A Comparative Analysis of Tools & Task Types for Measuring Computational Problem-Solving

Engin Bumbacher
engin.bumbacher@hepl.ch
Haute École Pédagogique du Canton de Vaud (HEP-VD)
Lausanne, Vaud, Switzerland

Jérôme Brender
jerome.brender@hepl.ch
Haute École Pédagogique du Canton de Vaud (HEP-VD)
Lausanne, Vaud, Switzerland

Richard Lee Davis
richard.davis@epfl.ch
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Vaud, Switzerland

ABSTRACT

How to measure students' Computational Problem-Solving (CPS) competencies is an ongoing research topic. Prevalent approaches vary by measurement tools (e.g., interactive programming, multiple-choice tests, or programming-independent tests) and task types (e.g., debugging problems or Parson problems). However, few studies have examined the measurement tools of CPS competencies themselves: affordances and limitations of the measurement tools and how they compare, or whether different task types might elicit CPS competencies differently. Research needs to address these questions in order to better understand how to design robust, generalizable, and effective measurement tools for CPS competencies. This paper presents an exploratory study that contributes to this research direction. It is part of a larger international project to develop an open-access formative assessment platform for CPS, which includes a novel authoring tool for a wide range of task types for interactive block-based programming. We used the tool to create an interactive programming experience with multiple task types and gave it to more than 300 secondary school students from different countries. We also administered a validated multiple-choice measurement of Computational Thinking with block-based programs. We focused on task complexity as a characteristic of task type, using a classification scheme based on task design features. Comparing students' performances on tasks of different complexity and using two distinct measurement tools, we found that the multiple-choice measurement only partially predicts performance in the interactive programming task. Additionally, its predictive capacity varies significantly between task types of differing complexity.

ACM Reference Format:
Engin Bumbacher, Jérôme Brender, and Richard Lee Davis. 2024. A Comparative Analysis of Tools & Task Types for Measuring Computational Problem-Solving. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635547

1 FURTHER INFORMATION

Computational Thinking (CT) is widely recognized as a crucial skill in the twenty-first century [2]. It is involved in the formulation, evaluation, and implementation of solutions to complex computational problems [3, 8]. There is an increasing number of learning environments and pedagogical approaches for teaching computational problem-solving (CPS), but also a lack of agreed-upon tools for measuring CT practices in CPS - tools that are necessary to help advance research [4]. Current approaches range from multiple-choice tests using programming-based tasks to interactive experiences with novel immersive virtual environments. In one of the more comprehensive studies of the few that examined the relationship between different measurement tools, Román-González et al. [6] found only a partial convergence between the most extensively used tools: the psychometrically validated paper-based assessment CTt, the Dr. Scratch [5] scoring tool for Scratch programs, and the programming-independent paper-based assessment of CT, Bebras [1]. This indicates that each test captures only some aspects of CT practices, leaving unexplained a significant proportion of other ones. A combination of the different measurement tools, as suggested by the authors, will not resolve this problem as long as we do not better understand what aspects of CT practices each measurement tool can and cannot capture, and how they relate to each other. In other words, we need to be able to answer questions such as whether a debugging task in multiple-choice format, as used in the CTt, captures the same aspect of debugging practices as an interactive programming task, or whether performance on a CPS task varies when changing design features of the task such as the degrees of freedom in finding a solution, the kind of interactivity in the task, etc.

This paper presents an exploratory empirical comparison of different measurement approaches to begin to address these questions, in order to contribute to more informed methodologies for designing future measurement approaches. Specifically, we address the following research question: To what extent does performance on a validated multiple-choice instrument of CT predict performance on interactive CPS tasks, and does this relationship depend on the complexity of the programming task?

In order to address this question, we used a novel web-based Platform for Innovative Learning Assessments (PILA) in collaboration with the OECD's Programme for International Student Assessment (PISA). This platform allows the creation and distribution of highly customized assessment experiences with interactive CPS tasks and records all process data. We worked with the "Karel" programming application, a block-based programming environment that requires students to guide the turtle-like agent "Karel" through a grid-like world while arranging stones in certain ways. The novel authoring tool allows for varying the complexity of the tasks by manipulating a wide range of design features such as the number, size and shape of the grid-like worlds to be solved with a program, the types of

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635547

programming blocks available, the limits to block use, the available starter code, etc.

We designed two assessment experiences containing eight and six interactive CPS tasks of varying complexity. Tasks required students complete, assemble, debug or modify given code, involved simple or more complicated worlds. Throughout the experiences, three programming concepts were introduced: functions, while loops, and conditionals. Using an a priori coding scheme for task complexity, which we will present in the poster, we classified each task as being of "low complexity" or of "high complexity". Task complexity was varied within each assessment experience.

The participants were 305 secondary school students between 12 and 18 years from a total of 28 teachers in various schools from five countries (Singapore, Ireland, the Netherlands, and Brazil). These schools responded to an open call for participation by the OECD's PISA. Prior to the start of the study, ethical approval was secured, but we could not collect any background information on the students.

