Whatever Happened to Richard Reid's List of First Programming Languages?

Robert M. Siegfried
siegfrie@adelphi.edu

Daniel M. Greco
DanielGreco@mail.adelphi.edu

Nicholas G. Miceli
NicholasMiceli@mail.adelphi.edu

Jason P. Siegfried
jasonpsiegfried@yahoo.com

Department of Mathematics and Computer Science
Adelphi University
Garden City, NY 11530 USA

Abstract

Throughout the 1990s, Richard Reid of Michigan State University maintained a list showing the first programming language used in introductory programming courses taken by computer science and information systems majors; it was updated for several years afterwards by Frances Van Scoy of West Virginia University. However, it has been 5 years since the last Reid List was released. An updated list was compiled revealing the most popular programming languages. The resultant correspondence with faculty members at many of the 410 Reid List colleges and universities indicates several trends, some of which are contradictory, as well as the reasons for the language choices of the participating schools. We present several conclusions from our findings.

Keywords: introductory programming, programming languages, objects early approach, Java, C++, Python

1. INTRODUCTION

The choice of programming language and pedagogic approach used in teaching an introductory programming course for computer science and information systems majors has been a subject of debate for the past forty years. Holt (1973) criticized the use of PL/I in beginning programming courses while Conway and Wilcox developed a PL/I compiler that was better suited to student use. Pascal was the dominant programming language in introductory courses after Wirth (1971) introduced it, but not even Pascal at the peak of its popularity was immune from criticism. Kernighan (1981) described it as "meant for learning" but he found it ill-suited for serious programming work; Habermann (1973) concurred with this assessment. Brilliant and Wiseman (1996) found that most of the faculty whom they surveyed favored Pascal but consid-

ered it too dated for continued use as an instructional language. Johnson (1995) considered C too complex a language for beginning programming students (the faculty surveyed by Brilliant and Wiseman agreed).

More recently, the Advanced Placement exams in Computer Science has moved from using Pascal to C++ and more recently to Java. While the move from Pascal to C++ reflected the growing popularity of object-oriented programming and the maturity of the C++ language, the shift to Java came about partly because of the belief that it was an easier language to learn (Hadjerroult 1998; Madden and Chambers 2002). However, Java presents its own challenges as a teaching language. King (1997) considered Java to have many advantages as an introductory language, although he recognized that it also had many disadvantages. This has led to a sort of dichotomy, where many computer science and information systems programs use Java because of its popularity, or its inherent advantages, while other schools choose not to use it as a first language because they consider it too difficult to teach to beginners.

The very fact that Java is an inherently object-oriented language has led to a debate on the approach that ought to be used in teaching programming, i.e., whether objects should be introduced early or somewhat later. Bruce (2004), Buck and Stucki (2000), and Decker and Hirschfeld (1994) all argue in favor of an object-early approach. However, Reges (2006) claimed that returning to an objects later approach helped improve retention in the introductory programming sequence at the University of Washington. McConnell and Burhans (2002) noted how much thicker introductory programming texts had become and the need to cover objects led to fewer pages on fundamental topics such as repetition and selection statements.

As a result, the question of the language and pedagogic approach to be used when teaching introductory programming courses remains a "hot button" topic within the computer science and information systems educational communities. The adoption of Java as the language of the Advanced Placement courses appears to make it the unofficial programming language of introductory programming; however, there are several other languages in common use, and many colleges that use Java as their programming language of instruction differ in their choice of approach, with some schools teaching objects early, others teaching objects later and some essentially teaching Java as an imperative language.

The purpose of this study was to see if there is any commonality among computer science and information systems programs in the way in which they teach introductory programming. It would be ideal to conduct a census similar to the ones conducted by deRaadt, Watson and Toleman (2004; 2002), where they surveyed university computing programs in Australia and New Zealand, to determine their language of instruction, programming paradigm and the reasons for these choices. Unfortunately, while it is possible to do this in Australia and New Zealand, where there are only 37 and 8 teaching universities respectively, it becomes much more difficult to do this in the United States where there are over 3000 colleges and universities, of which an estimated 1350 have a computing program (Davies, Polack-Wahl and Anewalt 2011). For this reason, we elected to use the Reid List of First Programming Languages as a representation of the population.

