Investigating Pauses in Writing Code

Harshitha Akkaraju, William Kwok, University of Washington
Email: akkarh@uw.edu, wkwok16@uw.edu

Abstract: Understanding an individual's thought process as they encounter programming problems would enable the development of better techniques for teaching computer science and more comprehensive technical assessments. With this study, we intend to gain a better understanding of a user's mental model as they encounter computer science problems. In specific, we wanted to investigate pauses in coding and what students think before, during, and after these pauses. To run this study, we use a custom programming tutor that allows us to log all the user's timestamped input for writing code. We ran a pilot study that required participants (3) to answer eight programming questions, two of which involved writing code. We found some similarities in pause clustering for participants who answered the same questions. We also noticed that participants whose pauses were due to utilizing development environment affordances, identifying misconceptions, and identifying gaps in knowledge semantics exhibited more self-regulation as programmers.

Introduction

Computing education researchers work towards understanding learners' mental models for solving programming problems. The advent of web-based programming environments that facilitate user data collection provides researchers with new affordances such as problem-solving time or gradual progression towards a solution. Pauses in coding is an important subject to study because they indicate when one is thinking critically and what strategies led to a solution. The purpose of this study is to investigate what students may be thinking before, during, and after pauses when they approach new programming questions. To answer this question, we conducted cognitive interviews using the think-aloud strategy while participants worked on new programming problems on a custom platform that logged keystrokes. Using this data, we understood a user's gradual progression towards their submission, pause frequencies, and pause locations. We analyzed this log data in conjunction with think-aloud interviews to infer the reason behind the pauses among participants. This work has important implications on how we teach programming to novices or how we conduct technical interviews in industry.

Literature review

There were many approaches to studying and evaluating student learning. *Think-aloud* is a research method in which participants speak aloud any words in their mind as they complete a

task. The theoretical roots for think-aloud come from cognitive psychology. Leo Vygotsky (1962) formalized the the think-aloud method claiming that “inner speech of verbalized adult thought processes is a form of thinking aloud with the goal of solving [complex] problems”. In her analysis Elizabeth Charter (2003) found that analyzing think-aloud transcripts is comparatively harder than analyzing speech or writing. Complex tasks are more likely to result in large gaps in think-aloud data; thus, interference with verbalization is an important consideration. Depending on participants’ familiarity with the method, the session might involve varying levels of prompting from the researcher. Lastly, think-aloud transcripts cannot be the only source of data for a research study. Due to the limitations and challenges associated with think-aloud, there is an inherent need for some form of triangulation to account for missing data (Charter, 2003).

For the purposes of this study, we were interested in metrics that determine participants’ understanding and progression on computer science problems. This measurement gives us insights into the reason a participant may be pausing. Our literature review revealed several methods for observing participant comprehension of complex programming tasks. The study conducted by Ihantola et al. (2015) focused on quantitative data collection for learning more about a participant’s behavior in the context of programming education by logging keystrokes, line-level edits, file saves, compilations, executions, and submissions.

Leijten and Waes (2013) explored using keystroke logging in writing research alongside the think-aloud study format using their tool, InputLog. They tested various different writing events, one of which was charting number of characters over time that we replicated in the context of computer programming (Leijten & Waes, 2013). From this, we can gain insight into their immediate thought process.

Bixler and D'Mello (2013) utilized keystroke logging in another writing study which explored boredom and engagement while writing. With just a count of keystrokes between set intervals, they predicted boredom and engagement with about 56% accuracy. They recorded the relative timing, how many characters were typed, timing between keystrokes, and number of pauses (Bixler & D'Mello, 2013). While it is far from accurate, we want to make use of their models and modify their procedures by incorporating a more precise interval of keystrokes as well as examining the actual content of their code alongside a think-aloud study.

Using keystroke data in computing science education studies has not been adequately investigated. We looked at existing methods of student performance prediction in order to see what novel methods we could provide insight on. We were unable to find any previous studies which tried to compare collected quantitative data to a study done in a think-aloud format. We used a custom-built tool in order to perform time-stamped keystroke and mouse click logging to compare with a think-aloud recording to gain insight on patterns that reflect the user's potential thought processes.

Method description

Study design

While the project's broader goal was to understand participants' mental model when they approach new programming models, for this paper, we chose to investigate what led to pauses in coding that are longer than 5 seconds. With our study design, we explained the nature of these pauses and what happened before, during and after these pauses.

