Assessing How Pre-requisite Skills Affect Learning of Advanced Concepts

Greg L. Nelson*
University of Washington
Seattle, Washington
glnelson@uw.edu

Filip Strömbäck*
Linköping University
Linköping, Sweden
filip.stromback@liu.se

Ari Korhonen*
Aalto University
Espoo, Finland
archie@cs.hut.fi

Ibrahim Albluwi
Princeton University
Princeton, New Jersey
isma@cs.princeton.edu

Marjahan Begum
Copenhagen Business School
Copenhagen, Denmark
mbe.msc@cbs.dk

Ben Blamey
Uppsala University
Uppsala, Sweden
ben.blamey@it.uu.se

Karen H. Jin
University of New Hampshire
Manchester, New Hampshire
karen.jin@unh.edu

Violetta Lonati
Università degli Studi di Milano
Milano, Italy
lonati@di.unimi.it

Bonnie MacKellar
St John's University
Queens, New York
mackellb@stjohns.edu

Mattia Monga
Università degli Studi di Milano
Milano, Italy
mattia.monga@unimi.it

ABSTRACT

Students often struggle with advanced computing courses, and comparatively few studies have looked into the reasons for this. It seems that learners do not master the most basic concepts, or forget them between courses. If so, remedial practice could improve learning, but instructors rightly will not use scarce time for this without strong evidence. Based on personal observation, program tracing seems to be an important pre-requisite skill, but there is yet little research that provides evidence for this observation.

To investigate this, our group will create theory-based assessments on how tracing knowledge affects learning of advanced topics, such as data structures, algorithms, and concurrency. This working group will identify relevant concepts in advanced courses, then conceptually analyze their pre-requisites and where an imagined student with some tracing difficulties would encounter barriers. The group will use this theory to create instructor-usable assessments for advanced topics that also identify issues caused by poor pre-requisite knowledge. These assessments may then be used at the start and end of advanced courses to evaluate to what extent students' difficulties with the advanced course originate from poor pre-requisite knowledge.

*Leaders

ITiCSE '20, June 15–19, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-6874-2/20/06.
https://doi.org/10.1145/3341525.3394990

CCS CONCEPTS

• **Applied computing → Education**; • **Theory of computation** → *Concurrency*; *Design and analysis of algorithms*; • **Information systems** → *Data structures.*

KEYWORDS

computer science education, concurrency, data structures and algorithms, tracing

ACM Reference Format:
Greg L. Nelson, Filip Strömbäck, Ari Korhonen, Ibrahim Albluwi, Marjahan Begum, Ben Blamey, Karen H. Jin, Violetta Lonati, Bonnie MacKellar, and Mattia Monga. 2020. Assessing How Pre-requisite Skills Affect Learning of Advanced Concepts. In *2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'20), June 15–19, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3341525.3394990

1 INTRODUCTION

In computing education, we have initial evidence that learners have weaknesses with pre-requisite knowledge when they take advanced classes, which may negatively affect learning outcomes. The Lister 2004 working group suggested many students lack mastery of program tracing after CS1 [5]. Other studies show this may continue into later courses. Valstar et al. showed more than 30% of students could not do questions on pointers or tracing recursion at the start of an upper level data structures class, and this correlated with final exam scores [11]. Fisler et al. showed more than 30% of 3rd and 4th year CS majors failed questions on scope, parameter mutation, and/or variable mutation [2]. These prior studies did not use learning outcome questions carefully designed to detect difficulties caused by poor pre-requisite knowledge, or assessments

Table 1: Abilities in advanced concepts in concurrency based on tracing knowledge

Tracing weaknesses	Concurrency stage #1: Identify shared data	Concurrency stage #2: Individual locking of shared data
No weaknesses	+ Can identify concurrency issues - Unable to solve any issues present	+ Can solve simple tasks with individual variables - Unable to solve problems with multiple dependent variables
Scoping weakness: Local variables are shared between different calls	+ Can identify concurrency issues - Will identify additional variables as problematic	+ Can solve simple tasks with individual variables - Too much synchronization due to misconception - Incorrect placement of lock variables
Scoping weakness: Reference vs. value semantics	+ Can identify some concurrency issues - Unable to identify data shared by reference	+ Can solve simple tasks with simple variables - Missing synchronization for data shared by reference

with validity arguments. The Basic Data Structures Inventory with a validity argument was just published in 2019 [8]. Other work towards validity has designed assessments or assessment questions for particular advanced topics [1, 3, 4, 10] or tracing [6, 7].

To deeply understand why and how often pre-requisite knowledge difficulties affect later learning, careful assessment design and validity work is needed; sometimes assessment questions are not detailed enough to diagnose 1) difficulties with the more advanced concept vs. 2) weaknesses with more basic concepts that may lead to incorrect application of the advanced concept. For example, suppose the advanced concept involves synchronizing data for concurrency, and the weakness is scoping (what variables are shared across function calls/class instances). The following seemingly good question does not separate case 1 vs. 2: asking a learner to add concurrency controls to a piece of code that reads a shared (perhaps global) variable and adds the value to a local variable, since the learner may arrive at a working solution even while holding the belief that local variables are shared. The learner may also get the question wrong due to weakness with scoping, not a misunderstanding of how to add concurrency controls or how concurrency controls work.

