What Do "CS1" and "CS2" Mean? Investigating Differences In the Early Courses

Matthew Hertz
Computer Science Department
Canisius College
Buffalo, NY 14208
hertzm@canisius.edu

ABSTRACT

Thirty-one years ago, the ACM Computing Curricula used the terms "CS1" and "CS2" to designate the first two two courses in the introductory sequence of a computer science major. While computer science education has greatly changed since that time, we still refer to introduction to programming courses as CS1 and basic data structures courses as CS2. This common shorthand is then used to enable students to transfer between institutions and as a base of many research studies.

In this paper we show that while there is wide agreement on the connotation of CS1 and CS2, there is little agreement as to the denotation of these terms. Surveying CS1 and CS2 instructors, we find little agreement on how important various topics are to each of these course and less agreement on how well students master the material. Even after limiting the analysis to whether a topic has ANY important or students complete a course with ANY mastery of the material, we continue to find significant disagreements between instructors.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computers and Information Science Education—*Computer science education; Curriculum*

General Terms

Standardization

Keywords

CS1, CS2, Survey, Curriculum Design

1. INTRODUCTION

The introductory sequence of Computer Science courses includes a first course introducing students to programming fundamentals and an additional one or two courses teaching data abstractions/data structures. Using the terminology originally developed by the ACM's 1978 Computing Curricula [2], the introductory course is commonly referred to as *CS1* and the latter course(s) are called *CS2*. While earlier Computing Curriculum had been developed [6, 1] and changes continue to be made to the (now joint ACM-IEEE) curriculum [13, 12, 11], these concepts endure. The accepted nature of CS1 and CS2 can further be seen in how these ideas have been incorporated into other model curricula that have been developed [3, 14, 5].

The names and general principles of CS1 and CS2 have continued, but the past 30 years have changed the concepts covered in these courses. Between the Liberal Arts Curricula and the most recent ACM-IEEE curricula there are 3 proposed ways to organize CS1 and CS2 courses. Each of these curricula includes courses organized in an object-first manner. The ACM-IEEE curricula also defines an imperative-first approach to these courses while the liberal arts curricula defines a functional-first model. Many departments deviate from these models focusing more than required on specific topics, introducing new topics, or ignoring topics.

Given all of these potential differences, we must ask whether the terms "CS1" and "CS2" actually convey any real meaning anymore. The answer to this question is very important. Each year at SIGCSE and other computer science education conferences there are many papers examining student performance in CS1 and CS2 (for example, [7, 8, 10]). This research assumes a common understanding of these courses. The validity of cross-institutional studies also rely upon a single definition of CS1 and CS2. Whenever students consider transferring and faculty evaluate the transcripts of transfer students, they rely upon the common agreement of these topics. For all these situations, it is important to verify that these assumptions are valid. If there is no common understanding, it is important that this fact be understood and we always make explicit the information needed to evaluate a course.

To investigate whether there is any underlying agreement on what occurs in CS1 and CS2 courses, we examined a range of topics from the ACM Small College and ACM-IEEE Computing Curricula. Respondents to our survey were asked to rate both the importance and student achievement on each topic in their CS1 and CS2 courses. Using Fleiss' kappa [9], we then determine the level of overall agreement between instructors. We further examine the level of agreement of whether instructors felt topics had any importance and students left their course with any ability on each topic.

As we show, there is little of the agreement suggested by

SIGCSE'10, March 10–13, 2010, Milwaukee, Wisconsin, USA.

Control constructs	Testing
Methods, functions, subroutines	Debugging
Variables, types, expressions	Recursion
Error handing (e.g., exceptions)	Arrays, strings
I/O (e.g., files, printing to screen)	Sorting
Style (e.g., indentation, variable names)	Trees
Stacks, queues, lists	Graphs
Abstraction mechanisms (e.g., interfaces, abstract classes)	Other data structures
Object-oriented programming (e.g., inheritance, class anatomy)	

Table 1: List of Topics Used in the Survey

use of the terms CS1 and CS2. While some general agreement exists on the most basic of topics, beyond this we found little to suggest any common definitions of CS1 and CS2. Even at the coarsest of granularities, there is little agreement on which topics are taught in each of these courses. So while we often discuss CS1 and CS2, these courses are taught so differently as to make them nearly meaningless.

Section 2 explains the design of the survey and the steps taken to minimize survey error. Section 3 presents our results and Section 4 discusses what the results mean. Finally, Section 5 presents our conclusions and discusses future work.