.

2. WHAT IS THE REID LIST?

Richard Reid, who taught Computer Science at Michigan State University, began tracking colleges computing programs and the languages that they used in their introductory programming course in the early 1990s. To some extent, the sample was self-selecting; colleges were included on the list if they replied to Dr. Reid and provided him with reliable information about the language used in the computing program. The list was updated continuously and when 10% of the colleges on the list changed their language of instruction, a new list was released (Reid 1992). New lists appeared approximately twice per year until Reid's retirement in 1999. Subsequently, Frances Van Scoy, a former student of Dr. Reid, continued compiling the list, with the twenty-fifth Reid List in 2006 being the last one released (Van Scoy 2006).

The twenty-fifth Reid List included 410 colleges and universities, with 391 of the colleges representing the District of Columbia and 49 states (Wyoming is the only state without representation). A breakdown by region appears in Table 1. While there is reasonable geographic balance, there are some states that are far more heavily represented than others. Table 2 shows the states with 10 or more colleges in the Reid List. While New York, California and Pennsylva-

nia are among the more populous states, their influence on the List may be overstated when compared to the number of colleges in Texas and Florida. Additionally, Massachusetts and the New England states as a whole are significantly overrepresented in comparison to its college-age population. This is partially due to the presence of all eight Ivy League colleges and MIT, in addition to four of the five University of Massachusetts' campuses (the fifth is the Medical School). Both New York and California have decentralized public universities; all four of the main campuses of the State University of New York (SUNY) are included as well as five of the smaller SUNY colleges. Eight of the ten University of California campuses are included as well as eleven of the twenty-three California State University campuses.

There were also nineteen universities from outside the United States. Fourteen of the schools were from English-speaking countries, with eight from the United Kingdom, five from Canada and one from Australia. The other five universities were European.

Table 3 shows the breakdown by the highest degree program offered in computing. There is an almost even breakdown between undergraduate, master's- and doctorate-granting departments; however, only nine of the programs were in community colleges, which are significantly underrepresented. There was one vocational/technical school on the list.

Table 1. Geographic Breakdown of the US colleges in the Reid List

Region	Colleges
New England	41
MidAtlantic (incl. DC)	87
Southeast	72
Kentucky and W. Virginia	10
MidWest	95
SouthWest	68
Northwest	16
Alaska and Hawaii	2

A breakdown of the sample indicates that 250 of the colleges were public and the rest were private; however, with the exception of the University of Delaware which is a state-supported private university. Of the 158 private colleges, seventy-four are affiliated with religious denominations, with the thirty-one Catholic colleges being the most heavily represented religious affiliation.

Finally, seven of the schools, including the only vocational/technical school, no longer offer a computing program. E-mail correspondence and telephone conversations confirmed that these programs were discontinued due to low enrollment.

Table 2. States with ten or more colleges in the Reid List.

States	Colleges
New York	34
California	32
Pennsylvania	29
Massachusetts	20
Ohio	17
Missouri	13
Texas	13
Virginia	13
Illinois	11
North Carolina	11
Florida	10
Indiana	10
Michigan	10
New Jersey	10

Table 3. Breakdown by Highest Degree Offered in Computing

Highest Degree Awarded in Computing	Colleges
Associate's	9
Bachelor's	128
Master's	109
Doctorate	157
No longer offering a computing program	7

3. METHODOLOGY

The colleges and universities included in this survey were taken from the twenty-fourth Reid List; many of the 410 schools listed on the twenty-fourth list did not appear on the twenty-fifth list, which only listed 153 schools. The requirements for the Bachelor's program in Computer Science were examined to determine what the first required programming course was. If the school offered both Bachelor of Arts and Bachelor of Science programs, the requirements for the BS were used. In the case of the community colleges, the requirements for an Associate's

degree in Computer Science were examined. Finally, if the school did not have a Computer Science program, the requirements for the Information Systems program were used.

After finding the first programming course, the course description was examined to see if it included the programming language of instruction; however, most did not specify the language. If a current syllabus for the course was available online, then an examination of its content was used to make a determination of the language used in the course. However, if there was no syllabus online, the bookstore's web site was checked for a textbook adoption; in some cases, the bookstore was called in an attempt to get this information. Lastly, if these steps did not provide the programming language in use, then members of the department were contacted to obtain this information.

