we observe that the students in the *shorter pause* group had higher exam scores than those students who took longer pauses. The distributions of pause frequencies are approximately normal and t-tests suggest that there is a difference between *short*, *mid*, and *long* distributions.

4.4 Initiating pauses

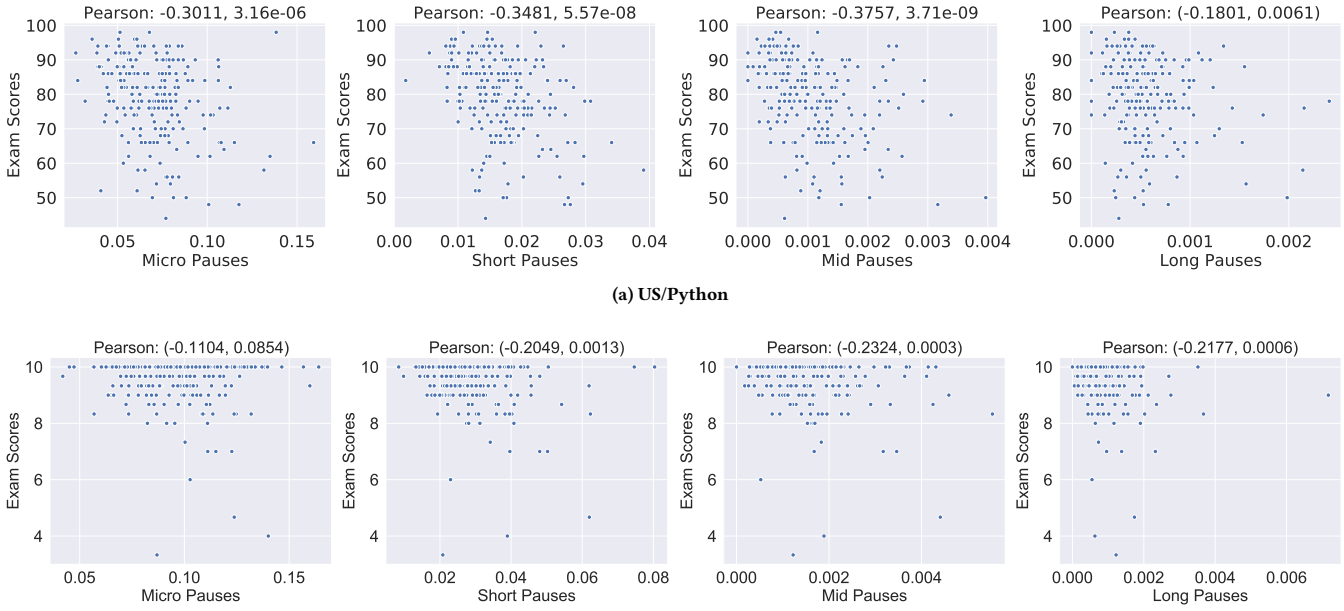
In Figure 6 we see relative frequencies of event types by pause length. We define relative frequency for an event type E as the percentage of pauses of a given length initiated by an event of type E. For example, in Figure 6 we see that in the Python context, *alphanumeric* keystrokes initiate 27% of all *micro* pauses (2-15 seconds) and 15% of *long* pauses (> 10 minutes) while accounting for 57% of all events, regardless of whether the events initiated a pause or not. In the Java context, the distribution related to *alphanumeric* events that start a pause is very similar. Roughly 29% of *micro* pauses and 18% of *long* pauses are initiated by the events while they account for 28% of all events.

Certain types of events in both contexts decrease in frequency with increasing pause length. *Alphanumeric*, *return*, *space* and *special characters* seem to follow this trend preceding to a greater extent shorter pauses.

In Table 4 we see that *alphanumeric* events initializing *micro* pauses have a positive correlation with exam score, but that the correlation weakens until it is not detectable for *long* pauses. Conversely, pausing after *special characters* is not necessarily correlated with success. In fact, a weak negative correlation exists with *special characters* initializing *micro* pauses.

In the Python context, the percentage of pauses preceded by the *delete*, *return*, and *space* keystroke events remains roughly the same across pause lengths (Figure 6). The *return* keystroke is unique among the three in that, despite being so infrequent in the data, it precedes so many pauses (11-13%). This tendency does not repeat in case of *delete* and *space* events.

