Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment

Miranda C. Parker, Mark Guzdial
Georgia Institute of Technology
85 5th Street NW
Atlanta, GA 30308
miranda.parker@gatech.edu,
guzdial@cc.gatech.edu

Shelly Engleman
SageFox Consulting Group
675 Seminole Ave NE Suite 303
Atlanta, GA 30307
+1-404-633-9005
sengelman@sagefoxgroup.com

ABSTRACT

Computing education lags other discipline-based education research in the number and range of validated assessments available to the research community. Validated assessments are important for researchers to reduce experimental error due to flawed assessments and to allow for comparisons between different experiments. Although the need is great, building assessments from scratch is difficult. Once an assessment is built, it's important to be able to replicate it, in order to address problems within it, or to extend it. We developed the Second CS1 Assessment (SCS1) as an isomorphic version of a previously validated language-independent assessment for introductory computer science, the FCS1. Replicating the FCS1 is important to enable free use by a broader research community. This paper is documentation of our process for replicating an existing validated assessment and validating the success of our replication. We present initial use of SCS1 by other research groups, to serve as examples of where it might be used in the future. SCS1 is useful for researchers, but care must be taken to avoid undermining the validity argument.

CCS Concepts

• **Social and Professional topics** ➔ **Student assessment**

Keywords

Assessment; CS1; validity; replication

1. ROLE OF VALIDATED ASSESSMENTS IN EDUCATION RESEARCH

All discipline-based education research communities need assessments of learning. At least some of these assessments should be validated to create an argument that they are actually measuring the concepts they are intended to measure. If the community accepts the argument, the instrument can be a useful tool for the research community.

We use *assessments* to mean instruments and methods for evaluating and documenting the nature, quality, or ability of students [1]. In our context, assessments are used to measure a student's understanding, learning, or ability within a course or subject area. We will focus on multiple-choice assessments of learning made of a series of questions, composed of a stem followed by response options [18]. The stem is the question, which may also be referred to as an item. The response options are comprised of correct and incorrect options, and the latter can also be referred to as distractors or foils. Assessments are abundant in discipline-based education fields older than computing, such as physics, mathematics, and engineering education [15, 19]. However, computing education has few validated assessments [41].

Concept inventories are a class of assessment instruments with some of the features discussed above. They are standardized, multiple-choice, assessment tools to identify misconceptions, investigate learning, measure student understanding of core concepts, and, if desired, ascertain the pedagogical impact on the student's achievement towards expert-level thinking [1, 10, 18]. Concept inventories are not final exams, but include broader topics more central to the subject and can inform instructional or curricular changes [34]. Advanced topics in computing have concept inventories, including algorithms and architecture [16, 26, 28]. Similar to the broader area of assessments, CS1 has few concept inventories.

1.1 History of Assessments and Concept Inventories

We briefly summarize here the history of standardized exams. Standardized, objective tests of learning originated in the early 1900s [30]. Previously, most learning assessments were essay-based. Objective testing presented a more efficient and reliable way to test a student's learning than to have (for example) essay questions that need to be interpreted and which different graders might interpret differently. The first large-scale use of objective tests was the Pennsylvania Study in 1928, which tested no less than 70 percent of all Pennsylvanian college seniors and approximately 75 percent of Pennsylvanian high school seniors [30]. Besides being 12 hours and 3,200 questions long, what marked the Pennsylvania Study was the focus on *learning* rather than *achievement*. Students were tested in their senior year of high school, and then again in their sophomore and senior years of college. The distinction between *achievement assessment* and *assessment of learning* is important, as achievement is the

ICER '16, September 08-12, 2016, Melbourne, VIC, Australia
© 2016 ACM. ISBN 978-1-4503-4449-4/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2960310.2960316

accumulation of learning up to a certain time and learning is a change in student's knowledge over time [30]. For example, achievement assessments such as assignments, quizzes, and exams are not in and of themselves measurements of learning. Rather, they serve as a method to evaluate performance in the form of a grade for a course. Also, these instruments are not validated and tend to be based on the instructor's perception of the subject [1]. On the other hand, assessments of learning are integral parts of course design. These learning instruments allow instructors to reflect on what learning outcomes they want for their students and also decide which assessment devices might be best suited for those outcomes [32].

Validated, broadly applicable learning assessment instruments are essential for growth of the research discipline. Validation is an argument that the assessment measures what it purports to measure [36]. Validated assessment instruments can be used, for example, to compare different instructional approaches [10]. They can be used to measure a student's understanding of the material, which may not be reflected in their grades, since grades could be influenced by factors other than learning, e.g., class attendance. With the recent proliferation of introductory computing curricula across the globe, validated assessment instruments are needed to explicitly compare approaches for learning.

Concept inventories, as a type of validated assessment, were first developed in the 1980s within the physics education community, beginning with the Force Concept Inventory (FCI) [15, 19]. This concept inventory assessed students' conceptions of Newtonian physics. The FCI was used in understanding and identifying where students and instructors conceptions differed [34]. Even conventionally high performing students failed simple conceptual FCI questions, which is indicative of the difference between concept inventories and final exams. The FCI was validated and shown to be a reliable assessment by a study with 6,000 students and their scores on the FCI [12]. The results of using the FCI helped support the shift within the physics community to teach with a more active, student-engagement approach rather than lecture-based teaching.

