



PDF Download
3568813.3600120.pdf
14 March 2026
Total Citations: 16
Total Downloads: 1551

Latest updates: <https://dl.acm.org/doi/10.1145/3568813.3600120>

RESEARCH-ARTICLE

Taking Stock of Concept Inventories in Computing Education: A Systematic Literature Review

MURTAZA ALI, University of Washington, Seattle, WA, United States

SOUROJIT GHOSH, University of Washington, Seattle, WA, United States

PRERNA RAO, University of Washington, Seattle, WA, United States

RAVEENA DHEGASKAR, University of Washington, Seattle, WA, United States

SOPHIA JAWORT, University of Washington, Seattle, WA, United States

ALIX MEDLER, University of Washington, Seattle, WA, United States

[View all](#)

Open Access Support provided by:

[University of Washington](#)

Published: 10 September 2023

[Citation in BibTeX format](#)

ICER 2023: ACM Conference on
International Computing Education
Research

August 7 - 11, 2023
IL, Chicago, USA

Taking Stock of Concept Inventories in Computing Education: A Systematic Literature Review

Murtaza Ali
University of Washington
Seattle, Washington, USA
mali53@uw.edu

Sourojit Ghosh*
University of Washington
Seattle, Washington, USA
ghosh100@uw.edu

Prerna Rao*
University of Washington
Seattle, Washington, USA
prernar@uw.edu

Raveena Dhegaskar†
University of Washington
Seattle, Washington, USA
raveenad@uw.edu

Sophia Jawort†
University of Washington
Seattle, Washington, USA
sgj3@uw.edu

Alix Medler†
University of Washington
Seattle, Washington, USA
amedler@uw.edu

Mengqi Shi†
University of Washington
Seattle, Washington, USA
shi21@uw.edu

Sayamindu Dasgupta
University of Washington
Seattle, Washington, USA
sdg1@uw.edu

ABSTRACT

Background and context. Concept inventories (CIs) are a widely used tool in STEM education that can help instructors identify specific misconceptions students hold about key concepts. Over the past several years, much research has been published contributing to CIs in computer science education.

Objectives. This research aims to investigate the state of CIs in computer science education through a systematic literature review built around two research questions: 1) What do we know about CIs for computer science today, and 2) Have researchers been able to effectively tackle the 4 challenges specific to computer science CIs outlined by Taylor et al. in their previous 2014 literature review on this topic?

Method. We conducted a systematic literature review focusing on CIs in computer science education. We included research from any country, provided it was published in English between 2004 and 2022. We searched with key terms across four databases (Compendex, Inspec, Education Source, and ERIC) and collated a final list of 65 studies for analysis from 175 initial results.

Findings. From these 65 papers, we identified 33 total CIs in computer science, 12 of which are currently validated. Of the 65 studies, the majority used established techniques for choosing concepts, identifying misconceptions, and/or developing questions, but a few ($n = 4$) reported on successful alternative techniques. We found that although no papers explicitly articulated an objective to tackle

the aforementioned challenges, they are a recurring theme in the literature and have been addressed in various ways.

Implications. At the time of the previous literature review, there were few computer science CIs. Our findings are promising, suggesting that researchers have found effective ways to tackle Taylor et al.'s challenges and that the number of CIs across computer science is gradually increasing. Future work is warranted regarding non-traditional techniques researchers proposed for developing CIs.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; *Student assessment*.

KEYWORDS

concept inventories, systematic literature review, computer science education, computing education

ACM Reference Format:

Murtaza Ali, Sourojit Ghosh, Prerna Rao, Raveena Dhegaskar, Sophia Jawort, Alix Medler, Mengqi Shi, and Sayamindu Dasgupta. 2023. Taking Stock of Concept Inventories in Computing Education: A Systematic Literature Review. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V.1 (ICER '23 V1)*, August 07–11, 2023, Chicago, IL, USA. ACM, New York, NY, USA, 19 pages. <https://doi.org/10.1145/3568813.3600120>

1 INTRODUCTION

A concept inventory (CI) is a standardized set of questions which can be used to measure student understanding of core concepts within a discipline [120]. CIs are usually structured as a set of multiple-choice questions where all answers but one are distractors constructed to correspond to a specific conceptual error. Thus, overall student performance on a CI can directly inform teachers about what aspects of their teaching and curriculum need improvement.

CIs are not unique to computer science (CS) education, and have in fact taken stronger root in other STEM fields over the last 2.5 decades. However, since the late 2000s, CS educators and

*These authors contributed equally to the paper.

†These authors contributed equally to the paper.



This work is licensed under a Creative Commons Attribution International 4.0 License.

ICER '23 V1, August 07–11, 2023, Chicago, IL, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9976-0/23/08.
<https://doi.org/10.1145/3568813.3600120>

researchers have been making steady progress in developing CIs for use in various computing subjects. In this paper, we present a formal systematic review [28] of CI research in CS to address the following research questions:

- What do we know about CIs for computer science today?
- Have researchers been able to effectively tackle the 4 challenges outlined by Taylor et. al. [120] in the previous literature review of computer science CIs?

We adopted a systematic approach to ensure this literature review is as thorough as possible and can be replicated by researchers interested in taking stock of the field at some point in the future. Our work focuses on any and all CIs developed for computing or computing-adjacent (e.g. cybersecurity) fields. We did not limit our systematic review to research that necessarily produces new CIs. We cover the development of CIs, recommendations for building effective CIs, progress toward building specific parts of a CI, validating and administering CIs, and novel ideas researchers have presented as viable alternatives to CIs.

Taylor et al.'s 2014 literature review [120] cataloged existing CIs for computer science and identified challenges unique to computer science regarding the development of CIs. We see a pressing need for a follow-up literature review. This paper reports on all work in the field of computer science CIs published in English between 2004 and 2022. Specifically in the years since Taylor, et. al.'s contribution, we have identified 47 publications concerning the development of CIs for computer science, many of them directly addressing the challenges Taylor, et. al., originally presented.

This research provides a comprehensive overview of the state of computer science CIs today, as well as what the field still needs, establishing a powerful resource for computer science educators and researchers alike.

2 RELATED WORK

2.1 CI Research in STEM Education

In order to situate our work in the broader field of computer science education, we present a brief overview of the origins of CI research and its eventual adoption with computer science education research.

CIs as an educational tool first emerged in 1998, when Hestenes et al. published the Force Concept Inventory (FCI), widely known as the first CI and a model for future ones [59, 120]. This CI focuses on the principles of force in Newtonian mechanics, identified as a crucial topic for introductory physics students. The authors were inspired to develop this tool in light of research that had revealed that introductory physics students across the United States held misconceptions about how force worked based on everyday common sense, and that conventional methods for physics instruction at the time were failing to bring about any meaningful change in these beliefs [49].

Hestenes et al. constructed the FCI to help instructors pinpoint the exact misconceptions individual students held as a precursory step to improving their instruction. The authors emphasize that the FCI is not a test of intelligence, but a survey of belief systems. They outline three potential uses: 1) as a diagnostic tool to identify and classify misconceptions, 2) as a tool to evaluate the effectiveness of instruction, and 3) as a placement exam (this final use is described,

but not recommended by the authors, who maintain that the FCI is not designed to be a test of ability). These suggestions by the FCI authors set a standard, strongly influencing the design and usage of subsequent CIs in STEM education [75].

During the two decades following its inception, the FCI precipitated and guided positive changes in introductory physics education [23]. Education researchers in other STEM disciplines have since worked to develop CIs in an attempt to replicate this education revolution within their own fields, including chemistry [e.g., 82], astronomy [e.g., 9, 63, 77], biology [e.g., 5, 43], mathematics [e.g., 33], statistics [e.g., 117], and geosciences [e.g., 76].

Spurred by the success of CIs in other STEM fields, computer science educators began making progress toward full-fledged CIs in the mid- to late 2000s. In 2008, Goldman, et al., published work identifying student misconceptions for discrete mathematics, logic design, and programming fundamentals [45]. Other researchers followed suit, contributing to Goldman, et al.'s work to develop this set of misconceptions [68]. Work was additionally done to develop CI-like assessments that were only usable in restricted settings due to lack of testing [27].

However, it was not until the early 2010s that the first truly validated and empirically tested CIs for computer science—a language-independent assessment for CS1 (FCS1), and a digital logic CI (DLCI) resulting from the preliminary work above [54, 124]—were published. At the time of Taylor et al.'s literature review of computer science CIs in 2014, these were the only validated computer science CIs in existence [120].

2.2 Systematic Literature Reviews in Computer Science Education

An established way to evaluate the breadth of research within any given field is a literature review. A literature review can serve many purposes, but a primary one is to review and evaluate the state of knowledge within the field to identify potential gaps or areas for improvement [72]. Although it is most common in medicine and associated fields [e.g., 20, 62, 74, 106, 110], it is becoming increasingly utilized outside those spaces in fields such as teaching pedagogy [e.g., 2, 104], machine learning [e.g., 13, 100], accessibility [e.g., 17, 80], and others. Such literature reviews might be systematic and follow rigid protocols that are later replicable for building multiple studies (e.g. Cochrane Reviews [60]), semi-systematic to explore themes within a field or subfield [113], or integrative toward forming a theoretical conceptualization of a specific idea within a field [125].

Within the field of CS education, Austing et al. [7] conducted one of the first literature reviews to explore the status of and potential revisions of the Curriculum '68, the foundational document for CS education in the US. This led to the publication of the iterated Curriculum '78 [7] and established a standard of literature reviews as an effective tool for identifying opportunities to iterate and improve upon practices.

Since then, there have been several literature reviews conducted to explore various subfields, such as difficulties in learning programming [21], pedagogical trends at the CS1 level [92], patterns of shifts in goals within CS classrooms [121], effectiveness of intelligent tutoring systems to supplement classroom learning [24],

importance of self-regulated learning using e-tools [40], and computing education in non-formal settings [41], to name a few. One review, published in 2019, looked broadly at measurements in computing education research [81]. Our work indirectly builds upon this, with a specific focus on CIs.

Literature reviews within this field are usually systematic or semi-systematic, and provide good coverage of their respective subfields. Many of these have left lasting legacies of being among the most prominent voices in driving change and moving the field forward, and we hope to add to this legacy.

2.3 Key Ideas and Techniques in CI Research

This section introduces key terms and ideas related to CI research that are repeatedly mentioned in the body of this paper. An understanding of these ideas is important to fully understand the findings of our systematic review.

2.3.1 Delphi Process. The Delphi process is a research method that has been repeatedly referenced and used in computer science CI research since it was popularized in this field by Goldman et al. [45]. The Delphi process refers to a structured multistep process that uses a group of experts to reach a consensus on the main points or contents of a research topic [25]. Effective panels aim to bring together as many experts as possible of different ages, genders, and regions to ensure the diversity of the group. Seeking consensus among experts helps eliminate the natural bias that comes from each expert's experiences [84]; additionally, the structured, multistep nature of the process prevents excessive influence from a limited number of experts [93]. This process generally includes 4 phases: 1) concept identification, 2) initial rating, 3) negotiation, and 4) final rating [45], with anonymous feedback between rounds to ensure that an expert's reputation within a field does not bias the results [46]. The ratings resulting from the Delphi process are then used to finalize a list of concepts and begin the next phase of developing a CI: identifying misconceptions.

2.3.2 Think-Aloud Interviews, Classsourcing, and Crowdsourcing. Once concepts are chosen for a CI, researchers must identify associated misconceptions to develop distractor answers before building the final CI. Prevalent methods for identifying student misconceptions include think-aloud interviews, classsourcing, and crowdsourcing.

Think-Aloud Interviews. In a think-aloud interview, participants are asked to articulate their thinking process verbally and in detail to the researcher while performing a task of interest [15]—in this case, solving problems related to a CI's concept list. Think-aloud interviews are commonly used during CI development for computer science [e.g., 26, 31, 68]. Their utility arises from research which supports the idea that “verbal reports, elicited with care and interpreted with a full understanding of the circumstances under which they were obtained, are a valuable and thoroughly reliable source of information about cognitive processes” [34, p. 247]. In such an interview session, participants explain their understanding of the CI questions and the process by which they think about the answers. This helps researchers identify common student misconceptions [26] and ensure that the questions in the study are understood without bias [89].

Classsourcing. Classsourcing involves reviewing previous classroom materials, such as exams, assignments, and quizzes to identify common student misconceptions for a CI [99]. Classsourcing can uncover unique perspectives through students' incorrect examples by providing great insight into potential misconceptions that may not be revealed with experts [99].

Crowdsourcing. Crowdsourcing is similar to classsourcing, but it does not draw on previous knowledge from the classroom. Instead, it looks online to access the solutions of a distributed network of individuals [16]. Through crowdsourcing, researchers can collect a high frequency of diverse opinions from individuals around the globe. Crowdsourcing is an additional option for CI developers who need to identify common misconceptions about a concept.

2.3.3 Validation Techniques. The final step in producing an CI is to verify that its questions are valid and reliable according to established education standards [6] to ensure it is an unbiased estimator of student conceptual knowledge [54]. Validation is an iterative process which aims to improve and refine the questions on a CI [136].