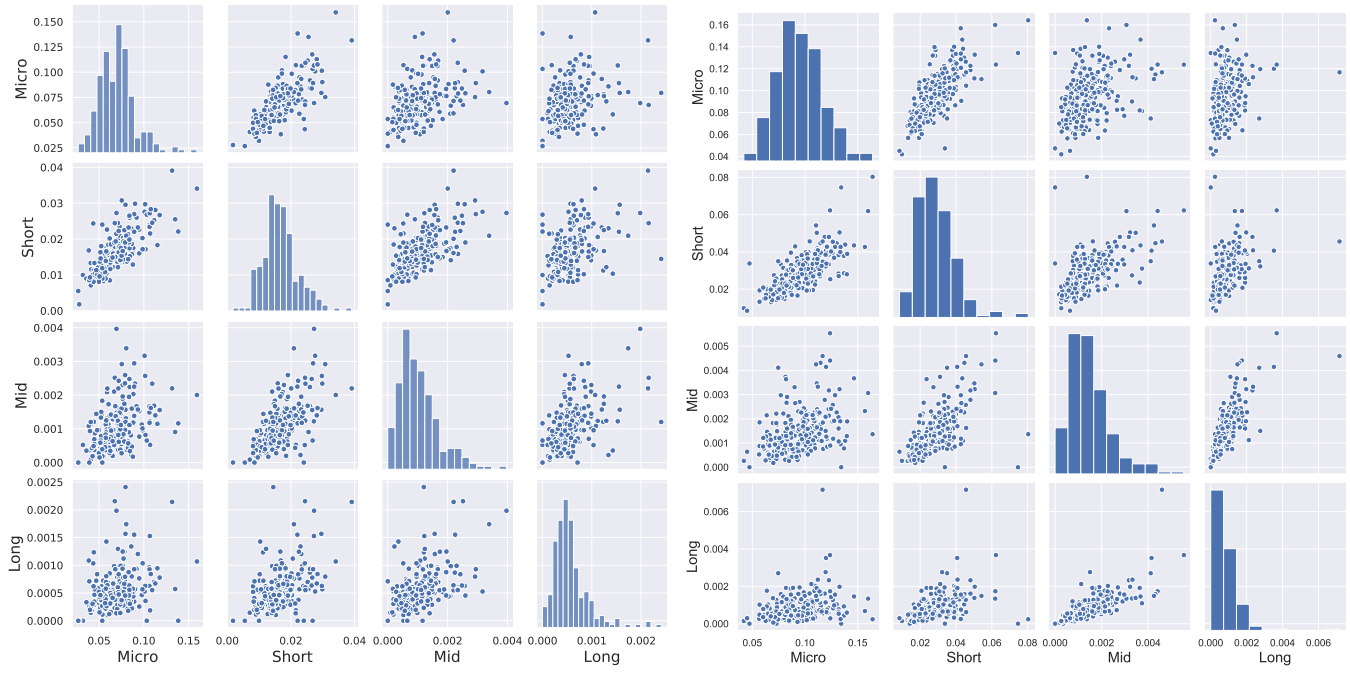
In the Java context, the situation is different. *Return* and *space* events show steady decline in percentages of preceding pauses. The longer the pause, the less common it is for those event to precede it. The opposite applies to the *delete* events. This could be accounted for differences in programming languages and their relations to students' native languages [18]. Even though the deleting behaviour differs across the contexts, correlation of most *delete* pauses with exam scores remains negative in both cases.



(a) US/Python

(b) European/Java

Figure 2: Frequency of the different type of pause correlated with exam score. Frequency is calculated as number of pauses divided by total number of events.



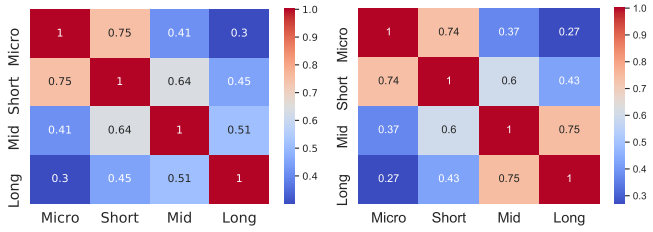
(a) US/Python

(b) European/Java

Figure 3: Correlations of total pauses with each other per student. As expected, the number of micro pauses a student takes has a strong positive correlation with the number of short pauses. While there are still strong and medium correlations between shorter and longer pauses, the correlations become weaker.

Context	Cluster	Centroid Micro	Short	Mid	Long	Students	Average keystrokes $\times 10^4$	Average pauses $\times 10^3$	Average exam
Python (US)	*shorter*	0.81	0.17	0.010	0.0060	71% (164)	2.6 ± 1.1	2.3 ± 1.0	80.2 ± 11.3
Python (US)	*longer*	0.75	0.23	0.016	0.0077	29% (67)	2.4 ± 1.3	2.1 ± 1.1	76.2 ± 11.5
Java (Europe)	*shorter*	0.79	0.20	0.01	0.01	57% (139)	5.2 ± 2.1	6.2 ± 2.8	9.59 ± 0.90
Java (Europe)	*longer*	0.72	0.26	0.01	0.01	43% (105)	5.8 ± 3.0	7.6 ± 3.5	9.35 ± 0.88

Table 3: Statistics of the clusters for the two contexts, US and European. For the Cluster column, "*shorter*" means "*shorter pause*" and similar with *longer*. A t-test for the two distributions of exam scores yields ($t = 2.2, p = 0.026, d = 0.35$) in the US context and ($t = 2.1, p = 0.034, d = 0.28$) in the European context.



(a) US/Python **(b) European/Java**

Figure 4: Similar to Fig. 3, this figure shows correlation coefficients for the different pause lengths.

Both *successful* and *failed run* attempts have disproportionate prominence among events preceding pauses relative to their overall frequency.

5 DISCUSSION

5.1 Frequency of pauses

Before answering the first research question, *Is there a correlation between the relative number of pauses a student takes and their performance (exam score)?*, we checked whether our bucketing was sensible by performing a correlation test. As we can see from Figs. 3 and 4, there are correlations between all types of pauses which is not surprising since the pauses lengths are on the time continuum. However, neighbouring types of pauses do not show a very high degree of similarity, which justifies our choice. Moreover, *micro* and *short* pauses, having the highest correlation coefficient, yield quite different correlation coefficients in terms of relationships with exam scores (see Figure 2).

