Toward CS1 Content Subscales: A Mixed-Methods Analysis of an Introductory Computing Assessment

MIRANDA C. PARKER, San Diego State University, USA
MATT J. DAVIDSON, University of Washington, USA
YVONNE S. KAO, WestEd, USA
LAUREN E. MARGULIEUX, Georgia State University, USA
ZACHARY R. TIDLER, Georgia Institute of Technology, USA
JAN VAHRENHOLD, University of Münster, Germany

Background and Context. There is a constant, demonstrated need for valid and reliable assessments in computing education research. While there exist assessments at a course-based level (e.g., CS1, Data Structures, Discrete math, etc.), instructors and researchers would also like concept-based subscales that are more fine-grained. However, assessments designed and validated at the course level need additional work to determine whether they can reliably and validly measure individual concepts.

Objectives. In this paper, we explore the content and factor structure of an existing CS1 assessment, the Second CS1 (SCS1) assessment, which consists of nine CS1 concepts and three question types (definitional, code tracing, and code completion). We investigate the underlying structure of the assessment in terms of these concepts and question types to determine whether the assessment could be separated into subscales.

Method. We used a mixed-methods approach to answer our research question. We first investigated the designer-intended subscales of concepts and question types using confirmatory factor analysis, constructing multiple models using data from 547 students. We then qualitatively coded the assessment using an established framework to better understand our findings. Finally, we used a combination of exploratory and confirmatory factor analysis to take a data-driven approach to understanding the underlying factor structure.

Findings. We argue that SCS1 cannot discretely measure student knowledge in terms of the concepts or question types. However, more work is needed before the assessment can be split into reliable, valid, and useful subscales.

Implications. We discuss the future work needed to create a CS1 assessment to fit the needs of the computer science education community; namely, the need to revise and expand SCS1 to more appropriately measure its intended constructs and to be used outside of controlled experiments.

CCS Concepts: • **Social and professional topics → Student assessment**.

Additional Key Words and Phrases: assessment, CS1, subscales, concepts, factor analysis

ACM Reference Format:
Miranda C. Parker, Matt J. Davidson, Yvonne S. Kao, Lauren E. Margulieux, Zachary R. Tidler, and Jan Vahrenhold. 2023. Toward CS1 Content Subscales: A Mixed-Methods Analysis of an Introductory Computing Assessment. In *Koli '23: Koli Calling, November 13–19, 2024, Finland*. ACM, New York, NY, USA, 19 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Assessments are used in education research to evaluate the impact of an intervention, to inform the design of learning activities or curricula, or to simply understand what students know at a point in time. Many modern education research assessments are modeled after the Force Concept Inventory (FCI), which was developed by the physics education community [17]. The FCI has become iconic in part because it is scoped for a single concept: force. However, because a thorough understanding of force is contingent upon understanding force's connection to other concepts, the assessment must be multifaceted. The authors of the FCI designed the instrument to measure six conceptual dimensions that, when combined, represented a student's knowledge of force [17]. However, when this purported dimensionality was tested using exploratory factor analysis, Huffman and Heller found only two distinct dimensions for high school students and one for university students [18]. In fact, the authors explicitly cautioned against decomposing FCI scores into the designed six dimensions, even though the questions seemed distinct enough to warrant division into subscales [18].

This mismatch between assessment design and post-hoc measurement of factor structure holds important takeaways for standardized assessments. Unless explicit verification steps are taken, designers of assessments cannot reliably say what their assessment measures in practice. Put differently, if the factor structure of an assessment is of importance, it needs to be verified to ensure the structure is what was intended for each of the assessment's target populations.

There are a growing number of assessments in computer science education research (CSER) [28], though most of them are focused at the course level, such as introductory computer science (CS1) [14, 32], Data Structures [35], and Discrete Mathematics [1]. Because of this course-focused approach to assessment design, researchers must take additional steps to interpret and validate the assessment of individual concepts, such as conditionals or arrays. As illustrated with FCI, there is no guarantee that these assessments discretely measure the concepts they contain. We sought to engage with this level of analysis on a CSER assessment: the Second CS1 (SCS1) assessment [31].

SCS1 is a pseudocode-based assessment for introductory undergraduate computer science (CS) courses [32] that is commonly used in CSER [31]. Based on the Foundational CS1 (FCS1) assessment [13], SCS1 includes 27 questions designed to cover nine intro-level computing concepts: basics (e.g., variables, assignment), logical operators, conditionals, `for` loops, `while` loops, arrays, function parameters, function return values, and recursion. To measure different aspects of these concepts, SCS1 also contains three styles of items: definitional, code tracing, and code completion.

Despite its broad use, the construct(s) SCS1 actually measures is yet to be explored. Similar to the creators of the FCI, the FCS1 creators had notions of the dimensions that the assessment should measure [14]. However, neither the FCS1 nor SCS1 have been analyzed to verify that the intended dimensionality is reflected in real student responses. Without this analysis, we cannot be confident that SCS1 is robust to adaptations that use a subset of the original items. Precisely understanding what an assessment measures is important for ensuring that conclusions drawn from the use of the assessment are appropriate. Validity is the concept that an instrument measures what we want it to measure. To establish the validity of an assessment for a specific use, we need to collect evidence and establish a chain of reasoning that connects student responses with scores and specific interpretations of those scores. In this paper, we address two types of validity evidence: *content-oriented* evidence and evidence regarding *internal structure* [2].

To develop evidence based on SCS1's *internal structure*, we performed factor analysis on SCS1 data from 547 students to explore what dimensions currently exist in the assessment, using factors designed within the assessment (i.e., concepts and question types). To develop *content-oriented evidence*, we qualitatively analyzed the content of the assessment with expert coders using established computing curricula and conceptual frameworks. Finally, we attempted a data-driven approach to analyzing the items, using exploratory analytical methods to let the underlying structure of SCS1 be determined without preconceived structures being applied. We discuss our findings within the context of existing assessment frameworks and computing education literature. Our findings will help the CSER community understand the implications of the assessments we use and guide future development of introductory CS assessments.

We conducted the above set of analyses to examine whether or not SCS1 can be used as a valid and reliable measure of distinct subconcepts. We asked the following research question to guide our study:

> **RQ:** Does there exist usable and reasonable subscales within SCS1? If so, what SCS1 items align to which subscales?

In this context, we believe that subscales would be usable if they matched dimensions that instructors or researchers use, such as concepts (i.e., CS1 knowledge) or question types (i.e., CS1 skills). Additionally, subscales would be reasonable if a factor analysis resulted in acceptable model fit, fair factor loadings, and appropriate reliability for the subscales.

Terminology. Throughout this paper, we use the terms *assessment* and *instrument* interchangeably. Often in psychometric analyses (i.e., dealing with measurement and assessment), questions in assessments are referred to as *items*. Examinees' performance on an assessment may vary in a pattern that suggests the presence of latent *dimensions*. These dimensions can roughly be thought of as the tendency for performance on some set of test items to covary with one another. If performance on all items of an assessment are strongly covariant with one another, it might be the case that the assessment is unidimensional, and it is often reasonable to assume that all of the items are, therefore, measuring knowledge of the same underlying concept. If performance on subsets of items covary, the assessment might be considered multidimensional, and it is often reasonable to assume that knowledge of any single concept cannot explain differences in performance on the assessment. Sometimes the dimensionality of an assessment is interpretable in a way that warrants the inclusion of *subscales* in the scoring of the assessment. Other times the dimensionality of an assessment may not be easily interpretable. At times, we will use the terms *factor* and *factor structure* to refer to the assessment's individual dimensions and the overall dimensionality respectively.

2 BACKGROUND AND RELATED WORK

Our study builds on prior work on CSER assessments. Here, we summarize the development and use of SCS1 to contextualize the importance of our present analysis. We also provide background on the use of subscales on instruments in computing education, highlighting the lack of subscales on assessments of computing knowledge.