Item Response Theory (IRT). IRT is a theory that may be used in the CI Validation process. IRT techniques target statistical analysis of validity claims [124], and serve as an effective measure of the difficulty of each CI question and the degree of differentiation among them [89]. IRT helps to understand the relationship between the difficulty of the question and the knowledge of the learner by placing the learner and the question on the same continuum [136].

Classical Test Theory (CTT). CTT is another test theory used in the validation phase. It assumes that a learner's score on a test is a combination of their true score (expected score, with no errors in measurement) and the errors in score measurement [136]. However, compared to IRT, CTT analyzes the problem as a whole, and so is prone to larger sample effects. Additionally, the measure of students' knowledge of the research questions and of the questions themselves are combined in CTT, increasing the error value. This has resulted in researchers preferring IRT over CTT in terms of validation [133, 137].

2.3.4 Cronbach's alpha. Cronbach's alpha is a widely known reliability measure that checks the internal consistency of tests and measures—how closely related they are to each other [114]. There is no universally accepted Cronbach's alpha value to indicate a reliable assessment, but if a test is used in a high-stakes situation, such as assigning grades to students, the convention within STEM education research is that its value should be at least 0.7 or 0.8 on a scale of 0 to 1 [118]. Cronbach's alpha is often used to screen concepts for a test. For example, if eliminating a test item increases the reliability of that test, then the item may be of poor quality and removal from the test may be warranted [98].

2.4 Summary of Taylor et al.'s 2014 Literature Review

As this work builds on a previous review of the literature on computer science CIs by Taylor et al. [120], in this subsection, we summarize the key points of this prior work. We pay special attention

Table 1: Unvalidated CIs studied by Taylor et al. [120]

Unvalidated CIs
Algorithms and Data Structures (Paul and Vahrenhold [91])
Binary search tree & hash tables CI (Karpierz and Wolfman [69])
Computer Architecture (Porter et al. [94])
Operating Systems (Webb and Taylor [129])
Validated CIs
Digital logic CI (Herman et al. [54])
CS1 language independent CA (Tew and Guzdial [124])

to the challenges proposed by Taylor et al. to provide context for our second research question.

2.4.1 Concept inventories in engineering and other sciences. To set the stage for their discussion, Taylor et al. begin by tracing the history of CIs in STEM fields, starting with the lasting impact of the Force Concept Inventory [59] and how it led to the adoption of CIs in other disciplines. The spirit and content of this section are similar to our own discussion above, so we do not repeat it here.

That said, we do note the authors' observation that while the CIs they studied were generally developed with similar overarching protocols (as detailed in the previous section), the specifics often varied from one CI to the next [78]. This observation held true in our own review as well.

2.4.2 Concept inventories in computer science. Taylor et al. acknowledged that CI development has been slow within computer science, noting that even the two validated CIs at the time—the FCS1 [124] and the DLCI [54]—were not in widespread use. In particular, they expressed concern that while many researchers and educators were introducing new pedagogical innovations, the CS education community as a whole lacked standardized assessment tools to measure the effectiveness of these measures. In this regard, the findings of our systematic review are encouraging, as they show that much progress has been made in CI development within computing education.

The authors then discuss preliminary and in-progress CI work at the time for CS1 [46, 68], CS2 [26, 69], computer architecture [95], and operating systems [96, 129]. This work is encompassed in our first research question, and we discuss it in greater detail in our findings below.

The authors also briefly mention informal assessments in CS1, pointing out poor student results across the board as part of their argument that student misconceptions are widespread in computer science and thus research in CIs is warranted. We omit a more detailed discussion of these informal assessments here, as they are not within the scope of our systematic review.

In summary, Taylor et al.'s survey of CIs in computing education revealed a total of 6 CIs in 2014, 2 complete and validated, and 4 in progress. Table 1 below displays these CIs.

2.4.3 Challenges for computer science and groundwork for future CI development. When Taylor et al. [120] published their literature review of computer science CIs in 2014, they identified 4 challenges specific to computer science that future educators and researchers should consider:

- (1) Pre-test limitations
- (2) Programming language dependence and designing robust CIs
- (3) Difficulty assessing skills
- (4) CIs vs. comprehensive assessments

We provide a detailed description of each of these challenges below to situate our second research question.

Pre-test limitations. In other STEM fields, CIs are often administered as pre-tests and post-tests so that a student's improvement can be measured over a course offering [59]. However, this is difficult to do in a computer science context because students generally do not enter computer science with pre-existing misconceptions. Some early CI researchers cited by Taylor et al. noted that computer science tends to use a highly specialized vocabulary, and that student misconceptions in computer science likely originate from the instruction itself as opposed to having been formed previously [95, 123, 129]. This differs from other STEM education settings. For instance, it is reasonable to assume that a student's experiences prior to taking an introductory physics class may have led to major misconceptions about how Newtonian force works, but it is not reasonable to make the same assumption about a student's thoughts on loops and conditionals before taking an introductory computer science class.

Programming language dependence and designing robust CIs. As a relatively young field, the content and underlying technology of computer science are significantly more fluid than other fields. When the FCI was designed, it was almost naturally robust due to the fact that Newtonian force is an extremely stable subject which can be tested purely conceptually. On the other hand, many computer science concepts are taught in a specific programming language. Furthermore, although programming language popularity may be stable for several years at a time, it most certainly does shift across decades. This timeline may seem fine at first glance, but it is an important issue to consider. There is a reason why the FCI is still relevant today, 30 years after its initial development. One could imagine getting around this issue by using pseudocode, but Taylor et al. aptly mention some limitations of this technique [120]. It is much harder to validate a language-agnostic CI, as researchers must ensure that the provided pseudocode makes sense and maps effectively to actual programming languages. Additionally, within computer science, the choice of language can impact the underlying concepts and paradigms as well: A CS1 course in Python will likely

take an imperative approach to programming, whereas the same course in Scheme will likely take a functional one.

Difficulty assessing skills. Within computer science, it may be necessary to differentiate between *concepts* and *processes*. The term process here refers to skills such as code analysis and program design, which are not computer science concepts in the traditional sense but are still essential for learning. Taylor et al. raise an important issue: Can processes be considered as concepts or as containing concepts within them, or are processes strictly distinct from concepts [120]?

CIs vs. comprehensive assessments. This challenge relates to the recommendations Hestenes et al. gave upon publishing the FCI in physics. One way the FCI has been used is to evaluate the effectiveness of physics instruction [48]. Taylor et al. noted that some CS researchers have also argued for such usage of CIs, but others have argued that because CIs are not comprehensive assessments, they are not well suited for this purpose.

Taylor et al. conclude by emphasizing that despite the many challenges of computer science CI development, the research community at the time had laid a strong foundation for moving forward. They make particular note of the progress in identifying concepts and misconceptions as groundwork for future CIs, and end with a hopeful outlook on the future of computer science CIs.

3 METHODS

3.1 Data Collection

Toward our stated objective of conducting a comprehensive review of the field of computer science CIs, we decided to conduct a systematic literature review, pursuant to other research with similar objectives [e.g., 2, 13, 41, 80]. To curate the data set of papers to review, we began with consultations with the Head Engineering librarian at the first author's university, seeking advice on potential databases to examine. Given our interest in studying papers in STEM and computer science education and our prior experience in reading such work, we expected to encounter papers from conferences and journals under umbrellas such as the ACM (specifically, venues sponsored by the ACM Special Interest Group on Computer Science Education, i.e., SIGCSE), the IEEE, and several other venues that commonly publish work in the space. We began by looking at the Engineering Village search platform, based on the librarian's recommendation, and on the fact that it indexes over 3,500 journals and conference proceedings, including several major ACM, SIGCSE, and IEEE-sponsored venues such as the ones we are interested in. Within the Engineering Village platform, we chose the Compendex and Inspec databases, which cover venues of interest such as the SIGCSE Technical Symposium on Computer Science Education (SIGCSE-TS), the ACM Conference on International Computing Education Research (ACM ICER), and the ACM Conference on Innovation and Technology in Computer Science Education (ACM ITiCSE). To ensure complete coverage of the field, we also included the Education Source and ERIC databases through the EBSCOhost platform. We believe that the databases chosen are appropriate for this study due to their collective coverage of their space and their effective metadata management which facilitates successful searches.

The list of databases consulted, along with the corresponding lists of venues/titles indexed by these databases, is provided in Table 2.

We additionally consulted with the librarian to determine the most appropriate search keywords for our review. Drawing on the librarian's advice and a collective discussion, we queried these databases with five keywords: 'concept inventory', 'concept inventories', 'computer science', 'computer science education', and 'CS.' In forming the query, we treated these keywords as two sets, since they are related to each other within each set. With the Engineering Village databases, we initially searched only the titles, abstracts, and keywords of papers (query ending in WN KY), in order to avoid a large number of results that only mention our search terms without focusing on them. However, this search output only 78 results, so we pivoted to searching all search fields ¹ available on the platform (query ending in WN ALL), which output 148 results. These results were manually screened. Therefore, the formal expression for our queries is as follows:

3.2 Screening and Quality Control

Based on our database search, which returned 175 papers (Compendex: 95, Inspec: 53, Education Source: 11, ERIC: 16), we identified a total of 72 unique and relevant papers on which to conduct our literature review. Given the young but growing nature of the field of CIs in computer science, this number seemed plausible to us. We conducted a few confirmatory searches using the same query in Google Scholar and found papers within our list appearing there too, which solidified our confidence in this list. We did not use Google Scholar as a primary search database, as research has shown it to be an inappropriate principal search system for systematic reviews [47].

We then conducted quick read-throughs of the paper titles and abstracts within our list, as well as examination of the metadata such as date and venue of publication. We intended to screen out any paper that was not in English, since it is the only common language in which all members of our team were fluent, and we wanted to avoid losing meaning in translation [37]. Furthermore, we wanted to avoid papers published prior to 1990 and those published in predatory journals (according to Beall's List²). Since no papers from predatory journals appeared in the database search results, we did not need to exclude them. Finally, we excluded papers that were not related to STEM or CS, or were otherwise confusing or unclear to us. These inclusion/exclusion criteria are also detailed in Table 4. Given the subjectivity of the final criteria, every paper was subjected to this screening process by two researchers independently, with ties being broken by the first author. Through this screening process, we arrived at a final list of 65 papers to conduct this systematic literature review, all of which are listed in the References section, with the title of the article in bold font. This process is depicted via a PRISMA diagram—a diagram that depicts the flow of information through the different phases of a systematic review, including the number of records identified, included, and excluded [86]—in Figure 1.

¹Subject/Title/Abstract, Abstract, Author, First Author, Author Affiliation, Title, CODEN, Conference Information, ISSN, Publisher, Source Title, Controlled Term, Uncontrolled Term, Country of Origin, Funding Number, Funding Sponsor, and Funding Information

²<https://bealllist.net/>

Table 2: Databases used for our search queries

Platform	Database	List of venues/titles indexed
Engineering Village	Compendex	https://www.elsevier.com/_data/assets/excel_doc/0007/56392/CPXSourceList.xlsx
	Inspec	https://www.theiet.org/publishing/inspec/inspec-content-coverage/
EBSCOhost	Education Source	https://www.ebsco.com/products/research-databases/education-source
	ERIC	https://www.ebsco.com/products/research-databases/eric

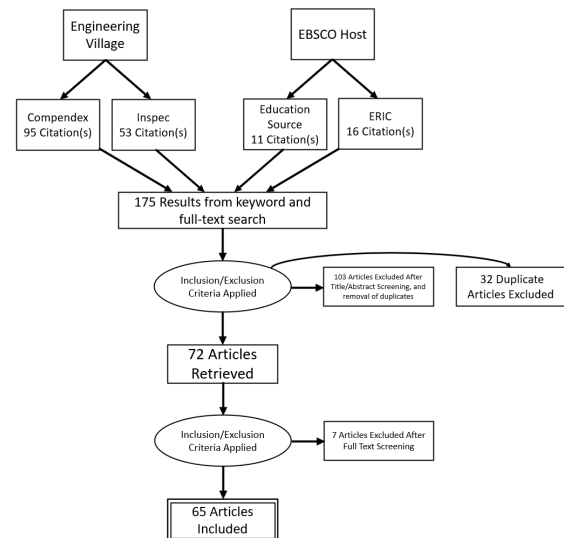
Table 3: Formal Expression of Queries Used

Database	Query
Engineering Village (Compendex and Inspec)	((("Concept inventory" or "concept inventories") WN KY) AND ((("computer science" or CS) WN KY)) ((("concept inventory" or "concept inventories") WN ALL) AND ((("computer science" or "CS") WN ALL)))
EBSCOhost (ERIC and Education Source)	("concept inventories" OR "concept inventory") AND ("computer science education" OR CS)

Overall, the data set encompasses 65 papers, with an average page count of 9.7 (19 pages on average for 17 single column papers, 6.4 pages on average for 46 double column papers, 5.5 pages for 1 triple column paper, and 1 unknown due to the paper being inaccessible) spanning 23 publication venues with the following breakdown:

