Results of using a Multi-Programming Language Approach to Decrease Drop-Fail Rates in CS1

Ed Lindoo, Ph.D
Regis University
3333 Regis Blvd.
Denver, CO 80221-1099
303-964-6358
elindoo@regis.edu.

ABSTRACT

In late 2018, we presented a paper [6] pointing out the high drop-out and failure rates (D/F) in our two introductory programming courses within the CIS department at Regis University. The study performed over a three-year period found D/F rates in excess of 60%. High drop and failure rates are, of course, a big concern for institutions, instructors and students because there are a lot of ramifications for each of these. One of the things we pointed out through our research is that even though languages such as Java, C and C++ are popular, there has been much debate about the suitability of these languages for education, especially when introducing programming to novices.

In the previous paper we presented the idea of going back to the basics, BASIC programming that is, but not with traditional programs like QBASIC or MS-BASIC. Instead we introduced a product called Xojo (pronounced Zo-Jo). Upon implementing this, we decided to take it a step further. Following the work of Hong et. al. [5] a multi-programming environment was implemented in which we teach BASIC and Java at the same time, comparing and contrasting the two languages. Since introducing this in the fall of 2018, students went from a 38% pass rate to an 76% pass rate in this multi-programming course, where Xojo and Java are taught together. While this is a great success, it should be noted that those who went on to the follow-up Java course passed with a 75% success rate, as opposed to 47% previously. The audience for this paper includes computing faculty that are planning, designing, developing, revising or implementing a new course that introduces programming concepts to novice programmers.

INTRODUCTION

The challenge at our university, specifically within the College of Computer Information Sciences (CC&IS), was to reduce the high drop-fail (D/F) rates that we have seen over the past three years. Figure #1 shows these rates for CIS and CS majors for the two-and-a-half-year period starting in August 2015, ending with the January 2018 term.

