Developing a Concept Inventory for Computer Science 2: What should it focus on and what makes it challenging?

Lea Wittie
Dept. Comp. Sci.
Bucknell University
Lewisburg, PA
USA
lwittie@bucknell.edu

Anastasia Kurdia
Dept. Comp. Sci.
Tulane University
New Orleans, LA
USA
akurdia@tulane.edu

Judy Peng
Dept. Comp. Sci.
Bucknell University
Lewisburg, PA
USA

James Kelly
Dept. Comp. Sci.
Bucknell University
Lewisburg, PA
USA

Meriel Huggard
School of Comp. Sci. and Stat.
Trinity College Dublin
Dublin 2
Ireland
meriel.huggard@tcd.ie

***Abstract*—This Work-In-Progress Research Paper reports on an international study that is being undertaken in order to develop a validated concept inventory for the second introductory computer science course (CS2).**

A concept inventory is a research-based multiple-choice test that measures a student's knowledge of a set of concepts while also capturing conceptions and misconceptions they may have about the topic under consideration. Development of a concept inventory for a course requires identifying course topics that are both difficult and important. This paper details how the Delphi method is being used to develop a concept inventory for CS2; in particular, it focuses on the initial process of identifying the set of topics that should be covered by a concept inventory for CS2.

***Index Terms*—Concept inventory, Delphi method, Computer Science 2, CS2, Assessment**

I. Introduction

Almost every computer science program contains two semester-long introductory courses, usually named Computer Science 1 (CS1) and Computer Science 2 (CS2). While CS1 focuses on basic coding and computational thinking, CS2 moves beyond basic programming skills to an exploration of data structures, algorithms, and design. A quick literature review quickly reveals that there is a huge numerical preponderance of articles that focus on CS1 [1] [2] [3] [4] rather than on CS2; though aspects of CS2 have begun to receive more attention in the literature in recent years [5], [6], [7]. This may be because there is broad agreement on the key elements of CS1, while the curriculum for CS2 can vary significantly between institutions [8].

It can be argued that the intent behind the teaching of CS2 is open for debate – is its purpose to acquaint students with a standardized set of tools and frameworks, or it is to give them an appreciation of the design of such frameworks and the implementations underpinning them? One phenomenographic [9] study of this question [10] identified five categories of intent in the teaching of CS2: "developing transferable thinking, improving students' programming skills, knowing 'what's under the hood', knowledge of software libraries, and component thinking". Regardless of the purpose behind the teaching of CS2, it has been a mandatory element of the ACM Computing Curriculum since 1978 [11] and is likely to remain so for many years to come.

Most undergraduate courses, including CS2, come with a clearly identified set of prerequisites that ensure that those taking the course have the skills needed to successfully navigate and complete it. Underpinning these prerequisites is an expectation that students have acquired a set of preconceptions that enable them to develop a deeper understanding and appreciation of the concepts they will meet in the course of their studies. However, many students also enter each course with misconceptions that can impede their progress [12] and led them to become frustrated and disengaged. In order to address these deficiencies, instructors must first identify them and then develop classroom interventions and strategies that address them [13]. One way of doing this is through the use of a concept inventory.

A concept inventory (CI) seeks to determine how well each student's conceptual framework matches an accepted conceptual framework of the given subject [14]. This is achieved using a research-based multiple-choice test where each question includes one correct answer and a set of incorrect answers that result from misconceptions of the topic.

This work in progress reports on an international study [15], [16] that is being undertaken in order to develop a concept inventory for CS2. When the authors' embarked upon this study in 2017, there had been little progress towards the development of a concept inventory for CS2, since then once such concept inventory has been developed [7]. It focused solely on North American institutions and the six authors drew on a small expert panel of nine faculty members drawn from a range of institutions. In addition, the study drew on interviews with 50 students from 3 separate institutions. By contrast, the study reported on in this work draws upon an international pool of expert faculty members and is closely following a Delphi process [17], a well-established process for creating concept inventories [14], [18], [13], [19]. Once the study reported on in this paper is complete, it will broaden

