Programming in the IS Curriculum: Are Requirements Changing for the Right Reason?

John H. Reynolds
john.reynolds@gvsu.edu

D. Robert Adams
robert.adams@gvsu.edu

Roger C. Ferguson
roger.ferguson@gvsu.edu

Paul M. Leidig
paul.leidig@gvsu.edu

School of CIS, Grand Valley State University
Allendale, Michigan 49401, USA

Abstract

All curricula for any given academic discipline evolves over time. This is also true for the Information Systems (IS) model curriculum. Curriculum evolution is driven by several factors, such as changes in technologies, industry shifts to meet customer needs, and perceived student deficiencies. One outcome of such factors has been a change in the entry point into the IS major due to the perception that IS majors need a different method of entry from other computing majors (e.g., Computer Science (CS)). The current entry point for many IS majors is a programming course, often taken by a variety of majors. This paper addresses the question: is there a difference in performance in this initial programming course for students of different majors? More precisely, does major differentiate performance in the first programming course, such as CS1? The data clearly show this is not the case when there is a level playing field. The paper demonstrates that non-computing majors perform as well as computing majors given equal preparation. It is a misconception that changes to the IS curriculum are necessary when based on the belief that IS majors, as compared to other computing majors, need a different entry point. The data presented in this paper suggest the underlying presuppositions for IS curricular changes are misguided – supporting the need for preparation prior to a first programming course.

Keywords: IS Model Curriculum, CS1, Prior Programming Experience, Student Success

1. INTRODUCTION

Identifying which learning units to require in a curriculum is always challenging; and getting it "right" often takes several iterations. These learning units ultimately are linked directly to courses that are offered at an institution. Model computing curricula (Computer Science (CS), Information Systems (IS), and Information Technology (IT)) are no exception to this process, and evolve through several iterations prior to acceptance by the computing community.

The IS model curriculum has gone through several revisions; the 1997, 2002 and 2010 model curricula are examples of recent editions with significant changes. The reason for these changes are often linked to changes in technologies, attempting to meet new requirements that industry places on new IS

graduates and other related factors. "All aspects of the global computing field continue to face rapid and frequent change. As a result, university-level Information Systems curricula need frequent updating to remain effective." (Topi, Valacich, Wright, Kaiser, Nunamaker, Sipior, & de Vreede, 2010).

An important characteristic in the development of any model curriculum is the assumptions that are made about the student's ability to succeed in the initial set of programming courses, which are often at the beginning of the major. In this paper, this is referred to as the entry point into the major. This is an important feature for any curriculum, since the depth of knowledge within a major is dependent on where a student begins required coursework. If the entry point for a model curriculum programming sequence assumes a too-high level of preparation for the typical student, then a high failure rate occurs in these initial classes. In this scenario, capable students are not prepared to take these initial courses. If, on the other hand, an entry course begins with material already known by the typical student, then there is little to gain from these initial courses, and the depth of knowledge that can be obtained by a four-year degree has been decreased due to wasted time in those initial courses. Getting this entry point "right" is difficult, since students come from diverse backgrounds and have different skill sets.

The problem is further complicated in that many of these initial courses are populated by students that are not in the major. Subsequently, instructors are faced with teaching students with a wide range of interests and goals for the course.

2. BACKGROUND

Woszczynski, Haddad, & Zgambo (2005) reported that "IS students continue to struggle to complete the programming courses often required in their course of study," stating an observation held by many IS faculty. They go on to describe the need to inform IS faculty of the factors that contribute to success in programming courses in order to help advise students with course selection. Their methodology was to survey IS educators asking them what factors they believe predict success in programming principles courses.

It seems this is a case where the IS and CS worlds have not kept pace with each other's insights. As early as 1968, computer scientists were examining factors that contribute to success in programming courses (Bauer). Since then, several factors have been investigated: gender (Byrne & Lyons, 2001), learning style (Allert, 2004), abstraction (Bennedsen & Caspersen, 2006), and math and science background (Bergin & Reilly, 2005).

The one significant factor identified in several studies is prior programming experience. Kersteen, Linn, Clancy, & Hardyck (1988) discovered that the "amount of prior computing experience was found to predict course performance for males", but that females generally had little prior computing experience, and what little they did have, had no effect on course grade. Taylor & Mounfield (1989) found that "high school computer science as well as prior college computer science coursework were found to be significant factors in the success rate". Later Taylor & Luegina (1991) found "students who arrived from high school without some computer science preparation were at a great disadvantage." In a follow-up study Taylor & Mounfield (1994) found that prior computing experience for females specifically did help their performance: "The results show a significant correlation between early prior computing experiences and success by females." The pattern is clear that prior programming experience has a positive influence on the grades in early programming courses such as CS1. Another recent study found that even with the addition of new pedagogies "they have not yet leveled the playing field based on prior experience" (Alvarado, Lee, & Gillespie, 2014). While these conclusions appear to be self-evident, little interest has been exhibited for a change in the entry point for the first programming course in Computer Science programs. This ignores the differences in prior knowledge, leaving those with no prior programming experience to catch up on their own.

