On the Uses of Process Data in Psychometric Research: Response Process Validity, Theory-building, and Operational Research

Matthew John Davidson

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:

Min Li, Chair

Chun Wang

Amy J. Ko

Program Authorized to Offer Degree:

College of Education

University of Washington

Abstract

On the Uses of Process Data in Psychometric Research: Response Process Validity, Theory-building, and Operational Research

Matthew John Davidson

Chair of the Supervisory Committee:
Min Li
College of Education

Digitally-based assessments create opportunities for collecting moment to moment information about how students are responding to assessment items. This information, called log or process data, has long been regarded as a vast and valuable source of data about student performance. Despite repeated assurances of its vastness and value, process data research has not yet lived up to this promise. Interesting and essential work has been done to explore methods of analyzing process data, but less attention has been paid to how process data, and results from analyses of it, should be used. This dissertation addresses this gap by providing three examples of how process data can be used: to understand how non-construct related item features influence student response processes, to describe domain performance in an undertheorized domain, and to inform operational decisions about digital-only components of assessment items. Special attention will be paid to justifications for using process data in each example, as way of establishing preliminary guidelines to evaluating the use of process data for research purposes.

Acknowledgements

There are so many people without whose support, mentorship, or patience, this dissertation would not have been possible. First and foremost are my wife Leah, and son Sam. I simply could not have finished this document were it not for Leah's support and patience as I tried (with mixed results) to be a PhD candidate, husband, and father. And Sam may not yet be aware of how helpful he was, but he was the most motivating and distracting part of my life during this time. I would also like to thank all of my extended family, who I'm sure always meant well when they said "oh you haven't graduated yet?"

My committee, too, deserves thanks for their mentorship and guidance not just for this dissertation, but as I formed my identity as a scholar and researcher. Thank you to my advisor Min Li, who was always available to talk and help me think through things, as well as the support I received from Measurement and Statistics faculty members Elizabeth Sanders, Chun Wang, Oscar Olver-Astivia. Also a huge thanks to Amy Ko, who, despite not being my advisor, filled that role spectacularly, and was always so generous with her time, thoughts, and mentorship. I have also been lucky enough to have two amazing internships, with fantastic mentors. So thank you to NCCPA and Fen Fan, and AIR and Fusun Sahin.

Thank you too to all of the wonderful colleagues that I have met along the way, many of whom I have also collaborated with on specific projects or papers. These include, in particular order, Benjamin Xie, Greg Nelson, Ni Bei, Brett Wortzman, Ruoyi Zhu, Nate Abe, Dongsheng Dong, Stefania Druga, Alannah Oleson, Mara Kirdani-Ryan, Jayne Everson, Mina Tari, and Dastyni Loksa.

I would also like to thank the National Science Foundation for funding my graduate studies in exchange for work on three fascinating grants. Having that funding, and the

opportunity to work on those projects, gave me the space to develop my skills and values as a researcher.

Finally, I would like to thank all the participants in my studies, and in the studies whose data I was able to analyze. Without those students, and their willingness to be observed and have their information recorded, none of these projects would have been possible.

Table of Contents

Introduction

As more and more assessments shift to delivery via digital platforms, there are more and more opportunities for the collection of moment to moment data about how students are answering items. This moment to moment data is captured in log files that contain students' timestamped interactions with the testing platform. Data can be extracted from these log files based on the construct to be measured and the specific research questions being considered: this extracted information is called *process data*. Process data represent a potentially rich source of information about what students are doing when they respond to items, over and above the answer they provide.

Since process data has been collected, researchers have explored many aspects of analyzing and modeling process data. This literature has been far-ranging, investigating analytical techniques for process data from numerous disciplines, such as sequence analysis (Bergner, Shu, & von Davier, 2015), Bayesian networks (Levy, 2019), event history analysis (Chen, Li, Liu, & Ying, 2019), machine learning techniques like feature selection (He & von Davier, 2015) and boosting (Sinharay, Zhang, & Deane, 2019), and both semi hidden and hidden Markov models (Guo, Zhang, Deane, & Bennett, 2019, Blikstein et al., 2014, respectively). Some work has also adapted commonly-used psychometric models for use with process data, like clustering methods (Hao, Shu, & von Davier, 2015) and IRT models (Liu, Liu, & Li, 2018, Shu, Bergner, Zhu, Hao, & von Davier, 2017). In addition, a large body of work has focused solely on analyzing timing information extracted from process data (De Boeck & Jeon, 2019). This has been used to identify rapid guessing (Lu et al., 2020), examinee disengagement (Sahin & Colvin, 2020), and as an index of effort (Lundgren & Eklöf, 2021). Some research has also considered how process data could be used to help students learn, for example by using a

keystroke log to create a replay of a students' writing process (Ranalli, Feng, Chukharev-Hudilainen, 2018). These efforts have all contributed greatly to the literature on how to work with and analyze process data.

Largely missing from that literature, and work on process data in general, is in-depth discussion and examples of *how* and *for what purposes* process data can and should be used. The body of work with process data has now established a broad set of analytical methods for process data, but the specific uses of that data, and specific uses of results from those methods, remains ambiguous. This ambiguity is particularly present if looking for uses of process outside of supporting inference about students' knowledge, skills, and abilities.

While limited, there has been some amount of research and discussion of such uses. One notable exception is recent work to understand group differences using process data, such as by examining action counts (Eichmann et al., 2020), time spent in certain states (Guo et al., 2019), and modified DIF models to incorporate timing information (Ercikan, Guo, & He, 2020). However, many research papers that use process data begin with a discussion of the potential of process data to provide previously un-gatherable data about student performance, and some discussion of how such data could be used. For example, Jiang, Gong, Saldivia, Cayton-Hodges, and Agard (2021) begin their analysis of drag-and-drop items by pointing out how much additional information process data has about student performance. Similarly, Bergner and von Davier (2018) describe how process data could be incorporated into NAEP assessment, not just as "additional" information about student performance, but as the targeted outcome of the assessment. To date, these discussions of the promise of process data have not yet coalesced into a clear agenda for process data research.

Provasnik (2021) summarizes discussions at a 2019 Conference titled “Opportunity versus Challenge: Exploring Usage of Log-File and Process Data in International Large Scale Assessments”, which sought to address the fractured nature of research on process data in assessments. Two clear points of consensus among process data researchers that emerged were: 1) the need for a systematic approach to what exactly log files should capture, and 2) the need “to answer the question of how to use process data” (p. 3). Addressing that second gap, how and for what purposes process data should be used, is the primary contribution of this dissertation. By providing three examples of using process data for applications other than inferences about student ability, this dissertation will build on prior work on analysis methods for process data. The examples will extend the field’s understanding of what process data can be used for, calling attention to important issues in validity of process data interpretation and use, while also providing some suggestions about what information log files could and should be capturing.

The broad question this dissertation explores is *how can process data fulfill its promise*? That question is explored through three more specific questions:

1) How can process data be used in a way that is both valid and relevant to important psychometric and measurement questions?
2) How can process data be used to support both psychometric aims as well as basic research, for example by providing rich descriptions of phenomena?
3) How can process data be used for operational research?

Each question is chiefly tackled by one of the three papers comprising the dissertation; each paper is itself an example of how process data can be used, while also providing explicit discussion and justification of those uses. Two of the papers work with process and response data from the Computer Science (CS) domain. This was chosen for two reasons. First, as a domain,

CS has many open psychometric and measurement questions. Second, research in CS has established that learning how to write programs requires multiple skills, but these skills are not well understood as cognitive processes (Xie et al., 2019). Therefore, there are important opportunities for furthering psychometric understanding of CS items, and for describing processes that students use in CS. The first paper treats thinkaloud data as kind of process data. The thinkalouds were collected as university students answered a set of CS questions. The questions contained modifications that were intended to vary aspects of the item that may influence how a student responds, but that would not change the underlying construct being measured. Those thinkalouds were qualitatively coded, and the sequence of codes was considered an "action sequence" that represented the students' moment-to-moment activities in responding to the item. The analysis focused on understanding how those activities were influenced by the item modifications, showing how process data can be used to evaluate how items are functioning. This paper also demonstrates how process data, in the form of the coded thinkaloud transcripts, can be used as a source of evidence for student response processes, and considers how such data can be used in validation studies.

The second paper shows how process data can be used as a basis to develop and test theory about a construct. One difficult aspect of teaching and learning Computer Science is how to write programs. This paper argues that a chief difficulty is that program writing is not well understood as a psychological phenomenon. Drawing on theories and methods from research on the natural language writing process, this paper explores the extent to which the moment to moment structure of program writing is similar to that of natural language writing. This is based on an analysis of keystroke logs collected while university students completed assignments for an introductory programming course. The factor structure of variables extracted from those

keystroke logs is explored, and the findings suggest some promising directions for future research on program writing. This paper also shows how process data can provide rich information about the processes involved in the target construct, and, with further work, can be used to characterize performance in the domain.

The final paper uses process data from PISA 2012 items to demonstrate an approach to using process data in operational research, from generating operationally relevant research questions to operationalizing necessary concepts, and interpreting results for operational purposes. Much of the work on process data has focused on how that data can be used to make inferences about examinees' ability or "level" of the target construct. This paper demonstrates how process data can be used not as a supplement or alternative way of measuring ability, but as a way to understand how items themselves were working. In particular, two problem solving scenarios from PISA 2012 were analyzed, to see if any of the interactive components of those items seemed to present challenges for examinees. The results showed that some examinees may have had difficulty understanding how to use the interactive components of one of the scenarios. More broadly, this paper shows that process data, particularly well-designed process data collection, can be used to evaluate claims about how examinees are interacting with items.

The question of how to use process data is complex. Not only because it is context- and construct-dependent, but also because, for many domains, theory and predictions about the exact process(es) students follow in responding may be unclear. To the extent that that theory is ambiguous or non-existent, it poses an important threat to the validity of any interpretation of process data as an indicator of students' ability. As a result, the first and second papers in this dissertation focus on ways of using process data that do not rely on well-developed theory of domain performance, but still provide valuable information about items, students, and the target

construct. The approach described in the third paper benefits greatly from collaboration with subject-matter experts and item designers, and relies on their domain knowledge to generate valid interpretations of process data.

The specific results reported in the papers are not nearly as important a contribution as the approaches to thinking through what process data can be used for. These papers serve as instructive examples of how to use process data, and are important first steps in developing broader theories about how to use process data in ways that are valid, provide useful inferences, and ultimately help process data to fulfill its promise.

Each of the three papers follows, with a conclusion that discusses important findings and implications for how process data can and should be used in psychometric research.

References

Bergner, Y., & von Davier, A. A. (2019). Process Data in NAEP: Past, Present, and Future. *Journal of Educational and Behavioral Statistics*, *44*(6), 706–732. https://doi.org/10.3102/1076998618784700

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming. *Journal of the Learning Sciences*, *23*(4), 561–599. https://doi.org/10.1080/10508406.2014.954750

Chen, Y., Li, X., Liu, J., & Ying, Z. (2019). Statistical Analysis of Complex Problem-Solving Process Data: An Event History Analysis Approach. *Frontiers in Psychology*, *10*. https://www.frontiersin.org/article/10.3389/fpsyg.2019.00486

De Boeck, P., & Jeon, M. (2019). An Overview of Models for Response Times and Processes in Cognitive Tests. *Frontiers in Psychology*, *10*. https://doi.org/10.3389/fpsyg.2019.00102

Eichmann, B., Goldhammer, F., Greiff, S., Brandhuber, L., & Naumann, J. (2020). Using process data to explain group differences in complex problem solving. *Journal of Educational Psychology*, *112*(8), 1546–1562. https://doi.org/10.1037/edu0000446

Eichmann, B., Greiff, S., Naumann, J., Brandhuber, L., & Goldhammer, F. (2020). Exploring behavioural patterns during complex problem-solving. *Journal of Computer Assisted Learning*, *36*(6), 933–956. https://doi.org/10.1111/jcal.12451

Ercikan, K., Guo, H., & He, Q. (2020). Use of Response Process Data to Inform Group Comparisons and Fairness Research. *Educational Assessment*, *25*(3), 179–197. https://doi.org/10.1080/10627197.2020.1804353

Greiff, S., Wüstenberg, S., & Avvisati, F. (2015). Computer-generated log-file analyses as a window into students' minds? A showcase study based on the PISA 2012 assessment of

problem solving. *Computers & Education*, *91*, 92–105. https://doi.org/10.1016/j.compedu.2015.10.018

Guo, H., Zhang, M., Deane, P., & Bennett, R. E. (2019). Writing Process Differences in Subgroups Reflected in Keystroke Logs. *Journal of Educational and Behavioral Statistics*, *44*(5), 571–596. https://doi.org/10.3102/1076998619856590

He, Q., & von Davier, M. (2015). Identifying Feature Sequences from Process Data in Problem-Solving Items with N-Grams. In L. A. van der Ark, D. M. Bolt, W.-C. Wang, J. A. Douglas, & S.-M. Chow (Eds.), *Quantitative Psychology Research* (pp. 173–190). Springer International Publishing. https://doi.org/10.1007/978-3-319-19977-1_13

Jiang, Y., Gong, T., Saldivia, L. E., Cayton-Hodges, G., & Agard, C. (2021). Using process data to understand problem-solving strategies and processes for drag-and-drop items in a large-scale mathematics assessment. *Large-Scale Assessments in Education*, *9*. https://doi.org/10.1186/s40536-021-00095-4

Levy, R. (2019). Dynamic Bayesian Network Modeling of Game-Based Diagnostic Assessments. *Multivariate Behavioral Research*, *54*(6), 771–794. https://doi.org/10.1080/00273171.2019.1590794

Liu, H., Liu, Y., & Li, M. (2018). Analysis of Process Data of PISA 2012 Computer-Based Problem Solving: Application of the Modified Multilevel Mixture IRT Model. *Frontiers in Psychology*, *9*. https://www.frontiersin.org/article/10.3389/fpsyg.2018.01372

Lu, J., Wang, C., Zhang, J., & Tao, J. (2020). A mixture model for responses and response times with a higher-order ability structure to detect rapid guessing behaviour. *British Journal of Mathematical and Statistical Psychology*, *73*(2), 261–288. https://doi.org/10.1111/bmsp.12175

Provasnik, S. (2021). Process data, the new frontier for assessment development: Rich new soil or a quixotic quest? *Large-Scale Assessments in Education*, *9*. https://doi.org/10.1186/s40536-020-00092-z

Ranalli, J., Feng, H.-H., & Chukharev-Hudilainen, E. (2018). Exploring the potential of process-tracing technologies to support assessment for learning of L2 writing. *Assessing Writing*, *36*, 77–89. https://doi.org/10.1016/j.asw.2018.03.007

Sahin, F., & Colvin, K. F. (2020). Enhancing response time thresholds with response behaviors for detecting disengaged examinees. *Large-Scale Assessments in Education*, *8*. https://doi.org/10.1186/s40536-020-00082-1

Sinharay, S., Zhang, M., & Deane, P. (2019). Prediction of Essay Scores From Writing Process and Product Features Using Data Mining Methods. *Applied Measurement in Education*, *32*(2), 116–137. https://doi.org/10.1080/08957347.2019.1577245

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., Tan, A. H., Hwa, L., Li, M., & Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, *29*(2–3), 205–253. https://doi.org/10.1080/08993408.2019.1565235

Nonconstruct Item Features and Response Processes in Computer Science Assessments: Evidence From Think-Alouds and Sequence Analysis

Abstract

Validation studies of instruments to measure Computer Science (CS) have primarily focused on content validity. This study proposes a framework of non-construct related item features that are theorized to influence student response processes. Thinkalouds were conducted with university students studying CS as they completed assessment items that were modified according to the framework. Coded transcripts of the thinkalouds were analyzed according to the proportional frequency of codes, the code sequence dissimilarity, and sequential pattern mining. Results show that features influenced response processes most strongly on items that required code writing. Possible applications of the sequence analytic methods are discussed with regard to validity studies.

Introduction

Researchers in computer science (CS) education have worked to develop validated instruments to assess the knowledge and skills of students in introductory computer science courses (Tew & Guzdial, 2010). These instruments have been shown to measure important aspects of the construct through examinations of content validity, and research has investigated statistical properties of some items as well (Davidson et al., 2021, Parker et al., 2016, Xie, Davidson, et al., 2019). However, less work has been done to systematically examine item characteristics that may have an effect on students' responses, which has important implications for validity as well as test design (Hubley & Zumbo, 2017). This study proposes a framework for non-construct related item features and applies them to existing items to understand how varying item features influenced student response processes.

Item features are understood to be parts of the item that do not bear directly on the construct being measured. For example, whether an item presents multiple-choice answer options or asks for an open-ended response would be an item feature that can be varied without directly affecting the construct being measured. Drawing on research in psychometrics and computer science education, the item features framework captures a range of features for CS items, as shown in Table 1.

To understand how varying item features influences responses, this study aims to describe student response *processes*. Response processes are "something that is inferred from the observed patterns of test responses, is further explained by relevant theory, and represents a dynamic, unfolding mechanism" (Launeanu & Hubley, 2017, pp. 116-117). The use of response process data in test validation has been part of the APA/AERA/NCME testing standards since 1999, but as Launeanu and Hubley (2017) argue, psychometric research on response processes

has been somewhat limited. This study takes a preliminary step by applying theory from CS education research to support inferences about response processes present in student responses. This is intended to lay the foundation for future work to develop a model of response processes in the domain, as well as to highlight the influence of item features on those processes.

Three analytical techniques are used to examine how response processes changed as a result of item features: frequency analysis, sequence dissimilarity, and sequential pattern mining. Qualitative codes applied to the thinkaloud transcripts made up the data these techniques were used on. Analyzing the frequency of codes under each item feature allows for broad description of differences in the process students took to solve items, while glossing over idiosyncrasies of any specific sequence or set of sequences. Sequence dissimilarity measures, on the other hand, quantify how similar two entire process sequences are for each person-item combination. This means that the order of actions comes into the analysis, as well as any individual-specific approaches. Sequential pattern mining (SPM) is used to find the most discriminating subsequences for each item feature. Whereas the frequency analysis shows how the frequency of actions differ, and dissimilarity measures show if sequences differ, SPM shows how sequences differ in detail. Each of these methods provides a different way to answer the motivating research question: *Did student response processes change for different item features, and if so, how?*

Item Features Framework

Based on prior work in computer science assessment, as well as in psychometrics more broadly, the authors developed the following framework of item features. Each dimension is explained in more detail below, along with supporting literature.

Table 1

Item features framework

Dimension	*Category*			
Cognitive Demands	Reading Syntax	Reading Templates	Writing Syntax	~~Writing Templates~~
Openness	Closed	~~Semi-open~~		Open
Evidence of thinking	~~None~~	Inferred		Explicit

Note: Cells with a line through them were not tested as part of this study

Cognitive Demands

The first dimension is *cognitive demands*: the required skills for an item, including four possibilities as reading syntax, reading templates, writing syntax, and writing templates. Prior research in computer science education has argued these skills are distinct but overlapping and that there is a meaningful difference in complexity between syntax and templates (Nelson, Xie, & Ko, 2017, Xie, et al., 2019). The type of thinking involved in responding to an item has also been found to influence its difficulty (Kasto & Whalley, 2013).

Nelson, Xie, and Ko (2017) investigated the effects of explicitly teaching comprehension of code to participants in an introductory CS course. Results showed that participants taught to read first learned more and had less variation on their midterm scores. Xie et al. (2019) built on the work of Nelson et al. (2017) and proposed a theory of instruction for introductory computer science that distinguishes between four skills: reading syntax, writing syntax, reading templates, and writing templates. The utility of this theory to successful teaching of computer science was explored through an experimental curriculum designed according to the theory. Results showed that novice participants taught according to the proposed theory had deeper comprehension of

the material, and their performance was less prone to errors than participants taught in the typical manner.

The assertion that the cognitive requirements of an item influence its difficulty is reasonable on face. However, the cognitive demands dimension of the item features framework proposes a specific way of conceptualizing those demands which does not rely on a generic taxonomy of cognitive difficulty. Instead, the authors suggest a domain-specific set of cognitive demands for items measuring CS1 knowledge and skills. While the domain specificity of the demands limits the generalizability of our framework, the demands of reading and writing code provides assessment engineers with valuable information about item difficulty as well as enabling decisions about the coverage of skills in an instrument.

Openness

The second dimension is *openness*: the amount of flexibility for examinees to engage with process (what students do to solve an item), product (what students produce, e.g., a program), and output (what output or outcome is required), varying from open to closed. Research suggests that openness contributes to item difficulty in assessments of science (Baxter & Glaser, 1998), computational thinking (Zhong et al., 2016), and computer science (Sheard et al., 2013). In addition, decades of research in educational measurement has found some important differences between the difficulty of different item formats (Bennett, Rock, & Wang, 1991, Le Hebel et al., 2017, Wan & Henly, 201,).

Baxter and Glaser (1998) distinguish between an item's required "cognitive proficiencies", subject-matter knowledge, and the activity performed with/on the knowledge (p. 38). They propose a framework of thinking about item features that situates process skills on a

continuum from constrained to open, and content knowledge from rich to lean. These dimensions can be combined factorially (e.g. process constrained content rich, process open content lean, etc.). In determining the cognitive complexity of an item, they argue that analysis of the task itself must be combined with analysis of the scoring system, to ensure alignment between measurement objectives and elicited performances, in particular the quality of cognition.

Sheard et al. (2013) proposed a framework for investigating the difficulty of CS1 exam items. They suggest a framework to measure the complexity of an item by considering external domain references, explicitness, linguistic complexity, conceptual complexity, length of code, and intellectual complexity. All of the features were correlated with expert judgements of an item's overall complexity as well as expected difficulty. Similarly, Lonati et al. (2017) analyzed the difficulty of Bebras Tasks, part of an international challenge of informatics and computational thinking skills, and found evidence that items may have intrinsic difficulties, like the concepts or processes involved, as well as surface difficulties, such as the item format, mark scheme, and wording. This suggests that the openness and cognitive demands dimensions of our framework may influence item difficulty as well as response processes.

Zhong, Wang, Chen, and Li (2016) designed a framework for assessments of computational thinking. They propose 3 dimensions to consider, of which openness and process are relevant to the current study. Openness refers to the outcome and process for an item, which can be closed, semi-open, or open. Closed tasks have a single method to arrive at a defined outcome, while open tasks may have an undefined outcome and a variety of possible methods to arrive at a solution. The process dimension is how the item tries to capture information about students' processes. This framework suggests that item openness and the means of gathering evidence of student thinking can both have an influence on item difficulty and the quality of

information collected about students, so both are included as dimensions of the proposed framework.

Le Hebel et al (2017) set out to describe the difficulty of science items from PISA. They found that the cognitive complexity of an item, as well as the item's format, explained significant variance of scores. Results showed that the more open formats (open and closed response) were more difficult than the more closed formats (multiple and complex multiple choice), explaining 12.5% of the variance in scores. Wan and Henly (2012) compared multiple choice (MC) with free response (FR), short constructed response (S-CR), and extended constructed response (E-CR) items in a K-12 science assessment. They found that both types of CR items produced more information per minute as well as evidence that all item formats measured the same construct, suggesting that modifying item features is possible while not modifying the construct measured. Bennett, Rock, and Wang (1991) reached a similar conclusion after examining the equivalence of MC and FR items on the Advanced Placement Computer Science exam. They found that a single-factor solution was a parsimonious fit, suggesting that MC and FR items were measuring the same construct. These studies demonstrate that variations in openness may have an influence on difficulty; more importantly, they also provide evidence that it should be possible to vary item features while not modifying the construct an item measures.