In general, we observe that students who pause more often perform more poorly on exams (Figure 2), which is in line with the results observed by Leppänen et al. [41]. This effect is not large, but it is consistent across pause types and contexts. We note that this measurement is frequency of pauses, so it is normalized across students regardless of the number of total events they execute. In this paper we do not make any claims regarding what students were doing during their pauses, whether they were thinking, drawing on other resources, or disengaged. But the correlations in our data indicate that regardless of pause activity, pauses correlate negatively with exam score, at least in the aggregate. We note that certain activities may not cause negative correlation with achievement, but it appears that these activities are in the minority and are dominated by negative-effect activities.

Frequency of *mid* pauses (3-10 minutes) in both contexts have the strongest negative correlation with exam score of all the pause types. Comparing to the *long* pause which does not have an upper bound, it is clear that after at most 10-minutes long *mid* pause students get back to typing. We conjecture that *mid* pause may be the most harmful because it potentially can cause the longest resumption. If the activity taking place during the pause is not related to the task, the pause may be treated as irrelevant interruption [23]. According to Altmann and Trafton [4] and many others (for example, see [6, 22, 29]), the length of interruption correlates with the time of task resumption and numbers of possible errors.

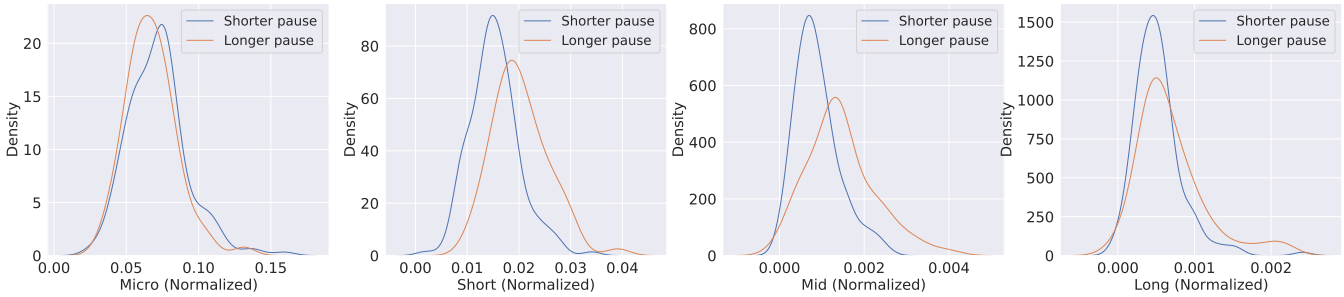
5.2 Student types

Our second research question is: *What groups of students exist when clustering on pausing behavior?* We clustered students into two types, *longer pause* students and *shorter pause* students. *Shorter pause* group tends to take proportionally more *micro* pauses, whereas *longer pause* students take fewer *micro* but more of *short*, *mid*, and *long* pauses. The *shorter pause* students appear to perform better in the exam in the both contexts.

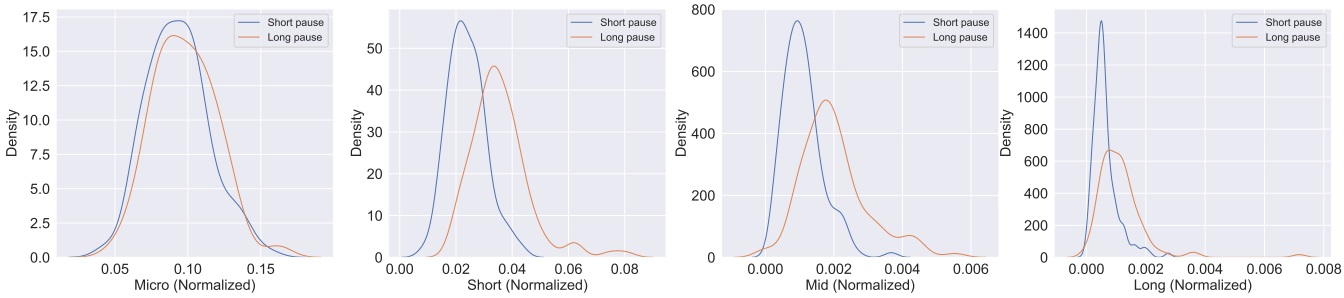
In a sense, grouping students into clusters is arbitrary: Figure 2 shows that pauses of all lengths are negatively correlated with exam score, indicating that the 4-dimensional feature vectors are not linearly independent, effectively making our clustering single-dimensional and not particularly interesting regardless of choice of k. Nevertheless, the analysis reveals one difference between the contexts that may be of interest: in Table 3, we see that more events correspond to more pauses across the contexts. However, groups which produce more events are not the same. In the US/Python context, the *shorter pause* group tends to type and pause more, whereas in the European/Java context the opposite applies. Additionally, proportions of pauses in the US/Python context remain roughly consistent across the groups and equal to 0.09 and in the European/Java context similarly, being 0.12 and 0.13. This observation could be due to a number of context-specific factors, such as the way how each context uses programming assignments.

