Developing Course-Level Learning Goals for Basic Data Structures in CS2

Leo Porter
University of California, San Diego
leporter@eng.ucsd.edu

Daniel Zingaro
University of Toronto Mississauga
daniel.zingaro@utoronto.ca

Cynthia Lee
Stanford University
cbl@stanford.edu

Cynthia Taylor
University of Illinois at Chicago
cynthiat@uic.edu

Kevin C. Webb
Swarthmore College
kwebb@cs.swarthmore.edu

Michael Clancy
University of California, Berkeley
clancy@eecs.berkeley.edu

ABSTRACT

Establishing learning goals for a course allows instructors to design course content to address those goals, helps students to focus their learning appropriately, and enables researchers to assess learning of those goals. In this work, we propose six learning goals for a topic prevalent in CS2 courses: Basic Data Structures. These learning goals arise from reviewing several CS2 courses at a variety of institutions, surveying faculty experts who commonly teach CS2, and meeting and working closely with these experts. We outline our process for creating learning goals, identify important topics underlying these goals, and provide examples of how the goals developed on the path to consensus. We also document that the term "CS2" does not have a unified interpretation within the CS education community and describe how this hurdle influenced our decision to focus on Basic Data Structures.

CCS CONCEPTS

• **Social and professional topics → Computing Education**;

KEYWORDS

CS2, data structures, learning goals

ACM Reference Format:
Leo Porter, Daniel Zingaro, Cynthia Lee, Cynthia Taylor, Kevin C. Webb, and Michael Clancy. 2018. Developing Course-Level Learning Goals for Basic Data Structures in CS2. In *SIGCSE '18: The 49th ACM Technical Symposium on Computing Science Education, February 21–24, 2018, Baltimore , MD, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159457

1 INTRODUCTION

Course-level learning goals capture what instructors expect students to know at the end of a course. They offer a number of benefits for instructors, students, and researchers. For instructors, these learning goals facilitate discussions of course outcomes, help convey to new instructors the core content in a course, and serve as a defensible basis for evaluating student learning [15]. For students, such goals can help identify what must be learned in the course and enhance learning [15]. Finally, researchers and curriculum designers can use these goals for programmatic assessment [13] and the design of concept inventories [2]. Unfortunately, learning goals are not frequently reported in the CS education literature.

Given the value of learning goals for instructors and students, this work seeks to codify a set of common learning goals in CS2 courses across a variety of institutions. We focus on CS2, rather than CS1, as there is comparably less known about CS2 goals and important topics [11].

Our team consisted of faculty at six different institutions working with CS2 experts (i.e., instructors and researchers of CS2) at an additional eight institutions. Through discussions with the experts, it became clear that the meaning of "CS2" depends on the particular institution. However, a common thread among CS2 variants is the teaching of Basic Data Structures.

To establish learning goals, we examined syllabi and final exams at our experts' institutions to determine common topics, surveyed these faculty experts about CS2 and Basic Data Structures topics, and organized multiple large meetings to discuss and refine learning goals for Basic Data Structures. We detail the methods and results found at each step of the process in this work.

The primary contributions of this work are:

- Discussion of the topics covered in variants of CS2.
- Analysis of the perceived importance of CS2 topics by instructors.
- Presentation of course-level learning goals for Basic Data Structures.

2 BACKGROUND

2.1 Learning Goals

Students often struggle to learn the content presented in class, but the culprit may not be the difficulty of the content itself. For example, students may lack requisite context or scaffolding, making it more difficult for them to organize new knowledge into the framework of what they already know [15]. Learning goals have been proposed as tools to enhance student-instructor communication and make explicit the instructor's aims for studying particular content.

