The FCS1: A Language Independent Assessment of CS1 Knowledge

Allison Elliott Tew
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4
aetew@cs.ubc.ca

Mark Guzdial
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332-0760
guzdial@cc.gatech.edu

ABSTRACT

A primary goal of many CS education projects is to determine the extent to which a given intervention has had an impact on student learning. However, computing lacks valid assessments for pedagogical or research purposes. Without such valid assessments, it is difficult to accurately measure student learning or establish a relationship between the instructional setting and learning outcomes.

We developed the Foundational CS1 (FCS1) Assessment instrument, the first assessment instrument for introductory computer science concepts that is applicable across a variety of current pedagogies and programming languages. We applied methods from educational and psychological test development, adapting them as necessary to fit the disciplinary context. We conducted a large scale empirical study to demonstrate that pseudo-code was an appropriate mechanism for achieving programming language independence. Finally, we established the validity of the assessment using a multi-faceted argument, combining interview data, statistical analysis of results on the assessment, and CS1 exam scores.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*computer science education, curriculum*

General Terms

Experimentation, measurement

Keywords

Assessment, CS1, programming, validity

1. INTRODUCTION

Measuring student learning is fundamental to any educational endeavor. A primary goal of many computer science education projects is to determine the extent to which a given instructional intervention has had an impact on student learning. However, the field of computing lacks valid and reliable assessment instruments for pedagogical or research purposes. Without such valid assessments, it is difficult to accurately measure student learning or establish a relationship between the instructional setting and learning outcomes. The goal of assessment research in computer science is to have valid ways of measuring student conceptions of fundamental topics, which will enable both research into how understanding of knowledge in the domain develops as well as enable curricular innovation and reform grounded in this knowledge.

Many science, technology, engineering and mathematics (STEM) disciplines have standard validated assessment tools that allow educators and researchers to accurately measure student learning and evaluate curricular innovations (e.g., [6, 9]). However, computer science does not have a similar set of validated assessment tools. There are four existing validated exams of computer science conceptual knowledge. Two, the CS Advanced Placement (AP) exam and the Advanced Level General Certificate of Education (A-level) in Computing, focus on the introductory sequence in computing. These exams, written in the Java programming language, are controlled and administered by specific educational testing boards and thus are not widely applicable and available to educators and researchers for general use. The remaining two exams, the Major Field Test for Computer Science and the GRE Computer Science Subject Test, cover a wide range of material as they are designed to evaluate end of degree program learning objectives.

The CS education community has shown growing interest in assessment research, and two related projects are underway. In dissertation work, Decker [3] designed an assessment for an introductory sequence of programming courses (CS1 and CS2) in Java. The instrument was developed and tested at a single institution, and therefore its validity claims cannot be generalized beyond that context. Craig Zilles and colleagues have received funding to develop a set of concept inventories for computing, in discrete math, digital logic design, and programming fundamentals. The researchers have elicited a set of troublesome concepts from educational experts [4] and are conducting think-aloud interviews to capture student misconceptions [7]. The work is preliminary and instruments are still being developed.

Our work differs from these efforts in that we created a rigorously validated exam that could be widely adopted and used in any introductory CS course. The goal was to create an exam to measure understanding of fundamental computing concepts, independent of programming language, that would not be overly biased by any particular pedagogical paradigm. We focused on three questions regarding assessment of introductory concepts in computer science. How can existing test development methods be applied and adapted to create a valid assessment instrument for CS1 conceptual knowledge? To what extent can pseudo-code be used as the mechanism for achieving programming language independence in an assessment instrument? And to what extent does the language independent instrument provide a valid measure of CS1 conceptual knowledge?

This paper begins with an overview of the method used to develop the first assessment instrument for introductory computer science concepts that is applicable across a variety of current pedagogies and programming languages.

2. METHOD FOR DEVELOPING A VALIDATED CS1 ASSESSMENT

The fields of education and psychology have developed a rich history in developing and validating measurement instruments for a variety of purposes [1]. We applied these established methods and practices for developing valid measures, summarized below, adapting them where necessary for the field of computer science.

The first step in test development is to establish the purpose and definition of the test — what is to be measured and what the scores mean. The test specification includes the definition of the conceptual content, or constructs, that is to be measured, the format of the questions, and the scoring procedures. The test specification should be reviewed by a panel of experts to provide content validity evidence and ensure that all constructs are adequately represented and extraneous constructs are not included.