2.1 History of the SCS1 assessment

The first pseudocode-based assessment validated for use with introductory undergraduate CS1 courses was the FCS1 assessment [13]. FCS1 was created by Elliott Tew, who began by surveying common CS1 textbooks to gather a list of common topics covered in CS1 courses, which was then verified by a panel of experts [13]. Elliott Tew used this list of topics to draft three items related to each concept: a definitional question, a code tracing question, and a code completion question [14]. To create multiple-choice options for each item, Elliott Tew drafted open-ended items with code written in the Java, Python, and MATLAB programming languages. Students' responses to these open-ended items were used to formulate distractors that would capture the most common misconceptions. Elliott Tew then translated the assessment into pseudocode. The style of pseudocode used in the FCS1 was created from beginner programmer style guides with minimal syntax, capitalized keywords, and specific end commands to close program blocks [14]. Students were given both the language-specific version of the assessment (according to what language their CS1 course was using) and the pseudocode-based version. Students' final exam scores in their CS1 course were also gathered. Using these data, Elliott Tew argued that the interpretation and use of the scores obtained through this pseudocode-based assessment were valid for undergraduate students at the end of a CS1 course taught in Java, Python, or MATLAB [13].

FCS1 was not broadly disseminated due to concerns that cheating would negatively impact the validity of the assessment as a research instrument. To create an instrument available to the community, SCS1 replicated FCS1 [32] with isomorphic versions of the items developed by altering the context of the problem, the variable names, and the corresponding answer choices. Everything else, including the content and types of items and the ordering of the items, remained the same. FCS1 and SCS1 were administered to students in a counter-balanced study to verify that SCS1 measured CS1 concepts in the same way that FCS1 did and the scores on the two assessments were found to be correlated, $r(183) = 0.57$, $p < 0.01$ [32]. Following publication of this result, SCS1 was released to be used in CSER.

Since its release, SCS1 has been used in research on CS learning and to inform the development of additional valid assessment instruments in CS [31]. Psychometric research has also been done on SCS1, including identifying items that could be improved [49]. Some researchers have adapted SCS1, including shortening it when the original 60-minute version was too cumbersome for research projects [4, 34]. The SCS1 assessment has also been translated into Java, Python, and MATLAB and into other world languages, including Finnish and German [3, 11, 12, 24, 40, 45].

SCS1 has been used by the CSER community in a variety of ways, but there is not always evidence to support the validity of a particular use. All of the aforementioned modifications altered SCS1 in order to use it in a condition that was amenable to the researchers. This demonstrates, in part, why SCS1 is not an ideal tool for studying CS1 performance: it had to be modified before use. When it was published, SCS1 intentionally included items that were difficult for students to correctly answer to avoid ceiling effects in research; 22 of the 27 items had less than 50% of the students answer them correctly [32]. Seven of the items, all of which were identified as "hard," were found to have poor discrimination, indicating that students' performance on those items was not a good predictor of their performance as a whole [32]. Further analysis using item response theory confirmed that some items were especially difficult and may not be assessing the same knowledge as other items on the assessment [49]. Additionally, SCS1 is based on nine concept areas that were identified when Elliott Tew created FCS1 [14]. Not all CS1 courses, then or now, necessarily cover those concept areas. Additionally, some concepts may be outside the scope of what a researcher intends to study. For these reasons, a version of SCS1 that included concept-specific subscales could better serve the CSER community.

2.2 Subscales and their use in Computing Education Instruments

Subscales are pieces of an assessment that are theoretically distinguishable from each other. Depending on the type of construct measured, an instrument could provide one or more subscales designed towards assessing factual knowledge [14], designed towards capturing beliefs and attitudes [10], designed towards measuring traits [27], or any combination thereof [38].

It is an accepted practice to revisit scales and their factor structures, which may depend on underlying traits of the study population and may change over time. For instance, such analyses have been done to (1) confirm factor structures obtained in earlier analyses [5], (2) to understand the effects of alterations to the instrument [21], or (3) to examine whether the instrument's factor structure is invariant for different demographic groups [26]. For instance, Torkzadeh and Koufteros [46] found that the original factor structure of a scale developed by Murphy et al. [30] could be interpreted in such a way that justified breaking a larger subscale into two smaller pieces. In another case, the factor structure of Ramalingam and Wiedenbeck's self-efficacy scale [37] was revisited in the context of different populations and changed course content [8, 42].

Instrument subscales have been used in CSER previously with regards to measuring student attitudes towards computing [10, 48] or self-efficacy [8, 37, 42, 47]. While instruments that intend to assess computing knowledge have been *designed* with subscales in mind, e.g., grouping items by content area tested [14, 50], the verification of these factors often proceeds by assessing the internal structure of the instrument as a whole (e.g., [33]). Other researchers have assessed the model fit of different CSER instruments' scores through methods from item response theory [22, 49].

Most of the work around assessments of knowledge in computing education focuses on the particular class of assessments known as concept inventories [44]. While the name may suggest that concept inventories measure a singular concept, and are thus unidimensional, this is often not the case. Our earlier discussion of the multidimensionality of "force" as it is represented in the Force Concept Inventory exemplifies this discrepancy [18]. The establishment of subscales within SCS1, which already purports to measure knowledge across nine different computing concepts, would be novel with regard to past computing knowledge assessment efforts.

3 RESEARCH PLAN

3.1 Study Data

When SCS1 was released to the community, one way to access it was through a centrally-hosted Qualtrics-based version of the assessment. We conducted a secondary data analysis on the responses to that instance of SCS1. The responses come from multiple independent administrations of SCS1 in a number of different contexts. These data include multiple studies that used SCS1 to do research on specific courses, workshops, or tools. We did not use timing data to filter SCS1 responses, given the unknowns around the context of the various administrations of SCS1. Addressing a possible *statistical regression* threat to internal validity, we only include responses from SCS1 administrations that were *not* a pre-test. Pre-test scores can vary widely due to an uneven distribution of prior experience with CS, in addition to statistical artifacts like regression to the mean, which can make the estimation of factor structures unstable and challenging to interpret. Further, we were not interested in using the pre-test scores to control for the post-test scores or analyze changes over time, due to the increased complexity that either analysis can introduce [25]. There were a total of 617 participants across all contexts.

We conducted a complete-case analysis on the data from SCS1. Not all participants completed all items. Given the format of the Qualtrics survey, students could only move forward and could not revisit items after they had submitted an answer. SCS1 items do not include an "I don't know" response option. We could not assume that the incomplete submissions were from students who ran out of time, nor could we assume that they were from students who gave up; these two scenarios have different implications for the answers that the participants did submit. We considered imputing the missing data values but eliminated it as a valid method in our case given the lack of randomness of the missing data. We ultimately performed listwise deletion for participants missing more than one response to a question, which resulted in the use of 547 observations of data. This is the set of data we used for all of our quantitative analyses.

Due to the setup of the Qualtrics-based version of the SCS1 assessment, demographic questions were not mandatory. When they were administered, the demographic questions were presented at the end of the assessment. Owing to this, we have demographic information for only 265 students, or 48% of our sample. The demographics that we have are presented in Table 1. All respondents were at least of college age (18 years of age).

Table 1. Demographics for the 265 students that completed the relevant questions when attached to the end of SCS1.

Gender	*n*	Race/Ethnicity	*n*	Primary Language	*n*	Programming Skills	*n*
Female	98	White/Caucasian	106	English	164	None	47
Male	163	Asian/Pacific Islander	119	Chinese	42	Very little	60
Other	4	Black	3	Korean	17	Some	113
		Hispanic	7	Vietnamese	7	Strong	37
		Multiple	30	Spanish	5	Very Strong	5
				Other/(No answer)	30	(No answer)	3

3.2 Analytical Frameworks

Our efforts are guided by two assessment frameworks from the education research literature. We draw on the assessment triangle presented in [6], which hinges on three components: *cognition*, *observation*, and *interpretation*. In our context, *cognition* refers to the design of SCS1, namely that the questions are constructed to provide a representation of students' knowledge in CS1. This has previously been established through the construction of FCS1 and its replication [13, 14, 32]. *Observation* would refer to whether the questions are carefully designed to link to students' knowledge of CS1. While this has been previously discussed with regards to the types of questions on SCS1 (definitional, code tracing, and code completion) [32], there has yet to be a mapping of what knowledge unit each question is eliciting. *Interpretation* is a characterization of patterns based on student results, drawing on observations of students' CS1 knowledge. In this paper, we are investigating the cognition side of SCS1 through observation and interpretation. We apply a framework of CS1 concepts in order to observe the potential responses from students. Then, we interpret student responses quantitatively to understand whether student performance on SCS1 can be used to indicate their understanding of specific concepts.