To address these barriers, the working group will contribute assessment designs that distinguish among these cases, which will be useful for answering the question: "How does tracing knowledge affect learning of advanced topics, such as data structures, algorithms, and concurrency?", according to the method in the next section. While the group's report will only include theory-based assessment designs and questions, they may be used in later research to answer the motivating research question.

2 METHOD

Based on previous work on difficulties with tracing and advanced topics, we will identify advanced concepts and developmental stages for learning them, which consist of what learners can and cannot do at each stage. These will form the basis for developing assessment questions to distinguish among these stages. For example, in concurrency: 1) finding shared data, 2) locking shared data individually, 3) locking overly large sections of code, and 4) fine-grained locking.

Using our identified advanced concepts, we will analyze their required basic tracing knowledge, and where an imagined student with tracing difficulties would encounter barriers, in order to focus our assessment work on the most affected concepts. Table 1 shows an example of what that analysis may look like. It asks "What can learners do and not do, at different developmental stages of understanding an advanced concept in concurrent programming, with or without pre-requisite tracing knowledge?". Similar analysis will be carried out for data structures and algorithms.

Based on our conceptual analysis, our working group will make assessments that can detect differences between 1) knowing related tracing concepts without knowing the advanced concept, 2) knowing both, and 3) knowing the advanced concept but misapplying it due to weak tracing knowledge. One way to do this is to create questions with distractor answers that correspond to particular knowledge combinations, as shown in Table 1. For example, we may decompose the problematic assessment question in the introduction into 3 separate questions, similar to what was done by Strömbäck et al [9]. First, one on identifying shared data (which depends on tracing knowledge). Second, a question about what kind of controls shared data needs to have for concurrency to work correctly. Lastly, a question on modifying a given code example to work, by adding concurrency controls, with distractors that correspond to different combinations of tracing and advanced knowledge.

REFERENCES

[1] Holger Danielsiek, Wolfgang Paul, and Jan Vahrenhold. 2012. Detecting and Understanding Students ' Misconceptions. *SIGCSE'12* (2012), 21–26.

[2] Kathi Fisler, Shriram Krishnamurthi, and Preston Tunnell Wilson. 2017. Assessing and teaching scope, mutation, and aliasing in upper-level undergraduates. *ITiCSE* (2017), 213–218. https://doi.org/10.1145/3017680.3017777

[3] Sally Hamouda, Stephen H Edwards, Hicham G Elmongui, Jeremy V Ernst, and Clifford A Shaffer. 2017. A basic recursion concept inventory. *Computer Science Education* 27, 2 (2017), 121–148. https://doi.org/10.1080/08993408.2017.1414728

[4] Kuba Karpierz and Steven A. Wolfman. 2014. Misconceptions and concept inventory questions for binary search trees and hash tables. *SIGCSE* (2014), 109–114. https://doi.org/10.1145/2538862.2538902

[5] Raymond Lister, Elizabeth S. Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, and et al. 2004. *A Multi-national Study of Reading and Tracing Skills in Novice Programmers.* ACM, 119–150. https://doi.org/10.1145/1044550.1041673

[6] Andrew Luxton-Reilly, Jacqueline Whalley, Brett A. Becker, Yingjun Cao, Roger McDermott, Claudio Mirolo, Andreas Mühling, Andrew Petersen, Kate Sanders, and Simon. 2017. Developing Assessments to Determine Mastery of Programming Fundamentals. In *ITiCSE-WGR '17*. ACM Press, 47–69. https://doi.org/10.1145/3174781.3174784

[7] Greg L. Nelson, Andrew Hu, Benjamin Xie, and Amy J. Ko. 2019. Towards validity for a formative assessment for language-specific program tracing skills. *Koli Calling* (2019). https://doi.org/10.1145/3364510.3364525

[8] Leo Porter, Daniel Zingaro, Soohyun Nam Liao, Cynthia Taylor, Kevin C Webb, Cynthia Lee, and Michael Clancy. 2019. BDSI: A Validated Concept Inventory for Basic Data Structures Leo. In *ICER '19*. ACM Press, 111–119. https://doi.org/10.1145/3291279.3339404

[9] Filip Strömbäck, Linda Mannila, Mikael Asplund, and Mariam Kamkar. 2019. A student's view of concurrency – A study of common mistakes in introductory courses on concurrency. *ICER* (2019), 229–237. https://doi.org/10.1145/3291279.3339415

[10] Jan Vahrenhold and Wolfgang Paul. 2014. Developing and validating test items for first-year computer science courses. *Computer Science Education* 24, 4 (2014), 304–333. https://doi.org/10.1080/08993408.2014.970782

[11] Sander Valstar, William G. Griswold, and Leo Porter. 2019. The relationship between prerequisite proficiency and student performance in an upper-division computing course. *SIGCSE* (2019), 794–800. https://doi.org/10.1145/3287324.3287419