Learning goals can target different levels of specificity, from topic-level to school-level [13]. The most specific are topic-level learning goals; for example, "at the end of this lecture, students will be able to briefly describe the parts of a web search engine" [15]. Such goals are valued by students: the goals explicitly communicate expectations to students and help them to prepare for upcoming

SIGCSE '18, February 21–24, 2018, Baltimore , MD, USA

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159457

exams. Furthermore, the existence of topic-level learning goals enhances collaboration between instructors and improves the quality of assessments [15].

Course-level learning goals state the outcomes expected of a student at the end of the course. While a topic-level goal may inform course-level goals, course-level goals are often more broad than topic-level goals and are not as closely tied to specific course topics.

Course-level goals have been shown to be important for effective intervention on struggling students [7] and are at the core of successful, evidence-based course transformation [4]. Course-level learning goals are also critical to the generation of *concept inventories* (CIs). A CI is a conceptual test of the level to which students demonstrate expert-level thinking [2]. Whereas the important topics drive the content of the CI, the goals determine what students are asked about the topics [5]. Key to the selection of learning goals and topics for a CI is often an expert panel; regular meetings, extended discussions, and an iterative workflow preclude reliance on more lightweight approaches such as community-based surveys. In a representative study in physics education, for example, a panel of 13 experts was used [5].

We draw a crucial distinction between course topics themselves and corresponding learning goals. Topics determine the material of importance; e.g., for CS1, important topics could include loops and conditionals [16]. Learning goals, on the other hand, specify what students are expected to be able *to do* with course material. To summarize, topic-level learning goals focus on what students can do with a particular topic whereas course-level learning goals focus on what students can do at the end of the course.

2.2 Important Concepts in Data Structures

The CS education literature contains many studies of important topics. Most concern CS1 [9, 14, 16]. One paper offers an analysis of important CS2 topics using survey data from CS2 instructors, but does not discuss learning goals [11]. (See Herman and Loui [10] for an innovative use of topics to argue a conceptual framework for digital logic.)

There is precedent for the importance of studying student conceptions of data structures. For example, Karpierz and Wolfman [12] identify prevalent misconceptions related to searching binary search trees (BSTs) and assumptions that BSTs are always balanced. Others note student difficulties with distinguishing between BSTs and heaps [6] and checking invariants of BSTs [8].

2.3 Curricular Guidelines for Data Structures

Curriculum documents typically state both topics and "learning outcomes" (similar to learning goals); indeed, we often referred to the CS 2013 Curriculum [1] throughout the development of our work. However, such curriculum documents are necessarily comprehensive and cover knowledge units that span multiple courses, or include topics that are not covered consistently in the same course at representative institutions [1]. Additionally, some institutions have developed learning goals for their local courses [3]. In this paper, we focus on topics and learning goals that underlie a common core of CS2-level expertise suitable for use across institutions.

3 RESEARCH QUESTIONS AND CONTEXT

In this work, we examine three principal research questions:

- **RQ1:** Which topics are commonly taught in CS2?
- **RQ2:** Which of these common topics are valued the most by CS2 instructors?
- **RQ3:** What are the course-level learning goals for these central CS2 topics?

In beginning to answer these questions, it became apparent that a step-by-step process was required (i.e., progress on RQ1 was needed before progress could be made on RQ2). As such, this paper provides both the research methods and results for each step in the sequence. Throughout this work, as we seek to characterize the topics and learning goals underlying many CS2 courses, we strive for programming language-independence. That is, similar to research on CS1 learning goals [16], we focus on what students should be able to do independent of the specifics of any particular programming language.

3.1 Expert Panel

We formed an expert panel of CS2 instructors from a broad set of institutions and consulted with our panelists throughout the research process. The members of our expert panel have extensive experience teaching CS2, with one instructor having taught it for 31 years. In addition, they are active in the CS education research community. Further characteristics of the panelists and their institutions are provided in Table 1.

