Impact on Retention from a Change In Undergraduate Computing Curricula

Donna S. Reese[1], *T.J. Jankun-Kelly*[2], *Lisa Henderson*[3], *Sarah Lee*[4]

Abstract – Mississippi State University (MSU), like many other institutions across the country, has seen a significant decline in the number of computing majors since the early 2000's when the dot com crash caused many students to shy away from majors involving computing. In addition, the diversity of the students who have remained in the field has decreased, particularly with female students making up a smaller and smaller percentage of majors in these fields. In the 2008-09 year a significant effort was made to re-design the introductory programming sequence in the Computer Science and Engineering Department. This introductory programming sequence is taken by students in computer science, software engineering, computer engineering and electrical engineering. In addition, an introductory CSE course was added for computer science and software engineering majors. Data shows that this redesign has had a positive impact on the retention of students within the majors served by this sequence.

Keywords: Retention, computing

Background Information and Rationale for the Change

Computer science departments across the country are faced with many students who begin in their majors and ultimately transfer out to other departments. For those whose interest or ability does not match with the computing curriculum, this is expected. However, a large number leave the department due to a misunderstanding about the field of computer science.

A study at Georgia Tech found that when students were asked to define computer science and computing, there were significant differences in responses from students who remained in the major and those that left. Those that graduated with a degree in computer science defined the field as one where computing helped solve real world problems. The students that did not remain in the major described computer science as "the study of computer software" and "learning how to manipulate code". Many of those that left the major also described the computing career paths in a negative light. Comments included "Sitting at the computer with minimal human interaction" and "Boring. Coding/debugging code in front of a computer screen all day." [2] These perceptions are echoed in other studies, including research at Point Loma Nazarene University. This study concluded that many students do not consider computer science as a major due to a lack of or inaccurate information about the major and the types of careers available. [3]

The Computer Science & Engineering Department at Mississippi State was faced with these same attrition issues in the mid 2000's, particularly with students from underrepresented groups. [5] After much discussion with the faculty and its advisory board and a review of the curriculum, it was determined that the majority of loss of students occurred in their first three semesters, semesters dedicated to our introductory programming sequence (specifically,

[1] Mississippi State University, Department of Computer Science & Engineering, PO Box 9637, MSU, MS 39762, dreese@cse.msstate.edu

[2] Mississippi State University, Department of Computer Science & Engineering, PO Box 9637, MSU, MS 39762, tjk@acm.org

[3] Mississippi State University, Department of Computer Science & Engineering, PO Box 9637, MSU, MS 39762, lisah@cse.msstate.edu

[4] Mississippi State University, Department of Computer Science & Engineering, PO Box 9637, MSU, MS 39762, sblee@cse.msstate.edu

CSE1284 (CS1), CSE1384 (CS2), and CSE2383 (Data Structures)). Thus, a rethinking of those first three courses and a students' introduction to the major was pursued with three goals:

- Increase the retention of CSE majors, many who are lost in this sequence
- Specifically improve the recruitment and retention of underrepresented populations in CSE (not discussed in this paper)
- Facilitate better problem solving, not just rote programming

While the larger goals (the first two) were with retention and recruitment, a review of our introductory courses also indicated dissatisfaction in their approach—a focus on the nuances of the language of choice (C++) instead of the core concepts of problem solving via programming. Comments from current and past students also indicated that our approach to the introductory material was also a cause for loss of interest and students. Thus, more problem solving and less rote programming was seen as a necessary part of our plan to retain students.

The faculty committee in charge of this effort addressed these goals in a two part approach: First, steps were taken to better communicate that CSE was "more than programming" via real-world, exciting examples. Primarily, this was done via an introduction of a CS0 course that overviewed the field with special focus on the department's research areas. Part-and-parcel of this communication was integral to the second step: Demonstrating via our introductory sequence that CSE is about problem solving via computers and not programming to do problem solving. This required rethinking our programming assignments (e.g., use engaging examples such as Google-games, cryptography, and bioinformatics) and choosing an appropriate programming language. The chosen language needed to support problem solving but in an environment that is conducive to learning and exploration live during lecture time. At the same time, constraints from other majors taking our sequence limited what changes could be made (specifically, low level memory management was required before the end of CS2). After a year of proposals, our new sequence consists of:

