J. Appl. Environ. Biol. Sci., 4(7S)311-317, 2014

ISSN: 2090-4274
Journal of Applied Environmental and Biological Sciences
www.textroad.com

Choice of Pedagogical Approaches towards First Programming Languages

Muhammad Shoaib Farooq[1,2], Sher Afzal Khan[2], Farooq Ahmed[3], Saeed Islam[2], *Adnan Abid[1]

[1]Department of Computer Science, University of Management and Technology, Lahore, Pakistan
[2]Abdul Wali Khan University, Mardan, Pakistan
[3]Faculty of Information Technology, University of Central Punjab Lahore, Pakistan

Received: September 1, 2014
Accepted: November 13, 2014

ABSTRACT

The development of programming language has long not been driven by pedagogical requirements of novice programmers. Language designers are normally domain experts and they generally neglect problems faced by novice programmers in the language design process. In this research, we highlight different pedagogical approaches used by many schools of thought. We highlight the pros and cons of these approaches, and find imperative first object later approach is most suitable for novice programmers.

KEYWORDS: First Programming Language, Object Oriented Programming, Object Fist Approach. Imperative First Approach

1 INTRODUCTION

As the discipline of Computer Science emerged the importance of computer programming became even more pertinent, as it is considered that learning computer programming is an essential skill that must be mastered by all the students of this domain [27]. Most of the introductory CS courses primarily focused on problem solving skills throughout the history. In this context, there has always been an ongoing debate over the selection of a first programming language for introductory programming course [21][31][11][13]. The language chosen for the introductory course in computer programming is generally referred to as a First Programming Language (FPL).

The first programming language is regarded as a crucial factor in a student's subsequent progress in the discipline. As discussed by Howell et al. [4] and Stephen et al. [7], the purpose of first programming language is to provide conceptual knowledge for the 'understanding of programming constructs' and strategic knowledge to describe the 'abilities of problem decomposition and specification'. Furthermore, in terms of teaching, the first programming language should be easy to learn with a small learning curve. This will help the students to balance the learning while improving their problem solving skills [2][8].

For introductory programming courses selection of programming language normally has long been depends on faculty consensus, evaluation, and industrial demand. This process has increasingly gone unmanageable when faculty, students, language options, industrial demands, and disciplines in education grows.

Similarly, as the languages are evolving the quantity of stuff is increasing, but the credit hours in courses are not. So, the students are overburdened, especially if they have no prior programming experience, which ultimately results into student abrasion. Therefore, the selection of an appropriate programming language for an introductory course in computer programming requires serious attention, as Bjarne Stroustrup states, "the choice of a first language is always controversial" [31][37].

In this work we intend to focus on finding the criterion for choosing an appropriate first programming language for the disciplines of Computer Science and Information Technology. We intend to analyze the existing work and figure out the dimensions in which this problem has been addressed. We argue that this will help us composing a fairly comprehensive set of multi-dimensional criterion for the evaluation of an appropriate first programming language. This inherently triggers the question for the selection of most suitable introductory programming language from the existing languages. Therefore, we look forward to

* **Corresponding Author:** Adnan Abid, Department of Computer Science, University of Management and Technology, Lahore, Pakistan.

incorporate a quantitative scoring function to measure the strength of any language as a healthy FPL, which in turn, will help us choosing most suitable existing programming language. We also plan to improve the most suitable existing FPL so as to increase its conformance to the already defined criterion. Lastly, we also plan to build pedagogical tools for students and teachers, which will increase the productivity of a novice programmer, and may help the instructors in teaching activities.

2 RELATED WORK

First Programming language or an Introductory Programming Language has been discussed in the literature in different dimensions [17]. Some researchers have proposed new programming environments for learning computer programming. These environments are focused on learning computer programming interactively. Karel [30], Alice [23][16], BlueJ[15] some examples of such environments. This facet of research work on FPL addresses the pedagogical aspects related to the FPL, where the discussion mainly revolves around teaching methodologies [41]. The main teaching approaches which have been discussed in the literature are object first, object first imperative later, and imperative first object later [21][23][27][14].