The FCI showed that concept inventories can be effective instruments for supporting education reform [41]. When used with valid and reliable assessments, concept inventories can help match instruction to what the students need to learn [10]. This is achieved by providing a before and after view of student learning, based on different teaching strategies. Additionally, concept inventories can evaluate student understanding relative to the goals of the course, further promoting the use of assessments in curriculum development [1]. All of the benefits of a concept inventory are not held to a specific class or institution. Concept inventories can compare students' learning outcomes across instructors, institutions, curricula, and pedagogical practices [34].

Concept inventories as assessments of learning have been adapted for and promoted change within STEM fields beyond just physics and the FCI. However, within computing, concept inventories are in their infancy. There are validated concept inventories in the subfields of digital logic and discrete math [1, 14]. The AP CS A exam is a prime example of a valid concept inventory in computing that is used across institutions [34]. However, it is not useful as a research instrument because its questions were not designed to test individual learning objectives or concepts, but instead to achieve psychometric goals of having a normal distribution of student scores [35]. Core concepts in different computing areas have been identified, as well as common misconceptions [34], but there are not many valid concept inventories for CS1.

1.2 Motivation for Replication

In 2010, Allison Elliott Tew created the first validated language-independent content knowledge assessment for introductory computer science at an undergraduate level (a course commonly referred to as "CS1"), the Foundational CS1 Assessment (FCS1) [35, 38]. Tew followed an intensive procedure for the creation of FCS1. First, she defined a minimal content for the course by doing an analysis of the most popular CS1 textbooks [37]. She developed a test specification that was reviewed by a panel of experts [37]. She defined a pseudocode language to be used, and then generated four isomorphic tests: One in her pseudocode, and one in each of MATLAB, Python, and Java. She piloted the questions in an open-ended format, to test readability and generate distractors [38]. Finally, she validated the whole assessment by having participants take two tests (counter-balanced) one week apart: one in their "native" CS1 language, and one in the pseudocode.

The FCS1 was validated through a multi-step argument including expert panel review of the content, large-scale comparison between the FCS1 and language-dependent isomorphic tests, and comparison between performance on the FCS1 and on the students' final CS1 exams. As previously noted, there is a distinct difference between concept inventories and course exams. However, the course exams can be used as a part of a validity argument, and the argument is made stronger by different pieces of evidence working together.

FCS1 has been used by the computing education research community in various contexts, and has been useful in providing insights into student learning. For example, the assessment was used by the ITiCSE 2013 McCracken Working Group in comparison with a novice programmer's ability to solve a practical programming task [39]. The FCS1 was used to measure students' performance, which was then compared to teachers' expectations of the students' performance. The group found that teachers' expectations did not match student performance, and tended to be too optimistic in comparison with the anticipated score. The McCracken Working Group use of FCS1 and their results serve as an example of using FCS1 to answer a research question.

In the case of multiple-question assessments, such as the FCS1, an openly accessible assessment can easily reach a point of *saturation*. We define *saturation* in this context to be where the test or its answers could be easily found (e.g., with an Internet search engine), reducing the effectiveness of the exam. If answers can be found and memorized (or looked up dynamically during the test), then the test is measuring memorization or ability, not understanding. Saturation would weaken the argument for validity of the assessment. If there is a dearth of learning assessments, as in computer science education research, saturation might leave the community without a valid way of measuring learning. However, the existence of multiple assessments dampens the negative consequences of an assessment reaching saturation. As a community, we need to replicate valid knowledge assessments to prevent loss of information and research potential due to saturation.

In order to avoid saturation of the assessment, only a few people have had access to FCS1. For this paper, one author helped with the development of FCS1 and thus had legal access to the assessment. The FCS1 remains the intellectual property of the

Given the following code segment.

```
array = [5, 2, 1, 3, 4, 6, 0, 8, 9]
i = 0
even = 0
WHILE (i < length(array)) AND (array[i] != 0)
DO
        IF (array[i] % 2) == 0 THEN
                even = even + 1
        ENDIF
                i = i + 1
        ENDWHILE
```

What are the values of the variables `i` and `even` after the while loop completes its execution?

```
                A.      i = 1; even = 0
                B.      i = 5; even = 3
                C.      i = 5; even = 4
                D.      i = 6; even = 3
                E.      i = 6; even = 4
```

Given the following code segment.

```
array = [3, 6, 8, 1, 2, 0, 7, 2, 9]
i = 0
odd = 0
WHILE (i < length(array)) AND (array[i] != 0)
DO
        IF (array[i] % 2) == 1 THEN
                odd = odd + 1
        ENDIF
                i = i + 1
        ENDWHILE
```

What are the values of the variables `i` and `odd` after the while loop completes its execution?

```
                A.      i = 1; odd = 0
                B.      i = 5; odd = 2
                C.      i = 5; odd = 3
                D.      i = 8; odd = 4
                E.      i = 8; odd = 5
```

Figure 1. Example of an isomorphic mapping. The top box includes the original question from FCS1, and the bottom box has the isomorphic mapping we created for SCS1. The content area of indefinite loops and question style of tracing were maintained, but the problem and variables were altered.

original author, and we do not have permission to distribute it. We developed a process to iterate on the FCS1 questions to create the Second CS1 Assessment (SCS1), which we followed by a validation process.

Tew and Dorn explained the importance of validated assessments for computing education and described the process of developing two validated assessments [36]. We build on their work to highlight the importance of replicating those assessments, present a process for replicating and validating, and start a discussion for how validated assessments can be used most effectively for our community. The more assessments there are—made by replicating previously validated assessments—the more chances the community has to measure learning gains accurately and to determine the effectiveness of teaching approaches. By creating the SCS1, we have made a language-independent, validated measure of CS1 learning more accessible. Uses of the SCS1 exemplify how it might be used in making research arguments, and where the validity argument for SCS1 is undermined.