From our literary analysis we learned that cognitive interviews conducted using the think-aloud strategy helped with understanding the participants' thought processes, problem-solving strategies, misconceptions, and response processes (Keehner, Gorin, Feng, & Katz, 2017). We recognized that cognitive interviews are unstructured, making it more difficult to draw direct

comparisons between participants. We reasoned that they offered the most flexibility for participants to demonstrate their thought process in a novel problem space.

We received IRB approval to run this study. Three participants in the same computer science department were recruited by contacting the University of Washington graduate student mailing lists and offering monetary compensation for participation.

The cognitive interviews were hour long sessions where participants completed eight questions. Out of these questions, the first two were copying tasks, where the participant was required to retype a passage displayed on the screen. The next questions contained two short answer questions, two multiple choice questions, and two code writing questions. The studies were conducted using a custom-built web application. Our analysis only focused on the two code writing questions. The questions were obtained from two introductory computer science (*CS1*) concept inventories developed by Parker, Guzdial, and Engleman (2016) and Caceffo, Wolfman, Booth, and Azevedo (2016) and were adapted for this study. Exact form of questions used in study cannot be made public to preserve the integrity of the questions (which are being used for future studies).

During the cognitive interview sessions, participants were first informed about the purpose of this study which was to understand what people are thinking when they are solving programming questions. The interviewer then explained what the think-aloud strategy means and asked the participants to think-aloud as they are solving these questions. The participants were asked to complete an initial task sheet while thinking aloud which gave them an opportunity to practice before getting started on the programming questions. For each question in the session, participant spent a few minutes answering the question and thinking aloud. Once satisfied with

their answer, participants were asked to submit their answer. Participants were also given the option to take breaks between questions.

Custom web-based programming tool

We built a custom web-based programming tool to be used during the cognitive interviews. We considered conducting the studies without a web-based tool, using an exam-like question packet. However, since a good number of the questions involve reading through code or writing code, we determined that our participants would feel more comfortable using an environment with features such as code syntax highlighting or line numbers.

Data collection

During the think-aloud portion, we collected a video recording of the user's screen as well as observations of the user as he or she progressed. Participants were informed that the sessions were being recorded. Through the video recording, we were able to capture verbal thought processes as well as hand movements near the screen. Using our custom programming tool, we were able to collect immense amounts of time stamped events. We collected keystrokes, mouse clicks, highlights, pastes, and text content associated with participants' actions.

Analysis plan

To analyze our data, we plotted the text content length over time. We also drew time indicators for when there is a gap greater than 5 seconds. We chose 5 seconds as the minimum pause length because it turned out to be the smallest pause length for which we could overlay the think-aloud snippets and still gain insight into pause clusters. These lines conveyed the density of pauses in the time spent on a question. Along with this plot, we created a histogram of the lengths of pauses to visualize the distribution.

Using these pause times we generated from our time-stamped key logs, we determined which portions of the think-aloud data to examine. We noted observations before, during and immediately after the pause. One author went through video snippets around the pauses and

generated answers to the question of what the participant is doing before, during, and after the pause. The uniqueness of the categories was then discussed with the other author, finally condensing the number of categories to 11. We then used the final categories to code all of the pauses. Our inferred causes were: revisiting problem specification (4 occurrences), articulating confusion about the spec and scoping the problem (2), asking if conditional statement would ever be true (2), recalling language semantics (3), utilizing development environment affordances (1), identifying misconception (1), identifying gap in knowledge of language semantics (1), drawing analogies (4), articulating approach/pseudocode (3), verifying whether code makes sense (4), and proposing/considering an alternate approach (2). These inferred causes are not exhaustive by any means, but they provide a good starting point to combine the keystroke log and think-aloud data.

Results

Figure 1 shows a visualization of the pilot study log data. Participant 303 and 305 had the same question set, while 304 had a separate set of questions.