On the other hand, many researchers have proposed methods and criteria for evaluating programming languages as an appropriate FPL [6][32][[6][39]][40] . Some have focused on comparative analysis of the suitability of existing languages as FPL. Lastly, another branch deals with the investigation of the suitability of an existing programming language as a first programming language [12][5][18][19][39][40].

3 Pedagogical approaches

Many efforts have been made to address the pedagogical issues related to introductory programming languages [17]. These approaches relate languages for teaching programming using, Mini language approach, top-down or object-first approach, object first imperative later, bottom-up or imperative first-object later approach, functional first and pseudo language approaches [23][17][16]. Figure 1 shows pedagogical approaches followed and approved by different universities and faculty members.

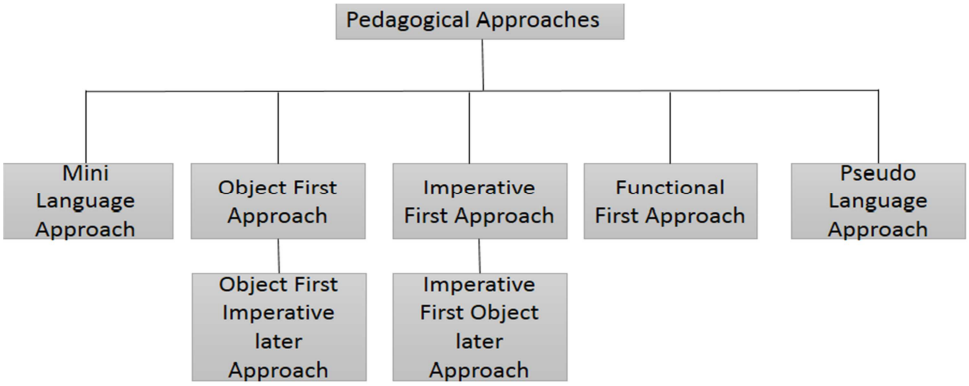

Figure 1 pedagogical approaches of first Programming languages

3.1 Mini Language Approach

In an effort to help novices to learn programming several Mini languages have been developed [22]. Here Mini language term is used for languages that use some actor (turtle or robot) which performs different operations in a micro-world using some predefined commands. The main purpose of such approaches is to help the novice programmers learn programming with ease[28]. Logo (the turtle) [10] was the first Mini Language.

There are a number of mini-languages which were directly encouraged by Karel and use several of its features: Martino [18] and Marta [23] in Italy, Darel[24] in Australia and Karel-3D [25] in Slovakia.

Strengths:
Learn programming using a visual interface with actor (Robot), which mainly help the students to learn the use of functions, and program flow (conditions, loops etc.).
Weaknesses:
All these languages have a common drawback that they do not have the concepts of variables and parameter passing. Thus, they fail to impart actual programming concepts to the students[26].

3.2 Object First Approach

Another approach related to teach first programming language is called the object first approach which is also referred to as top-down approach in which major focus is on modularization, encapsulation, reusability, recursion, creating, manipulating objects and classes [22][23][25]. Object first approaches are based on systematic software engineering.

The object First approach recommended by researchers at, Mærsk Institute at the University of Southern Denmark and Monash University Melbourne, Australia. They developed BlueJ environment for visualizing objects and classes for learning object oriented concepts. The BlueJ environment helps the students to create interesting and fun applications, which enables them to master the basic concepts of the object-oriented approach in early stages of their course-work.

Strengths: Certainly object first approach helps the students to understand the strong concepts of reusability, encapsulation, information hiding, association, inheritance, polymorphism related to object oriented paradigm in early stages.

Weaknesses: Ragonis and Ben-Ari [31] used BlueJ to investigate the learning of object-oriented programming by high school students over the course of an academic year. They found that, while BlueJ helped students understand object-oriented programming concepts, it did not help them understand the overall execution (or "flow") of a program. These finding suggest that the students were unable to trace the program's source code, an ability required to understand its execution. There is no concept of main (client program) in BlueJ, and the students trained through BlueJ did not even know how to run program using main.

