ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

A STUDY OF DIFFERENT PLATFORMS AND PROGRAMMING LANGUAGES FOR INDUSTRY APART FROM THE EDUCATION SECTOR

Dr E. SriDevi[1],
Asst. Professor, Department of C.S.E, Koneru Lakshmaiah Education Foundation, Guntur, A.P, India – 522502.
V.PremaLatha[2]
Assoc. Professor, Department of C.S.E, Koneru Lakshmaiah Education Foundation, Guntur, A.P, India – 522502.

Abstract:
In the digital age, computer programming is both a fundamental literacy and the universal language of our planet. Without a doubt, all students benefit from learning computer programming at a young age, if not for their daily lives. Adolescent students benefit from learning programming reimbursement in terms of improved communication, thought processes, and thinking abilities. These advantages can help young people acquire, build, and enhance the skills necessary for the twenty-first century. Making computer programming appealing and engaging for students in elementary, secondary, and university settings is one of the major concerns facing scientists and enlightening practitioners in the field. Using a variety of educational software programmes could help with this problem. There are numerous effective instances of educational software being utilised in classrooms. The focus of this paper is on the value of educational software tools for computer programming fundamentals instruction and learning. A detailed explanation is provided for some of the most important skin tones of these tools, such as object-oriented programming, instantaneous criticism, and visuals. The author offers a number of the instructional software tools that are mentioned above that can enhance computer programming instruction, teaching, and industry expectations in the education sector.

"Keywords: Computer Programming, Industry Need, Education Sector, Enlightening Software, Reimbursement Programming."

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

INTRODUCTION:

The background of computer languages as options for higher education institutions and industry use in programming courses. The study examines events in two developed nations and identifies themes that might be present in other urbanised countries' language collection histories. History demonstrates a recurring set of issues for individuals engaged in language selection.This study demonstrates that decision-makers in the selection process can draw insights from history.

A common way to describe the history of computing is in terms of important advances in hardware. Australia and the United States both contributed early to computing. Many people believe that Computer Corporation invented programmable computers. The government science organisation, the fourth programmable computer in the world, which executed its first test programme in 1949, is credited with starting the history of programmable computers in many nations. Produced by the government science organisation, this computer was operational at the University of Melbourne into the 1960s and is still in complete form at the Melbourne Museum of Victoria. Australia entered the computing world earlier than the United States, which makes a comparison with them interesting.

2. TIMES GONE BY LANGUAGE DEVELOPMENT

In computer programming, the first steps are very important. Prior to progressively moving on to C, older generations were taught languages like Basic, Fortran, and Pascal. However,These computer languages require an understanding of well-represented expression through logical and mathematical formulations. When young children enter a traditional .They become disinterested and demoralised when learning programming language because one of the main challenges Understanding a programming language's syntax is a necessary part of learning the language. As a result, programming has dealt with older elementary school pupils. These days, computers Languages such as Python, Delphi, C++, C#, and Java also need a comparable degree of prior awareness. The advent of new visual programming languages, such as Visual Basic and Visual C, was predicted to simplify the programming process. Unfortunately, it ended up that inexperienced programmers should not use them. The following is a list of some notable

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

programming languages, with the development year indicated in parenthesis: 1957 Fortran, Lisp

Basic 1964, Logo 1968, Pascal 1970, C 1972, C++ 1980, Python 1991, and Visual Scratch 2003, Delphi 1995, JavaScript 1995, Java 1995, and Basic 1991. Allowing for the fact that programming is one of the most important 21st-century skills, the primary issue is how to allow children to begin programming before they are able to read. It's commonly believed that learning programming and foreign languages at a young age is beneficial. To put it another way, "the earlier, the better" applies to learning programming; there is no age too young. It is evident that learning while spending a lot of time in front of a computer is not the best option for younger students. Quick interactive multimedia apps have been shown to be a viable substitute that makes learning programming simpler. This shift involves the use of instructional games and illustration programming languages.

3. TRENDS IN LANGUAGE ASSORTMENT

Unlike previous programming languages, which prioritised arithmetic computation, Logo was designed to manipulate words and sentences in language. Symbol is a general-purpose tool that can be used in a variety of ways, much like any programming language. Symbol programming is comprehensible at various levels of complexity. When the programme first came out, grades 6 through 8 used it, and its animation and turtle graphics were its main selling points. The teachers' daily work increased their confidence in utilising the programme and gave them a deeper understanding of symbols. This modifies the idea that symbols are a "child's language."