Evidence of Thinking

The final dimension of the framework is *evidence of thinking*: the extent which an item provides explicit evidence of the student's thought process. Generally, items that ask students to explain their thinking may be subjectively perceived as more difficult, but may result in better performance because asking for explanations can lead to reflection on the approach used (Lewis, 1996, Lonati et al., 2017). "Explain in plain English" (EIPE) items have been studied previously

in the domain of CS assessment, and research has shown that there is a positive correlation between students' explanations of code and their ability to write code (Murphy et al., 2012). Whalley and Kasto (2013) found that, for code tracing and EIPE items, the properties of the block of code may not have much influence on the difficulty of the item, suggesting that the primary source of difficulty is the explanation itself.

It is therefore important to understand how asking students to explain thinking changes responses, because the information obtained from such an item can be so valuable. Obtaining more or higher quality evidence of student thinking often involves changing the format of the item, such as from multiple choice to an open response; this can influence the difficulty of the item (Bennett, Rock, & Wang, 1991, Le Hebel et al., 2017, Wan & Henly, 2012), while also providing different information for score interpretation and use (Zhong et al., 2016). However, asking students to explain their thinking may provide such information without changing the difficulty of the item.

Thinkaloud studies of programming

Thinkaloud studies are certainly not new in computer science education, though there are few examples of thinkalouds being used to explore items rather than individuals; much of the work has focused exclusively on novice programmers. For example, Teague and Lister (2014) investigated how novice programmers solved two specific items, describing students' understanding using neo-Piagetian theory. Whalley and Kasto (2014) used thinkaloud interviews to understand how students incorporated feedback when checking their solutions to 3 different programming problems and to describe their learning across 3 items. Allen, Bidjerano, and Hren (2017) used thinkaloud interviews with novice programmers as they solved a variety of different item types, such as tracing code, predicting outcomes, supplying missing code, reading and

explaining code, and writing code. The authors classified the cognitive activities that students engaged in on these items, and found that most of the students' time was spent explaining code. They further found that most students' understanding of the code was segmented, for example they could easily understand each in a series of statements, but not as easily what the general purpose of the set of series of statements was. Whalley and Kasto (2014) found the same with one of their participants. Izu, Pope, and Weerasinghe (2017) studied how CS2 and CS3 students reasoned about program behavior using a thinkaloud interview. They were narrowly focused on how these students reasoned about reversibility, an abstract concept related to coding that requires an understanding of core concepts like program state. Lister et al (2006) used thinkaloud interviews to explore how novice programmers understood reading items. They applied the SOLO taxonomy (Biggs & Collis, 1982) to transcripts to describe the depth of understanding. This set of studies illustrate the depth and detail of information that thinkaloud studies can provide. While these studies sought to explain how students reason and understand while they solve programming items, the present study focuses on how item features influenced students' response processes.

Response Processes

Previous research has used data about student response processes to better understand final responses and their associated scores. There are many examples of timing information being used to provide more insight into scores (see De Boeck & Jeon, 2019 for an overview); however, how long it takes to complete an item only provides limited information about what processes are involved. Thinkalouds, like those collected in the present study, can provide more detailed information about what is happening while a student is responding to an item. For example, Ercikan et al. (2010) used thinkalouds to understand whether surface features of test

items were contributing to differential item functioning (DIF). More recently, response processes on computer-based assessments have been examined through analysis of action sequences recorded in log files (e.g., Guo et al., 2019, Hao et al., 2015, Stadler et al., 2019). These studies use various analytical methods to summarize the specific steps students took in responding to item(s). Those steps can be combined with timing information (Guo et al., 2019), compared to ideal sequences (Hao et al., 2015), or to see whether students follow a specific strategy (Stadler et al., 2019). In each of these studies, whether looking at data from timing, thinkalouds, or action sequences, the purpose of the analysis has been to understand how students process was related to their product, in most cases a score.

The present study has a different goal. Rather than using the response process information to understand the response or score, this study uses the response process information to understand how the item features influence the response process. In terms of general goals, this study is more like an explanatory item response theory (De Boeck & Wilson, 2011) approach than previous work with response processes. In addition, while this study is based entirely on data collected through thinkalouds, the coded transcript of the thinkaloud is treated as an action sequence. This combines the detailed data from thinkalouds and the analytical methods for action sequences, which allows for straightforward quantitative comparisons of how response processes differed across the item features.

Method

Choosing Items

A series of concept inventories for introductory computer science courses were collected, and the items were put into an item database and coded by applying the item features framework.

Items were also coded according to the aspect of the construct being measured. Concept inventories were used because they have been empirically validated to measure conceptual understanding in the domain, and thus provide a strong foundation on which to build a theory of item properties. A set of 6 items were chosen that represented each of three cognitive demands and included only content that a first year CS student would be familiar with. These items were then modified so that each cell of the item features in Table 1 was included in the study. For example, item that required students to create a program was modified to include step by step instructions. Such an item is shown in Figure 1.

Figure 1

Example item and modified item

Consider the class:

```
public class Employee {
    int role;
    float salary;
}
```

Create a method with the following method header:

```
public int verifyEmployee(Employee e1, Employee e2);
```

The method has to obey the following rules:

R1: return -1 if e1's role is less than e2's and e1's salary is also less than e2's.

R2: return 0 if e1 and e1 have the same roles and salaries

R3: return 1 if e1's role is greater than e2's and e1's salary is also greater than e2's.

Consider the class:

```
public class Employee {
    int role;
    float salary;
}
```

Write code to:

1. Create a method `verifyEmployee` that takes parameters `Employee e1` and `Employee e2`
2. Write a statement that returns a -1 if e1's role is less than e2's and e1's salary is also less than e2's
3. Write another statement that returns 0 if e1 and e1 have the same roles and salaries
4. Write another statement that returns a 1 if e1's role is greater than e2's and e1's salary is also greater than e2's.
5. Write a final line that returns a 2 if none of the previous rules could be verified

Write your code below.

R4: return 2 if none of the previous rules could be verified Write your code below.	

Note: Left item is an "open" writing item, while the right item is a "closed" writing item. Both measure use of conditionals, but the right item has a line-by-line specification of the code participants need to write.

Thinkaloud interviews

Thinkaloud interviews were chosen because they illuminate response processes, including misconceptions, necessary skills, and problem-solving strategies (Keehner, Gorin, Feng, & Katz, 2017). Interview transcripts were first analyzed in relation to the item features framework, coding for feature-specific sources of difficulty. They were then analyzed in an exploratory mode: verbal content was summarized, key response processes were identified, and then a coding scheme was developed and applied (Launeanu & Hubley, 2017, Leighton, 2017).

Thelk and Hoole (2006) used think aloud interviews to explore the knowledge, skills, and processes involved in a set of items measuring quantitative and scientific reasoning skills, by coding interview transcripts according to the content involved, cognitive processes, and response strategies. This paper follows a similar procedure. Think aloud interviews can also provide evidence supporting claims about the response processes involved in responding to an item (Launeanu & Hubley, 2017, Padilla & Leighton, 2017). This study bases such claims on an analysis of qualitative coding of thinkaloud transcripts that focuses on the frequency of behaviors in student responses, working in an inductive mode as described by Launeanu and Hubley (2017).

Participants

Study participants were given a brief demographic survey upon completion of the interview. They were instructed that all questions were optional, so the collected demographic data represents only what participants were willing to self-identify, based on categories from the US Census. This information is presented in Table 2.

Table 2

Demographic information

	Experience Level			
	Expert	Intermediate	Novice	**Total**
Gender				
Female	3 (33.3)	3 (50.0)	5 (55.6)	11 (45.8)
Male	2 (22.2)	3 (50.0)	3 (33.3)	8 (33.3)
Not Reported	4 (44.4)	0 (0.0)	1 (11.1)	5 (20.8)
Ethnicity				
Asian	2 (22.2)	3 (50.0)	7 (77.8)	12 (50.0)
Hispanic	1 (11.1)	0 (0.0)	0 (0.0)	1 (4.2)
White	6 (66.7)	3 (50.0)	2 (22.2)	11 (45.8)
1st/Most Comfortable Language				
English	7 (77.8)	5 (83.3)	8 (88.9)	20 (83.3)
Other	1 (11.1)	1 (16.7)	1 (11.1)	3 (12.5)
Not Reported	1 (11.1)	0 (0.0)	0 (0.0)	1 (4.2)

Note: "Expert" participants were graduates students in CS or a related field; "intermediate" participants were 3rd or 4th year undergraduates studying CS; "novice" participants had recently completed introductory CS.

Booklet Design

The 6 original items and 12 modified items were split into booklets according to a Latin square design. This resulted in three booklets which were balanced in terms of the cognitive demands, openness, and evidence of student thinking. Each booklet had 2 reading syntax, 2

reading templates, and 2 writing syntax items. Booklets were also balanced by anticipated item difficulty, roughly judged by the count of concepts the item required. For example, an item which only had “conditionals” was expected to be easier than one which had both “conditionals” and “loops”.

Coding

Transcripts were made of participant thinkalouds, and then segmented according to the cognitive processes and problem-solving strategies involved. The first author trained a research assistant by segmenting one transcript together. The segmentation guide was then modified, and both individuals segmented a sample separately. Agreement was calculated by counting the number of segments with exact or majority agreement, divided by the total number of segments in the first author’s segmentation. This resulted in agreement of 77%. Each then segmented half of the remaining transcripts, with a brief check of those they did not segment.

A codebook was developed to capture distinct skills involved in learning computer science (Xie et al., 2019), problem solving stages (Loksa & Ko, 2016), metacognition (Ruiz-Primo et al., 2002), and code reading and writing (Lister et al., 2006, Xie et al., 2019). The set of codes posit that the response processes to solve items involved reading code, writing code, monitoring thinking, and problem-solving behaviors. The codebook was iteratively revised after being applied to transcripts. Then the first author and a research assistant coded a small subset of items together, discussing ambiguities and disagreements, and revising the codebook further. To calculate agreement, a sample of 6 items were coded separately, reaching agreement of 73%. The remaining items were coded separately, and then reviewed by the other coder. Table 3 presents a summary of the codes, and the full codebook can be found in Appendix A.

Table 3

Summary of coding categories

Process Category	Brief Description	Example sub-codes	Literature
Monitoring	directing attention, recognizing limits of understanding	applying strategy, noticing confusion,	Ruiz-Primo et al (2002)
Problem Solving	stages of arriving at a solution	read task requirements, eliminate answers	Loksa & Ko (2016)
Reading Semantics	basic reading of code	multistructural reading, tracing values	Lister et al (2006), Xie et al (2019)
Reading Templates	advanced reading of code	relational reading, recognizing template	Lister et al (2006), Xie et al (2019)
Writing Semantics	planning and code functionality	global planning, revising semantics	Xie et al (2019)
Writing Syntax	typing lines of code	typing syntax, revising syntax	Xie et al (2019)

Analysis

To understand how response processes were affected by modifying item features, the qualitative codes from thinkalouds were analyzed with three methods. Each method considers the thinkaloud data in different, but overlapping ways. The proportional frequency analysis tests whether the frequency of individual actions participants took (indicated by the qualitative code) differ across the item features. This considers whether the item feature affects response processes by looking at the counts of each action. Sequence dissimilarity analysis tests whether the sequence of actions was more similar among items with the same features than among items with different features. This shows if item features affected the sequence of actions participants took.

Finally, sequential pattern mining tests whether specific sets of actions distinguish response sequences from items with different features. This provides more insight into whether the sequence of actions was affected by item features, by showing which sets of actions were associated with a given feature.

For the proportional frequency of codes, χ^2 tests were conducted to see whether the difference of the proportion across item types was significantly different from zero. For example, the proportion of monitoring codes was computed for all open items, then compared to the proportion for all closed items. The χ^2 test used a Yates continuity correction because of uneven cell sizes, as well as family-wise Benjamini-Hochberg corrections to *p*-values to account for multiple comparisons, where each dimension of the features framework (e.g., openness) was considered a family.

Longest common subsequence (LCS) was chosen as the dissimilarity measure to highlight the difference in the order between two sequences (Studer & Ritschard, 2016). LCS computes dissimilarity by finding the longest set of elements, in the same order, that are present in both sequences. This is turned into a distance by taking the length of that common subsequence, multiplying it by 2 and then subtracting it from the combined length of the two sequences. Therefore, the distance can be interpreted as the number of elements in both sequences that were not paired. A dissimilarity matrix was computed that contains the distance for each pair of sequences in the dataset. This matrix was used to compute discrepancy, which allows for the application of ANOVA methods to compare the variability of groups of sequences (Studer et al., 2011). A sum of squares (SS) within a group of sequences can be computed (e.g., for all open items), as well as across groups of sequences. This leads to a straightforward application of the *F*-test as the ratio of the between-sequence SS to the within-sequence SS. A *p*-

value for the statistic was obtained through permutation tests, which involve randomly assigning sequences to groups and computing the *F* statistic 1000 times, providing an empirical sampling distribution for the statistic.

SPM seeks the most discriminant subsequence within each item feature. First, the frequency of each subsequence was determined. These frequencies were then analyzed with a Pearson independence χ^2 test, which measures the association between the presence of a subsequence and a grouping variable using a contingency table; in this case the grouping variable was an item feature (e.g. open or closed) (Ritschard, Burgin, & Studer, 2013). Subsequences are then sorted based on their χ^2 value, with the most discriminating subsequence having the largest value.

Two constraints must be placed on the data in order to make SPM tractable: minimum support, and maximum subsequence length. Minimum support stipulates the percentage of sequences that a subsequence must appear in in order to be included in the analysis. This helps to filter out exceptionally rare sequences, while also trimming the space of possible sequences for the χ^2 test. Like the support, limiting the maximum length of a subsequence limits the space of possible subsequences to analyze. This study stipulated that subsequences must appear in 50% of sequences, and that the maximum length of a subsequence was 3. These limits were chosen for a combination of substantive and computational considerations. Substantively, a maximum length of 3 guaranteed that sequences would remain interpretable, and support of 50% would lead to the discovery of subsequences that were more likely to represent a feature's general effect than idiosyncrasies of an individual participant's approach. Computationally, the analysis would not run successfully with support lower than 10% and maximum sequence length greater than 10.

All analyses were carried out in the *R* statistical software, using the TraMineR package (Gabadinho, Ritschard, Muller, & Studer, 2011).

Results

Results are presented separately for each of the proportional frequency, sequence dissimilarity, and discriminating subsequence analysis. Within each analysis, the results will be grouped by item feature. Student's processes are analyzed with two different sets of codes, both of which are described in Table 3: process categories, and complete sub-codes. The process categories summarize broad actions that students took, while the sub-codes are more specific descriptors, each belonging to just one process category. Different codes were used for different parts of the analysis, and those choices are made clear and justified.

Proportional Frequencies

Table 4

Summary of χ^2 tests of difference in proportions

	Openness		Evidence of Thinking		Cognitive Demands	
	χ^2	p	χ^2	p	χ^2	p
Monitoring	1.67	0.29	0.78	0.56	**8.62**	**0.01**
Problem Solving	2.87	0.18	0.27	0.73	**143.29**	**0.00**
Reading Semantics	0.41	0.52	**6.65**	**0.02**	**221.56**	**0.00**
Reading Templates	4.15	0.13	**17.80**	**0.00**	**64.13**	**0.00**
Writing Semantics	**20.68**	**0.00**	0.04	0.84	**240.11**	**0.00**
Writing Syntax	1.31	0.30	**18.64**	**0.00**	**405.83**	**0.00**

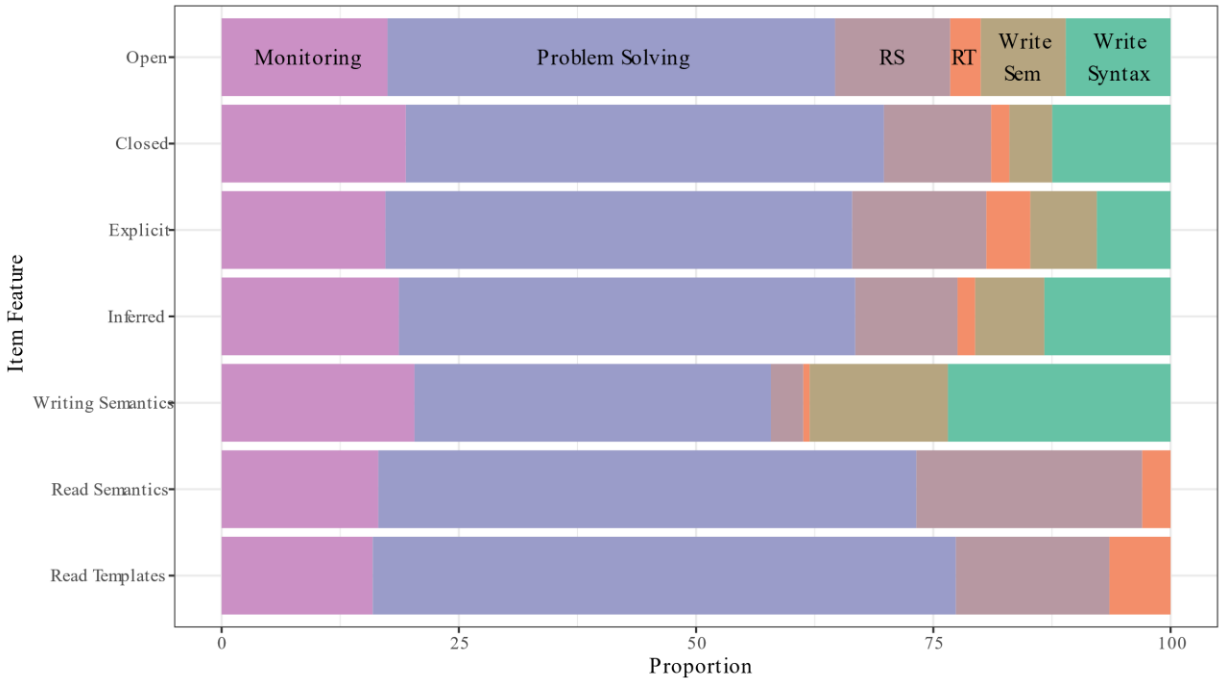
Note: Quantities in **bold** are significant at the $p < .05$ level.

Table 4 provides a summary of the χ^2 tests of the difference in observed proportions of each of the 6 response process categories. A glance at this table shows that varying item features did vary response processes, though in different ways for each item feature. For cognitive demands, the χ^2 test was a three-way test of the difference in the proportion of process steps between reading syntax, reading templates, and writing syntax. The significant result did not indicate precisely which demand was significantly different from the others, but simply that the difference between all three is greater than expected by chance.

Figure 4 shows the proportion of response processes across all item features. Each set of features will be discussed in detail, but it is apparent from a glance that there was variation across the features.

Figure 4

Proportion of response process categories for all item features



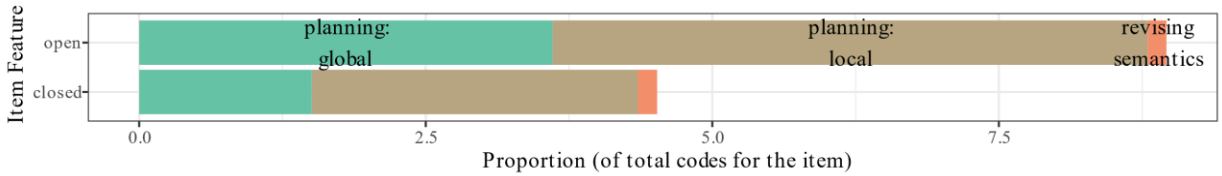
Note: RS = reading semantics, RT = reading templates, Write Sem = writing semantics.

Openness

Comparing open and closed items, the difference in the proportion of process steps was significantly different from zero only for writing semantics. This suggests that processes generally did not differ whether students were responding to an open or closed item, except for those steps related to writing semantics. This also means that the significant difference was only observed on writing items, since reading items did not involve any writing of code. Figure 5 shows the differences in writing semantics; open items saw more instances of both global and local planning than closed items, which suggests that open writing items elicited more planning behaviors from students.

Figure 5

Writing semantics processes for open and closed items



Evidence of Student Thinking

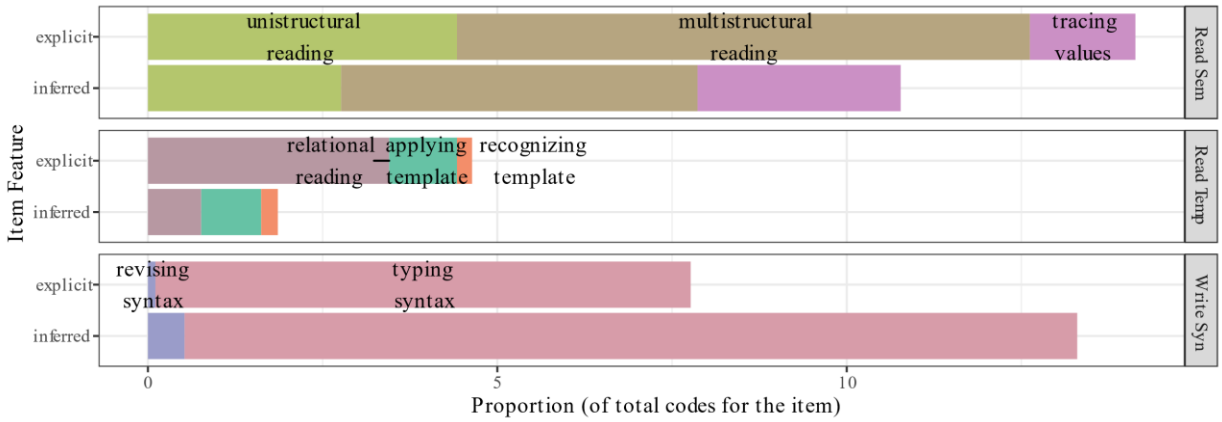
Comparing explicit and inferred evidence of student thinking, there were significant differences in the proportion of reading semantics, reading templates, and writing syntax codes; there was more reading in explicit items, while there was more writing syntax in the inferred items.

Figure 6 displays the reading semantics processes in more detail; there are notable increases in unistructural and multistructural reading in the explicit items. For reading templates processes, there was much more relational reading on the explicit items. This means that more

instances of reading (both semantics and templates) were observed for items that asked participants to explicitly explain their thinking. In addition to being more reading steps overall, there was a much larger increase in relational reading, an advanced type of reading code. This suggests that asking students to explain their thinking in addition to answering a question may have prompted more, and more advanced, reading of code than simply asking for an answer. The difference in writing syntax codes appears to be due to much more typing of syntax on inferred items than explicit items.

Figure 6

Detailed sub-codes for inferred and explicit evidence items



Note: Read Sem = reading semantics, Read Temp = reading templates, Write Syn = writing syntax.

