CLOSING THE CS I - CS II GAP: A BREADTH-SECOND APPROACH

Pearl Brazier[1], Laura Grabowski[2], and Gustavo Dietrich[3]

Abstract *- Many students experience difficulty making the transition from a traditional CS I course that consists primarily of learning to program in a high-level language to the higher level of abstraction required in the CS II course. Students also come to the discipline with a lack of understanding of the scope of computer science. A Foundations of Computer Science course as the second course in introductory sequence of three 3-hour courses that essentially covers the material proposed by the Breadth-first approach from Computing Curriculum 2001, incorporates additional programming experience to enhance the skills developed in CS I, introduces the discrete mathematics needed early in the curriculum, and introduces students to social and ethical issues addresses these concerns. The more traditional Programming-first approach to CS I is retained as the first course. The paper will present the curriculum in our Foundations course and review the problems and success we have had with this model.*

Index Terms Breadth-First, Breadth-Second, CS I, CS II, Foundations of Computer Science, Pedagogy.

INTRODUCTION

Discussions concerning the best approach to introducing students to computer science continue. Curriculum 2001[5] presents several models for an introductory two-course sequence that essentially break down into the traditional Programming-first approach and the Breadth-first approach proposed in Curriculum '91. Since Curriculum '91 discussions of Breadth-first [3] [12] [8] [14] [17] versus Programming First have abounded with texts developed to support one and two semester breadth-first approaches [1] [7] [14] [15]. Courses have been designed to integrate Breadth-first topics into the Introductory course [4] [13] [18] Breadth-also [8] [9] [11] courses have been proposed that integrate the programming and breadth topics throughout a two or three semester core of courses. While support for the concept of the breath-first introduction to the discipline remains strong with the computer science education community, very few programs using that approach have been implemented or survived [5]. Various forces have worked against it. Programming is a significant underpinning of the discipline and those tools must be developed early. The computer is an excellent tool to demonstrate the actualization of algorithms so attempts to teach algorithms independent of their implementation is hard to sell. The advanced placement program is programming first based so placing these brightest and best students into the breadth-first curriculum was difficult. Students are motivated by the hands-on computer experiences. Other disciplines depending on the CS I course to serve their programming needs, found a breadth-first course did not meet their needs. The amount of material to support a breadth-first, traditional CS I programming concepts, and the traditional CS II introduction to data structures and algorithms was prohibitive, a fact pointed out in Curriculum 2001 [5]

While successfully completing a traditional CS I course, many students find the transition to abstraction required in the CS II course a difficult one. This difficulty remains independent of attempts to introduce abstraction earlier such as functions first, objects first, and changes in programming languages. In 1998 we introduced our Foundations of Computer Science as a second course, as a prerequisite to CS II, keeping the traditional programming course then taught in Pascal as the first course. This course is used to bridge the gap between CS I and CS II and at the same time give a "breadth-second" introduction to the discipline.

CONTEXT

Our university is a regional state-supported institution of 14,000 students that requires completion of a liberal arts core in addition to requirements for a major field of study. Where possible we have used supporting courses from other disciplines to support the computer science curriculum. The Freshman-Sophomore core of 12 semester hours of computer science courses consists of CSCI 1380 CS I a traditional introduction to computer science in C++, CSCI 1381 Foundations of Computer Science a breadth approach to the discipline, and CSCI 2380 CS II, a traditional introduction to data structures in C++. We also require CSCI 2333 a sophomore level Computer Organization and Assembly Language Programming course, which students can take in parallel with CS II. Our three-hour CS I course serves as an introductory course for Computer Science as well as a service course for engineering mathematics and science students. We have retained the traditional programming-first approach to this course, in order to support the needs of other disciplines, to facilitate the students who change their majors and to accept advanced placement credit from high school students. The option of expanding the course sequence to four-hour courses so that breadth-first topics could be integrated was not feasible due

[1] Pearl Brazier, University of Texas – Pan American, Department of Computer Science, Edinburg, TX 78539, brazier@panam.edu
[2] Laura Grabowski, University of Texas – Pan American, Department of Computer Science, Edinburg, TX 78539, lgrabowski@cs.panam.edu
[3] Gustavo Dietrich, University of Texas – Pan American, Department of Computer Science, Edinburg, TX 78539, dietrichg@panam.edu

0-7803-7961-6/03/$17.00 © 2003 IEEE November 5-8, 2003, Boulder, CO
33rd ASEE/IEEE Frontiers in Education Conference

to scheduling issues and degree requirements of the departments using this course.

