International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:10, No:7, 2016

Evaluation of Introductory Programming Course for Non-Computer Science Majored Students

H. Varol

***Abstract*—**Although students' interest level in pursuing Computer Science and related degrees are lower than previous decade, fundamentals of computers, specifically introductory level programming courses are either listed as core or elective courses for a number of non-computer science majors. Universities accommodate these non-computer science majored students either via creating separate sections of a class for them or simply offering mixed-body classroom solutions, in which both computer science and non-computer science students take the courses together. In this work, we demonstrated how we handle introductory level programming course at Sam Houston State University and also provide facts about our observations on students' success during the coursework. Moreover, we provide suggestions and methodologies that are based on students' major and skills to overcome the deficiencies of mix-body type of classes.

***Keywords*—**Computer science, non-computer science major, programming, programming education.

I. Introduction

COMPUTERS and digital devices are all the rage among people today, across all ages. Tablets top the youngest student's wish lists and high school students would love to have a laptop computer as well. In professional life, when it comes to business intelligence, aerospace engineering, educational technology, etc., computer skills are necessary for an employee today. Therefore, when earning business, computer animation, variety of engineering degrees, etc., programming courses are now required for these non-computer science majored students. To accommodate the need from the other majors, while some computer science units in the universities offered separate courses for these non-majored students, most of the units simply increased the number of sections and/or class sizes and invite non-majored students to take the course together with computer science students. This second option has provided some advantages to the departments and students but also have its drawbacks as well.

Creating mix-student body classes has helped computer science departments to attract more students that changed their major from something else to computer related degrees [1]. We have seen that with the increased interaction between computer science students and positive results of modified content of the course to accommodate the need of all majors teared down the myth of computer science being a hard major [1]. From the administrative perspective, creating these mix-body classes created more freedom, i.e. in terms of date/time/instructor, for the students when choosing the course to be taken. From instructors' point of view, instead of preparing for two totally different classes, with this mixed-body environment, they were able to create set of documents that addresses the needs of all students no matter what the major is.

In contrast to all the advantages of hybrid type of class, this approach also brought some concerns to the table. For instance, while instructors were using unified course materials to all students, creating an appropriate level of course work that will address the needs for both computer science and non-computer science students has been a challenge [2]. Specifically, some computer science students have felt the level of the coursework is easier than they hoped for, while still quite a number of non-majored students struggled with the materials.

Based on the observations, there is a dire need to reflect the success rates of computer science and other non-majored students in the first programming course in a mixed-classroom environment. Secondly, we need to provide suggestions and solutions to the existing problems.

The rest of the paper is organized as follows. Section II provides related literature work on programming courses and observations on both majored and non-majored students. Section III provides information about the course model that we have been employing at Sam Houston State University. Observations and discussions on the results are shared in Section IV. At the end, the paper is finalized with conclusion section.

II. Background

Traditional computer science courses, especially first programming courses have had little success to attract and also increase the success rate of non-computer science majored students. Bennedsen and Caspersen studied the potential indicators of success in programming course and found out that math grade from high school and college level math course work are significant indicators to test whether a student can be successful [3].

Wiedenbeck analyzed the factors affecting the success of non-computer science majored students in programming course. The author found that perceived self-efficacy increased significantly during the course. Both perceived self-efficacy and knowledge organization directly affected the course grade, and specific programming tasks, such as debugging. The results on self-efficacy also showed the author that the participants were overconfident about their programming capabilities [4].

McCracken et al. investigated programming competency

H. Varol is with the Sam Houston State University, Huntsville, TX 77341 USA (phone: 936-294-1075; e-mail: hxv002@shsu.edu).

after the students completed their first and second programming courses in computer science [5]. They have created a framework to be used and several universities participated in the assessment. For a combined sample of 216 students from four universities, they have obtained an average score of 22.89 out of 110 points. The authors have found that the problems were to be independent of country and educational system. They also observed that the most difficult part for students was abstracting the problem to be solved from the exercise description.

To increase the success rate in first programming course and promote the field, a number of approaches have been used from pair programming [6], [7], to creating a friendly environment by introducing virtual human helpers [8], [9]. All those and additional measures are also introduced in number of studies [10]-[12]. Besides these, constructivism, which takes a tight control on the mental model construction process in the weak students, and allows the students to navigate through many conceptual pitfalls in programming fundamentals, were used by [13]. The six guidelines that they have used on weak students increased the success of these students in their first C language programming course. Overall, all these studies, in general provided a good foundation to increase the success in the field, but still they lack the facts about different majored students' success rate in the course.