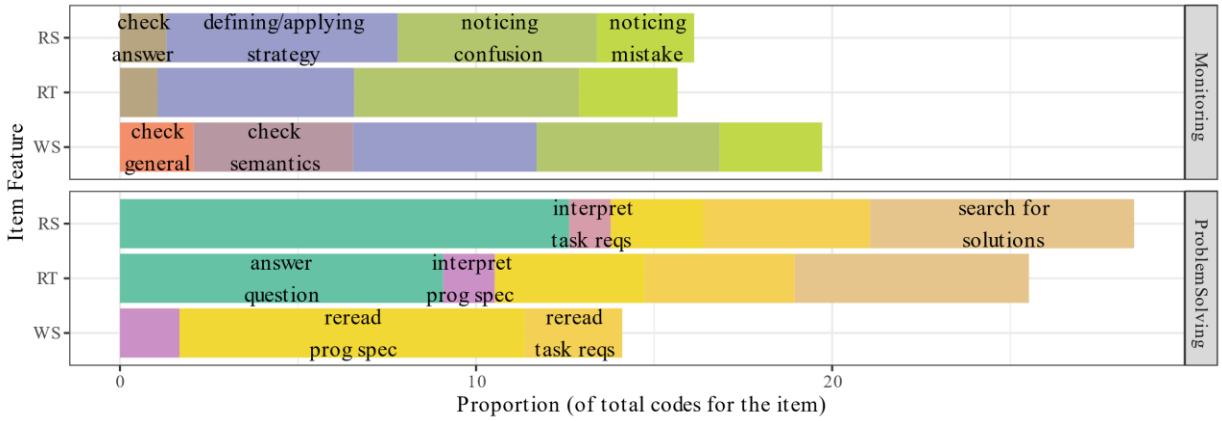
Cognitive Demands

Although there was a significant difference in all process categories when comparing cognitive demands, some of those differences were a result of how transcripts were coded. For example, neither reading semantics nor reading templates items were able to be coded with writing semantics or writing templates codes: so it is no surprise that there was a significant difference between those cognitive demands. Nonetheless, this provides solid evidence that items

have different cognitive demands that elicit different processes from students: the reading items did require more reading, and the writing items did require more writing. Furthermore, the reading templates items required more of the reading templates process steps than the reading semantics items. Figure 7 compares monitoring and problem solving for the three demands, to highlight differences in process besides just reading and writing.

Figure 7

Detailed sub-codes for different cognitive demands



Note: "task reqs" = task requirements; "prog spec" = program specification. "WS" = writing semantics; "RT" = reading templates; "RS" = reading semantics.

This shows that there was more monitoring overall on writing items, with most of the increase from "check general" and "check semantics". These actions are both reviews of a written program, either in general or its functionality, respectively. The other monitoring processes seem comparable across the cognitive demands. The biggest difference in problem solving is the absence of "answer question" and "search for solutions" for writing items, which again was a result of the codebook. A notable difference, though, is that writing items saw much more re-reading of the program specification than either type of reading item. This suggests that students had to refer back to the program specification more frequently on writing items than they did in solving reading items.

Sequence Dissimilarity

While the proportional frequency analysis combined process steps across individuals, the sequence dissimilarity analysis is based on the number of common steps, in the same order, between different action sequences. Because individuals responded to multiple items, and there can be substantial individual variation in response processes, a Levene's test of equality of variances was conducted with individual as the grouping variable. The *p*-value was 0.97, indicating that discrepancies were not significantly different across individuals: this means that any significant discrepancy was not likely to be a result of individual differences. Results are discussed for each item feature.

Table 5

Summary of Pseudo-ANOVA results for sequence dissimilarity measure

	Sum of Squares	*df*	Mean Square	*pseudo- F*	*p*
Openness (open, closed)					
Explained	0.56	1	0.55	2.42	0.004
Residual	29.74	129	0.23		
Evidence (inferred, explicit)					
Explained	0.31	1	0.31	1.32	0.141
Residual	29.99	129	0.23		
Cognitive Demands (RS, RT, WS)					
Explained	5.17	2	2.59	13.18	0.001
Residual	25.13	128	0.20		

Note: LCS distances were normalized to account for differences in sequence length; *p* values were determined through permutation test with 1000 replications. Parentheses after each comparison list the specific features being compared (e.g. for "Openness", open and closed items are compared).

Table 5 reports pseudo-ANOVA results based on dissimilarity between sequences, computed based on the process categories. Therefore, an example LCS might be "monitoring –

problem solving – reading semantics – monitoring". The same analysis was also conducted using the more detailed sub-codes. However, when using the sub-codes, it appeared that the homogeneity of variance assumption was violated for both openness and evidence of thinking. While a typical *F*-test might be robust to violations of that assumption if the group sizes are equal, it is not clear how robust pseudo-*F* tests might be. In addition, the group sizes were unequal; therefore, ANOVA results are reported based on the process categories rather than the detailed sub-codes.

Openness

The openness feature did seem to have an influence on sequence similarity, indicated by the statistically significant amount of discrepancy explained. This means that sequences for open items were more similar with each other than with sequences for closed items. This is both consistent with, and different from, the results from the frequency analysis. It is consistent because it points to the same conclusion: that there was a detectable influence of openness on response process. However, the pseudo-ANOVA result suggests that not only the proportion, but also the order of actions was more similar among open items as compared to closed items. This suggests that there might be some consistency in how individuals responded to items based on this item feature.

Evidence of Thinking

The pseudo-ANOVA results did not find significant discrepancy explained for evidence of thinking. Although the χ^2 tests did find that there were significant differences in the proportion of reading semantics, reading templates, and writing syntax codes for these items, the

pseudo-ANOVA result suggests that there were not consistent differences in the order of those codes.

Cognitive Demands

Comparison based on cognitive demands found significant discrepancy explained as well as significant differences in proportions of process categories. As with openness, this suggests that this feature explained not only a different proportion of process categories, but also a consistently different order of those process categories.

Discriminant Subsequences

While the ANOVA results are reported based on the process categories, discriminant subsequence analysis was conducted using the more detailed sub-codes. There are a few reasons for this. First, the frequency and dissimilarity analysis were much better suited to understanding the variation in process categories across the item features. Focusing on subsequences using the sub-codes allowed a finer-grained understanding of how response processes differed across item features. Discriminant subsequences are only reported for Openness and Cognitive Demands, because only these features had significant discrepancy explained in the dissimilarity analysis.

Tables 6 and 7 show the 10 most discriminating subsequences for Openness and Cognitive Demands. For each subsequence, we show the *p*-value and its frequency in responses for each item feature. While a subsequence with a *p*-value < .05 can be said to discriminate "significantly" between item features, even those with larger *p*-values can provide useful information about how item features influenced response processes. The lack of significant *p*-value means only that the result may not replicate. Considering the small sample size and

exploratory nature of this study, subsequences that did not discriminate significantly are included and considered in these results.

Openness

Comparing reading and writing items together did not provide any significantly discriminating subsequences, so reading and writing items were compared separately. For reading items, all subsequences had a *p*-value of 1, meaning there was no discriminating power to them. For writing items, while no subsequences were found to be significantly discriminating, some had p-values below 1, and are shown in Table 6.

TABLE 6

Subsequences that discriminated best between open and closed writing items

	Subsequence	*p*	Open	Closed
1	(typing syntax)-(planning: local)-(reread program specification)	0.23	0.75	0.19
2	(planning: local)-(typing syntax)-(reread program specification)	0.44	0.71	0.19
3	(planning: local)-(reread program specification)	0.61	0.75	0.25
4	(read program specification)-(planning: local)-(reread program specification)	0.61	0.75	0.25
5	(read task requirements)-(planning: local)-(reread program specification)	0.61	0.75	0.25
6	(typing syntax)-(planning: local)-(planning: local)	0.61	0.75	0.25
7	(planning: local)-(reread program specification)-(%)	0.87	0.71	0.25
8	(planning: local)-(reread program specification)-(typing syntax)	0.87	0.71	0.25
9	(typing syntax)-(reread program specification)-(planning: local)	0.95	0.75	0.31
10	(planning: local)-(typing syntax)-(typing syntax)	1.00	0.86	0.50

Note: Subsequences had to be present in at least 50% of sequences, and could not be longer than 3 actions. The "Open" and "Closed" columns show the frequency of that subsequence on open and closed items, respectively. Subsequences ending with a "%" indicate they are the end of the sequence, i.e. after this sequence the item was completed.

The five most discriminating subsequences were combinations of planning for a single line of code, typing syntax, and rereading the program specification; they were more frequently

observed on open items than on closed items. In each of those subsequences, line-based planning preceded rereading the program specification. These subsequences appeared much less frequently on closed items, which suggests that closed items may not have required as much checking back and forth between plans and the specification.

Cognitive Demands

Initially, all three cognitive demands were compared. However, this found that the most discriminating subsequences were largely artifacts of the coding: there was a process step called "answer question", for example, that was only applied for reading items, and was found to be highly discriminating. Then reading syntax and reading templates items were compared head to head, and no significantly discriminating subsequences were found. This led to combining all the reading items and comparing them to all the writing items. Even still, the most discriminating subsequences appeared in nearly all sequences for reading items, and in zero for writing items. Therefore, it seemed more instructive to view only the most discriminating subsequences which appeared in at least some of the writing items, which are shown in Table 7. These are the 11th through 20th most discriminating subsequences.

The first six subsequences listed in Table 7 were more frequent in reading items, and the common element in each is "multistructural reading", which is when a participant reads and understands syntax across multiple lines of code. For three of those subsequences, multistructural reading immediately precedes ending the item, which means that participants looked back at multiple lines of code before submitting their answer and moving to the next item. Subsequences 7, 9, and 10 were more frequent on writing items, and all include reading to understand the task or program specifications, but without direct reading of code (i.e. no multistructural reading).

This suggests that program specifications, while also present on reading items, were looked at more on writing items than on reading items.

TABLE 7

Subsequences that discriminated best between reading and writing items

	Subsequence	Read	Write
1	(multistructural reading)	0.89	0.39
2	(multistructural reading)-(%)	0.89	0.39
3	(read task requirements)-(multistructural reading)	0.78	0.32
4	(read task requirements)-(multistructural reading)-(%)	0.78	0.32
5	(read program specification)-(multistructural reading)	0.71	0.25
6	(read program specification)-(multistructural reading)-(%)	0.71	0.25
7	(read task requirements)-(read task requirements)-(read program specification)	0.52	0.95
8	(read task requirements)-(read program specification)-(multistructural reading)	0.70	0.25
9	(read task requirements)-(read task requirements)-(read task requirements)	0.39	0.84
10	(read task requirements)-(read program specification)-(read program specification)	0.37	0.80

Note: *p*-values for all subsequences were < .001, so are not displayed. Subsequences had to be present in at least 50% of sequences, and could not be longer than 3 actions. The "Read" and "Write" columns show the frequency of that subsequence in reading and writing items, respectively. Subsequences ending with a "%" indicate they are the end of the sequence.

Discussion

While there were important differences in the detailed results for each item feature, overall the analysis has demonstrated that varying item features did influence student's response processes for the CS items that were administered. The strongest evidence is for the cognitive demands feature, which posits that items that ask students to read code substantially differ from those that ask them to write code. Although this is not a surprising result, it is important that empirical evidence now supports the claim that such items are different. The second strongest

case is for the openness of an item, though the influence of that item feature seems to be larger on writing items than on reading items. Finally, there was mixed evidence for the influence of evidence of student thinking on processes. Results for each feature will be discussed in detail.

Cognitive Demands

Significant differences were found between all three cognitive demands in the dissimilarity and frequency analyses, and highly discriminating subsequences were found between reading and writing items. Some of those differences were as expected: participants wrote more on writing items and read more on reading items. Differing cognitive demands also elicited different response processes for monitoring, and problem solving: writing items had more monitoring than reading items, while the reverse was true for problem solving. The subsequence analysis found that program specifications were reread significantly more on writing items than on reading items.

These results show stark differences in response process based on cognitive demands. This has obvious implications for item design, but more importantly calls attention to ambiguity in the construct. The theory on which the cognitive demands feature is largely based (introduced in Xie et al., 2019) has not yet been tested widely. For example, it is not clear how much an instrument should assess each of the cognitive demands. It shows how important it is for the CS education field to continue work to develop robust, testable, and tested theories of how novices learn to program and how they demonstrate that learning.

Openness

The dissimilarity analysis found that sequences for open items were more similar with each other than with sequences for closed items. The frequency analysis found that open writing

items elicited more planning processes from students than closed writing items did; no differences were found for openness on reading items. In addition, the subsequence analysis showed that students more frequently did some planning and then checked the program specification on open writing items. Based on these results, it is reasonable to conclude that most of the differences observed were between open and closed writing items, and specifically differences in planning behaviors. Taken together, this means that specifying code line-by-line, as the closed writing items did, seemed to require both less planning overall, and less review of plans with respect to the specification.

This is important for designing items to measure code writing skills, since these results show at least three possible intended objects of measurement: simple translation of natural language to code, planning and executing a plan for a written program, or interpreting and implementing a specification. This feature also demonstrates the importance of looking at response process when considering an item's validity claims. A high score on an open item may be more indicative of a planning skill, while a high score on a closed item may indicate a translation skill. It is reasonable to argue that each is an important component of code writing, but a writing item may not be measuring both. This finding also has implications for a model of code writing in general, suggesting that translation and planning are distinct subskills of code writing.

Evidence of Thinking

The dissimilarity analysis did not find significant differences between the order of actions in sequences for inferred and explicit items. However, the frequency analysis did find differences: more instances of reading (both semantics and templates) were observed for items that asked participants to explicitly explain their thinking. In addition to more reading steps

overall, there was a much larger increase in relational reading, an advanced type of reading code. This suggests that asking students to explain their thinking prompts a greater amount of, and more advanced, reading of code than simply asking for an answer.

No discriminating subsequences were found for this item feature. While these results are mixed, there is some evidence that students were more deliberate in building their understanding of the code in reading questions. This has some implications for item designers who want to elicit student thinking, but also suggests to teachers or designers of instructional modules that asking participants to explain their thinking can lead to more close reading of code. Allen, Bidjerano, and Hren (2017) and Whalley and Kasto (2014) found that novice programmers spent most of their time explaining code, even for items where reading code was not the main objective. This shows that reading code is an important skill to be taught and assessed, and asking students to explain their thinking may both prompt, and allow for assessment of, a deeper reading process.

Conclusion

Understanding the influence of non-construct related item features on student response processes is critical for item design and valid interpretations of scores. Methods to analyze and compare processes are needed to support such inquiries. This study proposed a framework for item features in the CS domain, and examined that framework through the use of thinkaloud interviews, one type of process data. In addition, this study demonstrates three methods to analyze response process data: analysis of the frequency of process steps, comparisons of process dissimilarity, and identification of discriminating subsequences. The results from this study showed that open-ended items required students to do more planning, and that items where students had to explain their thinking required more reading of code. The significant differences

in student process based on cognitive demands supports the assertion in Xie et al. (2019) that there are discrete reading and writing skills in CS. This is an important result for item design and validity, but also suggests that future research on teaching and learning CS should explore these skills in more detail.

This study was exploratory in nature: robust theory about how students read and write code has not yet been developed or tested empirically in the CS domain, though there are recent efforts to do so (e.g. Malmi et al., 2019). Therefore, the analyses presented here were "data-driven" in the sense that there was not much theory to guide interpretation of the sequence data. This is in contrast with efforts like the NAEP TEL (e.g. Bergner & von Davier, 2018) or some PISA 2012 items (e.g. Greiff, Wustenberg, & Avvisati, 2015), where ideal sequences of actions could be created from theory related to the knowledge and skills the items tested, and then student's actions could be compared to those ideal sequences.

Despite that limitation, the three methods used in this paper illustrate the kind of detailed analysis that process data can support. If these items had been developed with a clear description of intended response processes, the methods in this paper could be used in a validation study to determine whether the items did indeed elicit those response processes. If the sample size had been larger, a DIF analysis could have been conducted which grouped students based on characteristics of their action sequence. This type of analysis could show whether there was a relationship between response processes and item difficulty or discrimination. Results could guide revision of the items to elicit specific responses from all student groups. And if the items were already eliciting the correct responses, analysis could provide formative information about how students could modify their process.

Process methods in general, and sequence analysis methods in particular, hold much promise for measurement. However, such methods should be applied with a great deal of care and close attention to the validity of the results.

References

Allen, J. T., Bidjerano, T., & Hren, T. (2017). What do novices think about when they program? *Journal of Computing Sciences in Colleges*, *33*(2), 171–181.

Bergner, Y., & von Davier, A. A. (2018). Process Data in NAEP: Past, Present, and Future. *Journal of Educational and Behavioral Statistics*, *44*(6), 706–732. https://doi.org/10.3102/1076998618784700

Biggs, J. B. & Collis, K. F. *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome).* Academic Press, 1982.

Baxter, G. P., & Glaser, R. (1998). Investigating the Cognitive Complexity of Science Assessments. *Educational Measurement: Issues and Practice*, *17*(3), 37–45. https://doi.org/10.1111/j.1745-3992.1998.tb00627.x

Bennett, R. E., Rock, D. A., & Wang, M. (1991). Equivalence of Free-Response and Multiple-Choice Items. *Journal of Educational Measurement*, *28*(1), 77–92. https://doi.org/10.1111/j.1745-3984.1991.tb00345.x

Clancy, M. J., & Linn, M. C. (1992). *Designing Pascal solutions: A case study approach*. Computer Science Press, Inc..

Gabadinho, A., Ritschard, G., Müller, N. S., & Studer, M. (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software, 40(4),* 1-37. http://dx.doi.org/10.18637/jss.v040.i04

Izu, C., Pope, C., & Weerasinghe, A. (2017). On the Ability to Reason About Program Behaviour: A Think-Aloud Study. *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, 305–310. https://doi.org/10.1145/3059009.3059036

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*. (pp. 75-101). John Wiley & Sons.

Greiff, S., Wüstenberg, S., & Avvisati, F. (2015). Computer-generated log-file analyses as a window into students' minds? A showcase study based on the PISA 2012 assessment of problem solving. *Computers & Education*, *91*, 92–105. https://doi.org/10.1016/j.compedu.2015.10.018

Hubley, A. M., & Zumbo, B. D. (2017). Response processes in the context of validity: Setting the stage. In *Understanding and investigating response processes in validation research* (pp. 1-12). Springer, Cham.

Launeanu, M., & Hubley, A. M. (2017). Some Observations on Response Processes Research and Its Future Theoretical and Methodological Directions. In *Understanding and investigating response processes in validation research* (pp. 93-113). Springer, Cham.

Launeanu, M., & Hubley, A. M. (2017). A model building approach to examining response processes as a source of validity evidence for self-report items and measures. In *Understanding and investigating response processes in validation research* (pp. 115-136). Springer, Cham.

Le Hebel, F., Montpied, P., Tiberghien, A., & Fontanieu, V. (2017). Sources of difficulty in assessment: Example of PISA science items. *International Journal of Science Education*, *39*(4), 468–487. https://doi.org/10.1080/09500693.2017.1294784

Leighton, J. P. (2017). *Using think-aloud interviews and cognitive labs in educational research*. Oxford University Press.

Lewis, E. L. (1996). Conceptual change among middle school students studying elementary thermodynamics. *Journal of Science Education and Technology*, *5*(1), 3–31.

Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin*, *38*(3), 118-122.

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

Lonati, V., Monga, M., Malchiodi, D., & Morpurgo, A. (2017). How presentation affects the difficulty of computational thinking tasks: an IRT analysis. *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 60-69. https://doi.org/10.1145/3141880.3141900

Malmi, L., Sheard, J., Kinnunen, P., Simon, & Sinclair, J. (2019). Computing Education Theories: What Are They and How Are They Used? *Proceedings of the 2019 ACM Conference on International Computing Education Research*, 187–197. https://doi.org/10.1145/3291279.3339409

Murphy, L., Fitzgerald, S., Lister, R., & McCauley, R. (2012). Ability to "explain in plain english" linked to proficiency in computer-based programming. *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, 111-118. https://doi.org/10.1145/2361276.2361299

Nelson, G. L., Xie, B., & Ko, A. J. (2017). Comprehension first: evaluating a novel pedagogy and tutoring system for program tracing in CS1. *Proceedings of the 2017 ACM Conference on International Computing Education Research Conference*, 2-11.

Padilla, J. L., & Leighton, J. P. (2017). Cognitive interviewing and think aloud methods. In *Understanding and investigating response processes in validation research* (pp. 211-228). Springer, Cham.

Parker, M.C., Guzdial, M., & Engleman, S. (2016). Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. *Proceedings of the 2016 ACM Conference on International Computing Education Research*, 93–101.

Ritschard, G., Bürgin, R. & Studer, M. (2013), Exploratory Mining of Life Event Histories, In J.J. McArdle & G. Ritschard (eds.), *Contemporary Issues in Exploratory Data Mining in the Behavioral Sciences* (pp. 221-253). New York: Routledge

Ruiz-Primo, M. A., Shavelson, R. J., Li, M., & Schultz, S. E. (2001). On the Validity of Cognitive Interpretations of Scores From Alternative Concept-Mapping Techniques. *Educational Assessment*, *7*(2), 99–141. https://doi.org/10.1207/S15326977EA0702_2

Sheard, J. (2013). How difficult are exams? A framework for assessing the complexity of introductory programming exams. *Proceedings of the Fifteenth Australasian Computing Education Conference*, 145-154.

Studer, M. & Ritschard, G. (2016). What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures. *Journal of the Royal Statistical Society, Series A*, *179*(2), 481-511. http://dx.doi.org/10.1111/rssa.12125

Studer, M., Ritschard, G., Gabadinho, A. & Müller, N.S. (2011), Discrepancy analysis of state sequences, *Sociological Methods and Research*, *40*(3), 471-510. http://dx.doi.org/10.1177/0049124111415372

Teague, D., & Lister, R. (2014). Manifestations of Preoperational Reasoning on Similar Programming Tasks. *Proceedings of the Sixteenth Australasian Computing Education Conference*, 65-74.

Thelk, A. D., & Hoole, E. R. (2006). What Are You Thinking? Postsecondary Student Think-Alouds of Scientific and Quantitative Reasoning Items. *The Journal of General Education, 55*(1), 17–39. https://doi.org/10.1353/jge.2006.0019

Wan, L., & Henly, G. A. (2012). Measurement Properties of Two Innovative Item Formats in a Computer-Based Test. *Applied Measurement in Education*, *25*(1), 58–78. https://doi.org/10.1080/08957347.2012.635507

Whalley, J., & Kasto, N. (2014). A qualitative think-aloud study of novice programmers' code writing strategies. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, 279–284. https://doi.org/10.1145/2591708.2591762

Xie, B., Davidson, M. J., Li, M., & Ko, A. J. (2019). An Item Response Theory Evaluation of a Language-Independent CS1 Knowledge Assessment. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 699–705. https://doi.org/10.1145/3287324.3287370

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., Tan, A. H., Hwa, L., Li, M., & Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, *29*(2–3), 205–253. https://doi.org/10.1080/08993408.2019.1565235

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An Exploration of Three-Dimensional Integrated Assessment for Computational Thinking. *Journal of Educational Computing Research, 53*(4), 562–590. https://doi.org/10.1177/0735633115608444

Program Writing as Natural Language Writing: An Analysis of Keystroke Logs

Abstract

Program writing can be one of the most challenging aspects of Computer Science for teachers to teach and students to learn. Research on CS education is often focused on program writing, developing discrete strategies to help learners, explain why it is difficult, or assess it without having students write. This paper argues that an underlying difficulty that all of this research faces is the lack of a comprehensive description of what program writing is, as a psychological phenomenon. To begin building that description, this paper draws on theories and research methods developed for natural language writing, to see whether those theories and methods can yield insights into program writing. More specifically, natural language writing research has been able to make inferences about writers' cognitive processes on the basis of data obtained from keystroke logs collected while composing. This study analyzed keystroke logs collected while students completed programming assignments, extracted variables from the data based on those used in research on natural language keystroke logs, and then investigated the factor structure of those variables. Results show that the cognitive process theory of natural language writing may be a promising theory to explain program writing, but that adaptations will be necessary.