Table 1. Areas considered when creating an isomorphic mapping of FCS1.

Isomorphic area	Area Choices
Content areas	• fundamentals • logical operators • conditionals • definite loops • indefinite loops • arrays • function/method parameters • function/method return values • recursion • object oriented basics
Question Types	• definitional • tracing • code completion

2. REPLICATING A VALID ASSESSMENT

We created an isomorphic version of the FCS1 Assessment to create the SCS1 Assessment. Our methods are described here so that they may be used in future replications of valid content knowledge assessments. Like FCS1, SCS1 was designed to measure understanding of introductory computer science concepts at the undergraduate level in a language-independent manner. SCS1 problems are all in a pseudocode language invented by Tew [35] which has been used successfully in other instruments [17]. The replication process is the same as in any education domain [3] but we know that education findings do not always map directly from one domain to another, especially in computing [23]. Therefore it is worth demonstrating the effectiveness of an approach in a new domain. This section details our application of proven replication techniques to a computer science assessment, FCS1.

We created the questions for SCS1 by creating isomorphic copies of the questions in the FCS1. An isomorphic question is created by maintaining the content area for a question as well as the style used to ask the question, but altering the word problem, variables, and answer choices (see examples in Figure 1). The original FCS1 covered nine content areas with three different question types (described in Table 1). Each question in the SCS1 maps to a question in the FCS1 with the same content area and question style, but with different question text and distractors. The difficulty level of each question was not directly addressed when creating an isomorphic mapping, but was assessed in our validation studies outlined in Section 4.

We created isomorphic copies instead of writing completely new questions in order to maintain the validity of FCS1 for SCS1. When replicating valid content knowledge assessments, it can be wise to make small, iterative changes to strengthen the validity argument of the replicated assessment. Arguments for validity are addressed in Section 3.

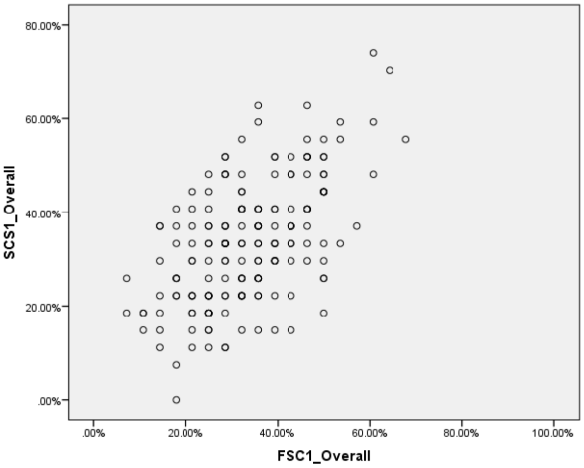

Figure 2. Scatterplot of scores for correlation of FCS1 and SCS1

2.1 Think-Aloud Interviews

While the underlying content remained the same, we changed the questions to create SCS1. The resulting questions might not be read as we intended. We might also have made mistakes in generating distractors. It was important to hear people interpret the questions and answers to ensure that students understood the assessment as we intended. One author conducted these interviews and took notes throughout. The notes of the interview were discussed with another author to analyze where issues were occurring and what needed to change.

We interviewed three students with think-aloud interviews. These students were from our target population: undergraduate students enrolled in introductory computing classes. These interviews were 90 minutes in length, which gave the participants enough time to finish approximately half of the exam. The interviews consisted of a student reading and solving each question aloud, with occasional questioning by the interviewer. The first two students completed the first and second half of SCS1 respectively. The third student was given questions that the first two interviews indicated needed further review, in order of criticality (e.g., problems that needed the most changes after an intervew were more critical to review in the next interview). As anticipated, there were typos (the use of the wrong word in a problem) and wording issues (such as vague pronouns, or subject-verb agreements) where the questions were not clear. One question was found to be too easy during the think-aloud interviews so the problem and answers were changed accordingly to increase the difficulty level. All other errors found in the assessment were "sanity check" changes, where the think-aloud interviews revealed that a question might have no right answer, multiple right answers, or nonsensical answers.

2.2 Validation Study

After the test was created, we set out to show construct validity. That is, we wanted to show that the SCS1 measured what it was intended to measure. Tew has created a construct validity argument for FCS1. We aimed to show that SCS1 had construct validity by showing that SCS1 measured the same content as FCS1. Validity arguments are not transitive, so we cannot claim that SCS1 would correlate (for example) with CS1 final exam scores. FCS1 did, but we did not explicitly validate that SCS1 measured the same constructs as the CS1 final exam scores. We have a more limited validity argument, as is typical for an instrument replication study.

Table 2. Overall correlation between FCS1 and SCS1

Course	FCS1	
	Pearson's Correlation Coefficient	p-value
All (n=183)	0.566	p=0.000**

Table 3. Overall correlations by course, where 1301 is traditional CS in Python, 1315 is Computational Media in Python, and 1371 is CS for Engineers in MATLAB

Course Descriptions	FCS1	
	Pearson's Correlation Coefficient	p-value
Python for CS Majors (n=140)	0.483	*p* = 0.000**
Media Computation (n=30)	0.298	*p* = 0.110
MATLAB for Engineering Majors(n=13)	0.509	*p* = 0.076

To validate the SCS1 against the FCS1, we administered FCS1 and SCS1 to a group of students (n=183), one week apart and counterbalanced. Half of the group took the FCS1 in Week 1 and the SCS1 in Week 2, where the other half of the group took the SCS1 in Week 1 and the FCS1 in Week 2. These students took the assessments near the end of an introductory computing course. There were three courses from which students were recruited—two courses taught CS using Python and one course using MATLAB. The three courses had different computer science emphases. One of the Python courses was a fairly common CS1 approach (e.g., using the *How to Think Like a Computer Scientist* text [6]). The second Python course used a media computation context [11]. The MATLAB course was focused on engineering problem solving. The range of kinds of classes is desirable when validating an assessment of this type.