4. THE TWENTY-SIXTH REID LIST

Of the 403 schools still offering computing programs, we were able to determine the first programming language for majors in 393 cases. The language (or languages) used in these courses and the number of occurrences appear in Table 4. It should not surprise anyone to see Java dominate the list, although it is interesting that it is the sole language of instruction or is used in conjunction with another language in just over half the colleges for which languages were determined. C++ remains fairly popular, with 88 colleges using it, 4 colleges teaching it after teaching C, and one using it in some sections of their first programming course. The Ohio State University uses C++ together with the Resolve programming framework. Additionally, 18 colleges use C in their first course without switching to C++.

Python has become much more popular in the past few years, with 47 schools currently using it in all or at least several of their course sections and a few others preparing to adopt it either this year or in 2012. The University of Minnesota begins their course in Scheme before switching over to Python. The remainder of the colleges used a variety of languages, including Visual Basic, Ada, C#, Haskell, and Processing. An examination of community colleges, undergraduate and graduate institutions showed that choice of language did not depend on highest degree offered by the department.

Table 4. The programming language(s) used and the frequency of occurrence

Language	Programs using it
Java	197
C++	82
Python	43
C	18
Java with another language	9
Scheme or Racket	11
Visual Basic	7
Ada	5
C/C++	4
Ada or Python	2
Alice and Java	2
Alice	1
C#	1
C or Matlab	1
C++ or Matlab	1
C++ and Resolve	1
Haskell	1
HTML/JavaScript	1
Processing	1
Processing / Java	1
Python/Java	1
Python or Java	1
Python or C#	1
Python or C# or Matlab	1
Scheme/Python	1
Visual Basic or C#	1

5. QUALITATIVE DISCUSSION

While most of the e-mail replies from faculty simply stated the programming language used in their introductory programming course for majors, there were many replies that provided more information about the decision to use a particular programming language, the previous language used in this course, the language used in subsequent courses and in some cases, the reasons for the choices that various departments had made. While the choices and the reasons behind them varied, there were some trends that could be discerned.

Many Programs Used Different Programming Languages after the Introductory Course

While many schools use the same language throughout much of their program, this is not

always the case; many schools that taught their introductory course in Python taught the subsequent course in another language, most likely Java or C++. But Python followed by Java was not the only sequence of languages that was used. Two schools started their students in C and then switched the following semester to Java; one school used Java followed by C. One program started their students in Haskell or Visual Basic (depending on the course and section) before switching over to Java. Another school began their students in C# before moving to Java while another went through a three semester sequence of Java to C and then to C++. In this last case, the sequence was dictated by the choice of language in later courses; C++ was used in Data Structures while the Operating Systems course used C (no clear reason was given for the use of Java in the introductory course.).

Movement Away From Java

Several instructors spoke of their department moving from Java to another language, most commonly Python, in their introductory course. Various reasons were given for this: Java was too difficult for beginners, "industrial" languages were not necessarily good as instructional languages; Java was too difficult for beginners. One professor said that "[it] seems now that many students feel that programming means searching the class library for a class that implements their program." Another faculty member said that his department "felt that the emphasis on objects was distracting students from fundamentals."

Movement To Java

Three instructors wrote how their departments are adopting Java. All three schools were using either C or C++. None of the replies included a reason for the transition at this point in time.

Different Themes and Language in the Introductory Course

A significant number of schools had different introductory courses or different sections of the same course where different approaches, different themes and/or different languages were used. This was done for several reasons: some departments were experimenting to see if one approach was more successful in attracting students than another approach; some programs designed different introductory courses to meet the needs of different programs. One school used different languages for introductory courses in computer science and information systems because the two programs had different goals for their graduates.

Several colleges used a different programming language in the programming course for non-majors than they used in the course for majors. Replies from two different schools spoke of courses for non-majors in Python and for majors in Java.

Language Should Not Matter

Two different instructors from different colleges spoke about the greater importance of teaching problem solving and algorithmic skills than language skills and how language is used as a tool in teaching the development of algorithms.