For the interpretation step, we use an analytical framework presented by Jorion et al. as a guide [20]. This framework provides guidance on an overall assessment level and on an individual concept level. We follow both, as we are trying to compare a unidimensional model of SCS1 (overall level) with different possible subscales (individual level). As suggested by this framework, we provide Classical Test Theory results, including item difficulty and discrimination (as seen in Table 2) and reliability (as seen in Table 4). We also follow the guidelines for structural analysis, presenting results of investigating the latent factor structure of the assessment through exploratory and confirmatory factor analyses.

The following sections use these frameworks as guidelines for our analysis. First, we present a structural analysis of the subscales that might conceivably exist within the assessment already using confirmatory factor analysis. Then, we present an *observation* of the SCS1 items, to help explain our findings from the first analysis. Last, we present an exploratory structural analysis using the dataset to guide the factors, in order to compare it with our hypothesis-driven model. Classical Test Theory results are referred to throughout these sections to provide further context for our findings.

4 DESIGNER-INTENDED SUBSCALES

To reveal a possible factor structure underlying SCS1, we conducted a quantitative analysis examining response data from SCS1. We started by investigating possible factor structures that could be inferred from the design of the instrument.

4.1 Methods for CFA

In multivariate statistics, there are two types of analyses used to gauge the potential latent structure in examinees' item responses: confirmatory factor analysis (CFA) and exploratory factor analysis (EFA). If researchers already have some hypothesis regarding the underlying structure, either informed by theory or by the design of the set of items, they can

Table 2. Difficulty, Item-total correlation, and discrimination index for each item on SCS1, as well as their factor labels. Values < 0.2 are bolded to indicate hard difficulty or poor discrimination.

Item	Difficulty	Item-total Correlation	Discrimination Index	9 Factor CFA	3 Factor CFA	3 Factor EFA
Q01	0.60	0.25	0.41	For	Definition	Factor 2
Q02	0.55	0.27	0.45	Logic	Tracing	Factor 2
Q03	0.67	0.40	0.57	While	Definition	Factor 1
Q04	0.29	0.22	0.34	Arrays	Definition	Factor 2
Q05	**0.17**	0.35	0.29	Return	Tracing	Factor 3
Q06	0.43	0.26	0.41	Conditional	Definition	N/A
Q07	0.24	0.31	0.31	While	Completion	Factor 3
Q08	0.35	0.24	0.30	For	Tracing	N/A
Q09	0.39	0.38	0.48	While	Tracing	Factor 2
Q10	0.43	0.35	0.49	Logic	Definition	Factor 2
Q11	0.38	0.37	0.48	Return	Definition	Factor 2
Q12	0.45	0.36	0.46	Basics	Definition	Factor 1
Q13	0.22	**0.12**	**0.16**	For	Completion	N/A
Q14	0.44	0.45	0.60	Recursion	Definition	Factor 2
Q15	0.24	0.25	0.28	Return	Completion	Factor 3
Q16	0.29	0.39	0.43	Parameters	Completion	Factor 3
Q17	0.31	0.32	0.39	Arrays	Completion	Factor 3
Q18	0.27	**0.16**	**0.18**	Recursion	Completion	N/A
Q19	0.49	0.50	0.71	Conditional	Tracing	Factor 1
Q20	**0.19**	**0.05**	**0.03**	Parameters	Definition	N/A
Q21	0.34	0.28	0.36	Conditional	Completion	Factor 3
Q22	0.42	0.29	0.42	Arrays	Tracing	Factor 1
Q23	0.59	0.35	0.52	Basics	Tracing	Factor 1
Q24	0.25	0.23	0.25	Recursion	Tracing	N/A
Q25	0.49	0.26	0.38	Basics	Completion	Factor 1
Q26	0.34	0.40	0.45	Logic	Completion	Factor 3
Q27	**0.17**	**0.12**	**0.13**	Parameters	Tracing	N/A

aggregate the items per factor of this structure and run a CFA in the examinees' item responses [19, 29]. If, on the other hand, no such hypothesis exists or can be confirmed, EFA can be performed [15, 41]. After the removal of ambiguous items, i.e., items similarly loading onto multiple factors or items that are not well explained by any factor, the fit of the data to the resulting factor structure then can be assessed using CFA. In both cases, the analysis results in what is called a "fit" of the data to the modeled factor structure. The analyses also report factor loadings, which demonstrate how strongly an item is related to a factor. Even though rules of thumb are by nature limited, one rule of thumb for factor loadings suggests that loadings larger than 0.45 should be considered "fair", with higher values indicating stronger relationships between the items and factors [43].

Given the goal of this stage of the project of investigating designer-intended subscales, we did have hypotheses about the underlying structure of SCS1. Thus, we used CFA for this phase and report on model fit and factor loadings.

We complemented our CFA with Classical Test Theory methods, including calculating item-level statistics and the reliability of each of the subscales for each of the models. Item-level statistics calculated include difficulty (proportion of correct responses to an item; i.e., 0 indicates no correct answers while 1 indicates every answer was correct), item-total correlation (correlation between performance on an item and performance on the assessment as a whole, minus the

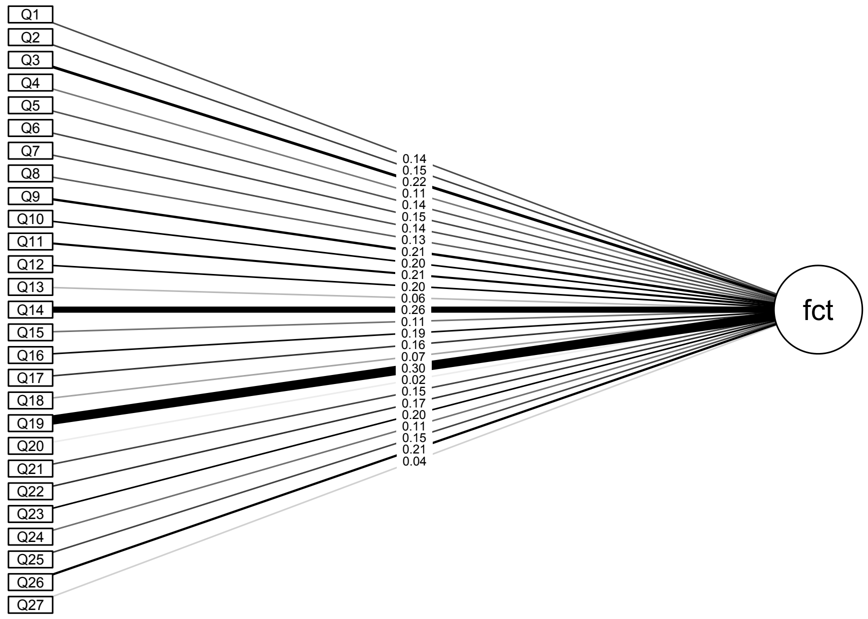

Fig. 1. Path diagram for 1-factor (unidimensional) CFA model. The numeric labels represent the path coefficients from the model. All paths were significant at the $p < 0.05$ level except for the path to Q20 ($p = 0.448$) and Q27 ($p = 0.062$).

item being analyzed), and discrimination index (the ability of an item to discriminate between those that score high on the assessment and those that score low on the assessment). These last two variables can both serve as indicators of the discrimination of the item, which is a measure of how the item is performing, and values below 0.2 indicate poor discrimination. The reliability of the subscales was measured via Cronbach's alpha; an alpha value of less than 0.5 is unacceptable, whereas a score of more than 0.7 is acceptable [23].