Higher education uses symbols more frequently, and an increasing number of educators think Logo is a useful tool for their own work. Teachers of informatics sometimes work with students to develop small-scale learning initiatives called Logo-projects.

Additionally, he noticed a shift away from creating a stand-alone system by writing an entire application "from scratch" in a single language and towards the use of general-purpose languages as the integrating medium for the extensive functionality provided by database packages, web-based services, GUIs, and a plethora of other COTS and customised products that interface through an application programme interface (API). Simultaneously, "contextual

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

concerns" pertaining to security, privacy, robustness, safety, etc., consistently rule all applications (p. 1027).

Roberts (2004a) noted another trend: more universities were adopting Java as the programming language for their introductory courses as a result of the object-oriented paradigm's increasing popularity and the College Board's decision to switch the Advanced Placement Computer Science programme to Java. He went on to note (2004b) that there were two more issues where sharp rises had an adverse effect on pedagogy: (1) students now need to learn a greater number of programming details, and (2) the languages, libraries, and resources used in beginning courses are changing at a faster rate than in the past. The use of scripting languages to teach programming concepts is a trend that Gee, Wills, and Cooke (2005) highlighted as being increasingly visible (and contentious) because they offer "not only a proper programming environment but also an instant link into the formation of active web pages." In their analysis of numerous studies, many of which are already mentioned, Parker et al. (2006a, 2006b) presented a set of criteria to be used when choosing a computer programming language for an introductory programming course. They also created an instrument that enables the weighting of each of those selection criteria to indicate their relative importance in the selection process.

4. LANGUAGE SELECTION STUDIES

A list of the elements that inflated the alternative of a programming language for a beginning course at many universities is skillfully presented. In a recent study, an introductory programming language for IT students is carefully examined. A follow-up, new study looks at more than 60 publications about language choice in higher education. With a few notable exceptions, the selection of programming languages in many university curricula is nearly identical. Languages have been created over time to address issues. For the purpose of making teaching algorithms easier, other languages have been created. This has given rise to two occasionally at odds schools of thought regarding the languages that should be taught in university courses: pick a language that best facilitates students' concept development, or favour a language that is typically or is anticipated to be widely used in industry.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

4.1 Categorizing Languages

As indicated in Table 1, we categories languages into classes according to a number of linguistic characteristics that are believed to affect language quality [11, 12, 15,]. Whether the project is written in a functional, procedural, or scripting language is indicated by the programming paradigm. Whether the project is statically or dynamically typed is indicated by the Compile Class.

Table 1: Different Types of Language Classes

Language Classes	Categories	Languages
Programming Paradigm	Procedural	C, C++, C#, Objective-C, Java, Go
	Scripting	CoffeeScript, JavaScript, Python, Perl, Php, Ruby
	Functional	Clojure, Erlang, Haskell, Scala
Compilation Class	Static	C, C++, C#, Objective-C, Java, Go, Haskell, Scala
	Dynamic	CoffeeScript, JavaScript, Python, Perl, Php, Ruby, Clojure, Erlang
Type Class	Strong	C#, Java, Go, Python, Ruby, Clojure, Erlang, Haskell, Scala
	Weak	C, C++, Objective-C, CoffeeScript, JavaScript, Perl, Php
Memory Class	Managed	Others
	Unmanaged	C, C++, Objective-C

4.2 Identifying Project Domain

We employ a combination of automated and manual techniques to categorise the examined projects into distinct domains according to their features and functionalities. Readme files and project descriptions are included with every project on GitHub, outlining its features.

First, we applied the well-known topic analysis algorithm Latent Dirichlet Allocation (LDA) [4] to the text outlining the project's features.