Reasons for Choosing a Particular Language

Given the number of complaints about the difficulties that students have with Java, C++, and C, one might wonder why anyone would choose to use any of these languages. Yet several faculty members articulated specific reasons for the choices.

Java's overall popularity was a significant reason for it being the most commonly used language. This very popularity led to its use in the AP Computer Science exams and the large number of textbooks covering introductory programming in Java; these, too, were cited as reasons for adopting Java. One instructor also appreciated the availability of IDEs available for neophyte Java programmers.

One correspondent wrote of his school's decision to use C++ because it facilitated the student's search for internships. While no one gave a similar reason for adopting Java, it is quite possible that it may have been the case, although someone did suggest that there is declining interest in Java in the private sector and that this may be responsible for switching away from Java.

The most common change in programming language that was reported was programs that were switching to Python. The reasons for the change all seemed related to Python's simplicity compared to Java and C++ and the fact that teaching students about objects could be easily postponed.

6. CONCLUSIONS

Because of the smaller number of schools included in the twenty-fifth Reid List, it is difficult to compare it to the current list, which has more than double the schools included. However, some trends were impossible to ignore.

While Java remains the most commonly used language in an introductory programming course, its popularity in the first course is waning. While Java was used by 60% of the schools on the twenty-fifth list, only 50% of colleges on the current list use it in their first course. While this may be somewhat misleading because of the inclusion of so many colleges left off the 2006 list, comments made by responding faculty suggest that the decline is real, even if it may not be as severe as indicated here.

C++ remains surprisingly popular, with no decline from the 2006 list. While the current list and the 2006 list may not offer a reasonable basis for comparison, the anecdotal evidence supplied by the responding faculty suggested that programs are as likely to switch to C++ as to switch from it.

The growth in Python's popularity is undeniable. Not only have more schools reported using it in their first programming course, but responding faculty talk about having adopted it, adopting it either last year or this coming year or how their programs are seriously considering the change.

These results corroborate the finding of Davies, Polack-Wahl and Anewalt (2011), who found that Java remained the most popular programming language in CS1 course, with C++ and Python in second and third place respectively. However, Python was nowhere near as popular in CS2 classes, with both Java and C++ being more popular for CS2 classes than CS1 classes. This suggests that many schools are starting their computing majors in Python and later switching to either Java or C++.

There seems to be many reasons why Python is replacing Java in many programs; complexity of the Java programming language and the difficulties of teaching objects early seem to make programs interested in considering alternative approaches. McIver (2001) points out that Java's modular structure and its requirement that every data item and method be part of a class mandate a certain minimum size for every program, no matter how simple it may be:

```
public class MyFirst {
  public static void main(String[] args)   {
    System.out.println
         ("This is my first Java program.");
  }
}
```

Writing a comparable program in C, C++, or Python will be significantly shorter and does not require teaching as much syntax to beginning programmers. And let's not forget the complexity that is added to this by introducing objects early. This can best be summed up by Elliot Koffman's (2005) comment on the SIGCSE mailing list, "I fear that we have reinvented the 'new math' syndrome and many of us are unaware of it." One faculty respondent said that many of his colleagues felt that the objects early approach was a major contributor to the confusion that their introductory students had. As a result, his department chose to adopt Ada.

It was also clear that there was no need to teach programming courses for non-majors using the same approach or language as in the introductory courses for majors. The survey of Davies et al. (2011) confirms this; the schools surveyed were more likely to use Alice, Python and Visual Basic than Java in courses for non -majors.

The language and approach used in an introductory programming course remains a controversial topic and many departments still have lengthy arguments over their approach to teaching introductory programming classes. Pears et al. published a review of the literature on this subject in 2007, citing one hundred and one papers and many others have been written since then. It is unlikely that there will clear consensus anytime soon.

Reid List 26 will be available at http://home.adelphi.edu/~siegfried/ReidList

7. REFERENCES

Astrachan, A. K., Bruce K., Koffman E., Kölling M. & Reges S. (2005). Resolved: Objects Early Has Failed. *ACM SIGCSE Bulletin* 37(1), 451-452. Quoted in Reges S. (2006). Back to Basics in CS1 and CS2. *ACM SIGCSE Bulletin* 38(1), 293-297.