3.3 Object First/ Imperative Later Approach

Dann et al. [25] introduced another variant of the object first approach, Alice (a character in Alice's Adventures in Wonderland). It supports either an "objects-first" or an "objects-early" and imperative approach for teaching Object-oriented programming. Alice [29] is a 3D environment in which students can build virtual worlds without writing source code using the drag and drop options. This results into low syntax errors and enhances programming skills without writing any code. Alice users also suffer the same experiences as reported by users of BlueJ. Powers, Ecott, and Hirshfield[30] apply Alice approach to teach programming suggested in "Learning to Program with Alice" by [31] in which they deferred to introduce variables late in Alice course.

Strength: Alice advocates believe that blending traditional problem-solving techniques with Hollywood-style storyboarding will enable students to readily create compelling object-based programs, thereby improving motivation, reducing attrition, and smoothing the road to learning professional object-oriented languages such as C++ or Java [1][25].

Weakness: The major drawback is that students are unable to grasp the concepts and usage of variables which makes difficult to learn the parameter passing concepts [16][18].

3.4 Imperative First/ Object Later Approach

The most widely used approach for teaching computer programming is imperative first/ object later approach. This approach involves object oriented, object based, and structural programming languages. Most of the universities which follow this approach span the computer programming course in two semesters. Here, in the first semester the students are taught imperative concepts (such as variables,

selection, assignment, expression, loops and functions) which are followed by programming using object oriented paradigm, in the next semester.

Strengths:

This is most widely used classical approach in many universities due to following advantageous features:

i) Compatible with von Neumann architecture.

ii) Its more natural to solve problem with procedural instructions (i.e. loops, selection, assignment, expressions and variables) and decompose a problem into set of functions/procedures

iii) Solve problems with high level abstraction called objects.

iv) Most of third and fourth generation High level Languages support these concepts.

Weaknesses:

i) It introduces the concepts of reusability, modularization, and recursion in later stages.

ii) Highly depends on general purpose programming language.

iii) Generally, the teachers focus more on teaching the syntax of the language in the first semester, instead of teaching the problem solving skills.

3.5 Functional First Approach

The functional-first approach introduces formal computational concepts with a simple functional syntax, such as Scheme [32], SML [33], Haskell [34]. The approach was named "functional first" in the ACM 2001 curriculum [9]. This approach requires the knowledge of lambda calculus. The advocates of this approach claim that such languages help the students implementing computationally non-trivial problems from the first day.

Strengths:

i) The syntax of functional languages is closely related to the core issues in computability which will enhance problem solving skills.

ii) Several important concepts such as data structures, map, sets, sequence, recursion, and functions, formally appears in this domain and covered a lot earlier in the curriculum.

Weaknesses:

i) The students may respond hesitantly to learning a language that is external of the conventional programming languages.

ii) This approach naturally requires students to think much formally at an early phase as compared to the traditional programming languages.

3.6 Pseudo Language Approach

Another approach is pseudo language approach defined by educators in CS Community. Pseudo languages are typically subsets of existing mainstream programming languages with some extra features, in order to teach the basic programming concepts [27]. The idea of a pseudo language is to create code with as simple syntax as possible. So a student can pay more attention on learning programming concepts and problem solving skills instead of learning typical syntax.

MiniJava falls under the category of pseudo language. MiniJava removes inner classes, do-while, Continue, break, and switch statement. It reduces larger set libraries from 700 classes to 17 only. Console window expressions like Lisp, console class for input/output, program and graphics program classes and primitive type as objects introduced as enhancements in MiniJava.

Strengths:

i) Learning a subset of the language allows a somewhat moderate learning curve, providing time for the learner to absorb and build knowledge incrementally.

Weaknesses:

i) This approach has not been so popular for the reason that there are some extra features which do not belong to the core language. This requires a new compiler implementation which deviates from the main objective of learning the language.

4 Language Evaluation and Suitability

The selection of a programming language for pedagogical purposes is often viewed as a tedious job because there is no well defined established technique for performing the evaluation. However, the choice of a programming language for introductory computer science course has severe educational repercussions [35]. With the increase in the number of programming languages, the number of faculties, and students, the selection of most suitable introductory programming language is becoming increasingly cumbersome [36].

