Shrestha, Raj; Leinonen, Juho; Zavgorodniaia, Albina; Hellas, Arto; Edwards, John

Pausing While Programming: Insights From Keystroke Analysis

Published in:

DOI:
10.1109/ICSE-SEET55299.2022.9794163

Published: 13/06/2022

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
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:

1 INTRODUCTION
Pausing during work is a natural behavior 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].
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.

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.
While pausing analyses in the context of programming have been (short-term) memory time capacity \[50\], a meaningful upper bound to be the most common \[5,44,62\], and thus, we adopt a 2-second intermsof pause thresholds. A lower bound of 1-2 seconds appears to be the most common due to 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 absence of pause analysis, in which 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 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.

#### 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

---

### Table 1: Summary of contexts.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>University A</th>
<th>University B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction</td>
<td>Lectures w/sections</td>
<td>Lectures, sessions</td>
</tr>
<tr>
<td>Language (prog.)</td>
<td>Python</td>
<td>Java</td>
</tr>
<tr>
<td>Language (inst.)</td>
<td>English</td>
<td>Finnish</td>
</tr>
<tr>
<td>Participants</td>
<td>231</td>
<td>244</td>
</tr>
<tr>
<td>Prog. Environment</td>
<td>Web-based</td>
<td>Desktop</td>
</tr>
</tbody>
</table>

### Table 2: Descriptive statistics of the study.

<table>
<thead>
<tr>
<th>Context</th>
<th>Students</th>
<th>events/ student</th>
<th>pauses/ student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python (US)</td>
<td>231</td>
<td>25186 ± 11243</td>
<td>2183 ± 1009</td>
</tr>
<tr>
<td>Java (Europe)</td>
<td>244</td>
<td>54698 ± 25538</td>
<td>6774 ± 3189</td>
</tr>
</tbody>
</table>

---

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

\[ \frac{p_l}{n} \] 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
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 behavior differs across the contexts, correlation of most delete pauses with exam scores remains negative in both cases.
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.

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.
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.

![Figure 4: Similar to Fig. 3, this figure shows correlation coefficients for the different pause lengths.](image)

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 pause 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
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%
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. Therefore, 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 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