2. SURVEY DESIGN

In order to determine what, if any, agreement exists on the content of CS1 and CS2 courses, we developed a survey. Our first step in developing this survey was to create a list of topics by which we could evaluate the courses. We examined existing curricula and chose to base our list on the division of topics presented in the LACS Model Curriculum for a Liberal Arts Degree. This outline is itself based upon the ACM-IEEE Model Curriculum, but is more detailed in specifying each topic. This topic listing is still general enough for to be mapped to courses taught using either object-oriented or functional languages. We therefore thought this made it ideal to analyze the range of introductory courses taught under aegis of CS1 and CS2. Starting from this list, we then made several small changes to insure the accuracy of our results. The list of topics we used in our survey can be found in Table 1 and the changes made can be found in Table 2. Our changes largely fit into 4 categories. The first set of changes was splitting possibly unrelated topics. These changes help insure we collect results at the finest grain possible. We also tried to generalize topics to best match all the programming languages that could be used. These resulted in our second set of changes. Our third set of changes combine closely related topics. These changes keep our survey to a reasonable reasonable size. Finally, we added a topic that we felt was frequently taught, but not included in any curricula.

Given the final list of topics, we next needed to determine how to best collect data that would enable us to see if there was agreement on these topics. We were interested in whether a universal definition of CS1 and CS2 exists. This focuses more on peoples' perception of these courses than on objective standards. We therefore used a subjective scale for instructors to rate their own courses. Instructors marked how important each topic is for success in their course from 1 ("unrelated") to 5 ("critical"). Instructors also rated how capable students were on each topic at the completion of their course from 1 ("none") to 5 ("mastered"). While these results are subjective (instructors may differ on what it means for a topic to be "important" rather than "critical" or for students to be "knowledgeable" on a topic rather than "demonstrate mastery"), we believe this to be an important consideration.

To gather respondents, we used a targeted snowball sample [4]. Initial invitations were repeatedly posted to the SIGCSE e-mail discussion list. Within this invitation was a request that the invitation be forwarded to other related mailing lists or forwarded directly to CS1 or CS2 instructors. This led to 143 respondents to the survey. (This and all future numbers treat a single respondent who answered questions about both the CS1 and CS2 as if they were two separate responses. As we are analyzing these two data sets separately, we feel this is less likely to cause confusion). Incomplete surveys were then pruned. After pruning we had 58 completed surveys about CS1 and 41 completed surveys on CS2.

3. RESULTS

It was clear, even before the survey response period was complete, it would be difficult to find any common definition of CS1 or CS2. Several respondents either added comments at the end of the survey or e-mailed us directly to discuss their difficulty responding to their survey. Most of these comments questioned our combining arrays and strings in a single topic. While agreeing that strings are important, the respondents either teach in languages that do not include arrays, such as Python, or utilize a heavily object-oriented style which prefers lists. Respondents suggested that this grouping showed a historical bias that was no longer reasonable. While these comments were too late for us to change our survey to capture these differences, they clearly point to an important deficiency in how even the most recent curricula describe what is learned in CS1 and CS2.

The responses we received about mixing arrays and strings provided a clear showing of how wide a range of material is subsumed under the CS1 and CS2 monikers. The comments left by respondents to the questions about CS2 more clearly showed the diversity of meanings of "CS2". Using the common [12, 5] and historical [1] definitions, we referred to the course as *CS2/Data Structures* throughout our survey. Yet, 5 out of 64 individuals (7.8%) added comments stating that CS2 was not their primary data structure course. At their institutions, CS2 now means "the course after CS1" and is stripped of any other connotation.

Given these two early findings, we also wanted to see if there was any agreement on either what topics were important in CS1 and CS2 courses or the capability of students at the end of these courses. To quantify this level of agreement, we evaluated our data using Fleiss' kappa [9]. Fleiss' kappa

Topic in LACS Curriculum	Topic in Survey
Split Topics	
Tracing, testing, debugging	Testing Debugging
Arrays, lists, strings	Arrays, strings
Searching, sorting	Sorting
Generalized Topics	
Streams, files	I/O (e.g., files, printing to screen)
Interfaces	Abstraction mechnisms (e.g., interfaces, abstract classes)
Exceptions	Error handling (e.g., exceptions)
Advanced structures	Other data structures
Combined Topics	
Anatomy of a class Inheritance	Object-oriented programming (e.g., inheritance, class anatomy)
Stacks, queues	Stacks, queues, lists
Added Topics	
	Style (e.g., indentation, variable names)

Table 2: Starting from the topics for CS1 and CS2 in the LACS Model Curriculum, we then made these changes to create our final list. The changes here are grouped by the reason for making the change.

evaluates the consistency of judges using categorical ratings and returns a score from 0 to 1. A score of 0 on Fleiss' kappa denotes complete disagreement amongst the judges; a score of 1 arises when the judges are in complete agreement. While there is no agreed upon standard for how to interpret these results, one guideline uses a linear binning of the results [15]. Each topic is evaluated independently using this metric, but we are interested in examining all of the results for each course. We therefore plotted these results using a radar plot. The results from CS1 can be seen in Figure 1(a) and the plot of CS2 results are found in Figure 2(a).