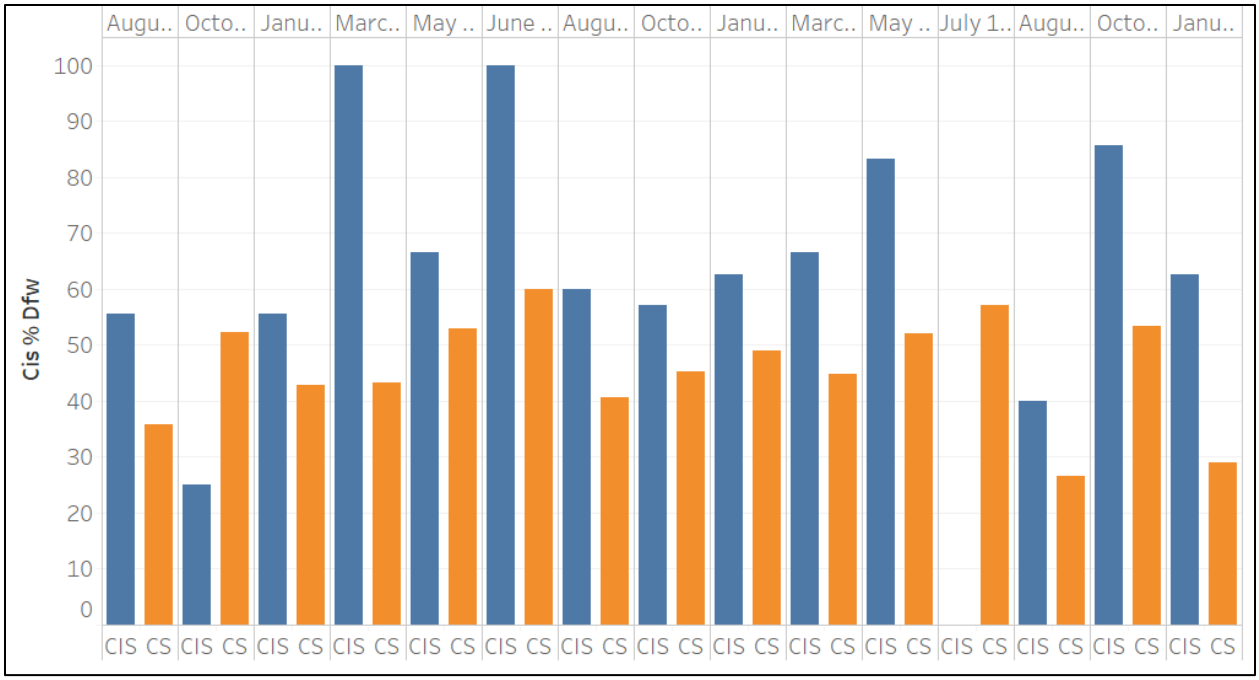

Figure 1

As shown in Figure 1, CIS majors struggle considerably more than their CS counterparts. When summed, CIS majors average a 62% DF rate while CS majors have averaged a 42% DF rate. Combined, the two programs average approximately a 50% DF rate, a rate that we cannot sustain. Therefore, the challenge was to develop an introduction to programming course for CIS majors that will help them better retain what they learn, write better code and have a good understanding of the programming language (PL) that will be used in their next course, Java.

To fix any problem, one needs to understand it, thus we performed extensive research on the successes and failures of intro to programming courses. We quickly came to realize that students' success or failure in the introductory programming course often determines their majoring in computer science and related fields, yet many schools are challenged with high D/F rates [4,6,7]. For this very reason our first paper looked at why this is happening, and while there is no single reason, we found a number of factors that contribute to failure. For example, during the past four decades, many languages have been used for teaching introductory programming. Pears, et al. [8] provided a well-researched survey on the literature in teaching introductory computer courses and found that, C, Java, C++ and Python top the list of the most widely used programming languages. The Tiobe programming community index [10] (https://www.tiobe.com/tiobe-index/) also provides this information for the industry, is updated monthly, and interestingly has changed little over the years. Even though these are the most popular in business, Pears et al. [8] concluded that the best choice of a programming language (PL) is elusive, and in fact, a popular PL (ie: C, Java or Python) is not necessarily the best for teaching intro to programming courses. Of course, the language choice is usually made locally, based on factors such as faculty preference, industry relevance, technical aspects of the language, and the availability of useful tools and materials. However, we found in research that this process has become increasingly cumbersome for professors as the number of languages has grown.

One of the important things we pointed out in our previous paper is that Alturki [1] found that most institutions are starting students out in Java, C, C++, Python or Ruby. However,

Barland [3] a professor at Radford, argues that C/C++ and Java are poor choices to learn as a first language. He went on to explain that, "Programming is a difficult task, learned over months and years. Object-oriented programming (the "++" part of "C++") is a more advanced topic which is important for larger programs but is best taught after the fundamentals have been learned." Barland [3] pointed out that most teachers realize that they should not distract from a topic by teaching advanced details to a beginner, yet that is exactly what's happening in these intro classes if we throw C++ or Java at them. As Malik [7] found in their intro to programming course, programming exercises were too difficult and as a result they were seeing about a 30% drop rate. Malik [7] also pointed out that the usual approach in teaching programming is to start with the syntax of a programming language (usually Java or a flavor of C) and move on to the associated semantics. This has proven to not be the best approach as motivation can quickly drop.

In the College of Computer and Information Sciences at this authors university, intro to programming and a second class in Java is required to continue in the CIS program. If a student drops out, fails, or passes with difficulty, Malik [7] found that it is unlikely that the student will enroll in a follow-up course. This, of course, represents a huge loss of revenue to the University, and with high D/F rates, not a model that can be sustained. As a result, the goal of our previous paper was to find a way to improve these D/F rates. Easier said than done, but as Alturki [1] pointed out, researchers have found that failure is not always associated with cognitive ability, but often with motivation and teaching style, both in industry and education. Likewise, A'lvarez [2] found that motivation in programming courses is directly related to student satisfaction. Because of this we decided to focus on motivation.