After the test specification has been developed and verified, a test bank of questions should be developed to cover all constructs identified in the specification. Piloting of the questions then takes place. Pilot tests examine the suitability of the questions, allowing necessary revisions to be made prior to the selection of the final candidate questions. The last stages of test development are empirical studies of individual responses to establish validity and reliability. Validity testing ensures that the test is measuring the intended constructs, and reliability testing verifies that the results are consistent over repeated examinations, and thus are dependable.

To create the FCS1 Assessment, two adaptations were required. The first methodological change centered around creating an exam focused on concepts not programming language syntax, so the assessment can be as widely applicable as possible. The method required the addition of a step to verify the programming language independence of the exam. To achieve language independence for a CS1 exam, we utilized a verbose pseudo-code as the exam programming language. We evaluated the effectiveness of this approach using a combination of think aloud and empirical studies[1]. These studies were required to ensure students are able to appropriately transfer understanding from their programming language of instruction to pseudo-code and to ensure that students are able to demonstrate their understanding adequately in the new language independent exam.

The second change is required because the standard methods for validating the instrument against existing valid measures did not apply to this exam, which is the first of its kind in the field of computing. So a validity argument was crafted using a combination of think aloud interviews and statistical analysis techniques.

The first three steps in this process have previously been described in [13]. Section 3 contains the results of our empirical study investigating the use of pseudo-code as the mechanism to achieve language independence, and the process used to validate the exam is explained in Section 4. The data collected will allow preliminary analysis of the internal reliability of the exam. However, full-scale reliability testing is left for future work.

3. FCS1 ASSESSMENT STUDY

We conducted a large scale empirical study comparing student performance on the FCS1 Assessment[2] to a comparable version of the assessment instrument written in the students' CS1 programming language. Participants were selected at the end of CS1 courses taught in Java, Matlab, or Python, and statistical analysis was used to look for correlations in student performance between the two exams.

3.1 Study Design

We recruited participants from four different introductory courses taught at two universities by four separate faculty members, so that the definition and understanding of CS1 knowledge was not tied to a particular faculty member or institution. There were a total of 952 participants who were enrolled in a CS1 course in Java ($n = 80$), Matlab ($n = 520$), or Python ($n = 352$).

The study consisted of two assessment exams given, one week apart, under testing conditions — no questions were permitted and collaboration was not allowed. Participants completed the FCS1 Assessment and a comparable version of the FCS1 rewritten in the programming language used in their introductory CS course. The comparable version was created using the alternate questions for each concept in the test bank. A counterbalanced quasi-experimental design was used to reduce bias from ordering effect.

3.2 Data Analysis Results

Before data analysis could begin, outliers from the data set that would bias or skew the results were removed. An external researcher verified the rules for exclusion and independently reviewed all of the exams that were removed from the data set to confirm that they met one or more of the exclusionary criteria.

3.2.1 Scoring the FCS1

The pseudo-code and CS1 language versions of the FCS1 were graded, awarding a 1 for a correct answer and a 0 for an incorrect answer. (Any question left blank was not scored.) Student participants answered an average of 8.82 (33.78%, $\sigma = 3.649$) questions correctly on the pseudo-code version

[1]Due to space constraints, we limit our discussion here to the empirical study. See [12] for more details.

[2]Unfortunately for the validity and reliability of the assessment instrument, exam questions must remain private and cannot be published. This prevents potentially biasing participants involved in validation studies.

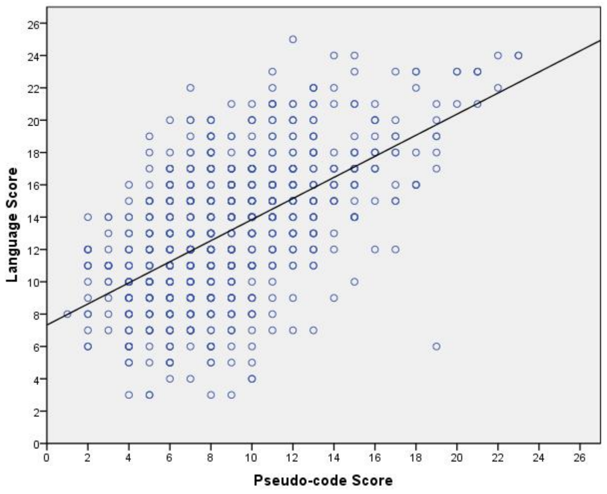

Figure 1: Scatterplot of Scores for Correlation of Pseudo-code and Language Versions of FCS1

of the FCS1. The maximum score was a 23, and the minimum score was a 1. Students were more successful answering questions on the CS1 language specific exam. An average of 13.13 (48.61%, $\sigma = 4.195$) questions were answered correctly, and the minimum and maximum score both increased by two points to 3 and 25, respectively. While the overall scores were higher on this version of this exam, similar concepts appear among those identified as least and most difficult across the populations.