Fleiss' kappa is defined only for nominal data and treats all disagreements equally. In reality, the difference between whether a topic is "important" or "critical" is much less significant than the difference between a topic being "beneficial" and "unrelated". This could result in Figures 1(a) and 2(a) providing an overly dismal view of the level of agreement that actually exists. We therefore generated a second set of results to examine if there was at least agreement on the topics covered in each course. To perform this measurement, we recoded responses as 0 if a topic was unrelated to their course or students were incapable with the material and recoded the response as a 1 for all other responses. Thus encoded, Fleiss' kappa measures the agreement that the topic belongs in the course at all or that students end the course with any ability to perform this work. The results of this second analysis can be seen in Figure 1(b) for CS1 and Figure 2(b) for CS2.

4. ANALYSIS

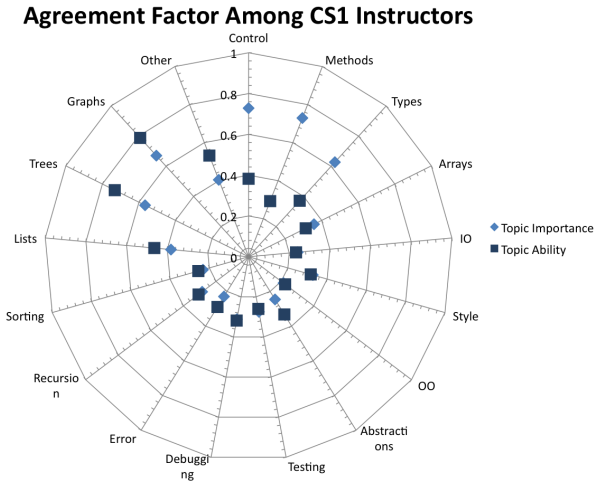
The results seen in Figures 1 and 2 are very telling, especially when combined with the qualitative responses from the previous section. As we now discuss, there would appear to be little agreement as to what it means to be a CS1 or CS2 course. While there is more agreement as to which topics are important for CS2 students, this agreement comes with the significant caveat that few of the agreed upon topics include any data structure. We will now discuss the results from each course in turn.

Figure 1(a) shows that there is very little agreement as to either importance of topics in CS1 or students' ability after CS1. While there was some agreement on how important some of the most basic underpinning of programming (control constructs; methods, functions, subroutines; variables, type, expressions) were to respondents CS1 courses, that was about all that could be agreed upon. The next 4 topics with the highest levels of agreement are ones for which 57% to 78% of respondents stated that the topics had nothing to do with their courses. When examining students' abilities after CS1, there was only agreement on those topics which a majority of respondents never cover. The next highest kappa was only a 0.38 for control constructs. This low-level of agreement suggests that there is little common skill levels among students across different institutions.

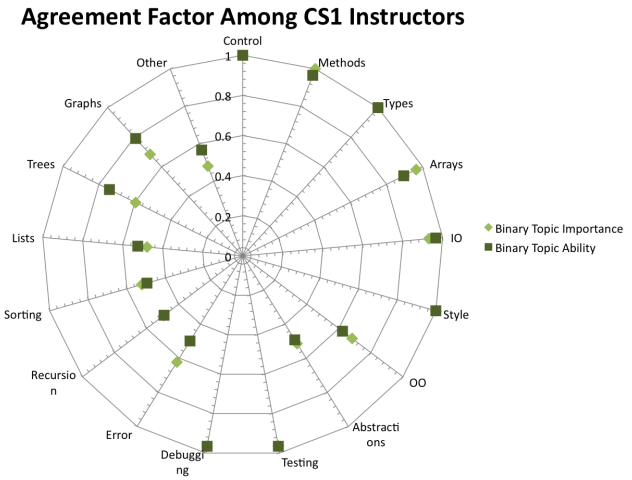
We continue to see very little agreement when we limited our analysis to look at if topics were at all related to their course or students passing this course had any ability with the material. As Figure 1(b) shows, most agree that basic programming concepts are part of their CS1 course and that student passing CS1 have some ability in them. But there are still very significant disagreements on about half of the possible topics. These results show that while we can expect all students coming out of CS1 to have some knowledge of basic programming concepts, exactly which topics and the level of this expertise is going to vary greatly.