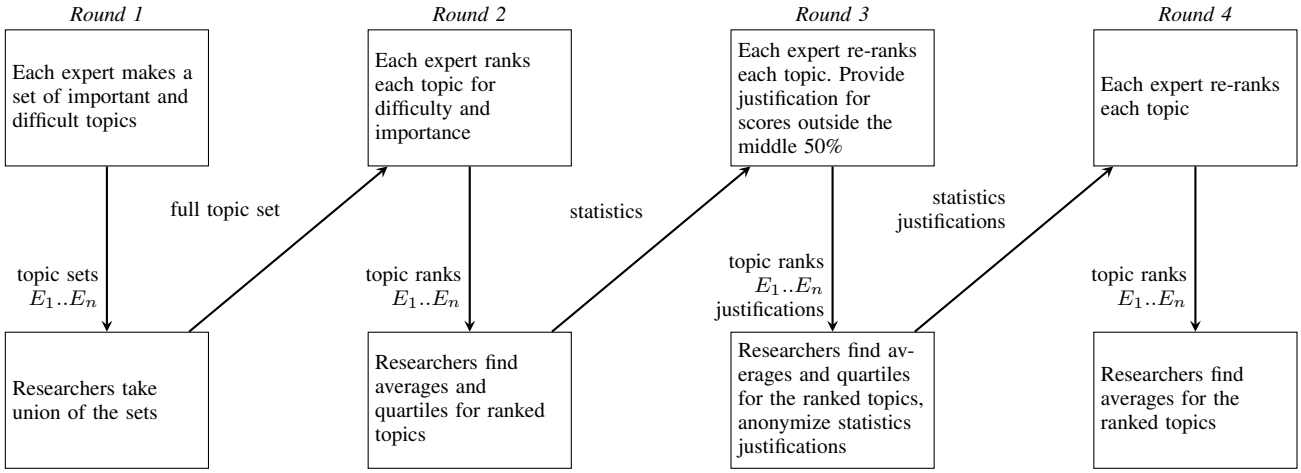

Fig. 1. Delphi method workflow for identifying the topics for CS2 concept inventory. The steps outlined in top and bottom rectangles in each column constitute one round of the Delphi method.

the understandings developed in [7].

II. Progress

A. Methodology: the Delphi method

In order to identify the topics for inclusion in the CS2 concept inventory, this study uses the Delphi method. The Delphi method is a systematic iterative process of arriving at a common opinion or decision by a diverse group of experts [20]. By seeking consensus amongst the expert group, the Delphi method is designed to eliminate the natural bias that any individual expert will have based on their experience and field of expertise [13]. At each round, the experts are provided with a questionnaire and an aggregate group response at the preceding round. The latter may affect the experts' responses at a current round. After several rounds, the group's response closely reflects the consensus opinion of the group. By giving each expert's opinion equal weight and through the use of anonymous feedback, each expert is only influenced by the opinions and arguments put forward by other experts, and not by their reputation in the field [21].

In this study, the experts are drawn from a pool of experienced CS2 instructors. They were first asked to identify C2 topics that have relatively high importance in the course and are difficult for the students to master. After that they rank the topics by difficulty and separately, by importance, using the feedback of other experts aggregated by the researchers. Two more ranking refinement rounds follow as outlined in Figure 1 and detailed in the rest of this section.

B. Preliminary step: Recruiting and selecting experts

We recruited the experts through a combination of emails, advertising at conferences, and networking. Emails went out to over 400 academics teaching CS2 worldwide, identified by scanning various university websites. We also sent emails to CS2 textbook authors and academics who published CS2 related research. We gave a Lightning talk [16] and also reached out to our network of contacts. Diversity was a key point in our recruiting attempts. We actively recruited to get academics from different continents (North America, South America, Asia, Oceania (New Zealand, Australia), Europe, and Africa). Overall, we enlisted 34 initial experts.

The initial experts filled out a survey and described the major portion (60% or more) of their CS2 as an introduction to either object-oriented programming or data structures. The data structures choice was further divided into two categories: using the data structures provided by libraries/APIs and implementing the data structures (and likely using them too). 25 of the initial experts returned the completed survey. Of those, 19 experts indicated the category of data structures and implementation (our concept inventory target).

We sent out an envelope information survey to those 19 "data structures and implementation" experts and received 18 responses. 17 of the responders identified as educators in the classroom, 6 as researchers in CS educational topics, one was a textbook author, and one had taught CS2 recently but not currently; several experts selected more than one role. All of our experts were at 4-year institutions; 8 at institutions with large graduate programs and 10 at primarily undergraduate institutions. 17 were at institutions in North America and 1 in Asia. 3 were female and 13 were male. We have achieved some diversity with respect to gender, institution level (graduate and undergraduate), geographic location, and role (teachers, researchers, textbook authors).