RESEARCH

Motivational and Demotivational Factors

As Alturki [1] explained, probably the biggest challenge in learning programming is to acquire different sets of skills at the same time. He found that new students had to learn both syntax and semantics of a programming language, while at the same time develop problem solving skills. To further compound this, most programming courses require students to study theoretical concepts and practice these concepts while designing and developing programs. Therefore, novice programmers must learn multiple concepts and apply these in a practical manner concurrently, often pushing students into overload [7]. Pillay and Jugoo [9] conducted a study to identify the relationship between student characteristics and performance in a first course in procedural programming. In this study, they found that for students who have a strong Mathematics background there is a positive correlation between the students' problem-solving ability and their programming performance in the course. In their 2016 study, A'lvarex and Larran~aga [2] found that the biggest problem novice programmers face is their lack of program solving skills (much of which involve math). This produces high drop-out and failure rates in programming courses.[2] Though we did not pursue this in our efforts, we point it out because most institutions including ours do not have a math class as a pre-req to the intro to programming course.

In our previous paper we presented the idea of going back to the basics, BASIC programming that is, but not with traditional programs like QBASIC or MS-BASIC. Instead we introduced a product called Xojo (pronounced Zo-Jo), which is a free programming language

that is quite like Visual Basic. As we started down that path, however, we came to realize that we're really just teaching another language, BASIC. Yes, a much easier language to teach (and learn) than "C" for example, but our ultimate goal is preparing students for the next class, which is Java. Time for something untraditional! Through further research, we found that Hong et al. [5] set out to make programming more interesting and more relevant to real-world problems. To accomplish this, their hypothesis was that "because children raised in an immigrant family are able to master two very different languages in their early childhood, first-time programming students should be able to learn multiple PLs at the same time." [5]. Hong et al. [5] also felt that just as a bilingual child can better appreciate two languages, programming students can also appreciate the designs and strengths of different PLs by way of a comparative study. Using control groups, the study [5] concluded that students can learn multiple PLs simultaneously without having a negative effect on their learning. They also found that students in a multiple PL environment learned and retained better and wrote better code than those students in a single PL environment. Specifically, within Hong's [5] experimental group, they observed that once a student came up with a solution in one PL, the student was often able to easily convert the solution to another in a different PL.

Teaching Multiple Programming Languages

Rather than teach one language, followed by another, like Hong [5] we proposed to teach multiple languages in our Introduction to Programming course. As previous authors [1, 3, 4, 8] have found, languages such as C/C++ and Java are poor choices to learn as a first language. Given the research presented, and to provide students with a more meaningful learning experience, a new course (CIS-275) was created within the CIS department to incorporate two programming languages, Xojo and Java. Xojo on the surface is a drag and drop programming environment with base code in BASIC. With Xojo, many different types of apps can be built, including web-based apps, console (or command line) apps, iOS apps and just released Android apps. Xojo is an object-oriented environment and is quite similar to VB.NET.

Research proved [5] that students can grasp multiple programming languages, thus we introduced our students to Xojo and Java. By using multiple programming languages our hypothesis was that the course will be more motivational, and students should be able to master multiple languages within the course. Overall, the goal was to greatly reduce these DF rates, and this was accomplished well beyond our initial goals. In the three terms completed since introducing this new method, on average, students went from a 38% pass rate to an 76% pass rate in this multi-programming course, where Xojo and Java are taught together. While this is a great success, it should be noted that those who went on to the follow-up Java course passed with a 75% success rate, as opposed to 47% previously. Also, of note, there were no students in the intro course that made it to the end and failed. In fact, all who did fail either never showed up for class or stopped contributing after the first week. This may be indicative of first courses that students take, but we did not pursue this observation.

In our informal experiment we found that assigning/offering both BASIC and Java simultaneously was beneficial to students understanding in that it helps them appreciate the differences among individual language designs. For example, for the same problem, we observed that when given a specific assignment in BASIC, students quickly and easily adapted it to Java. We also observed the existence of significant synergy in learning multiple PLs among the

students. In our study we found that students learned to write better code in one PL by learning the other at the same time. One piece of clear evidence is that students in the new class started to write clearer Java code with proper syntax after they learned to code in BASIC. Our choice to teach BASIC and Java simultaneously has also proven enlightening. As shown in table 1, we point out to students that BASIC and Java have many code similarities, and in the case of these statements the only real difference is the semicolon at the end of each line of Java.