5.3 How pauses are initiated

Our third research question is: *What events initiate a pause and how does this correlate with the performance of the student?* The first thing to note is that the distributions of event types, for each



(a) US/Python



(b) European/Java

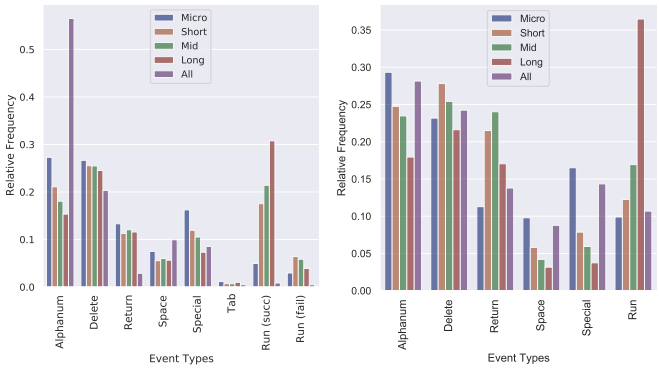
Figure 5: Clustering. In the Python context, t-test statistics (t, p) and effect sizes (d) between the two distributions are: Micro ($t = 1.5, p = 0.12, d = 0.23$), Short ($t = -6.9, p = 3.6e{-}11, d = -1.01$), Mid ($t = -6.4, p = 8.4e{-}10, d = -0.93$), and Long ($t = -3.2, p = 0.0014, d = -0.47$). In the Java context, t-test statistics (t, p) and effect sizes (d) between the two distributions are: Micro ($t = 1.7, p = 0.08, d = -0.22$), Short ($t = -10.8, p = 2.4e{-}22, d = -1.35$), Mid ($t = -9.3, p = 1.1e{-}17, d = -1.15$), and Long ($t = 2.1, p = 2.6e{-}9, d = -0.77$).

	Pause length	Enter		Alphanum		Delete		Special		Space		Tab		(Success) run		Fail run	
		r	*p*	*r*	*p*	*r*	*p*	*r*	*p*	*r*	*p*	*r*	*p*	*r*	*p*	*r*	*p*
Python	Micro	0.27	1e−5	0.28	1e−5	−0.33	1e−7	−0.23	4e−4	0.21	0.001	0.058	0.41	−0.17	0.015	−0.38	1e−7
	Short	0.11	0.1	0.19	0.004	−0.245	1e−4	0.01	0.81	0.19	3e−3	0.062	0.42	−0.05	0.45	−0.38	1e−7
	Mid	0.076	0.29	0.14	0.04	−0.13	0.05	0.13	0.07	0.14	0.08	0.22	0.23	−0.007	0.91	−0.22	8e−4
	Long	−0.022	0.80	0.005	0.95	−0.12	0.11	−0.03	0.74	0.037	0.73	0.13	0.57	0.067	0.32	−0.06	0.37
	All	0.2687	3e−5	0.29	5e−6	−0.36	1e−6	−0.20	0.002	0.22	5e−4	0.073	0.30	−0.13	0.04	−0.43	1e−6
Java	Micro	0.18	4e−3	0.31	0.0	−0.20	9e−4	−0.18	3e−3	−0.029	0.64			−0.034	0.58		
	Short	−0.012	0.84	0.21	5e−4	−0.16	0.010	0.055	0.36	0.021	0.73			−0.024	0.69		
	Mid	-0.028	0.66	0.23	2e−4	−0.23	2e−4	0.023	0.72	0.034	0.60			0.079	0.19		
	Long	−0.012	0.86	0.033	0.62	0.067	0.28	−0.24	2e−4	0.014	0.86			0.070	0.25		
	All	0.082	0.18	0.23	1e−4	−0.24	1e−4	0.21	5e−4	0.12	0.042			−0.088	0.15		

Table 4: Pearson r correlations with p values between a student's tendency to initiate a given length of pause with a given event type and exam score. "All" indicates percentage across all events (both those initiating pauses and not). We do not have data on tabs or whether a run was successful or not in the Java context, so tab values are not included for the Java context and the *(Success) run* column should be interpreted as a successful run for the Python context and all runs for the Java context.

of the four pause lengths, do not match the overall distribution of events (Figure 6). This confirms, as one might expect, that, in general, students are not pausing at arbitrary times, meaning that pauses are generally purposeful and not taken at random times while typing.

5.3.1 Deletes and failed runs. There is some abruptness regarding what initiates a *long* pause. *Long* pauses, those of 10 minutes or more, may indicate that the student is disengaged from working on the project [38]. One would expect the most natural way to take a break would be a *successful run*. Yet, in the Python context, only 30% of *long* pauses are initiated as such. Another intuitive, natural break would be a *failed run*, as the student might need a break or an extended session of reviewing external materials after a failure. Yet *failed runs* account for only 4% of *long* pauses. This means that 66% of long pauses are initiated with a keystroke. The most common event for *long* pauses, the *delete* keystroke, initiates 25%



(a) US/Python **(b) European/Java**

Figure 6: Grouped bar chart showing normalized/relative frequencies of keystrokes by pause length. "All" are all events, whether they precede a pause or not.

of the pauses. The Java context is similar, with 22% of *long* pauses initiated by *delete*. This seems remarkable. A *delete* press often indicates an error and so, after the *delete* press, the student needs to execute keystrokes to replace the incorrect code. At times, however, students are taking a break instead of completing the correction. If this happens it could indicate that the student may lack motivation, diligence, or the corrective know-how without consulting external help. Other types of pauses were also rather often preceded by a *delete* event (26-27%). From Table 4, we can see that the correlations of the exam score with such pauses are negative. This may signify that *deletes* are used less often for removing unneeded code (e.g., print statements or comments) and more often when students are confused and do not know how to proceed. This same reasoning could be used to explain the negative correlation of *failed runs* initiating pauses, i.e., that an extended pause after a *failed run* indicates the student does not know how to fix the problem and has to take time to either consult other materials or take a break. Indeed, the correlations of *failed runs* with exam score closely mirror those of *delete* key presses.