Only one study looked specifically at IS students, (Zhang, Zhang, Stafford, & Zhang, 2013), and they corroborated the findings of the CS community: "students' current programming skills, prior programming experience, and grade expectations are significant antecedents of learning performance". Further, there were no studies that looked specifically at the performance of non-Computing majors in CS1.

3. THE PROBLEM

The entry point for all computing majors at our institution is an introduction to programming course using Java, commonly referred to as Computer Science I (CS1). There is a growing belief at many institutions that CS1 is not a valid

entry point for IS majors. This belief is based upon an assumption that IS majors do not do as well in a CS1 course compared to CS majors. This paper dispels that assumption. Further, this paper shows that non-majors within CS1 do as well as their computing counterparts if they are adequately prepared. The methodology of this study was to evaluate all four sections of a first programming course during the same semester by comparing student success with prior preparation. For the purposes of this study, success is defined using final course grades where a B- or above is considered successful.

Students in each class were given a brief survey asking about their prior experience with programming. The possible answers were: None, Self-Taught, High-School, College Course, or repeat of this class. When students checked multiple options, the most recent one was recorded (e.g. High School and College Course would have been coded as College Course). Out of 133 students, 105 answers were given for a response rate of 79%.

The students were divided into three groups: Computer Science majors, Information Systems majors, and non-Computing majors. The following major hypotheses reflect this grouping and for each hypothesis there are five tests evaluating each of the five survey answers.

1) Are CS majors more successful than IS majors in the first programming course?

> *H_1: The percentage of CS majors with a grade of B- or higher in the first programming course will be different, regardless of prior preparation, when compared to IS majors.*

2) Are CS majors more successful than non-Computing majors in the first programming course?

> *H_2: The percentage of CS majors with a grade of B- or higher in the first programming course will be different, regardless of prior preparation, compared to non-Computing majors.*

One might argue that CS majors, regardless of their level of success, are more prepared for a first programming class, thus the following hypotheses:

3) Are CS majors more prepared than IS majors for a first programming course?

> *H_3: The percentage of CS majors will be different, regardless of prior preparation, when compared to IS majors.*

4) Are CS majors more prepared than non-Computing majors for a first programming course?

> *H_4: The percentage of CS majors will be different, regardless of prior preparation, when compared to non-Computing majors*

Finally, some might suggest that there is a significant difference in prior preparation that leads to success in the first programming course, thus the final hypothesis:
5) Are successful students, regardless of major, more prepared for a first programming course?

> *H_5: Based on prior preparation, the percentage of students with a grade of B- or higher in the first programming course will be different from those with a grade below B-.*

Statistical Methodology

For each of the five hypotheses, the null hypothesis will be accepted or rejected using the significance level of .05. To compare two independent groups based on binary variables, most statistics guidelines suggest using the chi-square test of independence as long as the sample sizes are large enough. Sauro and Lewis (2008) contend, however, that the "latest research suggests that a slight adjustment to the standard chi-square test, and equivalently to the two-proportion test, generates the best results for almost all sample sizes" (p. 75).

To determine whether a sample size is adequate for the chi-square test, calculate the expected cell counts in the 2x2 table to determine if they are greater than 5. When the values in this study met this test, the chi-square test results were used. When the values of one or the other of the subgroups did not meet this test, the N-1 chi-square test was used. The formula for the N-1 chi-square test (Sauro and Lewis, 2008) is shown in the next equation using the standard terminology from the 2x2 table:

$$\chi^2 = \frac{(ad - bc)^2(N-1)}{mnrs}$$

When the values for both groups in the study failed to meet the threshold, the more conservative Fisher Exact Test was used. The formula for this test is also given by Sauro and Lewis:

$$\rho = \frac{m!\, n!\, r!\, s!}{a!\, b!\, c!\, d!\, N!}$$

Test Results

Hypotheses are supported when the null hypothesis is rejected. In this study, the null hypothesis is rejected when there is a statistically significant difference between the proportions represented by p<.05. The first hypothesis (H_1) is rejected for all categories of prior preparation except those who answered "None." There is a significantly higher percentage of successful CS majors (23%) who had no prior programming experience. All successful IS majors in this sample had some type of prior experience.