Questions about math operators and if statements were among the most commonly answered correctly. The programming constructs related to function parameters, function return values, and recursion were the most difficult questions on both the pseudo-code and the language specific version of the FCS1.

3.2.2 Correlation with CS1 Language Version

We investigated the effectiveness of pseudo-code as the mechanism to achieve programming language independence by examining whether students were able to transfer their understanding of fundamental computing concepts from their CS1 programming language to pseudo-code. A correlation analysis was conducted to look for evidence of a positive correlation between the overall scores, i.e. the number of questions answered correctly, for each participant on both versions of the FCS1 Assessment.

A Pearson product-moment correlation coefficient was computed to assess the relationship between the score on the pseudo-code version and the score on the language specific version of the assessment instrument. There was a strong, positive correlation between the two variables, Pearson's $r(850) = .572$, $p <= 0.001$. A scatterplot summarizes the results in Figure 1.

Having found a positive correlation, which meets the guidelines for large effect size ($r >= .37$) in the social sciences [2], subsequent analyses focused on correlating exam scores for each CS1 programming language sub-population. Were students from each of the CS1 programming languages examined – Java, Matlab, and Python – able to transfer their understanding to pseudo-code? Or is the syntax of the pseudo-code too distinct from what the participants have learned to facilitate the expression of conceptual understanding in a new programming language?

Pearson product-moment correlation coefficients were computed to assess the relationship between the score on the pseudo-code version and the score on the language specific version of the assessment for each programming language participant group (see Table 1). There was a strong, positive correlation between the scores on the pseudo-code and language versions of the assessment for each of the programming languages studied. Participants from the CS1 taught in Java had the strongest correlation, Pearson's $r(74) = .665$, $p <= 0.001$. Although the pseudo-code syntax had more elements in common with Python than the other programming languages studied, the Python participant group had the lowest correlation coefficient, Pearson's $r(285) = .415$, $p <= 0.001$. The Python population was comprised of students, normally students from STEM majors, enrolled in an introductory computing course as well as students who were enrolled in a non-traditional media computation course designed for liberal arts students. A Pearson's correlation coefficient was also computed for each of these subpopulations of the Python participant group. There was a strong positive correlation, (Pearson's $r(43) = .615$, $p <= 0.001$), for students enrolled in the CS-based python CS1 course that was similar to the strength of the correlation found in the Java and Matlab populations. While the effect size for the media-based Python course was smaller, $r = .372$, it met guidelines to be classified as a strong, positive correlation.

Overall the results demonstrate that there was a strong, positive correlation between the scores on the pseudo-code and CS1 language versions of the FCS1 for both traditional and non-traditional pedagogical approaches to CS1.

4. ESTABLISHING VALIDITY

After the FCS1 Assessment was piloted, the final step in the development process was to establish the validity of the exam. In general, there are two classes of evidence used to support validity claims. *Content* related evidence ensures that the assessment's content appropriately operationalizes the constructs it is intended to measure. Content validity for the FCS1 was previously established by expert panel review. *Construct* related evidence provides the second set of support for validity and is the focus of the discussion here. Together, content and construct validity enable a test developer to provide evidence that the instrument is measuring student knowledge as intended.

Given the position of this exam as the first of its kind in the field, the standard correlation methods for establishing validity do not apply. Since construct validation is dependent on inferences drawn from a variety of data [8, 10], we proposed a three-pronged approach to establishing the validity of the assessment instrument. The validity evidence is three-fold: think aloud interview data, student responses to the FCS1, and student CS1 exam scores.