- SIGCSE (Computer Science Education) - 14 papers (21.5%)
- FIE (Frontiers in Education Conference) - 10 papers (15.4%)
- ITiCSE (Innovation and Technology in Computer Science Education) - 7 papers (10.8%)
- Computer Science Education (Journal) - 5 papers (7.7%)
- TOCE (Transactions on Computing Education) - 5 papers (7.7%)
- ASEE (American Society for Engineering Education) - 3 papers (4.6%)
- IEEE Transactions on Education - 3 papers (4.6%)
- ICER (International Computing Education Research) - 2 papers (3.1%)
- NCS (National Cyber Summit) - 2 papers (3.1%)
- ACE (Australasian Computing Education Conference) - 1 paper (1.5%)
- DATA (International Conference on Data Science, E-learning and Information Systems) - 1 paper (1.5%)
- EASEAI (Education through Advanced Software Engineering and Artificial Intelligence) - 1 paper (1.5%)
- EDUCON (Global Engineering Education Conference) - 1 paper (1.5%)
- Eurasia Journal of Mathematics, Science and Technology Education - 1 paper (1.5%)
- FECS (Frontiers in Education Computer Science & Computer Engineering) - 1 paper (1.5%)
- FM (Formal Methods) - 1 paper (1.5%)
- ICCSE (International Conference on Computer Science & Education) - 1 paper (1.5%)
- ISSEP (Informatics in Schools: Situation, Evolution, and Perspectives) - 1 paper (1.5%)

- JEET (Journal of Engineering Education Transformations) - 1 paper (1.5%)
- Journal of Educational Technology Systems - 1 paper (1.5%)
- REM (Research and Education in Mechatronics) - 1 paper (1.5%)
- SE (Southeast Conference) - 1 paper (1.5%)
- Security & Privacy - 1 paper (1.5%)

**Figure 1: PRISMA Diagram outlining the selection of papers for the current review**

3.3 Conducting the Literature Review

We started our systematic review of the literature with an open coding of the 65 papers within our dataset, pursuant to similar

Table 4: Inclusion and Exclusion Criteria for Screening Studies

Category	Inclusion Criteria	Exclusion Criteria
Publication Language	English	Any language other than English
Grey Literature	Pre-prints on ArXiv, that have been accepted to be published	PhD dissertations and Masters' theses
Publication Date	Published in 1990 or later	Published before 1990
Publisher	Peer-reviewed journals or conference proceedings	Published by predatory publishers as listed in Beall's List
Population of Interest	Computer Science (CS) learners of all ages	None
Study Topic	Usage of concept inventories in computer science, student misconceptions in computer science, assessments in computer science, development of concept inventories in STEM fields	Unrelated to computer science or STEM, unclear or unfocused research topic
Page Length	Any page length	None

studies [e.g., 13, 80]. We prepared a shared Google Sheet with all papers and randomly assigned ~9 papers to the first 7 researchers. At this open coding stage, we read the papers in their entirety and took extensive notes. After the first round of coding, the researchers met to consolidate their notes into a series of questions, which are as follows:

- (1) What was the focus of the paper (identifying student misconceptions, developing and designing a new CI, evaluation of an existing CI, etc.)?
- (2) Who is the intended audience for the CI/what is the research setting (CS1/CS2/Advanced CS/High School or lower/Instructors/etc.)?
- (3) What were the concepts that the CIs were based on/what was the specific subject matter of the CI (Intro CS, Cybersecurity, Digital Logic, etc.)?
- (4) Does this paper make use of a Delphi Process to develop a CI, or otherwise discuss the potential for its use? If yes, briefly describe how Delphi process is mentioned and comment on the diversity of academics in the panel, specifically with regard to teaching and content expertise.
- (5) Do the authors differentiate between concepts (loops, objects, etc.) and processes (program design, code analysis, etc.), or discuss this in any way?
- (6) What programming language was used to define a CI? If a specific programming language was used, do the authors discuss strategies for making the CI portable or language agnostic?
- (7) Did the paper have a section discussing funding source?
- (8) Does this paper conduct a qualitative or quantitative analysis? If yes to either of these, what is the relevant data that we can extract?
- (9) Do the authors discuss the use of CIs as pre-tests, post-tests, both, or neither? If pre-tests are used, how are they compared (e.g., raw gain, normalized gain, gain of averages, etc.)?
- (10) Do the authors iterate upon their CI? Is the CI validated?
- (11) Do the authors classify concepts as fundamental to understand, hard to understand, etc.? If yes, how are they classified into these categories?

We chose these questions collectively by considering what specific information from the literature would enable us to answer our two overarching research questions. Questions 1-4, 6-8, and 10-11 were chosen to discover specific aspects of what we know about computer science CIs today. Questions 1, 5-6, and 9 were chosen to align with Taylor et al.'s challenges [120].

Having arrived at this set of questions, the first 7 authors were assigned another random set of 9 papers from within the data set to verify whether these questions were sufficient and applicable, and whether additional questions needed to be added. This second stage of open coding yielded no change to the aforementioned list of questions.

Having determined a set of questions with which to analyze our data set, we applied them for the closed coding process. Each of the the first 7 authors coded approximately 18 papers—9 papers for which they were the first or primary coder, and 9 papers where they were the second or confirmatory coder. All ties were broken by the first author, and for ties where he himself was involved, by the senior author.

Following closed coding, the team met to conduct a thematic analysis [113] of the result of applying our questions to the dataset. Additionally, we extracted quantitative metrics and descriptive statistics relevant to the questions above.

4 RESULTS

4.1 RQ1: What do we know about CIs for computer science today?

We structure this subsection in the same way a traditional paper on CIs would detail the development and validation process of its CI [1]. Within each section, we provide the relevant results from our systematic review to convey the state of concept inventories in computer science today.

4.1.1 Who, what, how? Student Profile, Topic, and Language. Among the 65 papers identified in our database search, the most common study setting was by far undergraduate students (62 papers). Other settings included middle grades, graduate school, and high school, each of which was covered by 1, 1, and 2 papers, respectively (one paper [131] covered multiple settings). One of the few

Table 5: Breakdown of Different Topics Covered by Studies

Topics Covered	#	Citations
CS1	36	[3, 8, 11, 12, 18, 19, 22, 26, 35, 38, 39, 45, 46, 51, 64–71, 73, 79, 83, 102, 103, 107, 115, 116, 120, 126, 127, 131, 132, 136]
CS2	7	[36, 50, 69, 119, 120, 130, 134, 135] ³
Cybersecurity	6	[85, 88, 98, 109, 111, 112]
Digital Logic	6	[53, 54, 56–58, 109]
Computing Theory	4	
Formal Reasoning	2	[30, 61]
Finite State Machines	1	[52]
Theory of Computation	1	[87] ⁴
Operating Systems	2	[87, 129]
Basic Electronics	2	[55, 105]
K-12 CS	1	[101]
Compiler Construction	1	[87]
Signals & Systems	1	[87]

Table 6: Breakdown of Different Languages Used by CI Studies.⁵

Programming Language	#	Citations
Java	12	[11, 18, 39, 50, 51, 66, 68, 79, 83, 103, 131, 132]
C	8	[8, 11, 19, 38, 50, 70, 83, 115]
Python	6	[64, 65, 73, 79, 116, 127]
C++	4	[8, 70, 131, 132]
Scheme	2	[39, 116]
MATLAB	2	[8, 70]
C#	2	[131, 132]
RESOLVE	1	[30]
Eiffel	1	[83]
Language Agnostic	7	[12, 46, 69, 98, 101, 130, 136]

papers we identified which focused on younger students explicitly stated there is a “significant need for a more general middle grades CS concept inventory assessment” [101, p. 1].

Notwithstanding the focus on the university setting, current CI work in computer science still covers a wide range of topics. We identified studies for a total of 12 different topic areas. Table 5 shows a detailed breakdown of the different topics covered. We additionally note a wide range of different programming languages used across CIs, with some also being language-agnostic or not requiring active programming (e.g., discrete mathematics). This breakdown is displayed in Table 6.

Overall, there is meaningful spread among topics and languages in computer science CI research, but it may be apt to direct research efforts toward younger student profiles. This is an excellent focus area for future work.

³We include Farghally et al. [36] within the list of papers at the CS2 level, even though they refer to their work at the ‘post-CS2’ level, because we find their work to fall within the definition of CS2 we have used in this paper [32].

⁴After our analysis, we noticed that the ACM has issued a “Notice of Concern” for the conference this paper was published in, and has strongly suggested that papers from the conference be not cited till further determinations are made. As we had already concluded our analysis, we decided to keep the paper in our database.

⁵Note that this table shows all *studies*, including those that are in progress, and that some CIs have been developed in multiple languages. As such, the sum total here is higher than the total number of existing languages.

4.1.2 Choosing Concepts, Identifying Misconceptions, and Building the CI. A total of 22 papers in our analysis discussed the process of choosing concepts for a CI, either as the primary subject of the paper or as part of the larger CI development process being reported. The Delphi process is the most popular method of choosing concepts, with 17 studies employing this technique. Even studies that didn’t use a proper Delphi process often incorporated some variation of soliciting expert opinions. In one paper, the developers of a basic data structures CI detail how they still used a group of experts to reach a consensus but did so without a formalized Delphi process [119].

4 papers proposed alternatives to the expert-driven Delphi process (and its variations), with some pointing out that despite its popularity, it has some notable weaknesses. Namely, the set of concepts identified by discussion among domain-specific experts still needs to be empirically validated after the fact, which can be time consuming and cumbersome. These alternative techniques, designed to be data driven and efficient, are discussed further in Section 4.3 below.

To identify misconceptions after choosing concepts, researchers primarily used two methods: 1) analysis of past student exams and 2) think-aloud interviews with open-ended questions about the concepts. Out of the 33 total CIs identified in our review (including those that may not yet be fully validated), 6 identified misconceptions by analyzing past exams, 14 did so using think-aloud interviews, and 4 did so with a combination of both techniques.

Table 7: Validated CIs

Validated CIs
Binary search tree & hash tables CI (Karpierz and Wolfman [69])
First-year computer science CI (Vahrenhold and Paul [126])
Digital logic CI (Herman et al. [57])
Basic recursion CI (Hamouda et al. [50])
Second CS1 (SCS1) (Xie et al. [136]) ⁶
Java array CI (Saarinen et al. [103])
Middle-grades computer science CI (MG-CSCI) (Rachmatullah et al. [101])
Basic data structures inventory (BDSI) (Taylor et al. [119])
Prerequisite and advanced CS CI (Nelson et al. [83])
Cybersecurity CI (Sherman et al. [111])
Object-oriented programming CI (Julie et al. [66])
Cybersecurity CI (Poulsen et al. [98])

4.1.3 Validating the CI. Though this step is traditionally referred to as CI validation, it generally encompasses two steps: ensuring the CI is both *valid* and *reliable*. Our analysis identified a total of 12 validated computer science CIs in existence today, listed in Table 7. Among these 12, 2 were statistically validated using IRT, and 6 using both IRT and CTT. Of these CIs, 5 measured internal reliability, 3 using the popular Cronbach’s alpha metric, 1 using multiple methods including Cronbach’s alpha, and 1 using an alternative method, the Kuder-Richardson Formula 21 (KR-21) [54, 138]. 21 CIs are currently in-progress, or unvalidated, as listed in Table 8.

Our findings here underscore the great progress computer science educators and researchers have made in developing CIs over the last decade. The total number of validated CIs has grown by a considerable 10, up from 2 in 2014 to 12 today.

4.2 RQ2: Have researchers been able to effectively tackle the 4 challenges specific to computer science CIs outlined in Taylor et al.’s 2014 review?

As we discussed in detail above, Taylor et al. identified 4 challenges specific to computer science CIs that future educators and researchers should consider:

- (1) Pre-test limitations
- (2) Programming language dependence and designing robust CIs
- (3) Difficulty assessing skills
- (4) CIs vs. comprehensive assessments

Before beginning our analysis, we hypothesized that the CI papers published after 2014 would directly deal with the above issues. We identified 47 studies after Taylor et al.’s review, and although none directly mentioned that they dealt with or addressed one or more of the challenges above, our analysis revealed them to be a prominent and recurring theme.

In each subsection below, we briefly review the initial challenge outlined by Taylor et al., along with the associated findings from

our analysis. Table 9 displays which studies addressed which challenge(s).

4.2.1 Pre-test limitations. Because of the human-made nature of computer science, it is unlikely for students to enter courses with existing misconceptions, which in turn makes it difficult to take reliable pre-tests [95, 123, 129].

This is a particularly interesting point because our analysis suggests it did not come to light until after concrete computer science CI research began. One of the earliest papers identified in our review mentioned a hope to “replicate the physics education revolution in computing education” [45, p. 1] and went on to explicitly state that a “CI is typically administered as both a *pre-test* at the beginning of a course and a *post-test* at the end, to measure the ‘gain’ resulting from instruction, on a select subgroup of representative topics” [45, p. 1]. This indicates that some early CI researchers who had not yet reached the point of building a CI (the above paper only reports on an initial Delphi process to identify concepts) still expected to be able to precisely mimic their usage in other STEM fields. This feeling shifted as early computer science CIs were developed and validated. When a subset of authors on this early paper went on to develop the digital logic CI, they detailed how they tried to use their CI as a pre-test, but had to discard the results due to unacceptable reliability and high standard error rates [57].