4.2 Findings of CFA

We began by running a CFA[1] under the assumption of unidimensionality, i.e., assuming that all items measured a single underlying concept. The result of running a CFA for this factor structure indicated an acceptable fit. As shown in Table 3, the model fit was reasonable for the population (as indicated by the Root Mean Square Error of Approximation (RMSEA) and Standardized Root Mean Square Residual (SRMR) statistics), as well as within the acceptable range for two more restrictive, global statistics, the Comparative Fit Index (CFI) and Tucker Lewis Index (TLI). Despite finding overall good model fit, the loadings of each item (as seen in Figure 1) were relatively poor, ranging from a minimum of 0.02 to a maximum of 0.30, as seen in Figure 1. The reliability of the test overall is acceptable ($\alpha = 0.78$).

Elliott Tew presented the concept that each question on FCS1 is intended to measure [14]. When SCS1 was created by replicating FCS1, that conceptual correspondence was maintained as part of the isomorphic mapping procedure [32]. In other words, the first question on FCS1 was intended to measure `for` loops, so the first question on SCS1 was also intended to measure `for` loops. Given that the designer of SCS1 had classified the 27 SCS1 items according to nine

[1]All models in this paper were created by one author in R, using the `lavaan` package.

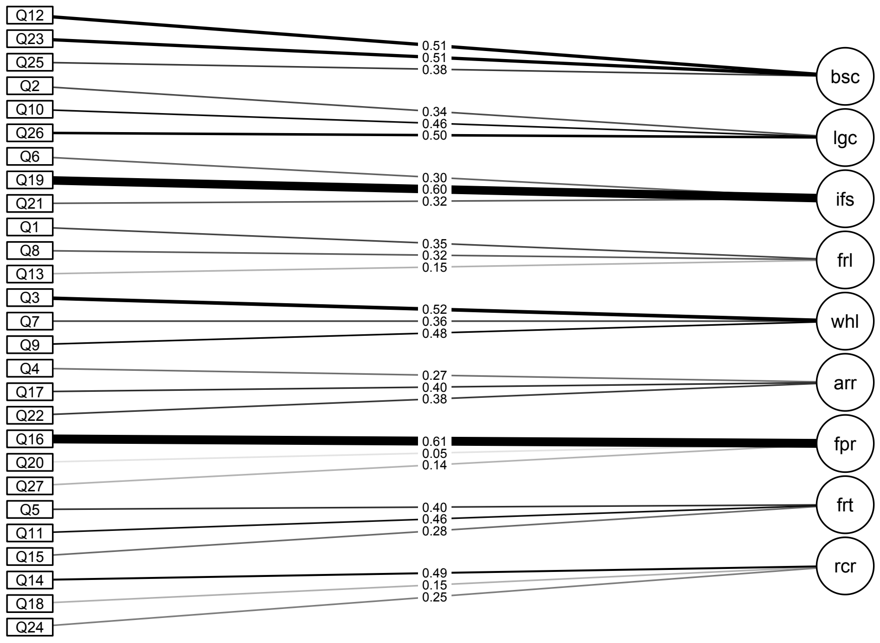

Fig. 2. Path diagram for 9-factor CFA model, based on the nine concepts: basics (bsc), logical operators (lgc), conditionals (ifs), `for` loops (frl), `while` loops (whl), arrays (arr), function parameters (fpr), function return values (frt), and recursion (rcr). The numeric labels represent the path coefficients from the model. All paths were significant at the $p < 0.05$ level except for the path to Q20 ($p = 0.409$).

concepts the items were intended to measure [14, p. 56], we continued our analysis by assessing the fit of the data to the induced nine-factor structure. The results, shown in the second row of Table 3 indicate that this assumption leads to a model with a consistently better fit across all statistics than the one-factor model. Figure 2 shows that the factor loadings for this model look much stronger than the 1-factor model. In this case, 10 of the 27 items had loadings that were fair or higher. This higher maximum loading is noteworthy because it indicates that these factors may better explain item responses than a single factor alone, despite little difference in the model fit statistics. This suggests that, despite similar model fit, the 9-factor model might be closer to the true structure of the factor space of SCS1. However, all of the concept-based subscales have reliability measures of $\alpha < 0.5$, indicating unacceptable reliability. That is to say, administering these items as concept-based subscales would not produce consistent results.

For the final CFA informed by the design of the instrument, we assumed that the underlying model was induced by the question type of items: definitional, code tracing, and code completion [14, p. 58]. As seen in Table 3, this model had similar fit statistics to the 9-factor model. Figure 3 shows that the factor loadings range from 0.04 up to 0.58. In this model, five items meet or exceed the threshold for factor loadings, which is less than the concept-based 9-factor model. Thus, although the model fit statistics were slightly (and not significantly) better than the 9-factor model, the factor loadings do not indicate that the 3-factor question-type-based model is any better. On the other hand, the reliability of each of these subscales is above 0.5, though not above the 0.7 threshold for acceptability. This higher reliability may simply be due to there being more items per subscale in this case (9 items per subscale, compared to 3 items per

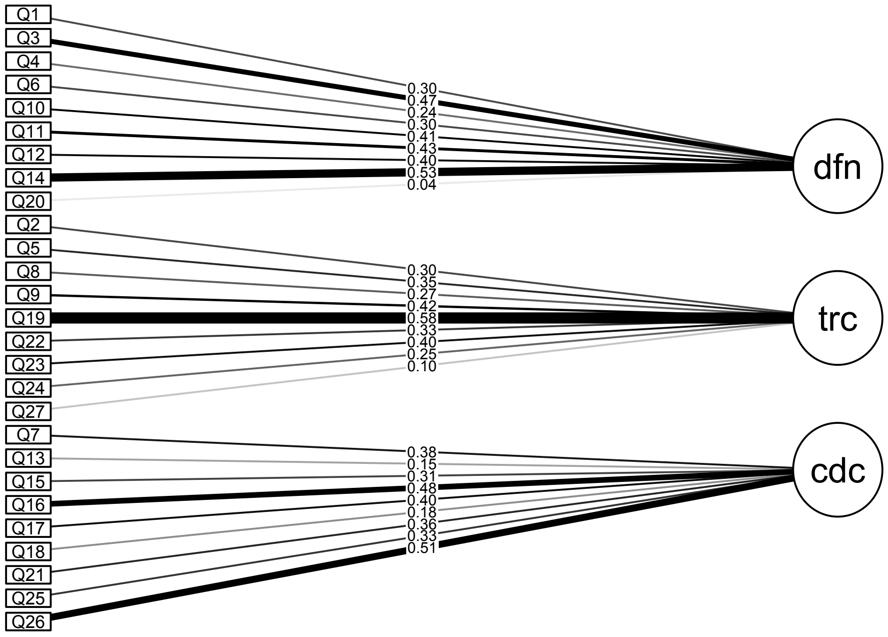

Fig. 3. Path diagram for 3-factor CFA model, based on the item types: definitional (dfn), code tracing (trc), and code completion (cdc). The numeric labels represent the path coefficients from the model. All paths were significant at the $p < 0.05$ level except for the path to Q20 ($p = 0.485$) and Q27 ($p = 0.063$).

Table 3. Model fit statistics of the models induced by the instrument design [14, p. 58] as well as induced by our data-driven exploratory factor analysis (see Section 6) along with "acceptable thresholds", as reported by Schreiber et al. [39].

	CFI	TLI	RMSEA	SRMR
Acceptable Range [39]	>0.95	>0.95	<0.060	<0.080
One-factor (unidimensional) structure	0.97	0.97	0.024	0.046
Nine-factor concept-based structure [14, p. 58]	0.98	0.97	0.022	0.042
Three-factor question-type-based structure [14, p. 58]	0.98	0.98	0.020	0.044
Three-factor data-driven structure (Section 6)	0.98	0.98	0.021	0.042

concept-based subscale). Because the number of items is a factor in the calculation of Cronbach's alpha, the α value does tend to increase with more items present [7].