- LDA finds a set of topics from a set of documents where each topic is expressed as the likelihood of producing a distinct word.
- LDA additionally calculates the likelihood of assigning a given document to every topic.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

Table 2: Characteristics of Domains

Domain Name	Domain Characteristics	Example Projects	Total Proj
Application (APP)	end user programs.	bitcoin, macvim	120
Database (DB)	sql and nosql databases	mysql, mongodb	43
CodeAnalyzer (CA)	compiler, parser interpreter etc.	ruby, php-src	88
Middleware (MW)	Operating Systems, Virtual Machine, etc.	linux, memcached	48
Library (LIB)	APIs, libraries etc.	androidApis, opencv	175
Framework (FW)	SDKs, plugins	ios sdk, coffeekup	206
Other (OTH)	-	Arduino, autoenv	49

5. EXPECTATION OF INDUSTRY

Industry getting, or the use of a language in commerce and industry, is the term used to describe the market penetration of a specific language in industry. Often called "industrial relevance," this can be evaluated by counting the number of current and predictable positions as well as the current and projected usage. Stephenson asserts that 23.5% of the schools in his study support his claim that this feature is under the most pressure when it comes to language selection. As evidenced by the previous use of ALGOL and Pascal, Lee and Stroud point out that real-world suitability was once a factor with little weight, but that attitude does appear to be changing. They observe that one important factor for their students is their ability to resume in a language that is widely accepted in technology. According to a 2001 survey conducted across all universities, the primary determinant of introductory language choice was perceived industry demand. King acknowledges that many language choices are based on current appeal or the likelihood of future popularity. However, he points out that picking popular languages has several real-world advantages, such as encouraging students to study a language they are familiar with and know is in demand and providing a wide range of books and language resources for those languages.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

5.1 Identifying top languages.

The number of open source GitHub projects created in each language is the first step in determining which languages are the most popular on GitHub. The top languages with the most projects are then selected. Assigning a single language to a project can be challenging, though, since multiple languages are frequently used to develop a project. A GitHub project repository's language distribution can be measured using GitHub Linguist [9].GitHub Linguist counts the number of source files with various extensions because a project's source file extensions can be used to identify languages.

The primary language of the project is designated as the one with the greatest number of source files. This data is stored in GitHub Archive. We group projects according to the main language used in each. Next, as indicated in Table 1, we choose the top languages with the greatest number of projects for additional examination.

Table 3: Top three projects in each language

Language	Projects
C	linux, git, php-src
C++	node-webkit, phantomjs, mongo
C#	SignalR, SparkleShare, ServiceStack
Objective-C	AFNetworking, GPUImage, RestKit
Go	docker, lime, websocketd
Java	storm, elasticsearch, ActionBarSherlock
CoffeeScript	coffee-script, hubot, brunch
JavaScript	bootstrap, jquery, node
TypeScript	bitcoin, litecoin, qBittorrent
Ruby	rails, gitlabhq, homebrew
Php	laravel, CodeIgniter, symfony
Python	flask, django, reddit
Perl	gitolite, showdown, rails-dev-box
Clojure	LightTable, leiningen, clojurescript
Erlang	ChicagoBoss, cowboy, couchdb
Haskell	pandoc, yesod, git-annex
Scala	Play20, spark, scala

6. CONCLUSION

Teachers, scientists, experts, and educational institutions should be aware of current scientific and technological advancements. The ability to use technology to access, administer, incorporate, and evaluate information, create new knowledge, and effectively communicate with others are all examples of 21st century skills that they should value and promote. These skills also include thinking and problem-solving abilities, communication and self-directed learning abilities, and information and communication skills. In addition, there are other important aspects to take into account when choosing a programming language. I.e. Though practical and educational concerns remain paramount, consideration of other factors

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

that influence the selection process must also be present. A number of factors have been the subject of recent studies. The bottom line is that when choosing a language, academics must carefully consider what is in the best interests of the students and take all factors into account.

REFERENCES

[1] Ad Hoc AP CS Committee (2000). "Round 2: Potential Principles governing language selection for CS1-CS2." http://www.cs.grinnell.edu/~walker/sigcse-ap/99-00- principles.htm

[2] Allison, I., Ortin, P., and Powell, H. (2002). "A virtual learning environment for introductory programming." Proceedings of the 3rd Annual conference of the Learning and Teaching Support Network Centre for Information and Computer Sciences, Loughborough, UK.

[3] Bergin, T.J. and Gibson, R.G. (1996) History of Programming Languages-II. USA: ACM Press.

[4] Boom, H. J. and de Jong, E. (1980). "A critical comparison of several programming language implementations." Software: Practice and Experience 10, 435–473.