Figure 2(a) shows that CS2 has even less agreement than we found in CS1. While there continued to be some agreement on the four most basic topics, the next highest kappa we measured was a 0.46 for object-oriented programming concepts. These results show that there is less agreement on both the importance of these topics and on students ability after passing the course. Most of this decrease is explained by the increasing rate respondents reporting that a topic is at least partially related and that students complete their course with at least some ability. This effect can better be witnessed by the greatly increased agreement in Figure 2(b). Even when limiting our analysis to this binary analysis, we still find strong disagreements on several important data structures such as trees, graphs, and other data structures. These kappas all stay below 0.82 and several remain below 0.5.

These results are not due to the range of languages used



(a) Graph showing the level of agreement among CS1 instructors.



(b) Graph showing how much CS1 instructors agree that a topic has any importance or students had any ability in a topic.

Figure 1: Graph showing the level of agreement among CS1 instructors as to the importance of each topic in their CS1 course and how capable students were at the completion of that course. The level of agreement was measured using Fleiss' kappa. A result of 1 denotes perfect agreement and a result of 0 denotes no agreement.

in these courses. When we examined the results for object-oriented, procedural, and functional languages separately, our results were identical. This finding was true for every analysis: both CS1 and CS2 and both the importance and student abilities. Rather than being a result of the language used, these differences arise from the wide variety of styles and methodologies used by the respondents.

There are a number of important conclusions that one can draw from these results. While it appears that CS1 continues to be synonymous with an "Introduction to Programming" course, CS2 is no longer used strictly as another name for a course on data structures. Further, both CS1 and CS2 have been stretched to include a much wider range of activities than when originally proposed. As a result of these changes, there is little agreement on what topics are important to these courses and whether students passing these courses will have any knowledge of this material.

Continuing to use the CS1/CS2 terminology could cause more harm than good in the long run. These terms gives students a false sense that the skills learned in CS1 at one institution will automatically prepare them for CS2 at another school. Similarly, faculty need to know more than whether a class at another institution was called CS2 to determine if it could be used to replace a data structures course they offer. These results also suggest that we, as education researchers, must do more to place our results in context. To understand how others' research would apply at our own institutions, or to explain how our own research would apply to others, we must understand what the content of the course really is – just saying it is a CS1 or CS2 course provides very little explanation. By providing this extra information, we can make explicit the pedagogical assumptions that underlie the research and enable everyone to better understand the results.

5. CONCLUSION

While the community continues to use the terms CS1 and CS2, the reality has moved away from these terms having any meaning. In our survey of educators, we found that there is little agreement about how important a range of topics are to these courses. Moreover, we show that there exists only very weak agreement as to whether these topics even apply to each course. Sadly, there is even less agreement as how capable students are with these concepts after passing each courses. We think that these disagreements are part of, and vital to, the continued evolution of how we teach computer science. Rather than try and pigeonhole courses into terms we, as a community, have outgrown we instead propose that in research, course descriptions, and other important works, we provide more complete descriptions of the course and provide the context needed for others to understand what it includes.

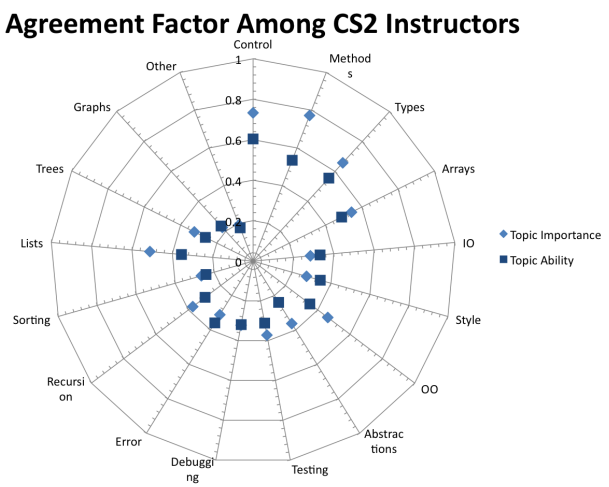
Acknowledgements

The authors would like to thank Sarah Ford, R. Mark Meyer, Paul Gestwicki, the survey respondents, and the anonymous reviewers for their help in improving this paper.