- CSE1002: Introduction to Computer Science. A discussion of the larger role of computer science, software engineering and computer engineering in society and an overview of research areas in the department are covered. This is a new course in the curriculum.
- CSE1284: Introduction to Programming. Taught in Python, the course was designed to be "the only course a non-major would need to learn about problem solving with computers." Core programming concepts are introduced in-class via interactive examples and engaging labs. This was a complete revision of our previous introductory course.
- CSE1384: Intermediate Programming. This course is bi-lingual, using Python to introduce OO and linear data structures, and progressing to a coverage of C++ for low level memory management required by other majors in our courses. This course is a significant revision from the previous course.
- CSE2383: Data Structures. Non-linear data structures are introduced in this course. Python and C++ are used as the language of lectures and examples while the assignments are in C++. This course is a slight revision from the previous version.

Changes to the Introductory Sequence

Brief History

Prior to the change discussed in this paper, students entering computer science, software engineering or computer engineering took a two-semester introductory programming sequence (CSE 1284 & CSE 1384) that required no previous programming experience as a prerequisite. This sequence included two courses that each had three hours of lecture per week plus a three-hour lab session. Pair programming has been used in the laboratory since 2005 in an attempt to help with retention in the introductory sequence. [4] The language for this new introductory programming sequence was C++. Topics covered in the first course included[5]:

[5] In all course outlines given, hours are approximate. There were 3-4 tests given during the semester in addition to lectures.

© American Society for Engineering Education, 2013

- Design (3 hours)
- Creating and running a program (1 hour)
- Input / output (1 hours)
- Memory (1 hour)
- Variables and data types (1 hour)
- Mathematical Expressions (1 hours)
- Relational and logical expressions (1 hour)
- Selection structures (2 hours)
- Pre- and post-test repetition structures (5 hours)
- Functions (5 hours)
- Text files (2 hours)
- Single-dimensional arrays (3 hours)
- Multi-dimensional arrays (2 hours)
- Searching and sorting arrays (2 hours)
- Pointers (but not dynamic memory) (5 hours)
- Strings (including C-style strings) (3 hours)
- Classes (5 hours)

This course was fairly rigorous and covered a good bit of C++. The problem was that there were many students who did not successfully complete this course.

Once students finished this introductory course, they moved into the intermediate programming course (CSE 1384) before going to data structures. This course started with a review of C++ classes and added more advanced topics before going into dynamic memory. Generally it ended with recursion. The topics in CSE 1384 included:

- Algorithm analysis (3 hours)
- Classes (6 hours)
- Static class variables (1 hour)
- Operator overloading (2 hour)
- Friend classes and functions (1 hour)
- Inheritance and polymorphism (3 hours)
- Dynamic memory (arrays) (2 hours)
- Linked lists (5 hours)
- Stacks (3 hours)
- Queues (3 hours)
- Generic programming (template functions and classes) (3 hours)
- Standard Template Library (3 hours)
- Recursion (3 hours)
- Searching and sorting (3 hours)

This class also gave students an even better understanding of C++, classes, and dynamic memory. However, this course also exhibited a relatively high failure rate and some students changed majors even if they were successful in this introductory sequence.

Change for Retention

There were several problems with learning C++ as a first programming language. One problem was that students had a steep learning curve. In order to print a line of text to the screen, the programmer has to include libraries, use a namespace, create a function which returns a value, use an insertion operator, and a stream manipulator. While all

of these elements were discussed, the students did not really understand them until they were covered in depth later on in the semester.

Another problem was dealing with an integrated development environment. The earlier versions that were used (Microsoft Visual Studio version 6 for example) would allow a student to simply type the source code into a file and then compile and run it. The later versions were very frustrating, however. They forced the programmer to set up a project and save the source code as a part of the project. If the project was set up incorrectly, the program would not compile even though the code was correct. Students had enough problems with logic errors and syntax errors; they did not cope well with errors caused by their programming environment.

The previous version of the class also had the disadvantage that most students did not see the programs that they were writing as "real" software. All of their programs were text only. They were used to software that had a window with boxes and buttons and where the user could interact by using a mouse. What they were writing just was not something that they could relate to the software that they used every day.

Changes in CSE 1284

Other educators report using Python as an introductory language to help address some of the issues we were seeing. [1,6]. Python allowed a gentler beginning – nothing had to be taken on faith. If a student wanted to print to the screen, only a print statement was needed. Concepts were added as the students were ready for them. There were two goals: to retain more students and for the students to know C++ by the end of the three semester sequence so that none of the upper level courses had to change languages.

