TYPOS: A Computer Science Exercise Platform

Adam M. Gaweda
North Carolina State University
Raleigh, NC, USA
agaweda@ncsu.edu

Collin F. Lynch
North Carolina State University
Raleigh, NC, USA
cflynch@ncsu.edu

ABSTRACT

Computer Science has a number of exercise types available for learning. However, it is unknown when the appropriate exercise type should be given to students on their path to learning CS. This paper describes TYPOS, a Computer Science Exercise Platform that hosts a variety of exercise types. These CS exercises range in complexity and interactivity based on the ICAP framework. As part of this paper, we provide a brief overview over each exercise type and their respective complexity. Finally, we present considerations for future research on using TYPOS for activity sequence mining and suggested next practice activities for students.

Author Keywords

computer science exercise, exercise type, activity sequences

CCS Concepts

•**Social and professional topics** $\rightarrow$ **Computing literacy;** *Model curricula;* •**Applied computing** $\rightarrow$ *Interactive learning environments;*

INTRODUCTION

Learning Computer Science (CS) is difficult and college-level introductory CS courses often experience high attrition rates [2]. CS is a combination of the absolute rigor of mathematics and logic with the flexibility of planning and problem solving. However, this combination creates many issues for novices attempting to learn the domain. There are lower-level aspects, such as constructing semantically valid sentences from syntax, to higher-level aspects, such as problem decomposition and subgoal generation; each of which novices struggle to learn.

CS Education often experiments with different types of learning aids. These additional aids often come in the form of *exercise types*, providing learning opportunities through a variety of tasks [12]. These exercise types come in the form of worked examples, typing exercises, Parsons Puzzles, debugging erroneous code, self-explanation, or traditional programming exercises. Each of these exercise types can be mapped to Chi's ICAP framework, which categorizes different types of activities based on their user engagement (interactive, constructive, active, and passive, respectively) [5]. By increasing the level of engagement, the learner engages in information seeking and must process the activity at a finer detail than passively observing it.

However, many of the studies that use these aids only focus on the benefits of a single exercise type. Each type reinforces different elements, but a particular exercise may not be the most appropriate intervention for a given student at that point in time. Ericsson presents a similar criticism about considering all types of practice equal [8]. Some activities only benefit particular skills, for example Parson's Puzzles improving code writing but not code reading [6]. Similarly, high complexity exercises like self-explanation are only found beneficial for more knowledgeable students [1]. Thus, determining the appropriate learning activity for each student can be a non-trivial manner while learning CS.

TYPOS EXERCISE PLATFORM

In an effort to study the most appropriate exercise to present students, we have developed TYPOS (named for how common syntax errors are in programming). TYPOS is a Computer Science Exercise Platform where students are presented with various novel exercise types[1]. The platform was designed to allow instructors to place exercises in modules within a course. Prior research involving TYPOS showed self-selected students that regularly completed *typing exercises* performed better in the course and submitted less syntax errors in course programming projects [9]. Since that publication, we have expanded TYPOS to support eight (8) different exercise types for CS practice. We list each exercise type, its ICAP category, and a description of the exercise in Table 1.

In addition to supporting a variety of exercise types, TYPOS records student activity within exercises. Students are given a *heartbeat* that monitors when a student begins an exercise, their engagement during the exercise, when they make an incorrect submission, and when they make a correct submission. Each submission is also stored with the feedback given to the student. This allows researchers to build an accurate timeline of events while the student is practicing on the platform.

TYPOS also allows for 'tags' to exercises. Tags allow instructors and researchers to provide metadata about the exercise, such as the particular CS concepts used, prerequisites necessary for the exercise, or specific techniques employed. For example, an exercise the requires loops could be tagged with variable and conditional prerequisites and indicate the use of the 'Gatherer' role from the 'roles of variables' literature [11].

MINING ACTIVITY SELECTION

Incorporating additional practice options would provide researchers the opportunity to mine activity selection. However, it

Sixth SPLICE Workshop at L@S 2020 "Building an Infrastructure for Computer Science Education Research and Practice at Scale", SPLICE'20, August 12, 2020, Virtual Event

[1]https://research.csc.ncsu.edu/arglab/projects/typos/

Exercise Type	ICAP Category	Description
Typing Exercises	Active	Retype worked examples of completed code template [9].
Fill in the Blank	Active	Replace a blanked out segment with the appropriate code or syntax keyword [1].
Parsons Puzzles	Constructive	Reassemble shuffled code segments into the appropriate order [6].
Output Prediction	Constructive	Analyze a snippet of code to determine the value of a variable or output of the snippet [12].
Documenting Code	Constructive	Create a documentation string describing how a snippet of code operates [1].
Find the Bug	Constructive	Highlight the area of a snippet of code where an error occurs.
Fix the Bug	Interactive	Locate and resolve errors in a snippet of code [4].
Coding Exercises	Interactive	Implement a coding solution given a problem statement.

