PRIME Technical Report 1

The Development and Validation of the Computer Science Concepts Assessment for Undergraduate Students (UG-CSCA): Preliminary Results

July 2020

Arif Rachmatullah

Bita Akram

Danielle Boulden

Eric Wiebe

The William and Ida Friday Institute for Educational Innovation
North Carolina State University
Raleigh, N.C.

NC STATE

College of Education
Friday Institute
for Educational Innovation

Background, Purposes and the Intended Uses of the Assessment

The Undergraduate Computer Science Concept Assessment (UG-CSCA) is intended to assess STEM undergraduate students' understanding of basic computer science and programming concepts – variables, conditionals, loops, and algorithms. The validation process of this assessment was guided and informed by a Focal Knowledge, Skills, and Abilities (FKSAs) Framework proposed by Grover and Basu (2017) and the K-12 CS Framework (K–12 Computer Science Framework, 2016). Block-based programming is used as the context for each item in the UG-CSCA. Several studies suggest that block-based programming is effective and appropriate computer programming for novices (Grover, Pea, & Cooper, 2015; Weintrop & Wilensky, 2015), and thus aligns with the intention of this assessment.

The current version of the UG-CSCA was written for undergraduate students who are novices in computer science and programming. We believe that this assessment will be useful to instructors who teach introductory computer science and programming courses, as well as computer science education researchers. The instrument was designed for use in pre-intervention-post or longitudinal contexts, as well as for a diagnostic tool. We suggest providing 30-35 minutes for students to complete the assessment which consists of 26 multiple-choice questions.

The Process of the Development of the Assessment

The UG-CSCA was developed based on the FKSAs framework (Grover & Basu, 2017) and our prior validation study on the Middle Grades Computer Science Concepts Assessment (MG-CSCA, Rachmatullah et al., 2020). A total of 30 multiple choice questions were used for the initial version of the UG-CSCA. Ten of these items were taken from the MG-CSCA that we used as a baseline to develop the 20 additional questions. The MG-CSCA items were scored as the most difficult for middle grades students and used with the assumption that those items would then anchor the easy end of the difficulty scale for undergraduate students. Next, 20 new items were created with a focus on developing medium and hard-level questions to create a wider spread of difficulty levels within the instrument.

The validation process was conducted in three phases. First, the original 30 items were piloted with 74 STEM undergraduate students enrolling in introductory computer programming classes. We used this data to determine the range of item difficulty and to identify possible problematic items. We then conducted cognitive interviews with six of these students to focus on their cognitive processes in solving the problematic items and their interpretation of the directions. These interviews were then used to revise the problematic items. In the final phase, after the problematic items had been revised, we then administered the refined version of the UG-CSCA to 594 undergraduate students enrolled in one of two different introductory classes offered over four semesters (Spring 2019, Summer 2019, Fall 2019, and Spring 2020). This data set was used to validate the assessment further.

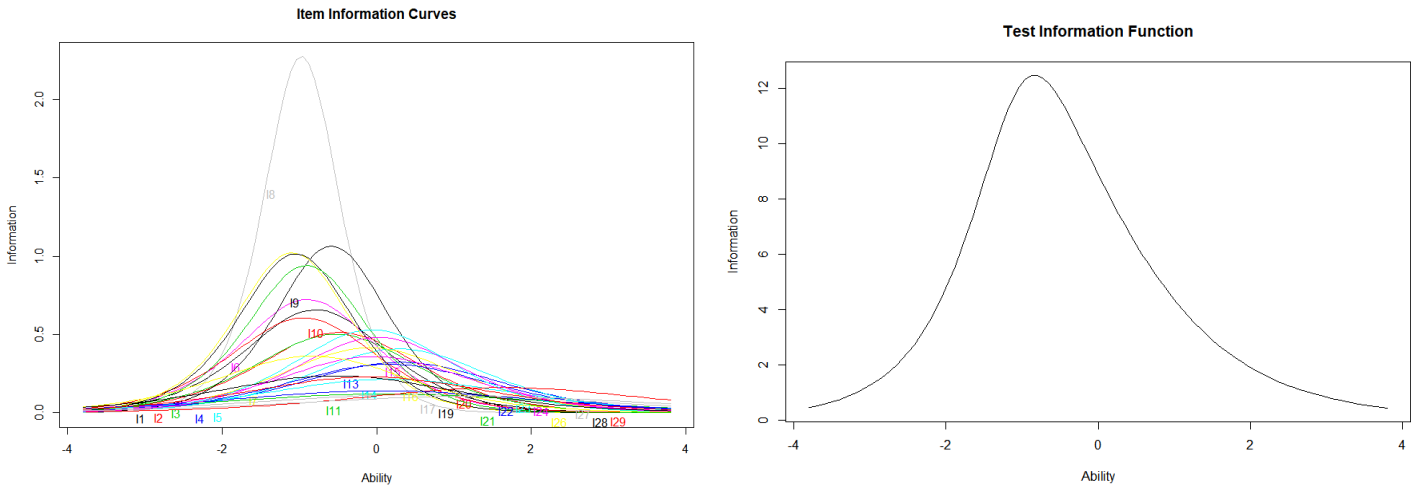
The combination of classical test theory (CTT) and Item Response Theory — 2 parameters logistic (IRT-2PL) — was used to validate the revised version of the assessment. Based on these analyses, we found four problematic items. We removed these four items, resulting in a total of 26 items remaining in the final version of UG-CSCA.