Students were given one hour to take the exam. Each student was provided with a pseudocode overview to use as reference throughout the assessment. Students were compensated for their participation according to their instructor's discretion, though typically they were given some form of extra credit. Their participation in the study served as practice for their final exam, which was a couple of weeks away from being administered at the time of the study.

It should be noted that the two groups did not take identical versions of the SCS1 Assessments. There was a slight change to one of the questions on the SCS1 between the administrations of the exams due to a typo that was not found during the think-aloud interviews. Analysis on the effect of the typo can be found in the next section.

3. VALIDATING A REPLICATION

A replicated assessment is validated to ensure that the assessment is measuring what the creator or user thinks it is measuring. The use of a non-validated assessment could result in incorrect inferences being drawn about learning and knowledge. The replication can be validated against the original, validated assessment.

In this section, we present our detailed argument for the *construct* and *content* validity of the SCS1. *Construct validity* considers the extent to which performance on an assessment can be interpreted in terms of one or more constructs. *Content validity* considers the extent to which assessment questions provide an adequate and appropriate sample of the domain tasks. We maintained *content validity* from the FCS1 by constructing questions on the exact same content. We only need to argue for *construct validity,* as that is the only thing that changed in the new instrument. Construct and content validity work together to provide evidence that an assessment is working as intended. Since construct validation is dependent on inferences drawn from a variety of data [3, 22], we present a quantitative analysis of the questions on the SCS1 assessment, as well as discuss the correlation between the scores on the FCS1 and SCS1 assessments.

It should be noted that the counter-balanced structure of our validation study could have resulted in unintended priming effects [7]. However, we dismiss the possibility of a priming effect by considering that the students were already primed for the first test by being enrolled in an introductory computing course for almost a full semester before taking our test [27]. In addition, as the tests served as practice final exams, students were motivated to perform on both tests. While it is possible that students explicitly studied concepts they were unfamiliar with on the first test, it is likely that they studied most content from the course in preparation for their final exams during this period.

3.1 Correlation with FCS1

An important step in validation is to correlate scores on the new assessment with other measures of CS1 learning, such as a previously validated assessment. Concurrent validity describes the characteristic of an assessment correlating with a previously validated assessment [22]. Correlating the scores on the FCS1 assessment with final exam scores showed concurrent validity with FCS1. In the development of SCS1, we did not access final exam scores, and so we do not have a direct relation between SCS1 and final exam or course scores. Instead, Pearson's correlation analysis was used to investigate whether student scores on the SCS1 can be positively correlated with their scores on the FCS1.

A Pearson correlation coefficient was computed to assess the relationship between the score on the FCS1 and SCS1 Assessments (see Table 2). There was a strong positive correlation between the two variables, Pearson's $r(183) = .566$, $p = 0.000$. A scatterplot summarizes the results in Figure 2. This is key to our argument that SCS1 is a validated replication of the FCS1.

After finding a positive correlation between the FCS1 and SCS1 scores, we analyzed the correlation based on the course the student participant was enrolled in. Pearson correlation coefficients were computed to assess the relationship between the course and the scores on the FCS1 and SCS1 Assessments (see Table 3). There was a strong, positive correlation for one course, Pearson's $r(140) = .483$, $p = 0.000$. This course was taught in Python and is the one required of computer science majors. The other two courses did not show statistically significant correlations between the scores. The lack of statistically significant correlations by course means that SCS1 is valid for CS1 students *in general*, but may be less accurate for subpopulations. If a CS1 course does not cover object-oriented programming or recursion, for example, it would not match the minimal model that Tew defined, and SCS1 (and FCS1) would probably be less accurate for measuring knowledge for the specific course. The course differences do suggest the need for further research to understand where the SCS1 is most accurate and where it is not.

One possible explanation for the lack of statistical significance is the number of participants in the two classes, both of which are significantly smaller than the computer science majors' class. Overall, our number of participants is much smaller than Tew's. It is possible that the classes could reach statistical significance if more students were sampled from the two classes. Another possible explanation is that the students in those sections without statistical significance were for non-computer science majors. Tew found that correlation with the pseudocode test was strongest for the higher-performing students, and weaker for lower-performing students [35]. As our IRT analysis (see Section 3.2) suggests, SCS1 shows better discrimination among student participants with higher ability. The computer science majors may have greater internal motivation for learning for the subject and would be more successful at demonstrating their knowledge on an unfamiliar assessment.

Our results with SCS1 may be pointing to possible limitations of any pseudocode-based assessment. A pseudocode test may always bias in favor of students with greater understanding. We know that greater knowledge results in better transfer of cognitive skill [31]. High performance on a pseudocode test demands greater knowledge of the original material for the student to successfully transfer knowledge to pseudocode.

Overall, our results demonstrate a strong positive correlation between the scores on the SCS1 and FCS1 assessments. In addition there is a strong positive correlation between the assessments for traditional approaches to CS1 taught in Python. This suggests that the SCS1 has concurrent validity as it corresponds to an established measure, the FCS1.