BASIC Code	Java Code

Dim myName As String
myName = "Elvis Presley"
Dim oneNumber As Integer
oneNumber = 5
Dim oneNumber As Double
oneNumber = 5.3
Dim thisIsAwesome As Boolean
thisIsAwesome = True
 	
Note in Xojo variables can also be declared and initialized in one line.
Dim myName as String = “Elvis Presley”
Dim onNumber as Integer = 5
	String myName;
	myName = "Elvis Presley"; //don’t forget the semicolon.
	int oneNumber;
	oneNumber = 5;
	double oneNumber;
	oneNumber = 5.3**;**
	boolean thisIsAwesome;
	thisIsAwesome = true;
	
	Just as in Xojo, Java variables can also be declared and initialized in one line.
	String myName = “Elvis Presley”;
	Int oneNumber = 5;

Table 1

In one early exercise for example, students learn how an if-then-else works. This is a simple program that takes input from a form and presents the results in a message box.

```
Dim myAge As Integer
myage = cdbl(age.text)    'cdbl converts string to Integer
If myAge > 17 Then
  MsgBox "You are old enough to vote"
Else
  MsgBox "You are not yet old enough to vote"
End If
```

Next students write basically the same program in Java:

```
public class NewClass
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      // Prompt the user to enter age
      System.out.print("Enter an age ");
      double myage = input.nextDouble();

      if (myage > 17)
      {
         System.out.println("You are old enough to vote");
```

```
      }
      else
        {
        System.out.println("You are not yet old enough to vote" + myage);
        }
   }
}
```

The “If” statement is similar, however the rest is a bit different, and it is in these early programming examples that we explain the use of brackets, comments, and semicolons in Java. By the time we get to week 6 of the course, students are not only following the Xojo manual and writing pretty sophisticated BASIC code such as sending an email, but they also do it in Java, as shown here.

```
public class SendEmailApp {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) throws IOException, AddressException,
MessagingException {
       //Setup system properties
       Properties properties = System.getProperties();
       properties.put("mail.smtp.starttls.enable", "true");
       properties.put("mail.smtp.auth", "true");
       properties.put("mail.smtp.host", "smtp.gmail.com");
       properties.put("mail.smtp.port", "587");
       String senderEmail = "XXX@gmail.com";
       String senderPassword = "xxx";

       //Instantiate mail session, compose email including subject, receipient & content
       Session mailSession = Session.getDefaultInstance(properties, null);
       MimeMessage mailMsg = new MimeMessage(mailSession);
       mailMsg.addRecipient(Message.RecipientType.TO, new
InternetAddress("XXX@some.edu"));
       mailMsg.setSubject("Test java mail - Test");
       mailMsg.setContent(" CIS 275 - This is test email. ", "text/html");

       //Connection  to sender email & send
       Transport transport = mailSession.getTransport("smtp");
       transport.connect("smtp.gmail.com",senderEmail, senderPassword);
       transport.sendMessage(mailMsg, mailMsg.getAllRecipients());
       transport.close();
   }
}
```

As we move through each week in the course, our focus is on programming concepts and required syntax, with an emphasis on the comparison of Xojo and Java code in the visual environment. The goal is to prep the students for their next course which is purely a Java based course. For the Java piece, we introduce students to the Apache Netbeans visual environment. Similar to Xojo, Netbeans is a visual, drag and drop environment with Java as the underlying code. Building upon weekly concepts in both Xojo and Netbeans, students write simple, but meaningful, real-world programs in both Xojo and Netbeans. We believe this simple concept is what kept students motivated in the course, while at the same time preparing them for their future programming course.

RESULTS

Since going to this new format, the class has run three semesters with the following results in the fall of 2018, spring 2019 and fall 2019.

