# Leveraging Open Source Principles for Flexible Concept Inventory Development

Leo Porter
Mathematics and Computer Science Department
Skidmore College
Skidmore, NY
leo.porter@skidmore.edu

Cynthia Taylor
Computer Science Department
Oberlin College
Oberlin, OH
ctaylor@oberlin.edu

Kevin C. Webb
Computer Science Department
Swarthmore College
Swarthmore, PA
kwebb@cs.swarthmore.edu

## ABSTRACT

Concept Inventory (CI) assessments, which target high-level learning goals, have proven highly valuable for higher education research. These assessments have helped to evaluate pedagogical practices among individual instructors, both within and across institutions, and have hence elevated the level of discourse on education within the community. The success of CIs in physics has inspired similar developments in computer science, with a few CIs now developed for computer science courses. However, the development of a CI typically follows a burdensome process, requiring a significant investment to produce a single CI that may be difficult to deploy due to institutional curricular differences. Furthermore, as our field continues to be shaped by technological advances, a path to faster, more modular CI development is critical.

This paper proposes an alternative CI development model and continues the discussion within the community about the need for, and path to, concept inventories throughout the computer science curriculum. Specifically, we explore the implications of an open collaboration system for CI development that would mimic the principles common to open source software communities, which have regularly demonstrated their ability to produce high-quality results.

## Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science Education

## General Terms

Human Factors

## Keywords

Concept Inventory, Assessment, Community Discussion

## 1. INTRODUCTION

In this paper, we propose an alternative process for the development of *Concept Inventories* (CIs). A Concept Inventory is an assessment mechanism consisting of a series of forced-choice questions, given as both a pre- and post-test. Unlike traditional assessment mechanisms such as final exams, CIs are designed to measure the success of the course in raising the level of students' conceptual knowledge, rather than assessing individual students. By focusing on the common concepts, CIs are applicable to similar courses taught by different instructors or at different institutions.

CIs provide valuable insight into how students conceptualize course content by focusing on student understanding of core concepts, as opposed to relying on detailed calculations or rote memorization. In recent years, concept inventories have gained traction in the sciences as mechanisms for exposing student misconceptions before and after instruction, contrasting instructor and student views of classroom material, and measuring course effectiveness.

Moreover, as standard assessment tools, CIs enable comparison of pedagogical techniques between courses, previous semesters, or across multiple institutions. In physics, where CI adoption has been most prominent, work with the Force Concept Inventory [18] led to a dramatic revamping of the way introductory physics lectures are taught, resulting in significant gains in student learning [10, 15].

Previously, the Delphi Process has been proposed as a rigorous method for developing concept inventories [13] that aims for consensus among a panel of fifteen to twenty subject-area experts. The process requires panelists to identify key concepts, propose assessment questions, and iterate though multiple rounds of refinement in which questions are scored across several metrics. Experts involved in the Delphi process are often compensated for their time, which makes developing a concept inventory a difficult proposition without significant financial support. The end result of this method is a rigorously-evaluated concept inventory, and while we strongly believe in the process's efficacy, its overhead is relatively high. The time and effort required to develop concept inventories in this manner is daunting enough to effectively hamper the development of new concept inventories, computer science topics included.

Additionally, CI development practices for other disciplines may not properly address the needs of computer science. Unlike a physics course on Newtonian mechanics where key topics are well established, curricular differences between institutions commonly lead to topics appearing in different courses (e.g., recursion may appear early in a functional programming CS1 but might not appear until CS2 at institutions that teach CS1 using an iterative language). As such, constructing a concept inventory as a collection of independent, topic-based modules may be more appropriate than a whole-course CI. Moreover, the content of courses may change as our field advances and/or technology changes. For example, while single-chip multiprocessors are a key topic in computer architecture classes today, the topic would have scarcely been mentioned ten years ago. As such curricular shifts occur, the field needs a mechanism by which CIs can similarly adapt.

In this paper, we propose an alternative concept inventory devel-


*ITICSE'14,* June 21–26, 2014, Uppsala, Sweden.

http://dx.doi.org/10.1145/2591708.2591722.

opment process inspired by the open source software model. In recent years, the open source community has demonstrated the effectiveness of rapid project development by incorporating large groups of contributors with varying levels of expertise [24, 27]. We believe that by applying similar multi-author development models to the construction of computer science concept inventories, the educational community will benefit from high-quality, modular concept assessments with significantly increased course coverage.