One could suggest that the consistent negative correlations of *delete* and *failed run* events initiating pauses with exam score simply reflect the overall correlation of these event types with exam score. We note, however, that distributions of the two events are different and they demonstrate different degree of involvement in a *long* pause initiation. While *deletes*, constituting 20%/24% of all events and preceding 25%/22% of *long* pauses, *failed runs* account for only 0.5% of all events but precede 4% of long pauses. Thereby, it is six times more likely that *failed run* event will initiate a long pause than a delete keystroke. A plausible interpretation of this observation is that students are deliberately pausing after failed runs, at least more often than after deletes.

5.3.2 Events decreasing in frequency with pause length. *Alphanumeric* keystrokes are what we might call "middle" events – they are the most common while being somewhat less significant in terms of reflecting thinking processes. The fact that the frequency of *alphanumeric* events preceding a pause decreases with increasing pause length as much as it does (Figure 6) suggests that students are completing their lower-level processing thoughts before taking longer breaks. Indeed, it appears that students are deliberate in taking longer breaks rather than getting interrupted, as would be the case if *alphanumeric* pauses were more common.

In Python, statements generally do not end with a *special character* as they do in Java (e.g. semicolon for a single-line statement and closing brace for a block). So it is not surprising that pauses initiated by *special characters* decrease in frequency with increasing pause length in the Python context. What is surprising, however, is that the Java context has a very similar phenomenon. We expected longer pauses to be frequently initiated by *special characters* in the Java context, as ending a line with a semicolon seems like a natural stopping point. We do not know why this is not the case, but we suspect that this is, again, a consequence of the difference in instructional methods between the two contexts. The Java students work on smaller projects and run more often, and so they may be more likely to complete their thought or work session with a *run* event.

In Table 4 we see that, in both the Python and Java contexts, *alphanumeric micro* pauses are positively correlated with exam score while *special character micro* pauses are negatively correlated. As these two event types behave similarly in other respects, we discuss a possible explanation for this difference. Roughly half of *special characters* require further processing: an open parenthesis expects formal parameters for a function call; quotes expect a string; an open bracket expects list/array indices; etc. It may be that a *micro* pause, which may last as many as 15 seconds, indicates student hesitancy and lack of fluency with Python or Java syntax. This lack of fluency with a fundamental aspect of programming may be why the student exam scores are lower. Problems with special characters being indicative of struggling has been hypothesized also in previous work [19, 40]. If *special character* pauses do indicate an uncertainty with syntax then instructors may consider an increased focus on syntax fluency for students initiating pauses with *special characters*.

5.3.3 Events constant across pause length. In the Python context, the percentage of pauses initiated by the *return* event remains roughly the same across pause lengths (Figure 6). This makes sense in the context of both shorter and longer pauses: pressing *return* requires short-term planning for the next line, so its prevalence among *micro* and *short* pauses is logical; pressing *return* is also a natural stopping point before taking a break, so it is frequent among mid and long pauses. Being every 33rd event in the Python data and every 7th in Java data, return initiates approximately 12% of any type pauses in the Python context and as much as 12-24% of of any type pauses in the Java context. This seems to confirm the return keypress being a natural stopping point.

5.3.4 Run events. Being rather rare in the typing data in both contexts, *run* events are notably evident among events preceding pauses, especially *short*, *mid* and *long*. This is not unexpected: it would be highly unusual for a *run* event to take fewer than two seconds, so the great majority of *run* events would precede at least a *micro* pause. A large proportion of *long* pauses are initiated by *successful runs* in the Python context and *runs* in the Java context. It

is instructive to consider why students would pause after a *successful run*. It is possible that a student takes a pause to consult external resources (e.g., internet, textbook, another person) regarding how to proceed with their program, but it seems more likely that the student would have at least an idea of what to do next after a *successful run*. Therefore, we suggest that the more likely scenario is that the student is instead disengaging from working on their assignment. If this is the case, then we could possibly use the percentage of *successful run* long pauses in the Python context as a lower bound for the number of *long* pauses in which students are disengaging. In our data, this indicates that students are disengaging during at least 30% (roughly) of *long* pauses. We expect that this is a conservative lower bound.

5.4 Threats to validity

Internal validity. As is natural in educational studies, our study comes with an inherent self-selection bias. It is possible that the way the studied courses were organized and the way the student population at both universities is formed influences the observed outcomes. It is unclear, for example, whether similar results would be observed if the study would have been conducted in the context of primary or secondary education, or in life-long learning. When considering the outcome of the courses, we used exam score as a proxy for performance, which can be affected by factors such as exam stress. In addition, the European/Java context had a noticeable ceiling effect in the exam outcomes. It is possible that this also influenced some of our findings and that lifting the ceiling effect would affect the correlations.

External validity. We studied keystrokes in two contexts to increase the degree to which our findings can be generalized to other contexts (see Section 3.1). The strength of the correlations and the p values varied somewhat between the contexts and we cannot state which context-specific factors contributed to the differences.

