

A Debugging Learning Trajectory for Text-Based Programming Learners

Hanxiang Du
University of Florida
Gainesville, FL, USA
h.du@ufl.edu

Wanli Xing
University of Florida
Gainesville, FL, USA
wanli.xing@coe.ufl.edu

Yuanlin Zhang
Texas Tech University
Lubbock, TX, USA
y.zhang@ttu.edu

ABSTRACT

Novice programming learners encounter programming errors on a regular basis. Resolving programming errors, which is also known as debugging, is not easy yet important to programming learning. Students with poor debugging ability hardly perform well on programming courses. A debugging learning trajectory which identifies learning goals, learning pathways, and instructional activities will benefit debugging learning activities development. This study aims to develop a debugging learning trajectory for text-based programming learners. This is accomplished through (1) analyzing programming errors in a logic programming learning environment and (2) examining existing literature on debugging analysis.

CCS CONCEPTS

• **Social and professional topics** → **Computing education.**

ACM Reference Format:

Hanxiang Du, Wanli Xing, and Yuanlin Zhang. 2021. A Debugging Learning Trajectory for Text-Based Programming Learners. In *26th ACM Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2021)*, June 26–July 1, 2021, Virtual Event, Germany. ACM, New York, NY, USA, 1 page. <https://doi.org/10.1145/3456565.3460049>

1 INTRODUCTION

Debugging is inevitable and important in programming learning. One way to improve students' debugging skills is through thoughtfully developed learning activities. Learning trajectories (LTs), which are defined as the projection of how learning proceeds over time through learning activities, are essential to curriculum development [2, 3]. Although a number of studies have focused on error analysis, there is a lack of work on debugging LT development for text-based programming languages.

Rich et al. [2] developed a debugging LT for computational thinking learning in elementary schools through a systematic examination of existing literature. They extracted learning goals from scholarly research work and synthesized consensus goals. Consensus goals were categorized as three dimensions of debugging learning (strategies for finding and fixing errors, types of errors, the role of errors in problem solving) and three levels of debugging skills (beginning, intermediate, advanced). This study aimed to adapt their work to text-based programming languages.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ITiCSE 2021, June 26–July 1, 2021, Virtual Event, Germany

© 2021 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8397-4/21/06.

<https://doi.org/10.1145/3456565.3460049>

2 OVERVIEW

As innovative as Rich et al.'s [2] debugging LT is, several aspects worth further investigation. First, the starting point of the LT. As mentioned in their work, their debugging LT best suits graphic programming. Graphic programming environment does not report an error in the way which text-based programming languages do. Most of the time, to know if there is an error is effortless using text-based programming languages as the compiler or the environment will tell programmers so. However, this task is the foundation skill in [2]. Second, their error categorization compromises on the complexity of errors and the difficulty in resolving different types of errors. Rich et al.'s [2] three levels of debugging skills (beginning, intermediate, and advanced) is based on students' age in the studies, rather than the complexity of errors. They also assume all errors (small errors and errors of omission) belong to the same debugging skill level, which is not the case in text-based programming languages. Third, the misuse of iterative refinement and iterative debugging process. In its original context, iterative debugging process describes a student resolving an error with multiple attempts, comparing with those in one attempt [1]. From programming practice perspective, iterative refinement describes a developing process which starts from a simple version and enriching it bit by bit. Last, a definition of debugging, which is missing in their work, is necessary as it specifies what is and what is not debugging.

This work defines debugging as the activities to locate the error, come up with a solution and resolve the error. We propose that to exclude knowing if there is an error as a foundation skill, and levels of debugging skills shall be based on error complexity, instead of learners' age.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation (NSF) of the United States under grant number 1901704. Any opinions, findings, and conclusions or recommendations expressed in this paper, however, are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

- [1] Louise P Flannery and Marina Umaschi Bers. 2013. Let's dance the "robot hokey-pokey!" children's programming approaches and achievement throughout early cognitive development. *Journal of research on technology in education* 46, 1 (2013), 81–101.
- [2] Kathryn M Rich, Carla Strickland, T Andrew Binkowski, and Diana Franklin. 2019. A k-8 debugging learning trajectory derived from research literature. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 745–751.
- [3] Martin A Simon. 1995. Reconstructing mathematics pedagogy from a constructivist perspective. *Journal for research in mathematics education* 26, 2 (1995), 114–145.