Introduction

Learning how to write programs is an important part of learning computer science (CS). Research on learning CS has converged on a set of necessary skills for program writing, including tracing, explaining, and writing code (Lopez et al., 2008, Nelson et al., 2017, Xie et al., 2019). Each of these skills can be challenging for teachers to teach and students to learn. Students particularly struggle to learn program writing, especially those from non-dominant communities (DiSalvo et al., 2013). One strong focus of the research in the computer science (CS) education community is how to help novice programmers learn to write programs. The methods used to teach programming are both highly prescriptive and normative: diverse students are taught generic rules of programming language syntax and semantics, and then asked to solve arbitrary standard quizzes. In fact, writing is so difficult that assessments of program writing are moving away from having students actually write code (Du et al., 2020). Finally, there is little support for self-regulated learning of program writing, though there is growing recognition that metacognitive strategies can help students learn (Prather et al., 2020).

At the heart of all these issues is a lack of understanding of what program writing is. This paper addresses this by theorizing program writing as a cognitive process, to contribute a deeper understanding of the ways in which program writing relates to natural language (NL) writing. Lacking that deeper understanding makes it difficult to answer basic questions like: What does program writing look like when it goes well, or poorly? How can tools be designed to support it going well? Existing theories about program writing have been pedagogically focused: concerned with required skills (Lopez et al., 2008), or improving pedagogical practices (Xie et al., 2019). This has resulted in a series of promising findings that are difficult to understand as parts of a whole. To arrive at a description of program writing in itself, this paper applies the

cognitive process theory of writing (Hayes, 2012), to understand program writing. This will help guide research on program writing while helping deconstruct the program writing process, which can ultimately lead to more effective methods for instruction and assessment.

More specifically, this paper addresses a small part of this larger challenge by analyzing keystroke logs collected while student wrote programs. This will provide a preliminary comparison of how, on a process level, program writing differs from NL writing, and how those processes change over time. This may contribute to measurements of student's programming skills, by identifying useful diagnostic information that could be adapted to work at scale. Finally, applying theory developed for NL literacy will make a strong case for program writing as a similar but distinct literacy skill (Vogel et al., 2020), which may lead researchers and practitioners to draw on pedagogical strategies developed for NL literacy. Among other things, this could make space in CS classrooms for students to develop their identities as computer scientists while also learning “how to program”.

Program writing seems to be similar to NL writing in a number of ways, and this is especially true in an assessment context. Programming on an assessment is generally done in response to a prompt of some kind, which sets out a specification for the program and may include code that examinees must include or reference in their program. If the prompt requires a sufficiently complex program, students may come up with a plan for how to write a program according to that specification and then execute it; if the question is somewhat simple, students may just begin typing. It follows, therefore, that the processes a programmer engages in to write a program may be similar to, or the same as, those that they engage in to write a natural language essay. As a result, it may be that the very same features of the keystroke log can explain

performance in both domains. The similarities of programming and writing on assessments suggest, at least, that this may be a fruitful place to start an exploration of program writing.

Cognitive Psychology and the Writing Process

Cognitive psychology research in the writing process has informed much of the development of process features extracted from keystroke logs. The Hayes and Flower (1980) model of the writing process has been particularly influential. Updated in Hayes (2012), the most current version of the model specifies 4 basic writing processes: a proposer, which suggests ideas; a translator, which translates the proposed mental objects into language; a transcriber, which writes or types the language; and an evaluator, which monitors inputs and outputs of each of the three previous processes. Writers cycle between these processes continually as they write, and any process may interrupt the work of any other (Beauvais et al., 2011). For example, a writer may propose an idea for what should be written next, translate that idea to language, transcribe the language, and then re-read it to evaluate whether the written text conforms with the idea in their head. Or they may propose an idea and immediately evaluate the idea in their head, revise the idea, and then move forward with translation and transcription. In addition, how much of each process the writer engages in changes throughout the composition. Later in a writing task there tend to be more episodes of evaluation for revision, and fewer episodes of planning (Beauvais et al., 2011).

The Hayes and Flower (1980) model has guided work to make inferences about the writing process on the basis of features extracted from keystroke logs. One important area of research has been production pauses, or times when a writer stops producing text. (e.g. Chanquoy, Foulin, & Fayol, 1990, 1996, Chukharev-Hudilainen, 2014, Matsuhashi, 1981). This research has demonstrated that longer pauses (generally) reflect more challenging processing,

and that planning and evaluating are more effortful than translating (Olive, Favart, et al., 2009). Also, pauses and that pauses between paragraphs, sentences, and clauses are (generally) longer than those between individual words. Less experienced writers find it difficult to activate proposing or translating while they are transcribing, while more experienced writers concurrently engage in translation and transcription (Beauvais et al., 2011). Importantly, proposing and evaluating are typically activated during production pauses for skilled writers, though translating can be as well (Olive, Alves, et al., 2009).

Research on production pauses have found a typical behavioral structure of written composition: that writing occurs in bursts that take place between longer pauses (Chenoweth & Hayes, 2001, 2003). As a result of this, many of the features extracted from keystroke logs are related to bursts: the duration of a burst, the amount of text produced during a burst, or the type of burst (generally pause or revision as above, though other types have been explored). Timing data is also often extracted such as the duration within and between keypresses, words, sentences, or other meaningful structures like phrases and clauses. A final category of features are counts of specific events as well as ratios of those counts, like the number of keys pressed, the ratio of insertions to deletions, or the number of bursts. Although research has clearly demonstrated that more effortful processing takes place during these pauses, there is often not a direct connection between pauses of a certain type and exactly one cognitive process. However, combinations of these timing measures may be indicative of the processes, as discussed below.

Defining Pauses and Bursts

Earlier research on keystroke logs has focused on how to define pauses as well as the factor structure of sets of features. One issue is to define a threshold for what amount of time counts as a pause. Chenoweth and Hayes (2001) used the threshold of 2 seconds as indicative of

significant processing, and the boundary for a burst event (whether invention or revision). Chukharev-Hudilainen (2014) empirically arrived at a threshold of 500ms for pauses, and further argued that pauses above 1.2 seconds indicated higher-level processing. Medimorec and Risko (2017) investigated 3 different definitions of pauses (less than 1 second, between 1 and 2 seconds, and greater than 2 seconds) and explored relationships of those pause definitions with various aspects of writing. They found that the different pause thresholds had an influence on later analysis, for example finding stronger relations between pauses at sentence and paragraph boundaries as the pause threshold became larger.

In addition to identifying thresholds, the substantive definition of a pause or burst has been explored. For example, Baaijen, Galbraith, and de Glopper (2012) examined a sample of 80 keystroke logs to explore different ways of categorizing pauses and bursts to relate them back to the writing process. Rather than defining bursts based only on what happened immediately after the burst (as in Chenoweth & Hayes, 2006), they developed a number of burst types that were designed to isolate situations of linear text production (contrasted with non-linear, which involves jumping around in the text). They also introduced a third type of burst to capture inserted text based on where it has been inserted.

Underlying Structures of Keystroke Log Features

Research has also explored the factor structure of sets of features, both to eliminate redundancy in measures for later modeling, and to support substantive judgements of what features measure in the writing process. Baaijen et al. (2012) used principal components analysis to find factors in the burst data described above. Five factors were extracted, but three had much higher reliability than the rest. Those were planned sentence production, which included longer pauses at grammatical boundaries and very long pauses between words; within-sentence

revision, which was leading edge revision and nonlinear transitions; and revision of global text structure, between-sentence linearity, i.e. the extent to which global text is revised.

Almond, Deane, Quinlan, Wagner, and Sydorenko (2012) focused on pause behavior in the keystroke log, including burst lengths, in-word pauses, and pauses between words, sentences, and paragraphs. They defined a burst as having no greater than 2/3 of a second between events (like a keypress), and as being uninterrupted by cut/copy/paste events or the use of a mouse. Pause times were able to be modeled as mixtures of lognormal distributions, meaning that the observed pause times had multiple components that were drawn from different lognormal distributions. This suggests that variation in the pause time data is from multiple sources: the authors suggest that some was due to transcribing text, while a second mixture component was a result of "more complex cognitive activity" (p. 49). Baaijen et al. (2012) found a similar result, that a mixture of three distributions described pauses, which were theorized to indicate word retrieval, boundary processes, and higher-level planning. Both studies suggest that a single median fails to capture important information in the distribution of pause times.

Deane (2014) explored the factor structure of writing process features that were derived from keystroke logs in Almond et al (2012). Three factors were found, described as latency (text production speed including between and within word pauses), editing (time on task, between sentence pauses, and cut/paste/jump events), and burst span (means and standard deviations of burst length). Zhang and Deane (2015) also studied the internal structure of a set of keystroke log features. Factor analysis suggested four underlying factors in the process features, which the authors labelled general fluency (interkey and word-related pauses), phrasal/chunk-level editing (e.g. multiword deletion, deleted character), local world-level editing (e.g. minor edits, word choice), and planning and deliberation (e.g. time between bursts, time before starting).

Task-based Differences in Keystroke Features. Deane, O'Reilly, Chao, and Dreier (2018) use keystroke logs collected from two writing tasks to explore how process data could distinguish between the process students used for each task. One task involved listing words in relation to a topic, while the second was to write several sentences to demonstrate background knowledge on a topic. Features were extracted from the keystroke log to represent specific factors on the basis of prior research, which were the tempo of text production (at the burst level and at a lower level to indicate keyboarding and language skills), fluency of text production, and pause behaviors. Factor analysis was conducted on the set of features across all of the prompts, finding four factors: production effort, keyboarding effort, editing effort, and overall deliberation time.

Variation in features across different writing tasks has also been explored. Deane and Zhang (2015) undertook a study to examine the consistency of process features derived from keystroke logs, and whether those features could predict human essay scores. They found that features varied quite a bit across testing occasions, and had varying patterns of correlation with human scores depending on the genre and topic of the writing. The most stable features, time on task, burst length (in words), and pauses within words, had a moderate relationship with human ratings. This shows that pause behavior will vary between task types, and that behavior may not be predictive of the quality of the written work. Conijn, Roeser, and van Zaanen (2019) also found variation in keystroke features across two tasks (copying and academic summary). Most of the differences were in features related to time between words and sentences. Intervals within words did not change across tasks, but between words and sentences did when the cognitive demands of the task were different.

Relationships between Keystroke Log Features and Outcome Measures

Research has also examined the relationship of keystroke logs with outcome measures, such as the quality of the written product. Almond et al. (2012) conducted some preliminary work in this area, and found that timing features were moderately correlated with scores. Deane (2014) compared process features to product features from the e-rater automatic essay grading software as well as human scorings. All of the process factors explained variance in human scoring over and above the automatic product features, suggesting that process features reflect some individual differences that human scores do not. However, the process factors seemed to provide different information on different tasks. On an argumentative prompt, slow text production was associated with lower performance on literacy tasks, while on a literary analysis prompt, slower production was not as strongly associated with performance on those same literacy tasks.

Zhang and Deane (2015) also found process factors were related to automated scoring of product features like writing quality. The factors of fluency and planning/deliberation showed significant relation to the quality of the text. Deane and Zhang (2015) found that total time on task and standard deviations of burst length, between-sentence pauses, and cut/paste/jump events all seemed to be stably related to human judgements using a fundamental text production rubric. These results show that process features measuring fluency are related to product quality.

Allen et al. (2016) collected keystroke data while undergraduate students wrote in response to essay prompts. The authors computed a number of indices based on the keystroke logs, and found those indices accounted for 76% of the variance in essay quality ratings. The strongest correlated index was simply the number of keystrokes; consistently shorter pauses were also related to higher essay quality, suggesting that writers producing higher quality texts wrote more and had shorter pauses when they paused.

Sinharay, Zhang, and Deane (2019) used product and process features to predict essay scores on two different writing prompts. The authors use boosting, a procedure that constructs multiple regression trees in order to increase stability and increase the accuracy of out of sample prediction. A by-product of this method is that variables entered into the analysis are ranked in terms of their relative importance in making the score prediction (more on this method is included in response to question 2). Time on task and typing speed were found to be influential predictors in the models, with higher values for each leading to a higher predicted score.

Two studies found results that seem to contradict this relationship between fluency and written product. Galbraith and Baaijen (2019) argue for a distinction between events during linear and non-linear text production, and introduce global measures of text linearity to try to capture this feature of the log. They connect their proposed measures with text quality, and find that when participants worked without an outline higher quality text was associated with lower global linearity (indicating more revision of global structure), and with more controlled sentence production (i.e. longer pauses between sentences and less revision during sentence production). This result would seem to be in conflict with those just reported, since decreased linearity could be interpreted to mean decreased fluency. The second finding, that less revision during sentence production was related to higher quality text, conforms with the prior findings on fluency.

Deane et al. (2018) also found a different relationship between fluency as measured by the keystroke log and the quality of written product. By comparing keystroke logs across two task types, Deane et al. (2018) found that for both tasks neither keyboarding nor editing effort contributed to scores. This conforms with the tasks, which are assessing background knowledge through writing, rather than directly assessing writing fluency or quality. Differences were

observed in the features based on the task, with more latency between words on the word list task and generally more fluency on the task requiring students to write sentences.

A different approach entirely is taken by Brizan et al. (2015), who split process features into keystroke dynamics (timing information of individual keypresses and sets of keypresses), "stylometry" (linguistic measure of the text like sentence lengths, word lengths, lexical diversity), and language production (like burst length in words, pauses for specific punctuation, timing related to part of speech). Their set of features also include mechanical aspects of typing, e.g. which hand/finger likely typed the key. These features were found to capture some of the cognitive demand in the writing tasks (which had been established with a 6-level ordinal coding system based on Bloom's taxonomy), and to infer demographic information about the writer (whether they were a native English speaker, gender, and handedness). This provides additional evidence that pausing can be a reflection of effort, and that there are aspects of a keystroke log that capture unique information about the writer, even if that information may not be related to the quality of their written product.

Deane and Zhang (2020), reviewing research using keystroke logs, provide a broad overview of what writing skills seem to be indicated by extracted features. Typing speed can be a good measure of an orthographic planning and transcription skill, if the speed is measured when typing words the writer likely knows how to spell. Sentence-level planning and translation can be indicated by the variation of between-word pauses as well as the length of pauses between sentences. Planning more broadly can be measured by the length of the initial pause before writing (or before fluent writing begins), which writers often use to generate a plan before they begin composing text. Revising behaviors have been measured with some success through ratios

of deletions to insertions. Jumps in cursor position are clear indicators of monitoring and generally imply re-reading of the text.

In summary, prior work on keystroke logs of NL writing have primarily extracted and summarized information based on pausing, bursts, and counts. However, the operationalization of pauses, how to categorize bursts, and what counts are relevant has varied. Strong relationships have been found between features that measure fluency, like total time and typing speed, and outcome measures like judgements of quality. This relationship can vary, though, depending on the task. In the cognitive process theory, writing fluency may indicate a writer can deftly move between the different writing processes, which is an effective way to manage constraints like working memory (McCutchen, 2011). Despite different operationalizations, pauses have widely been found to be indicative of cognitive effort, with the pause location (e.g. between words vs. within words) and duration suggesting what the focus of the effort may be. Finally, with respect to the structure underlying keystroke features, there are some consistent findings that suggest a set of features measure fluency, but beyond that much seems to be context- and task-dependent.

Uses of Process Data in Studying Program Writing

Although it has not been as strongly guided by theory, efforts have been made to extract features from keystroke logs captured while students are programming. Like those used in writing, research has focused on the relationship of those features with various outcome measures. Most of the research has focused on digraph latencies (explained below) or on features obtained by comparing code snapshots and how those features change while composition occurs.

Thomas, Karahasanovic, and Kennedy (2005) used digraph latencies to explore whether keystroke logs could provide information about programming performance. A digraph is any set of two characters typed in succession. The word “for” contains two digraphs: “fo” and “or”, and

the latency is the time elapsed between the two keystrokes. The authors categorize keystrokes into a few types, which leads to a system to classify the digraphs. Characters can be alphabetic, numeric, control (e.g. ctrl), other (e.g. shift), or browsing. If the two characters in a digraph are the same, it is classified according to the character type; if they are different and none are browsing, they are classified as "edge" digraphs. Edge digraphs are more likely to be on the boundary of a chunk than the other digraph types. Median digraph latencies for each type were compared to summed scores on an assessment, and showed that alphabetic and edge digraphs were correlated around .50 with scores.

Digraph latencies have also been used to uniquely identify programmers. Leinonen et al. (2016) use similar features derived from logs collected across a whole course and a subsequent exam. Their goal was to identify individual programmers based on their keylogs as a test security measure for an in-class and a take-home exam. Classification was more accurate for the in-class exam than the take-home, above 90% and just below 90%, respectively.

Leinonen, Longi, Klami, and Vihavainen (2016) build on the work in Thomas et al. (2005) by analyzing keystroke logs collected during 7 weeks of a programming course. Observed latencies were related to exam scores and used to classify programmers as novice or experienced. All 7 weeks of data had a stronger relationship with written exam scores than any smaller set (e.g. weeks 1 to 4), but ultimately the correlation was weak for edge and numeric digraphs, at -0.23 and -0.17, respectively. They were also able to classify programmers as novice or more experienced with roughly 75% accuracy across three methods, again using a partial or whole dataset. The most predictive digraphs for this classification were those involving common commands (e.g. the digraphs in "i++"), common keywords (e.g. "true"), the use of shift (e.g. "{}"), or the speed of backspacing to various characters.

Differences in CS courses have also been compared using digraph latencies. Edwards, Leinonen, and Hellas (2020) compared process data from two introductory CS courses where the programming language, teaching approach, and spoken language differed. They used digraph latencies to predict exam performance, as well as to compare keystrokes across the contexts. The latter used the distribution of digraphs to find that there were detectable differences in frequency based on the programming language (e.g. “or” vs. “||” since the two languages are Python and Java), spoken language (more common digraphs in each language were faster) , and instructional approach (e.g. guidance on the use of whitespace between operators). In terms of the exam scores, a weak correlation (-0.20) was found for the Python course and a non-significant relation (-0.05) was found for the Java course.

Outside of the digraph work, Fwa (2019) used a set features similar to that used in analysis of writing keystroke logs to predict if a student would complete a given practice exercise in a set of sequential exercises. These features included the latency of keystrokes, count of deletes, and pausing information. Also included was the dwell time, the time between when a key is pressed until it is released. Unique to programming logs, the number of compile actions and the number of lines with errors were also included. Five different classification methods were used and compared. Only longer dwell time and higher exercise id number (i.e. a more difficult exercise) were found to be significant features in predicting non-completion of an exercise.

Digraph latencies show some relation with outcome measures, but overall seem to be somewhat limited in their explanatory power, both in terms of statistical explanation and substantive explanation. Put another way, outside of identifying a programmer and their general proficiency level, it is not clear what substantive information digraph latencies provide about the

process or product. Although Fwa (2019) found a relationship between dwell time and exercise completion, it is not clear whether or if any of the features examined relate to an outcome measure like quality of product.

Other studies have analyzed periodic snapshots of code in progress. This research has found metrics to drive automated feedback systems (Marwan, Gao, Fisk, Price, & Barnes, 2020), detect procrastination in time for early intervention (Kazerouni, 2020), and predict final exam scores (Blikstein et al., 2014). Marwan et al. (2020) created a system to provide students with positive feedback as they completed parts of a programming assignment or fixed bugs in their code. This feedback was prompted by the number of objectives students had completed already, or by a certain amount of idle time while students were working. Kazerouni (2020) found that procrastination could be predicted by the amount of software tests students had written by the time they had completed half of an assignment. Blikstein et al. (2014), were able to predict final exam scores by using the size and frequency of code updates. They modeled these code updates, and then compared students' pattern of code updates across 4 assignments, finding that students whose patterns changed significantly between assignments scored higher on midterm and final exams. Taken together, this work demonstrates that log-based analysis of program writing may help describe the process students engage in. Specifically, looking at how students spend their time during an assignment (e.g. producing code, revising code), how they add to code, and how frequently they run their code may all provide insight into the final submission.

Research Questions

The present study builds on this work in both NL and program writing, and takes features developed for keystroke logs of NL writing and applies them to keystroke logs collected during program writing. Prior studies of process data for program writing have focused on discrete

aspects of a student's written production, such as the digraph latencies, or on how the written output itself changes over time. The present study is unique in this space since it considers how different aspects of a student's program writing are potentially related to cognitive processes. This differs from the prior work not only because it includes multiple features that are intended to capture different cognitive processes, but also because it draws on a theory of written production from research on NL writing. One of the most basic and durable findings from work on NL writing, that longer pauses generally reflect more challenging processing or thinking, seems to be true for a variety of tasks involving a computer, including program writing (Fogarty et al., 2005). This finding, as well as some of the surface-level similarities between NL and program writing, form the basis of most of the features extracted from the logs of program writing (described more fully in Table 2). The specific research questions to be addressed are:

1) What is the underlying structure of a large set of features extracted from keystroke logs of program writing? Fluency is a consistently found dimension of NL writing features. Is that dimension present in program writing features? What other dimensions are present?
2) To what extent does that structure correspond to a hypothesized grouping of features as indicative of specific cognitive processes? If it does not, what other interpretations of the structure might there be?
3) Does that structure differ for different programming assignments? The NL writing research has found that task characteristics influence feature values. Is this also the case for program writing? If so, how does the structure differ? If not, what structure is consistent across different tasks?

Methods

Data

The data for this study are the keystroke logs collected while university students completed assignments for an introductory CS course. Students completed their coursework on a Python environment called Phanon, which logged each keystroke along with a timestamp, in addition to other data like pasting, switching between tasks, and running code (Edwards et al., 2020). This study focuses in on the final three assignments of the first half of the course. Each assignment included two separate tasks that received a single grade. These assignments were chosen because they asked students to use increasingly complex syntactic structures, and because they appeared after the first few weeks of instruction. This latter choice was made due to the potential for the program writing process to be more erratic as students were first learning, and with the expectation that variables extracted from the keystroke logs would be more stable as students become more capable programmers, a desirable property for measurement.

The first assignment's (referred to as assignment 1 herein) first task asked students to calculate the area of a regular polygon and display the result, after prompting a user to input the length and number of sides. The second task for assignment 1 had students prompt a user for a series of inputs about a fictional employee in order to calculate, and display on the screen, information about their pay like hours worked, pay rate, and deductions. The first task from assignment 2 required students to write a number guessing game, which given a guess from a user, will output specific messages based on how close the guess is. The second task asked students write a program that would ask a user to input 2 names, location (as an x, y coordinate) and radii, and then calculate if the circles overlapped. Assignment 3 had two tasks focused on while and for loops. The first required student to write a program that would find all numbers from 1 to 10,000 with a specific set of properties, and then display which numbers had those

properties. The second task asked students to write a program to simulate a series of random draws from the numbers in 1 to 6, and then display some results about the simulation. In addition to assignment grades, there are also grades for the midterm and final exam for each participant, all of which are summarized in Table 3. The actual assignments, along with scoring rubrics, are included in Appendix 1.

In order to be able to extract features from the keystroke log, the data had to be manipulated into a suitable format. First, data were cleaned. This included removing the entire log of any student with less than 100 keypresses, as well as removing any keypresses that were not recorded. This resulted in removing 23, 30, and 24 students for assignments 1, 2, and 3, respectively. After cleaning, the rows were grouped into bursts; discussion of how burst was operationalized appears below. Individual keystrokes were then combined into strings, so that each row in the data had the characters typed so far. This was done for each burst, allowing the production of each burst to be analyzed. In addition, the contents of each burst was categorized as being “text” or “code”.

One important issue this brought up was how to deal with comments in the code. While their purposes vary, typically comments are intended to explain the functionality of a line or group of lines of code. Although comments are generally not written in syntax, they may be considered part of the task of writing a program. For this analysis, comments were considered to be part of the code produced, and therefore were included in the analyses. In addition, bursts were categorized as being “code” or “text” depending on their contents. For example, a burst that contained a character with a parenthesis and no space, such as in “print()”, was categorized as code, while a burst containing only characters and numbers was considered to be text. This

categorization was primarily used in determining the phrasal burst, which was defined as a student's longest burst (in terms of characters produced) of code in the assignment.