For historical reasons and advanced level degree requirements, a junior level discrete mathematics course taught by the mathematics department is required and hence we have not implemented a discrete mathematics course at the freshman level as many programs have done. A professional ethics course taught by the Philosophy department is also required and satisfies a university core requirement. This course partially addresses the societal and professional issues that are commended by ACM Curricula and ABET/CAC accreditation guidelines. We have observed as others that many students come to the discipline with the perception that computer science is only about programming and we felt that an early introduction to a broader view of the field would be beneficial. It is within this context and the need to fill in some gaps as well as to bridge the gap between the CS I and CS II that led to the design of our Foundations of Computer Science course.

CURRICULUM OBJECTIVES

Our specific objectives for the Foundations of Computer Science course are:

- Present a broad overview of the computing field
- Introduce societal/ethical/legal issues related to computing
- Provide a bridge from the programming skills and level of abstraction obtained in CS I to that required in CS II.
- Provide the discrete mathematics required for the freshman-sophomore courses.
- Provide programming exercises that illuminate the breadth of the discipline while also taking advantage of the presentation of abstraction within the context of the text topics.
- Further develop written communication skills.

CONTENT

We have used *Computer Science an overview* by Brookshear[2] as a text and have found it to support our objectives almost entirely. It has a good overview of the field, has excellent problem sets at the end of each chapter addressing the concepts of the chapter, and includes a good set of societal and ethical thought questions for each chapter relating these issues to the concept area just covered. We added DeMorgan's Laws to the logic section to expand the basic logic coverage and we designed our own programming assignments to tie in with certain chapters.

Specific content and assignments are listed as follows:

1. Discrete mathematics
    a. basic logic
    b. digital logic and digital systems
    c. number systems and representation
2. Survey of computer science
3. Machine architecture
4. Software
    a. Operating Systems and Networks
    b. Algorithms
    c. Programming Languages
    d. Software Engineering
    e. Database
5. Artificial Intelligence
6. Theory of Computation
7. Social and ethical implication of computing
    a. history
    b. social context
    c. professional and ethical responsibility
    d. intellectual property
    e. privacy
8. Homework on topics
9. Homework on social and ethical issues
10. Programming assignments
    a. review of control structures
    b. simple recursion versus looping
    c. array of structures

To show how we tied the programming to the topics in the course, we assigned the number conversion from binary to base 10 problem as a review of control structures while presenting the algorithm chapter. The greatest common devisor is a good alternative to the factorial problem to compare a recursive solution to an interative solution. Arrays of records or structures assignment can be given in conjunction with the database section.

RESULTS

Assessment methods

The departmental curriculum committee meets several times annually to discuss the introductory curriculum. Students submit anonymous evaluations of the instructor for each course at the end of each semester in which multiple choice opinions of the instructor are given, as well as written opinions on texts, assignments, exams, successful completion of objectives, and general opinions of the course and suggestions for improvements. These opinion surveys were inspected to gain the short term perception of the course. In order to access the students perception of the course down-stream, a survey gathering opinions about the Foundation of Computer Science course was conducted with students that are enrolled in Junior-Senior level computer science courses. We also tracked students who took the course from Summer 1999 through Fall 2002 to analyze their success in CSCI 1381 Foundations of Computer Science and CSCI 2380 CS II noting their success in each course.

Students Reaction to the Course

Anecdotal data from instructors and discussions during advising sessions indicate that the course is meeting the objective of giving a broad view of the field. For the most part student attitudes to the value of the course have changed as a result of taking the course. CS II instructors note a familiarity with the abstract data types and students seem better prepared to complete the programming implementation.