As the six organizers and authors of this study, we interpreted results from the expert panel and conducted follow-on discussions with expert panel members. As such, the diversity of our backgrounds both contributes to and informs the analysis of results. Four of us are currently at large, public research-intensive universities. Three of us are either presently at, or previously held professor positions at, small liberal arts colleges. Our current institutions range from highly selective private schools, to minority-serving public research universities (where a majority of students are Pell-grant eligible, first-generation college students). Years of teaching varies from 5 to 34 years. All are active in the CS education community, and four have taught CS2 multiple times in person or online.

4 RQ1: COMMON CS2 TOPICS

In the summer and fall of 2015, we approached members of the expert panel and requested syllabi and final exam materials for their CS2 courses. Eight of the nine members of the panel responded with the requested materials. We then reviewed the syllabi, identifying major topics addressed in each CS2 course. After generating a large list of topics, we worked together to cluster the topics into larger components. This clustering was informed by chapter organization of relevant CS2 textbooks, our expertise as CS2 instructors, and the Computing Curriculum 2013. Six major components of CS2s emerged, which we summarize in Table 2. Note that each component appears in some, but not necessarily all, of our experts' CS2 courses, as illustrated in Table 3.

As we reviewed the courses and further discussed with our experts, it became clear that the definition of CS2 was not homogeneous. Rather, there appeared to be two largely disjoint courses that are referred to in the CS education community as CS2. The

Table 1: Instructor, Course, Institution Characteristics.

Identifier	A	B	C	D	E	F	G	H	I
Institution[a]	CC	RIU	PUI	LAI	LAI	RIU	RIU	CC	RIU
Class Sizes[b]	Small	Large	Small	Small	Small	Large	Large	Small	Large
Public / Private	Public	Public	Public	Private	Private	Public	Private	Public	Public
Language	Java	C/C++	Java	Java	Java	C/C++	Java	Java	Java

[a] Institutions categorized as Community College (CC), Liberal Arts Institution (LAI), Primarily Undergraduate Institution (PUI), and Research-Intensive University (RIU).

[b] Class sizes of typically less than 30 students considered small. Class sizes greater than 100 considered large.

Table 2: Broad Topics

Component	Description
Object-Oriented Programming	Class Design, Inheritance, Polymorphism
Basic Data Structures	List, Stack, Queue, Array, Linked-List, Binary Tree, Binary Search Tree
Recursion	Recursive strategies, methods, and/or their efficiency
Sorting	Sorting algorithms - O(n^2) and/or O($n * log(n)$) approaches
Algorithm Analysis	Runtime analysis and/or space complexity
Advanced Data Structures	Balanced Trees, Hash Tables, Heaps, Priority Queues, Graphs, Union-Find, Minimum Spanning Trees

Table 3: Broad Topics by Course

Component	A	B	C	D	E	F	H	I
Object-Oriented Programming	X	X	X					X[a]
Basic Data Structures	X	X[b]	X	X	X	X	X	X
Recursion	X		X	X	X	X		X
Sorting				X	X	X	X	X
Algorithm Analysis				X	X	X	X	X
Advanced Data Structures					X	X	X	X

[a] Object-Oriented Programming is taught previously in a different language so there is a brief discussion of how Object-Oriented Programming works in Java at the start of this course.

[b] This course has an extensive treatment of Object-Oriented Programming, with comparably less time dedicated to Basic Data Structures.

first heavily emphasizes object-oriented programming and its related concepts. The second heavily focuses on sorting, algorithm analysis, and data structures. This is perhaps most evident at one of our panelist institutions that in fact offered two different "CS2" courses. At that institution, we began with the first course (course B per Table 3), but noted the absence of some topics commonly seen in other CS2 courses. When we checked with that instructor, we were pointed to a follow-on course at the same institution, course F, which addressed other CS2 topics. We retained both of that institution's experts in our panel to accurately represent both perspectives. In general, combining all of the topics covered in our experts' CS2 courses, we see that there is too much for a single CS2 course.