3.2 Quantitative Analysis using IRT

The data gathered during our study provides a quantitative argument towards construct validity. Item response theory (IRT) is an important method for assessing the validity of measurement scales [13]. In particular, IRT measures the *difficulty* and *discrimination* of each question. In this context, *difficulty* measures the percentage of the test-takers that answered a given question correctly. *Discrimination* measures how well a student's performance on a given question predicts their performance on the overall test. If a question has good discrimination then a student that answers that question correctly is very likely to do well on the test. An ideal assessment using a multiple choice format with five options should have a difficulty of 70-74% on the overall assessment [20] and discrimination levels should be in the "good" range. Overall, if the questions have appropriate difficulty and discrimination and the scores between the two assessments are correlated, then the argument can be made that the assessment shows evidence of validity [13].

While the FCS1 and SCS1 assessments are positively correlated, both are considered very difficult assessments. In our IRT analysis of both assessments on one sample of students, the difficulty of problems was skewed towards a "hard" difficulty level where less than fifty percent of students answered a given problem correctly (see Table 4). The majority of problems were "fair" discriminators rather than "good." "Fair" is defined as having a point-biserial correlation of .1 to .3 [8]. Point-biserial correlation is a correlation

Table 4. Item response theory classifications of SCS1 questions.

		Difficulty (0-100%)			
		Hard (0-50%)	Moderate (50-85%)	Easy (85-100%)	Total
Discrimination	Poor (<0.1)	5, 8, 15, 18, 20, 24, 27	--	--	**7 items**
	Fair (0.1-0.3)	4,6,7,9,10,11,12,13,16,17,21,22,25,26	23	--	**15 items**
	Good (>0.3)	14	1, 2, 3, 19	--	**5 items**
	Total	**22 items**	**5 items**	**0 items**	**27 items**

between student performance on an item (right or wrong) and test score. We would like to note that these results differ from what was previously found regarding the FCS1 assessment. More questions were "hard" and "fair" than was determined in the original FCS1 work, though this is to be expected given the different population of participants between studies. Cronbach's alpha is a measurement of reliability, or the internal consistency based on correlations between different items an assessment. A Cronbach's alpha of 0.65 is considered acceptable [8]. For FCS1 Cronbach's alpha was 0.53; for SCS1 Cronbach's alpha was 0.59. The results imply that the internal consistency for these assessments is slightly below an acceptable level. Taken together, this suggests that there is a need to iteratively refine both instruments and re-test using a larger, more diverse sample of students.

As mentioned previously, there was a typo in one question that was found in between test administrations. Upon detection of this typo, we fixed this item to more accurately measure understanding with half of our test population. After analyzing the results, the typo in the question did matter. Students performed worse on the question with the typo (25% of students received the correct answer) than without it (59% of students received the correct answer). This was later than we would have liked to be still catching typos, but the statistical analysis suggests that SCS1 is still considered to have concurrent validity with FCS1.

4. EXAMPLE USES OF SCS1

Validated assessments offer us well-defined yardsticks for comparing populations, e.g., between experimental conditions or over time. We welcome the CS Education research community to use the SCS1 to measure performance or learning gains (e.g., by using the SCS1 as a pre-test and post-test). We include three of the first uses of SCS1 outside of its initial development.

4.1 Measuring Teachers' Knowledge

Our group used the SCS1 to measure knowledge of teachers during a professional development workshop. Assessments are necessary during these sessions to show effectiveness and impact [5]. We asked teachers (n=18) in a weeklong professional development workshop on Computer Science Principles [2] to complete the SCS1. Half of the teachers had taught computer science or programming for two or more years and all of the teachers were teaching CS at the high school and undergraduate level.

The teachers, on average, got one more question correct compared to the students in the validation study. The average student score on SCS1 in our validation study was 9.68 (σ=3.5), or 35% (σ=13.1%) and the average teacher score on SCS1 was 10.72 (σ=6.1) or 39% (σ=22.4%). This is likely the first measurement of a high school CS teacher population with a validated instrument that can be compared to an undergraduate student population.

We have little information about the quality of high school computer science teachers, at least in the United States. What we do know suggests great variability in teacher knowledge, with most teachers we have interviewed saying that they know too little and would like to know more [4, 25]. We don't know enough about the teachers who took the workshop to make any claims about high school CS teacher knowledge more broadly. We do see potential in using SCS1 to compare CS in-service teachers to undergraduate computing students, and in order to consider the relative strengths of computing knowledge learned as a student in a classroom versus as a teacher in a classroom.

4.2 Comparing CS1 Approaches

A team in the Philippines used the SCS1 as an achievement assessment within a pilot class for introductory computing at their institution. They wanted a method to assess progress with the changes they were implementing, such as a shift in programming language used. The Philippine team wanted to use the scores on the SCS1, as well as grades from within the class and past courses, as a way to compare different approaches taken in the introductory course.

The Philippines team's use of SCS1 points to how instructors might use a validated assessment to inform their instructional practices. The single use of the SCS1 does not provide enough information to support any hypotheses. If there were two comparable populations enrolled in two different classes, then the SCS1 might be used to compare the post-class students understanding. If it was used both as a pre-test and post-test, the SCS1 could be used to make an argument about learning, much as how Hake used the FCI [12]. The Philippines' team's use of SCS1 points to potential hypothesis testing in the future.

4.3 Translation and Adaptation

A group in Germany translated and adapted the SCS1 to fit their needs of measuring learning in their CS1 course. We present their story here to highlight the challenges of replicating a validated assessment.