It appears from comments given on the course opinion surveys issued at the end of the semester that students for the most part indicate the course has met the objectives in presenting the breadth first view of computer science, however a small number still indicate they found it boring and not relevant, indicating they would have preferred a programming course. Students indicated the homework assignments were good, but some felt there were too many and they were difficult. Some commented negatively about the required written discussions of the societal and ethical issues. One instructor has recently incorporated required online discussion of the societal and ethical issues assignments and has reported much greater success with the extent and quality of answers and the attitude of students towards this aspect of the homework assignments. The actual use of online WebCT in support of the course is in itself contributing to the overall objective of the course.

The surveyed Juniors and Seniors who had taken the Foundations course reported the course gave them a better understanding of the computer science field, helped their programming skills, and overwhelmingly gave them an understanding of the societal and ethical aspects associated with computing. A summary of the analysis follows.

Results of assessments

For comparison and background, of the 1121 students who enrolled in the prerequisite CSCI 1380 over 5 semesters between S99-F02, 55% finished the course with grades of A, B, C, or D, with 46% completing the course with C or better.

Analysis of the 1381 survey given to 53 students enrolled in junior-senior computer science courses shows that downstream, students had quite a positive response as to the benefits of the course relative to our course objectives.

We asked how many semesters ago the students had taken 1381 and if they had also taken 2380. We also asked them to rate the following questions as Not at all, Somewhat, More than somewhat, A Great deal:

1. Did 1381 help your programming skills? Comment.
2. Do you now feel you have a better view of the computer science field as a result of taking the course than you did before taking the course? Comment.
3. To what extent do you feel 1381 gave you an understanding of the societal, ethical, and legal aspects associated with computing? Comment.
4. What topics in 2380 were first introduced in 1381?

The results are displayed in Figure1 through Figure 3. The strongest result we attained was meeting our objective of introducing the student to societal and ethical issues [Figure 3] with 58% responding a great deal or more than somewhat and 92% responding at least somewhat to a great deal. We also are achieving our objective of giving the students a broader perspective of the computer science [Figure 2] with 49% responding great deal or more than somewhat and 94% responding at least somewhat to a great deal. Our goal of enhancing the programming skills obtained in 1380 before attempting 2380 [Figure 3] was less successful with 30% responding a great deal or more than somewhat and 92% responding at least somewhat to a great deal. However, the first three semesters the course was taught only one or two programming assignments were required. This number was increased to three or four more advanced problems and included recursion, arrays and structures. The juniors and senior students surveyed confirmed this in their answers. Not surprisingly, the student's specific recollection as to what topics they were exposed to in 1381 that were subsequently covered in 2380 decreased as the time from when they had completed the courses increased.