In the introductory course, many topics were easier for students to grasp when using Python -- even though many of these things were taught at a faster pace. Moving faster also allowed for more time to better develop the topics that were covered. The new version of CSE 1284 includes the following topics:

- Design (2 hours)
- Creating and running a program (1 hour)
- Input / output and formatting (3 hour)
- Data types (1 hour)
- Memory (1 hour)
- Mathematical Expressions (2 hours)
- Relational and Logical Expressions (1 hour)
- Selection structures (1 hour)
- Pre-test repetition structures (2 hours)
- Strings (1 hour)
- Functions (6 hours)
- Files (2 hours)
- Lists (4 hours)
- Dictionaries (3 hours)
- Modules (1 hour)
- GUIs (3 hours)
- Exception Handling (2 hours)
- Classes (6 hours)

The new language made the class more fun and exciting for the students, and this is evidenced by improved performance in the class as described in the Assessment Results section below. Some concepts were much easier for the students to grasp in Python. In fact, selection structures and repetition structures were covered very early in the semester. Strings and lists seemed to come naturally to the students, though they did have to get used to the first element having an index of 0, not 1. The methods that were available made some tasks extremely easy. For instance, sorting a list was a simple matter of calling the sort method.

In addition to the concepts covered, Python allowed students to do much more exciting tasks. It was easy for them to access data found on a web page rather than having to download the file and save it in the "right" place so that they could access it. They could access a webpage as easily as a file. In fact, some students found it easier. They could also do fun assignments that they could relate to. For example, one of the lab assignments required students to access Twitter which would have been difficult in the first semester in C++.

Also they were able to create a program that looked like real software – something that they could show to their parents and friends. There were no problems with the integrated development environment. IDLE which comes with Python was easy for the students to use. Students also seemed to have fewer frustrations with the translated language since the program would always run even if it did crash during execution.

The change in CSE 1384

Although we would have preferred this course to continue to use Python throughout, electrical engineering majors take this course as a pre-requisite to one of their required courses with an expectation of becoming proficient with C/C++. Because of this requirement, we implemented the 1384 course to include not only more advance programming concepts but also an introduction to C++ as a part of the course.

The CSE 1384 course starts out in Python covering the following topics:

- Algorithm analysis
- Review of classes
- Python memory model
- Operator overloading
- Inheritance
- Stacks
- Queues
- Recursion

After these topics were covered, C++ is introduced and the following topics are then covered in C++:

- C++ Memory model
- Data types
- Selection structures
- Repetition structures
- Functions
- Arrays
- Multi-dimensional arrays
- Classes
- Pointers and dynamic memory
- Linked lists

New Survey of the Discipline Course

To assist with overcoming negative perceptions of computing majors and careers, a two hour introductory course was designed and implemented as an additional part of the curriculum change. This class is intended for entering freshmen and is offered only in the fall semester. Outcomes for the course include the following:

- To develop basic problem solving skills
- To increase understanding of the fields of computer science and software engineering
- To practice both oral and written communication skills in relation to computing
- To demonstrate the ability to work in teams
- To introduce ethical issues unique to computing

Student activities are chosen to align with those objectives. Through two team-based projects, students learn problem solving skills and practice communication skills through presentations and summary papers. For each team project, one student is assigned by the team to be an observer. The role of the observer is to share with the entire class their reflection on how the team worked together. Problems are chosen that do not involve a technical solution. This is done to put the emphasis on the problem analysis rather than enabling the students to jump to a technical solution. Additional individual homework assignments cover ethical issues in computing and computer science specific topics such as the Turing Test.

Guest speakers are scheduled frequently to address the perceptions of the definition of computer science and the computing career fields. Faculty from each of the core competency areas in the CSE department visit the class to share work in their research area and discuss the types of careers that their knowledge area may lead a student to pursue. In addition, representatives from various employers of CSE graduates are brought into the classroom to share their company's career opportunities. These individuals are asked to share his or her career path and discuss the types of jobs available for computing graduates. Particular attention is paid to ensuring that the diversity of the industry representatives reflects the racial and gender diversity of the classroom.