That said, Basic Data Structures and, to a lesser degree, Recursion are common to both the object-oriented programming and the algorithm analysis/data structures variants of CS2. Commonality alone, however, does not necessarily warrant creation of course-level learning goals unless the topics are also important.

5 RQ2: IMPORTANCE OF CS2 TOPICS

Results from the prior section identify Basic Data Structures and Recursion as topics common to the variants of CS2 that we encountered. But those results say nothing about the importance to these topics ascribed by the experts. Thus, we surveyed the expert panel about the importance of each component for their courses.

We asked participants the following questions for each component from Table 2:

- How critical is the following topic for student success in YOUR CS2 class?
- What is the importance that students know the content of the following topic as prerequisite for courses that follow YOUR CS2 class?

Participants responded on a 7-point Likert scale where 1 was "not at all critical" and "not at all important" and 7 was "absolutely essential" and "very important". All nine experts participated in the survey. The average ranking for each topic by question appears in Table 4.

Results from Table 4 confirm that Basic Data Structures is indeed an important topic both for student success in CS2 and in follow-on courses. One might be surprised at the high scores for object-oriented (OO) programming for success in CS2, particularly as some of our experts' curricula cover some OO concepts in CS1. Based on feedback provided by our experts at the end of the survey, it was clear that the high level of importance for OO concepts was viewed by some participants as knowledge prerequisite to CS2.

Table 4: Average Importance of each component for student success in CS2 and as a prerequisite for later courses.

Component	CS2 Success	As Prereq.
Object Oriented Prog.	6.4	5.9
Basic Data Structures	6.4	5.7
Recursion	6.3	5.9
Sorting	5.3	3.5
Algorithm Analysis	5.1	3.6
Advanced Data Structures	3.9	2.3

6 RQ3: LEARNING GOALS FOR BASIC DATA STRUCTURES

Having found common CS2 course components and established their importance, we now report on our methods and results for generating learning goals for Basic Data Structures.

6.1 Method

We began by constructing an initial set of eleven learning goals to foster discussion among the expert panel. These initial goals were based on a review of final exams, the Computing Curriculum 2013, and project team discussions. The development process proceeded by conferring with the expert panel through surveys, meetings at professional conferences, and one-on-one discussions. After receiving such feedback, the project team iteratively discussed and refined the goals.

At various points, we consulted with an outside expert, Stephanie Chasteen, who has a background in developing learning goals for physics education. This expert was critical for ensuring that best practices for goal development were being followed and that our goals were at the right level of generality.

The most significant steps of our process were:

(1) Development of the initial draft of eleven learning goals.
(2) Survey of the expert panel for feedback.
(3) Meeting at the SIGCSE 2016 conference with the expert panel to gather additional feedback.
(4) Presentation of the revised six goals as an interactive poster at the ICER 2016 conference to gather informal feedback from the community. Attendees could approach the poster and score each learning goal as either very important to their course, somewhat important, not at all important, or "not important now but should be". Feedback for the six goals was quite positive: the majority of participants selected very important for five goals and, for the remaining goal, that it should be important.
(5) Meeting at the SIGCSE 2017 conference with the expert panel, where our panelists provided universally positive feedback and suggested only minor revisions.

This method mirrors similar practices in determining common course-level goals: using a working group with course and pedagogy expertise, and frequent opportunities for discussion and modification of goals [5].

6.2 Examples of Goal Development

It is difficult to summarize conversations across multiple years and multiple drafts of goals. Learning goal development often followed a progress of slow change, with many small steps that are not noteworthy in themselves. Nonetheless, this section aims to provide intuition to the reader about how our goals evolved from their initial instantiation to their final form.

Specific Data Structures and Goals. Early versions of the learning goals mentioned specific interfaces or data structures. For example, one learning goal spoke specifically to an ability to write code to implement operations on a binary search tree. However, this led to repeating the same goal for a variety of data structures and pushed the goals to become more topic-level than course-level. This approach also led to learning goals that were very specific to particular data structures, limiting their applicability to only courses that covered precisely the same data structures.