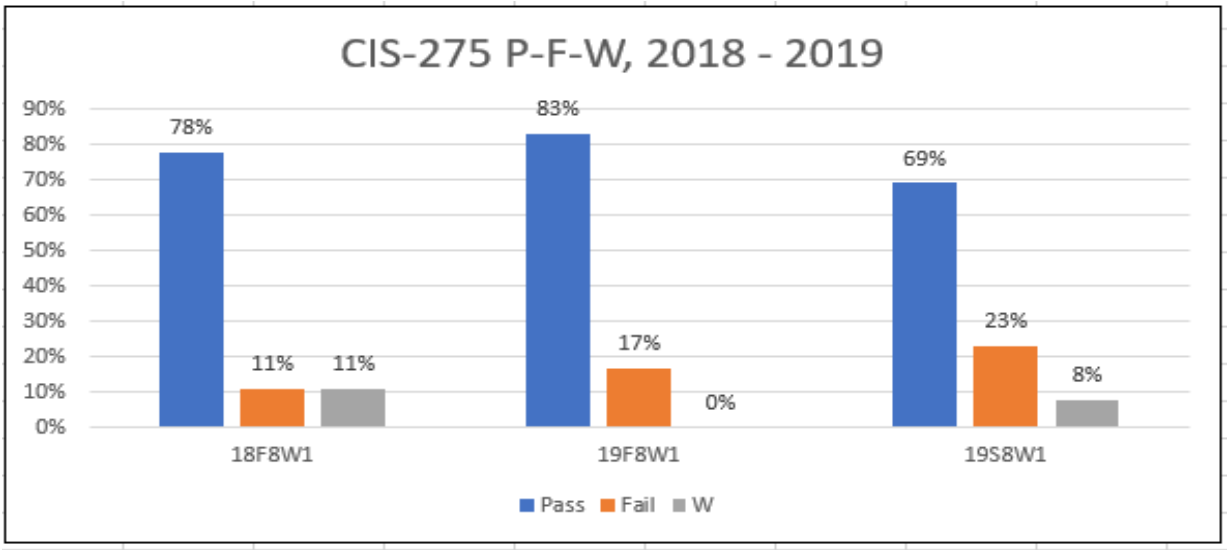

Figure 2

As shown in figure 2, we've gone from a previous passing average of 38% for CIS students to an average of 76.6% or more than double. It should be noted that of the 17% (average) who failed the course, all stopped attending after the first week. Why they didn't withdraw is unknown.

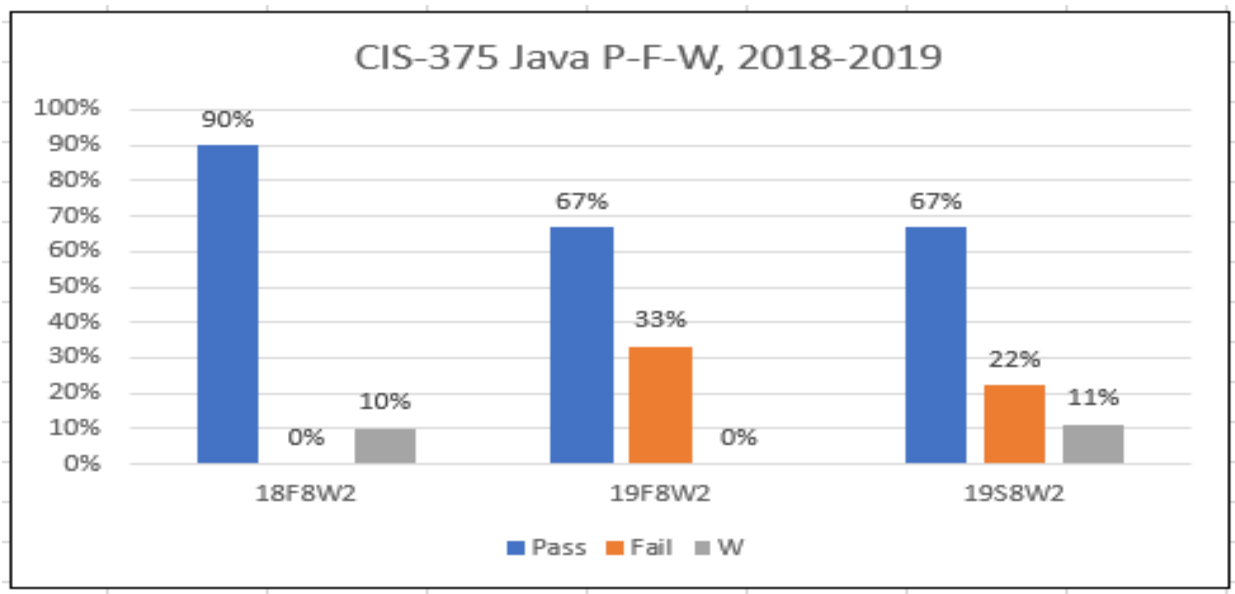

Figure 3

In the course that follows, CIS-375 (Java) we now see a passing rate of about 75% (figure 3) as opposed to 47% previously.