With this paper, we aim to incite discussion surrounding concept inventories in the computer science community and solicit involvement from the community. We are not presenting results or arguing that this is a solved problem with a single correct approach. Rather, we examine the advantages of this approach, describe our goals and expectations for the resulting concept inventories, and explore the potential challenges (e.g., validation, and versioning) of the design space.

# 2. MOTIVATION

Concept inventories have proven to be essential in measuring student conceptual understanding in the physical sciences. In this section, we advocate CI adoption in computer science and characterize early efforts toward developing inventories for upper-division CS courses.

## 2.1 Value of CIs

Many STEM fields have adopted concept inventories with positive results [18, 20]. Unfortunately, the adoption of CIs in computer science, a field that has been defined by (and prides itself on) rapid improvement, remains slow. While the currently developed CIs for computer science are excellent [11, 28], they only cover a small number of courses and, in some cases, have limited availability.

The lack of concept inventories plainly reduces the potential for informed discussion in the computer science education community. A significant percentage of work in our field reports solely on student satisfaction. While satisfaction is a relevant and meaningful metric, it has limited value when reported in isolation. Ultimately, the most important metric for the success of a pedagogical practice or educational tool is *student learning*.

Recognizing the value of reporting on student learning, some work has succeeded in producing controlled studies across multiple courses using similar or identical final exams as the metric of student learning success. However, even this metric leaves room for criticism as the quality and content of the final exam is often unknown or focused on other aspects of the course (e.g., calculations rather than high-level concepts). As an established set of educational benchmarks, concept inventories would elevate the level of discourse considerably and enable stronger educational claims by CS education researchers.

## 2.2 Related Work

Concept inventories are now widely used in a variety of disciplines [12] to assess the impact of pedagogical methods. In the field of computer science, concept inventories have been created for digital logic [16, 17] using the Delphi process, algorithms and data structures [11], and introductory courses [13, 14, 19]. Although these CIs are a good start, the majority of computer science topics remain without concept inventories.

Other work in computer science education research has examined student learning using assessments not developed using a CI development process (such as the Delphi process) [6, 8, 21–23, 25, 28–31]. These works have successfully underscored the need for more established assessment mechanisms in computer science by providing important intuition as to student understanding.

In both formal CIs and other assessment mechanisms, findings often demonstrate that students understand far less than instructors expect [21, 23, 30, 31]. Table 1 provides results from the concept assessments. These results serve to underscore the importance of established assessment mechanisms, which facilitate meaningful comparisons between curricular and pedagogical computer science practices. Despite the Delphi process being a standard CI development method, we note that (to the best of our knowledge) many more of the non-Delphi-developed concept assessments were administered in courses with published results than Delphi-developed concept inventories. We argue that test availability should be a key goal of concept inventory development.

**Table 1: Post-test results from various concept assessments in computer science. The only available data from an assessment using the Delphi Process is [17].**

| Exam Content | Correctness (avg.) |
|---|---|
| BASIC Programming (Table I in [6]) | 31% |
| Number and Date Sorting [8] | 59% |
| Code Comprehension/Tracing [21] | 60% |
| Value and Reference Assignment [22] | 63% and 17% (resp.) |
| Write Calculator Program [23] | 21% |
| Fundamental Intro. Concepts [31] | 42% |
| Digital Logic CI [17] | 55% |
| CS1 Language Independent CI [30] | 34% |

## 2.3 Preliminary work

Absent vetted CIs in Operating Systems and Computer Architecture, we have recently developed basic concept tests for these topics [26, 32]. This work applies to the present discussion, as these concept assessments were developed by just a few experienced instructors, who did not follow the Delphi Process, but nevertheless offered insight into student understanding of course content.

The concept assessment in computer architecture was designed by two experienced instructors who are published researchers in computer architecture. We designed the questions with the intent that *every* student successfully completing an architecture class should be able to answer the question correctly. The CI was run in four different courses by four different faculty at two different institutions under different conditions (as an online pre-test and as an online post-test before the end of the term, in an in-class final review session, and on the final exam). Although results on different questions varied somewhat by course, the CI revealed two interesting findings. First, there were questions for which students performance did not improve significantly from pre-test to post-test. Additionally, across these different classes, institutions, and instructors, students answered only 56% of questions correctly.