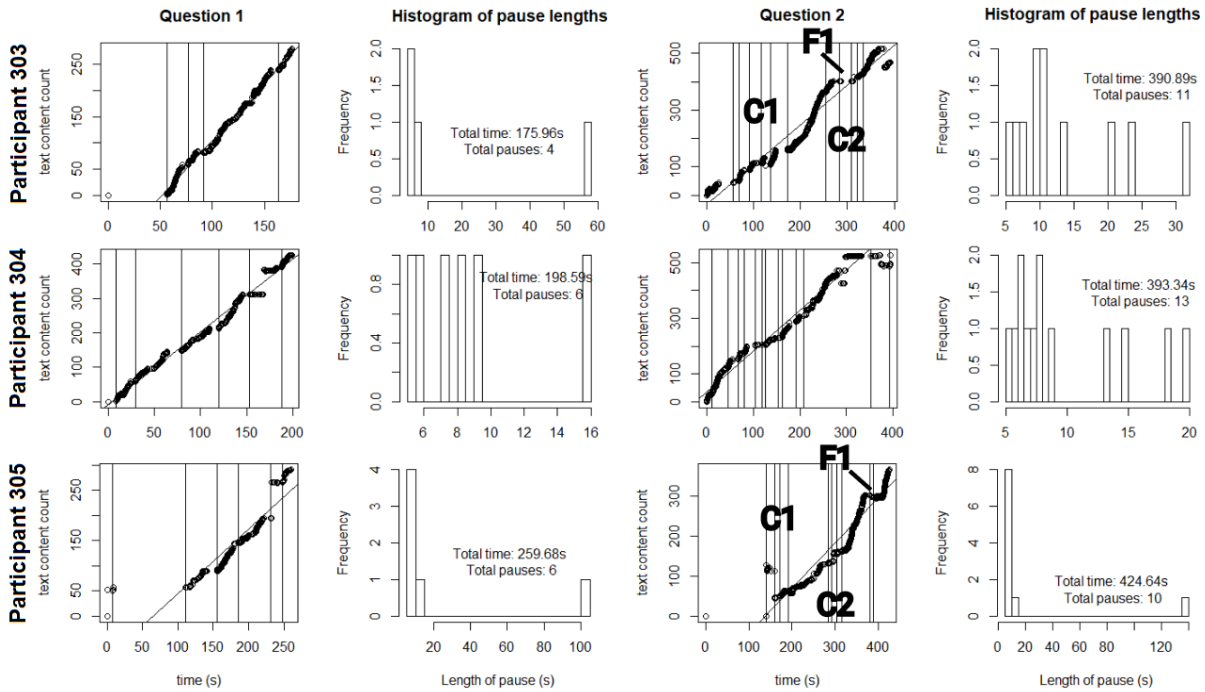

Figure 1: Pilot study log data pause clustering and lengths

We saw a notable pause at the beginning for most of the participants; this is because the participant clicked on the code editor while they were still reading through the question and asking clarifying questions. Comparing the pause clusters in question 2 for participants 303 and 305 denoted in Figure 1 by C1 and C2, we observed matching pause clusters near the beginning and end, but not in the middle. We also see a matching flat portion in 303 and 305 denoted by F1 where there is no new text content, which indicates deleting and retyping, highlighting, or usage of arrow keys to move the cursor in both cases.

We also compared the think-aloud data around these pause clusters for participants 303 and 305 and noticed some similarities in pause categories. Participant 303 recognized a gap in knowledge of language semantics and then scoped down the problem. Whereas participant 305 did not explicitly state the lack of awareness of language semantics. On the other hand, participant 304 worked on a variation of the same question but had explicit instructions on how to implement the program; consequently, the pauses are much more evenly spaced with the most frequent inferred cause being *revisits requirements*.

Discussion

Pausing to recall problem specifications, language semantics, and verifying whether code made sense were the most frequent reasons that led to pauses in coding. Other notable categories included utilizing development environment affordances, identifying misconceptions, and identifying gaps in knowledge semantics which were all categories exhibited by participant 303. The data suggests that participants whose pauses fell in latter categories exhibited higher self-regulation as programmers.

From the 11 categories for pauses in coding we identified, we saw similar patterns emerge among different participants. This suggested that there was scope for developing a framework to explain pauses in coding. Prior work by Loksa and Ko (2016) discussed the

development of a theoretical framework for understanding the role of self-regulation in the problem-solving process and success. The steps of problem-solving are as follows: reinterpreting the problem, searching for analogous solutions, searching for solutions, evaluating a potential solution. This problem-solving framework has important implications on how programming is taught to novices. Analysis of pause data could help instructors determine conceptually challenging concepts, and what students are struggling with.

To further analyze our data, it would be valuable to identify the how much time is spent in each step when the participant is working on a problem. We noticed that participants' behavior was analogous to steps identified in the problem-solving framework. After reading the problem statement, one participant voiced: "so it is a comparator function" which told us that they were looking for analogous solutions based on their prior experience. Revisiting the problem requirements (one of the inferred causes we identified) also happened to be the most frequent coding of the pauses recorded.

To modify the study, we would start logging data as soon as the user reaches the question page instead of when the user clicks into the editor box. We lost data on the user's true start time because we were not logging this. We also would modify our scripts to produce charts where the axes are normalized for both the keystrokes over time and pause frequencies, in order to better visualize our data. We would also adjust our instrumentation as described in the appendices.

