Personalized Assessment Worksheets for Scratch (PAWS): Exploring a Bridge between Interviews, Written Assessments, and Artifact Analysis

Jean Salac
University of Chicago
Chicago, IL, USA
salac@uchicago.edu

ABSTRACT

The computer science community has struggled to assess student learning, especially at the early elementary level. Prior work has included one-on-one interviews, written assessments, and artifact analysis, each with their own benefits and drawbacks. Through our Personalized Assessment Worksheets for Scratch (PAWS) tool, we explore personalized assessments as an assessment technique that lies in between interviews, written assessments, and artifact analysis. PAWS creates personalized written assessments that integrates code from student Scratch projects. We hope that our PAWS tool, and more generally personalized assessments, will lead to an assessment technique that is both more accurate than written assessments and artifact analysis, and less time-consuming than interviews.

KEYWORDS

K-8 education, assessment, Scratch, computational thinking

ACM Reference Format:
Jean Salac. 2019. Personalized Assessment Worksheets for Scratch (PAWS): Exploring a Bridge between Interviews, Written Assessments, and Artifact Analysis. In *Proceedings of ACM International Computing Education Research Conference (ICER 2019).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3291279.3339428

1 MOTIVATION & KEY IDEAS

Many countries worldwide are integrating Computer Science (CS) and Computational Thinking (CT) instruction into their K-8 school systems, including the United States, New Zealand, Israel, and India [7]. Moving from the informal, optional domain into the formal school classroom increases the pressure on developing accurate assessment techniques that match the pedagogical approaches and tools used for this age group.

A popular programming language and development environment used in elementary school is Scratch [4]. Three assessment techniques are common in this realm: analyzing the programs that students create, giving written assessments, and interviewing student. Each has their own benefits and drawbacks, detailed in the next section. With our PAWS tool, we explore the space between interviews, written assessments, and artifact analysis. The ultimate goal is to create written assessments that allows students to demonstrate their understanding of code both present and not present in their artifacts. Our research seeks to answer two research questions:

- How should personalized questions and student code selection criteria be designed?
- How does integrating a student's code from their artifacts affect how they answer written assessment questions?

2 RELATED WORKS

We provide background on three methods of assessment: artifact analysis, written assessments, and interviews.

There is a wealth of literature on artifact analysis, including Scrape [12], Hairball [2], and Dr. Scratch [10]. However, any technique focused on artifact analysis assumes that students understand the code they use in their projects. This is not necessarily true, as identified by Brennan [3]. Students can use code in their projects that they do not truly understand, either by copying exact code they were taught or remixing from the Scratch community. Written assessments or interviews are necessary to find out whether students understand the concepts both included and not included in their code.

Traditional written assessments are frequently used to assess student learning in Scratch, both in the school [6, 9] and the extracurricular setting [5, 8]. However, while several groups have *used* written assessments, they have not been validated, so they may not measure what they are intended to measure. Very few validated assessments exist, and those that do are designed for older audiences, such as college-level CS1 students [11], and middle school students students [1].

Interviews provide a more nuanced and personalized way of assessing student learning. Brennan and Resnick found that through artifact-based interviews, they were able to identify the depth of a student's understanding of a particular concept and assess how students were employing computational thinking practices while developing their projects [3]. While interviews can provide a more complete picture of student learning, they are very time-consuming, making them unrealistic for teachers who are already very time-constrained.

Our PAWS tool builds upon previous research in assessing student learning by exploring personalized written assessments that use code snippets from student artifacts as a possible bridge between the three methods of assessment.

3 RESEARCH APPROACH & METHODS

3.1 PAWS Tool

Our PAWS tool searches Scratch projects for code snippets that are "suitable" for personalized questions. These code snippets have to meet different sets of requirements for each question in order to be "suitable". If suitable code exists in the student project, PAWS randomly assigns the student a personalized question using their code snippet or code from a generic question to allow for comparison between the personalized and generic questions. Designed to take students about 20 to 30 minutes to complete, PAWS assessments consist of multiple-choice, fill-in-the-blank and open-ended questions.

3.2 Question Design

At the crux of designing questions using students' own code is this research question: *How do we design questions that assess student understanding of their code, as opposed to their memory of the code execution?* One-on-one interviews face similar challenges, where students are recalling what their code did, as opposed to reading the code presented to them [3].

Multiple-choice and fill-in-the-blank questions may ask students to predict the outcome of a code execution. If code were to be used verbatim from student projects, students may just remember the code execution, instead of actually tracing the code. Similarly, students answering an open-ended question about their own code may draw from their memory of their code when describing it. We would like to discuss our question design and code selection criteria with DC mentors and attendees to get different perspectives regarding this challenge.