C. Round 1: Topic set generation

For Round 1 of our application of the Delphi method, each expert generated a list of 10-15 topics they considered to be both difficult and important in CS2 (the leftmost box in the top row of Figure 1). They were asked to consider their entire CS2 course, not just the portion focused on data structures. 17 experts completed this step.

When analyzing Round 1 responses we found that the joint topic set was very large. We also observed that the judgment whether two answers were describing essentially the same

TABLE I
TOPIC CATEGORIES

Abstraction and ADT	Array
Analysis and Big O	Graph
Map and Hash Table	Heap
Recursion	List
Searching	Memory
Software Engineering	Set
Sorting	Tree
Stack and Queue	Other

topic or two different topics was very subjective. Our initial plan for the analysis portion of Round 1 was to produce a joint set of topics mentioned by 2 or more experts. However, we wanted to preserve the diversity of opinions that the experts had to offer. Additionally, there could have been topics considered by several experts that only one expert chose to include in the list at this round. For these reasons, we decided to keep all topics mentioned by the experts in the joint set of topics (the leftmost box in the bottom row of Figure 1). Topics that are of true interest only to one expert will be eliminated after Round 2 is completed.

After removing duplicate answers, the joint set has 120 topics. To aid the experts in ranking them at Round 2, they were partitionedinto 16 categories as shown in Table I.

D. Round 2: Ranking the topics by difficulty and importance

At Round 2 experts are asked to rank the topics from the joint set by difficulty and, separately, by importance (the second box in the top row of Figure 1). Each expert received a spreadsheet file of the 120 topics organized into 16 categories. The order of categories and the topics inside each category were randomized for each expert individually to prevent the ordering biases from influencing the results. The ranks such as "80^{th} most important topic" or "120^{th} most difficult topic" are both hard to determine and of little use for the purpose of the study. Therefore, the experts were asked to select 30 of the most difficult topics and rank them by difficulty, and separately, select 30 of the most important topics and rank them by importance, and then to return these two ranked lists.

As of April 2020, 9 experts have returned their Round 2 lists. Two experts have indicated that due to issues handling the COVID-19 pandemic, they will not be able to complete phase 2 but hope to rejoin the study at a later date. The remaining 6 experts are still working on Round 2.

III. CURRENT OBSERVATIONS

Analysis of Round 2 responses makes it clear that there isn't a single topic that all experts agree is important. Figure 2 shows ten of the most frequently occurring topics in the selections of the 9 experts who completed Round 2. Binary search (Searching category) and Hash tables (Map and Hash Table category) have the most occurrences, with 7 of 9 experts putting them in their list of top 30 most important topics.

It is of note that there is no single topic that all experts agree is challenging. Figure 3 shows ten of the most frequently occurring difficult topics in the difficulty rankings. Analysis of Recursive Algorithms (Analysis and Big-O category) has the most occurrences with 7 of 9 experts putting it in their top 30 most difficult topics. Hash tables and Recursion (Recursion category) appear in both top 10 lists making them the two topics that experts consider both difficult and important.

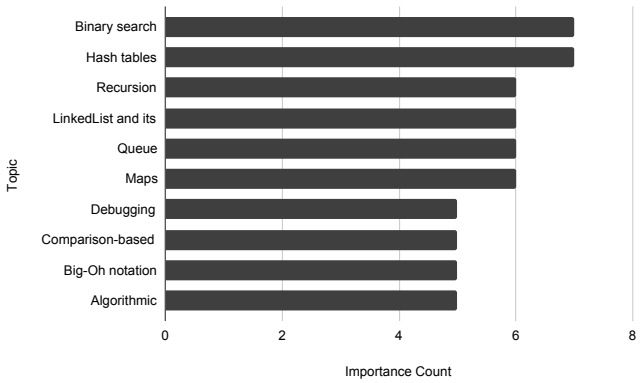

Fig. 2. Top ten most important topics as identified at Round 2 of Delphi method application ordered by the number of occurrences in experts' importance rankings