Studying pauses in writing code would have large implications in coding interviews and computer science education. Companies that perform technical interviews could apply this research to make better classifications of candidates. Universities on the other hand could use this research to determine what their students find difficult, and teach those topics more thoroughly.

Appendices

Appendix A - Original procedure

Current implementation of our tool only supports solutions coded in Java. 2 out of 3 participants expressed that they were more proficient with Python. In addition, the current implementation started logging user input as soon as the participant clicked on the editor; this was problematic because the participant might have been reading the problem statement. Our current analysis only focused on qualitative analysis pauses that are longer than 5 seconds. We also did not consider the correctness of participants' solutions to the problem and participants' confidence in the submitted solution.

Appendix B - Revised procedure

The next iteration of this study should be language agnostic. While we do not know whether language of choice affects pauses in coding, we can reason with considerable confidence that participants' proficiency with a language certainly affects pauses in coding. We would also like to adjust the instrument to start logging when the question loads, so scales are a little more accurate. For the next iteration of the study, we would like to analyze 1 second pauses as well. We also plan to gather additional data on participant's confidence about their submission. Further, in our analysis, we would like to revise our analysis process so that it also focuses on the correctness of participants' responses. Lastly, participants reported some confusion over the problem specification and we also noticed some typos. We plan on fixing these for the next iteration of this study.

Reference list

Bixler R. & D'Mello S. (2013). Detecting Boredom and Engagement During Writing with Keystroke Analysis, Task Appraisals, and Stable Traits. *In Proceedings of the 2013 International Conference on Intelligent User Interfaces (IUI'13)*. ACM, New York, NY, USA, 225–234. https://doi.org/10.1145/2449396.2449426

Caceffo, R., Wolfman, S., Booth, K. S., & Azevedo, R. (2016). Developing a Computer Science Concept Inventory for Introductory Programming. *In Proceedings of the 47th ACM Technical Symposium on Computing Science Education.* ACM, New York, NY, USA, 364-369. https://doi.org/10.1145/2839509.2844559

Charter E. (2003). The Use of Think-aloud Methods in Qualitative Research An Introduction to Think-aloud Methods. Brock Education Vol. 12, No. 2, 2003 68. https://doi.org/10.26522/brocked.v12i2.38

Ihantola P., Vihavainen A., Ahadi A., Butler M., Börstler J., Edwards S. H., Isohanni E., Korhonen A., Petersen A., Rivers K., Rubio M. A., Sheard J., Skupas B., Spacco J., Szabo C., & Toll D. (2015). Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. *In Proceedings of the 2015 ITiCSE on Working Group Reports (ITICSE-WGR '15)*. ACM, New York, NY, USA, 41–63. https://doi.org/10.1145/2858796.2858798

Keehner, M., Gorin, J.S., Feng, G., & Katz, I. R. (2017). Developing and Validating Cognitive Models in Assessment. In Rupp, A. A., & Leighton, J. P. (Eds.), *The Wiley Handbook of Cognition and Assessment: Frameworks, Methodologies, and Applications.* John Wiley & Sons. https://doi.org/10.1002/9781118956588

Leijten M., & Waes L. V. (2013). Keystroke Logging in Writing Research: Using Inputlog to Analyze and Visualize Writing Processes. *Written Communication 30, 3 (2013).* 358-392. https://doi.org/10.1177/0741088313491692

Loksa D. & Ko, A. (2016). The Role of Self-Regulation in Programming Problem Solving Process and Success. *In Proceedings of the 2016 ACM Conference on International Computing Education Research.* ACM, New York, NY, USA, 83-91. https://doi.org/10.1145/2960310.2960334

Parker M. C., Guzdial M., & Engleman S. (2016). Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. *In Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM, New York, NY, USA, 93-101. https://doi.org/10.1145/2960310.2960316

Vygotsky, L. S. (1962). Thought and language. (E. Hanfmann & G. Vaker Eds., Trans.). Cambridge, MA: MIT Press. https://doi.org/10.1007/BF02928399

Acknowledgements

We would like to thank Benjamin Xie from the University of Washington Information School for helping us brainstorm the project direction and proofread.

We would also like to thank Matthew Davidson from the University of Washington College of Education for providing the pilot study think-aloud protocol and conducting the three interviews.

This material is based upon work supported by the National Science Foundation under Grant No. 1703304, 1735123, 1314399, 1639576, 1829590, and 12566082.