Through work with the expert panel at SIGCSE 2016 and conversations within the project team, we determined that the learning goals should be generic so as to apply to any reasonable set of topics within Basic Data Structures. This simplified the discussion about the learning goals themselves, and also led us to explore the topics that are commonly taught. Hence, we provide the commonly taught elements of Basic Data Structures in Section 6.5.

Interface versus Implementation. In the meeting with experts at SIGCSE 2016, a division quickly emerged among the panel regarding the facets of Basic Data Structures that matter the most. In particular, there were energized discussions on the right level of abstraction on which to focus. Do we care more about students' learning to select the appropriate interface (e.g., stack, queue, or list), selecting the appropriate implementation for a given interface for best performance (e.g., linked list or array-based list), or learning how to implement the data structures themselves? Ultimately, we struck a balance in which we agreed that students should be able to do all of these, and the resulting learning goals encompassed all three of these elements.

Consolidating Multiple Goals. Our many conversations with the expert panel enabled us to consolidate the initial goals (which admittedly border on topic-level rather than course-level goals) into broader, more widely applicable goals. An example of this process started with the following two initial goals:

- "Implement insertion into, location in, and deletion from linear data structures."
- "Implement insertion into, location in, and deletion from a binary search tree."

They became this final goal: "Design and modify data structures capable of insertion, deletion, search, and related operations."

To merge these two goals in this way, we first combined "linear data structures" and "binary search tree" into simply data structures. Next, we noted that the expert panel viewed "implement" as too narrow a goal: they wanted students to be able to design data structure operations, not just implement a given design. Also, limiting students to just insertion, deletion, and search was viewed as too narrow as students are often asked to work with other operations besides these; the phrase "related operations" was therefore added.

Table 5: Topics in Basic Data Structures by Course

Topic	A	B	C	D	E	F	H	I
Stacks	X		X	X	X	X	X	X
Queue	X		X	X	X	X	X	X
List	X	X	X	X	X	X	X	X
Set	X		X					X
Interface/Abstract Data Type			X	X	X			X
Arrays	X	X	X	X	X	X	X	X
Array-based List	X	X	X	X	X			X
Linked List	X		X	X	X	X	X	X
Binary Tree	X			X	X	X	X	X
Binary Search Tree	X			X	X	X	X	X

6.3 Resultant Goals

We now present the learning goals as agreed upon by our experts.

At the end of a course on Basic Data Structures, students should be able to:

(1) Analyze runtime efficiency of algorithms related to data structure design.
(2) Select appropriate abstract data types for use in a given application.
(3) Compare data structure tradeoffs to select the appropriate implementation for an abstract data type.
(4) Design and modify data structures capable of insertion, deletion, search, and related operations.
(5) Trace through and predict the behavior of algorithms (including code) designed to implement data structure operations.
(6) Identify and remedy flaws in a data structure implementation that may cause its behavior to differ from the intended design.

6.4 Discussion of Learning Goals

The learning goals presented in the preceding section have gained agreement within the project group and expert panel and were the result of a number of rigorous discussions. This section contextualizes and shares some insights from those discussions.

As mentioned in Section 6.2, our experts debated the side of the abstraction layer on which students should focus: deciding on the correct interface to help solve a problem or, separately, deciding on the best implementation of that interface. We resolved this contention by including both Goal 2 and Goal 3, which are similar but focus on distinct levels of abstraction.

Reflecting its importance in many CS2 courses, programming is present as part of Goals 4–6. Goal 4 includes implementing data structures in code; Goal 5 includes tracing and understanding data structure code; and Goal 6 includes testing and debugging code.

In discussions, Goal 5 and Goal 6 have generally been perceived as connected with the other goals. The reason for this is that if one wishes to assess student learning of Goals 1–4 in a manner involving code, it is hard to see how Goals 5 and/or 6 would not also be involved. Through discussions with our expert panel and our outside consultant, we have come to agree that the goals need not be entirely independent in terms of assessing student learning.