Table 1.
Descriptive statistics for grades

	N	*M*	*SD*
Assignment 1	207	87.85	22.61
Assignment 2	200	95.33	16.02
Assignment 3	199	86.14	24.83
Midterm	--	73.98	13.68
Final	--	64.18	12.38

Note: *N* is students who had both valid log data and grades on each assignment after cleaning; *N* for the midterm and exam scores varies depending on which assignment is being analyzed.

Keystroke log features

After cleaning and formatting, features were extracted from the data. These features were based on the cognitive process model of the NL writing process, prior work with keystroke logs of NL writing, and research on process data for program writing. These features are listed in Table 2, along with a description of the feature, and organized by which of the four cognitive processes from Hayes (2012) the feature is thought to represent.

The set of features for the proposer is intended to capture idea generation in the text. This could be the number of bursts that are more than one line, or the number of characters per burst, both of which could indicate more or more complex ideas that had to be translated into written text. It could also be indicated by the amount of active time and the proportion of active time, with less active time associated with more idea generation. The phrasal burst, defined as the largest burst of code production in terms of characters typed, could also be a window into the proposer. Translation, according to Hayes (2012), is the source of bursts in text. Therefore, the set of features designed to capture translation measure the duration and frequency of bursts. For transcription, the features are all related to typing and brief pauses, such as those between spaces

and shorter than the long pauses. These briefer pauses may indicate thinking or processing related to transcription, where longer pauses tend to indicate more complex cognitive activities. Finally, the set of features for the evaluator are meant to capture editing behaviors like deleting, directing attention behaviors like switching between tasks, and evaluative behaviors like running code.

Table 2.
Planned features to be extracted from keystroke logs of program writing. Adapted from Sinharay, Zhang, and Deane (2019).

Feature name	Description
Proposer	
Multiline bursts	Count of bursts that included more than one line
Characters per burst	Total number of characters produced in bursts divided by the number of bursts
Active time	Total time spent active (inputting, deleting, running, etc)
Active proportion	Length of time spent not pausing divided by total time
Phrasal burst number	The burst number of the phrasal burst
Phrasal burst length	The number of characters in the phrasal burst
Phrasal burst duration	The duration of the phrasal burst
Translation	
Long pause median	Median duration of long pauses, defined as 10 times student's median pause
Burst duration	Median duration of bursts, defined by production after a long pause
End of line pause median	Median duration of pauses occurring before typing newlines
Burst count	Count of the number of bursts in the assignment
Transcription	
Keystroke count	Total number of keystrokes
Typing speed	Total number of input keystrokes divided by total time
Interkey interval	Median length of pauses between keystrokes
Pause after space median	Median length of pauses after space characters
Short pause median	Median duration of pauses greater than twice the median pause, but shorter than a long pause
Evaluator	
Run bursts	Count of bursts that included running the code at least once
Delete count	Count of delete (backspace, delete) actions
Delete proportion	Length of time deleting divided by total time
Input proportion	Length of time inputting divided by total time
Total time	Total time spent completing assignment
Task bursts	Count of bursts where switching between tasks at least once

Because the students were all university undergraduates, it is reasonable to assume that most were skilled writers, and therefore that translation and transcription processes were likely concurrent (as found in Olive & Kellogg, 2002). This means that, in terms of the factor structure, variables categorized under these two processes may well load together. Variables related to the evaluator process may also load on other factors as well, primarily because the evaluator may interrupt any of the other processes

These features were extracted initially for each burst and each student, resulting in a data frame with one row per student-burst combination (i.e. each student had multiple rows, one per burst in their log). That data frame was then summarized to give one row per student, meaning that each student had a single value for each of the variables in described in Table 2, for each assignment. This summarized data was used in subsequent analyses. Means and standard deviations for each of these features on each assignment are provided in Table 3 below.

Factor Analysis

There were two purposes for undertaking factor analysis on this set of variables extracted from the keystroke logs. The first was to see the extent to which the factor structure corresponded (or not) to the hypothesized relation of each variable to the four cognitive processes. The second purpose was to see whether the factor structure was similar to that found in work on NL writing, such as by Zhang and Deane (2015).

For each assignment, a parallel analysis was conducted to determine the optimal number of factors to extract from the set of variables. Next, an EFA was conducted to extract that number of factors, using an oblique rotation because of expected correlations between factors. At this point, the loadings of variables onto factors was inspected to see whether any factor seemed to represent one of the cognitive processes, which is reported in the results.

These factor structures were investigated further using confirmatory factor analysis. Due to the limited sample size, this analysis had to be conducted on the same datasets that the exploratory analysis was conducted on. Despite this, the factor models suggested by EFA did not produce well-fitting confirmatory models. Confirmatory factor models were also specified based on the four cognitive processes, with each process being one of the factors. These models also did not provide a good fit to the data. This result is taken up in the discussion.

Results

Based on how the keystroke features were operationalized in Table 2, the means and standard deviations of each extracted variable are presented in Table 3. These were calculated by averaging over all students in a given assignment, from the summarized data described above. This shows that there was some variation in these features across the three assignments. Although all three assignments were analyzed for this study, this goes into detail about the results for a single assignment because of the similarity between the findings. Results from the other two assignments will be contrasted with the focal assignment.

Table 3.
Means and standard deviations of all extracted keystroke log variables

	Assignment 1		Assignment 2		Assignment 3	
Variable	*M*	*SD*	*M*	*SD*	*M*	*SD*
Proposer						
Multiline bursts	13.97	9.39	14.33	10.82	16.04	12.51
Characters per burst	7.52	2.48	9.00	2.75	8.86	4.87
Active time	1751.12	917.21	1547.42	715.28	1579.98	1083.40
Active proportion	0.18	0.05	0.21	0.06	0.18	0.06
Phrasal burst number	214.30	237.11	245.14	215.49	285.69	310.39
Phrasal burst length	28.73	13.47	34.29	17.25	28.78	15.42
Phrasal burst duration	80.77	40.65	93.68	53.48	88.77	50.56
Translator						
Long pause median	5.09	1.63	4.27	1.11	4.95	2.05
Burst duration	2.66	0.95	2.70	0.72	2.62	0.75
End of line pause median	1.45	0.86	1.13	1.23	1.48	1.30
Burst count	426.83	230.32	339.29	166.42	369.04	265.52
Transcriber						

Keystroke count	5755.68	3053.81	5625.33	2665.57	5872.37	4075.72
Typing speed	2.59	0.76	2.88	0.76	2.85	0.89
Interkey interval	0.25	0.10	0.21	0.05	0.21	0.08
Pause after space median	0.22	0.12	0.19	0.08	0.20	0.11
Short pause median	0.81	0.28	0.70	0.18	0.71	0.22
Evaluator						
Run bursts	36.20	38.46	20.69	26.86	37.97	51.61
Delete count	1219.16	935.90	1126.74	644.43	1344.40	1074.01
Delete proportion	0.24	0.07	0.22	0.06	0.25	0.08
Input proportion	0.59	0.10	0.60	0.09	0.55	0.12
Total time	10336.28	5714.50	7636.00	4042.58	9262.53	6532.89
Task bursts	13.22	9.62	9.91	7.43	12.49	8.96

Note: Variables are defined as specified in Table 3. All times and durations are in seconds. Variables are grouped according to the cognitive process they may represent

EFA for Assignment 2

Parallel analysis indicated that four factors should be extracted from the data, and the loadings of each variable on those four factors is presented in Table 4.

This set of factors is able to explain 64% of the variance in the dataset in total. The first factor explained 25% of the variance, and nine variables loaded onto this factor with loadings > .300 (Bandalos, 2018). Among the variables loading onto this factor were the count of keystrokes, count of bursts, active time, count of deletions, and multiline bursts. These all seem to represent two underlying aspects: the length of a student's submission, indicated by the burst counts, and the amount of typing and time that it took to produce. Taken together, this can be interpreted as a factor representing the amount of activity a student engaged in on the assignment. The second factor explains 20% of the variance in the set of variables, and has 8 variables with loadings larger than .300. These include transcription measures like typing speed and interkey interval, as well as some translation measures like the median burst duration and the median long pause duration. This seems to represent a student's pausing, all the way from the interkey level up to the shorter pause and longer pause level, and ultimately the duration of bursts. The third factor explained 10% of the variance, with 4 variables loading > .300. These

variables represent a set of measures of student's efficiency, with the characters per burst, proportion of time inputting and active, and a negative relationship with the delete proportion. The final factor explained 8% of the variance, and only had two variables that loaded > .300: the number of characters and duration of the phrasal burst. These could be taken as a measure of how much code a student can produce in one burst.

Table 4.

Factor loadings for variables on assignment 2

Variable	Factor 1	Factor 2	Factor 3	Factor 4
Active time	**0.994**	0.263	0.228	0.000
Burst count	**0.970**	-0.092	-0.049	-0.073
Keystroke count	**0.939**	-0.243	0.139	0.041
Total time	**0.899**	0.180	-0.157	0.017
Delete count	**0.878**	-0.143	-0.060	-0.008
Phrasal burst number	**0.611**	-0.128	0.039	-0.050
Run bursts	**0.450**	**0.396**	-0.038	0.039
Multiline bursts	**0.443**	0.045	-0.155	0.008
Task bursts	**0.392**	0.070	-0.169	-0.002
Burst duration	0.003	**0.724**	**0.645**	0.011
Interkey interval	-0.013	**0.881**	0.002	0.032
Pause after space median	-0.033	**0.660**	0.078	-0.051
Typing speed	-0.052	**-0.888**	0.080	0.076
Short pause median	0.012	**0.913**	0.000	-0.098
Long pause median	-0.092	**0.777**	-0.191	0.026
Delete proportion	0.270	0.022	**-0.404**	0.039
Input proportion	0.013	-0.111	**0.566**	-0.008
Active proportion	0.011	0.105	**0.829**	0.013
Characters per burst	-0.092	-0.108	**0.662**	0.167
Phrasal burst length	0.011	-0.058	-0.053	**1.001**
Phrasal burst duration	0.008	0.381	0.047	**0.869**
End of line pause median	-0.085	0.219	-0.235	-0.071

Note: Loadings > .3 or < -.3 are noted in **bold**.

Similarity of factors across assignments. While the factor structure from EFA on assignment 2 seems to have reasonable interpretations, as noted above, a confirmatory model was not able to achieve acceptable fit to the data (e.g. for Assignment 1, RMSEA = .18, TLI = .69, CFI = .74). Although this is probably due to the small sample size, this severely limits what

additional analyses can take place. Nonetheless, the pattern of what variables load on which factors can be compared across the three assignments, as a limited way of understanding how stable this factor structure might be. Table 5 shows the variables and which factors they loaded on for each assignment.

Table 5.

Pattern of factor loadings across the three assignments

Variable	Assign. 1	Assign. 2	Assign. 3
Active time	1	1	1
Burst count	1	1	1
Delete count	1	1	1
Delete proportion	1	3	1
Keystroke count	1	1	1
Phrasal burst duration	1	4	4
Phrasal burst length	1	4	4
Phrasal burst number	1	1	1
Run bursts	1	1	1
Total time	1	1	1
Burst duration	2	2	3
End of line pause median	2		
Interkey interval	2	2	2
Long pause median	2	2	2
Pause after space median	2	2	2
Short pause median	2	2	2
Typing speed	2	2	2
Active proportion	3	3	3
Characters per burst	3	3	1
Input proportion	3	3	3
Multiline bursts	4	1	1
Task bursts	4	1	1

Note: Number in the cell indicates which factor (1, 2, 3, or 4) the variable loaded on to. For variables that had substantial (> |.3|) loadings on more than one factor, the highest loading was chosen for this table.

This table shows a good degree of agreement between the EFA models for the three assignments. No variable loaded differently for all three assignments, but one variable, end of line pauses, only had a substantial loading on assignment 1. The most consistently estimated factor was factor 2, which had 5 variables that loaded onto it for all assignments. Factor 4 was the least

consistent, with a different set of two variables loading onto it for assignment 1 and for assignments 2 and 3.

Discussion

It is clear that the pattern of loadings onto these factors does not coincide with the hypothesized set of cognitive processes described in Table 2. However, the results may still be interpretable as representing some of the cognitive processes. The first factor could be thought of as a measure of the proposer. Students who generate more ideas may have more time typing, and thus more keystrokes, as well as more bursts overall. They may also have produced more multiline bursts because of the density or complexity of the ideas they were generating. Even though the count of deletes seems a counterintuitive part of this factor, it may simply be the case that the more a student typed the more they had to delete. The second factor could be a measure of translation: it certainly includes pauses related to bursts, as hypothesized. It may be the case that other pause lengths, such as short or interkey pauses, and thus typing speed, all could indicate the efficiency of the translation process. The third factor may be a measure of transcription, since it includes the proportions of input, active, and characters per burst. Rather than direct measures of typing speed, transcription may be indicated in this factor by the proportion of time spent typing and the characters a student was able to type in a burst. The fourth factor, however, does not have a clear interpretation as indicating a cognitive process. It may, though, be a proxy for how much a student can produce before running into a limitation of one of the resources in writing. Hayes (2012) notes that the writing processes are constrained by the writer's attention, working memory, long-term memory, and reading skills. Long-term memory would include things like knowledge of the programming language or understanding of the program specification in the assignment. This factor does not suggest which of those

resources may be constraining production for these students writing programs, but transcription is a widely-recognized bottleneck in composing written text (Hayes, 2012, Hayes & Chenoweth, 2006).

Another interpretation of this factor structure comes from Zhang and Deane (2015). They found a somewhat stable factor across different tasks that represented a writer's fluency, which factor 2 in this analysis may represent. Like the factor found by Zhang and Deane (2015), factor 2 in this study included measures of pausing between spaces (which would include both between words and operators in the Python syntax). The addition of burst durations could still be consistent with this factor as a measure of fluency, since longer duration bursts could be a result of efficient control of a cycle of proposing, translating, and transcribing. If factor 2 is a measure of fluency, then factor 1 may be a measure of the student's overall activity on an assignment, factor 3 may indicate how efficiently a student was able to produce text, and factor 4 may be a proxy for a student's production capacity over a brief interval. As was noted above, writing fluency has been found to have a strong relationship with the quality of written text, because fluent production indicates a writer can efficiently coordinate the cognitive processes.

These interpretations of the factors, as indicators of some of the writing processes themselves or as indicators of more distant measures of the processes, are plausible based on the results of this study. This is a promising finding in itself, even though it is a very preliminary result. It suggests that theory developed for NL writing may be able to explain some of how program writing takes place. In particular, there seems to be some evidence that measures of NL writing, focused on pauses in production, overall production, and efficiency of production, may also be able to explain how program writing takes place.

One goal of this paper was to compare NL writing and program writing. The discovery of a fluency dimension in program writing, which has similar indicators as in NL writing, is promising. While this paper fall short of a full comparison, this finding does suggest that there is some merit to further investigation of the similarities and differences between NL writing and program writing. Future work could explore the relationship of program writing fluency to NL fluency, to help determine the extent to which fluency in program writing is determined by NL fluency. Investigating the relationship between program writing fluency and outcomes like grades or code complexity would also extend understanding of this fluency dimension.

Unfortunately, the results of this study are somewhat limited in what they can demonstrate about this factor structure. While far from a confirmatory analysis, the consistency of which variables loaded onto which factors across assignments provides some evidence that this factor structure may be found in additional samples. However, as mentioned above, none of the CFA models (the model suggested by EFA, or the model based on the four cognitive processes) were able to be fit to the data. Because neither of these factor models fit the data well, the relationship between the factors and student's grades was not able to be investigated—so even though there is some evidence of these factors in the data, it is not clear how those factors are or are not related to student's grades. There are a few possible reasons that the factor structure did not fit well. First, the sample size (around 200 individuals) is the lower end of guidance on necessary sample sizes for CFA. Considering the CFA model included 4 factors and 22 variables, there were about 4 individuals per parameter to be estimated, which may simply not have been enough. As a result, the observed variable values from this sample may have been particularly tricky to fit the model to. Another possibility, of course, is that one or both models were truly not good models for the data.

While EFA was able to provide some suggestions about what factors may underlie the keystroke variables, the evidence is limited. Another limitation is in the measures themselves. As discussed in the methods section, comments in the code were treated as "code" for this analysis. This means that at least some of the bursts that were found in the data are bursts of NL production as opposed to code. There are still some code-only measures, such as the phrasal burst, but NL writing was not entirely separated from program writing in this study. There are good reasons for that, as explained in the methods section, but future work may wish to explore this further.

In addition to replicating the present analysis with a larger sample size, future work could engineer additional program writing-specific features from the data, such as the kind of editing that took place after running the code. This study does include the count of bursts that include running code, but looking at editing behaviors could provide more information about how students are using the information they get from runs. Another important step in future work would be to explore code-specific measures of fluency. While this study drew fluency measures from NL writing literature, there may be some aspects of fluency in program writing that were not adequately captured by these. In addition, future work should look at outcome measures besides grades. These might be measures of the complexity of the code, the efficiency of the code at performing the computations in the specification, or others.

While far from conclusive, the results from this study were able to demonstrate the potential of adapting and modifying theories and measures from NL writing research to understand program writing. In particular, the cognitive process theory of NL writing may also be able to explain the structure and constraints of program writing. Variables based on measures used in NL writing research were extracted from keystroke logs collected while university

students completed programming assignments, and EFA was conducted to see whether the latent structure of those variables conformed to one of two possible structures. The first was a hypothesized categorization of each variable as indicating one of the four cognitive processes, and the second was based on prior work with NL writing keystroke logs by Zhang and Deane (2015). The EFA results were more in line with the structure found by Zhang and Deane (2015), with factors representing fluency, overall written production, and efficiency of written production. However, an interpretation of the factors as indicating cognitive processes was also justifiable, though which variables indicated which processes was not as predicted. Future work should explore whether these models can be confirmed with CFA and by examining the relationships of these factors with outcome measures like grades.

References

Allen, L. K., Jacovina, M. E., Dascalu, M., Roscoe, R. D., Kent, K. M., Likens, A. D., & McNamara, D. S. (2016). {ENTER}ing the Time Series {SPACE}: Uncovering the Writing Process through Keystroke Analyses. *Proceedings of the 9th International Conference on Educational Data Mining*, 22–29.

Almond, R., Deane, P., Quinlan, T., Wagner, M., & Sydorenko, T. (2012). A preliminary analysis of keystroke log data from a timed writing task. *ETS Research Report Series*, *2012*, i–61. https://doi.org/10.1002/j.2333-8504.2012.tb02305.x

Baaijen, V. M., Galbraith, D., & de Glopper, K. (2012). Keystroke Analysis: Reflections on Procedures and Measures. *Written Communication*, *29*(3), 246–277. https://doi.org/10.1177/0741088312451108

Bandalos, D. L. (2018). *Measurement theory and applications for the social sciences*. Guilford Press.

Beauvais, C., Olive, T., & Passerault, J.-M. (2011). Why are some texts good and others not? Relationship between text quality and management of the writing processes. *Journal of Educational Psychology*, *103*(2), 415–428. https://doi.org/10.1037/a0022545

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming. *Journal of the Learning Sciences*, *23*(4), 561–599. https://doi.org/10.1080/10508406.2014.954750

Brizan, D. G., Goodkind, A., Koch, P., Balagani, K., Phoha, V. V., & Rosenberg, A. (2015). Utilizing linguistically enhanced keystroke dynamics to predict typist cognition and demographics. *International Journal of Human-Computer Studies*, *82*, 57–68. https://doi.org/10.1016/j.ijhcs.2015.04.005

Chanquoy, L., Foulin, J.-N., & Fayol, M. (1990). Temporal management of short text writing by children and adults. *Cahiers de Psychologie Cognitive/Current Psychology of Cognition*, *10*(5), 513–540.

Chanquoy, L., Foulin, J.N., & Fayol, M. (1996). Writing in adults: A real-time approach. In G. Rijlaarsdam, H. van den Bergh, & M. Couzijn (Eds.). Theories, models, and methodology in writing research (pp 36-43). Amsterdam: Amsterdam University Press.

Chenoweth, N. A., & Hayes, J. R. (2001). Fluency in writing: Generating text in L1 and L2. *Written Communication*, *18*(1), 80-98.

Conijn, R., Roeser, J., & van Zaanen, M. (2019). Understanding the keystroke log: The effect of writing task on keystroke features. *Reading and Writing*, *32*(9), 2353–2374. https://doi.org/10.1007/s11145-019-09953-8

Deane, P. (2014). Using Writing Process and Product Features to Assess Writing Quality and Explore How Those Features Relate to Other Literacy Tasks: Writing Process and Product Features to Assess Writing Quality. *ETS Research Report Series*, *2014*, 1–23. https://doi.org/10.1002/ets2.12002

Deane, P., O'Reilly, T., Chao, S.-F., & Dreier, K. (2018). Writing Processes in Short Written Responses to Questions Probing Prior Knowledge: Writing Processes in Short Written Responses. *ETS Research Report Series*, *2018*, 1–30. https://doi.org/10.1002/ets2.12226

Deane, P., & Zhang, M. (2015). Exploring the Feasibility of Using Writing Process Features to Assess Text Production Skills: Using Writing Process Features to Assess Text Skills. *ETS Research Report Series*, *2015*, 1–16. https://doi.org/10.1002/ets2.12071

Deane, P., & Zhang, M. (2020). Automated Writing Process Analysis. In D. Yan, A. A. Rupp, & P. W. Foltz (Eds.), *Handbook of Automated Scoring* (1st ed., pp. 347–364). Chapman and Hall/CRC. https://doi.org/10.1201/9781351264808-19

Edwards, J., Leinonen, J., & Hellas, A. (2020). A Study of Keystroke Data in Two Contexts: Written Language and Programming Language Influence Predictability of Learning Outcomes. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 413–419. https://doi.org/10.1145/3328778.3366863

Fogarty, J., Ko, A. J., Aung, H. H., Golden, E., Tang, K. P., & Hudson, S. E. (2005). Examining task engagement in sensor-based statistical models of human interruptibility. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 331–340. https://doi.org/10.1145/1054972.1055018

Fwa, H. L. (2019). Predicting Non-Completion of Programming Exercises using Action Logs and Keystrokes. *2019 International Symposium on Educational Technology (ISET)*, 271–275. https://doi.org/10.1109/ISET.2019.00064

Galbraith, D., & Baaijen, V. M. (2019). Aligning keystrokes with cognitive processes in writing. In E. Lindgren & K. P. H. Sullivan (Eds.), *Observing writing: Insights from keystroke logging and handwriting* (pp. 306-325). Brill. https://doi.org/10.1163/9789004392526_015

Grabowski, J. (2008). The internal structure of university student's keyboard skills. *Journal of Writing Research*, *1*(1), 27–52. https://doi.org/10.17239/jowr-2008.01.01.2

Guo, H., Zhang, M., Deane, P., & Bennett, R. E. (2019). Writing Process Differences in Subgroups Reflected in Keystroke Logs. *Journal of Educational and Behavioral Statistics*, *44*(5), 571–596. https://doi.org/10.3102/1076998619856590

Hayes, J. R. (2012). Modeling and Remodeling Writing. *Written Communication*, *29*(3), 369–388. https://doi.org/10.1177/0741088312451260

Hayes, J. R., & Chenoweth, N. A. (2006). Is Working Memory Involved in the Transcribing and Editing of Texts? *Written Communication*, *23*(2), 135–149. https://doi.org/10.1177/0741088306286283

Hundhausen, C. D., Olivares, D. M., & Carter, A. S. (2017). IDE-Based Learning Analytics for Computing Education: A Process Model, Critical Review, and Research Agenda. *ACM Transactions on Computing Education*, *17*(3), 1–26. https://doi.org/10.1145/3105759

Leinonen, J. (2019). *Keystroke Data in Programming Courses*. University of Helsinki.