The problem of language selection and suitability has been discussed in different dimensions, which involve the cross comparison of existing languages in terms of their suitability as introductory programming languages [36][12], whereas, some people have proposed evaluation methods for the suitability of a programming language [13][5][20][21].

4.1 Formal Evaluation Creation/processes

Formal evaluation programs for the assessment of programming languages are few and far between, and most evidence gathered is anecdotal in nature. Some approaches have been proposed to evaluate the first programming language, for instance, Parker et al. compiled a list of criteria for introductory programming courses at universities [15]. However, this criterion has not been discussed with technical details of the involved measures, which can be useful for scoring purpose.

Gupta [14] discussed requirements for programming languages for beginners. He has done a thorough requirement analysis for an appropriate first programming language. In his work, he presents technical and environmental requirements for an appropriate programming language. However, there is no formal assessment mechanism devised for the evaluation of the language for its suitability as an introductory first programming language.

5 Conclusion

As discussed in Section 3, we have conducted a preliminary survey of the literature pertaining to the pedagogical approaches and language evaluation and assessment. In our opinion imperative first and object latter approach is good for novice with some healthy programming tool. We can start programming using this approach from school level to university level. The functional first approach is not recommended because it requires too much mathematical background from novices, which is not possible in the early stage of novices who have not enough mathematical background. Mini language approach is good for starting stage, but it lacks proper conceptual programming, transition to other languages are very tedious after using this approach.

REFERENCES

[1] Gosling J., Java: an Overview, *(White Paper)* Retrieved April 2014 from http://www.cs.dartmouth.edu/~mckeeman/cs118/references/OriginalJavaWhitepaper.pdf.

[2] Grout, J. C., Strader, R. G., Hanks, J. B. Essential C++, *ACM SIGCSE Bulletin*, 1996; 28(2); 3-14.

[3] Hoare, C. A. R.The Emperor's Old Clothes*, ACM Turing Award Lecture, Association for Computing Machinery.*1980

[4] Howell, K. First computer languages**,** *Journal of Computing Sciences in Colleges.* 18(4); 317-331; 2003.

[5] Jacquot, J. P. Which use for Java in introductory courses?, *In Proceedings of the inaugural conference on the Principles and Practice of programming, 2002, and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002;* 119-124.

[6] Kölling, M. The Problem of teaching Object Oriented Programming Part 2: Environments, *Journal of Object Oriented Programming*, 1999; 11(9), 6-12.

[7] Stephenson C., West T. Language: Choice and Key Concepts in Introductory Computer Science Courses, *Journal of Research on Computing Education*, 1998; 31(1), p89.

[8] Wirth N., The Programming Language Pascal (Revised Report), *Berichte der Fachgruppe Computer-Wissenschaften*1972.

[9] Joint Task Force on Computing Curricula. Computing Curricula 2001 Computer Science, *Journal of Educational Resources in Computing (JERIC)*, 1(3), 2001.

[10] Pattis, R. E Karel - the robot, a gentle introduction to the art of programming. *Wiley, London*1981.

[11] Chandra S. S., and Chandra K. A Comparison of Java and C, *Journal of Computing Sciences in Colleges*, 2005; 20(3), 238–254.

[12] Holtz, N. M., and Rasdorf, W. J. An evaluation of programming languages and language features for engineering software development, *Engineering with Computers*, 1998; 3(4); 183–199.

[13] Hadjerrouit, S. Java as first programming language: a critical evaluation. *ACM SIGCSE Bulletin*, 1998; 30(2), 43–47.

[14] Gupta, D. What is a good first programming language? *Crossroads*, 2004; 10(4),7–7.

[15] Parker, K. R., Ottaway, T. A., Chao, J. T., Chang J. A Formal Language Selection Process for Introductory Programming Courses, *Journal of Information Technology Education*, 2006; 5:133–151.