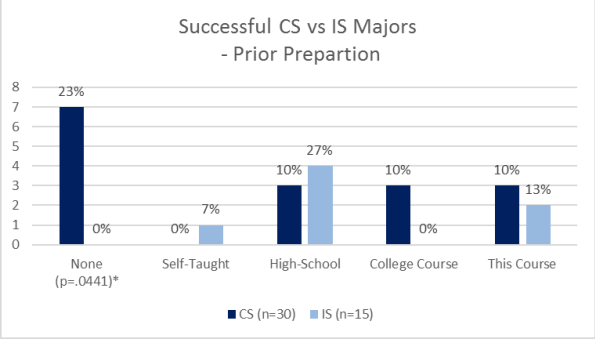

Chart 1.0 – Successful CS vs IS

The second hypothesis (H_2) is also rejected for two of the four categories of prior preparation. There is a significantly higher percentage of successful non-Computing majors (17%) taking the first programming class who had previous High School programming experience. There is also a significantly higher percentage of CS majors (10%) who are re-taking this first programming course compared to non-Computing majors.

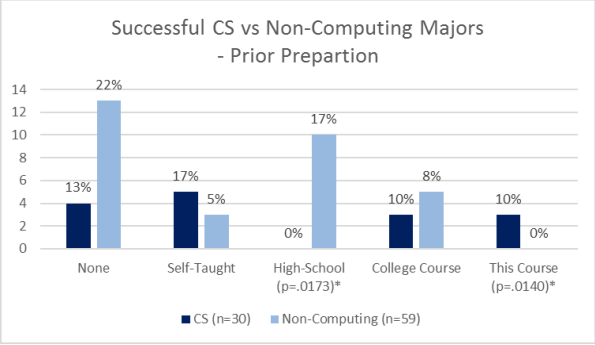

Chart 2.0 – Successful CS vs non-Computing

When comparing all CS majors to all IS majors, the third hypothesis (H_3) is also rejected for all categories of prior preparation except for those who answered "High School." There is a significantly higher percentage of IS majors (40%) who had prior programming experience.

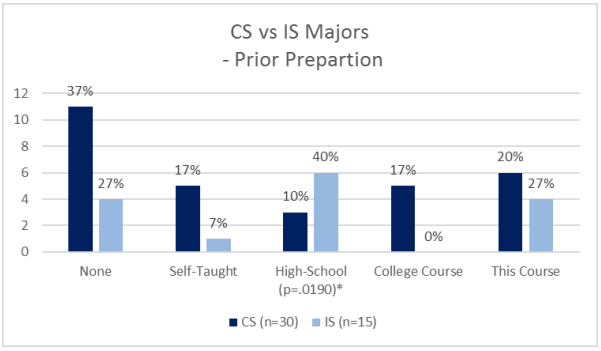

Chart 3.0 – All CS vs IS

The fourth hypothesis (H_4) is also rejected for all categories of prior preparation except for those who are re-taking this course. There is a significantly higher percentage of CS majors (20%) who had taken this course before.

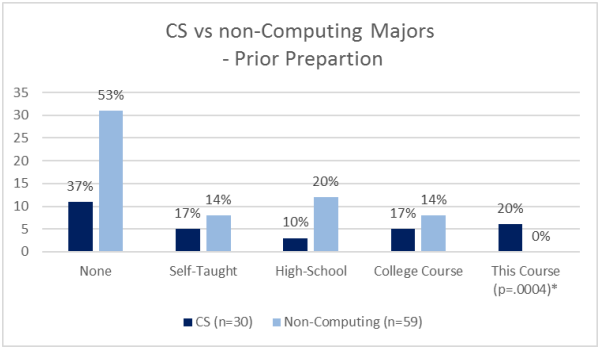

Chart 4.0 – All CS vs non-Computing

When combining all students who earned a B- or above, the fifth hypothesis is rejected for all of the categories of prior preparation except one – those who answered "None". There is a significantly higher percentage of students who earned less than a B- (57%) who had no prior programming experience.

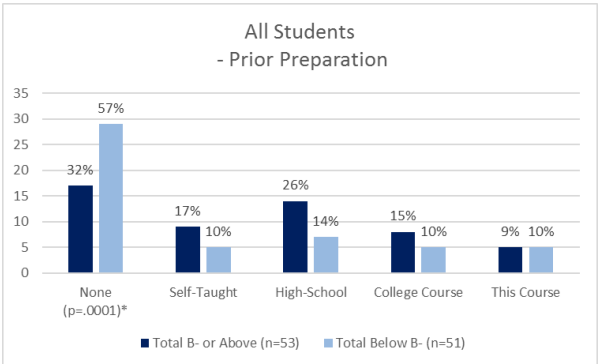

Chart 5.0 – Successful vs Unsuccessful Students

4. DISCUSSIONS AND CONCLUSIONS.

A summary of the study results is that; there was no difference in the success of students based on their declared major, and perhaps most importantly, programming experience prior to an introductory course improves the level of success. The results of this study match and confirm earlier research indicating prior programming experience is the primary predicting element for successful completion in an introductory programming course. While on its face value, this seems like a simplistic conclusion, the data show a large portion of students in an introductory programming course (CS1) have already had some exposure to programming. This creates the dilemma that instructors must find a starting point common to the majority of the class. This often leaves the minority group of students with no prior programming experience to fend for themselves. This begs the question if these "introductory" courses are truly an introduction to programming.