The first stage of the German process was replication, as in our approach described in this paper—a careful creation of an isomorphic test from the SCS1. The research group in Germany translated the SCS1 assessment into German over two versions. The first version was created by translating the SCS1 from English into German, which we will refer to as SCS1-G. SCS1-G was translated back to English (SCS1-E) by a second individual and compared with the original English version. Differences between the SCS1-E and the original SCS1 assessment were discussed and collaboratively adjusted in the German translation when needed. Two more individuals looked at SCS1-G and SCS1-E to look for any differences or inconsistencies, which were primarily in variable names and formatting. The second German version of SCS1 was created after the last round of revisions of the first version, when the two individuals noticed an overall inconsistency in wording of questions. The translation team addressed inconsistencies such as phrases preceding code segments that did not previously precede all code segments. There were also cultural differences that fed into the second German version of SCS1, including how questions were worded and how words are connoted (versus denoted) between the languages.

At this point, the process could continue the way that we validated the SCS1 against the FCS1. SCS1-German could be validated against our SCS1, given enough students who understood both German and English. However, the German team wanted to extend SCS1 to meet their special needs.

The German team was concerned that the students might be able to guess at the answer if the student had even a small amount of previous CS knowledge. They created a sixth answer choice for every multiple-choice question so that students could state, “I am unsure.” This option was present in both German versions of SCS1. Although this addition reduces comparability with the original SCS1, the German team felt the lack of such an answer choice would not be fair to the students and might introduce a lot of false positive answers or blank answers. A blank answer is difficult to interpret – did the student run out of time or just not know the answer?

Even without the extension, we cannot make a validation argument that SCS1-G is equivalent to SCS1 in terms of the constructs it tests. Translation weakens the validity argument. Changing an assessment by adding questions or distractors *always* weakens the validity argument. The German example does point to the need for more validated assessments, like the SCS1. It also serves as an example of how we can use the replication and validation process that we describe in this paper as a template for viewing other attempts, e.g., we can see how the SCS1-G process mapped to our replication process but not the validation process.

5. PARAMETERS OF FUTURE WORK WITH SCS1

As with any validated assessment of learning, the use of SCS1 is complex. It holds significant research potential in serving as a tool to measure student learning and answer hypotheses we could not before. However, SCS1 also has limited support for different use-cases.

5.1 Following Hake’s Lead

The SCS1 assessment can also be used like other validated concept inventories from other subjects, especially following the lead in Physics Education Research. The Force Concept Inventory [15] was used by Hake to measure learning interventions in Physics classrooms, with n=6542 students [12]. Hake found that interactive-engagement methods correlate to better problem-solving abilities. This was an important first step in making the argument for active learning approaches [9].

As with the Force Concept Inventory, the SCS1 might be used with large sample sizes to demonstrate the effectiveness of learning interventions. We could gather metadata from the test takers, such as demographics, tools used in their class, and their perceptions and attitudes towards computing. This metadata and the scores on the assessment could be analyzed for interesting correlations between achievement and the metadata points.

The SCS1 could be used to further the work in discipline-based computing education research by comparing student scores on the assessment in comparable, but different, approaches to introductory computing. For example, we might compare students in and not in interdisciplinary computing courses [21]. Furthermore, the SCS1 might be used to measure differences in courses taught using different programming languages. To make useful comparisons we would need to use SCS1 as a pre-test and a post-test so that we measure learning *gains* and not just achievement, as discussed earlier [11, 33]. We believe that many researchers in the computing education community could productively use the SCS1 to quantitatively measure student understanding, though only on the content areas represented in the test.

5.2 The Fragility of a Validated Assessment

Validation is an argument that an assessment measures what it purports to measure [36]. We have presented an argument here that SCS1 measures knowledge related to CS1 (as defined by [37]) across students who learned CS using Java, MATLAB, and Python. The argument is fragile, though. It does not withstand changes to SCS1.

While portions of the SCS1 focus on different parts of CS1 (e.g., conditionals, assignments, or loops), we cannot easily construct an argument that subsets of the SCS1 are equally valid. We have shown that SCS1 is equivalent to FCS1. We have not shown that portions of SCS1 are equivalent to portions of FCS1. The developers of FCS1 did not show that portions of FCS1 measure portions of CS1 knowledge.

Consider a possible example: the conditional questions in SCS1 only cover *parts* of students’ understanding of conditionals. The gap between what the questions cover and what students understand may not be significant when we are considering the whole test (e.g., the assignment and loops sections may be so effective that they mask the weaknesses in conditionals). However, if the questions were pulled out separately, the gap may make the questions a poor assessment of knowledge of conditionals. Similarly, reporting just the conditionals section when students take the whole SCS1 does not constitute a valid measurement. Although it is easy to look only at how students performed on the conditional questions, the analysis does not represent a valid measurement of the student’s understanding of conditionals since our validation argument only extends to the *whole* test.

Similarly, adding additional questions, additional question choices, or changing the order of questions invalidates the instrument. Additional questions may be more difficult, or may measure something different than CS1 knowledge. Changing the order of questions means that questions are not primed the way that they were when we validated SCS1. For example, if assignment questions came before loops questions, students might do better on loops because they were reminded how variables worked by the assignment questions. Reversing the order might lead to worse performance on the loops questions because students didn’t have the priming effect first. It *might* not make any difference, but we did not test for different orderings, so we cannot make the argument for validity if the ordering is changed.