5 THEORY-INFORMED DIMENSIONS

Our first phase presented some evidence for the concept- and question-type-based subscales. However, the factor loadings in these models and reliability measures of the subscales were generally poor. In response, we sought to investigate the subscales within SCS1 using CSER theory. To this end, we conducted a qualitative analysis to observe SCS1 items so as to best evaluate student responses on the assessment and, thus, the data on which our models are based. Our qualitative analysis looked at each item individually and assigned codes based on which CS concepts are

Table 4. Cronbach's alpha measures for potential subscales of SCS1

Overall Cronbach's α = 0.78					
Concept [14, p. 58]	α	Question Type [14, p. 58]	α	EFA (Section 6)	α
Basics	0.46	Definitional	0.54	EFA Factor 1	0.63
Logic	0.41	Tracing	0.54	EFA Factor 2	0.58
Conditionals	0.35	Code Completion	0.56	EFA Factor 3	0.60
`for` Loops	0.14				
`while` Loops	0.36				
Arrays	0.27				
Function Parameters	0.23				
Function Returns	0.34				
Recursion	0.18				

either present in the item or the knowledge of which is required to correctly respond to this item. These codes then could be used to better understand which concepts are truly being tested in SCS1 and whether or not those concepts are distributed across items as intended in the original design.

5.1 Methods for Qualitative Analysis

We followed a deductive coding technique to analyze the SCS1 items. Below, we describe what and how we coded SCS1 to find theory-informed subscales.

5.1.1 Coding Framework. To analyze the concepts on SCS1, we used a set of concepts identified by Goldman et al. [16] for use in a concept inventory. The work by Goldman et al. presents a set of concepts and competencies that can be taught and assessed in CS1-like courses ranked by difficulty and importance. This set was derived through a Delphi process with 20 experts and is presented as the foundation upon which a CS1 concept inventory might be built. Since Parker et al. report that SCS1 has been used to "build more valid instruments" [32], e.g., in the context of the Basic Data Structures Inventory [35], we used Goldman et al.'s framework to assess the degree to which these concepts are covered in SCS1.

5.1.2 Item Analysis. Every item on SCS1, as on any multiple-choice assessment, has two parts: the question stem and the answer choices. In the case of SCS1, every question stem has a prompt–a direct question to the test-taker. When coding based on Goldman et al. [16], we focused on what concepts are needed to understand and accurately respond to the question prompt. We deliberated also coding the answer choices. However, some answer choices introduce other concepts to serve as distractors. Knowledge of these concepts helps to eliminate answer choices but is not essential to correctly answering the item. Thus, we only coded the question prompt.

5.1.3 Coding Process. Two authors, both faculty members with experience in CSER for over a decade, acted as coders on the assessment, using MaxQDA 2022 for this analysis. One coder had not used SCS1, nor seen the items in detail prior to the coding, but had ample experience with teaching CS1-like courses, including the use of concept inventories, at different institutions and for different audiences. This coder did an initial pass of deductively coding the items using the concept framework. Then, a second coder looked over the initial codes. This coder had used SCS1 before and was familiar with the items, providing an expert checking of the codes. If the second person disagreed with a code, the two coders discussed these discrepancies until an agreement was reached. The items and corresponding codes were also presented to the research team as a whole to provide an overall quality check. Total agreement was ultimately reached in the codes as a result of this process.

5.2 Findings of Qualitative Analysis

From work by Porter et al., we expected that items on programming fundamentals (e.g., "Assignment Statements") would appear more often than items on more advanced concepts (e.g., "Call by Reference/Call by Value") [36]. This pattern can be seen in our results in Table 5–more advanced topics are seen in fewer items. Meanwhile, the most common concepts are "Assignment Statements,", and "Conditionals", which is to be expected. Our coding provides a replication of this prior work, which helps to verify that our coding process was accurate.

We coded many items with more than one concept. In some cases, this is evidence of the inevitable structure of CS courses. For example, Q08, which was intended to measure `for` loop knowledge, was coded for "Using loop variable inside a loop," but also "Tracing nested loops." It would be difficult to include the latter code without the former. However, the latter code is not strictly necessary in order to assess `for` loop knowledge. This coupling of codes, which occurs for every question that was intended to measure `for` loop knowledge, indicates a heightened complexity of the item that may be unnecessary for every question on that topic. There are some items that have multiple codes that are certainly not necessary to assess the intended concept. For example, Q15, which was intended to measure knowledge of function return values, was coded for "Assignment Statements," "Conditionals," and "Using loop variables inside a loop." While some of these appear necessary to assess the concept (e.g., "Assignment Statements"), it is not clear that the other two codes are strictly necessary; again, these additional codes add complexity to the item beyond what is required to assess the intended concept. Coincidentally, Q20 was labeled with the most Goldman concepts and was identified as a difficult question with poor discrimination, as seen in Table 2.

Regardless of the exact codes, the finding that most items cover more than one concept suggests that SCS1 may have a complex factor structure. The items do not map consistently or uniquely to concepts. Additionally, it is not the case that every tracing type question has "Control Flow Tracing." This code is also applied to some code completion items, further complicating the question-type factor structure. We did not attempt running a CFA using the 12 codes from the Goldman framework, given the clear lack of unique loadings of items onto concepts. As a result, this phase of our analysis provided two bits of information: insight into the findings from the designer-intended subscales, and that an exploratory, data-driven analysis of the underlying factor structure of SCS1 was needed.

6 DATA-DRIVEN FACTOR ANALYSIS

Although we could find some evidence for subscales when using designer-intended constructs, such as concepts and question types, we continued with a third phase of analyzing SCS1 to determine if there was a different latent factor structure within our data. We investigated what factors were within SCS1 without prior declaration of the factors (e.g. concepts) or items assigned to those factors (e.g. Q01 being a `for` loop question). We thus turned to a data-driven, exploratory analysis.

6.1 Methods for EFA

We started by using both parallel analysis and principal component analysis to discover the number of factors, both of which pointed to a 3-factor solution. Then, in using EFA with three factors, we found items with low communalities in all factor solutions, meaning that they did not clearly fit into data-driven factors. This resulted in the removal of items Q6, Q8, Q13, Q18, Q20, Q24, and Q27, which all had communalities less than 0.1. Coincidentally, four of those

Table 5. Mapping of SCS1 items onto competencies from the Goldman et al. framework.

	Intended Concept [14, p. 56]	Type	Goldman et al. [16]
Q01	`for` Loops	Definitional	Identifying characteristics of a problem
Q02	Logic	Tracing	Boolean Logic
Q03	`while` Loops	Definitional	Tracing nested loops
Q04	Arrays	Definitional	Assignment Statements Declaring/Manipulating Arrays
Q05	Function Returns	Tracing	Assignment Statements Control Flow Tracing
Q06	Conditionals	Definitional	Conditionals
Q07	`while` Loops	Code Completion	Assignment Statements Control Flow Tracing Using loop variables inside a loop
Q08	`for` Loops	Tracing	Tracing nested loops Using loop variables inside a loop
Q09	`while` Loops	Tracing	Control Flow Tracing Using loop variables inside a loop
Q10	Logic	Definitional	Boolean Logic
Q11	Function Returns	Definitional	Assignment Statements Conditionals
Q12	Basics	Definitional	Assignment Statements
Q13	`for` Loops	Code Completion	Tracing nested loops Using loop variables inside a loop
Q14	Recursion	Definitional	Recursion (tracing/patterns/structures)
Q15	Function Returns	Code Completion	Assignment Statements Conditionals Using loop variables inside a loop
Q16	Function Parameters	Code Completion	Assignment Statements Designing/using Procedures/Functions/Methods
Q17	Arrays	Code Completion	Control Flow Tracing Declaring/Manipulating Arrays Using loop variables inside a loop
Q18	Recursion	Code Completion	Recursion (tracing/patterns/structures)
Q19	Conditionals	Tracing	Assignment Statements Conditionals
Q20	Function Parameters	Definitional	Assignment Statements Call by Reference/Call by Value Designing/using Procedures/Functions/Methods Formal/Actual Parameters
Q21	Conditionals	Code Completion	Assignment Statements Conditionals
Q22	Arrays	Tracing	Assignment Statements Declaring/Manipulating Arrays
Q23	Basics	Tracing	Assignment Statements
Q24	Recursion	Tracing	Assignment Statements Conditionals Recursion (tracing/patterns/structures)
Q25	Basics	Code Completion	Assignment Statements
Q26	Logic	Code Completion	Boolean Logic Control Flow Tracing
Q27	Function Parameters	Tracing	Assignment Statements Call by Reference/Call by Value