Table 1. Computer Science Exercise Types Supported by TYPOS with its ICAP category and brief description

becomes a non-trivial task to determine which exercise is most appropriate as the number of exercise types increases. Randomizing exercise types would no longer be a feasible baseline without compromising the ultimate goal of teaching students. Randomly selecting higher complexity exercises types too soon would be ineffective for lower-performing students and could lead to frustration.

Another approach would be to allow students to select which exercise type to practice next with a final module competency test to show proficiency. Allowing students to select their practice regimen would then open the door to researchers to study the exercise types selected. Lower-performing or slower learning students would be able to receive equitable practice opportunities while higher-performing or faster learning students would not need to practice already learned objectives [7]. This would allow researchers to analyze individual practice behaviors that may support proposed learning theory models.

Analyzing activity sequences have been used to recommend e-textbook links [3], but can expand beyond passive reading to include additional learning activities. Similar approaches have found success on a smaller scale, by using student behaviors to suggest 'next-step' hints while learning block-based programming [10]. In Price's work, students that requested a hint were provided with adaptive hints pointing to similar, correct solutions. This approach could be adopted to practice recommendations where a correct solution was passing the module competency test.

CONCLUSION

In this paper, we presented TYPOS, a CS Exercise Platform designed to provide students with different exercise types while learning Computer Science. TYPOS monitors student behaviors within the platform and can be used to study exercise effectiveness, student exercise selection, and activity sequences. These activity sequences are beneficial to both instructors and designers of intelligent tutoring systems. Activity sequences that identify struggling students could suggest appropriate lower-level practices to ensure learning. As activity sequencing becomes more well-defined, instructors and intelligent tutoring systems will be able provide a tailored list of practice problems for each student based on their needs.

REFERENCES

[1] Robert K Atkinson, Sharon J Derry, Alexander Renkl, and Donald Wortham. 2000. Learning from examples: Instructional principles from the worked examples research. *Review of educational research* 70, 2 (2000), 181–214.

[2] Jens Bennedsen and Michael E Caspersen. 2019. Failure rates in introductory programming: 12 years later. *ACM Inroads* 10, 2 (2019), 30–36.

[3] Peter Brusilovsky. 2012. Adaptive hypermedia for education and training. *Adaptive technologies for training and education* 46 (2012), 46–68.

[4] Nick Cheng and Brian Harrington. 2017. The Code Mangler: Evaluating Coding Ability Without Writing any Code. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 123–128.

[5] Michelene TH Chi and Ruth Wylie. 2014. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist* 49, 4 (2014), 219–243.

[6] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2008. Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*. ACM, 113–124.

[7] Shayan Doroudi and Emma Brunskill. 2019. Fairer but not fair enough on the equitability of knowledge tracing. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*. 335–339.

[8] K Anders Ericsson. 2016. Summing up hours of any type of practice versus identifying optimal practice activities: Commentary on Macnamara, Moreau, & Hambrick (2016). *Perspectives on Psychological Science* 11, 3 (2016), 351–354.

[9] Adam M Gaweda, Collin F Lynch, Nathan Seamon, Gabriel Silva de Oliveira, and Alay Deliwa. 2020. Typing Exercises as Interactive Worked Examples for Deliberate Practice in CS Courses. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*. 105–113.

[10] Thomas W Price, Rui Zhi, Yihuan Dong, Nicholas Lytle, and Tiffany Barnes. 2018. The impact of data quantity and source on the quality of data-driven hints for programming. In *International Conference on Artificial Intelligence in Education*. Springer, 476–490.

[11] Jorma Sajaniemi. 2002. An empirical analysis of roles of variables in novice-level procedural programs. In *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments*. IEEE, 37–39.

[12] Greg Wilson. 2019. *Teaching Tech Together: How to Make Your Lessons Work and Build a Teaching Community around Them*. CRC Press.