To see if such results are common in upper-division computing courses, we have similarly developed a concept test for Operating Systems following a similar methodology. Across three different instructors and four different courses at two institutions, we have found similar results to the architecture study. Students do not demonstrate significant improvement between the pre- and post-test for some questions, and on average, students answer 55% of the questions correctly.

Despite these CIs not being developed using a standard CI development process, we believe that the results are nonetheless hugely informative for instructors. That students are misunderstanding nearly half of the tested core concepts in upper-division classes is worrisome. Under the status quo, their conceptual difficulty might have gone unnoticed. With the introduction of a concept inventory, instructors can repeatedly collect student performance data, enabling them to more easily guide modifications to their courses

and/or pedagogical practices. This is the true value of such evaluation mechanisms: *they provide instructors with a motivation to improve and a means to measure improvement.*

Undoubtedly, because these inventories were not rigorously vetted, their results come with some caveats. Our experience in developing them informs our belief that the availability of a respectable CI, even an imperfect one, strongly outweighs having no inventory at all. We believe this presents an opportunity to employ the collective insight of our community. With open access and contributions from community members, the quality of our inventory tools will only get better. This motivates our analogy to open source software, and in the remainder of this paper, we discuss the goals and design challenges of our proposal.

# 3. GOALS

To guide the design of collaborative concept inventories, we turn to the open source software community for inspiration. We aim to formulate a development process for concept inventories that leverages community involvement to quickly create and deploy a concept inventory. Ideally, any interested instructor should be able to use the test in their class and compare the results with results from other classes. Additionally, instructors should be able to contribute new questions and suggest modifications to existing ones based on their experience with the concept inventory and their classes. In this section, we describe our vision, goals, design requirements for cooperative CI development.

The following objectives are critical to the construction of a community-driven concept inventory:

**Customizable.** As topics appear in different courses based on differences in computer science program curricula, an assessment for a course should be customizable to the content of that course. This customizability may be at a module level (e.g., a module on sorting) or at an individual question level (e.g., a question on inheritance). At either level, assessment results will then need to be associated with the context/course in which the assessment was given.

**Collaborative.** Much of the value of concept inventories comes from their ability to enable comparisons across courses, instructors, and institutions. However, to fully realize this potential, a CI needs as many participants as possible. We believe that open instructor access to CI questions and response data, including instructors who do not identify as CS education researchers, is paramount to establishing the collaborative support environment around a CI.

Any interested instructor should be able to contribute to the development of a concept inventory, by contributing new questions or recommending changes to existing questions. Ultimately, as the number of CI users and contributors increases, so does the quality and availability of questions and response data, which enhances the overall impact and utility to the CS community. Thus, it should be straightforward for instructors to use a concept inventory in their class, make their results available to other instructors, compare their results with others, and contribute feedback, adjustments, or new questions for future inclusion.

**Adaptable.** Unlike introductory Newtonian physics, which has been relatively quiescent for decades, the technology-driven computer science discipline has evolved rapidly in recent years. Such an accelerated progression suggests that we need flexible assessment tools. Thus, inventories should be developed as dynamic exams, with a clearly-defined procedure for making modifications in response to deployment data or new technologies.

**High-quality.** A concept inventory requires high-quality questions if instructors and education researchers are to draw reliable conclusions from the results. "Linus's Law" [27], which states that a large enough developer base will efficiently detect and correct software bugs, is an essential underpinning of the open source software world. The open source community has repeatedly demonstrated the ability to produce high-quality software, and we believe the same principles will serve the CI development process.

We propose leveraging the adaptable nature of community-driven project development to iteratively reexamine and refine CI questions. Thus, inventory development should converge on a set of questions and presentation style that are clear, accessible to students, and carefully crafted to elucidate concept comprehension.

**Maintainable.** Maintaining the quality of collaborative concept inventories will require an organized management infrastructure. Like open source software, the availability of simple tools to document and support development will lower the barrier to entry for new contributors and help to cultivate a CI's community. Thus, an inventory's development process should clearly track question contributions, collaborator discussions, alteration history, and recorded response data with low administrative overhead.