The study was structured in three parts. Initially, students completed a self-report questionnaire on their prior computer science and programming experience. Subsequently, they completed the B+CTt paper-based multiple-choice test developed by Wiebe et al. [7]. After a brief tutorial on using the "Karel" application, students worked through the two 45-minute assessment sessions on consecutive days.

For each student, we calculated a B+CTt test score as the percentage of the maximum score possible, a compound measure of prior expertise as the aggregate of students' responses to three closed-form self-report questions on prior programming experiences, and a programming score for the performance in the CPS assessment experiences. Both the success rate and the percentage of students who could attempt a task varied significantly between tasks, with an average overall score of 45% (SD=22%) across both experiences. The rate of non-attempts increased with each subsequent task within an experience. At the same time, the 7 low complexity and 7 high complexity tasks were arranged differently within each assessment experience, with the low complexity tasks being placed mostly in the first half of an experience. Thus, the average rate of non-attempts was significantly different between low and high complexity tasks. This led us to decide to run separate regression models for the two categories instead of combining them into a single model with the task category as an independent interactive factor. Qualitative comparisons of coefficients between models are possible because we used standardized covariates. We used linear mixed-effect models on performance in programming tasks, with random intercepts for country and for teachers nested in country. Table 1 shows the three mixed-effects models, with dependent variables total score on all tasks, total score on just the low complexity tasks, and total score on just the high complexity tasks. The results show that both the B+CTt score and the compound measure of prior expertise are significantly predictive of performance on the interactive programming tasks in all three models. However, while the main effect of prior expertise remains fairly similar across models, the main effect of B+CTt score varies significantly, and is highest for the low complexity tasks. Similarly, the interaction effect between the two covariates varies across models, and is highest for the high complexity tasks. These results suggest there is partial

Table 1: Mixed-effects models for total scores per task category. Covariates have been standardized. Numbers in parentheses indicate the 95% confidence intervals.

	All Tasks	**Low Complexity**	**High Complexity**
	Total Score	*Total Score*	*Total Score*
(Intercept)	0.09 (-0.32–0.51)	0.00 (-0.12–0.12)	0.12 (-0.40–0.63)
B+CTt Score	0.51 *** (0.40–0.63)	0.60 *** (0.50–0.71)	0.40 *** (0.28–0.51)
Prior Expertise	0.14 *** (0.07–0.21)	0.14 ** (0.05–0.23)	0.12 ** (0.05–0.19)
B+CTt Score:Prior Exp.	0.07 * (0.00–0.13)	-0.06 (-0.14–0.02)	0.16 *** (0.10–0.23)
Random Effects			
σ^2	0.32	0.53	0.33
ICC	0.37	0.08	0.46
Marginal / Cond. R^2	0.38 / 0.61	0.42 / 0.46	0.27 / 0.60

** p<0.05** p<0.01*** p<0.001*

convergence between the B+CTt and the interactive programming experiences. However, the strength of prediction of the B+CTt for performance on the Karel tasks varies with task complexity. There is a larger proportion of variance in performance that cannot be explained by the B+CTt for tasks of high complexity compared to tasks of low complexity. Taken together, this exploratory study emphasizes the importance of paying more attention to how task characteristics such as complexity influence how students engage in computational problem-solving practices and how they perform across different measurement tools. Further research is needed to better understand the relationship between task complexity and predictive capacity of the paper-based closed-response tasks. This requires research tools like the novel web-based PILA, that we are co-developing with the OECD's PISA organization. We will present more details in the poster on both the web-based platform that we used and the study design and results.

REFERENCES

[1] Valentina Dagiene and Gabriele Stupuriene. 2016. Bebras–A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in education* 15, 1 (2016), 25–44.

[2] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational researcher* 42, 1 (2013), 38–43.

[3] Shuchi Grover and Roy Pea. 2018. Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school* 19, 1 (2018), 19–38.

[4] Josef Guggemos, Sabine Seufert, and Marcos Román-González. 2023. Computational thinking assessment–towards more vivid interpretations. *Technology, Knowledge and learning* 28, 2 (2023), 539–568.

[5] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2015. Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia* 46 (2015), 1–23.

[6] Marcos Román-González, Jesús Moreno-León, and Gregorio Robles. 2019. Combining assessment tools for a comprehensive evaluation of computational thinking interventions. *Computational thinking education* (2019), 79–98.

[7] Eric Wiebe, Jennifer London, Osman Aksit, Bradford W Mott, Kristy Elizabeth Boyer, and James C Lester. 2019. Development of a lean computational thinking abilities assessment for middle grades students. In *Proceedings of the 50th ACM technical symposium on computer science education*. 456–461.

[8] Jeannette Wing. 2017. Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology* 25, 2 (2017), 7–14.