4.1 Qualitative Analysis of Student Interviews

Qualitative analysis of transcripts from think aloud interviews with students provides evidence whether students were answering questions based on their knowledge of the conceptual content. The goal of the analysis was to determine whether students were using knowledge about the intended construct to answer the question. Alternatively,

Table 1: Significant Pearson Correlations of FCS1 Assessment[‡]

Population	Version	Language	Midterm 1	Midterm 2	Midterm 3	Final
Total	Pseudo	.572	.111	.149	.309	.499
	Language	*	.120	.128	.491	.542
Java	Pseudo	.665	.408	.446	—	.511
	Language	*	.406	.582	—	.680
Matlab	Pseudo	.547	.500	.527	.443	.438
	Language	*	.531	.536	.488	.505
Python	Pseudo	.415	.200	.216	*n.s.*	.453
	Language	*	.424	.429	.392	.539
CS-Python	Pseudo	.615	.719	.685	.625	.679
	Language	*	.525	.445[†]	.429[†]	.437[†]
Media-Python	Pseudo	.372	.246	.220	.305	.262
	Language	*	.456	.453	.542	.601

[‡] All correlations, unless otherwise noted, are significant at the $p <= 0.001$ level.
[†] Correlation is significant at the $p <= 0.01$ level.
[*] Language-Language self comparison omitted.
— No midterm 3 measure was analyzed because only two midterms were administered in the course.

additional information could be required to correctly answer the question or other cues could be enabling correct responses without knowledge of the concept.

Thirteen student participants, of varying ability levels, were recruited from introductory courses taught in three different programming languages (Java, Matlab, and Python). Students were asked to participate in a think aloud interview conducted while they were completing the pseudo-code version of the FCS1. The interview data was transcribed and content analysis was used to analyze the participant responses. Specifically, responses were coded for correctness, errors made, and evidence of reasoning using the new pseudo-code language syntax.

Valid responses demonstrated evidence of student reasoning about the question's intended construct and reaching a correct or incorrect response based upon the level of understanding. That is, if a student held misconceptions about a topic, they selected an incorrect answer. Otherwise, they answered correctly. A majority (83.07%) of the responses analyzed were categorized as valid. The multiple-choice testing format allowed a few correct responses (5.92%) despite students' misunderstanding about the questions conceptual content. Further, there were a small number of responses (4.44%) where student misconceptions about another topic prevented them from answering the question correctly. However, there were no instances where students reasoning about another construct enabled a correct response to a question without knowledge of the concept.

Overall, the interview evidence provides a qualitative argument that students are able to read and reason with the pseudo-code syntax. Further, they are using their knowledge about the intended construct to answer the question.

4.2 Quantitative Analysis using IRT

The FCS1 Assessment study data and analysis provides a quantitative argument towards construct validity. The Pearson correlation analysis demonstrated that students have a comparable knowledge to that measured in a language specific version of the exam. If the questions are representative measures of knowledge in students' CS1 programming language, then validity of the exam is contingent upon the questions themselves. Item response theory (IRT) is the statistical analysis technique employed to make this validity claim. If the questions are shown to be "good" questions (i.e., of appropriate difficulty and discrimination) and students demonstrate comparable knowledge to a language specific exam, then the argument is made that this is an accurate representation of students' understanding of the topics.

Using a three-parameter logistic model (3PL) [5], IRT was used to analyze the FCS1 participant responses to discern the strength and weakness of each item in the test. Overall, most (24 of 27) questions displayed strong item discrimination, adequate difficulty, and low guessing probability. Two questions, Q12 and Q13, were too difficult. That is $b_i > 3.0$, which implies that less than 10% of the examinees had a 50% probability of answering the question correctly. Overall, the items on the exam showed adequate levels of discrimination. However due to the overall high level of difficulty of the exam, the test shows better discrimination among student participants of higher ability than those with lower ability.

Question 2 displayed a guessing probability that exceeded recommended limits. The guessing item parameter, $c_i = 0.442$, a probability of a low ability participant guessing the question correctly over 40% of the time. Two other questions, Q21 and Q25, had guessing parameter values that were elevated but were within one standard deviation of the expected value of c_i =0.20 for a 5 item multiple-choice question.

Overall, IRT analysis identified only four questions, Q2, Q12, Q13, and, Q25 that need to be revised or dropped from the exam. Question 2 has a high guessing probability, questions 12 and 13 are too difficult, and question 25 has a relatively high guessing probability and provides relatively little additional information about an examinees's ability level.