The SCS1 validity argument suggests a wealth of opportunities for improvement with future iterations of CS1 content knowledge assessments. The SCS1 is the first, albeit fragile, step towards future development of these critical concept inventories.

6. CONCLUSION: A CALL FOR MORE ASSESSMENTS

The SCS1 assessment was replicated from the FCS1 and demonstrates concurrent validity. However, there are many aspects of SCS1 that can be improved to better gauge students' CS1 content knowledge. Due to the replication and validation processes, SCS1 inherits any limitations of the FCS1 assessment. Thus, any issues with the FCS1 still hold true with SCS1. Additionally, the IRT analysis and the Cronbach's alpha suggest that the current SCS1 questions should be improved. That is, additional adjustments should include moderating difficulty level of questions on the SCS1 assessment and improving the discrimination of each question to make them more useful in influencing the overall result on the assessment. Each question can be improved in terms of difficulty and discrimination in order to make an objectively better assessment.

Similar to looking to other fields for information about replication and validation, we can look to other concept inventories and work done to improve them. FCI, although central to the research of concept inventories, is not without its flaws either. Rebello and Zollman have worked on the distractors on the FCI could be improved by gathering open-ended responses to questions and inserting the common responses as answer choices [29]. Additionally, Mühling et al. have created tests of basic programming ability by iterating on the assessment until the assessment had appropriate difficulty [24]. These processes can be used with the SCS1 assessment to create a revised version of the test with revised distractors and until appropriate IRT results are reached.

More assessments could be created using the SCS1 and the process for replication and validation described here. For example, we do not know if the SCS1 is useful for measuring understanding of blocks-based languages. Assessments are beginning to emerge for these languages, but a stronger claim for their validity can be made if scores on these assessments are correlated with SCS1 [40]. A blocks-based assessment could be created for content using the process defined for the FCS1 and SCS1 assessments. Performance on the SCS1 could be correlated with the blocks-based assessment to make an argument for the validity of the blocks-based assessment. Even though students who studied blocks might not perform as well on the pseudocode-based SCS1, the SCS1 could be used to measure understanding separated from the medium students use to program. Weintrop and Wilensky used a similar development process to create their blocks-and-text commutative assessment [40], but did not validate against an existing instrument. The SCS1 could provide that validation comparison.

These recommendations for the future improvement of the SCS1 assessment are not an exhaustive list. The community of computer science educators and researchers will decide which assessment to use, to validate, and to replicate, and we hope that SCS1 plays a role in that work. By engaging the community, the SCS1 may be considered an open source assessment—freely available for use, for replication, and for extension or modification.

As is, the SCS1 assessment can provide insight to differences in instructional approaches, effectiveness of interventions, and how the students and teachers, or different subsets of those groups, differ in their knowledge of CS1. As with any assessment, there are caveats. The argument for validity is fragile, and thus the assessment can only be used as-is or else it needs to be re-validated. As the use and re-validation of this instrument grows, it and its counterparts can provide an important resource for the ICER community.

Information will be made available on how researchers can get access to SCS1 at the presentation of this paper and upon request to the authors.

7. ACKNOWLEDGMENTS

We would like to thank the students who participated in the study and their instructors who graciously gave us the time.

This material is based on work supported by the National Science Foundation under Grant No. 1432300 and the National Science Foundation Graduate Research Fellowship under Grant No. DGC-1148903. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] Almstrum, V.L., Henderson, P.B., Harvey, V., Heeren, C., Marion, W., Riedesel, C., Soh, L.-K. and Tew, A.E. 2006. Concept inventories in computer science for the topic discrete mathematics. *ACM SIGCSE Bulletin*. 38, 4 (2006), 132.

[2] Astrachan, O. and Briggs, A. 2012. The CS Principles Project.

[3] Brennan, R.L. 2006. *Educational measurement*.

[4] Bruckman, A., Biggers, M., Ericson, B., Mcklin, T., Dimond, J., Disalvo, B., Hewner, M., Ni, L. and Yardi, S. 2009. "Georgia Computes !": Improving the Computing Education Pipeline. *Proceedings of the Special Interest Group on Computer Science Education (SIGCSE'09)*. (2009), 86–90.

[5] Cooper, S., Grover, S. and Simon, B. 2014. Building a virtual community of practice for K-12 CS teachers. *Communications of the ACM*. 57, 5 (2014), 39–41.

[6] Downey, A.B. 2014. Think Python: How To Think Like a Computer Scientist. *Green Tea Press Think X series*. June (2014), 300.

[7] Fazio, R.H., Sherman, S.J. and Herr, P.M. 1982. The feature-positive effect in the self-perception process: Does not doing matter as much as doing? *Journal of Personality and Social Psychology*. 42, 3 (1982), 404–411.

[8] Field, A. 2005. *Discovering Statistics Using SPSS*.

[9] Freeman, S., Eddy, S.L., McDonough, M., Smith, M.K., Okoroafor, N., Jordt, H. and Wenderoth, M.P. 2014. Active learning increases student performance in science, engineering, and mathematics. *PNAS Proceedings of the National Academy of Sciences of the United States of America*. 111, 23 (2014), 8410–8415.

[10] Goldman, K., Gross, P., Heeren, C., Herman, G.L., Kaczmarczyk, L., Loui, M.C. and Zilles, C. 2010. Setting the Scope of Concept Inventories for Introductory Computing Subjects. *ACM Transactions on Computing Education*. 10, 2 (2010), 1–29.

[11] Guzdial, M. 2013. Exploring hypotheses about media computation. *Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13* (2013), 19.

[12] Hake, R.R. 1998. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*. 66, 1 (1998), 64.