Leinonen, J., Longi, K., Klami, A., Ahadi, A., & Vihavainen, A. (2016). Typing Patterns and Authentication in Practical Programming Exams. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 160–165. https://doi.org/10.1145/2899415.2899472

Leinonen, J., Longi, K., Klami, A., & Vihavainen, A. (2016). Automatic Inference of Programming Performance and Experience from Typing Patterns. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 132–137. https://doi.org/10.1145/2839509.2844612

Marwan, S., Gao, G., Fisk, S., Price, T. W., & Barnes, T. (2020). Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science. *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 194–203. https://doi.org/10.1145/3372782.3406264

McCutchen, D. (2011). From Novice to Expert: Language and Memory Processes in the Development of Writing Skill. *Journal of Writing Research*, *3*(1), 51–68. https://doi.org/10.17239/jowr-2011.03.01.3

Medimorec, S., & Risko, E. F. (2017). Pauses in written composition: On the importance of where writers pause. *Reading and Writing*, *30*(6), 1267–1285. https://doi.org/10.1007/s11145-017-9723-7

Olive, T., Alves, R. A., & Castro, S. L. (2009). Cognitive processes in writing during pause and execution periods. *European Journal of Cognitive Psychology*, *21*(5), 758–785. https://doi.org/10.1080/09541440802079850

Olive, T., & Kellogg, R. T. (2002). Concurrent activation of high-and low-level production processes in written composition. *Memory & Cognition*, *30*(4), 594–600.

Piech, C., Sahami, M., Koller, D., Cooper, S., & Blikstein, P. (2012). Modeling how students learn to program. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 153-160. https://doi.org/10.1145/2157136.2157182

Ranalli, J., Feng, H.-H., & Chukharev-Hudilainen, E. (2018). Exploring the potential of process-tracing technologies to support assessment for learning of L2 writing. *Assessing Writing*, *36*, 77–89. https://doi.org/10.1016/j.asw.2018.03.007

Sinharay, S., Zhang, M., & Deane, P. (2019). Prediction of Essay Scores From Writing Process and Product Features Using Data Mining Methods. *Applied Measurement in Education*, *32*(2), 116–137. https://doi.org/10.1080/08957347.2019.1577245

Thomas, R. C., Karahasanovic, A., & Kennedy, G. E. (2005, January). An investigation into keystroke latency metrics as an indicator of programming performance. In *Proceedings of the 7th Australasian conference on Computing education 42*, 127-134.

Xu, C., & Xia, J. (2021). Scaffolding process knowledge in L2 writing development: Insights from computer keystroke log and process graph. *Computer Assisted Language Learning*, *34*(4), 583–608. https://doi.org/10.1080/09588221.2019.1632901

Zhang, M., & Deane, P. (2015). Process Features in Writing: Internal Structure and Incremental Value Over Product Features. *ETS Research Report Series*, *2015*, 1–12. https://doi.org/10.1002/ets2.12075

Zhu, M., Zhang, M., & Deane, P. (2019). Analysis of Keystroke Sequences in Writing Logs. *ETS Research Report Series*, *2019*, 1–16. https://doi.org/10.1002/ets2.12247

Process Data in Operational Research: Untapped Potential for Understanding how Items Function

Abstract

Process data, or moment to moment data captured while examinees respond to assessment tasks, has become an important focus of psychometric research. While it is relatively easy to capture large amounts of process data, appropriate analytical methods and uses of such data are not as straightforward. Much research on process data has focused on what analytical methods can be used, while less attention has been paid to what uses of process data may be valid. And much of the research on analytical methods has focused on using process data to make or support inferences about examinee ability. This study proposes, instead, to use process data to answer questions with operational relevance, such as whether items are functioning as intended. In particular, the suitability of process data to provide evidence of how examinees use interactive components of items is explored using data from two problem-solving scenarios on the PISA 2012 administration. Results show that process data may have been able to detect some examinees who were confused about how to use interactive components on these items. More importantly, this study demonstrates how to use process data in support of operational research questions by carefully choosing how to operationalize concepts and interpret results.

Introduction

Process data has become an important data source in psychometric research (Provasnik, 2021). It can be defined as a moment-to-moment collection of what examinees are doing as they interact with assessment tasks. This is a purposefully broad definition, since process data can be collected on both more traditional items like multiple choice questions as well as on newer assessment formats like interactive scenarios. As more and more assessments are delivered on digital devices, opportunities to collect process data have expanded greatly. And as more data has been collected, researchers have explored methods to analyze process data. However, less attention has been paid to how results of process data analysis might be used; and this is particularly true for operational research (Provasnik, 2021).

In general, operational research focuses on whether items and tests are working as designed, have good psychometric properties, and function similarly for different groups of test-takers: based on the results of that research, items may be removed or revised. It is often characterized by a narrow focus on a single test or item, primarily aimed at supporting operational decisions, and focused on an internal audience such as test owners and item developers. Operational research does often have applicability outside of the specific test or item that is the object of analysis, but the goal generally not to create widely generalizable findings.

While process data could be used to support any of these research goals, this paper will consider how process data can be used to evaluate the extent to which assessment items are working as designed. By looking at two scenarios from the PISA 2012 problem solving assessment to see how interactive components of the items are used by examinees, this paper will demonstrate how process data can be used in assessment operations. The goal is to understand how examinees' used interactive components of items; whether there were differences in the

responses of examinees who used those components compared to those who did not; and finally whether there are patterns in component use that suggest confusion with the component.

Process data, particularly that is collected from large-scale assessments, has largely been analyzed to support inferences about scoring, ability, and/or what students know and can do. However, process data can offer valuable insights about student performance for operational research. This includes validity evidence from student response processes, and data to test hypotheses about item designs and how students interact with items. One important result of operational research with process data can be improvements to data collection on the testing platform, which could allow more robust analyses of process data in future administrations.

While the collection of process data is not a new aspect in psychometric research and operational testing, much of the work remains exploratory in nature. Indeed, many studies focus on the feasibility of applying different analytical methods to the data to see what sorts of analyses are possible. In addition to exploring methods to use, other work has investigated how timing information can be used, while a more recent set of studies use process data to examine group differences. In many of these studies, the goal is to understand how process data can be used to make, or to support, inferences about examinee ability. Operational research, on the other hand, tends to be more focused on understanding items than about modeling student abilities.

The next sections will show examples of how process has typically been analyzed, describe some typical aspects of operational research supporting the test development process, and finally discuss prior research on the focal PISA 2012 problem-solving scenarios.

Analytical Methods for Process Data

Exploration of how to analyze process data has drawn from a number of different disciplines, one of which has been machine learning methods. For example, Qiao and Jiao (2018)

compared the performance of a variety of classification techniques on data from the PISA 2012 "Tickets" task. Other studies have explored how sequence analysis methods can be used to analyze process data. For example, Bergner, Shu, and von Davier (2014) used sequence similarity measures to explore data collected from the National Assessment for Educational Progress (NAEP) Technology and Engineering Literacy (TEL) Wells task. Similarly, He, Borgonovi, and Paccagnella (2019) used sequence mining on data from the Programme for the International Assessment of Adult Competencies (PIAAC) to study problem-solving behaviors. Still others have extended IRT models to explicitly incorporate information from process data. For example, the Markov IRT model was introduced by Shu, Bergner, Zhu, Hao, and von Davier (2017).

Timing information

One specific use of process data has been the analysis of response times. This is a natural extension of work done in cognitive psychology on reaction times, which has been used to make inferences about the kind of cognitive processing a person is engaged in (see De Boeck & Jeon, 2019, for a summary of such work). Applied to process data from testing, inferences can be made about test-taker behaviors like rapid guessing and supplement estimations of item difficulty and examinee ability (Lu et al., 2020). Sahin and Colvin (2020) built on this response time work by including other process data, in addition to response times, to determine the threshold for considering an examinee to be disengaged. Similarly, Lundgren and Eklöf (2021) explored how response time and other examinee actions could be combined to create an index of effort. Some of these actions were defined based on the specific item being analyzed, the "Traffic" task from PISA 2012. They found that the level of effort before giving up was related to overall performance on the exam.

Group differences

While work on response times has been ongoing for a few decades, more recent analyses of process has explored whether and how process data can be used to detect important group differences. For example, Guo, Zhang, Deane, and Bennett (2019) used data from keystroke logs to understand differences in the writing process between examinees with different SES, gender, and race. They classified sequences of keystroke actions into one of three states, including the duration within each state, and found that there were differences in time spent in states across some of the groups compared.

Eichmann et al. (2020) used process data from the PISA 2012 "Tickets" item to see whether that data could explain differences in performance based on an examinees' gender or migration status. They derived measures from examinees' action sequences based on the number of actions they took, and the number of exploratory actions. Similar to other studies looking at action sequences, the authors computed the distance between an ideal sequence and an examinees' sequence, and then used the difference in the number of steps between those sequences to determine how many exploration steps an examinee took. They found that both of these measures, interactions and exploration steps, were related to an examinees' gender but not their migration status.

Ercikan, Guo, and He (2020) propose an extension of DIF methods to include process data, allowing statistical testing of whether groups of examinees differed not only on their response but also their process. In one example the authors used an examinees' differential response time, conditioned either on the total score or on the total response time. The former model looked at differences in response time after matching on total scores, while the latter looked at those differences after matching on total response time. Comparing between ELL and

non-ELL students, they found that while items did not show "traditional" DIF, there was DIF based on response times. In a second example, the authors used data from the 2015 PISA science assessment to incorporate the number of actions an examinee took into the DIF model. Comparing cultural and language groups, this model did find evidence of differences in response processes.

Previous Analyses of PISA 2012 Traffic and Climate Control Scenarios

Many of the studies discussed in the previous section, exploring ways to analyze and use process data, have used data from the PISA 2012 Problem Solving assessment. This section provides a few additional examples, to highlight previous work with data from the two focal items in this study.

Traffic. Lundgren and Eklöf (2021), discussed earlier, used process data from this item to examine the relationship of examinee motivation, as determined by the process data, and performance on the item. Liu, Liu, and Li (2018) used data from the Traffic item to propose a multilevel mixture model, which incorporates the action sequence into estimation of a latent process ability. It specifies a process level and a student level, with processes nested within students, and assumes there are latent groups of students indicated by shared response patterns. The authors compare the relationship between the two different ability estimates and features like the number of clicks, resets, response time, and others. Results suggest that process-level ability is measuring something like efficiency: it is associated with fewer clicks and fewer resets of the map.

Climate Control. Like many studies of this item, Greiff, Wüstenberg, and Avvisati (2015) explored the extent to which examinees followed a specific strategy called vary one thing at a time, or VOTAT. The authors determined whether an examinee followed that strategy by

looking at a specific rubric score, seeing whether the examinee manipulated a single control at a time while leaving all others untouched. Then they compared the performance of students who did and did not follow that strategy, finding that using the strategy was strongly related to performance.

Pejic and Molcer (2019) used machine learning techniques to predict examinee performance based on a series of features meant to indicate the VOTAT strategy. They compared four different models to see which performed best in predicting success. The authors extended this work in Pejic and Molcer (2021) to include more machine learning models as well as two different sets of features. One set of features was based on timing information, while the second was intended to indicate use of the VOTAT strategy. The VOTAT features were a much better predictor of performance than the timing features, averaging about 90% correct classification compared to 70% for the timing.

Han, He, and von Davier (2019) used a random forest model to generate features from process data collected on this item, and to see which features were best at predicting performance. They created features based on groups of connected actions (e.g. groups of two actions, groups of three actions, etc), the use of the VOTAT strategy, and the use of specific actions like “apply” and “diagram” (explained further in the Methods section below). The authors found that specific single actions and the use of VOTAT were predictive of performance.

Chen, Li, Liu, and Ying (2019) used event history analysis model process data on the Climate Control item. Their research question is focused on prediction: based on the time a student has spent solving the item, how much more time they will need and will get the item correct. Success on the item is predicted based on a vector of time-varying features from the log and time spent solving the item so far. Parameter estimates from the model show whether feature

values make a right answer more or less likely, and whether they increase or decrease time to solve. Xu, Fang, Chen, Liu, and Ying (2018) proposed two latent class event history models to analyze data from the Climate Control item. Their method is primarily a latent class approach where class membership is determined by the counts of different events that students took when solving the item as well as total time. They find that classes which demonstrated strategies known to be efficient had higher percentages of members getting the item correct.

Eichmann et al. (2020) analyzed the full sequence of examinees' responses to the Climate Control item, to understand what aspects of the sequence were related to performance. They coded all actions as initial or repeated exploration, initial or repeated goal-directed behavior, or resetting the task. The authors looked for similarities among the sequences of those who got the item right or those who got the item wrong. This analysis showed that those who got the item wrong engaged in more exploratory behavior and may have applied the VOTAT strategy, but not to all of the variables, while those who got the item right did more checking of their solution.

Operational Psychometric Research

Lane, Raymond, Haladyna, and Downing (2016) identify twelve components of test development, from developing an overall plan to specifying the domain to be measured to preparing technical reports. Although the entire volume that article appears in is devoted to the test development process, the majority of the chapters are about item development, showing how important this component is to the test development process. As a result of this, much attention is paid to understanding how items work in operational research. Developing items includes not only choosing the item formats, but also collecting and analyzing evidence about the items. That evidence often comes from field testing the items to determine their measurement properties, typically focused on an item's difficulty, discrimination, relationship with other items, and DIF

(Haladyna, 2016). The primary data for this sort of analysis of items is item responses provided by participants in a field study.

Another important aspect of item development is collecting and analyzing evidence about the validity of score interpretation and use for an item. Especially in the case of items with novel formats (i.e. formats that have not been as thoroughly researched as multiple choice or constructed response have), validity evidence may be captured from thinkaloud protocols to ensure that items are accessible and that responses are evidence of the intended cognitive processes (Lane et al., 2016). Some novel item types include drag and drop items, where students respond to the item by dragging from sources to certain targets (Arslan et al., 2020, Jiang et al., 2021). An example of novel components with traditional item types would be a multiple-choice item that has some interactive aspect of its stimulus, like the interactive map or climate control device in the two PISA scenarios analyzed in this paper. These interactive components of items can be either optional or required. If optional, examinees may use them to aid in responding to the item, but the interactive component does not provide examinees information that is necessary for a correct response. If required, either the interactive component provides information that examinees need, or the interactive component is how examinees provide a response to the item. In a review of computer-based item formats, Sireci and Zenisky (2016) identify validity issues specific to these kinds of novel items, including construct-irrelevant variance from an examinees' lack of proficiency with computers and anxiety or disengagement due to unfamiliarity with computers.

Research Questions

Process data has primarily been used to explore how measurements of an examinees' process are related to their performance on items. This study takes a different direction, and

proposes to use process data to answer questions about properties of the items themselves. In this way, this study seeks to use process data to answer an operational question: how did students interact with the items being studied?

The two PISA problem-solving scenarios chosen for this study highlight important aspects of both process data analysis and operational research. In terms of process data analysis, the literature review demonstrates how much work has been done to understand and analyze process data from these two scenarios. However, that work has been almost exclusively focused on being able to model examinee problem-solving or make inferences about ability or strategy use. This study instead takes a fresh look at that same process data to see what can be understood about how the items functioned.

The operational research aspect of this study demonstrates how process data might help determine how, for a given assessment and context, these novel digital aspects of items are used by students, whether their use is related to student performance, and how data about use could lead to improvements of the items. These kinds of questions have typically been evaluated through thinkalouds. And while validity evidence from thinkalouds is valuable, it can be cumbersome to collect and analyze. If the validity claim to be evaluated is how examinees are interacting with the items, process data may be able to provide a large amount of relevant data. The broad research question is broken down into specific research questions for each of the scenarios, focusing on how students used interactive components and their responses to the items.

Traffic

1. Did students use the map for item 1?
2. Were there differences in scores based on map usage?

3. How did different operationalizations of "map usage" change the results?

Climate Control

1. How were item 1 scores related to item 2 scores?
2. Were there differences in the action sequences on item 2 for examinees who got item 1 wrong?
3. Do any of those sequences suggest that examinees were confused about how to use the interface?

Methods

Participants and Data

This study analyzes data from two different problem solving scenarios on the PISA 2012 administration. Two items are analyzed for each scenario, leading to a total of four items. This section of the PISA 2012 sought to assess general cognitive processes in problem solving, as opposed to problem solving within specific domains (OECD, 2014). This assessment also made use of interactive problems to assess these processes, by providing examinees information that they had to understand and use in order to answer some items correctly. There are three key aspects of problem-solving competence assessed in this portion of the exam: capacity to understand and resolve problems, being able to do so when a solution is not obvious, and a willingness to engage with these situations.

The sample analyzed in this study is drawn from examinees in the USA who completed both items in each scenario. The data was cleaned by removing examinees with invalid process data, such as no recorded actions, or those missing key information like a start time. After completing this, there were $N = 406$ examinees from the Traffic scenario, and $N = 381$

examinees for the Climate Control scenario. Further details are presented in a description of each of the two focal scenarios in the sections that follow, including further manipulations of the data.

Traffic

The Traffic scenario presented students with a map of a set of cities, the roads connecting them, and the travel time between each point on the roads. This was a clickable interface, where students could select and deselect road segments, and a total travel time was shown. Two items from this scenario will be analyzed. The first is a multiple-choice question asking students to find the shortest travel time between two locations. This question does not require students to use the interactive components of the map—a correct answer can be found just by summing the travel times between the appropriate segments. The second item requires students to use the interactive map to indicate the shortest route between two points.

Figure 1.
Interactive map for the Traffic item from PISA 2012

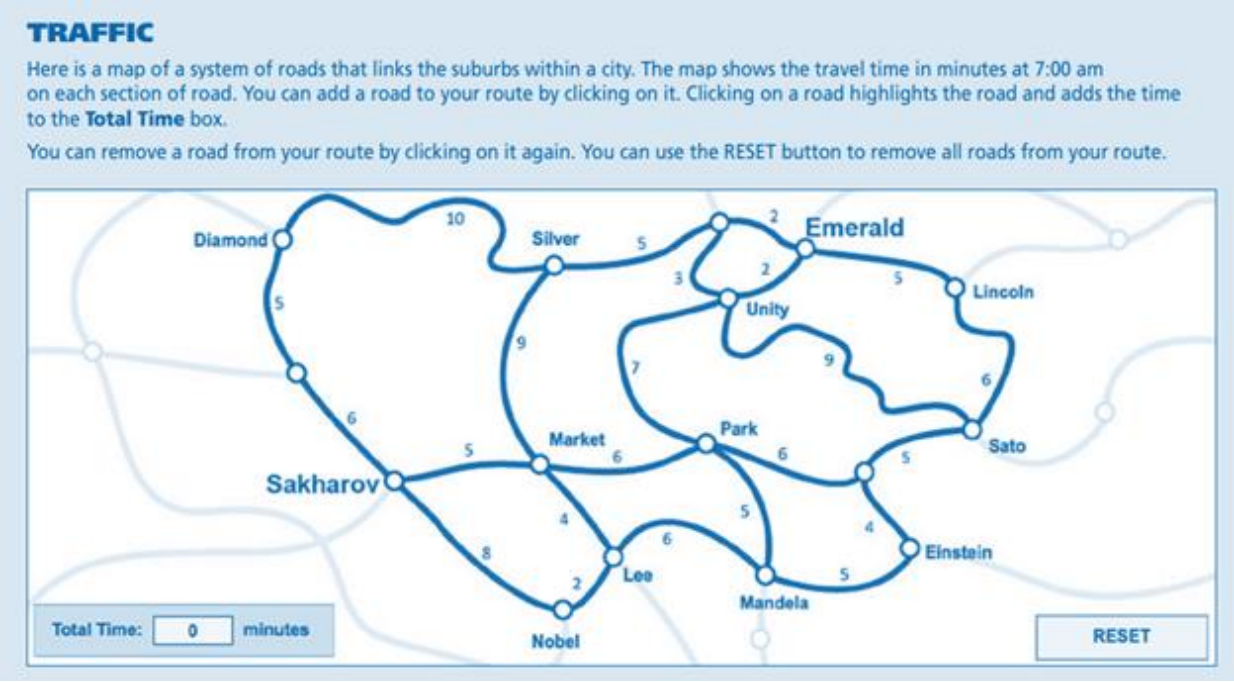
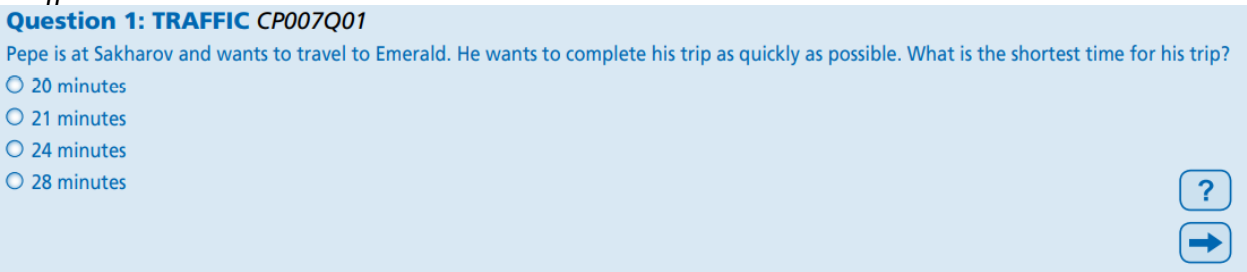


Figure 2.
Traffic Item 1



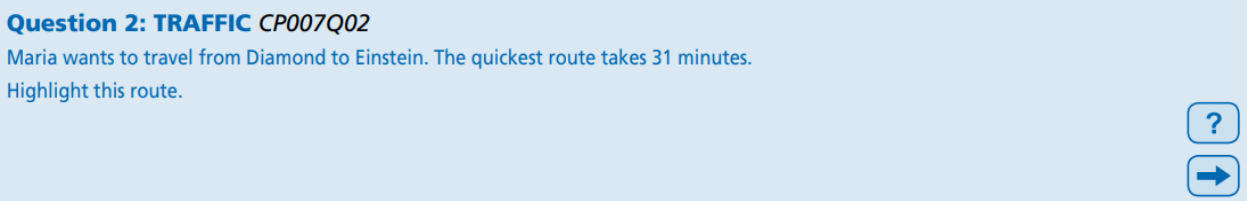
In order to answer questions about how students used the interactive map, "use" had to be operationalized. This was trivial for the second item, since examinees had to indicate their answer using the interactive map. For the first item, three different definitions were created. These are described in Table 1. These operationalizations are intended to capture substantive differences in how examinees used them map: by marking out the whole correct route, by marking out a partially correct route, by simply using the map at some point, or by not using the map at all. It is expected that these different ways of interacting with the map will be related to whether examinees responded correctly, with the highest percentage of correct response among those who marked the full correct route.

Table 1.
Description of different operationalizations of "map use" for Traffic item 1

Use variable	Operationalization
Use	True if any event was a map click
Correct start and end	True if any map state had the correct start and end routes selected
Full correct use	True only if any map state had the shortest route between the points highlighted.
Use count	Number of times map routes were selected or de-selected; the sum of counts is dichotomized into two levels at the median, so that each level represents 50% of the distribution of counts

For the second research question, a chi-square test is used to see whether there is a significant association between map use in item one and whether an examinee's item two response was correct. Chi-square tests are carried out for the different operationalizations presented in Table 1, to see how those operationalizations affect the result. Each test has both a Yates continuity correction and a Benjamini-Hochberg p-value adjustment applied, to control for small cell sizes and multiple comparisons, respectively.