# 4. DESIGN

Our proposal to accelerate the development process of computer science concept inventories is not without obstacles, and in this section, we examine the design space of our approach. Since this is a discussion paper describing a space of potential choices (rather than describing an existing solution), we opt to explore potential options, rather than prescribe solutions. Ultimately, the goal is to see a variety of CIs developed across many computer science subject areas, and we recognize that each may make different design decisions.

## 4.1 Composition and versioning

The need for CIs to be developed openly and iteratively is critical to our *collaborative* and *adaptable* design goals. To support these goals in a realistic fashion, the community will require CI tracking and revision history.

The first point of contention we envision with respect to inventory composition is version control granularity:

> **Should a concept inventory be a loosely related set of questions or a holistic exam?**

One design option is to represent a concept inventory not as a synoptic test, but as a collection of questions, some of which may be available in multiple versions. Instructors may choose which of the set of questions available for their class they will use in the concept inventory they give, and which version of each question.

This model enables instructors to mix-and-match self-contained questions that are independently versioned. By allowing instructors to choose the questions that best fit their courses, this model helps to address the diversity in course material coverage. The potential trade-off is that collected responses would only be comparable per-question, rather than representative of student performance across a holistic CI. This complicates comparing results, as instructors who wish to make direct comparisons would need to purposely select the same set of questions.

On the other hand, a holistic approach simplifies data comparison but requires a stronger consensus from the community regarding question inclusion and may reduce a CI's dynamism. Continuing our open source analogy, we believe that a holistic CI development method might resemble the branched structure [9] common to many software development projects. A CI developed in this fashion might consist of core set of "stable" questions augmented by "experimental" questions that help to inform future revisions. Opting for a hybrid model, the experimental questions could be optionally chosen by instructors to help tailor the CI to their course.

**How do changes get integrated into a release of a concept inventory?**

The field of computer science is by nature constantly evolving, and with it some of the core concepts in our courses may change (e.g., energy consumption and related concepts have become a critical to systems design in the past decade). This necessitates concept inventories that are capable of adapting to the current knowledge in the subject area. The CI development process must balance the trade-off between frequent updates, which complicates data comparisons, and remaining unchanged, which runs the risk of becoming outdated.

This issue is particularly important for holistic CI development, which requires a procedure for transitioning questions from "experimental" to "stable". We envision a core group of test developers who curate the stable version and decide when to announce a new release. As instructors use a CI and provide feedback on the questions, the developers may utilize a mechanism to automatically move questions into the stable group based on quantitative instructor evaluations. Alternatively, they might employ a scheduled release cycle for new stable versions, and at that point the top-rated questions become the new stable test.

**How are versions managed?**

With the ability to update questions comes the need to reference specific versions of a question. Version control is a common and well-understood problem in our field, and numerous solutions, such as CVS, Subversion and Git already exist [1, 3, 5]. However, all of these systems specialize in maintaining code rather than text documents. One way to deal with this might be to simply split up each question into a separate file for ease of editing.

There is also the need to keep results associated with a specific version of the test or question. This necessitates a strong set of version numbers that are easy to reference and associate with data. Likewise, instructor feedback and discussions will also need to be associated with specific versions.

## 4.2 Instructor Access

**Who should be given access to read and write concept inventory materials?**

Central to the idea of an open source concept inventory is the ability for any interested and experienced instructor to contribute to it. However, by nature a concept inventory must be more restricted than a typical open source project, as the questions cannot be publicly available. This necessitates an approval process for access, at the very least verifying that the requester is a course instructor or researcher, and not a student attempting to locate test questions.[1]

There must be some way to verify that someone attempting to gain access to a CI is actually an instructor. The standard practice in the physics community involves password protected documents, for which instructors acquire the password by sending an email to the CI maintainers with their name, institution, and website. However, this requires instructors to be separately approved for each individual concept inventory. An alternative system might make use of a cryptographic key system, in which instructors could be verified once, and then simply provide their key afterwards.

Once instructors are approved to access a CI, they might automatically be approved to contribute, or there may be some further application process. Depending on their frequency, contributions may need to be voted on democratically or submitted to a group of maintainers prior to inclusion in the public CI repository.

[1]Ideally, a CI should be used as an educational metric, and not for credit, disincentivizing students from attempting to gain access.