6 CONCLUSIONS

In this article, we presented an analysis of keystrokes with an eye toward understanding pausing behavior of CS1 students and its implications on academic outcomes. In this section we draw conclusions from our results in each of our three research questions.

RQ1 *Is there a correlation between the relative number of pauses a student takes and their performance (exam score)?* We observe that negative correlations between pause frequency and exam score exist as illustrated in Fig. 2. The most illustrative is the frequency of *mid* pauses – those of length 3-10 minutes. We suggest that these pauses indicate that a student may be distracted easily, but it could also indicate students who are spending time using external resources for help on their projects. Révész et al. [48] suggests, since keystroke logs alone do not allow us to "make inferences about the specific cognitive processes that underlie pausing behaviors", that combining event logs with "other techniques such as verbal reports and eyetracking" could be helpful in obtaining more detailed information. Further study could help us understand what these students are doing during pauses and what they were working on when they paused. But in the meantime, the pause/exam score correlation appears actionable. We suggest that a tool that allows practitioners to visualize students' pausing behavior could be particularly useful. In addition, as previous studies that have used keystroke data for predicting course outcomes have mainly focused on latencies smaller than 750ms [19, 40], future research should seek to combine such keystroke data with pausing data and study whether these phenomena have the same underlying tacit factors.

RQ2 *What groups of students exist when clustering on pausing behavior?* We found in a cluster analysis that students whose pausing behavior tended toward short pauses performed better in general on exams. The cluster analysis primarily indicated a correlation between typical pause length for a student and exam score. When considering the identified student types in the light of CER studies that have identified student types such as the tinkerers, stoppers, and movers [30, 45], most of the students in the studied contexts could be categorized as movers, despite the differences in their pausing behavior. As pausing is linked with cognition and thought processes, and as writing code is linked with a multitude of factors including understanding syntax and the given problem [52, 66], further research is needed to understand the lack of stopping and the differences in pausing.

RQ3 *What events initiate a pause and how does this correlate with the performance of the student?* We have presented evidence that pauses do not occur randomly while a student is programming – students tend to finish their thoughts and pause after a natural stopping point. This observation is in line with the studies on student cognition and programming and how students solve programming problems [15, 51], where students write constructs informed by schemas that engage procedural memory. Fully 25% (Python) and 22% (Java) of *long* pauses (>10 minutes) are initiated by *delete* events. We suggest that students who pause after *delete* are possibly less engaged (taking a break instead of writing the code to replace the deleted characters) or they lack the knowledge to write a fix (consulting external resources to learn how to fix the problem). This presents interesting questions for future research, such as what percentage of *delete* pauses indicate a disengaged student. Beyond identification of at-risk students, the negative correlations of *special character* and *failed compile* pauses suggest possible pedagogical and material innovations to improve student fluency after special characters and minimize the number of failed runs.

In addition to the directions for future research discussed above, there are additional avenues for further research. As an example, while previous research in syntax errors has noted that there are differences in the time that it takes to fix syntax errors [2, 16], our study highlights that pause durations are related to the pressed keys. Combining information on present syntax errors (or the lack of them) with information on pauses could create more in-depth understanding of students knowledge and actions – for example, pauses preceded by a syntax error likely indicates different thought processes than pauses not preceded by a syntax error. Similarly, looking at what syntactic construct was just typed or is being typed could affect pausing behavior. While our definitions of pauses were based on related literature (e.g., [41]), future work could explore alternative bins, including higher resolution bins for the micro pause, which spans lengths from 2 to 15 seconds in the work reported in this paper. Language specific differences should also be studied further – as an example, we noted that in the European/Java context students took a *micro* pause on average after 8 keystrokes,

while students in the US/Python context took a *micro* pause on average after 11 keystrokes. It would be meaningful to understand where this difference stems from. If it is simply the language, then one possible implication is that the relative verbosity of Java when compared to Python would not only require the students to type more, but also to pause to think more. On the other hand, if it is a product of a contextual factor, then it could be something that could be sought to disseminate to other contexts as well. Future studies could also focus on differences between the beginning and end of the course to see if programming behavior changes with experience.

REFERENCES

[1] Alireza Ahadi, Raymond Lister, Heikki Haapala, and Arto Vihavainen. 2015. Exploring machine learning methods to automatically identify students in need of assistance. In *Proceedings of the eleventh annual international conference on international computing education research*. 121–130.

[2] Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (Kansas City, Missouri, USA) *(SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 522–527. https://doi.org/10.1145/2676723.2677258

[3] Amjad Altadmri, Michael Kolling, and Neil CC Brown. 2016. The cost of syntax and how to avoid it: Text versus frame-based editing. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 748–753.

[4] Erik M Altmann and J Gregory Trafton. 2007. Timecourse of recovery from task interruption: Data and a model. *Psychonomic Bulletin & Review* 14, 6 (2007), 1079–1084.

[5] Rui A Alves, São Luís Castro, Liliana de Sousa, and Sven Strömqvist. 2007. Chapter 4: Influence of Typing Skill on Pause–Execution Cycles in Written Composition. In *Writing and Cognition*. BRILL, 55–65.

[6] Mark B. Edwards and Scott D Gronlund. 1998. Task interruption and its effects on memory. *Memory* 6, 6 (1998), 665–687.

[7] Jens Bennedsen and Michael E Caspersen. 2006. Abstraction ability as an indicator of success for learning object-oriented programming? *ACM Sigcse Bulletin* 38, 2 (2006), 39–43.