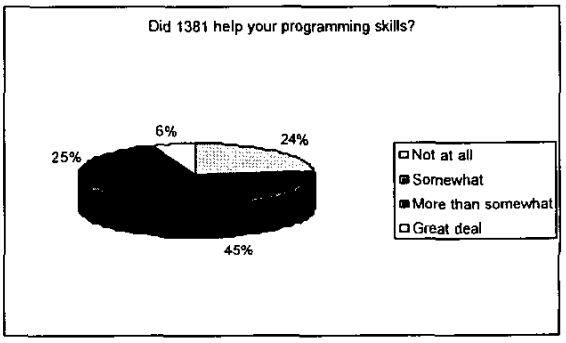

FIGURE 1
SURVEY OF JUNIOR-SENIOR OPINION OF CSCI 1381 QUESTION 1

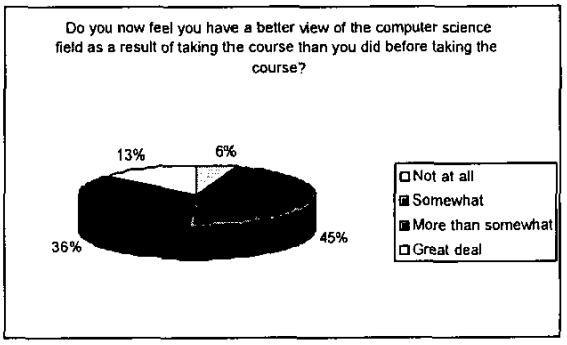

FIGURE 2
SURVEY OF JUNIOR-SENIOR OPINION OF CSCI 1381 QUESTION 2

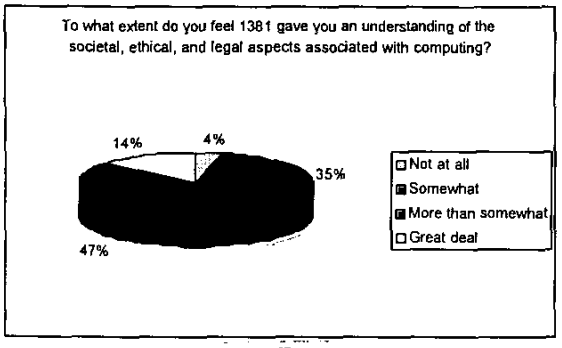

FIGURE 3
SURVEY OF JUNIOR-SENIOR OPINION OF CSCI 1381 QUESTION 3

In order to analyze if 1381 was contributing to successfully completing 2380, we tracked students who were enrolled in both courses from Summer of 1999 through Fall of 2002 [Table 1]. We found that having completed 1381 was only somewhat helpful in completing 2380. 56% of those who successfully completed 1381, then successfully completed 2380. However the number of programming assignments that were required for the course increased from 1 to 3 or 4 after the first several semesters of offering the course. As noted previously, this was verified by the responses from the analysis of the survey of juniors and seniors. Some students either did not take 1381 or took it after 2380. We did not include these students in reaching our conclusion. We are now paying closer attention to the prerequisites for 2380.

TABLE 1
ANALYSIS OF 1381-2380 SEQUENCE

Course	Description	Number	Percent
1381	Semesters analyzed: F99-F02		
	Students who took course	306	
	Students with passing grades A,B,C)	194	63%
2380	Semesters analyzed: SU99-F02		
	Students who took course	277	
	Students with passing grades A,B,C)	169	61%
1381-2380	Students who took 1381 then 2380	78	67%
	Students who took 1381 and 2380	17	15%
	Students who took 2380 then 1381	22	19%
Passed 2380	Students who did not take 1381	52	31%
	Students who benefited from 1381	95	56%

OTHER ADVANTAGES

Students who come with a CS I course that was taught in different language due to transferring from other institutions or taking advanced placement courses in high school can use the programming that is part of this course to transition to the language used in CS II, such as going from Java to C++ or C++ to Java, without too much difficulty. We have had several students do this.

A separate course with the primary objective of giving the students a breadth-first view of the field gives a more focused emphasis to the objective, which can get lost or not given the proper attention in a model in which the breath is integrated into the entire introductory sequence.

PROBLEMS

We have noticed two significant problems:

Students need a lot of motivation to take a course with little or no programming. Students have ignored the prerequisites and delay taking the course until later, thus defeating our objectives. We have begun to pay closer attention to this. Significant efforts must be made at the beginning of the course and during advisement sessions to explain the objectives of the course. Results of opinion surveys at the end of the course indicate for the most part attitudes toward the course have been changed and are positive. A recently conducted Alumni survey and Senior Exit Interview Survey have further supported the effectiveness of the course.

For most students including this course adds an additional semester to the total sequence. This is a problem for students who decide to pursue the computer science major after several semesters of college work. This is the case for a significant number of our students in part due to the demographics of our region with many first generation college students from low economic status who take some time to decide what they are interested in and with community college transfers. But because of these factors, many students currently do not complete their degree in four years, so this is part of a more general problem. For a student who comes prepared to start the major, this problem is solved by doubling up in CS courses by taking Assembly Language simultaneously with CS II or doubling up in the junior level computer science core. Having had the broad perspective only enhances their ability to do so.