## 4.3 Leadership and Quality

**Who should have leadership roles in concept inventory development, and what should these roles be?**

We imagine most concept inventories will have some small group of instructors who do most of the management of the concept inventory, either because they are doing research involving the concept inventory, or because it is for a class they frequently teach. How much control this group has over the concept inventory will naturally vary. One compromise between fully open source CI development and the traditional Delphi process might be to carefully select a group, consisting of both pedagogy researchers and experts in a subject area, to maintain the concept inventory. This approach could help to bootstrap new inventories during their initial development.

**How will test questions be validated?**

Traditionally, CI questions have been validated by a number of methods, including having students perform think-alouds in order to check if questions are misleading or rely on concepts other than what they are designed to measure, and comparing student performance on CIs to performance on final exams. This validation is not incompatible with open source CI design. Once questions are submitted, they can be validated via think-alouds or other methods, and the results of that validation used to improve the questions. Contributors can indicate if questions have been validated, and how. Instructors can choose to use only validated questions, or to include newer questions that have not yet been validated as well.

**How can we assure a high-quality CI?**

The rigorous structure of the Delphi process was developed specifically to assure an accurate concept inventory that assesses key concepts agreed upon by experts in the field. Without this structure, we run the risk of developing concept inventories that lead to misleading results or focus on the wrong concepts. We believe that despite this risk, quality concept inventories can be developed via open source development methodologies. Open source software projects have shown their ability to produce quality products, and we believe that open source CI development will produce better end results with more community member involvement. It is reasonable to assume that those researchers and instructors who take a leadership role in the development of concept inventories will work to assure their quality. Ultimately, we believe that even imperfect concept inventories will still provide some standard way to assess student learning, something we currently lack in a large number of computer science courses.

## 4.4 Response data

In addition to versioning and access control, a successful CI repository will need to record metadata (e.g., usage frequency or question development discussions) and student response statistics. We believe that collected CI results should be made available, in an anonymized form, for use by the broader computer science community. For this design aspect, we hope to steer the discussion toward several questions, the first of which is:

**How much data should we collect, and what kinds?**

The granularity at which we collect results is important, especially if instructors are including different questions within the versions of the CI that they administer. As instructors, we would like to easily compare results across the test as a whole, individual questions, and individual versions of questions. One solution would be for raw scores for each question to be collected each time a test was

given, and those scores associated with that version of the question. Collecting results on the granularity of individual versions of questions will allow instructors to compile results for the questions as a whole, and different versions of the test.

Tagging questions with metadata such as subject area could also allow comparison across subjects. For example, an instructor may wish to measure the correlation between results from a specific question with others that assess a related concept. However, this will require instructors to be willing to contribute their data in this more detailed form. Additionally, more fine-grained data may be more difficult to anonymize, especially if it is possible to track how an individual did on each question, even if their name is removed. This becomes especially of concern when results from small classes are contributed.

Another tactic would be to only compare results from stable versions of the test, which would allow instructors to contribute their results as a whole. However, this means that an instructor can no longer compare across different test versions, and means that even when the same individual question is in multiple stable versions of the test with the same format and wording, we can no longer compare results for that question across versions.

There is also the issue of collecting meta-data about the questions. Since the concept inventory is being continually developed, feedback from instructors who have given the question to their classes would be extremely valuable. This could take the form of a simple "up vote/down vote," long form comments, and ideally the distribution of how their class performed on the question. Questions for which answers vary wildly may indicate a poorly worded or unclear question. This kind of empirical data can be collected and analyzed in order to validate test questions.

**How should response data be collected and anonymized?**

In the collection of response data, we need to anonymize data to protect student privacy, but also make sure we retain enough data to make the results as useful as possible. Information like the type of institution where the concept inventory was given, the experience level of the instructor, and what pedagogical techniques were used in the class will all make the data more useful for potential comparisons.

One solution to streamline both data collection and anonymization could be to administer the exam via the response data collection system. For example, an instructor could set up an account with the system (at which point items like institutional characteristics could be provided) and then request a web-link for their students to take the exam. Students could then take the exam on-line with results (potentially non-anonymized) sent to the instructor. That data could then be automatically anonymized by the system and incorporated into the collected data (with instructor permission).