At Georgia Institute of Technology, two tailored introductory courses were introduced as an alternative to the traditional mix-body course [14]. According to the authors, more non-majors succeeded in these tailored courses than in the traditional mix-body course; less negative reactions received about the course content and this approach increased the interest in taking another tailored computer science course. However, as discussed in the previous section, not all universities created separate courses for different student bodies.

Guzdial and Forte proposed a design process for non-majored computing course. In their design process, first they set the objectives based on ACM recommendation and computer science education research literature. Second, they selected a context that allowed them to meet the objectives and motivate the non-majored students. Third, they set up feedback process from faculty and students. As the last step, they defined the infrastructure (choosing the programming language to be used) and the course. Overall, the authors achieved 90% of success rate defined as earning C or better from the course [15]. While this approach is promising, their work is focused on only non-majored students while discarding the expectations from the computer science students.

III. Course Model and Methodology

Each year at Sam Houston State University multiple sections of Programming I course are offered in Fall and Spring semesters. Roughly, around 40% of the students are majored in computer science, while rest of them are majored in the other areas. Students belong to College of Sciences (COS), College of Business Administration (COB), College of Social Sciences and Humanities (CSSH), and College of Criminal Justice (COCJ), can take Programming I as a core or an elective course based on their degrees' curriculum. Specifically, we see science students from Physics, Math, Computer Animation, Electrical and Computer Engineering Technology, Forensic Chemistry, Geography, and Geology, business students from Management Information Systems (MIS) and Finance, social science students from History, Applied Arts, Health Sciences, Kinesiology, and Interdisciplinary Studies. Also, there are some general studies students who have either not selected any major yet or failed to have a certain grade point average to be a part of degree plan.

A. Course Logistics

This course teaches Java programming language to all non-majored and majored students. Every semester, five to seven sections of the course are offered to these students while two to four instructors are assigned to those classes. PowerPoint slides, theory work on white board, tutoring videos, and hands on examples are part of the course module. Exams, multiple assignments, and weekly laboratory work are the core of grading items. Course materials are same for every section. These course materials, including hands on examples, laboratory tasks, and assignment materials are created in a way that will consist of questions based on simple mathematics, geometry, finance, business, and social sciences. Class sizes in Programming I are limited to 20 students. The laboratory is supervised with a teaching assistant with a support of (helper) two qualified graduate students.

B. Participants in This Study

The participants in this study were 107 non-computer science majored students. Although most of our discussions will be based on non-computer science students, we will make occasional comparisons with computer science majored ones which we had obtained grades for 31 of them. Most of the participants were majored in Mathematics (26%), followed by Physics at 14%. Other than couple of MIS students, the rest of them indicated that they had no prior programming experience in the first class of the semester.

In this study, the results were obtained from multiple instructors' courses in one-year period of time.

C. Obtained Results

Fig. 1 reflects the grade point averages (GPA) for different colleges based on a scale of 4.0. As shown in the figure, overall GPAs are close to each other among colleges, while CSSH has the lowest GPA. This is not a surprising result based on CSSH curriculum and non-math student profile. However, we also need to notify that computer science students grades are part of the COS GPA. Without Computer Science students, the GPA of COS would drop to 2.56, the lowest among all colleges, which is a concern that needs to be addressed.

As shown in Fig. 2, Computer Science, Computer Animation, and Geography student groups are all fared well

and had scored an average of 3.0 in Programming I course. However, Forensic Chemistry, Geology, and especially Physics majored students, who are part of the COS, have struggled from the course.

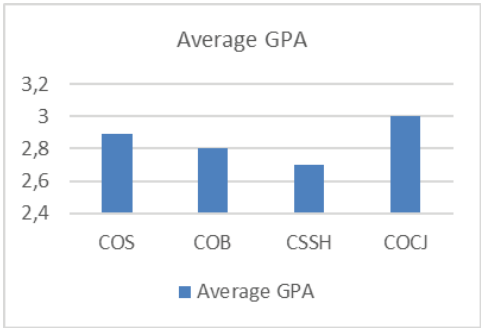

Fig. 1 Grade Point Averages Based on College

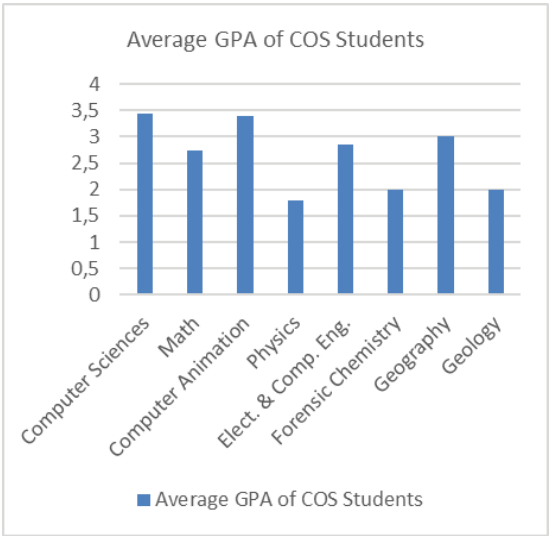

Fig. 2 Grade Point Averages of College of Science Students

When we look at Fig. 3, for all other notable majored students' GPA, except History, Health Sciences, and especially General Studies students, the rest of the majors scored acceptable GPA (2.5+) in average.