Out of the 47 studies in our review published *after* Taylor et al.’s literature review, 6 used a CI only as a post-test, and 16 used a CI as a pre-test and a post-test. Among the 16, 1 did not formally administer a CI and instead conducted pre- and post-interviews about misconceptions [66]; 1 tested formal reasoning (important for computer science but more general than the programming topics for which pre-tests are difficult) [61]; and 2 administered the pre-test at a point in time when students had already received instruction on the relevant material [39, 44]. 5 papers in total discuss limitations of the use of CIs as pre-tests in CS [19, 57, 65, 98, 136]. One of these simultaneously discusses the growing relevance of CIs as a pre-test in CS, stating—“computer science topics are increasingly part of the K-12 curriculum. Because students will increasingly have discovered basic programming concepts before the university, the CI as a pre-test is becoming more and more relevant” [65, p. 3]. The

⁶We cite Xie et al. here because their paper which described and validated the SCS1 appeared in our systematic search, whereas the original SCS1 paper [89] did not. This is discussed further in our limitations section (§5.4).

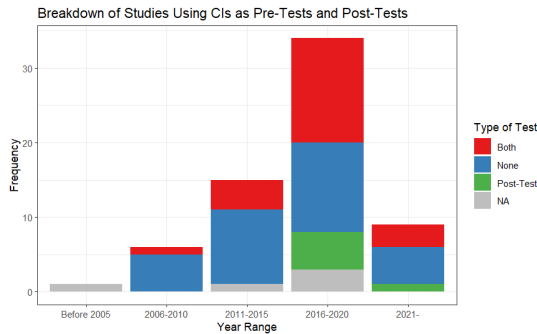
Table 8: Unvalidated or Incomplete CIs

Unvalidated or Incomplete CIs
Discrete math, programming fundamentals, and logic design (Goldman et al. [45])
Medium-scale integrated circuits CI (Herman et al. [55])
CS1 Perceptions of parallelism (PoPS) (Rague [102])
Operating systems CI (Webb and Taylor [129])
Algorithm Analysis CI (Farghally et al. [36])
Formal reasoning CI (Hollingsworth and Sitaraman [61])
Introductory programming CI (Caceffo et al. [19])
CS1 CI (Stephens-Martinez et al. [116])
Robotics CI (Gerndt and Lüssem [44])
Operating systems, theory of computation, compiler construction (Pamplona et al. [87])
Cybersecurity CI (Ngambeki et al. [85])
CS1 CI (Bari et al. [10])
Basic electronics CI (Sachan et al. [105])
Cybersecurity CI (Sherman et al. [112])
Variables in CS CI (Julie and Bruno [64])
Variables, conditional statements, and functions in CS CI (Julie and Bruno [65])
Matlab programming CI (MCS1) (Barach et al. [8])
Tangible-augmented CI (Julie et al. [67])
CS1 CI (Henry and Dumas [51])
Problem-based learning CS1 CI (Souza and Bittencourt [115])
Introductory programming CI (Kecskemety et al. [70])

Table 9: Number of Studies Which Addressed Taylor, et al.'s Challenges

Taylor, et al.'s Challenges	#	Citations
Programming Language Dependence	17	[8, 12, 18, 19, 39, 45, 46, 51, 68, 79, 83, 101, 103, 116, 119, 130, 136]
Pre-Test Limitations	6	[19, 52, 57, 65, 98, 136]
Difficulty Assessing Skills	6	[3, 36, 46, 67, 87, 119, 132]
CIs vs Comprehensive Assessments	1	[46]

use of CIs as both a pre-test and post-test, as opposed to just as a pre-test, seems to support this claim, as visualized in Figure 2.

**Figure 2: Breakdown of Studies Using CIs as Pre-Tests and Post-Tests**

However, our analysis revealed that even though CIs may be used as pre-tests, there are still struggles and inconsistencies with doing this. One paper which aimed to extend validation of the SCS1 [136] through IRT found that although it had previously been used

as a pre-test, this usage may not have been appropriate. Specifically, the authors state, “Our results suggest that prior work using it as a pre-test may have risked encountering floor effects because of the test’s difficulty. At its current level of difficulty, the SCS1 may be more appropriate only as a post-test” [136, p. 704].

Our findings suggest that computer science CIs have been used as both pre-tests and post-tests, but more effectively as the latter than the former. Exceptions to this appear to be cases in which students are being “pre”-tested on material they should already know from a previous class or when the concepts being tested are more abstract in nature than traditional programming. This is in line with Taylor, et al.’s predictions and indicates a notable difference between CIs in computer science and other STEM fields.

4.2.2 Programming language dependence and designing robust CIs.

The reality that computer science is a young field with fluid content, coupled with the fact that courses with similar concepts are often taught in different programming languages, led Taylor et al. to raise an important challenge: How do we design long-lasting and comprehensive CIs for computer science [120]? Our systematic review revealed promising findings with respect to this challenge.

We classified CIs into three categories: 1) those for which programming language is irrelevant (e.g. formal reasoning), 2) those

which adopt a language-agnostic approach, and 3) those which utilize a single language. 12 out of 33 CIs we identified involve computing-adjacent topics that are not reliant on a programming language, and an additional 5 do not mention the use of a specific programming language or pseudocode. Of the remaining 16, 3 are language-agnostic and 13 are designed in a specific programming language.

One study evaluated a previously validated language-agnostic CI for basic data structures [130]. The authors administered the CI to 1,963 students with varying backgrounds in programming language (Python, Java, or C) and reported an encouraging finding: Despite the language differences, students rarely indicated being confused by the pseudocode, suggesting the pseudocode maps effectively to real programming languages. However, they did acknowledge their findings are limited to languages taught in an imperative programming paradigm.

Among language-specific CIs, a technique some researchers used to address this portability issue was replicating the same fundamental CI across multiple languages. For example, Caceffo, et al., proposed a method for taking an existing CI for introductory programming in C and Python and converting it to Java [18, 19]. This approach, while more involved than building a single language-agnostic CI, provides the most direct method we discovered for ensuring the robustness and generality of a CI in computer science.

Overall, computer science CI researchers appear to be at least implicitly aware of this challenge, with some actively taking steps to ensure the CIs they develop are robust. We did not identify methods beyond the two above for achieving this goal. Future work should focus on 1) developing novel methods for ensuring robustness of CIs as well as 2) comparing different methods to empirically determine the associated benefits and drawbacks.

4.2.3 Difficulty assessing skills. In their original literature review, Taylor et al. noted a potential distinction between *concepts* and *processes*, using the term process to encompass skills such as code analysis and program design—essential skills in a computer science course that are not concepts in the classical sense [120]. Specifically, the authors posed the issue of whether these two ideas truly were distinct or if they could be treated as the same for the purposes of computer science CI design.

In our review, we did not identify any study that explicitly differentiated between concepts and processes, but we did find studies that implicitly used processes as part of their CI research. In particular, we marked 5 CIs as including what Taylor, et. al. defined as processes, as listed in Table 10.

For example, one study directly states, “The concepts we considered relate to code analysis (program tracing, debugging, and expression evaluation), but not synthesis (code design, writing) or evaluation (e.g., refactoring) skills as defined in the Bloom’s taxonomy of the cognitive domain” [131, p. 24], indicating that the authors simply consider code analysis—listed as an example of a process above—to be a concept.

A second study [79] came close to drawing this distinction by formulating a hierarchical structure of concepts. In particular, the researchers developed a master list of 12 overarching concepts, each of which consists of more specific “sub-concepts” within it.

One such overarching concept is “Programming Process,” with sub-concepts such as reading code, designing programs, and testing. However, we do not count this as a true differentiation, as the authors’ wording implies that they consider “Programming Process” to be a concept rather than its own entity.

Our findings indicate that computer science CI researchers tend to consider processes directly as a type of concept without feeling the need to formulate a precise distinction between the two. There is nothing to suggest this has been problematic or detrimental in any way, but there are also no studies empirically confirming or refuting this practice.

4.2.4 CIs vs. comprehensive assessments. When Hestenes, et al. published the FCI in physics, they noted one potential use of it could be to evaluate the effectiveness of physics instruction [48]. In their literature review, Taylor et al. mentioned mixed opinions on this matter among computer science CI researchers at the time: Some argued CIs could indeed be used in this way within computer science education, but others argued that because CIs are not comprehensive assessments, they are not meant for this purpose [120].

Very few papers in our analysis dealt with this challenge directly, but we mention an important note that may be useful to future researchers. We found evidence that CI researchers may occasionally be tempted to make a CI more comprehensive than it needs to be. In one paper which focused on giving practical advice to those building a computer science CI [119], the authors reported that they benefited immensely from consulting with an external evaluator who specialized in learning sciences. At one point, this external evaluator explained to the authors that many researchers who set out to design CIs possess the misconception (pun intended!) that they need to cover all course content, when this is not the case. We mention this point because for fully validated CIs, this is often a given that those new to the field may not immediately notice.

Our findings thus indicate that CI researchers over the last decade have worked in accordance with the standard structure of CIs and focused on key concepts needed for a course, but that little work has been done to determine if a CI or a comprehensive assessment is better suited for the task of evaluating teaching effectiveness. This is a potential area of future work.

4.3 Deviations from the norm

In this section, we outline some important deviations from the generally accepted methodology for constructing CIs. Many of the techniques discussed below warrant potential future work.

4.3.1 Alternative Methods for Choosing Concepts and Identifying Misconceptions. As mentioned above, some papers pointed out that carrying out the Delphi process can be cumbersome because the expert consensus on concepts still needs to be empirically validated after the fact. We identified 4 studies in our review which discussed alternative methods for choosing concepts and identifying misconceptions. 3 of these proposed a novel technological approach, and the 4th discussed a variation of the exam analysis technique for identifying misconceptions.

Table 10: Processes Implicitly Counted As Concepts

Concept Inventory	Processes Implicitly Counted as Concepts
Basic Data Structures Inventory (Webb et al. [130])	Algorithm & code tracing, identifying and remedying flaws in DS implementation
K-12 Computer Science (Rachmatullah et al. [101])	Debugging
Object-Oriented Programming, Data Structures, Concurrency, Algorithms (Nelson et al. [83])	Code tracing, Design and analysis of algorithms
Discrete Math, Programming, Logic Design (Goldman et al. [46])	Algorithm Design, Debugging, Test Design
Algorithm Analysis CI (Farghally et al. [36])	Asymptotic running time analysis and Analysis of algorithms

2 studies (one discussed in 2 papers by the same authors, for a total of 3 papers) utilized a data-driven approach for concept identification based in coevolution and co-optimization theory [10, 131, 132]. At a high level, the associated algorithms identify structural relationships among students and problems to determine different performance levels. The exact algorithms used by the 2 studies were different (with one building on the other), but they have the same theoretical foundations.

We draw attention to these 2 studies because their results suggest coevolutionary algorithms can be useful at *both* the choosing concepts and identifying misconceptions stages of the CI development process. In particular, Wiegand et al. produce a set of representative concepts using a novel Dimension Extraction Coevolutionary Algorithm (DECA) [132], and Bari et al. produce a set of evolved problems with distractors directly corresponding to common student misconceptions with their algorithm [10]. Regarding the former study, it is worth noting that although this algorithm refined the list of concepts by identifying difficult ones, it still required the authors to start with an initial list of potential concepts.

The remaining 2 studies used variations of crowdsourcing in their methods. Saarinen, et al. [103] developed a new technique called “Adaptive Tool-Driven Concept Generation.” This method combines classsourcing with machine learning to generate questions. The authors reported that the generated questions were similar to those designed by experts and helped identify potential misconceptions, providing a potentially useful method for aspects of developing a CI.

The final study [108] attempted to generate questions by crowdsourcing open-ended responses to the *existing* cybersecurity and digital logic CIs on Amazon Mechanical Turk and using them to generate potential question distractors. Although not a novel algorithmic approach, the authors argue that their results indicate this method is faster and cheaper than the traditional method of identifying misconceptions with think-aloud interviews.

The 4 methods above appear to be uncommon but show promising preliminary results, and may be a fruitful area of future work.

4.3.2 Novel classifications of concepts. The CI studies we analyzed generally did not place concepts into specific hierarchies of difficulty, and instead focused on pinpointing core concepts that students in a given course should know. However, there was one notable exception to this we mention here.

Wiegand, et al., in addition to presenting the coevolution approach to concept identification mentioned above, also propose the notion of “informatively easy” and “informatively hard” concepts as foundational work for an eventual introductory programming CI [131]. In their Dimension Extraction Coevolutionary Algorithm, dimensions classify various student-problem structures. They split students into two categories—the most “dominant” students being the ones who answered the greatest number of questions correctly within a dimension and the most “dominated” students being the ones who answered the fewest. They then defined a question as informatively hard if the most dominant students solved it incorrectly the most, and as informatively easy if the most dominated students solved it correctly the most.