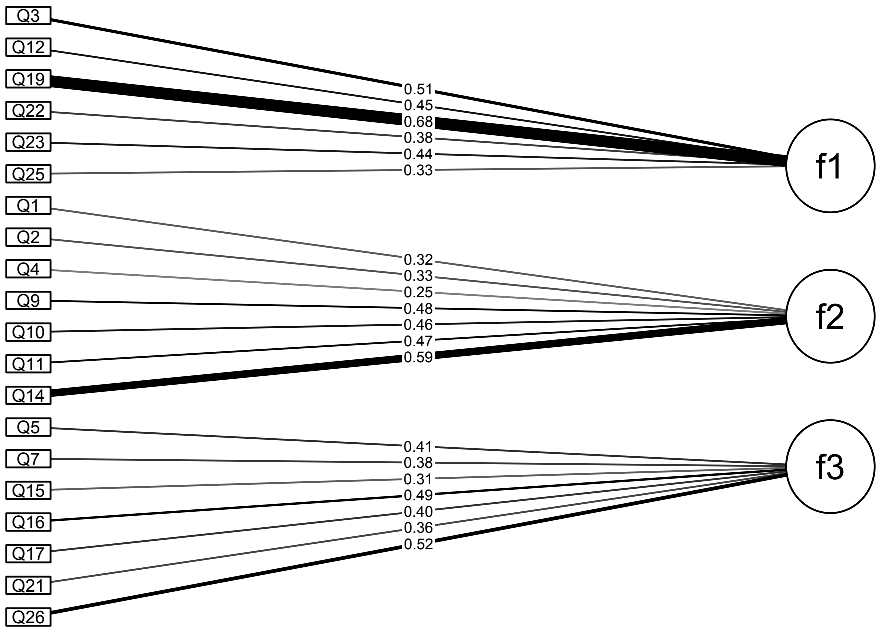

Fig. 4. Path diagram for 3-factor data-driven CFA model. The numeric labels represent the path coefficients from the model. All paths in this model were significant at the $p < 0.05$ level.

items also had discrimination values less than 0.2 (as seen in Table 2), further indicating their inadequacy. After the removal of these factors, we confirmed again that a 3-factor solution was most appropriate. Then an EFA model was fit with a varimax rotation, which was used to maximize the differences between the factors and thus force items to load strongest onto a single factor. The loadings from that model were used to specify which items loaded onto which factors. We used this information to run a final CFA with our data.

6.2 Findings of EFA

Our data-driven CFA, using our EFA factors, had acceptable fit statistics (3-factor data-driven CFA in Table 3). Overall, the model fits were comparable with the designer-intended models and thus not significantly better. As shown in Figure 4, loadings of items onto factors were similar to the 9-factor model, with 9 items with loadings of fair or better. This is better than the 3-factor question-type-based model and only slightly below the 9-factor concept-based model. However, the reliability of these EFA-produced factors as subscales of SCS1 is markedly better than the other subscales. The α values range from 0.58 to 0.60. While not yet meeting the acceptable threshold, they are higher than the other 3-factor model, even with fewer items per factor.

We examined which items loaded onto which factors to see if this data-driven model had any explainable features. The model and factor loadings were presented to the research team and discussed. Each item within a factor was brought up and compared in order to find patterns among the items. Some team members were very familiar with the items, as they were the coders for the theory-based analysis, while other team members were seeing the items for the first time. This variety of experience with the items allowed for a diversity of theories to emerge as to what brought items together onto a factor.

Factor 1 was the only factor to have every question type represented. It did include all of the items on the Basics concepts, as well as items on `while` loops, conditionals, and arrays. Factor 1 and Factor 2 split the easiest items between them, though Factor 1 did have two of the three items that 60% or greater of students answered correctly (Q03 and Q23). Coincidentally, this factor has the highest reliability measure of $\alpha = 0.63$. This factor was also interpreted by the research team as the items that were easy to read, at least compared to other items on the assessment.

Factor 2 was observed to have most of, and be mostly defined by, the definitional items, aside from two code tracing items. These items tended to have longer question stems and answer choices. One conclusion of the team was that this second factor represented code-free items or items with minimal code involvement.

Factor 3 had the most difficult items on the assessment. All of the items in Factor 3 were code completion items (with the exception of Q05, a tracing question), with most of the items involving some form of nesting behavior. As a result, the research team determined that the third factor of the data-driven model represented items that placed a high demand on working memory.

Our interpretations of these data-driven factors suggest that features of the items were the primary predictor of whether a student got them correct or not, rather than conceptual knowledge.

7 DISCUSSION AND LIMITATIONS

We set out to determine if there were usable and reasonable subscales within SCS1. Among the evidence that we collected across our three phases of analysis, we cannot justify the existence of subscales within SCS1. We did find appropriate model fits for concept- and question-type-based concepts. However, the factor loadings in these models were not all within a reasonable range. Further, Cronbach's alpha measures of reliability for these subscales were consistently below acceptable ranges. Our qualitative analysis informs our model findings, indicating that most SCS1 items address multiple concepts, providing evidence for why our factor loadings and reliability values were low. While we find subscales with higher reliability values using data-driven exploratory analysis, these factors are far less usable. In this model, the items fell into categories that would be undesired by instructors and researchers, such as item features instead of content or approach. Based on these results, we conclude that subscales cannot be defensibly constructed from the SCS1 assessment.

Our results align with prior work that identified problematic items [49]. In particular, Q13, Q18, Q20, Q24, and Q27 seem to not measure what they are supposed to measure. This finding from Xie et al. hinted at the multidimensionality of SCS1, which our findings confirm. Unfortunately, the dimensions appear to be more aligned to non-computing constructs, such as question readability and possibly working memory capacity.

Our findings, particularly with respect to the lack of a clearly interpretable factor structure of SCS1, can be explained, in part, through the theory of Knowledge in Pieces [9]. According to the theory of Knowledge in Pieces, knowledge is contained in small cognitive units, called upon in certain, specific situations. These small cognitive units combine to create an understanding of a concept or phenomenon. However, these pieces of knowledge are context-specific [18]. The Knowledge in Pieces theory can help explain why there is not a strong connection between SCS1 items in terms of concepts or item types: students may be able to use their units of knowledge around `for` loops for one item type (context) but not all. Similarly having familiarity with one item type does not guarantee the ability to answer that type of item across concepts. Instead, students may be demonstrating their familiarity with a context more than their understanding of the concepts. This would help explain our data-driven model findings, which resulted in factors based on context-specific item features rather than concepts targeted within the items.

Beyond contributing evidence about the shortcomings of SCS1, this work presents several methodological contributions to the CSER community. Our methods, both qualitative and quantitative, provide a road map for other computing assessment developers and evaluators to follow to analyze assessment items. Additionally, our qualitative coding strategy can also be used to identify gaps in an assessment, or *construct underrepresentation*. For example, two of the three `for` loop items involve tracing nesting loops. More items should be added to SCS1 on `for` loops without nesting.

At the same time, the methods presented in this paper have limitations. In our qualitative analysis, our coders may have been biased toward or against certain computing concepts or SCS1 items. While coder biases will always exist, we attempted to mitigate them by selecting coders with very different relations to SCS1 and by reaching total agreement through discussion. In our quantitative analysis, the sample was large but potentially non-representative of CS1 students. The data is from numerous administrations of SCS1 in different contexts, and, thus, little is known about the individual research design and incentives placed on participants taking SCS1. There is little meta-data attached to the students' SCS1 responses, so we cannot say how many countries or universities this data represents. We can say, though, that we have found at least one population that serves as a counterexample to the existence of a reliable decomposition of SCS1 into subscales.