Figure 3.
Traffic Item 2



Climate Control

The second scenario is called Climate Control. Students were presented with a climate control machine that had three unlabeled knobs, and an interface where they can adjust sliders and then "apply" to see how the temperature and humidity change. The first item asks students to experiment with the controls in order to determine what aspect of the climate (temperature or humidity) each control affects. The second item asked students to change the temperature and humidity in the room to specified values by using the climate control machine, and taking no more than four actions.

Analysis of this scenario focuses on the second item, with a grouping variable based on examinees' response to item one. In particular, action sequences on item two will be examined for examinees who got item one incorrect. The goal of this is to determine whether there were similarities in the approach of these students to item two, and whether any of those approaches

indicate that examinees were confused about the interface as opposed to the construct being measured.

Figure 4.
Interactive Stimulus for the Climate Control Item from PISA 2012

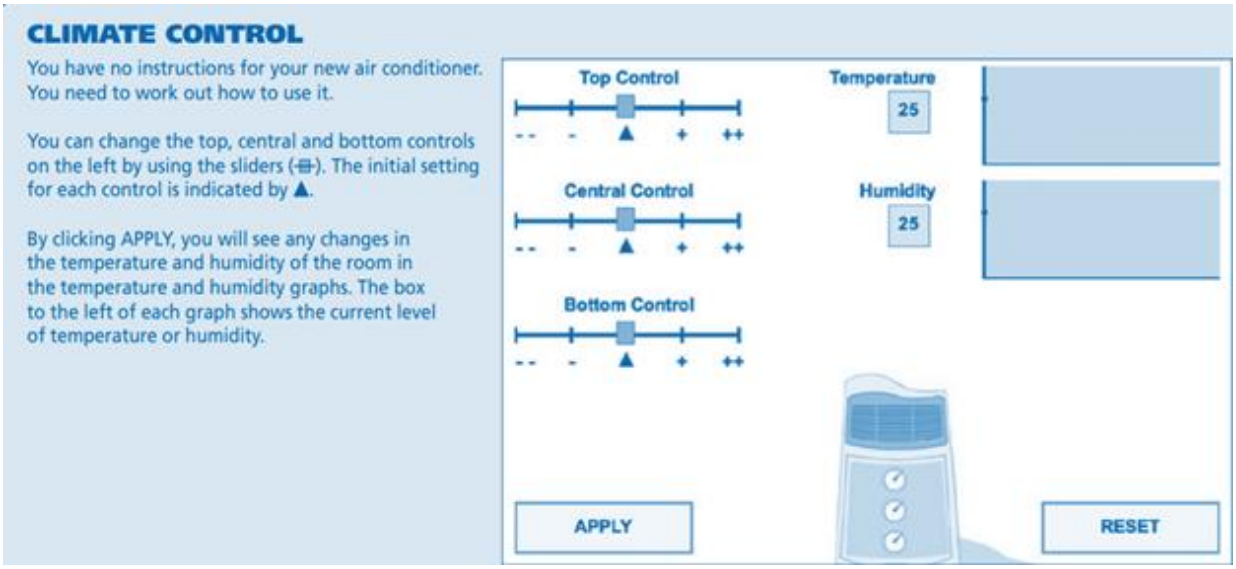
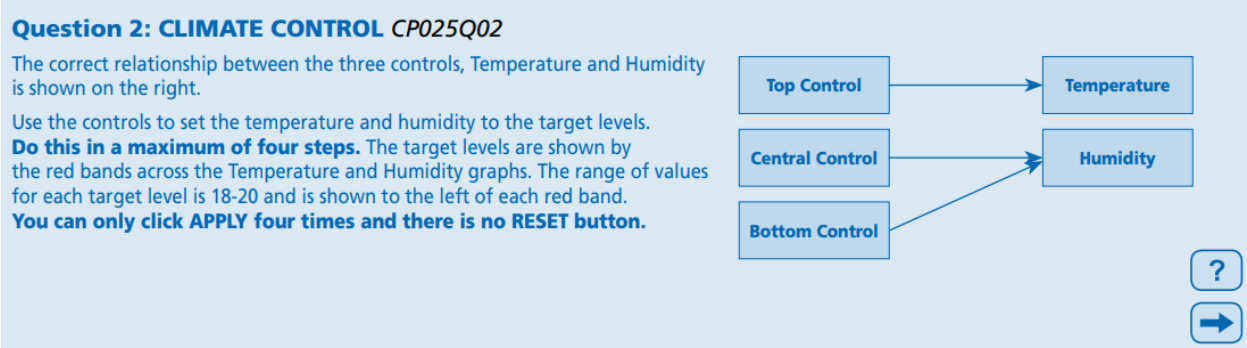


Figure 5.
Climate Control item 2



Raw data for this analysis was in a longitudinal format with multiple rows for each examinee. Because examinees were limited to only four "apply" actions for item two, the most informative data how they did or did not adjust the controls. This led to the creation of a variable for each control, indicating whether that control had been changed from the previous "apply" action. These were then summarized to a count of the number of "apply" actions for which the

setting on that particular control was the same. The intention here was to capture whether examinees were responding to the output in the interactive portion of the stimulus that indicated the current (and past) temperature and humidity. This was further summarized into the count of actions where all controls were the same (i.e. the controls had not been adjusted from the previous apply action), and a binary variable indicating whether the controls were the same for all actions that examinee took.

A correct answer for item 1, in which examinees had to manipulate the controls and determine how the air conditioner worked, suggests that an examinee was able to understand information from the interactive stimulus and react to it. Therefore, if an examinee got item 1 right but item 2 wrong, it can be assumed that it was largely not due to confusion about how to use or understand information from the interface. However, if an examinee got item 1 wrong, it may be the case that they were not able to get information they needed from the interactive stimulus, or were confused about how to use the controls. For examinees who got item one wrong, and did not change the controls at all for item two, it can be argued that confusion about how to use the interface may have been a factor; whereas, for examinees who got item 1 right but item 2 wrong, with the same pattern of control use, it may be that they simply were not systematic in their approach to completing item 2. Analysis will focus on distinguishing examinees who may have been confused about the use of the interactive component and examinees who simply did not act systematically.

Results

Results are presented separately for each scenario analyzed in this study, starting with the Traffic items.

Traffic

Descriptive statistics and correlations for both items and use variables in the Traffic scenario are presented in Table 2. These show that item 1 was answered correctly by the overwhelming majority of students, while slightly fewer answered item 2 correctly. It also shows that the different definitions of use, particularly the binary use compared to more specific definitions, yielded different percentages of examinees. The correlations will be discussed later.

Table 2.
Means, standard deviations, and correlations for Traffic scenario

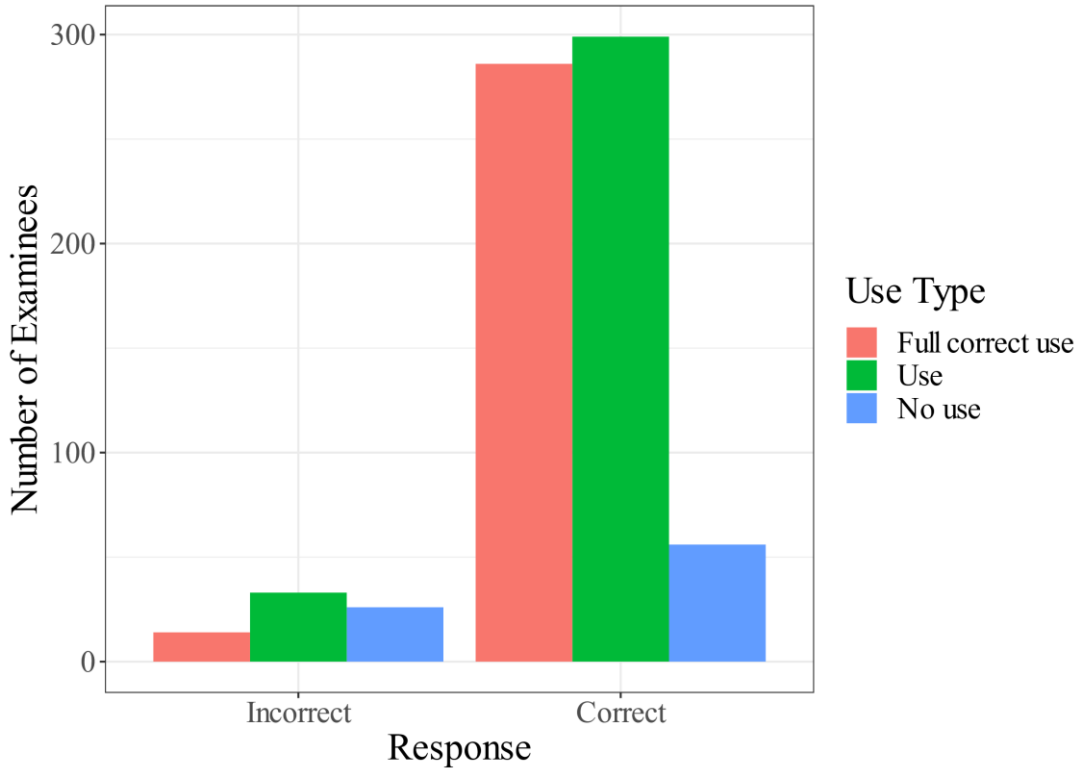
Variable	*M*	*SD*	1		2		3		4		5	
1. Item 1 correct	0.86	0.35										
2. Item 2 correct	0.76	0.43	0.21	**								
3. Use	0.81	0.39	0.27	**	0.05							
4. Full correct use	0.74	0.44	0.46	**	0.23	**	0.80	**				
5. Correct start/end	0.75	0.43	0.40	**	0.23	**	0.83	**	0.97	**		
6. Use count	0.51	0.50	0.13	*	0.13	*	0.48	**	0.50	**	0.54	**

Note: $N = 406$ examinees; Item 1 and 2 correct, Use, Full correct use, Correct start/end, and Use count are all binary variables so percentages are reported here (the variables are defined in Table 1). Spearman correlations are used, with a Benjamini-Hochberg correction for multiple comparisons. Significance of correlations is indicated by asterisk, with ** for $p < .001$, and * for $p < .05$.

To understand how students used the map on item 1, and whether this was related to their scores, Figure 6 shows the breakdown of different use types and the score examinees received. This shows that many examinees got the item correct while not using the map at all, and that nearly all of the examinees who used the map to mark the full correct route between the two location got the item correct. Also noteworthy is that nearly the same number of examinees used the map at all as used the map to mark the correct route, indicating that if examinees used the map for item 1, they generally were successful in marking the correct route. Among those who got the item wrong, a majority did use the map, but nearly the same number did not use the map at all. A small portion did mark the correct route, but still got the item wrong.

Figure 6.

Item responses and map usage for Traffic item 1.



Comparing examinees' item 1 map use and item 2 response was conducted through a Chi square test of independence for each definition of use, reported in Table 3. This table shows that there was not an association between the binary use variable and item 2 score, but a significant association was found between item 2 scores and other definitions of use. More specifically, examinees who marked the correct route on item 1, and those who had more than the median number of map actions on item 1, were more likely to get item 2 correct. Importantly, the minimum number of actions required to mark the correct route for item 1 is 4, while the median number of map actions was 11. This means that heavy map users in item 1 may have been exploring how the interface worked or trying out different possible solutions.

Table 3.
X^2 tests for examinees' item 1 map use and item 2 response

Variable	X^2	df	*p*
Use	0.77	1	.379
Full correct use	20.86	1	< .001
Use Count	5.88	1	.023

Note: All *p*-values were adjusted for multiple comparisons using a Benjamini-Hochberg correction.

Climate Control

Descriptive statistics for the items in the Climate Control scenario are presented in Table 4. This shows that the second item, in which examinees had a limited number of actions to change the temperature and humidity, was much more difficult than the first time. A significant positive correlation was observed between the and examinees' score for each item, and a significant negative correlation between item 1 score and whether an examinee kept all the controls the same for item 2. This means that examinees who kept controls the same for item 2 were less likely to have gotten item 1 correct. This is explored further with a Chi-square test.

Table 4.
Means, standard deviations, and correlations for Climate Control variables

Variable	*M*	*SD*	1		2
1. Item 1 correct	0.64	0.48			
2. Item 2 correct	0.17	0.38	0.22	**	
3. All controls same	0.18	0.39	-0.20	**	-0.07

Note: N = 381 examinees. All variables are binary so means are percentages; Spearman correlations are used, with a Benjamini-Hochberg correction for multiple comparisons. Significance of correlations is indicated by asterisk, with ** for p < .001, and * for p < .05.

A Chi-square test of independence was conducted between an examinees' item 1 score and whether they kept all controls the same. This test found a significant association between these variables, $X^2(1) = 13.50$, $p < .001$. In particular, a higher percentage of examinees who got item 1 wrong kept all controls the same for item 2, compared to those who got item 1 right. This suggests that they may not have understood how to use the interactive component. A Chi-square

test was not able to be conducted on the association between item 2 score and the controls variable due to one of the cells being exceptionally small. The contingency table for this association is included in Table 5, to provide some sense of these variables.

Table 5.
Contingency table for Climate Control item 2

	Controls	
	Same	Not Same
Item 2 correct	8 (2%)	57 (15%)
Item 2 incorrect	62 (16%)	254 (66%)

Note: N = 381 examinees; "controls same" means that controls were not adjusted for any action in item 2, while "not same" means that controls were adjusted for at least one "apply" action on item 2.

Table 5 shows that very few students who got item 2 correct kept all of the controls the same, while the vast majority changed the controls at least once. For those who got the item incorrect, the vast majority adjusted the controls, while about 16% of examinees did not adjust the controls at all. Although a test of association was not possible, looking at the percentage of examinees in each cell of Table 5 suggests that more examinees who adjusted the controls got the item right than did not.

Discussion

Results for each item will be discussed separately, followed by the implications of this study for the use of process data in evaluating how examinees interact with novel item types.

For the Traffic items, a significant association was found between how examinees' used the map on item one and their response on item 2. This was true for two different operationalizations of "map use" on item 1. The first was if examinees used the map to highlight the correct shortest route between the two given points. Most examinees who used the map at all for item 1 used the map in this way. Because using the interactive component of the map was not required for item 1, this suggests that these examinees were able to understand how to interact

with the map to highlight their desired route. Whether they needed to explore how the map worked or began the item understanding how it worked, these examinees were able to use the map as specified for item 2.

The other operationalization of use what whether an examinee took greater or fewer than the median number of map actions for item 1. The median number of actions was 11 for this item—importantly, the minimum number of actions required to highlight the correct route for item 1 was only 4. This means that any examinee who took more than the median map actions did at least some exploration of the map. The exploration have been intentional, to try out different routes, or it may have been unintentional, for example by accidentally clicking the wrong segment. Regardless of the intention, examinees who took more map actions either or became familiar with how the map worked, since these examinees were more likely to answer correctly for item 2.

For the Climate Control items, a significant association was found between an examinees' score on item 1 and whether they kept all of the controls the same for item 2. What makes this set of items interesting is that, as part of the stimulus for item 2, the answer to item 1 is given. This means that, in theory, examinees who got item 1 wrong could still get item 2 correct. Yet, only about 6% of examinees who got item 1 wrong went on to get item 2 correct. Further, analysis of item 2 actions suggests that examinees who got item 1 wrong either experienced some confusion with the interactive components, or were disengaged and simply clicked through all four of their allotted actions.

While the association between using the controls and examinees' item 2 score was not able to be tested statistically, the frequency table strongly suggests that examinees who left the controls the same were less likely to answer correctly. This provides further support to the

inference that, for at least some of these examinees, confusion about how the interactive interface worked, or difficulty in understanding the dynamic information presented by the interactive interface, may have contributed to their answering item 2 incorrectly.

Overall, this study illustrates a few important takeaways for using process data in operational research. The first is a basic one, but crucially important: how variables are operationalized has substantial implications for what analysis will show. On the Traffic item, different operationalizations of "use" led to substantially different statistical results, as well as different inferences about examinee behavior. This result highlights the importance of collaboration between process data analysts and subject matter experts. In using process for operational purposes, this kind of collaboration can ensure that inferences from process data are valid with respect to the target construct.

The second important point is that process data can provide information about how examinees are interacting with items. In this study, process data was used to make inferences about whether examinees were comfortable using interactive components of the items. This was a conclusion not about examinees' ability or construct knowledge, but instead a conclusion about their interactions with the components of the item, based on analysis of those interactions. The results showed that there may have been some confusion with using the interactive components, which could be addressed by revising items.

Finally, the study helps underscore the importance of carefully designed process data collection. The process data collected on these PISA 2012 scenarios has been an extremely valuable source of data for the measurement field, as shown by the large number of analyses that have been conducted. That data was able to be adapted and summarized to answer the present research questions. However, it is easy to imagine more informative data could have been

gathered: for example, the Climate Control items could have captured each time the control was adjusted, rather than just the state of the controls when an “apply” action was taken. This could have been used to detect examinees who were uncertain or random in their adjustment of the controls, providing additional evidence of how examinees used the interactive components of the item. Related, the process data collection could be designed with validity claims in mind. That is, if a testing program is exploring a novel item format, specific process data could be collected to evaluate validity claims about that format. If data collection and analysis plans are aligned, process data can fulfill its potential as a source of fine-grained data about examinees.

This study has a few limitations. One of the most important is what was just discussed, that the process data analyzed was not captured to answer these questions in particular. Inferences made based on this data are still justifiable considering the results of analysis, but would have been stronger had the process data collection taken these questions into account. Another limitation is on the Climate Control item 2, which saw extremely disproportionate cells in the contingency table. This meant that a Chi-square test could not be performed, and so only a descriptive result was possible. A final limitation is that the sample size, though adequate for the analyses conducted, was somewhat small.

However, this study does intend to produce generalizable inferences about how examinees used interactive components of these items. The purpose is to demonstrate an approach to using process data to make inferences about items and how examinees interacted with them. By looking at how item scores were associated with process data detailing an examinee’s interaction with the items, this study has shown that some incorrect responses may have been due to issues with the interactive component. Future work could enhance support for this conclusion by including estimates of examinee ability, to see whether low ability or

confusion about the interactive component is a more plausible explanation of the item response. The results of this study, and that future work, could then be used to guide the revision of those item components. While obviously not practicable for the PISA 2012 administration, this study demonstrates some of the value that process data may have for operational research.

References

Arslan, B., Jiang, Y., Keehner, M., Gong, T., Katz, I. R., & Yan, F. (2020). The Effect of Drag-and-Drop Item Features on Test-Taker Performance and Response Strategies. *Educational Measurement: Issues and Practice*, *39*(2), 96–106. https://doi.org/10.1111/emip.12326

Bergner, Y., & von Davier, A. A. (2018). Process Data in NAEP: Past, Present, and Future. *Journal of Educational and Behavioral Statistics*, *44*(6), 706–732. https://doi.org/10.3102/1076998618784700

Chen, Y., Li, X., Liu, J., & Ying, Z. (2019). Statistical Analysis of Complex Problem-Solving Process Data: An Event History Analysis Approach. *Frontiers in Psychology*, *10*. https://www.frontiersin.org/article/10.3389/fpsyg.2019.00486

De Boeck, P., & Jeon, M. (2019). An Overview of Models for Response Times and Processes in Cognitive Tests. *Frontiers in Psychology*, *10*. https://doi.org/10.3389/fpsyg.2019.00102

Eichmann, B., Goldhammer, F., Greiff, S., Brandhuber, L., & Naumann, J. (2020). Using process data to explain group differences in complex problem solving. *Journal of Educational Psychology*, *112*(8), 1546–1562. https://doi.org/10.1037/edu0000446

Eichmann, B., Greiff, S., Naumann, J., Brandhuber, L., & Goldhammer, F. (2020). Exploring behavioural patterns during complex problem-solving. *Journal of Computer Assisted Learning*, *36*(6), 933–956. https://doi.org/10.1111/jcal.12451

Ercikan, K., Guo, H., & He, Q. (2020). Use of Response Process Data to Inform Group Comparisons and Fairness Research. *Educational Assessment*, *25*(3), 179–197. https://doi.org/10.1080/10627197.2020.1804353

Goldhammer, F., Hahnel, C., Kroehne, U., & Zehner, F. (2021). From byproduct to design factor: On validating the interpretation of process indicators based on log data. *Large-Scale Assessments in Education*, *9*. https://doi.org/10.1186/s40536-021-00113-5

Greiff, S., Wüstenberg, S., & Avvisati, F. (2015). Computer-generated log-file analyses as a window into students' minds? A showcase study based on the PISA 2012 assessment of problem solving. *Computers & Education*, *91*, 92–105. https://doi.org/10.1016/j.compedu.2015.10.018

Guo, H., Zhang, M., Deane, P., & Bennett, R. E. (2019). Writing Process Differences in Subgroups Reflected in Keystroke Logs. *Journal of Educational and Behavioral Statistics*, *44*(5), 571–596. https://doi.org/10.3102/1076998619856590

Haladyna, T. M. (2015). Item analysis for selected-response test items. In S. Lane, M. R. Raymond, & T. M. Haladyna (Eds.), *Handbook of test development* (pp. 408-425). Routledge/Taylor & Francis Group.

Han, Z., He, Q., & von Davier, M. (2019). Predictive Feature Generation and Selection Using Process Data From PISA Interactive Problem-Solving Items: An Application of Random Forests. *Frontiers in Psychology*, *10*. https://doi.org/10.3389/fpsyg.2019.02461

Hao, J., Shu, Z., & von Davier, A. (2015). *Analyzing Process Data from Game/Scenario- Based Tasks: An Edit Distance Approach*. *7*(1), 33-50.

Hao, J., Smith, L., Mislevy, R., von Davier, A., & Bauer, M. (2016). Taming Log Files From Game/Simulation-Based Assessments: Data Models and Data Analysis Tools: Taming Log Files From Game/Simulation-Based Assessments. *ETS Research Report Series*, *2016*, 1–17. https://doi.org/10.1002/ets2.12096

Jiang, Y., Gong, T., Saldivia, L. E., Cayton-Hodges, G., & Agard, C. (2021). Using process data to understand problem-solving strategies and processes for drag-and-drop items in a large-scale mathematics assessment. *Large-Scale Assessments in Education*, *9*. https://doi.org/10.1186/s40536-021-00095-4

Lane, S., Raymond, M. R., Haladyna, T. M., & Downing, S. M. (2015). Test development process. In S. Lane, M. R. Raymond, & T. M. Haladyna (Eds.), *Handbook of test development* (pp. 19–34). Routledge/Taylor & Francis Group.

Liu, H., Liu, Y., & Li, M. (2018). Analysis of Process Data of PISA 2012 Computer-Based Problem Solving: Application of the Modified Multilevel Mixture IRT Model. *Frontiers in Psychology*, *9*. https://doi.org/10.3389/fpsyg.2018.01372

Lu, J., Wang, C., Zhang, J., & Tao, J. (2020). A mixture model for responses and response times with a higher-order ability structure to detect rapid guessing behaviour. *British Journal of Mathematical and Statistical Psychology*, *73*(2), 261–288. https://doi.org/10.1111/bmsp.12175

Lundgren, E., & Eklöf, H. (2020). Within-item response processes as indicators of test-taking effort and motivation. *Educational Research and Evaluation*, *26*(5–6), 275–301. https://doi.org/10.1080/13803611.2021.1963940

OECD (2014). Assessing problem-solving skills in PISA 2012. In *PISA 2012 Results: Creative Problem Solving (Volume V): Students' Skills in Tackling Real-Life Problems*, OECD Publishing, Paris. https://doi.org/10.1787/9789264208070-6-en.