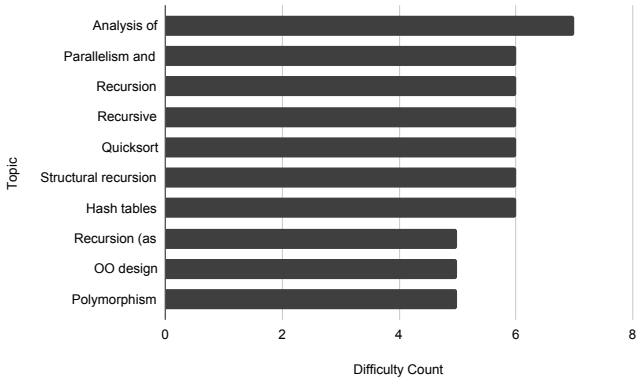

Fig. 3. Top ten most difficult topics as identified at Round 2 of Delphi method application ordered by the number of occurrences in experts' difficulty rankings

Figure 4 shows the relative importance of the topic categories. Each category was given a "popularity" score reflecting how often a topic was selected from this category.

$$popularity = \frac{selection\_count}{n \times category\_size}$$

$selection\_count$ is the number of times the topic was selected by different experts, $category\_size$ is the number of topics in this category, and n is the number of participating experts. Experts found Heaps, Recursion, and Searching to be the most important categories. Figure 5 illustrates the relative difficulty of the topic categories. The experts found Recursion, Heaps, and Analysis & Big-O to be the most difficult categories. It is interesting to note that Heaps and Recursion are on or near the top in both metrics, suggesting that Heap and Recursion

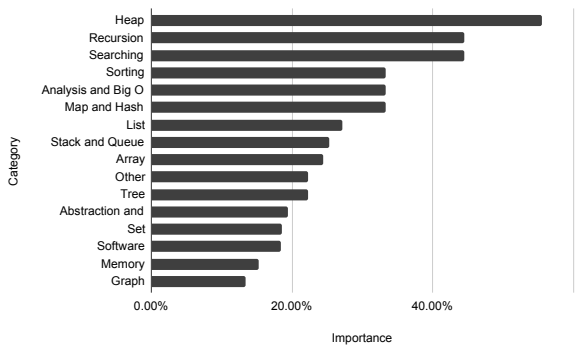

Fig. 4. Categories ordered by relative popularity in importance ranking

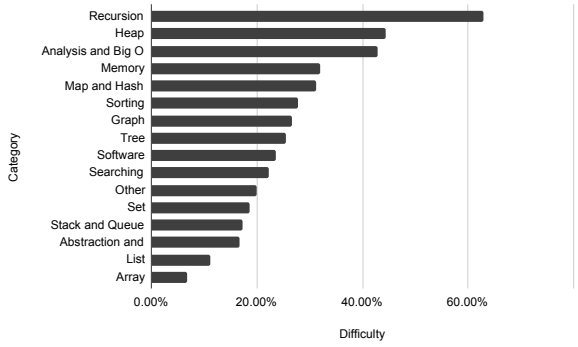

Fig. 5. Categories ordered by relative popularity in difficulty ranking

might be the most important and difficult category of topics taught in CS2. These are possibly the areas where pedagogical research could see the biggest impact on the students.

IV. Next steps and conclusions

After the remaining experts complete Round 2 topic rankings, we will find averages and quartiles for the topics that make the top 30 in a ranking. We will supply this data to the experts in Round 3 (the third-from-the-left box in the top row of Figure 1) and ask them to re-rank the topics and provide justifications for any scores they give that are outside the middle two quartiles.

Despite CS2 being a common and required course in any computer science program, there are no standards or universal agreement on what topics this course should cover. Three common foci of the various versions of the course are object-oriented programming, data structures and their usage, data structures and their implementation. (The latter one appears to prevail). There is a huge variety of topics that different instructors consider difficult and important (120 topics were named by 18 experts in our study). There is no single topic that all instructors consider the most difficult or the most important. This presents a challenge for developing a CS2 concept inventory that is intended for use in different institutions and by different instructors.