[8] Susan Bergin and Ronan Reilly. 2005. Programming: factors that influence success. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education*. 411–415.

[9] Jelmer P Borst, Niels A Taatgen, and Hedderik van Rijn. 2015. What makes interruptions disruptive?: A process-model account of the effects of the problem state bottleneck on task interruption and resumption. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. ACM, 2971–2980.

[10] Neil Christopher Charles Brown, Michael Kölling, Davin McCall, and Ian Utting. 2014. Blackbox: a large scale repository of novice programmers' activity. In *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 223–228.

[11] Brian L. Butterworth. 1980. *Evidence from pauses in speech*. New York: Academic Press.

[12] Adam S Carter, Christopher D Hundhausen, and Olusola Adesope. 2015. The normalized programming state model: Predicting student performance in computing courses based on programming behavior. In *Proceedings of the eleventh annual international conference on international computing education research*. 141–150.

[13] Jasone Cenoz. 2000. Pauses and hesitation phenomena in second language production. *ITL - International Journal of Applied Linguistics* 127-128 (Jan. 2000), 53–69.

[14] Markus F Damian and Hans Stadthagen-Gonzalez. 2009. Advance planning of form properties in the written production of single and multiple words. *Language and Cognitive Processes* 24, 4 (2009), 555–579.

[15] Simon P Davies. 1991. The role of notation and knowledge representation in the determination of programming strategy: a framework for integrating models of programming behavior. *Cognitive Science* 15, 4 (1991), 547–572.

[16] Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. 2012. All Syntax Errors Are Not Equal. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (Haifa, Israel) *(ITiCSE '12)*. ACM, New York, NY, USA, 75–80. https://doi.org/10.1145/2325296.2325318

[17] 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)*.

[18] John Edwards, Juho Leinonen, Chetan Birthare, Albina Zavgorodniaia, and Arto Hellas. 2020. Programming Versus Natural Language: On the Effect of Context on Typing in CS1. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*. 204–215.

[19] John Edwards, Juho Leinonen, and Arto Hellas. 2020. A Study of Keystroke Data in Two Contexts: Written Language and Programming Language Influence Predictability of Learning Outcomes. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 413–419.

[20] 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*. 715–724.

[21] Jean-Noël Foulin. 1995. Pauses et débits : les indicateurs temporels de la production écrite. *L'année psychologique* 95, 3 (1995), 483–504.

[22] Tony Gillie and Donald Broadbent. 1989. What makes interruptions disruptive? A study of length, similarity, and complexity. *Psychological research* 50, 4 (1989), 243–250.

[23] Alexander JJ Gould. 2014. *What makes an interruption disruptive? Understanding the effects of interruption relevance and timing on performance*. Ph.D. Dissertation. UCL (University College London).

[24] Winston Haynes. 2013. *Bonferroni Correction*. Springer New York, New York, NY, 154–154. https://doi.org/10.1007/978-1-4419-9863-7_1213

[25] Arto Hellas, Petri Ihantola, Andrew Petersen, Vangel V Ajanovski, Mirela Gutica, Timo Hynninen, Antti Knutas, Juho Leinonen, Chris Messom, and Soohyun Nam Liao. 2018. Predicting academic performance: a systematic literature review. In *Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education*. 175–199.

[26] Arto Hellas, Juho Leinonen, and Petri Ihantola. 2017. Plagiarism in take-home exams: Help-seeking, collaboration, and systematic cheating. In *Proceedings of the 2017 ACM conference on innovation and technology in computer science education*. 238–243.

[27] C. D. Hundhausen, D. M. Olivares, and A. S. Carter. 2017. IDE-Based Learning Analytics for Computing Education: A Process Model, Critical Review, and Research Agenda. *ACM Trans. Comput. Educ.* 17, 3, Article 11 (Aug. 2017), 26 pages. https://doi.org/10.1145/3105759

[28] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H. Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, Miguel Ángel Rubio, Judy Sheard, Bronius Skupas, Jaime Spacco, Claudia Szabo, and Daniel Toll. 2015. Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. In *Proc. of the 2015 ITiCSE on Working Group Reports* (Vilnius, Lithuania) *(ITICSE-WGR '15)*. ACM, 41–63.

[29] Shamsi T Iqbal and Brian P Bailey. 2006. Leveraging characteristics of task structure to predict the cost of interruption. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 741–750.

[30] Matthew C Jadud. 2005. A first look at novice compilation behaviour using BlueJ. *Computer Science Education* 15, 1 (2005), 25–40.

[31] Matthew C Jadud. 2006. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research*. ACM, 73–84.

[32] 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.

[33] Minna Kumpulainen. 2015. On the operationalisation of 'pauses' in translation process research. *Translation & Interpreting* 7, 1 (2015), 47–58.

[34] Isabel Lacruz and Gregory M. Shreve. 2014. Pauses and Cognitive Effort in Post-Editing. In *Post-editing of Machine Translation: Processes and Applications*. Cambridge Scholars Publishing.

[35] Joy Yeonjoo Lee, Jeroen Donkers, Halszka Jarodzka, Géraldine Sellenraad, and Jeroen J.G. van Merriënboer. 2020. Different effects of pausing on cognitive load in a medical simulation game. *Computers in Human Behavior* 110 (Sept. 2020), 106385.