A. Pejić & P. S. Molcer. (2019). Predicting the Outcome of a PISA Problem Solving Task Using Strategic Behavior Data. *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, 313–318. https://doi.org/10.1109/CogInfoCom47531.2019.9089942

Provasnik, S. (2021). Process data, the new frontier for assessment development: Rich new soil or a quixotic quest? *Large-Scale Assessments in Education*, *9*. https://doi.org/10.1186/s40536-020-00092-z

Sahin, F., & Colvin, K. F. (2020). Enhancing response time thresholds with response behaviors for detecting disengaged examinees. *Large-Scale Assessments in Education*, *8*. https://doi.org/10.1186/s40536-020-00082-1

Sireci, S. G., & Zenisky, A. L. (2016). Computerized innovative item formats: Achievement and credentialing. In S. Lane, M. R. Raymond, & T. M. Haladyna (Eds.), *Handbook of test development* (pp. 313–334). Routledge/Taylor & Francis Group.

Stadler, M., Fischer, F., & Greiff, S. (2019). Taking a Closer Look: An Exploratory Analysis of Successful and Unsuccessful Strategy Use in Complex Problems. *Frontiers in Psychology*, *10*. https://doi.org/10.3389/fpsyg.2019.00777

Wüstenberg, S., Greiff, S., Vainikainen, M.-P., & Murphy, K. (2016). Individual differences in students' complex problem solving skills: How they evolve and what they imply. *Journal of Educational Psychology*, *108*(7), 1028–1044. https://doi.org/10.1037/edu0000101

Xu, H., Fang, G., Chen, Y., Liu, J., & Ying, Z. (2018). Latent Class Analysis of Recurrent Events in Problem-Solving Items. *Applied Psychological Measurement*, *42*(6), 478–498. https://doi.org/10.1177/0146621617748325

Zhu, M., Shu, Z., & von Davier, A. A. (2016). Using Networks to Visualize and Analyze Process Data for Educational Assessment: Network Analysis for Process Data. *Journal of Educational Measurement*, *53*(2), 190–211. https://doi.org/10.1111/jedm.12107

Conclusion

The three papers proposed for this dissertation are initial attempts at answering the question of how, and for what purposes, to use process data. Each provides an example of how process data can be used to explore something other than an examinee's ability. Woven throughout these studies is a concern about the validity of process data uses. Just like test scores, evidence should be gathered and claims evaluated about the interpretations and uses of process data.

In that vein, each of the papers in this dissertation can be thought of as making a claim about a valid use of process data. The first and third papers make similar claims, which is that process data can be used to understand how items are functioning. While the first paper uses thinkaloud data as process data, the third paper suggests that in some narrow circumstances, process data may be able to be used in lieu of thinkaloud data. For both, evidence supporting that claim comes from the descriptive and explanatory power that process data has. The second paper makes a different claim, that process data can be used to begin building theory. This, too, may only be true in narrow circumstances: in this case much of the validity of this use is based on the decades of work to understand process data from natural language writing.

The first paper treats thinkaloud data as a kind of process data, specifically as evidence of students' response processes. By analyzing the sequence of actions students took in responding to the items, the precise influence of modifying item features on students' response processes can be understood. Results showed that response processes were indeed influenced by the item features, an important finding for future item development and test assembly. This provides valuable information about non-construct related item features for this domain, but also

illustrates a more widely-applicable method of analyzing thinkaloud data, and more broadly for using process data to understand item performance.

The second paper demonstrates how process data can be used as a starting point for developing an account of performance in a target domain. While theory about program writing as a psychological phenomenon is limited, a potentially similar phenomenon, natural language writing, has robust theory. Drawing on that theory allowed a preliminary description of the structure of program writing as a cognitive process, and the results showed that this theory may yet describe important aspects of program writing. This suggests a promising direction for future work both to understand and describe program writing more fully, and to develop more helpful diagnostic measures of program writing.

The final paper describes an approach to thinking about process data collection and analysis that centers results that are likely to be relevant to operational research. By demonstrating how process data can be used to evaluate how examinees are interacting with novel item formats, this paper shows that process data can be used in operational research. Results showed that some examinees may have struggled with the interactive aspects of the items, and also highlighted the important of carefully operationalizing concepts from process data. This paper shows how existing process data collection could be used to support assessment operations, and how future process data collection could be designed to capture even richer data about how students interact with test items.

Taken together, these papers exemplify three specific uses of process data: but more importantly, they each demonstrate how researchers, assessment designers, and test developers can think through how to use process data in their own contexts. Secondarily, the papers address

considerations about what process data should be collected, how it should be analyzed, and how the results of that analysis can be interpreted and used.

Process data is not new, and neither are the methods to analyze it. This paper does not introduce any new analytical methods, but, taken together, the work described here suggests a new approach to thinking about and applying process data in psychometrics. The key parts of that approach are: constant, iterative questioning and evaluation of validity arguments; designing assessment programs with process data in mind, rather than in an ad hoc fashion; and collaborations between subject matter experts, psychometricians, and process data experts.

Future research can flesh out best practices for each of these three parts. For example, while validity arguments must be customized to the intended interpretations and uses, there may be common elements to validity arguments using process data. One of those elements might be a 2-step inference: from an interpretation of the log data as indicative of a process, and then from the process data to an interpretation and use. Similarly, when designing assessment with process data in mind, there is likely not a single solution to fit all contexts. But here too, there may be common elements or procedures, like framing process data collection and use within an evidence-centered design process. Finally, future research should explore how to facilitate successful collaboration between subject matter experts, psychometricians, and process data experts; each brings unique and sometimes overlapping expertise, so considering how best to use those skills in designing, piloting, and implementing an assessment is important. These research efforts could contribute to a framework that could be followed by those working with assessment process data, which guides the use and interpretation of process data through a series of questions and procedures that researchers engage with. Until such research takes place,

researchers and practitioners should attend closely to the validity arguments linking log data, process data, interpretations, and uses.

This approach is intended to help process data live up to its promise, which is being a valid source of insight about what students actually do when they are responding to items. That promise can only be fulfilled if process data collection and analysis is centered in psychometric work--as a field, we have to move away from applying novel models to the same data, and instead to setting out a vision and program of research that considers the uses of process data at all stages of the assessment development and deployment cycle.

Much work remains to be done for process data to fulfill that promise. This dissertation addresses some of the gaps in the literature, particularly around using process data for item development, operations, describing the target construct, and how to think about validity in process data analysis. Future work needs to add to this toolkit of validity claims that process data might be used to investigate. In addition, more work needs to be done to extend the ideas of argument-based validity to process data analysis, by considering the interpretations and uses of process data. Exciting work can be done, too, to develop and specify domain performance in terms of what the process data will show—folding process data collection and analysis into test and item development processes. Future work on process data should also explore whether and how sharing process data with students and teachers can be beneficial. For example, watching a replay of a student writing a program may reveal an important area of confusion or misconception, which a teacher can then address. Or, because process data collection and analysis can be automated, it may lead to the deployment of tools to let students diagnose and iterate on their own program writing, potentially helping them learn and supporting their self-efficacy.

Process data truly does have a vast promise. As a field, psychometricians should be asking ourselves what questions we can actually use process data to answer--for whom are those questions relevant, and how those results can guide additional work. This can help process data live up that promise, by ensuring that it is used in ways that are valid, relevant, and useful.

Appendix A: Full Qualitative Codebook for Thinkaloud Transcripts

Category	Code	Explanation	Example
Reading Semantics (program behavior)	Unistructural reading	Reading of syntax verbatim. Natural language analogy: reading. "Simple meaning, focusing on one issue in a complex case"	"Int i = 0, i less than j, i plus plus"
	Multistructural reading	More abstract reading than unistructural, combines functionality of multiple lines together. Natural language analogy: paraphrase of multiple sentences. "Shopping list; (disorganized) collection of items"	
	tracing values	Finding the value of variable(s) after a portion of code has run	
Reading Templates	Relational reading	More abstracted from the code than multistructural reading. Reading of multiple lines of code as having a unified functionality. Natural language analogy: a summary or gist of a passage. "understanding"	
	recognizing template	Naming a template within provided code, or talking about multiple lines of code in relation to an objective (even if a template is not explicitly named)	"This looks like a ____"
	applying template	Using a template to understand code, or using an understanding of multiple lines of code to understand a proposed solution.. For example, recalling that code for a certain objective typically includes certain components; or reasoning about what individual lines of code do based on a known objective of the code.	"Because this is a ____ it should ____"
Writing Syntax	typing syntax	Participant is typing syntax (writing syntax code from segmentation) **Will be handled based on the log data, and encoded in the transcript**	**NOTE:** can copy/paste from teh segment code, usually
	revising syntax	Revision behavior focused on the syntax level. Natural language analogy: "checking grammar", making sure sentences have correct punctuation. Distinct from revising semantics in that the revision activity is NOT focused on what the code does.	

Writing Semantics	revising semantics	Participant is revising the function of their code (as opposed to the syntax). Natural language analogy: revising how an argument is made to ensure that the writer's point comes across. Key distinction from "revising syntax" is that revisions are focused on the functionality of the code.	
	Planning: local	Participant is figuring out how to meet a constraint/requirement for a single line. Natural language analogy: considering how to phrase as single sentence based on an objective (e.g. if writing from an outline). [brainstorming. If considering viability of articulated solution, use code EVALUATE POTENTIAL SOLUTION]	
	Planning: global	Participant is figuring out how to meet a constraint/requirement for a set of lines (possibly the entire program). Natural language analogy: considering how to form the overall argument of a paragraph or essay. Not focused on individual sentences, but rather on the argument in a larger chunk. [brainstorming. If considering viability of articulated solution, use code EVALUATE POTENTIAL SOLUTION]	
Problem Solving	Interpret task requirements	Stating task requirements in own words; indicating understanding of the task environment. Focused on what a question is asking ONLY, not on requirements of reading/writing code	Interpret task requirements
	Interpret program specification	Stating program specification in own words, indicating understanding of the requirements of the program to be created/interpreted	Interpret program specification
	Answer question	Participant is directly answering the question	Answer question
	Read answers	Reading answers in a MC item	
	Eliminate answers	Participant is eliminating multiple choice options.	
	Search for solutions	"Seek solutions that will satisfactorily solve the problem"	

Problem Solving	Evaluate a potential solution	"Evaluate how well this solution will address the problem. This includes actions like...mental algorithm simulations, or other techniques of sketching or prototyping a solution." Use with caution, only if the potential solution can be clearly identified, and if talk is about whether the proposed code will work. If brainstorming a solution, or description of how they will go about the work, use code PLANNING. Make a note in "notes" if you feel the segment could be coded as "considering alternatives".	
	Implement a solution	"Translate solution into source code", or use solution to find answer.	
	Evaluate implemented solution	"Evaluating how well their current implementation solves the problem."	
Monitoring	Defining/applying a strategy	Participant articulates a clear approach to answering the item; directs attention/effort on specific task/subtasks	E.g. goal-oriented statements, "I need to find the value…" "I'll come back and fix this syntax later"
	Noticing mistake	Participant sees an error in thinking, previously unobserved detail, a flaw in a proposed or implemented solution	
	Noticing confusion	Seeing boundaries of participant's own knowledge; expressing uncertainty about own knowledge or approach to problem	
	Checking answer	Some kind of review of answer for a non-writing question	

	Checking syntax	Participant reviews syntax in the answer. Natural language analogy: re-reading to make sure sentences are punctuated and words are capitalized.	E.g. “I want to make sure I have all my curly braces”
	Checking semantics	Participant reviews whether code performs the intended function; natural language analogy: re-reading to make sure your argument comes across clearly.	
	Checking general	Some kind of review seems to happen, but not clear whether syntax or semantics	

Appendix B: Programming Assignments

Assignment 1

Assn 6

Blueprint

Published

Introduction

You will complete two programs to give you experience with:

- Software Development Process
- Functions
- Import
- Strings
- Objects
- Formatting

Task 1

Save program in a file called task1.py and the plan in a file called plan1.txt. Calculate the area of a regular polygon. A regular polygon is one in which all sides are the same length and the angles all have the same degree. The formula for the area of a regular polygon is

$$Area = \frac{n \times s^2}{4 \times \tan\left(\frac{\pi}{n}\right)}$$

where *n* is the number of sides and *s* is the length of each side. Prompt the user for the length and number of sides and display the result, with an appropriate message, to the user.

You must use the math modules for pi, tan(), and pow(). You must use different techniques for importing in order to use items from the module exactly as follows (with the appropriate parameters) in your code:

```
tan()
math.pow()
pi
```

Output should be rounded to 5 decimal places using the round() function.

Here is a site you can use to create test cases:
https://keisan.casio.com/exec/system/1223282135
(https://keisan.casio.com/exec/system/1223282135)

Task 2

Save your program in a file called task2.py and the plan in a file called plan2.txt. Your program should look like the following.

```
Enter employee's name: Chad Mano
Enter number of hours worked in a week: 40
Enter hourly pay rate: 12.75
Enter federal tax withholding rate (ex. 0.12): 0.11
Enter state tax withholding rate (ex. 0.06): 0.07

       CHAD MANO PAY INFORMATION

                  Pay
              Hours Worked:          40
                  Pay Rate: $     12.75
                 Gross Pay: $    510.00

              Deductions
Federal Withholding (11.0%): $     56.10
   State Withholding (7.0%): $     35.70
           Total Deduction: $     91.80

                   Net Pay: $    418.20
```

Note the following:

1. The first 5 lines are asking for input from the user.
2. Notice the form tax rates are entered compared to how they're displayed.
3. Notice the alignment/formatting of the output
4. Headings of sections are centered and spaces between sections
5. Main output header is all capitalized
6. Output numbers are right aligned with decimals aligned
7. Colons and $ signs are aligned on output
8. No arithmetic calculations inside the print string (use variables)
9. A single print statement is used for output
10. Make sure to look at the rubric!

Software Development Lifecycle Plan

Follow the software development lifecycle in your assignment. Put the steps as comments at the top of your file below your name and other header comment information. This should be simple. Similar to what we have done in class. Submit a report with:

1. Requirements specification
2. System analysis
3. System design
4. Testing
    1. Write at least 2 test cases you will run to assess your program
    2. Each test case should have
        1. Input
        2. Expected Output

    3. Report if you ran into any problems and what you did to fix them. Write "Passed" if all tests passed on your first try.

Assignment 2

Assn 7

Blueprint

Published

Note: We will cover a topic on Monday 2/10 that you will need to finish Task 2.

Introduction

You will complete two programs to give you experience with:

- Selections
- Comparison operators
- Conditional statements
- Random numbers

Task 1 - The Force is Strong

Save your program as task1.py and your software development plan as plan1.txt. You are going to program a number guessing game. The computer will generate a random number from 1-10 (inclusive). The user will guess a number. You will print different messages depending on how close the guess was.

Example - Assuming the computer's number is 7

```
Welcome to the Guessing Game
The Computer has picked a number from 1 - 10. Try to match it.
What number do you choose (1-10): 4
You picked 4, and actual number was 7.
Youngling, your time will come.
```

The last messages are as follows:

- Exact match: Honored to play with you, Master.
- Off by 1: You are a worthy opponent, Knight.
- Off by 2: You have much to learn, Padawan.
- Off by 3: Youngling, your time will come.
- Off by 3+:Keep working hard in the Service Corps.

Code Requirements

- You may have only one print statement after the user makes their pick
- You may assume the user will give proper input
- You may not use nested if-statements or multiple single if-statements.

Task 2 - Social Situation Analysis

Save your program as task2.py and your software development plan as plan2.txt. You will write a program that will perform an analysis on a social situation. A user will input their name, location, and radius of their personal space. A second user will do the same. You will analyze this information and determine the results of two tests.

Person Test: This test determines if an individual has someone in their own personal space. One of the following outputs is possible:

- Person One is in Person Two's personal space
- Person Two is in Person One's personal space
- Person One and Person Two are in each other's personal space
- Neither Person One nor Person Two is in the other's personal space

Space Test: This test determines the relation of the personal spaces of each person. One the following outputs is possible:

- Person One and Person Two's personal spaces overlap
- Person One and Person Two's personal spaces do not overlap
- Person One's personal space is entirely inside Person Two's personal space
- Person Two's personal space is entirely inside Person One's personal space

NOTE: The messages you print out should look very similar to these. That means the user's name should actually be printed where *Person One* or *Person Two* is shown above.

Example

```
Welcome to the Social Situation Analyzer System
Person One
    Enter your name: Alice
    Enter your position (x, y): 50, -70
    Enter your personal space radius: 30

Person Two
    Enter your name: Bob
    Enter your position (x, y): 30, -50
    Enter your personal space radius: 20

Social Situation Analysis Results
    Person Test: Bob is in Alice's personal space
    Space Test: Alice and Bob's personal spaces overlap
```

Code Requirements

- You may have only one print statement after the second user enters their info
- You may assume the user will give proper input

- You may assume that the people cannot stand in the exact same location (distance between them cannot be zero)
- You must use a logical operator to determine if the users are both in each other's space
- Read the Rubric for all grading details

Illustrations

Personal Space Illustrations - Click to view some of the possible configurations of people and their personal spaces. It does not include every possible case. You can use these to help you think through how to solve the problem.

Helpers

Remember that you can find solutions to the even programming exercises online. Check Canvas for a link. These are suggestions for you to do. They are not part of the assignment, and you do not have to turn them in.

Exercises: 4.4, 4.16, 4.28

Assignment 3

Assn 8

Blueprint

Published

Introduction

You will complete two programs to give you experience with:

- While loop
- For loop
- Range()
- Break
- Continue

Task 1 - Fluky Numbers

Save your program in a file called task1.py and your software development plan in a file called plan1.txt. You will write a program that will find Fluky Numbers. A Fluky Number is a number that is equal to the sum of a set of random numbers, where the random numbers are the first random numbers generated after seeding a random number generator with each factor of the number, not including itself. That's quite a mouthful, so here it is again in simpler terms. All of the following requirements must be met for a number to be a Fluky Number.

1. A Fluky Number is a number, called The Number, that is equal to the sum of a set of randomly generated numbers.
2. Random numbers are generated with a random number function. We will use randint() from the random module.
3. The randomly generated numbers are in the range of 1 to the The Number, inclusive. (Ex. If The Number is 10, the random number can be 1 through 10, including 1 and 10)
4. Random numbers are the first number generated after setting a specific seed value (Use the random.seed() function).
5. The seed values are all of the factors of The Number, not including itself. (Ex. The factors of the number 10 that will be used for the seed are 1, 2, and 5)

Example

The number 8 is a Fluky Number. It's factors, not including itself, are 1, 2, and 4. The first random numbers generated between 1 and 8 after using the factors as a seed are 3, 1, and 4, respectively. The sum of those numbers is equal to 8. Thus, 8 meets the definition of a Fluky number. Here's

some code that illustrates this: seeddemo.py
(https://usu.instructure.com/courses/565650/files/76873570/download?download_frd=1)

Requirements

You must write a program to find all Fluky Numbers from 1 to 10000. As a tip to help make sure you're on the right track, the first 3 Fluky Numbers are: 8, 22, and 25. Your program must discover these numbers, you may NOT use code such as the following:

```
if testNum == 8:
    print("Found a Fluky Number!")
```

You may know that there are 7 Fluky numbers between 1 and 10000.

Your program must print the total running time in seconds to two decimal places. It must print the total number of inner loops executed. Here is an example output, without actual numbers showing: (Note: An 'X' does not indicate the number of digits or any other meaning. It's just a filler)

```
Fluky Number: 8
Fluky Number: 22
Fluky Number: 25
Fluky Number: XX
Fluky Number: XX
Fluky Number: XX
Fluky Number: XX
Total Time:  XX.XX seconds
Total Loops:  XXXXXXXXXX
```

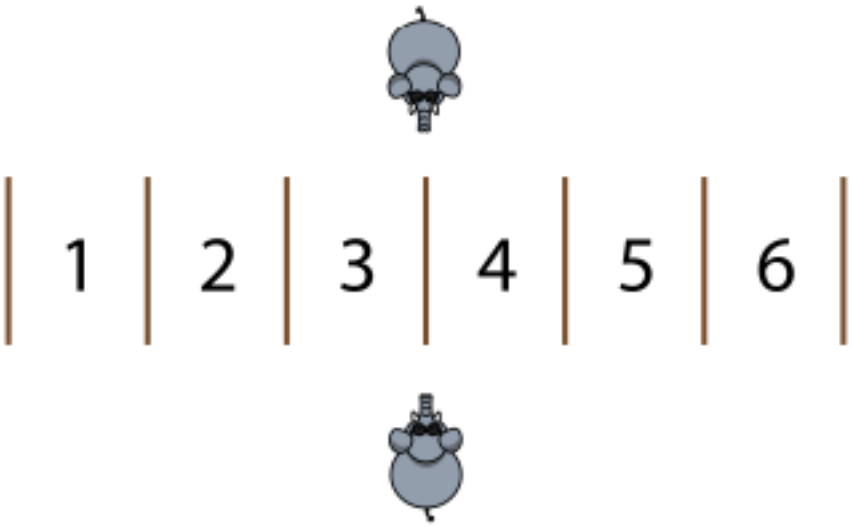
See Rubric for all requirements. Make sure to catch the points for reducing the number of required loops.

NOTE: Your code must discover all seven Fluky Numbers using a nested loop structure. You cannot use any information about properties of Fluky Numbers other than what is described. Your program may know that there will be exactly seven Fluky Numbers numbers with values of 10,000 or less. You may use basic math/logic principles to help you reduce the number of iterations. If you're reading through wikipedia or doing google searches then you are over-thinking it. Think through in your planning how you can reduce the number of iterations without affecting the results...and think simply.

Task 2 - Elephant Cages

Save your program in a file called task2.py and your software development plan in a file called plan2.txt. Here is a quick story about two elephants who live in the Potowatomi Zoo. Every evening the elephants each randomly select one of six sleeping pens to spend the night in. After deciding on which pen, they simultaneously enter the pen from opposite sides. Being slightly vision impaired, they cannot see if the other elephant entered the same pen until it is too late to turn around and go

to another pen to spend the night. Thus sometimes they must each spend an uncomfortable night sharing a small sleeping area with another elephant.



After the elephants are asleep the zookeeper randomly picks one of the pens to check. During a break, the zookeeper and the nighttime custodian were talking. The zookeeper told the custodian that 1/3rd of the time when he checks a pen, there's at least one elephant in there, and of the times that there's at least one elephant in the pen, 1/6th of the time the other elephant is in there as well. The custodian said that those percentages cannot be correct, but the zookeeper insisted they are.

Your job is to write a program to simulate this nightly activity 100,000 times to determine if the zookeeper or the custodian is correct. As a simulation, you will program all steps of this process. You are not writing a statistical analysis program to calculate probabilities.

You will report the following statistics.

1. What percentage of the time there is at least one elephant in the pen the zookeeper checks?
2. When there is an elephant in the pen the zookeeper checks, what percentage of time are both elephants in the pen?
3. Based on the results of your simulation, is the zookeeper or the custodian correct? (Use 2% points as the margin of error)

Your results should be displayed as percentages with two decimal places.

After your results are displayed, you will prompt the user to choose to end the program or run the simulation again. That way, whomever the results indicate is wrong, can easily try another simulation to see if the results were a fluke. You can use a simple message such as:

```
Run the simulation again? (yes or no):
```

Read the rubric for all requirements.