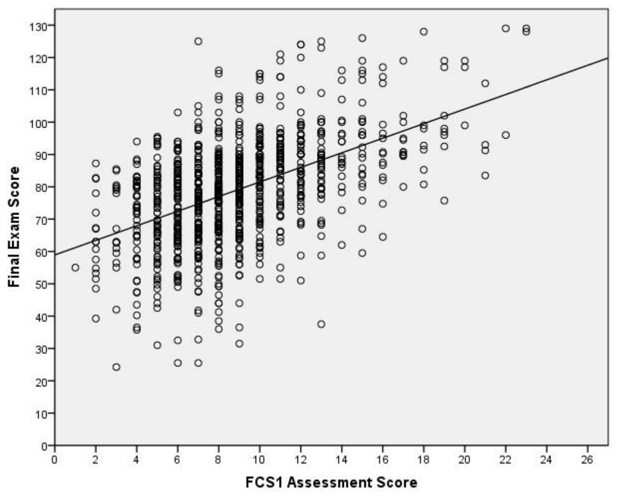

Figure 2: Scatterplot of Scores for Correlation of FCS1 Assessment Score and CS1 Final Exam Score

4.3 Correlation with External Measures of CS1 Knowledge

Student exam scores, as an external definition and measure of CS1 knowledge, provide the final piece of evidence for construct validity. Pearson's correlation analysis was used to investigate whether student scores on the FCS1 can be positively correlated with their scores on CS1 exams.

A Pearson product-moment correlation coefficient was computed to assess the relationship between the score on the FCS1 Assessment and the score on the final exam in CS1. There was a strong, positive correlation between the two variables, Pearson's $r(931) = .499$, $p <= 0.001$. A scatterplot summarizes the results (Figure 2). Further there were significant, yet weaker, correlations between scores on the assessment and scores on individual midterm exams (see Table 1).

Having found a strong positive correlation, subsequent analyses focused on correlating exam scores with each CS1 programming language population. Pearson product-moment correlation coefficients were computed to assess the relationship between the score on the pseudo-code version of the assessment and the final exam score for each programming language participant group (see Table 1). There was a strong, positive correlation between the scores on the assessment and the final exam for each of the programming languages studied. Participants from the CS1 taught in Java had the strongest correlation with final exam score, Pearson's $r(79) = .511$, $p <= 0.001$. The Python population was again comprised of students in CS and media computation versions of CS1. A Pearson's correlation coefficient was computed for each of these subpopulations of the Python participant group. There was the strongest positive correlation, (Pearson's $r(69) = .679$, $p <= 0.001$), for students enrolled in the CS-based python CS1 course. The effect size for the media-based Python course was smaller, $r = .262$, yet there was still a significant positive correlation with medium effect size.

Pearson correlation coefficients were also computed for each midterm score for each programming language participant group (see Table 1). There was a strong, positive correlation between the scores on the assessment and midterm exam scores for all CS1 courses in Java, Matlab, or Python except the non-traditional media computation approach. Participants from the CS-based Python course had the strongest correlations with each midterm exam score. The strongest correlation coefficient was for midterm 1, Pearson's $r(69) = .719$, $p <= 0.001$. The correlation between assessment score and midterm exam scores for the media-based Python course showed significant correlations with medium effect size.

Overall the results demonstrate that there was a strong, positive correlation between the scores on the FCS1 Assessment and final exam scores for both traditional and non-traditional pedagogical approaches to CS1. In addition there was a strong positive correlation with individual midterm exam scores for traditional approaches to CS1 taught in Java, Matlab, and Python.

4.4 Validity Argument

Two issues are central to construct validation of an assessment instrument: *construct under-representation* and *construct-irrelevant variance* [10]. These issues are explored by the following questions: (1) Does the assessment adequately operationalize the intended construct? and (2) Is performance on the assessment influenced by factors that are ancillary to the construct?

The matter of construct under-representation was resolved by the panel of expert reviewers confirming an adequate definition of fundamental CS1 concepts included in the test specification. Further, item response theory analysis indicated that a majority (24 out of 27) of the items on the assessment provide adequate information about student participant ability. Thus, overall the definition and measurement of the constructs specified are appropriate for the FCS1.

A variety of metrics were used to identify potential sources of construct-irrelevant variance, with almost all measures providing evidence to the contrary. Think aloud interviews with participants revealed that students were able to provide valid answers about the intended construct on over 85% of the questions. Scores on the pseudo-code version of the assessment had a strong positive correlation with scores on the CS1 language specific version of the exam. When combined with IRT results that demonstrate that 85.18% of the questions were of appropriate difficulty and discrimination, it is appropriate to infer that the FCS1 is a reasonable measure of CS1 knowledge.