Anecdotal observations of curricular changes indicate that CS and IS programs have reacted differently to this challenge. Many IS programs simply reduced or eliminated programming courses, evidenced by changes in the IS2010 Model Curriculum. Introductory programming courses in CS programs often evolved to the point where prior experience was assumed. Both of these evolutionary changes may have had a negative effect on the career preparation of graduates.

The data from this study suggest that all incoming CS and IS majors should have some kind of prior programming experience, whether through self-study or through a high school or preparatory programming course. In many disciplines this is the case as math, science, and foreign languages are included in almost all high school curricula, but computing programs (e.g. AP CS) are rarely offered, much less required, in high school. Even so, these disciplines also provide remedial courses for those students who do not have the assumed background, with varying policies on if or how these courses count toward a particular degree. It could be argued that in many programs, such as MIS majors in business schools, the solution to this lack of experience led to the elimination of programming altogether, and in many computing programs, the entry point evolved to the detriment of those without experience.

At a time when industry demand for computing talent, specifically programming talent, is far outpacing the output of university level graduates, this evolution of curricular changes could have a detrimental effect on the number of computing graduates. Thus, finding a solution is imperative. From a curriculum standpoint, these findings should be incorporated into specific expectations of incoming students and that a remedial level, truly introductory course in computing concepts and programming principles be incorporated as it is critical to improving the success rate of computing students.

5. FUTURE WORK

First, this pilot study has confirmed earlier research recommending that prior programming experience is critical to the success of computing majors. Second, the research at GVSU established that this is true for all students taking a first programming course, regardless of major. This study needs to be extended with additional data from other institutions, including a variety of computing majors taking the first programming course. Furthermore, additional data is necessary to determine which type and/or amount of prior programming experience may more influential in student success. The authors are seeking collaborators to continue this research in both areas.

6. REFERENCES

Allert, J. (2004). Learning Style and Factors Contributing to Success in an Introductory Computer Science Course. In *Proceedings of IEEE International Conference on Advanced Learning Technologies* (pp. 385-389).

Alvarado, C., Lee, C., & Gillespie, G. (2014). New CS1 Pedagogies and Curriculum, the Same Success Factors? In *SIGCSE '14 Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 379- 384).

Bauer, R. (1968). Predicting performance in a computer programming course. Retrieved from ERIC database. (ED026872)

Bennedsen, J., & Caspersen, M. E. (2006). Abstraction Ability as an Indicator of Success for Learning Object-oriented Programming? *ACM SIGCSE Bulletin*, 38(2), 39-43.

Bergin, S., & Reilly, R. (2005, February). Programming: Factors That Influence Success. *ACM SIGCSE Bulletin* (37(1), 411-415).

Byrne, P., & Lyons, G. (2001). The Effect of Student Attributes on Success in Programming. *ACM SIGCSE Bulletin* (33(3), 49-52).

Kersteen, Z. A., Linn, M. C., Clancy, M., & Hardyck, C. (1988). Previous Experience and the Learning of Computer Programming: The Computer Helps Those Who Help Themselves. *Journal of Educational Computing Research*, 4(3), 321-333.

Sauro, J. and Lewis, J. (2008). *Quantifying the User Experience: Practical Statistics for User Research.* Morgan Kaufmann, Burlington, Massachusetts.

Taylor, H. G., & Luegina, M. (1991). An Analysis of Success Factors in College Computer Science: High School Methodology is a Key Element. *Journal of Research on Computing in Education*, 24(2), 240-245.

Taylor, H. G., & Mounfield, L. C. (1989). The Effect of High School Computer Science, Gender, and Work on Success in College Computer Science. *ACM SIGCSE Bulletin,* 21(1), 195-198.

Taylor, H. G., & Mounfield, L. C. (1994). Exploration of the Relationship Between Prior Computing Experience and Gender on Success in College Computer Science. *Journal of educational Computing Research*, 11(4), 291-306.

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K., Nunamaker Jr, J. F., Sipior, J. C., & de Vreede, G. J. (2010). IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. *Communications of the Association for Information Systems*, 26(1), 18.

Woszczynski, A., Haddad, H., & Zgambo, A. (2005). An IS Student's Worst Nightmare: Programming Courses. *In Proceedings of the Southern Association of Information Systems Conferences* (pp. 130-133).

Zhang, X., Zhang, C., Stafford, T. F., & Zhang, P. (2013). Teaching Introductory Programming to IS Students: The Impact of Teaching Approaches on Learning Performance. *Journal of Information Systems Education*, 24(2), 147.