Assessment Results

In order to determine the impact of the new introductory sequence on the success rates in the introductory sequence and retention within the major, we have analyzed information available from our student information system about the success and retention rates both before the change and after. Since most students entering the majors that require our introductory programming sequence (computer science, computer engineering, software engineering and electrical engineering) begin in CSE 1284 in the fall semester, we studied the success/failure rates in this class in the fall semesters only. Students taking this class in the spring semester tend to have weaker math backgrounds or are repeating the courses. Table 1 shows the grade distribution for the CSE 1284 class for the two falls before the curriculum change and the two falls since the change.

Table 1. Failure Rates in CSE 1284

Semester	A	B	C	D	F	W	Total	Failed	%Failed	GPA
F2008	19	34	27	23	47	7	157	77	49%	1.70
F2009	23	40	41	24	34	14	176	72	41%	1.96
F2010	49	58	43	7	6	13	176	26	15%	2.84
F2011	43	56	53	12	30	22	216	64	30%	2.36

We saw a dramatic reduction rate in the percentage of students failing the class in the first semester it was taught (we require a C or better grade to continue so D grades are counted in the failing category). This effect was reduced somewhat the following fall but still showed improvement. The GPA for the class also improved dramatically. We will continue to monitor this for the coming semesters.

Since we were also concerned with the impact of this change on retention within the major, we compared the first and second year retention rates for students who started in CSE 1284 in the two fall semesters before the change to that of those after the change. Since there is a good deal of movement between the four majors, we considered a student to be "retained" as long as they were still in one of the four majors, regardless of whether or not it was the same as their initial major. Table 2 shows these results.

As revealed in Table 2, the number of students continuing in the major after one year was improved after the change in the introductory sequence. What is also encouraging is that this seems to continue to hold true into the second year. We only have one group of students currently that are into their second year with the new curriculum, but this data is certainly promising as well.

Table 2. Retention in the Major

	Majors in 1284	Retained to second year	One Year Retention Rate	Retained to third year	Two Year Retention Rate
Fall 2008	108	70	64.81%	59	54.63%
Fall 2009	121	74	61.16%	61	50.41%
Fall 2010	139	102	73.38%	85	61.15%
Fall 2011	145	108	74.48%	-	-

Conclusions and Future Work

Observed Issues With the Change in CSE 1284

A big hurdle the first time that the new version of CSE 1284 was taught was finding a text book. The decision was made to go with the latest version of Python, but most textbooks were using version 2.7, not 3.0. Most of the ones that we were able to find were for special applications (like multi-media) or were written for experienced programmers. One was found that seemed to be good for beginning students, but it left out quite a bit of material (for example, it had reading from a file but not writing to a file). Course notes made up the difference.

Although Python seemed to simplify some concepts for students, others continued to be difficult for students to grasp. For example, *for* loops in Python were easier to grasp but *while* loops were more difficult. In addition, general program design is a difficult concept, regardless of the language. Additionally, although programming GUIs was a topic that students enjoyed, the programming model for GUIs did prove more difficult than text-based programs. Finally, exception handling and files continue to be issues that students struggle with.

Observed Issues With the Change in CSE 1384

The new version of the Intermediate Programming course (CSE 1384) presented more challenges. The first half of the semester when the students were learning new concepts in Python seemed to go well. The students are enthusiastic, and for the most part, they enjoyed the more advanced concepts. The class was definitely more advanced, but the students seemed to be adjusting to the increase in rigor.

However, after the students learned all of the new concepts, they were then re-introduced to these concepts in C++. This presented challenges because the concepts were not new and students expressed frustration at re-covering these concepts in C++. The only new concept presented was pointers, and these are historically challenging for students the first time they are exposed to them. The introduction of pointers in a language that the students were not comfortable with seemed to exacerbate this issue.

Feedback from transfer students has also been negative regarding the recent changes to 1384. Most transfer students came into the curriculum knowing either C++ or Java. None have reported having a previous knowledge of Python. These students did not want to repeat the first course just to learn the Python syntax but had difficulty picking up Python on their own. The transfer students have had a much easier time with the C++ section, but they were already frustrated by that point in the semester.

With these results from our initial change, we attempted to correct the problems by reordering the presentation of some of the material in CSE 1384. After algorithm analysis, C++ is introduced and the new concepts are then taught in tandem. Concepts are introduced with Python and then taught in C++. Stacks, queues, and recursion which were not originally taught in C++ are covered in Python, but students may use C++ if they desire. This change has made the course feel more like one course rather than 2 half-courses taught sequentially. But this order also presents additional challenges, particularly that the switching back and forth frequently makes it easy for the students to confuse the languages.