Brilliant, S. & Wiseman T. (1996). The First Programming Paradigm and Language Dilemma. *ACM SIGCSE Bulletin* 28(1), 338-342.

Bruce, K. B. (2004). Controversy on how to teach CS 1: a discussion on the SIGCSE-members mailing list. *ACM SIGCSE Bulletin*, 36(4), 29-34.

Buck, D. & Stucki D. J. (2000). Design early considered harmful: graduated exposure to complexity and structure based on levels of cognitive development. *ACM SIGCSE Bulletin*, 32(1), 75-79.

Conway, R. W. & Wilcox T. R. (1973). Design and Implementation of a Diagnostic Compiler for PL/I. *Communications of the ACM* 16(3), 169-179.

Davies, S., Polack-Wahl J. A. & Anewalt K. (2011). A Snapshot of Current Practices in Teaching the Introductory Programming Sequence. *ACM SIGCSE Bulletin* 43(1), 625-630.

Decker, R. & Hirschfield S. (1994). The Top 10 Reasons Why Object Oriented Programming Can't Be Taught in CS 1. *ACM SIGCSE Bulletin* 26(1), 51-55.

de Raadt, M., Watson R. & Toleman M. (2004). Introductory Programming: What's Happening Today and Will There Be Any Students to Teach Tomorrow. *Australian Computer Science Communications* 26(5), 277 - 284.

de Raadt, M., Watson R. & Toleman M., (2002). Language Trends in Introductory Programming Courses. The Proceedings of Informing Science, 329 - 337.

Habermann, A. N. (1973). Critical Comments on the Programming Language Pascal. *Acta Informatica* 3(1), 47-57.

Hadjerroult, S. (1998). Java as First Programming Language: A Critical Evaluation. *ACM SIGCSE Bulletin*, 30(2), 43-47.

Holt, R. C. (1973). Teaching the Fatal Disease or Introductory Computer Programming Using PL/I. *ACM SIGPLAN Notices*, 8(5), 8-23.

Johnson, L. F. (1995). C In The First Course Considered Harmful. *Communications of the ACM,* 38(5), 99-101.

Kernighan, B. W. (1981). Why Pascal Is Not My Favorite Programming Language, *Computing Science Technical Report* 100. Murray Hill, NJ: AT&T Bell Laboratories.

King, K. N., (1997) The Case for Java as a First Language. *Proceedings of the 35th Annual Southeast ACM Conference* (Murfreesboro, TN, April, 1997), 124-131. New York: ACM Press. Print.

Madden, M. & Chambers D. (2002). Evaluation of Student Attitudes to Learning the Java Language. *Proceedings of Conference on the Principles and Practice of Programming in Java*, (Trinity College Dublin, June 2002), pp. 125-130. New York: ACM Press. Print.

McConnell, J. J., & Burhans D. T. (2002). The Evolution of CS1 Textbooks. *32nd Annual Frontiers in Education (FIE'02)*, (Boston, MA, USA, November 6-9, 2002), vol.1 p. T4G1-6.

McIver, L. (2001). Syntactic and Semantic Issues in Introductory Programming Education. *Ph.D. Dissertation, Monash University*. Received via e-mail from author, June 30, 2007.

Pears, A., Seidman S., Malmi L., Mannila L., Adams E., Bennedsen J., Devlin M. & Paterson J. (2007). A Survey of Literature on the Teaching of Introductory Programming. ACM SIGCSE Bulletin 39(4), 204-223.

Reges, S. (2006). Back to Basics in CS1 and CS2. ACM SIGCSE Bulletin 38(1), 293-297.

Reid, R. J. (1992). First Course Language For Computer Science Majors. Retrieved from http://www.csee.wvu.edu/~vanscoy/REID06.HTM on July 12, 2011.

Van Scoy, F. (2006). The Reid List 25. Retrieved from http://groups.google.com/group/comp.edu/browse_thread/thread/4f00b5f437ce261a/3267514419052033?q=Reid+List#3267514419052033 on July 15, 2011.

Wirth, N. (1971). The Programming Language Pascal. *Acta Informatica* 1(1), 35-63.