The authors argue these classifications can give deeper insight into question quality than traditional definitions of hard and easy, which are based in raw counts of correctness. In particular, if a question is informatively hard but traditionally easy, that may imply the general student population is correctly guessing the answer, and if a problem is informatively easy but traditionally hard, that may mean the general student population can make mistakes easily on a problem [131].

As this was the only study which presented novel ways of classifying problem difficulty, such approaches do not appear to be well studied. We highlight this as a potential area of future work, in particular because it has the potential to enhance our ability to construct effective CIs. In the authors’ own words, when student performance follows a pattern that does not fit into traditional definitions of easy and hard, “it is likely that student behavior is confounding a linear expectation of performance, and is therefore interesting from an assessment point of view. Informative difficulty preserves the idea that different students understand different concepts in different ways” [131, p. 11].

5 DISCUSSION

We now holistically consider the outcome of our systematic review for our two research questions, explore the potential usefulness of periodic reviews of computer science CIs, and acknowledge the limitations of our study.

5.1 What do we know about CIs for computer science today?

Perhaps the most high-level and encouraging outcome of this review is that CI research is rich and growing within computing education. The 12 validated CIs that exist today, along with an additional 21 that are in progress, represent considerable growth from the 2 validated and 4 in-progress CIs that existed nine years ago [120]. These CIs stretch across a wide range of topic areas either directly or peripherally relevant to computing education, including CS1, CS2, computing theory, operating systems, and cybersecurity [61, 98, 126, 129, 130]. The full list of topic areas is in Table 5.

However, despite the wide range of topics covered by CIs, much work still needs to be done on expanding target demographics. Very limited work has been done beyond undergraduate courses; there are only 4 total CIs across middle-grades [101], high school [107, 131], and post-graduate student settings [52].

We suspect this almost exclusive focus on undergraduate students is partially due to the fact that many computer science CI researchers are involved in college computer science education themselves, and so it makes for a natural research setting. That said, the limited work in this area points to an ongoing need for expansion of equitable computer science education within primary and secondary schooling in general [128].

Regarding the development of CIs, the majority of studies followed variations of the established process for STEM CIs [1], using the Delphi process to choose concepts, exam analysis and/or think-aloud interviews to extract misconceptions, and CTT, IRT, and Cronbach's Alpha to measure validity and reliability. There were only a few studies which primarily focused on the development of novel methods for building CIs. It appears that the majority of current CI research in computing education aims to develop CIs through well-studied techniques, and there is little work being done on CI-building methodology itself.

5.2 Have researchers been able to effectively tackle the 4 challenges specific to computer science CIs outlined in Taylor et al.'s review?

3 of these 4 challenges have been thoroughly tackled in computer science CI research. Additionally, although the final challenge hasn't been a focus of recent work, there is nothing to suggest it has actively inhibited or damaged the field's progress.

Particularly interesting is that researchers rarely, if ever, stated that they meant to address these challenges, but they still ended up doing so extensively. This indicates excellent foresight by the first literature review's authors, who set a stable foundation for our research. We discuss the 3 challenges that were thoroughly present in the review below.

Computer science CI research currently adopts a unique position with respect to pre-tests. Although CIs in STEM are generally used as both pre-tests and post-tests, this usage is not as straightforward in computing education. Given that CS topics are growing increasingly popular in K-12 curricula, using a CI to assess misconceptions at a pre-test stage is growing in relevance, but only to an extent. Unlike the natural sciences, which govern the world we live in, computer science is human-made. As a result, while we develop intuition about natural scientific laws such as gravity simply by virtue

of growing up in the world, the same does not hold for computer science concepts. Until we are actually taught computer science in some form, we are unlikely to develop misconceptions about it. As such, administering a CI as a pre-test may not be as effective as in other STEM areas, and should be done taking into consideration the previous education of the students.

Ensuring the robustness of a CI takes some additional effort in computer science, primarily because computer science is still a relatively young field with potentially fluid content and multiple possible representations of the same concepts across programming languages. As such, researchers have the added task of ensuring that their CIs are stable across different programming languages.

For more theoretical topics, such as discrete mathematics [3] or finite state machines [52], this simply is not necessary. However, for concepts which are generally implemented in a programming language, there are two common solutions: 1) developing a pseudocode language for the CI or 2) replicating the CI in multiple languages. Both approaches appear to be effective. Furthermore, the latter one has opened a niche area of research within the field: developing methods for converting a CI to a new language without conducting the entire process from scratch [18].

In contrast to the second challenge above, the third is language-agnostic. Within computer science, it is important for students to understand certain procedural skills which we might only loosely consider to be concepts, such as designing algorithms and testing code. Within computer science CIs, the unwritten consensus seems to be that processes can simply be considered concepts. Not all CIs included such processes in their questions, and those that did simply labeled them as concepts in their CI. Among these, no studies mentioned any issues that arose as a result.

5.3 How often should we take stock of computer science CIs?

Here, we address the fact that our systematic review falls into a precarious position, as it is clearly not the first literature review of CIs in computing education, but it is also not quite a replication (per the definition of replication by Bollen et al. [14]), as our search methodology and research questions are different from, although related to, the previous one [120]. That said, we maintain it is a valuable contribution for two reasons.

First, CI research is a fast-growing subfield within the relatively niche area of computing education. However, the definition of "fast-growing" is highly dependent on the context of this field. From the perspective of standard systematic reviews in, for example, medical studies—where the number of papers included in a systematic review can easily stretch into the thousands—this growth may be viewed as extremely slow.

We make particular note of this because many systematic review guides may imply that this many studies is to be expected, but this is not the case in this context. We initially felt this way and worried that our 65 studies may not warrant a review. However, once we began, we quickly realized that the contributions being made were varied and important, and that documenting them could potentially be of great use to computer science CI researchers.

Second, as we noted earlier, most computer science CI research pertains to developing new CIs, but a select few studies have been

done aiming to contribute new methodologies to improve parts of the CI-building process itself [10, 103, 108, 132]. Because these papers are not the primary focus of most current CI research, they may not naturally rise to prominence. By documenting them in periodic reviews such as this one, we can 1) encourage further exploration of these avenues and 2) streamline the process of building better CIs within computing education.

In summary, based on the wealth of CI research we have identified in this review, as well as the considerable growth of existing computer science CIs since Taylor et al.'s 2014 literature review, we argue that periodically conducting systematic literature reviews of computing education CI literature is likely to be fruitful for the field as a whole. We predict that with the speed of growth we have seen, another decade may even be too long, and we recommend something of this nature be done every 5 years at minimum.

5.4 Limitations

We acknowledge several limitations of our study. One potentially significant limitation is the set of keywords used in our database queries. Although we attempted to balance comprehensiveness with precision, it is possible we may have missed one or more relevant keyword(s) and thus an associated set of literature. While this is a common limitation within literature reviews [e.g., 13, 41, 122], it is important to acknowledge. Notably, we observed that the set of papers captured within our systematic review did not include the original papers for two of the foundational and primary CIs in the field—the Foundational CS1 Assessment (FCS1) [124], and its replicated version, the Second CS1 Assessment (SCS1) [89]. Manual examination of these papers revealed that our search keywords do not appear within any of the fields searched for both these papers.

There are a couple of ways to interpret this omission. Firstly, while the FCS1 drew on CI work in other fields and is similar in nature to the computer science CIs that followed, the authors do not actually refer to it as a concept inventory, but rather as a concept assessment [124]. This is a point that Taylor et al. mention in their original review as well [120]. This, combined with the wide coverage of our search methods, gives us confidence that even with the absence of these two papers, our search was exhaustive. That said, in our research, we felt it was important to acknowledge the strong influence of the FCS1 (and the subsequent SCS1) upon this space, but also maintain the rigor of our systematic review. As such, following the example of Taylor et al., we did refer to both of these as CIs throughout our related work and discussion when relevant, but due primarily to the fact that they did not appear in our search, we did not include the original papers in our final paper counts. However, we were able to include the SCS1 (as a CI) in our final count of validated CIs, as our search included a paper that described the SCS1 in detail and performed an additional validation of it [136]. Similarly, our search queries did not return the original validated Basic Data Structures Inventory (BDSI) [97] paper because our search keywords do not appear within the fields searched, but we were still able to include this CI in our review through a later paper that discussed the practical details of building it and appeared in our search [119]. We also note that Taylor et al. did include the FCS1 in their count, and we did not, so readers should consider this when comparing tables and counts.

Our overall review could have been simplified slightly as well. In the screening stage, we did not explicitly screen out posters, panels, and extended abstracts. As such, we failed to screen out two panels [4, 29] and one extended abstract from the ICER 2018 Doctoral Consortium [71] that did not actively discuss work being done towards CI development in a manner that is relevant to this review. That said, these were the only such works we found, and their inclusion did not negatively impact our analyses and results in any way.

An additional limitation of this study is the inclusion of a paper on iteration-control constructs that was inaccessible to us [107]. We chose to include this paper in our review because the title and abstract contained promising information.

From our review, we can ascertain that the usage of CIs as a method of assessment for computer science is growing, and can be applied to a wide range of topics within the field. However, we acknowledge that our research takes a breadth-first approach in evaluating the field, and that depth-first work, such as Parker et al.'s [90] evaluation of the specific impact of the FCS1 [124] and SCS1 [89], is also incredibly important. We hope that researchers undertake such work as a complement to this review in the near future.

6 CONCLUSION AND FUTURE WORK

6.1 Overall Conclusions

This paper contributes a systematic literature review of current work in computer science CIs, explained in extensive detail to support replication. It includes research which made progress toward building a CI, provided recommendations for building CIs, or provided novel methods to improve the process of building CIs.

The 65 studies collated and analyzed in this work provide researchers with a thorough overview of the current state of CI research within computing education, and thus a foundation for making new contributions to the field.

With 33 total CIs in existence, 12 of which are validated, and 10 of which were validated *within the last nine years*, there is considerable evidence that this is a robust research area which will continue to grow. That said, research focuses almost exclusively on undergraduate students, with minimal excursions into other student demographics [52, 101, 107, 131].

In the last literature review of CIs for computer science in 2014, the authors identified 4 challenges of developing CIs unique to computer science: 1) pre-test limitations, 2) programming language dependence and designing robust CIs, 3) difficulty assessing skills, and 4) using CIs vs. using comprehensive assessments. Our review found that even though researchers did not necessarily set out to address these challenges as part of their work, the first three challenges repeatedly presented themselves and thus ended up being addressed. The fourth challenge is not a recurring theme in the literature, nor does it appear to be an inhibiting factor in CI research.

Several studies [19, 57, 65, 98, 136] noted the difficulty of attempting to use a CI as a pre-test due to the fact that it is difficult for students to form misconceptions about computing topics prior to taking a course, unlike with other STEM subjects. While other studies did indeed use pre-tests, our findings suggest that they tend

to be most relevant in more theoretical computing subjects, such as formal reasoning [e.g., 61], or in settings where the “pre”-test is still given at a point after some instruction has been given [e.g., 39, 44].

To ensure the robustness of CIs for courses dependent on a programming language, most studies took one of two approaches: 1) using a pseudocode language [e.g., 130], or 2) developing the CI in multiple programming languages [e.g., 18]. Both of these approaches were effective, with one of the former studies [130] explicitly stating students did not express confusion with the pseudocode regardless of which programming language they had experience in.

Our review did not identify any studies which explicitly differentiated between concepts and processes (such as code design and testing) in computing education, but we did find that such “processes” are often present in CIs and simply considered as concepts of their own. In 2014, Taylor et al. explicitly posed the question of whether processes should be thought of as concepts or distinguished from them; we conclude from our study that most computer science CI researchers faced no issues in treating both as the same.

6.2 Future Work Recommendations for Computer Science CI Researchers

Based on our findings, we make the following suggestions to future CI researchers:

- Very little work has been done to develop CIs for K-12 students. With the growing interest in computing education for these student profiles [42], it is an excellent time to develop CIs for them.
- While we identified the two techniques discussed above for ensuring robustness of CIs—using pseudocode or developing the CI in multiple languages—little work has been done in empirically determining the benefits and drawbacks. Rigorously evaluating these techniques, as well as developing new ones, may be apt.
- The fourth challenge posed by Taylor et al. involves determining if CIs or comprehensive assessments are better suited for the task of evaluating teaching effectiveness in computer science. This challenge appears to be mostly unaddressed as of yet.
- We identified 4 papers that presented novel methodologies for various aspects of the CI development process [10, 103, 108, 132]. Further evaluation of these techniques, as well as exploration of other techniques to improve the construction of CIs, is a potentially fruitful area of future work.
- One of the above papers [132] also proposed new classification categories for concepts: informatively easy and informatively hard. The authors argue such classifications provide more insight into student performance than traditional definitions of easy and hard and could be of great use in CI work.
- As we mentioned in the limitations section above, a few relevant papers did not appear in our database search due to certain keywords being missing. We recommend future CI researchers in computer science utilize, at minimum, “computer science” and “concept inventory” as keywords to ensure their papers are effectively indexed.