CONCLUSION

Inserting a Breadth-Second course to the Introductory computer science sequence is a viable alternative to the breadth-first approach or the traditional programming first approach or the integrated breadth-throughout approach. It addresses the issues that have influenced many programs to stay with the traditional approach, but introduces the breadth that is desired early in a student's career. It also supports the transition from CS I and CS II and adds room in the sequence to include other topics such as discrete mathematics that is needed as background early in the students experience. Including online discussion of the societal and ethical issues further enhances the written communication skills of students and broadens the understanding and discussion of these issues.

REFERENCES

[1] Aho, A. V., & Ullman, J. D. *Foundations of Computer Science.* New York: W. H. Freeman and Company, 1992.

[2] Brookshear, J. G, *Computer Science an overview,* Addison-Wesley, 2003.

[3] Bagert, D, Marcy, W. M, & Calloni, B, "A successful five-year experiment with a breadth-first introductory course". *26th SIGSCE Symposium on Computer Science Education,* New York: ACM Press, , 1995, 116-120.

[4] Close, R., Kopec, D, & Aman, J, "CS1: Perspectives on programming languages and the breadth-first approach." *Journal of the Consortium for Computing in Small Colleges, 15*(5), 2000, 231-237.

[5] Joint Task Force on Computing Curricula (2001). Computing curricula 2001. *ACM Journal of Educational Resources in Computing,* 1(3), New York: ACM Press.

[6] Long, T, J, Weide, B, W, Bucci, P, Gibson, D, S, Hollingsworth, J, Sitaraman, M, & Edwards, S,"Providing intellectual focus to CS1/CS2", *29th SIGSCE Symposium on Computer Science Education* New York: ACM Press, 1998, 252-256.

[7] Nagin, P., & Impagliazzo, J. *Computer science: A breadth-first approach with Pascal,* New York: John Wiley & Sons, 1995.

[8] Paxton, J, Ross, R, J, & Denbigh, S, "An integrated, breadth-first computer science curriculum based on *Computing Curricula 1991" 24th SIGSCE Symposium on Computer Science Education,* New York: ACM Press, 1993, 68-72.

[9] Paxton, J, Ross, R, J, & Denbigh S, J, "A methodology for teaching an integrated computer science curriculum. *25th SIGSCE Symposium on Computer Science Education,* New York: ACM Press, 1994, 1-5.

[10] Phillips, A, T, Stevenson, D, E, & Wick, M, R, "Implementing CC2001: A Breadth-first introductory course for a just-in-time curriculum design",. *34th SIGSCE Symposium on Computer Science Education,* New York: ACM Press, .2003, 238-242.

[11] Powers, K, D, "Breadth-also: A rationale and implementation", *34th SIGSCE Symposium on Computer Science Education,* New York: ACM Press, 2003, 243-247.

[12] Shackelford, R, L, & LeBlanc, R, J, " Integrating 'depth first' and 'breadth first' models of computing curricula", *25th SIGSCE Symposium on Computer Science Education* New York: ACM Press, 1994, 6-10.

[13] Shannon, C, "Another breadth-first approach to CS1 using Python", *34th SIGSCE Symposium on Computer Science Education,* New York: ACM Press, 2003, 248-251.

[14] Tucker, A, B, & Wegner, P, "New directions in the introductory computer science course", *25th SIGSCE Symposium on Computer Science Education,* New York: ACM Press, 1994, 11-15.

[15] Tucker, A, B, Bernat, A, P, Bradley, W, J, Cupper, R, D, & Scragg, G, W, *Fundamentals of computing II: Abstraction, data structures, and large software systems,* New York: McGraw-Hill, 1993.

[16] Tucker, A, B, Bernat, A, P, Bradley, W, J, Cupper, R, D, & Scragg, G, W, *Fundamentals of computing I: Logic, problem solving, and computers, C++ edition,* New York: McGraw-Hill, 1995.

[17] Vandenberg, S, & Wollowski, M, " Introducing computer science using a breadth-first approach and functional programming", *31st SIGSCE Symposium on Computer Science Education,* New York: ACM Press, 2000, 180-184.

[18] Wilson, R, "Integrating a breadth-first curriculum with relevant programming projects in CS1/CS2", *26th SIGSCE Symposium on Computer Science Education,* New York: ACM Press, 1995, 214-217.