Finally, to analyze SCS1 and examine what subscales are present, we used assessment frameworks [6, 20] and followed the steps taken by the Huffman and Heller [18]. Instead of basing our analysis around SCS1, we could have taken a different approach and designed a brand new CS1 assessment with subscales in mind. However, given the intensity of work needed to create a new assessment, we first set out to work with an existing assessment to ascertain the feasibility of subscales within it. Then, future work can build off what researchers have already carefully designed and validated, and thereby conserve resources and focus efforts accordingly.

8 CONCLUSION

In this paper, we present evidence that the SCS1 assessment does not currently have reliable subscales. Rather, most items include multiple concepts. High performance on SCS1 may also depend on test-taking abilities and reading comprehension. Thus, we strongly advise against using only parts of SCS1 to measure knowledge on certain concepts or coding skills, such as code tracing. The most valid use of SCS1 is to administer the assessment as it has been validated: in its entirety, with undergraduates, at the end of a CS1 course, and in a research context.

Still, the SCS1 assessment provides an excellent starting point for future work. Based on how the CSER community has used the SCS1 assessment [31], there is a demonstrated need for an introductory computing assessment that has subscales and can be customized to meet instructional needs or answer narrower research questions. This paper motivates future work to revise and expand SCS1 to meet these community needs.

To build a new SCS1 that includes concept subscales, we can start with the evidence from the 9-factor model which is based on the designer-intended factor structure. As we found this model to have weak factor loadings, existing items should be revised to facilitate stronger loadings onto their associated factors. Additionally, items should be added to diversify the difficulty of the assessment overall. These items should be written such that they focus on singular concepts as much as possible, as these kinds of items are not well-represented in the current version. After item creation, piloting, and thorough testing with representative samples, we can begin to construct an argument for a valid assessment with subscales for CS1 knowledge.

ACKNOWLEDGMENTS

This work is funded in part by the National Science Foundation under grant #1941642.

REFERENCES

[1] Vicki L Almstrum, Peter B Henderson, Valerie Harvey, Cinda Heeren, William Marion, Charles Riedesel, Leen-Kiat Soh, and Allison Elliott Tew. 2006. Concept inventories in computer science for the topic discrete mathematics. In *Working group reports on ITiCSE on Innovation and technology in computer science education*. 132–145.

[2] American Educational Research Association et al. 2014. *Standards for educational and psychological testing*. American Educational Research Association.

[3] Ada E Barach, Connor Jenkins, Serendipity S Gunawardena, and Krista M Kecskemety. 2020. MCS1: A matlab programming concept inventory for assessing first-year engineering courses. In *2020 ASEE Virtual Annual Conference Content Access*.

[4] Ryan Bockmon, Stephen Cooper, Jonathan Gratch, and Mohsen Dorodchi. 2019. (Re)Validating Cognitive Introductory Computing Instruments. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 552–557.

[5] Rudolf J. Bosscher and Johannes H. Smit. 1998. Confirmatory Factor Analysis of the General Self-Efficacy Scale. *Behaviour Research and Theory* 36, 3 (March 1998), 339–343. https://doi.org/10.1016/S0005-7967(98)00025-4

[6] National Research Council et al. 2001. *Knowing what students know: The science and design of educational assessment*. National Academies Press.

[7] Lee J Cronbach. 1951. Coefficient alpha and the internal structure of tests. *psychometrika* 16, 3 (1951), 297–334.

[8] Holger Danielsiek, Laura Toma, and Jan Vahrenhold. 2017. An Instrument to Assess Self-Efficacy in Introductory Algorithms Courses. In *Proceedings of the 2017 ACM Conference on International Computing Education Research, ICER 2017*, Josh Tenenberg, Donald Chinn, Judy Sheard, and Lauri Malmi (Eds.). ACM Press, New York, NY, 257–265. https://doi.org/10.1145/3105726.3106171

[9] Andrea A DiSessa. 2018. A friendly introduction to "knowledge in pieces": Modeling types of knowledge and their roles in learning. In *Invited lectures from the 13th international congress on mathematical education*. Springer, 65–84.

[10] Brian Dorn and Allison Elliott Tew. 2015. Empirical validation and application of the computing attitudes survey. *Computer Science Education* 25, 1 (2015), 1–36. https://doi.org/10.1080/08993408.2015.1014142

[11] Rodrigo Duran, Jan-Mikael Rybicki, Juha Sorva, and Arto Hellas. 2019. Exploring the value of student self-evaluation in introductory programming. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 121–130.

[12] Shannon Duvall, Scott Spurlock, Dugald Ralph Hutchings, and Robert C Duvall. 2021. Improving content learning and student perceptions in CS1 with scrumage. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 474–480.

[13] Allison Elliot Tew and Mark Guzdial. 2011. The FCS1: A Language Independent Assessment of CS1 Knowledge. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, SIGCSE 2011*, Thomas J. Cortina, Ellen Lowenfeld Walker, Laurie A. Smith King, and David R. Musicant (Eds.). ACM Press, New York, NY, 111–116. https://doi.org/10.1145/1953163.1953200

[14] Allison Elliott Tew. 2010. *Assessing fundamental introductory computing concept knowledge in a language independent manner.* Ph. D. Dissertation. Georgia Institute of Technology.

[15] Leandre R. Fabrigar, Duane T. Wegener, Robert C. MacCallum, and Erin J. Strahan. 1999. Evaluating the Use of Exploratory Factor Analysis in Psychological Research. *Psychological Methods* 4, 3 (Sept. 1999), 272–299. https://doi.org/10.1037/1082-989X.4.3.272

[16] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey L. Herman, Lisa C. Kaczmarczyk, Michael C. Loui, and Craig Zilles. 2008. Identifying Important and Difficult Concepts in Introductory Computing Courses using a Delphi Process. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, J. D. Dougherty, Susan H. Rodger, Sue Fitzgerald, and Mark Guzdial (Eds.). ACM Press, New York, NY, 256–260. https://doi.org/10.1145/1352135.1352226

[17] David Hestenes, Malcolm Wells, and Gregg Swackhamer. 1992. Force Concept Inventory. *The Physics Teacher* 30, 3 (March 1992), 141–158. https://doi.org/10.1119/1.2343497

[18] Douglas Huffman and Patricia Heller. 1995. What does the force concept inventory actually measure? *The Physics Teacher* 33, 3 (March 1995), 138–143. https://doi.org/10.1119/1.2344171

[19] Karl Gustav Jöreskog. 1969. A general approach to confirmatory maximum likelihood factor analysis. *Psychometrika* 34, 2 (June 1969), 183–202. https://doi.org/10.1007/BF02289343

[20] Natalie Jorion, Brian D Gane, Katie James, Lianne Schroeder, Louis V DiBello, and James W Pellegrino. 2015. An analytic framework for evaluating the validity of concept inventory claims. *Journal of Engineering Education* 104, 4 (2015), 454–496.

[21] John H. Kranzler and Frank Pajares. 1997. An exploratory factor analysis of the Mathematics Self-Efficacy Scale-Revised (MSES-R). *Measurement & Evaluation in Counseling & Development* 29, 4 (Jan. 1997), 215–228. https://doi.org/10.1080/07481756.1997.12068906

[22] Rina P. Y. Lai. 2021. Beyond Programming: A Computer-Based Assessment of Computational Thinking Competency. *ACM Transactions on Computing Education* 22, 2 (Nov. 2021), 14:1–14:27. https://doi.org/10.1145/3486598

[23] Charles E Lance, Marcus M Butts, and Lawrence C Michels. 2006. The sources of four commonly reported cutoff criteria: What did they really say? *Organizational research methods* 9, 2 (2006), 202–220.

[24] Lucas Layman, Yang Song, and Curry Guinn. 2020. Toward predicting success and failure in cs2: A mixed-method analysis. In *Proceedings of the 2020 ACM Southeast Conference*. 218–225.