[36] Marianne Leinikka, Arto Vihavainen, Jani Lukander, and Satu Pakarinen. 2014. Cognitive flexibility and programming performance. In *Psychology of programming interest group workshop*. 1–11.

[37] Juho Leinonen. 2019. *Keystroke Data in Programming Courses*. Ph.D. Dissertation. University of Helsinki.

[38] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. 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.

[39] 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*. ACM, 200–208.

[40] 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.

[41] Leo Leppänen, Juho Leinonen, and Arto Hellas. 2016. Pauses and spacing in learning to program. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. ACM, 41–50.

[42] Soohyun Nam Liao, Daniel Zingaro, Kevin Thai, Christine Alvarado, William G. Griswold, and Leo Porter. 2019. A Robust Machine Learning Technique to Predict

Low-Performing Students. *ACM Trans. Comput. Educ.* 19, 3, Article 18 (Jan. 2019), 19 pages. https://doi.org/10.1145/3277569

[43] Sharon O'Brien. 2006. Pauses as Indicators of Cognitive Effort in Post-editing Machine Translation Output. *Across Languages and Cultures* 7, 1 (June 2006), 1–21.

[44] Thierry Olive, Rui Alexandre Alves, and São Luís Castro. 2009. Cognitive processes in writing during pause and execution periods. *European Journal of Cognitive Psychology* 21, 5 (Aug. 2009), 758–785.

[45] David N Perkins, Chris Hancock, Renee Hobbs, Fay Martin, and Rebecca Simmons. 1986. Conditions of learning in novice programmers. *Journal of Educational Computing Research* 2, 1 (1986), 37–55.

[46] Andrew Petersen, Jaime Spacco, and Arto Vihavainen. 2015. An exploration of error quotient in multiple contexts. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*. 77–86.

[47] Leo Porter, Daniel Zingaro, and Raymond Lister. 2014. Predicting Student Success Using Fine Grain Clicker Data. In *Proceedings of the Tenth Annual Conference on International Computing Education Research* (Glasgow, Scotland, United Kingdom) *(ICER '14)*. Association for Computing Machinery, New York, NY, USA, 51–58. https://doi.org/10.1145/2632320.2632354

[48] Andrea Révész, Marije Michel, and MinJin Lee. 2017. *Investigating IELTS Academic Writing Task 2: Relationships between cognitive writing processes, text quality, and working memory*. British Council, Cambridge English Language Assessment and IDP.

[49] Andrea Révész, Marije Michel, and Minjin Lee. 2019. EXPLORING SECOND LANGUAGE WRITERS' PAUSING AND REVISION BEHAVIORS. *Studies in Second Language Acquisition* 41, 3 (July 2019), 605–631.

[50] Russell Revlin. 2013. *Cognition : theory and practice*. Worth Publishers, New York, NY.

[51] Robert S Rist. 1989. Schema creation in programming. *Cognitive Science* 13, 3 (1989), 389–414.

[52] Robert S Rist. 1995. Program structure and design. *Cognitive science* 19, 4 (1995), 507–562.

[53] Nathan Rountree, Janet Rountree, Anthony Robins, and Robert Hannah. 2004. Interacting factors that predict success and failure in a CS1 course. *ACM SIGCSE Bulletin* 36, 4 (2004), 101–104.

[54] Joost Schilperoord. 1996. *It's about time: Temporal aspects of cognitive processes in text production*. Vol. 6. Rodopi.

[55] Richard C Thomas, Amela Karahasanovic, and Gregor E Kennedy. 2005. An investigation into keystroke latency metrics as an indicator of programming performance. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42*. 127–134.

[56] Robert L Thorndike. 1953. Who belongs in the family? *Psychometrika* 18, 4 (1953), 267–276.

[57] Markku Tukiainen and Eero Mönkkönen. 2002. Programming Aptitude Testing as a Prediction of Learning to Program.. In *PPIG*. 4.

[58] Jeroen JG Van Merrienboer and Fred GWC Paas. 1990. Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behavior* 6, 3 (1990), 273–289.

[59] Arto Vihavainen, Juha Helminen, and Petri Ihantola. 2014. How Novices Tackle Their First Lines of Code in an IDE: Analysis of Programming Session Traces. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '14)*. ACM, New York, NY, USA, 109–116. https://doi.org/10.1145/2674683.2674692

[60] Arto Vihavainen, Matti Luukkainen, and Petri Ihantola. 2014. Analysis of source code snapshot granularity levels. In *Proceedings of the 15th annual conference on information technology education*. 21–26.

[61] Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Martin Pärtel. 2013. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 117–122.

[62] Luuk Van Waes and Peter Jan Schellens. 2003. Writing profiles: the effect of the writing mode on pausing and revision patterns of experienced writers. *Journal of Pragmatics* 35, 6 (June 2003), 829–853.

[63] Ronald L Wasserstein and Nicole A Lazar. 2016. The ASA statement on p-values: context, process, and purpose.

[64] Christopher Watson, Frederick WB Li, and Jamie L Godwin. 2013. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *2013 IEEE 13th international conference on advanced learning technologies*. IEEE, 319–323.

[65] Laurie Honour Werth. 1986. Predicting student performance in a beginning computer science class. *ACM SIGCSE Bulletin* 18, 1 (1986), 138–143.

[66] Leon E Winslow. 1996. Programming pedagogy—a psychological overview. *ACM Sigcse Bulletin* 28, 3 (1996), 17–22.