**Should we incentivize participation?**

The more CI results aggregated for a subject area, the larger the impact it will have on the CS community. To aid in data collection, it may be beneficial to incentivize instructors to share their collected CI data. For example, a CI might only release questions to instructors who will pledge to contribute their results or aid in question development. Likewise, a CI may choose to only share previously-recorded results with instructors who commit to making their own results available. Whatever the model, it is important that the barrier to entry is not so high as to dissuade instructors from pursuing the concept inventory.

## 4.5 Pre- and Post-tests

**How should a CI be administered?**

Instructors traditionally administer concept inventories as a pre- and post-course assessment. Pre-tests help pinpoint misconceptions that may interfere with new student learning and identify existing knowledge that can be leveraged when explaining new subject material [7, 8, 19]. Comparisons between the pre- and post-test enable instructors to assess student learning gains.

Unfortunately, for many upper-level computer science classes, converting conceptual vocabulary into "plain English" that students can understandable at the start of the course is often a daunting task. While some concepts lend themselves easily to analogies (for example, tickets versus guest lists to represent capabilities and access control lists when discussing file protection), this can lead to students answering based on details of the chosen analogy, rather than the underlying concept. On the other hand, if a CI is written in technical prose, students may be intimidated by the prospect of an incomprehensible test on the first day of class. It may be desirable to administer only accessible questions as a pre-test, followed by a post-test including those and other more technical questions.

## 4.6 Intellectual property

**How should a concept inventory be copyrighted or licensed?**

Finally, CI developers must answer questions of copyright and ownership. Will individual instructors own questions they contribute? Will the test as a whole have a single copyright? If so, who will own the copyright?

Current licenses that allow for sharing and modification of work, such as the Creative Commons license [2], may be a practical solution. We believe that the chosen license should allow modification of the work, to prevent the concept inventory from becoming locked by one group of people (in the parlance of the open source community, it should be possible to fork a version). There are also questions of whether commercial use should be permitted and how to copyright development metadata. A license such as the Open Data Commons [4] may be more reasonable for sharing results.

# 5. CONCLUSION

The overarching objective of our concept inventory (CI) proposal is to enhance CS education, and the research surrounding it, by enabling data-driven comparisons of different pedagogical techniques. Preliminary concept inventories for CS have exposed the value of measuring student understanding of fundamental subject area concepts. The available results strongly suggest that students are *not* mastering the ideas that every student *ought* to know after completing the corresponding course. CIs represent reliable tools for measuring this high-level mastery, and their results provide desperately-needed factual data for informing pedagogical changes to improve student concept mastery.

We believe that a comprehensive collection of concept inventories would serve as a critical step towards achieving this lofty goal. CIs in other STEM disciplines, most notably physics, have already demonstrated a significant and lasting impact on educational methods and research in their fields. If we know it will be effective, why *shouldn't* we adopt a similar approach?

Some of those already working on CI development have followed the traditional Delphi Process development cycle which aims to ensure CI quality by requiring consensus from a panel of experts. Unfortunately, necessitating such formality is expensive and requires a significant time commitment. We laud the recent efforts by computer science education researchers in rigorously producing

CIs for computer science classes. However, most of the courses in our field still have no such exam available. Thus we advocate an alternative: reducing CI developmental rigidity in favor of an agile process that encourages broad collaboration and adapts to changes in technology and course content.

In this paper, we explored the application of open source software principles to computer science concept inventory development. The open source development model offers three promising advantages over current CI practices: (1) a collaborative development strategy capable of producing a quality product, (2) a clearly defined path for revision when content changes, and (3) a means to manage versioning of questions/modules/exams in order to compare results for research purposes.

This work identifies clear goals for developing high-quality, collaborative concept inventories whose open-access and availability will strongly support CS education research. We propose a basic framework for "open source" CI development, identify the obstacles faced by such a framework, and sketch the design space and trade-offs necessary to make our vision a reality. We believe that the design challenges of our proposal are surely tractable for a highly-motivated community like ours.

Ultimately, we hope that this paper will intensify the discussion of concept inventories in computer science, inspire new thinking about concept inventory development, and elicit increased participation in CI development from the CS education research community. The best way for an open development process to succeed is with your help!

## 6. ACKNOWLEDGMENTS

Thank you to the anonymous reviewers for their suggestions. This work was supported in part by NSF grant 1140731.