IV. Observations and Discussion

Majors not belonging to College of Sciences performed similar to those students. This indicates that the course materials were generic enough so that the understanding level of the students in one college were in the same level with the students from another college, which reflects the success of the unified course design we had implemented for Programming I course. Moreover, it is a known fact that the well-prepared related teaching materials are directly associated with retention [16]. For this reason, the laboratory materials and assignments were prepared to contain great detail of the given task. This also contributed to this successful outcome. However, for those students either have not decided on the major that they want to follow, or failed to success a particular major; thus, dropped to a general studies student level, were the ones struggled from the course.

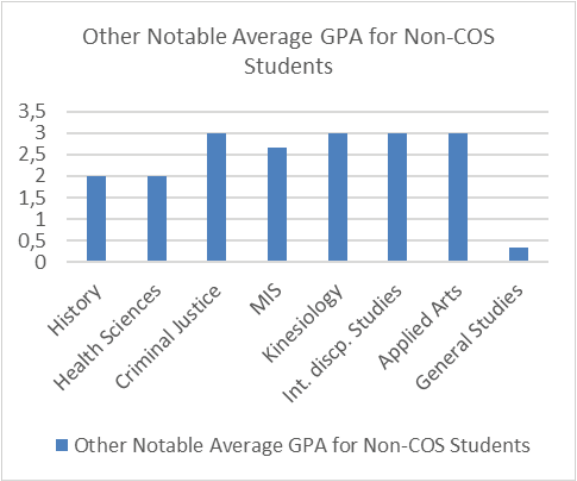

Fig. 3 Other Notable Average Grade Point Averages for Non-College of Sciences Students

Out of all College of Science students, Physics and Forensic Chemistry students were the ones who struggled from the course. This can be correlated to the school year of the students. Specifically, while most of the participants from these two majors were sophomores, more senior level participants took the class from Mathematics, Geography, etc. Another reason we observed to contribute to this result was the type of questions that the students received. As mentioned earlier, during laboratory times and hands-on experience majority of the questions required basic computing, algebra, statistics, finance, business intelligence, and/or geometry knowledge to solve them. Physics majored students struggled to answer these type of questions. The main reason for that was the lack of understanding the concept before creating the algorithm to solve it. If these materials would contain some physics and forensic chemistry related materials, then we would see higher scores in these events.

Arguably, one of the most challenging issue of teaching programming course to a mixed-body classes is while Computer Science majored students easily understand the class materials, non-majored students struggle to learn. Since many Computer Science majored students have knowledge/experience about programming, they get the class topics quickly and ask questions to consolidate or even learn more about the material. However, most of the non-majored students do not have any programming experience. Therefore, while conducting hands on examples with classroom, the instructor has to explain way(s) to solve that particular problem step by step for non-majored students. In the meantime, since most of the Computer Science majored students were able to implement their codes, they were starting to get bored and even started to disturb the classroom environment. Therefore, in order to not to lose Computer Science students' interest throughout the class time, the

instructors asked them to enhance the given examples via algorithm changes or extending it with additional capabilities. This tremendously helped to have a focused environment in the classroom.

Keeping the students engaged in the classroom is one of the most important aspects in learning. Creating a question/answer type of a class environment is proved to be an effective way of teaching. However, when you have mixed-body type classes, some negative aspects may occur in this type of education. For instance, when a question is asked to the students, most likely the answer will be given immediately by one of them who can get the materials easily or have prior knowledge about the topic, while others need some time to think over. This results in discouragement on the students who has slim knowledge on the concept/question. Eventually, these students start to avoid finding answers in the following questions. In order to keep those students motivated, it is needed to ask the questions gradually from easy to challenging. Moreover, we observed that picking a name from the class roster to answer these basic/easy questions and providing hints to the student while finding the correct answers for these questions helped to increase the attention and success rate in the course.

V. Conclusion

In this work, we have demonstrated the performances of different majored students in Programming I course at Sam Houston State University. Our main goal from this work was to demonstrate that the success level of non-majored students was similar to that of the Computer Science students. We have seen that modified course content to address the need for all students taking the course resulted as a success story. Besides, the changes on the course content, we strongly think having multiple mentors in the laboratories to help, creating collaborative learning environment in which students can reach these mentors throughout the course for exam and assignment preparation also helped us to improve the success rate of non-majored students.