Overall, we find that CI research in computing education is a promising field that has seen considerable growth over the last decade and will continue to grow quickly. In 2014, Taylor et al. expressed the following wish [120, p. 20]:

Should the community embrace this challenge and develop CIs for a range of computer science courses, these CIs could usher in a new era of curricular and pedagogical innovation and evaluation.

Dare we say this era has begun.

ACKNOWLEDGMENTS

We acknowledge the support of Mel DeSart (Head, Engineering Library, University of Washington) for assisting us with identifying the databases that we used in this paper and supporting us through the process of familiarizing ourselves with them.

REFERENCES

Note: If an article’s title is in bold font, it indicates that the article was in the final set of 65 papers analyzed in this study.

- [1] Wendy Adams and C. Wieman. 2011. Development and Validation of Instruments to Measure Learning of Expert-Like Thinking. *International Journal of Science Education* 33 (06 2011), 1289–1312. <https://doi.org/10.1080/09500693.2010.512369>
- [2] Nicole J Albrecht, Patricia M Albrecht, and Marc Cohen. 2012. Mindfully teaching in the classroom: a literature review. *Australian Journal of Teacher Education* 37, 12 (2012), 1–14.
- [3] 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 (ITiCSE-WGR '06)*. Association for Computing Machinery, New York, NY, USA, 132–145. <https://doi.org/10.1145/1189215.1189182> Number of pages: 14 Place: Bologna, Italy.
- [4] Vicki L. Almstrum, David Klappholz, Steven Condly, and John Clement. 2006. **Are they learning what (we think) we’re teaching?**. In *Proceedings of the 37th SIGCSE technical symposium on computer science education (SIGCSE '06)*. Association for Computing Machinery, New York, NY, USA, 195. <https://doi.org/10.1145/1121341.1121402> Number of pages: 1 Place: Houston, Texas, USA.
- [5] Dianne L Anderson, Kathleen M Fisher, and Gregory J Norman. 2002. Development and evaluation of the conceptual inventory of natural selection. *Journal of research in science teaching* 39, 10 (2002), 952–978.
- [6] American Educational Research Association, American Psychological Association, National Council on Measurement in Education, et al. 1999. *Standards for educational and psychological testing*. American Educational Research Association.
- [7] Richard H Austing, Bruce H Barnes, Della T Bonnette, Gerald L Engel, and Gordon Stokes. 1979. Curriculum’78: recommendations for the undergraduate program in computer science—a report of the ACM curriculum committee on computer science. *Commun. ACM* 22, 3 (1979), 147–166.
- [8] Ada E Barach, Connor Jenkins, Serendipity S Gunawardena, and Krista M Kecskemeti. 2020. **MCS1: A matlab programming concept inventory for assessing first-year engineering courses**. In *2020 ASEE Virtual Annual Conference Content Access*.
- [9] Erin M Bardar, Edward E Prather, Kenneth Brecher, and Timothy F Slater. 2007. Development and validation of the light and spectroscopy concept inventory. *Astronomy Education Review* 5, 2 (2007), 103–113.
- [10] A.T.M. Bari, Alessio Gaspar, R. Wiegand, Jennifer Albert, Anthony Bucci, and Amruth Kumar. 2019. EvoParsons: design, implementation and preliminary evaluation of evolutionary Parsons puzzle. *Genetic Programming and Evolvable Machines* 20 (06 2019). <https://doi.org/10.1007/s10710-019-09343-7>
- [11] ATM Golam Bari, Alessio Gaspar, R Paul Wiegand, Dmytro Vitel, Kok Cheng Tan, and Stephen John Kozakoff. 2019. **On the potential of evolved parsons puzzles to contribute to concept inventories in computer programming**. In *2019 ASEE annual conference & exposition*.
- [12] Brett A. Becker, Catherine Mooney, Amruth N. Kumar, and Sean Russell. 2021. **A simple, language-independent approach to identifying potentially at-risk introductory programming students**. In *Proceedings of the 23rd australasian computing education conference (ACE '21)*. Association for Computing

- Machinery, New York, NY, USA, 168–175. <https://doi.org/10.1145/3441636.3442318> Number of pages: 8 Place: Virtual, SA, Australia.
- [13] Abeba Birhane, Pratyusha Kalluri, Dallas Card, William Agnew, Ravit Dotan, and Michelle Bao. 2022. The values encoded in machine learning research. In *2022 ACM Conference on Fairness, Accountability, and Transparency*. 173–184.
 - [14] Kenneth Bollen, John T. Cacioppo, Robert M. Kaplan, Jon A. Krosnick, James L. Olds, and Heather Dean. 2015. *Social, Behavioral, and Economic Sciences Perspectives on Robust and Reliable Science*. Technical Report. National Science Foundation. http://www.nsf.gov/sbe/AC_Materials/SBE_Robust_and_Reliable_Research_Report.pdf
 - [15] Craig W. Bowen. 1994. Think-Aloud Methods in Chemistry Education: Understanding Student Thinking. *Journal of Chemical Education* 71 (1994), 184–190.
 - [16] Daren C Brabham. 2008. Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence* 14, 1 (2008), 75–90.
 - [17] Emeline Brulé, Brianna J Tomlinson, Oussama Metatla, Christophe Jouffrais, and Marcos Serrano. 2020. Review of quantitative empirical evaluations of technology for people with visual impairments. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.
 - [18] Ricardo Caceffo, Pablo Frank-Bolton, Renan Souza, and Rodolfo Azevedo. 2019. **Identifying and Validating Java Misconceptions Toward a CS1 Concept Inventory**. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland UK) (ITICSE '19). Association for Computing Machinery, New York, NY, USA, 23–29. <https://doi.org/10.1145/3304221.3319771>
 - [19] Ricardo Caceffo, Steve Wolfman, Kellogg S. Booth, and Rodolfo Azevedo. 2016. **Developing a Computer Science Concept Inventory for Introductory Programming**. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) (SIGCSE '16). Association for Computing Machinery, New York, NY, USA, 364–369. <https://doi.org/10.1145/2839509.2844559>
 - [20] Hsiao-yun Chang, Marianne Wallis, and Evelin Tiralongo. 2007. Use of complementary and alternative medicine among people living with diabetes: literature review. *Journal of advanced nursing* 58, 4 (2007), 307–319.
 - [21] Chin Soon Cheah. 2020. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology* 12, 2 (2020), ep272.
 - [22] Qingwan Cheng, Angela Tao, Huangliang Chen, and Maira Marques Samary. 2022. **Design an assessment for an introductory computer science course: A systematic literature review**. In *2022 IEEE frontiers in education conference (FIE)*. 1–8. <https://doi.org/10.1109/FIE56618.2022.9962584>
 - [23] Catherine H. Crouch and Eric Mazur. 2001. Peer Instruction: Ten years of experience and results. *American Journal of Physics* 69, 9 (2001), 970–977. <https://doi.org/10.1119/1.1374249> arXiv:<https://doi.org/10.1119/1.1374249>
 - [24] Tyne Crow, Andrew Luxton-Reilly, and Burkhard Wuensche. 2018. Intelligent tutoring systems for programming education: a systematic review. In *Proceedings of the 20th Australasian Computing Education Conference*. 53–62.
 - [25] Norman Dalkey and Olaf Helmer. 1963. An experimental application of the Delphi method to the use of experts. *Management science* 9, 3 (1963), 458–467.
 - [26] Holger Danielsiek, Wolfgang Paul, and Jan Vahrenhold. 2012. **Detecting and understanding students' misconceptions related to algorithms and data structures**. In *Proceedings of the 43rd ACM technical symposium on computer science education (SIGCSE '12)*. Association for Computing Machinery, New York, NY, USA, 21–26. <https://doi.org/10.1145/2157136.2157148> Number of pages: 6 Place: Raleigh, North Carolina, USA.
 - [27] Adrienne Decker. 2007. *How students measure up: An assessment instrument for introductory computer science*. State University of New York at Buffalo.
 - [28] David Denyer and David Tranfield. 2009. Producing a systematic review. (2009).
 - [29] Nafeesa Dewji, Steven A. Wolfman, Geoffrey L. Herman, Leo Porter, Cynthia Taylor, and Jan Vahrenhold. 2015. **The CS concept inventory quiz show**. In *Proceedings of the 46th ACM technical symposium on computer science education (SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 669–670. <https://doi.org/10.1145/2676723.2677315> Number of pages: 2 Place: Kansas City, Missouri, USA.
 - [30] S Drachova-Strang, M Sitaraman, and J Hollingsworth. 2011. **Experimentation with tutors for teaching mathematical reasoning and specification**. In *Proceedings of the international conference on frontiers in education: Computer science and computer engineering (FECS)*. 1. tex.organization: The Steering Committee of The World Congress in Computer Science, Computer ...
 - [31] Anna Eckerdal and Michael Thuné. 2005. Novice Java programmers' conceptions of "object" and "class", and variation theory. *ACM SIGCSE Bulletin* 37, 3 (2005), 89–93.
 - [32] Richard J Enbody, William F Punch, and Mark McCullen. 2009. Python CS1 as preparation for C++ CS2. In *Proceedings of the 40th ACM technical symposium on Computer Science Education*. 116–120.
 - [33] Jerome Epstein. 2007. Development and validation of the Calculus Concept Inventory. In *Proceedings of the ninth international conference on mathematics education in a global community*, Vol. 9. Citeseer, 165–170.
 - [34] K Anders Ericsson and Herbert A Simon. 1980. Verbal reports as data. *Psychological review* 87, 3 (1980), 215.
 - [35] D.L. Evans, G.L. Gray, S. Krause, J. Martin, C. Midkiff, B.M. Notaros, M. Pavelich, D. Rancour, T. Reed-Rhoads, P. Steif, R. Streveler, and K. Wage. 2003. **Progress on concept inventory assessment tools**. In *33rd annual frontiers in education, 2003. FIE 2003*, Vol. 1. T4G–1. <https://doi.org/10.1109/FIE.2003.1263392>
 - [36] Mohammed F Farghally, Kyu Han Koh, Jeremy V Ernst, and Clifford A Shaffer. 2017. **Towards a concept inventory for algorithm analysis topics**. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 207–212.
 - [37] Sandy Farquhar and Peter Fitzsimons. 2011. Lost in translation: The power of language. *Educational Philosophy and Theory* 43, 6 (2011), 652–662.
 - [38] Rafael Corsi Ferrao, Igor Dos Santos Montagner, Ricardo Caceffo, and Rodolfo Azevedo. 2022. **How much C can students learn in one week? Experiences teaching C in advanced CS courses**. In *2022 IEEE frontiers in education conference (FIE)*. 1–8. <https://doi.org/10.1109/FIE56618.2022.9962662> ISSN: 2377-634X.
 - [39] Kathi Fisler, Shriram Krishnamurthi, and Preston Tunnell Wilson. 2017. **Assessing and Teaching Scope, Mutation, and Aliasing in Upper-Level Undergraduates**. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) (SIGCSE '17). Association for Computing Machinery, New York, NY, USA, 213–218. <https://doi.org/10.1145/3017680.3017777>
 - [40] Rita Garcia, Katrina Falkner, and Rebecca Vivian. 2018. Systematic literature review: Self-Regulated Learning strategies using e-learning tools for Computer Science. *Computers & Education* 123 (2018), 150–163.
 - [41] Tracy Gardner, Hayley C Leonard, Jane Waite, and Sue Sentance. 2022. What do We Know about Computing Education for K-12 in Non-formal Settings? A Systematic Literature Review of Recent Research. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 264–281.
 - [42] Varvara Garneli, Michail N. Giannakos, and Konstantinos Chorianopoulos. 2015. Computing education in K-12 schools: A review of the literature. In *2015 IEEE Global Engineering Education Conference (EDUCON)*. 543–551. <https://doi.org/10.1109/EDUCON.2015.7096023>
 - [43] Kathy Garvin-Doxas and Michael W Klymkowsky. 2008. Understanding randomness and its impact on student learning: lessons learned from building the Biology Concept Inventory (BCI). *CBE—Life Sciences Education* 7, 2 (2008), 227–233.
 - [44] Reinhard Gerndt and Jens Lüsssem. 2018. Structuring Student Projects along Concept Inventories. In *2018 19th International Conference on Research and Education in Mechatronics (REM)*. 42–47. <https://doi.org/10.1109/REM.2018.8421783>
 - [45] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey Herman, Lisa 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 (SIGCSE '08)*. Association for Computing Machinery, New York, NY, USA, 256–260. <https://doi.org/10.1145/1352135.1352226> Number of pages: 5 Place: Portland, OR, USA.
 - [46] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey L. Herman, Lisa Kaczmarczyk, Michael C. Loui, and Craig Zilles. 2010. **Setting the Scope of Concept Inventories for Introductory Computing Subjects**. *ACM Trans. Comput. Educ.* 10, 2, Article 5 (jun 2010), 29 pages. <https://doi.org/10.1145/1789934.1789935>
 - [47] Michael Gusenbauer and Neal R Haddaway. 2020. Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources. *Research synthesis methods* 11, 2 (2020), 181–217.
 - [48] Richard Hake. 1998. Interactive-Engagement Versus Traditional Methods: A Six-Thousand-Student Survey of Mechanics Test Data for Introductory Physics Courses. *American Journal of Physics - AMER J PHYS* 66 (01 1998). <https://doi.org/10.1119/1.18809>
 - [49] Ibrahim Abou Halloun and David Hestenes. 1985. The initial knowledge state of college physics students. *American Journal of Physics* 53, 11 (1985), 1043–1055. <https://doi.org/10.1119/1.14030> arXiv:<https://doi.org/10.1119/1.14030>
 - [50] Sally Hamouda, Stephen H Edwards, Hicham G Elmongui, Jeremy V Ernst, and Clifford A Shaffer. 2017. **A basic recursion concept inventory**. *Computer Science Education* 27, 2 (2017), 121–148.
 - [51] Julie Henry and Bruno Dumas. 2020. **Developing an Assessment to Profile Students Based on Their Understanding of the Variable Programming Concept**. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) (ITICSE '20). Association for Computing Machinery, New York, NY, USA, 33–39. <https://doi.org/10.1145/3341525.3387400>
 - [52] Julie Henry, Bruno Dumas, Andreea Vescan, and Alexandra Maria Pasca. 2022. **Student misconceptions about finite state machines: Identify them in order to create a concept inventory**. In *Proceedings of the 4th international workshop on education through advanced software engineering and artificial intelligence (EASEAI 2022)*. Association for Computing Machinery, New York, NY, USA, 2–9. <https://doi.org/10.1145/3548660.3561330> Number of pages: 8