[16] Kunkle, W. M., The Impact of Different Teaching Approaches and Languages on Student Learning of Introductory Programming Concepts, *(Doctoral dissertation*), Retrieved March, 2014 from http://hdl.handle.net/1860/3380

[17] BlueJ. Retrieved 04 May, 2014, from http://www.bluej.org/

[18] Cooper, S., Dann, W., and Pausch, R. Using animated 3d graphics to prepare novices for CS1. *Computer Science Education*, 2003; 13(1):3–30.

[19] Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J. *A survey of literature on the teaching of introductory programming,ACM SIGCSE Bulletin, 2007;* 39(4), 204-223.

[20] Weiderman, N., and Coulter, V. W. Using Ada as an introductory programming language, *J. Pascal Ada Modula- 1987; 2* 6(1), 10-34.

[21] Bell, D.Visual BASIC.Net as a first language: an evaluation. *ACM SIGCSE Bulletin,*2002; 34(4); 107-108.

[22] Olimpo, G. The Robot Brothers: An environment for learning parallel programming oriented to computer education, *Computers and Education*1988*;* 12(1), 113-118.

[23] Calabrese, E. Marta - the "Intelligent Turtle". *Proceedings of Second European Logo Conference, EUROLOGO'89,*1989; 111-127.

[24] Kay, J. Tyler P. A microworld for developing learning design strategies, *Computer Science Education* 3(1); 111-122; 1993.

[25] Hvorecky, J. Karel the Robot for PC, *Proceedings of East-West Conference on Emerging Computer Technologies in Education,*1992; 157-160.

[26] Clancy, M. Misconceptions and attitudes that interfere with learning to program, *Computer Science Education Research,*2004*;* 85-100.

[27] Kelleher, C., Pausch, RLowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys, 2005; 37(2), 83-137.

[28] Ragonis, N., and Ben-Ari, M. A long-term investigation of the comprehension of OOP concepts by novices, *Computer Science Education*, 2005; 15(3).

[29] Alice v3.0. Pittsburgh: Carnegie Mellon University. Retrieved May, 2014, from http://www.alice.org/.

[30] Powers, K., Ecott, S., and Hirshfield, L. M. Through the looking glass: Teaching CS0 with Alice. *In Proceedings of the 38th SIGCSE technical symposium on Computer science education*, New York: ACM Press, 2007; 213-217.

[31] Dann, W., Cooper, S., and Pausch, R. Learning to Program with Alice. Upper Saddle River: Pearson Prentice Hall 2006

[32] Abelson, H., Sussman, G. J., and Sussman, J.Structure and interpre-tation of computer programs, MIT Press 1985.

[33] Milner, R., Tofte, M., Harper, R., and MacQueen, D. The Definition of Standard ML – Revised. MIT Press 1997.

[34] Richards, H., Haskell, Jr.The Craft of Functional Programming by Simon Thompson, Addison-Wesley, (1998); 633-637.

[35] Schneider, G.M. The introductory programming course in computer science: Ten principles, *ACM SIGCSE Bulletin*, (1978);10 (1); 107-114.

[36] Smolarski, D.C. A first course in computer science: Languages and goals, *Teaching Mathematics and Computer Science*, 2010; 1 (1), 137-152.

[37] Bjarne Stroustrup (2009) Programming in an undergraduate CS curriculum, In *Proceedings of the 14th Western Canadian Conference on Computing Education* (WCCCE '09), ACM, New York 2009;82-89.

[38] Phipps, G., Comparing observed bug and productivity rates for Java and C++, *Software-Practice and Experience*, 1999; 29(4); 345–358.

[39] Farooq, Muhammad Shoaib, Sher Afzal Khan, Farooq Ahmad, Saeed Islam, and Adnan Abid. "An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages". (2014) PloS one 9(2); e88941.

[40] Muhammad Shoaib Farooq, Sher Afzal Khan, Adnan Abid "A Framework for the Assessment of a First Programming Language", Journal of Basic and Applied Scientific Research, (2012); 2(8); 8144-8149.

[41] Muhammad Shoaib Farooq, Adnan Abid, Sher Afzal Khan, Muhammad AzharNaeem, Amjad Farooq, Kamran Abid, "A Qualitative Framework for Introducing Programming Language at High School", Journal of Quality and Technology Management, Punjab University, Pakistan. 2012; 8(2).