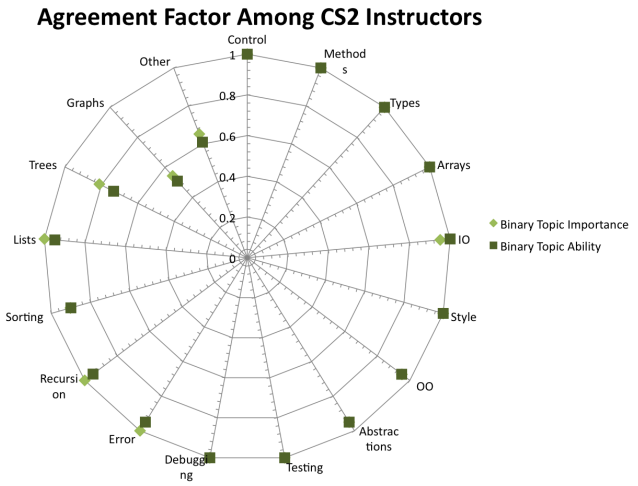
6. REFERENCES

[1] W. F. Atchison, S. D. Conte, J. W. Hamblen, T. E. Hull, T. A. Keenan, W. B. Kehl, E. J. McCluskey, S. O. Navarro, W. C. Rheinboldt, E. J. Schweppe, W. Viavant, and D. M. Young, Jr. Curriculum 68: Recommendations for academic programs in computer science: a report of the acm curriculum committee on computer science. *Commun. ACM*, 11(3):151–197, 1968.

[2] R. H. Austing, B. H. Barnes, D. T. Bonnette, G. L. Engel, and G. Stokes. Curriculum '78: recommendations for the undergraduate program in



(a) Graph showing the level of agreement among CS2 instructors.



(b) Graph showing how much CS2 instructors agree that a topic has any importance or students had any ability in a topic.

Figure 2: Graph showing the level of agreement among CS2 instructors as to the importance of each topic in their CS2 course and how capable students were at the completion of that course. The level of agreement was measured using Fleiss' kappa. A result of 1 denotes perfect agreement and a result of 0 denotes no agreement.

computer science— a report of the acm curriculum committee on computer science. *Commun. ACM*, 22(3):147–166, 1979.

[3] J. Beidler, R. H. Austing, and L. N. Cassel. Computing programs in small colleges. *Commun. ACM*, 28(6):605–611, 1985.

[4] *Implementation and Analysis of Respondent Driven Sampling*, New York, NY, Nov. 2006. Springer New York.

[5] L. A. C. S. Consortium. A 2007 model curriculum for a liberal arts degree in computer science. *J. Educ. Resour. Comput.*, 7(2):2, 2007.

[6] S. D. Conte, J. W. Hamblen, W. B. Kehl, S. O. Navarro, W. C. Rheinboldt, D. M. Young, Jr., and W. F. Atchinson. An undergraduate program in computer science—preliminary recommendations. *Commun. ACM*, 8(9):543–552, 1965.

[7] J. Gal-Ezer, T. Vilner, and E. Zur. Has the paradigm shift in cs1 a harmful effect on data structures courses: a case study. In *SIGCSE '09: Proceedings of the 40th ACM technical symposium on Computer science education*, pages 126–130, New York, NY, USA, 2009. ACM.

[8] E. Howe, M. Thornton, and B. W. Weide. Components-first approaches to cs1/cs2: principles and practice. In *SIGCSE '04: Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 291–295, New York, NY, USA, 2004. ACM.

[9] J. J. Randolph. Free-marginal multirater kappa: An alternative to fleiss' fixed-marginal multirater kappa. In *Joensuu University Learning and Instruction Symposium*, Joensuu, Finland, 2005. Available at: `http://www.eric.ed.gov/contentdelivery/servlet/ERICServlet?accno=ED490661`.

[10] H. Roumani. Practice what you preach: full separation of concerns in cs1/cs2. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 491–494, New York, NY, USA, 2006. ACM.

[11] SIGPLAN programming language curriculum workshop. *SIGPLAN Notices*, 43(11), 2008.

[12] C. The Joint Task Force on Computing Curricula. Computing curricula 2001. *J. Educ. Resour. Comput.*, page 1, 2001.

[13] A. B. Tucker. Computing curricula 1991. *Commun. ACM*, 34(6):68–84, 1991.

[14] H. M. Walker and G. M. Schneider. A revised model curriculum for a liberal arts degree in computer science. *Commun. ACM*, 39(12):85–95, 1996.

[15] Fleiss' kappa. In *Wikipedia*, Retrieved 4 Sept. 2009. Available at: `http://en.wikipedia.org/wiki/Fleiss'_kappa`.