Overall the validity studies provide evidence that students are reading and reasoning with the pseudo-code to answer questions in the manner intended. In addition, there is empirical evidence of the quality of the questions used to measure understanding that further correlates with external faculty definitions and measures of CS1 knowledge. Therefore, the FCS1 does provide a valid measure of introductory computing concepts for procedurally-based introductory computing courses taught in Java, Matlab, or Python at the university level.

5. CONCLUSION AND FUTURE WORK

Research and development on the FCS1 Assessment is continuing along a number of paths. This research has focused on establishing the validity of the exam for a limited number of constructs with a focused population of university students studying common introductory CS1 programming languages. Natural extensions of this work include adding additional concepts, establishing the reliability of the exam, and implementing the test on-line to reduce the resources required for data collection and test administration. We are also exploring the applicability and validity of using the FCS1 for measuring student knowledge across different pedagogical paradigms and programming languages, such as graphical or functional approaches.

The availability of a valid assessment instrument to measure student understanding of CS1 concepts enables a variety of directions for CS education research involving the FCS1. As an example, a programming language independent assessment instrument permits the comparison of pedagogical approaches in ways that were not previously available. In particular, it is now possible to investigate whether there are identifiable and persistent differences in student understanding of fundamental computer science concepts based upon the pedagogical approach or programming language used in the first course. When combined with other research methods, it would be possible to begin to identify which of the many factors in a CS1 learning environment (e.g., the instructor, programming language, integrated development environment (IDE), pedagogical approach, student motivation) are the levers that drive student mastery of computing concepts.

Assessment is an important piece of the computer science education research agenda. But as assessment practices "shape people's understanding about what is important to learn" [11, p. 111] it is vital that the field continues to expand the scope and availability of validated assessment tools.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Nos. 0512213 and 0634629. Any opinions findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

[1] American Educational Research Association, American Psychological Association, and National Council on Measurement in Education. *Standards for educational and psychological testing.* American Educational Research Association, Washington, DC, 1999.

[2] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences.* Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd edition, 1988.

[3] A. M. Decker. *How Students Measure Up: An Assessment Instrument for Introductory Computer Science.* PhD thesis, University at Buffalo (SUNY), Buffalo, NY, 2007.

[4] K. Goldman, P. Gross, C. Heeren, G. Herman, L. Kaczmarczyk, M. C. Loui, and C. Zilles. Identifying important and difficult concepts in introductory computing courses using a Delphi process. In *SIGCSE '08: Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, pages 256–260, 2008.

[5] R. K. Hambleton, H. Swaminathan, and H. J. Rogers. *Fundamentals of item response theory.* Sage Publications, Newbury Park, CA, 1991.

[6] D. Hestenes, M. Wells, and G. Swackhamer. Force concept inventory. *The Physics Teacher*, 30:141–158, March 1992.

[7] L. C. Kaczmarczyk, E. R. Petrick, J. P. East, and G. L. Herman. Identifying student misconceptions of programming. In *SIGCSE '10: Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, pages 107–111, 2010.

[8] M. T. Kane. Validation. In R. L. Brennen, editor, *Educational Measurement*, pages 17–64. American Council on Education/Praeger Publishers, Westport, CT, 4th edition, 2006.

[9] J. C. Libarkin and S. Anderson. Assessment of learning in entry-level geoscience courses: Results from the geoscience concept inventory. *Journal of Geoscience Education*, 53:394–401, 2005.

[10] M. D. Miller, R. L. Linn, and N. E. Gronlund. Validity. In *Measurement and assessment in teaching*, pages 80–99. Pearson Education, Upper Saddle River, NJ, 10th edition, 2009.

[11] P. A. Moss, B. J. Girard, and L. C. Haniford. Validity in Educational Assessment. *Review of Research in Education*, 30(1):109–162, 2006.

[12] A. E. Tew. *Assessing fundamental introductory computing concept knowledge in a language independent manner.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2010.

[13] A. E. Tew and M. Guzdial. Developing a validated assessment of fundamental CS1 concepts. In *SIGCSE '10: Proceedings of the 41st ACM Technical Symposium on Computer Science education*, pages 97–101, 2010.