## 7. REFERENCES

[1] Concurrent versions system. `http://cvs.nongnu.org/`.

[2] Creative commons. `http://creativecommons.org/`.

[3] Git version control. `http://git-scm.com/`.

[4] Open data commons. `http://opendatacommons.org/`.

[5] Subversion. `http://subversion.tigris.org/`.

[6] P. Bayman and R. Mayer. A diagnosis of beginning programmers' misconceptions of basic programming statements. *Communications of the ACM*, 26(9), 1983.

[7] J. Bonar and E. Soloway. Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human-Computer Interaction*, 1(2):133–161, 1985.

[8] T.-Y. Chen, G. Lewandowski, R. McCartney, K. Sanders, and B. Simon. Commonsense Computing: Using student sorting abilities to improve instruction. In *SIGCSE*, 2007.

[9] J. Corbet. A guide to the kernel development process. https://www.kernel.org/doc/Documentation/development-process/.

[10] C. H. Crouch and E. Mazur. Peer Instruction: Ten years of experience and results. *American Journal of Physics*, 69(9), September 2001.

[11] H. Danielsiek, W. Paul, and J. Vahrenhold. Detecting and understanding students' misconceptions related to algorithms and data structures. In *SIGCSE*, 2012.

[12] D. Evans, G. Gray, S. Krause, J. Martin, C. Midkiff, B. Notaros, M. Pavelich, et al. Progress on concept inventory assessment tools. In *IEEE Frontiers in Education*, 2003.

[13] K. Goldman, P. Gross, C. Heeren, G. Herman, L. Kaczmarczyk, M. C. Loui, and C. Zilles. Identifying important and difficult concepts in introductory computing courses using a Delphi process. In *SIGCSE*, 2008.

[14] K. Goldman, P. Gross, C. Heeren, G. L. Herman, L. Kaczmarczyk, M. C. Loui, and C. Zilles. Setting the scope of concept inventories for introductory computing subjects. *ACM Transactions on Computing Education (TOCE)*, 10(2):5, 2010.

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

[16] G. Herman and J. Handzik. A preliminary pedagogical comparison study using the digital logic concept inventory. In *IEEE Frontiers in Education*, 2010.

[17] G. Herman, M. Loui, and C. Zilles. Creating the Digital Logic Concept Inventory. In *SIGCSE*, 2010.

[18] D. Hestenes, M. Wells, and G. Swackhamer. Force Concept Inventory. *The Physics Teacher*, 30, March 1992.

[19] L. C. Kaczmarczyk, E. R. Petrick, J. P. East, and G. L. Herman. Identifying student misconceptions of programming. In *SIGCSE*, 2010.

[20] J. C. Libarkin and S. W. Anderson. Assessment of learning in entry-level geoscience courses: Results from the geoscience concept inventory. *Journal of Geoscience Education*, 53(4), 2005.

[21] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Moström, K. Sanders, O. Seppälä, et al. A multi-national study of reading and tracing skills in novice programmers. *SIGCSE*, 2004.

[22] L. Ma, J. Ferguson, M. Roper, and M. Wood. Investigating the viability of mental models held by novice programmers. *ACM SIGCSE Bulletin*, 39(1), 2007.

[23] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *SIGCSE*, 2001.

[24] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(3):309–346, 2002.

[25] R. Pea. Language-independent conceptual "bugs" in novice programming. *Journal of Educational Computing Research*, 2(1):25–36, 1986.

[26] L. Porter, S. Garcia, H.-W. Tseng, and D. Zingaro. Evaluating student understanding of core concepts in computer architecture. In *Proceedings of the 18th Annual Conference on Innovation and Technology in Computer Science Education*, 2013.

[27] E. Raymond. The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12(3):23–49, 1999.

[28] A. Tew and M. Guzdial. Developing a validated assessment of fundamental CS1 concepts. In *SIGCSE*, 2010.

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

[30] A. E. Tew and M. Guzdial. The fcs1: a language independent assessment of cs1 knowledge. In *SIGCSE*, 2011.

[31] A. E. Tew, W. M. McCracken, and M. Guzdial. Impact of alternative introductory courses on programming concept understanding. 2005.

[32] K. C. Webb and C. Taylor. Developing a pre-and post-course concept inventory to gauge operating systems learning. In *SIGCSE*, 2014.