- Place: Singapore, Singapore.
- [53] Geoffrey L. Herman, Michael C. Loui, Lisa Kaczmarczyk, and Craig Zilles. 2012. **Describing the what and why of students' difficulties in boolean logic.** *ACM Transactions on Computing Education* 12, 1 (March 2012). <https://doi.org/10.1145/2133797.2133800> Number of pages: 28 Place: New York, NY, USA Publisher: Association for Computing Machinery tex.articleno: 3 tex.issue_date: March 2012.
 - [54] Geoffrey L. Herman, Michael C. Loui, and Craig Zilles. 2010. **Creating the digital logic concept inventory.** In *Proceedings of the 41st ACM technical symposium on computer science education (SIGCSE '10)*. Association for Computing Machinery, New York, NY, USA, 102–106. <https://doi.org/10.1145/1734263.1734298> Number of pages: 5 Place: Milwaukee, Wisconsin, USA.
 - [55] Geoffrey L. Herman, Michael C. Loui, and Craig Zilles. 2011. **Students' misconceptions about medium-scale integrated circuits.** *IEEE Transactions on Education* 54, 4 (Nov. 2011), 637–645. <https://doi.org/10.1109/TE.2011.2104361>
 - [56] Geoffrey L. Herman, Craig Zilles, and Michael C. Loui. 2012. **Flip-flops in students' conceptions of state.** *IEEE Transactions on Education* 55, 1 (Feb. 2012), 88–98. <https://doi.org/10.1109/TE.2011.2140372>
 - [57] Geoffrey L. Herman, Craig Zilles, and Michael C. Loui. 2014. **A psychometric evaluation of the digital logic concept inventory.** *Computer Science Education* 24, 4 (2014), 277–303. <https://doi.org/10.1080/08993408.2014.970781> arXiv:<https://doi.org/10.1080/08993408.2014.970781>
 - [58] Geoffrey L. Herman, Craig Zilles, and Michael C. Loui. 2011. **How do students misunderstand number representations?** *Computer Science Education* 21, 3 (2011), 289–312. <https://doi.org/10.1080/08993408.2011.611712> Publisher: Routledge tex.eprint: <https://doi.org/10.1080/08993408.2011.611712>
 - [59] David Hestenes, Malcolm Wells, and Gregg Swackhamer. 1992. Force concept inventory. *The Physics Teacher* 30, 3 (1992), 141–158. <https://doi.org/10.1119/1.2343497> arXiv:<https://doi.org/10.1119/1.2343497>
 - [60] Julian PT Higgins, James Thomas, Jacqueline Chandler, Miranda Cumpston, Tianjing Li, Matthew J Page, and Vivian A Welch. 2019. *Cochrane handbook for systematic reviews of interventions*. John Wiley & Sons.
 - [61] Joseph E Hollingsworth and Murali Sitaraman. 2015. **Evaluation of formal reasoning abilities using a concept inventory.** In *FMSEE&T@FM*. 59–66.
 - [62] Khim Horton, Verena Tschudin, and Armorer Forget. 2007. The value of nursing: a literature review. *Nursing ethics* 14, 6 (2007), 716–740.
 - [63] Beth Hufnagel. 2002. Development of the astronomy diagnostic test. *Astronomy Education Review* 1, 1 (2002), 47–51.
 - [64] Henry Julie and Dumas Bruno. 2019. **Towards the Identification of Profiles based on the Understanding of Programming Concepts: the Case of the Variable.** In *2019 IEEE Frontiers in Education Conference (FIE)*. 1–8. <https://doi.org/10.1109/FIE43999.2019.9028697>
 - [65] Henry Julie and Dumas Bruno. 2020. **Approach to develop a concept inventory informing teachers of novice programmers' mental models.** In *2020 IEEE frontiers in education conference (FIE)*. 1–9. <https://doi.org/10.1109/FIE44824.2020.9274045> ISSN: 2377-634X.
 - [66] Henry Julie, Dumas Bruno, Heymans Patrick, and Leclercq Tony. 2020. Object-Oriented Programming: Diagnosis Understanding by Identifying and Describing Novice Perceptions. In *2020 IEEE Frontiers in Education Conference (FIE)*. 1–5. <https://doi.org/10.1109/FIE44824.2020.9273990>
 - [67] Henry Julie, Magis Tom, Clarinval Antoine, Vanderoose Benoit, and Dumas Bruno. 2020. **A tangible-augmented concept inventory to identify novices' misconceptions in programming.** In *2020 15th international conference on computer science & education (ICCSE)*. 370–374. <https://doi.org/10.1109/ICCSE49874.2020.9201806> ISSN: 2473-9464.
 - [68] Lisa C. Kaczmarczyk, Elizabeth R. Petrick, J. Philip East, and Geoffrey L. Herman. 2010. **Identifying student misconceptions of programming.** In *Proceedings of the 41st ACM technical symposium on computer science education (SIGCSE '10)*. Association for Computing Machinery, New York, NY, USA, 107–111. <https://doi.org/10.1145/1734263.1734299> Number of pages: 5 Place: Milwaukee, Wisconsin, USA.
 - [69] Kuba Karpierz and Steven A. Wolfman. 2014. **Misconceptions and Concept Inventory Questions for Binary Search Trees and Hash Tables.** In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (Atlanta, Georgia, USA) (SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 109–114. <https://doi.org/10.1145/2538862.2538902>
 - [70] Krista M Kecskeny, Ada Barach, Connor Jenkins, and Serendipity S Gunawardena. 2021. **Using programming concept inventory assessments: Findings in a first-year engineering course.** In *2021 ASEE virtual annual conference content access*.
 - [71] Cazembe Kennedy. 2018. **What are they thinking?.** In *Proceedings of the 2018 ACM conference on international computing education research (ICER '18)*. Association for Computing Machinery, New York, NY, USA, 272–273. <https://doi.org/10.1145/3230977.3231023> Number of pages: 2 Place: Espoo, Finland.
 - [72] Jeffrey W Knopf. 2006. Doing a literature review. *PS: Political Science & Politics* 39, 1 (2006), 127–132.
 - [73] 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 (ACM SE '20)*. Association for Computing Machinery, New York, NY, USA, 218–225. <https://doi.org/10.1145/3374135.3385277> Number of pages: 8 Place: Tampa, FL, USA.
 - [74] YIN Li and Cheryl B Jones. 2013. A literature review of nursing turnover costs. *Journal of nursing management* 21, 3 (2013), 405–418.
 - [75] Julie Libarkin. 2008. Concept inventories in higher education science. In *BOSE Conf.* 1–10.
 - [76] Julie C Libarkin and Steven W Anderson. 2005. Assessment of learning in entry-level geoscience courses: Results from the Geoscience Concept Inventory. *Journal of Geoscience Education* 53, 4 (2005), 394–401.
 - [77] Rebecca S Lindell and James P Olsen. 2002. Developing the lunar phases concept inventory. In *Proceedings of the 2002 Physics Education Research Conference*. New York: PERC Publishing.
 - [78] Rebecca S Lindell, Elizabeth Peak, and Thomas M Foster. 2007. Are they all created equal? A comparison of different concept inventory development methodologies. In *AIP conference proceedings*, Vol. 883. American Institute of Physics, 14–17.
 - [79] Andrew Luxton-Reilly, Brett A. Becker, Yingjun Cao, Roger McDermott, Claudio Mirolo, Andreas Mühling, Andrew Petersen, Kate Sanders, Simon, and Jacqueline Whalley. 2018. **Developing Assessments to Determine Mastery of Programming Fundamentals.** In *Proceedings of the 2017 ITiCSE Conference on Working Group Reports (Bologna, Italy) (ITiCSE-WGR '17)*. Association for Computing Machinery, New York, NY, USA, 47–69. <https://doi.org/10.1145/3174781.3174784>
 - [80] Kelly Mack, Emma McDonnell, Dhruv Jain, Lucy Lu Wang, Jon E. Froehlich, and Leah Findlater. 2021. What do we mean by “accessibility research”? A literature survey of accessibility papers in CHI and ASSETS from 1994 to 2019. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–18.
 - [81] 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. <https://doi.org/10.1080/08993408.2018.1562145> arXiv:<https://doi.org/10.1080/08993408.2018.1562145>
 - [82] Douglas R Mulford and William R Robinson. 2002. An inventory for alternate conceptions among first-semester general chemistry students. *Journal of chemical education* 79, 6 (2002), 739.
 - [83] Greg L. Nelson, Filip Strömbäck, Ari Korhonen, Marjahan Begum, Ben Blamey, Karen H. Jin, Violetta Lonati, Bonnie MacKellar, and Mattia Monga. 2020. **Differentiated Assessments for Advanced Courses That Reveal Issues with Prerequisite Skills: A Design Investigation.** In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (Trondheim, Norway) (ITiCSE-WGR '20)*. Association for Computing Machinery, New York, NY, USA, 75–129. <https://doi.org/10.1145/3437800.3439204>
 - [84] Mary A Nelson, Monica R Geist, Ronald L Miller, Ruth A Streveler, and Barbara M Olds. 2007. How to create a concept inventory: The thermal and transport concept inventory. In *Annual Conference of the American Educational Research Association*. 9–13.
 - [85] Ida Ngambeki, Phillip Nico, Jun Dai, and Matthew Bishop. 2018. **Concept inventories in cybersecurity education: An example from secure programming.** In *2018 IEEE frontiers in education conference (FIE)*. 1–5. <https://doi.org/10.1109/FIE.2018.8658474> ISSN: 2377-634X.
 - [86] Matthew J Page, Joanne E McKenzie, Patrick M Bossuyt, Isabelle Boutron, Tammy C Hoffmann, Cynthia D Mulrow, Larissa Shamseer, Jennifer M Tetzlaff, Elie A Akl, Sue E Brennan, et al. 2021. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *International journal of surgery* 88 (2021), 105906.
 - [87] Sonia Pamplona, Isaac Seoane, and Javier Bravo-Agapito. 2018. **Assessing Conceptual Knowledge in Three Online Engineering Courses: Theory of Computation and Compiler Construction, Operating Systems, and Signal and Systems.** In *Proceedings of the First International Conference on Data Science, E-Learning and Information Systems (Madrid, Spain) (DATA '18)*. Association for Computing Machinery, New York, NY, USA, Article 9, 6 pages. <https://doi.org/10.1145/3279996.3280005>
 - [88] Geet Parekh, David DeLatte, Geoffrey L. Herman, Linda Oliva, Dhananjay Phatak, Travis Scheponik, and Alan T. Sherman. 2018. **Identifying core concepts of cybersecurity: Results of two delphi processes.** *IEEE Transactions on Education* 61, 1 (Feb. 2018), 11–20. <https://doi.org/10.1109/TE.2017.2715174>
 - [89] 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 (Melbourne, VIC, Australia) (ICER '16)*. Association for Computing Machinery, New York, NY, USA, 93–101. <https://doi.org/10.1145/2960310.2960316>
 - [90] 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 *Proceedings of the 17th ACM Conference on International Computing Education Research (Virtual Event, USA) (ICER 2021)*. Association for Computing Machinery, New York, NY, USA, 60–68. <https://doi.org/10.1145/3446871.3469744>