Evidence of Construct Validity and Reliability

The IRT-2PL and Cronbach's alpha were used to gather evidence of construct validity and reliability. We assumed that the assessment is unidimensional, measuring only one construct – conceptual understanding of computer science. IRT-2PL was run first. IRT-2PL is a statistical method that can be used to predict students' abilities in a particular construct based on two parameters of the items, namely difficulty and discrimination. Difficulty refers to how hard the question is for students to answer it correctly, and discrimination refers to how well the items differentiate low and high ability students (Baker & Kim, 2017). Baker and Kim (2017) suggest prioritizing the discrimination parameter because this parameter provides rich information about the item itself. The decisions on whether to retain or remove the questions are based on discrimination values, and we used the value of > 0.65 as the cutoff (Baker & Kim, 2017). We then ran a reliability test on the remaining items using Cronbach's alpha. The IRT-2PL was performed using the "ltm" package (Rizopoulos, 2006) in R Studio (RStudio Team, 2018), and the reliability test was run in IBM SPSS 26 (IBM Corp, 2019).

Based on the analyses, we found four items that had discrimination values less than 0.65, ranging from 0.24 to 0.59. These items, I12, I18, I24, and Item_30, were removed and the analysis re-run. The second round of the analysis showed that all the remaining items had discrimination values greater than the cutoffs, indicating that those items can moderately differentiate low and high ability students. Table 1 presents all the difficulty and discrimination values for these items.

Figure 1. *Item Information Curves (ICC) and Test Information Function*



We also used Item Information Curves (ICC) to investigate the items that provide more information about the latent ability. The higher the discrimination value an item has, the higher information that item has regarding the latent ability. Figure 1a shows the ICC for the final items set for the UG-CSCA. For example, based on Figure 1a, I8 provides the most information for the lower ability students, around θ = -1, but almost no information for high ability students. On the contrary, I11 had lower discrimination value but covered a broader range of abilities. Test Information Function visualized in Figure 1b shows that the assessment provides the most information for slightly-lower-than average students, around θ = -1. Even though the assessment consisted of a broad variety of discrimination values, the assessment still lacks items with high information on high ability students. We plan on developing additional difficult items for the next version of the UG-CSCA.

For reliability, we found that the final version of UG-CSCA had a Cronbach's alpha value of .868, indicating a satisfactory value (DeVellis, 2017).

Table 1. *Psychometric properties of the UG-CSCA*

Item	Percentage correct (%)	Difficulty	Discrimination	Apha if Item Deleted
I1	68.8	-0.79	1.62	.861
I2	60.4	-0.48	1.43	.861
I3	61.4	-0.53	1.42	.863
I4	41.9	0.31	1.14	.863
I5	41.7	0.29	1.28	.862
I6	71.6	-0.90	1.70	.863
I7	50.9	-0.09	1.29	.861
I8	77.0	-0.97	3.02	.861
I9	64.9	-0.58	2.06	.861
I10	72.0	-0.95	1.56	.864
I11	26.3	0.14	0.70	.868
I13	45.0	0.16	1.11	.866
I14	45.7	0.17	0.93	.867
I15	49.5	-0.04	1.20	.862
I16	77.7	-1.09	2.02	.864
I17	39.5	0.65	0.69	.868
I19	76.6	-1.05	2.01	.861
I20	12.9	1.64	0.80	.866
I21	73.2	-0.91	1.94	.863
I22	48.0	0.06	0.75	.867
I23	49.9	-0.06	1.46	.863
I24	12.3	0.02	1.39	.862
I26	67.2	-0.83	1.20	.863
I27	17.3	1.27	0.65	.868
I28	58.8	-0.53	0.97	.865
I29	50.6	-0.09	0.96	.866

References

Bond, T. G., & Fox, C. M. (2013). *Applying the Rasch model: Fundamental measurement in the human sciences*. New York, NY: Psychology Press.

Baker, F. B., & Kim, S. H. (2017). The basics of item response theory using R. Berlin: Springer.

DeVellis, R. F. (2017). Scale development: Theory and applications (4th ed.). Los Angeles, CA: Sage.

Grover, S., & Basu, S. (2017, March). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and Boolean logic. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 267-272). ACM.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. Computer Science Education, 25(2), 199-237. https://doi.org/10.1080/08993408.2015.1033142

IBM Corp. (2019). IBM SPSS Statistics for Windows, Version 26.0 [Computer Program]. Armonk, NY: IBM Corp.

K–12 Computer Science Framework. (2016). K–12 Computer Science Framework. Retrieved from http://www.k12cs.org.

Rachmatullah, A., Akram, B., Boulden, D., Mott, B., Boyer, K., Lester, J., Wiebe, E. (2020). Development and Validation of the Middle Grades Computer Science Concept Inventory (MG-CSCI) Assessment. Eurasia Journal of Mathematics, Science and Technology Education, 16(5). https://doi.org/10.29333/ejmste/116600

Rizopoulos, D. (2006). ltm: An R package for latent variable modeling and item response theory analyses. Journal of Statistical Software, 17(5), 1-25.

RStudio Team (2018). RStudio: Integrated Development for R [Software]. Boston, MA: RStudio, Inc.

Weintrop, D., & Wilensky, U. (2015, July). Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs. In ICER (Vol. 15, pp. 101-110).