3.3 Research Design

At least three schools will be chosen to participate, each with different academic performance levels and diversity of student body. Students in 3rd grade will complete learning modules for sequence, events, and loops, whereas students in 4th and 5th grade will complete learning modules for conditionals. Assessments will be given at the end of the (1) events and sequence, (2) loops, and (3) conditionals modules.

In one mid-performing classroom for each grade, a separate interview-based personalized assessment will be created, and an interviewer will ask students about their code to assess the level of understanding of their code and the concepts involved. Interview questions will only be asked about code pertaining to the concepts covered in the assessment.

Both quantitative (i.e ANOVA, correlation, etc) and qualitative analysis (i.e. content analysis) methods will be used. Analysis will be performed on several pieces of data: (a) presence of code constructs in projects, (b) complexity of use of code constructs in projects, (c) performance on written assessment (generic questions), (d) performance on written assessment (personalized questions), and (e) knowledge demonstrated in interviews.

4 PROGRESS THUS FAR

In the 2017-18 school year, we launched an initial trial of PAWS in 14 classrooms from 4 schools in a large, urban school district, for a total of 296 4th-grade students. PAWS was used for two modules in their CT curriculum – one module on events and sequence, and another on loops.

For the current school year (2018-19), we revised our assessments on events and sequence, and loops, as well as the selection criteria for student code, based on the lessons learned from the previous school year. We have also developed a generic assessment for conditionals, which will be piloted in a few classrooms. After this pilot, we will revise the assessment questions and develop the personalization component.

Due to the enthusiastic student reception from seeing their own code in the assessments in the first year, 26 classrooms from 6 schools are now using PAWS. We are planning on submitting an experience paper to SIGCSE 2020, detailing the lessons learned from both initial trial years. The assessments from our initial trial have not been validated yet, but we are in the process of recruiting an assessment validation expert.

REFERENCES

[1] Min W. Wiebe E. Mott B. Boyer K. E. Lester J. Akram B. "Assessing Middle School Students' Computational Thinking Through Programming Trajectory Analysis". In: *Proceedings of the 50th ACM technical symposium on Computer science education.* ACM. 2019, p. 1269.

[2] Bryce Boe et al. "Hairball: Lint-inspired Static Analysis of Scratch Projects". In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education.* SIGCSE '13. Denver, Colorado, USA: ACM, 2013, pp. 215–220. ISBN: 978-1-4503-1868-6. DOI: 10.1145/2445196.2445265. URL: http://doi.acm.org/10.1145/2445196.2445265.

[3] Karen Brennan and Mitchel Resnick. "New frameworks for studying and assessing the development of computational thinking". In: *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.* Vol. 1. 2012, p. 25.

[4] Louise P Flannery et al. "Designing ScratchJr: support for early childhood learning through computer programming". In: *Proceedings of the 12th International Conference on Interaction Design and Children.* ACM. 2013, pp. 1–10.

[5] Diana Franklin et al. "Assessment of computer science learning in a scratch-based outreach program". In: *Proceeding of the 44th ACM technical symposium on Computer science education.* ACM. 2013, pp. 371–376.

[6] Michal Gordon, Assaf Marron, and Orni Meerbaum-Salant. "Spaghetti for the main course?: observations on the naturalness of scenario-based programming". In: *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education.* ACM. 2012, pp. 198–203.

[7] Peter Hubwieser et al. "A global snapshot of computer science education in K-12 schools". In: *Proceedings of the 2015 ITiCSE on working group reports.* ACM. 2015, pp. 65–83.

[8] Colleen M Lewis and Niral Shah. "Building upon and enriching grade four mathematics standards with programming curriculum". In: *Proceedings of the 43rd ACM technical symposium on Computer Science Education.* ACM. 2012, pp. 57–62.

[9] Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. "Learning computer science concepts with scratch". In: *Computer Science Education* 23.3 (2013), pp. 239–264.

[10] Jesús Moreno-León et al. "On the Automatic Assessment of Computational Thinking Skills: A Comparison with Human Experts". In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems.* CHI EA '17. Denver, Colorado, USA: ACM, 2017, pp. 2788–2795. ISBN: 978-1-4503-4656-6. DOI: 10.1145/3027063.3053216. URL: http://doi.acm.org/10.1145/3027063.3053216.

[11] Allison Elliott Tew and Mark Guzdial. "Developing a validated assessment of fundamental CS1 concepts". In: *Proceedings of the 41st ACM technical symposium on Computer science education.* ACM. 2010, pp. 97–101.

[12] Ursula Wolz, Christopher Hallberg, and Brett Taylor. "Scrape: A tool for visualizing the code of Scratch programs". In: *Poster presented at the 42nd ACM Technical Symposium on Computer Science Education, Dallas, TX.* 2011.