TABLE II
Important and/or Difficult Topics mentioned in this paper

Algorithmic complexity	Binary Search
Analysis of recursive algorithms	Debugging
Big-Oh notation & Hash tables	Maps
Comparison-based sorting	OO design
LinkedList and its implementation	Polymorphism
Parallelism and data structures	Queue
Recursive backtracking	Quicksort
Structural recursion on trees	Recursion
Recursion (as applied to usage in processing data structures)	

Based on the preliminary findings of the first two rounds of the Delphi process, we note that such CS topics as recursion, heaps, parallelism, sorting and searching, big-O notation and asymptotic algorithm analysis, linked lists, queues, maps and hashing are among most difficult and important ones in the CS2 courses (Table II). These are the topics that merit further research and that we anticipate our resulting concept inventory will focus on.

References

[1] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, "A survey of literature on the teaching of introductory programming," in *Working group reports on ITiCSE on Innovation and technology in computer science education*, 2007, pp. 204–223.

[2] M. Huggard, "Programming trauma: can it be avoided," *Proceedings of the BCS Grand Challenges in Computing: Education*, pp. 50–51, 2004.

[3] O. Ezenwoye, "What language? - the choice of an introductory programming language," in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–8.

[4] M. Huggard and C. McGoldrick, "Incentivising students to pursue computer science programmes," in *36th Annual Frontiers In Education Conference (FIE)*, Oct 2006, pp. 3–8.

[5] L. Layman, Y. Song, and C. Guinn, "Toward predicting success and failure in cs2: A mixed-method analysis," in *Proceedings of the 2020 ACM Southeast Conference*, 2020, pp. 218–225.

[6] L. Yeomans, S. Zschaler, and K. Coate, "Transformative and troublesome? students' and professional programmers' perspectives on difficult concepts in programming," *ACM Trans. Comput. Educ.*, vol. 19, no. 3, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3283071

[7] "BDSI – A New Validated Assessment for Basic Data Structures," https://computinged.wordpress.com/2020/02/24/bdsi-a-new-validated-assessment-for-basic-data-structures-guest-blog-post-from-leo-porter-and-colleagues/, 2020.

[8] 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*, 2010, pp. 199–203.

[9] I. Stamouli and M. Huggard, "Phenomenography as a tool for understanding our students," in *International Symposium for Engineering Education*, 2007.

[10] R. Lister, I. Box, B. Morrison, J. Tenenberg, and D. S. Westbrook, "The dimensions of variation in the teaching of data structures," in *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ACM, 2004.

[11] R. H. Austing, B. H. Barnes, D. T. Bonnette, G. L. Engel, and G. Stokes, "Curriculum '78: Recommendations for the undergraduate program in computer science— a report of the acm curriculum committee on computer science," vol. 22, no. 3, 1979.

[12] K. C. Webb and C. Taylor, "Developing a pre- and post-course concept inventory to gauge operating systems learning," in *Proc. 45th ACM SIGCSE*. ACM, 2014, pp. 103–108.

[13] M. A. Nelson, M. R. Geist, R. L. Miller, R. A. Streveler, and B. M. Olds, "How to create a concept inventory: The thermal and transport concept inventory," in *Annual Conf. of American Edu. Research Association*, 2007.

[14] M. F. Farghally, K. H. Koh, J. V. Ernst, and C. A. Shaffer, "Towards a concept inventory for algorithm analysis topics," in *Proc. ACM SIGCSE*. ACM, 2017, p. 207–212.

[15] L. Wittie, A. Kurdia, and M. Huggard, "Developing a concept inventory for computer science 2," in *2017 IEEE Frontiers in Education Conference (FIE)*, 2017, pp. 1–4.

[16] L. Wittie, A. Kurdia, and M. Huggard, "Recruiting experts: Toward a concept inventory for computer science 2 (abstract only)," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1104. [Online]. Available: https://doi.org/10.1145/3159450.3162210

[17] M. J. Clayton, "Delphi: a technique to harness expert opinion for critical decision-making tasks in education," *Educational Psychology*, vol. 17, no. 4, pp. 373–386, 1997.

[18] G. L. Herman, "The development of a digital logic concept inventory," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2011.

[19] M. J. Scott, M. G. Allen, and D. Gaur, "Challenges in Using a Delphi Method to Formalize Conceptual Understanding in Functional Reasoning," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2017.

[20] N. Dalkey and O. Helmer, "An experimental application of the Delphi method to the use of experts," *Management science*, vol. 9, no. 3, pp. 458–467, 1963.

[21] 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," *Trans. Comput. Educ.*, vol. 10, no. 2, pp. 5:1–5:29, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1789934.1789935