6.5 Common Topics in Basic Data Structures

As previously mentioned, the decision to decouple course-level learning goals from the course topics greatly improved the quality and applicability of the learning goals. However, we believe it is helpful to contextualize these learning goals with the topics to which they are frequently applied.

Following the same methods as in Section 4, we referred to eight expert panel syllabi and final exams to determine the topics from Basic Data Structures that commonly appear in CS2 courses. Table 5 provides topic coverage by course. This table shows that stacks, queues, lists, arrays, array-based lists, linked lists, binary trees, and binary search trees are all common data structures taught in CS2.

There was much discussion among the project team, and subsequently with the expert panel, about the line between Basic and Advanced Data Structure topics. We decided that those topics that rely on earlier data structures (e.g., balanced trees rely on BSTs, heaps rely on arrays) and those that often appear in later courses (e.g., graphs, minimum spanning trees) were considered Advanced topics. Two topics that seemed to straddle the border of Basic and Advanced Data Structures were the creation of iterators over structures (found in 3 out of 8 courses) and hash tables (found in 4 out of 8 courses). The lack of majority agreement resulted in these topics being excluded as not being commonly-taught topics in CS2 Basic Data Structures.

7 DISCUSSION

CS2 Terminology. As discussed previously, we learned through the development process that "CS2" has, in fact, two common interpretations. The first is a course focused on object-oriented programming; the second is a course focused on data structures and algorithm analysis. This lack of agreement on a common term within the CS education community is disappointing but does have some precedent in the literature [11]. We encourage the community to work toward an agreement regarding CS2 terminology. An interim fix may be simply to say CS2-OOP or CS2-DS to refer to the two common variants.

Intended Use of Learning Goals. We imagine two primary uses of the learning goals provided here. The first is for researchers aiming to assess student learning across multiple institutions. For this group, the learning goals can provide a common basis on which one

might develop concept inventories or other assessments of student learning. The second is for CS2 instructors and curricular designers looking to create their own learning goals or assessments. For this second group, we recognize that the learning goals provided here may not be a perfect fit for existing courses or for all CS curricula; however, we hope that they provide a strong first step in course and curricular design efforts. Most notably, we recognize that Basic Data Structures is simply one of many components that constitute a CS2 course. Future work is required to generate goals for other components; we hope that our methods and results provided here are useful in such efforts.

Programming Language-Independence. In order for the learning goals to support development of assessments of student learning across multiple institutions, the learning goals needed to be programming language-independent. Several different programming languages are represented among the CS2 courses of our expert panelists' institutions. One benefit of focusing on learning goals for Basic Data Structures is that this topic is amenable to the creation of learning goals that are not tied to a particular language. Abstract data types implemented by Basic Data Structures (e.g., stacks and queues) are by definition language-independent abstractions. In contrast, we suspect that it is much more difficult to create learning goals for object-oriented programming, with its emphasis on class design and inheritance, without resorting to language-specific considerations.

Future Plans. There are topics that many instructors consider important but which lie outside the boundary of Basic Data Structures. We are looking ahead at ways to implement *modules* of learning goals (organized as in Table 2) that would allow instructors more flexibility for matching their courses to available learning goals.

Challenges of Multi-Institutional Efforts. Our project empaneled the experts over a multi-year period during which we sought their attendance at professional conferences, asked them to fill out many surveys, requested course materials, and engaged them in discussions as the learning goals developed. We occasionally but only temporarily "lost" some experts — missing a survey here or a meeting there — but this feels inevitable in a multinational, multi-year project such as this one. Overall, our advice to those conducting similar research is to build an expert panel of those truly interested in the betterment of a course. We were fortunate to have such an expert panel helping to guide our work.