CONCLUSION

In this paper we presented an approach that teaches multiple programming languages simultaneously and comparatively to beginning programming students. Based on our previous proposal to go back to the basics, BASIC programming, we experimented with this approach by teaching BASIC and Java simultaneously. Going into this project there was concern that we might make matters worse. For example, is it possible for students to learn multiple programming languages simultaneously, or will this turn out to be a flop? The results overwhelming show that learning two programming languages simultaneously is not only possible, but also greatly reduced our D/F rates.

We have proven that when students are exposed to multiple PLs early, they can handle it without difficulty. Since our introduction of the two-language concept, we have seen that there is a clear benefit: learning multiple PLs simultaneously without loss of learning gains. Through this process we have found the need to focus more on teaching fundamental programming concepts rather than teaching distinct language syntax structures and features, and only teach these after students have a grip on the fundamentals.

Simplifying the course structure by reducing complexity has proven to have a positive effect on student performance, motivation and retention. Students focus more on the fundamental concepts in programming. By introducing these concepts in the Xojo environment, as well as contrasting Xojo programs to Java programs, students don't get bogged down in low-level programming languages that should be introduced later in a CS or CIS curriculum. If fact, as previously noted, we have a proven success rate of 76.6% versus 38% in the intro to programming course, and a 75% success rate of those who went from the intro course to Java.

REFERENCES

[1] Alturki, R. Measuring and improving student performance in an introductory programming course. *Informatics in Education*, 2016, Vol. 15, No. 2, 183–204, Vilnius University (2016)

[2] A´lvarez, A. and Larran˜aga. Experiences incorporating Lego Mindstorms robots in the BASIC programming syllabus: Lessons learned. *Springer Science+Business Media Dordrecht:* DOI 10.1007/s10846-015-0202-6 (2016)

[3] Barland, I. Why C and C++ are awful programming languages. https://www.radford.edu/ibarland/Manifestoes/whyC++isBad.shtml (2017)

[4] Carvalho, E. S. Analyzing the quality of student's interaction in a distance learning object-oriented programming discipline. *Interdisciplinary Journal of e-Skills and Life Long Learning*, 11, 85-99. Retrieved from http://www.ijello.org/Volume11/IJELLv11p085-099Carvalho0919.pdf (2015)

[5] Hong, G., Yao, J., Michael, C. and Phillips, L. A multilingual and comparative approach to teaching introductory computer programming. *The Journal of Computing Sciences in Colleges*, 33 (4), 4-12, (2018)

[6] Lindoo, E. Back to the basics, in an effort to improve student retention in intro to programming classes. *The Journal of Computing Sciences in Colleges*, 34(2), 72-80. (2018)

[7] Malik, S. and Coldwell-Neilson, J. A model for teaching an introductory Programming course using ADRI. *Springer Science+Business Media*, New York: DOI 10.1007/s10639-016-9474-0 (2016)

[8] Pears, A., Seidman, S., Malmi, L, Mannila, L., Adms, E., Bennedsen, J, Devlin, M., Paterson J. A survey of literature on the teaching of introductory programming. ACM SIGSCE Bulletin 39,4, 204-223 (2007)

[9] Pillay, N., Jugoo, V.R. An investigation into student characteristics affecting novice programming performance. *ACM SIGCSE Bulletin*. DOI: 10.1145/1113847.1113888 (2005)

[10] Tiobe Programming Index. https://www.tiobe.com/tiobe-index, *Victory House II, Company Number 2089*, Esp 401, 5633 AJ Eindhoven, The Netherlands (2020)