As a future work, we will enhance the course materials related with struggling students' major's content. Moreover, as an extension of this work, we would like to create a pair-programming environment, in which a computer science student is matched with a non-majored student for assignments.

References

[1] H. Varol and C. Varol, "Improving Female Student Retention in Computer Science during the First Programming Course*", International Journal of Information and Education Technology*, Volume 4, Issue 5, pp. 394-398, October 2014, DOI:10.7763/IJIET.2014.V4.437.

[2] M. Urban-Lurain and D. J. Weinshank, "Do non-computer science students need to program?", *Journal of Engineering Education*, 90 (4), pp. 535--541, 2001.

[3] J. Bennedsen and M Caspersen, "An investigation of potential success factors for an introductory model-driven programming course". *Association of Computing Machinery, Proceedings of the 2005 International Workshop on Computing Education Research (ICER '05)*, pp. 155-163, 2005.

[4] S. Wiedenbeck, " Factors affecting the success of non-majors in learning to program", *1st International workshop on Computing Education Research*, pp. 13-24, 2005.

[5] W. M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B. D. Kolikant, C. Laxer, L. Thomas, I. Utting and T. Wilusz, "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students", *ACM SIGCSE Bulletin,* 33(4), pp. 125-140, 2001.

[6] C. McDowell, L. Werner, H.F. Bullock, and J. Fernald, "Pair Programming Improves Student Retention, Confidence, and Program Quality", *Communications of the ACM*, Vol. 49, No. 8, pp. 90-95, 2006.

[7] B. Bowling, H. Bullen, M. Doyle, and J. Filaseta, "Retention of STEM Majors Using Early Undergraduate Research Experiences", *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, Denver, USA, March 6-9 2013, pp. 171-176.

[8] C. Arrington, D. M. Wilson, and L. Lehmann, "Improving Performance and Retention in Computer Science Courses Using a Virtual Game Show", *Proceedings of the 49th Annual Southeast Regional Conference*, Kennesaw, GA, March 24-26 2011, pp. 320-321.

[9] S. L. Finkelstein, E. Powell, A. Hicks, K. Doran, S. R. Charugulla, and T. Barnes, "SNAG: Using Social Networking Games to Increase Student Retention in Computer Science", *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education (ITICSE 2010)*, Ankara, Turkey, June 26-30 2010, pp. 142-146.

[10] L. MacLean, "Recruitment and Retention of Women in Computer Science and Information Systems: How and Why", *2nd International Conference on Education and New Learning Technologies*, Barcelona, Spain, July 5-7 2010, pp. 1585-1591.

[11] J. Peckham, P. D. Stephenson, J. Y. Hervé, R. Hutt, and L. M. Encarnação, "Increasing Student Retention in Computer Science Through Research Programs for Undergraduates", *Proceedings of The 38th SIGCSE Technical Symposium on Computer Science Education*, Covington, Kentucky, USA March 7-10 2007, pp. 124-128.

[12] R. M. Powell, C. Murphy, A. Cannon, J. Gordon, and A. Ramachandran, "Emerging Scholars Program- a PLTL-CS Program that Increases Recruitment and Retention of Women in the Major", *University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-12-16*. January 2012.

[13] K. A. Lui, R. Kwan, M. Poon, and Y. H. Y. Cheung, "Saving Weak Programming Students: Applying Constructivism in a First Programming Course", *SIGCSE Bulletin* 36(2) pp.72-76, 2004.

[14] A Forte, and M. Guzdial, "Motivation and Non-Majors in Computer Science: Identifying Discrete Audiences for Introductory Courses", *In IEEE Transactions on Education*, 48 (2) pp. 248-253, 2005.

[15] M. Guzdial and A Forte, "Design process for a non-majors computing course", *Proceedings of the 36th SIGCSE technical symposium on Computer science education,* pp. 361-365, 2005.

[16] L. Barker, C. McDowell, and K. Kalahar, "Exploring Factors that Influence Computer Science Introductory Course Students to Persist in the Major" *SIGCSE Bulletin*, v.41, n.2, 2009. pp.282-286.

Hacer Varol received his Bachelor of Science degree in Electrical and Electronics Engineering from Firat University, Elazig, Turkey in 2003, Master of Science degree from Applied Science from University of Arkansas at Little Rock, Little Rock, AR, USA in 2011, and currently pursuing doctorate degree in Electrical Engineering from Lamar University, Beaumont, TX, USA. She has been working as Lecturer at the Department Computer Science at Sam Houston State University, since 2011. Her research interests are space network communications, computer science education, educational technology, and biomedical signal processing.