- [91] Wolfgang Paul and Jan Vahrenhold. 2013. Hunting high and low: Instruments to detect misconceptions related to algorithms and data structures. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 29–34.
- [92] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Benndesen, Marie Devlin, and James Paterson. 2007. A survey of literature on the teaching of introductory programming. *Working group reports on ITiCSE on Innovation and technology in computer science education* (2007), 204–223.
- [93] Juri Pill. 1971. The Delphi method: substance, context, a critique and an annotated bibliography. *Socio-economic planning sciences* 5, 1 (1971), 57–71.
- [94] Leo Porter, Saturnino Garcia, Hung-Wei Tseng, and Daniel Zingaro. 2013. Evaluating student understanding of core concepts in computer architecture. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 279–284.
- [95] Leo Porter, Saturnino Garcia, Hung-Wei Tseng, and Daniel Zingaro. 2013. Evaluating Student Understanding of Core Concepts in Computer Architecture. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (Canterbury, England, UK) (ITiCSE '13). Association for Computing Machinery, New York, NY, USA, 279–284. <https://doi.org/10.1145/2462476.2462490>
- [96] Leo Porter, Cynthia Taylor, and Kevin C. Webb. 2014. **Leveraging open source principles for flexible concept inventory development**. In *Proceedings of the 2014 conference on innovation & technology in computer science education (ITiCSE '14)*. Association for Computing Machinery, New York, NY, USA, 243–248. <https://doi.org/10.1145/2591708.2591722> Number of pages: 6 Place: Uppsala, Sweden.
- [97] 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*. 111–119.
- [98] Seth Poulsen, Geoffrey L. Herman, Peter A. H. Peterson, Enis Golaszewski, Akshita Gorti, Linda Oliva, Travis Scheponik, and Alan T. Sherman. 2021. **Psychometric Evaluation of the Cybersecurity Concept Inventory**. *ACM Trans. Comput. Educ.* 22, 1, Article 6 (oct 2021), 18 pages. <https://doi.org/10.1145/3451346>
- [99] Siddhartha Prasad, Ben Greenman, Tim Nelson, John Wrenn, and Shriram Krishnamurthi. 2022. Making Hay from Wheats: A Classsourcing Method to Identify Misconceptions. In *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research (Koli, Finland) (Koli Calling '22)*. Association for Computing Machinery, New York, NY, USA, Article 2, 7 pages. <https://doi.org/10.1145/3564721.3564726>
- [100] M Praveena and V Jaiganesh. 2017. A literature review on supervised machine learning algorithms and boosting process. *International Journal of Computer Applications* 169, 8 (2017), 32–35.
- [101] Arif Rachmatullah, Bitu Akram, Danielle Boulden, Bradford Mott, Kristy Boyer, James Lester, and Eric Wiebe. 2020. **Development and validation of the middle grades computer science concept inventory (MG-CSCI) assessment**. *EURASIA Journal of Mathematics, Science and Technology Education* 16, 5 (2020), em1841.
- [102] Brian Rague. 2011. **Measuring CS1 perceptions of parallelism**. In *2011 frontiers in education conference (FIE)*. S3E–1–S3E–6. <https://doi.org/10.1109/FIE.2011.6142705> ISSN: 2377–634X.
- [103] Sam Saarinen, Shriram Krishnamurthi, Kathi Fisler, and Preston Tunnell Wilson. 2019. **Harnessing the Wisdom of the Classes: Classsourcing and Machine Learning for Assessment Instrument Generation**. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) (SIGCSE '19). Association for Computing Machinery, New York, NY, USA, 606–612. <https://doi.org/10.1145/3287324.3287504>
- [104] Elien Sabbe and Antonia Aelterman. 2007. Gender in teaching: A literature review. *Teachers and Teaching: theory and practice* 13, 5 (2007), 521–538.
- [105] Akhil Sachan, GN Bhadri, and Javed Kittur. 2019. **Design and development of concept assessment tool (CAT): A concept inventory**. *Journal of Engineering Education Transformations* 33, 1 (2019), 16–21.
- [106] Norman L Sanders. 1999. Evidence-based care in orthodontics and periodontics: a review of the literature. *The Journal of the American Dental Association* 130, 4 (1999), 521–527.
- [107] Emanuele Scapin and Claudio Mirolo. 2019. **An exploration of teachers' perspective about the learning of iteration-control constructs**. In *Informatics in Schools. New Ideas in School Informatics: 12th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2019, Larnaca, Cyprus, November 18–20, 2019, Proceedings 12*. Springer, 15–27.
- [108] Travis Scheponik, Enis Golaszewski, Geoffrey Herman, Spencer Offenberger, Linda Oliva, Peter Peterson, and Alan Sherman. 2020. **Investigating Crowdsourcing to Generate Distractors for Multiple-Choice Assessments**. 185–201. https://doi.org/10.1007/978-3-030-31239-8_15
- [109] Travis Scheponik, Enis Golaszewski, Geoffrey Herman, Spencer Offenberger, Linda Oliva, Peter A. H. Peterson, and Alan T. Sherman. 2019. Investigating crowdsourcing to generate distractors for multiple-choice assessments. arXiv: 1909.04230 [cs.HC].
- [110] Janet H Senf, Doug Campos-Outcalt, and Randa Kutob. 2003. Factors related to the choice of family medicine: a reassessment and literature review. *The Journal of the American Board of Family Practice* 16, 6 (2003), 502–512.
- [111] Alan T Sherman, Geoffrey L Herman, Linda Oliva, Peter AH Peterson, Enis Golaszewski, Seth Poulsen, Travis Scheponik, and Akshita Gorti. 2020. **Experiences and lessons learned creating and validating concept inventories for cybersecurity**. In *National Cyber Summit (NCS) Research Track 2020*. Springer, 3–34.
- [112] Alan T. Sherman, Linda Oliva, Enis Golaszewski, Dhananjay Phatak, Travis Scheponik, Geoffrey L. Herman, Dong San Choi, Spencer E. Offenberger, Peter Peterson, Josiah Dykstra, Gregory V. Bard, Ankur Chattopadhyay, Filipino Sharevski, Rakesh Verma, and Ryan Vrecenar. 2019. **The CATS Hackathon: Creating and Refining Test Items for Cybersecurity Concept Inventories**. *IEEE Security & Privacy* 17, 6 (Nov 2019), 77–83. <https://doi.org/10.1109/MSEC.2019.2929812>
- [113] Hannah Snyder. 2019. Literature review as a research methodology_ An overview and guidelines. *Journal of Business Research* 104 (2019), 333–339. <https://doi.org/10.1016/j.jbusres.2019.07.039>
- [114] Subhranil Som, Rana Majumdar, Mainak Ghosh, and Chesta Malkani. 2017. Statistical analysis of student feedback system using Cronbach's alpha and Utility Measurement Process. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*. 391–395. <https://doi.org/10.1109/ICTUS.2017.8286038>
- [115] Suenmy Mascarenhas Souza and Roberto A. Bittencourt. 2021. **Sentiments and Performance in an Introductory Programming Course Based on PBL**. In *2021 IEEE Global Engineering Education Conference (EDUCON)*. 831–840. <https://doi.org/10.1109/EDUCON46332.2021.9453963>
- [116] Kristin Stephens-Martinez, An Ju, Krishna Parashar, Regina Ongowarsito, Nikunj Jain, Sreesha Venkat, and Armando Fox. 2017. **Taking Advantage of Scale by Analyzing Frequent Constructed-Response, Code Tracing Wrong Answers**. In *Proceedings of the 2017 ACM Conference on International Computing Education Research (Tacoma, Washington, USA) (ICER '17)*. Association for Computing Machinery, New York, NY, USA, 56–64. <https://doi.org/10.1145/3105726.3106188>
- [117] A. Stone, K. Allen, T.R. Rhoads, T.J. Murphy, R.L. Shehab, and C. Saha. 2003. The statistics concept inventory: a pilot study. In *33rd Annual Frontiers in Education, 2003. FIE 2003*, Vol. 1. T3D–1. <https://doi.org/10.1109/FIE.2003.1263336>
- [118] Keith S Taber. 2018. The use of Cronbach's alpha when developing and reporting research instruments in science education. *Research in science education* 48 (2018), 1273–1296.
- [119] Cynthia Taylor, Michael Clancy, Kevin C. Webb, Daniel Zingaro, Cynthia Lee, and Leo Porter. 2020. **The Practical Details of Building a CS Concept Inventory**. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) (SIGCSE '20). Association for Computing Machinery, New York, NY, USA, 372–378. <https://doi.org/10.1145/3328778.3366903>
- [120] C. Taylor, D. Zingaro, L. Porter, K.C. Webb, C.B. Lee, and M. Clancy. 2014. **Computer science concept inventories: past and future**. *Computer Science Education* 24, 4 (2014), 253–276. <https://doi.org/10.1080/08993408.2014.970779> arXiv:https://doi.org/10.1080/08993408.2014.970779
- [121] Matti Tedre and Lauri Malmi. 2018. Changing aims of computing education: a historical survey. *Computer Science Education* 28, 2 (2018), 158–186.
- [122] Jonathan P Tennant and Tony Ross-Hellauer. 2020. The limitations to our understanding of peer review. *Research integrity and peer review* 5, 1 (2020), 6.
- [123] Allison Elliott Tew. 2010. **Assessing Fundamental Introductory Computing Concept Knowledge in a Language Independent Manner**. Ph. D. Dissertation. USA. Advisor(s) Guzdial, Mark. AAI3451304.
- [124] Allison Elliott 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*. 111–116.
- [125] Richard J Torracco. 2005. Writing integrative literature reviews: Guidelines and examples. *Human resource development review* 4, 3 (2005), 356–367.
- [126] Jan Vahrenhold and Wolfgang Paul. 2014. **Developing and validating test items for first-year computer science courses**. *Computer Science Education* 24, 4 (2014), 304–333. <https://doi.org/10.1080/08993408.2014.970782> arXiv:https://doi.org/10.1080/08993408.2014.970782
- [127] Ashok Kumar Veerasamy, Daryl D'Souza, and Mikko-Jussi Laakso. 2016. **Identifying novice student programming misconceptions and errors from summative assessments**. *Journal of Educational Technology Systems* 45, 1 (2016), 50–73. <https://doi.org/10.1177/0047239515627263> tex.eprint: <https://doi.org/10.1177/0047239515627263>
- [128] Sara Vogel, Rafi Santo, and Dixie Ching. 2017. Visions of Computer Science Education: Unpacking Arguments for and Projected Impacts of CS4All Initiatives. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) (SIGCSE '17). Association for Computing Machinery, New York, NY, USA, 609–614. <https://doi.org/10.1145/3017680.3017755>

- [129] Kevin C. Webb and Cynthia Taylor. 2014. **Developing a pre- and post-course concept inventory to gauge operating systems learning**. In *Proceedings of the 45th ACM technical symposium on computer science education (SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 103–108. <https://doi.org/10.1145/2538862.2538886> Number of pages: 6 Place: Atlanta, Georgia, USA.
- [130] Kevin C. Webb, Daniel Zingaro, Soohyun Nam Liao, Cynthia Taylor, Cynthia Lee, Michael Clancy, and Leo Porter. 2021. **Student Performance on the BDSI for Basic Data Structures**. *ACM Trans. Comput. Educ.* 22, 1, Article 8 (oct 2021), 34 pages. <https://doi.org/10.1145/3470654>
- [131] R. Paul Wiegand, Anthony Bucci, Amruth N. Kumar, Jennifer Albert, and Alessio Gaspar. 2021. **Identifying Informatively Easy and Informatively Hard Concepts**. *ACM Trans. Comput. Educ.* 22, 1, Article 7 (oct 2021), 28 pages. <https://doi.org/10.1145/3477968>
- [132] R. Paul Wiegand, Anthony Bucci, Amruth N. Kumar, Jennifer L. Albert, and Alessio Gaspar. 2016. **A Data-Driven Analysis of Informatively Hard Concepts in Introductory Programming**. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) (*SIGCSE '16*). Association for Computing Machinery, New York, NY, USA, 370–375. <https://doi.org/10.1145/2839509.2844629>
- [133] Titus Winters and Tom Payne. 2005. What Do Students Know? An Outcomes-Based Assessment System. In *Proceedings of the First International Workshop on Computing Education Research* (Seattle, WA, USA) (*ICER '05*). Association for Computing Machinery, New York, NY, USA, 165–172. <https://doi.org/10.1145/1089786.1089802>
- [134] Lea Wittie, Anastasia Kurdia, and Meriel Huggard. 2017. **Developing a concept inventory for computer science 2**. In *2017 IEEE frontiers in education conference (FIE)*. 1–4. <https://doi.org/10.1109/FIE.2017.8190459>
- [135] Lea Wittie, Anastasia Kurdia, Judy Peng, James Kelly, and Meriel Huggard. 2020. **Developing a Concept Inventory for Computer Science 2: What should it focus on and what makes it challenging?**. In *2020 IEEE frontiers in education conference (FIE)*. 1–8. <https://doi.org/10.1109/FIE44824.2020.9273941> ISSN: 2377-634X.
- [136] 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* (Minneapolis, MN, USA) (*SIGCSE '19*). Association for Computing Machinery, New York, NY, USA, 699–705. <https://doi.org/10.1145/3287324.3287370>
- [137] Andreas Zendler. 2019. cpm. 4. CSE/IRT: Compact process model for measuring competences in computer science education based on IRT models. *Education and Information Technologies* 24 (2019), 843–884.
- [138] Donald W Zimmerman. 1972. Test reliability and the Kuder-Richardson formulas: Derivation from probability theory. *Educational and Psychological Measurement* 32, 4 (1972), 939–954.