8 CONCLUSION

Through conversations with experienced CS2 instructors, we classified the elements that are central to CS2 courses and establish learning goals for those elements. We discovered that "CS2" actually has two general interpretations by members of the computing community: one is a course focused on object-oriented programming and the other is a course focused on data structures and algorithm analysis. Fortunately, Basic Data Structures (linked lists, array-based lists, stacks, queues, etc.) is common to both course variants, and is viewed as highly valuable for supporting student outcomes in CS2 and subsequent courses.

Given the importance of Basic Data Structures to the computing curriculum, we then worked with those same experienced CS2 instructors to produce a set of course-level learning goals. These course-level learning goals may be valuable to educational researchers for development of student assessments; and for practitioners wishing to assess their students, design their own learning goals, or make curricular changes.

We conclude by observing that there have been few efforts within computer science education to gain multi-institutional agreement on course-level learning goals for computing. Given the benefits from such work, we see this as a research gap deserving further attention from the CS education research community.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the contributions of the following collaborators: Meghan Allen, Owen Astrachan, Darci Burdge, Stephanie Chasteen, Maureen Doyle, John Glick, Paul Hilfinger, Kate Sanders, Paramsothy Thananjeyan, and Steve Wolfman. This work was supported in part by NSF award 1505001.

REFERENCES

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula. Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press, December 2013.
[2] W. Adams and C. Wieman. Development and validation of instruments to measure learning of expert-like thinking. *International Journal of Science Education*, 33(9):1289–1312, 2011.
[3] Carl Wieman Science Education Initiative at the University of British Columbia. Computer science learning goals. www.cwsei.ubc.ca/Files/ComSc_LG/CPSC_Learning_Goals.pdf, 2008.
[4] S. Chasteen, K. Perkins, P. Beale, S. Pollock, and C. Wieman. A thoughtful approach to instruction: Course transformation for the rest of us. *Journal of College Science Teaching*, 40:24–30, 2011.
[5] S. V. Chasteen, R. E. Pepper, M. D. Caballero, S. J. Pollock, and K. K. Perkins. Colorado upper-division electrostatics diagnostic: A conceptual assessment for the junior level. *Physical Review Special Topics - Physics Education Research*, 8(2), 2012.
[6] H. Danielsiek, W. Paul, and J. Vahrenhold. Detecting and understanding students' misconceptions related to algorithms and data structures. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, pages 21–26, 2012.
[7] L. Deslauriers, S. E. Harris, E. Lane, and C. E. Wieman. Transforming the lowest-performing students: an intervention that worked. *Journal of College Science Teaching*, 41:80–88, 2012.
[8] A. Fekete. Using counter-examples in the data structures course. In *Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20*, pages 179–186, 2003.
[9] 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. *SIGCSE Bulletin*, 40(1):256–260, 2008.
[10] G. L. Herman and M. C. Loui. Identifying the core conceptual framework of digital logic. In *Proceedings of the 2012 American Society for Engineering Education Annual Conference and Exposition*, pages AC2012–4637, 2012.
[11] M. Hertz. What do "CS1" and "CS2" mean?: Investigating differences in the early courses. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, pages 199–203, 2010.
[12] K. Karpierz and S. A. Wolfman. Misconceptions and concept inventory questions for binary search trees and hash tables. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 109–114, 2014.
[13] P. Marsh. What is known about student learning outcomes and how does it relate to the scholarship of teaching and learning? *International Journal for the Scholarship of Teaching and Learning*, 1(2), 2007.
[14] C. Schulte and J. Bennedsen. What do teachers teach in introductory programming? In *Proceedings of the Second International Workshop on Computing Education Research*, pages 17–28, 2006.
[15] B. Simon and J. Taylor. What is the value of course-specific learning goals? *Journal of College Science Teaching*, 39(2):52–57, 2009.
[16] A. E. Tew and M. Guzdial. Developing a validated assessment of fundamental CS1 concepts. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, pages 97–101, 2010.