Changing the order of the presentation of material helped improve the experience for transfer students, because as each new C++ topic is taught, it is also demonstrated in Python. The transfer students just have to learn it backwards from everyone else. They are introduced to the material first in a new language (Python) and then shown the C++ (which is review). Those that only knew Java have the hardest time since they are learning two languages in the course.

Another major concern with both phases of changes in 1384 is the amount of material covered. The memory models of the two languages are very different making scope of variables particularly hard to cover. In the original order, trying to get through C++ linked lists was difficult. In the re-ordered materials, recursion as well as searching and sorting are hard to get through by the end of the semester.

Observed Issues With the New Introductory Course

Feedback over the three years that the new CS 1002 course has been offered has been positive based on University designed student feedback surveys. However, there are still some students who have left the major and verbalized the perception of not wanting to sit in a cubicle and write programs for a living. It is unclear if these are outliers or if the class is failing to overcome the negative perceptions in its current form. In Fall 2013, with input from a social scientist at MSU, a survey will be administered at the beginning and end of the semester. This will provide more quantitative data on which to measure success of the class.

Future Work

Overall the new curriculum does seem to have had a positive impact on the success of students. We still need to assess the impact on underrepresented groups which was one target group that we hoped to positively impact with these changes. In addition, we need to perform statistical analysis of these results to determine if the differences that we are seeing are statistically significant. We are also exploring ways to improve the transition from Python to C++.

References

1. Krishna Agarwal, Achla Agarwal, and M. Emre Celebi, “Python puts a Squeeze on Java for CS0 and Beyond,” Journal of Computing Sciences in Colleges, 23, 6 (June 2008), 49-57.
2. Maureen Biggers, Anne Brauer, and Tuba Yilmaz. 2008. Student perceptions of computer science: a retention study comparing graduating seniors with CS leavers. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education* (SIGCSE '08). ACM, New York, NY, USA, 402-406.
3. Lori Carter. 2006. Why students with an apparent aptitude for computer science don't choose to major in computer science. *SIGCSE Bull.* 38, 1 (March 2006), 27-31.
4. Jeff Carver, Lisa Henderson, Lulu He, Julia Hodges, and Donna Reese, “Increased Retention of Early CS/SE Students Using Pair Programming,” *Proceedings of the 20th Conference on Software Engineering Education and Training (CSEE&T 2007)*, Dublin, Ireland, July 2007.
5. Jane Margolis and Allan Fisher, *Unlocking the Clubhouse: Women in Computing*, MIT Press, Cambridge, MA, 2002.
6. Aharon Yadin, “Reducing the Dropout Rate in an Introductory Programming Course, *ACM Inroads*, 2, 4 (December 2011), 71-76.

Donna S. Reese

Donna S. Reese received her BS from Louisiana Tech University and her MS and PhD degrees from Texas A&M University, all in computer science. She is Professor and Head of Computer Science & Engineering at Mississippi State University where she has been on the faculty since 1988. Donna is a senior member of ACM and IEEE. She is past chair of the Women in Engineering Division of ASEE. Her primary research interests include recruitment and retention of women and underrepresented minorities within computing and engineering.

T.J. Jankun-Kelly

T.J. Jankun-Kelly is an Associate Professor of Computer Science and Engineering within the James Worth Bagley College of Engineering, Mississippi State University, MS, USA. His research lies at the intersection of scientific and information visualization with an interest in undergraduate education. He received his BS from Harvey Mudd College in Physics and Computer Science and his PhD from the University of California, Davis, in 2003; he is a senior member of IEEE and a member of the ACM.

Lisa Henderson

Lisa Henderson received her BS in Chemical Engineering and MS in Computer Science from Mississippi State University. She is an instructor in the Computer Science & Engineering department where she has been on the

faculty since 1988. She has been teaching the computer science introductory programming courses for over 10 years. Her primary interests include retention of women within computing and engineering.

Sarah Lee

Sarah Lee received her BS from the Mississippi University for Women in Business Administration with a concentration in Computer Information Systems. She received her MCS at Mississippi State University and her PhD in Computer Science at the University of Memphis. She is currently the Director of Undergraduate Studies Computer Science & Engineering at Mississippi State University. Her research interests include software engineering and knowledge management and recruitment and retention of women and underrepresented minorities within computing and engineering.