[25] Frederic M Lord. 1967. A paradox in the interpretation of group comparisons. *Psychological bulletin* 68, 5 (1967), 304.

[26] Aleksandra Luszczynska, Urte Scholz, and Ralf Schwarzer. 2005. The General Self-Efficacy Scale: Multicultural Validation Studies. *Journal of Psychology* 139, 5 (2005), 439–457. https://doi.org/10.3200/JRLP.139.5.439-457

[27] Jessica L. Maples, Li Guan, Nathan T. Carter, and Joshua D. Miller. 2014. A Test of the International Personality Item Pool Representation of the Revised NEO Personality Inventory and Development of a 120-Item IPIP-Based Measure of the Five-Factor Model. *Psychological Assessment* 26, 4 (Dec. 2014), 1070–1084. https://doi.org/10.1037/pas0000004

[28] Lauren Margulieux, Tuba Ayer Ketenci, and Adrienne Decker. 2019. Review of measurements used in computing education research and suggestions for increasing standardization. *Computer Science Education* 29, 1 (2019), 49–78.

[29] Herbert W. Marsh, John R. Balla, and Roderick P. McDonald. 1988. Goodness-of-Fit Indexes in Confirmatory Factor Analysis: The Effect of Sample Size. *Psychological Bulletin* 103, 2 (May 1988), 391–410. https://doi.org/10.1037/0033-2909.103.3.391

[30] Christine A. Murphy, Delphine Coover, and Steven V. Owen. 1989. Development and Validation of the Computer Self-Efficacy Scale. *Educational and Psychological Measurement* 49, 4 (Winter 1989), 839–899. https://doi.org/10.1177/001316448904900412

[31] Miranda C. Parker, Mark Guzdial, and Allison Elliott Tew. 2021. Uses, Revisions, and the Future of Validated Assessments in Computing Education: A Case Study of the FCS1 and SCS1. In *ICER 2021: Proceedings of the 17th ACM Conference on International Computing Education Research*, Amy J. Ko, Jan Vahrenhold, Renée McCauley, and Matthias Hauswirth (Eds.). ACM Press, New York, NY, 60–68. https://doi.org/10.1145/3446871.3469744

[32] Miranda C. Parker, Mark Guzdial, and Shelly Engleman. 2016. Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. In *Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER 2016*, Judy Sheard, Josh Tenenberg, Donald Chinn, and Brian Dorn (Eds.). ACM Press, New York, NY, 93–101. https://doi.org/10.1145/2960310.2960316

[33] Miranda C. Parker, Yvonne S. Kao, Dana Saito-Stehberger, Diana Franklin, Susan Krause, Debra Richardson, and Mark Warschauer. 2021. Development and Preliminary Validation of the Assessment of Computing for Elementary Students (ACES). In *SIGCSE 2021: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, Pamela Cutter, Alvaro Monge, and Judithe Sheard (Eds.). ACM Press, New York, NY, 10–16. https://doi.org/10.1145/3408877.3432376

[34] Miranda C Parker, Amber Solomon, Brianna Pritchett, David A Illingworth, Lauren E Marguilieux, and Mark Guzdial. 2018. Socioeconomic status and computer science achievement: Spatial ability as a mediating variable in a novel model of understanding. In *Proceedings of the 2018 ACM Conference on International Computing Education Research.* 97–105.

[35] 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. In *Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER 2019*, Robert McCartney, Andrew Petersen, Anthony V. Robins, and Adon Moskal (Eds.). ACM Press, New York, NY, 111–119. https://doi.org/10.1145/3291279.3339404

[36] Leo Porter, Daniel Zingaro, and Raymond Lister. 2014. Predicting student success using fine grain clicker data. In *International Computing Education Research Conference, ICER 2014*, Quintin I. Cutts, Beth Simon, and Brian Dorn (Eds.). ACM Press, New York, NY, 51–58. https://doi.org/10.1145/2632320.2632354

[37] Vennila Ramalingam and Susan Wiedenbeck. 1998. Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Groups Analyses of Novice Programmer Self-Efficacy. *Journal of Educational Computing Research* 19, 4 (Dec. 1998), 367–381. https://doi.org/10.2190/C670-Y3C8-LTJ1-CT3P

[38] Marcos Román-González, Juan-Carlos Pérez-González, Jesús Moreno-León, and Gregorio Robles. 2018. Extending the Nomological Network of Computational Thinking with Non-Cognitive Factors. *Computers in Human Behavior* 80 (March 2018), 441–459. https://doi.org/10.1016/j.chb.2017.09.030

[39] James B. Schreiber, Amaury Nora, Frances K. Stage, Elizabeth A. Barlow, and Jamie King. 2006. Reporting structural equation modeling and confirmatory factor analysis results: A review. *The Journal of Educational Research* 99, 6 (2006), 323–338. https://doi.org/10.3200/JOER.99.6.323-338

[40] Yang Song, Yunkai Xiao, Jonathan Stephens, Emma Ruesch, Sean Roginski, and Lucas Layman. 2020. Suitability of SCS1 as a Pre-CS2 Assessment Instrument: A Comparison with Short Deliberate-practice Questions. In *Proceedings of the 2020 ACM Southeast Conference.* 313–314.

[41] Charles Spearman. 1904. “General Intelligence,” Objectively Determined and Measured. *The American Journal of Psychology* 15, 2 (April 1904), 201–292. https://doi.org/10.2307/1412107

[42] Phil Steinhorst, Andrew Petersen, and Jan Vahrenhold. 2020. Revisiting Self-Efficacy in Introductory Programming. In *Proceedings of the 16th ACM Conference on International Computing Education Research, ICER 2020*, Anthony Robins and Amy J. Ko (Eds.). ACM Press, New York, 158–169. https://doi.org/10.1145/3372782.3406281

[43] Barbara G Tabachnick, Linda S Fidell, and Jodie B Ullman. 2013. *Using multivariate statistics.* Vol. 6. pearson Boston, MA.

[44] Cynthia Taylor, Daniel Zingaro, Leo Porter, Kevin C. Webb, Cynthia B. Lee, and Michael Clancy. 2014. Computer Science Concept Inventories: Past and Future. *Computer Science Education* 4, 24 (2014), 253–276. https://doi.org/10.1080/08993408.2014.970779

[45] Dion Timmermann, Christian Kautz, and Volker Skwarek. 2016. Evidence-based re-design of an introductory course “programming in C”. In *2016 IEEE Frontiers in Education Conference (FIE).* IEEE, 1–5.

[46] Gholamreza Torkzadeh and Xenophon Koufteros. 1994. Factorial Validity of a Computer Self-Efficacy Scale and the Impact of Computer Training. *Educational and Psychological Measurement* 54, 3 (Sept. 1994), 813–821. https://doi.org/10.1177/0013164494054003028

[47] Meng-Jung Tsai, Ching-Yeh Wang, and Po-Fen Hsu. 2019. Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education. *Journal of Educational Computing Research* 56, 8 (Jan. 2019), 1345–1360. https://doi.org/10.1177/0735633117746747

[48] Laurie Williams, Eric Wiebe, Kai Yang, Miriam Ferzli, and Carol Miller. 2002. In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education* 12, 3 (2002), 197–212. https://doi.org/10.1076/csed.12.3.197.8618

[49] Benjamin Xie, Matthew J. Davidson, Min Li, and Amy J. Ko. 2019. An Item Response Theory Evaluation of a Language-Independent CS1 Knowledge Assessment. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, Elizabeth K. Hawthorne, Manuel A. Pérez-Quiñones, Sarah Heckmann, and Jian Zhang (Eds.). ACM Press, New York, NY, 699–705. https://doi.org/10.1145/3287324.3287370

[50] María Zapata-Cáceres, Estefania Martín-Barroso, and Marcos Román-González. 2020. Computational Thinking Test for Beginners: Design and Content Validation. In *Proceedings of the 2020 IEEE Global Engineering Education Conference (EDUCON)*, Alberto Cardoso, Gustavo Alves, and Teresa Restivo (Eds.). 1905–1914. https://doi.org/10.1109/EDUCON45650.2020.9125368