[13] Hambleton, R.K., Swaminathan, H. and Rogers, H.J. 1991. *Fundamentals of item response theory*.

[14] Herman, G. 2011. *The Development of a Digital Logic Concept Inventory*.

[15] Hestenes, D., Wells, M. and Swackhamer, G. 1992. Force Concept Inventory. *The Physics Teacher*.

[16] Karpierz, K. and Wolfman, S. a. 2014. Misconceptions and concept inventory questions for binary search trees and hash tables. *Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14*. (2014), 109–114.

[17] Lee, M.J. and Ko, A.J. 2015. Comparing the Effectiveness of Online Learning Approaches on CS1 Learning Outcomes. *ICER*. (2015), 237–246.

[18] Libarkin, J. 2008. Concept Inventories in Higher Education Science. *STEM Education Workshop 2*. (2008), 1–13.

[19] Libarkin, J.C. and Anderson, S.W. 2005. Assessment of Learning in Entry-Level Geoscience Courses : Results from the Geoscience Concept Inventory. *Journal of Geoscience Education*. 53, 4 (2005), 394–401.

[20] Lord, F.M. 1952. The relation of the reliability of multiple-choice tests to the distribution of item difficulties. *Psychometrika*.

[21] Magana, A.J., Falk, M.L. and Reese, M.J. 2013. Introducing Discipline-Based Computing in Undergraduate Engineering Education. *ACM Transactions on Computing Education*. 13, 4 (2013), 1–22.

[22] Miller, M.D., Linn, R.L. and Gronlund, N.E. 2012. Validity. *Measurement and assessment in teaching*. Pearson Higher Ed.

[23] Morrison, B.B., Margulieux, L.E. and Guzdial, M. 2015. Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. *ICER*. (2015), 21–29.

[24] Mühling, A., Ruf, A. and Hubwieser, P. 2015. Design and First Results of a Psychometric Test for Measuring Basic Programming Abilities. *WiPSCE '15*. (2015).

[25] Ni, L., Guzdial, M., Tew, A.E., Morrison, B. and Galanos, R. 2011. Building a Community to Support HS CS Teachers: the Disciplinary Commons for Computing Educators. *Proceedings of the 42th ACM technical symposium on Computer Science Education - SIGCSE '11*. (2011), 553–558.

[26] Paul, W. and Vahrenhold, J. 2013. Hunting High and Low: Instruments to Detect Misconceptions Related to Algorithms and Data Structures. *Proceedings of the 44th ACM technical symposium on Computer Science Education - SIGCSE '13*. (2013), 29.

[27] Pollatsek, A. and Well, A.D. 1995. On the use of counterbalanced designs in cognitive research: a suggestion for a better and more powerful analysis. *Journal of Experimental Psychology: Learning, Memory, and Cognition*. 21, 3 (1995), 785–794.

[28] Porter, L., Garcia, S., Tseng, H.-W. and Zingaro, D. 2013. Evaluating student understanding of core concepts in computer architecture. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13*. (2013), 279.

[29] Rebello, N.S. and Zollman, D. a. 2004. The effect of distracters on student performance on the force concept inventory. *American Journal of Physics*. 72, 1 (2004), 116.

[30] Shavelson, R.J. 2007. *A Brief History of Student Learning Assessment: How We Got Where We Are and a Proposal for Where to Go Next*.

[31] Singley, M. and Anderson, J.R. 1989. *The Transfer of Cognitive Skill*. Harvard University Press.

[32] Stefani, L. 2004. Assessment of Student Learning: promoting a scholarly approach. 1 (2004).

[33] Stefik, A. and Siebert, S. 2013. An Empirical Investigation into Programming Language Syntax. *ACM Transactions on Computing Education*. 13, 4 (2013), 1–40.

[34] Taylor, C., Zingaro, D., Porter, L., Webb, K.C., Lee, C.B. and Clancy, M. 2014. Computer science concept inventories: Past and future. *Computer Science Education*. 24, 4 (2014), 253–276.

[35] Tew, A.E. 2010. *Assessing Fundamental Introductory Computing Concept Knowledge in a Language Independent Manner Assessing Fundamental Introductory Computing Concept Knowledge*. Georgia Institute of Technology.

[36] Tew, A.E. and Dorn, B. 2013. The Case for Validated Tools in Computing Education Research. *Computer*. 46, 9 (2013), 60–66.

[37] Tew, A.E. and Guzdial, M. 2010. Developing a validated assessment of fundamental CS1 concepts. *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*. (2010), 97.

[38] Tew, A.E. and Guzdial, M. 2011. The FCS1 : A Language Independent Assessment of CS1 Knowledge. *Proceedings of the 42nd ACM technical symposium on computer science education* (2011), 111–116.

[39] Utting, I., Tew, A.E., McCracken, M., Thomas, L., Bouvier, D., Frye, R., Paterson, J., Caspersen, M., Kolikant, Y.B.-D., Sorva, J. and Wilusz, T. 2013. A Fresh Look at Novice Programmers' Performance and Their Teachers' Expectations. *Proceedings of the {ITiCSE} Working Group Reports Conference on Innovation and Technology in Computer Science Education-working Group Reports*. (2013), 15–32.

[40] Weintrop, D. and Drive, C. 2015. Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. (2015), 101–110.

[41] Yadav, A., Burkhart, D., Moix, D., Snow, E., Bandaru, P. and Clayborn, L. 2015. *Sowing the Seeds: A Landscape Study on Assessment in Secondary Computer Science Education*.