[5] de Raadt, M., Watson, R., and Toleman, M. (2003a). "Introductory programming languages at Australian universities at the beginning of the twenty first century." Journal of Research and Practice in Information Technology 35(3): 163-167.

[6] de Raadt, M. Watson, R., and Toleman, M. (2003b). "Language tug-Of-war: Industry demand and academic choice." Australasian Computing Education Conference (ACE2003), Adelaide, Australia., Australian Computer Society, Inc.

[7] Dijkstra, E. W. (1972). "The humble programmer." Communications of the ACM 15(10), 859– 866.

[8] Emigh, K L. (2001). "The impact of new programming languages on university curriculum." Proceedings of ISECON 2001, Cincinnati, Ohio, 18, 1146-1151. Retrieved July 10, 2005 from http://isedj.org/isecon/2001/16c/ ISECON.2001.Emigh.pdf

[9] Friedman, F. L. and Koffman, E. B. (1976). "Some pedagogic considerations in teaching elementary programming using structured FORTRAN." Proceedings of the ACM SIGCSESIGCUE Technical Symposium on Computer Science and Education, 1-10.

[10] Furugori, T. and Jalics, P. (1977). "First course in computer science, a small survey." ACM SIGCSE Bulletin, 9 (1), 119-122.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

[11]Gee, Q. H., Wills, G. and Cooke, E. (2005). "A first programming language for IT students." Proceedings of the 6th Annual Conference of the Learning and Teaching Support Network Centre for Information and Computer Sciences, York, UK.

[12] Gottliebsen, C. (1999). Computer market results 1999. C. Gottliebsen. Bayswater, GIMA [13] Gottliebsen, C. (2001). Icon index trend report 2001. Icon index Trend Report. B. Youston. Bayswater.

[14] Howatt, J. W. (1995). "A project-based approach to programming language evaluation." ACM SIGPLAN No-tices, 30 (7), 37-40. http://academic.luther.edu/~howaja01/v/lang.pdf

[15] Howland, J.E. (1997). "It's all in the language: yet another look at the choice of programming language for teaching computer science." Journal of Computing in Small Colleges, 12(4); 58-74, http://www.cs.trinity.edu/~jhowland/ccsc97/ccsc97/

[16] Jenkins, T. (2001). "The motivation of students of programming." ACM SIGCSE Bulletin , Proceedings of the 6th annual conference on Irmovation and technology in computer science education ITiCSE '01 33(3).

[17] Jenkins, T. (2002). "On the difficulty of learning to program." Proceedings of the 3rd annual conference of the Learning and Teaching Support Network Centre for Information and Computing Science, Loughborough, UK.

[18] Johnson, L.F. (1995). "C in the first course considered harmful." Communications of the ACM 38 (5): 99-101.

[19] Keet, E. E. (2004). "A personal recollection of software's early days (1960-1979): Part 1." IEEE Annals of the History of Computing (October-December).

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

[21] King, K.N. (1992). "The evolution of the programming languages course." Proceedings of the Twenty-Third SIGCSE Technical Symposium on Computer Science Education, Kansas City, Missouri, pp. 213-219.

[22] Kölling M. and Koch, B. (1995). "Requirements for a first year object-oriented teaching language." ACM SIGCSE Bullet in , Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education 27(1). Language History - A Tale of Two Countries 151

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

[23] Langley, N. (2004). "COBOL integrates with Java and .NET." Computer Weekly. http://www.computerweekly.com/articles/article.asp?liArticleID=133085

[24] Lee, P.A., and Stroud, R.J. (1996). "C++ as an introductory programming language." In M. Woodman (Ed.), Programming Language Choice: Practice and Experience, London: International Thomson Computer Press, pp. 63-82. http://www.cs.ncl.ac.uk /oId/publications/books/apprentice/lnstructorsManual/C-H-_Choice.html

[25] Levy, S. P. (1995). "Computer Language Usage In CS 1: Survey Results." SIGCSE 27(3): 21- 26.

[26] Luker, P. (1989). "Academic staff development in universities with special reference to small group teaching." (Unpublished PhD Thesis), University of Nottingham.

[27] McCauley, R. and Manaris, B., (1998). "Computer science programs: what do they look like?" Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education, February, pp. 15-19.

[28] Mclver, L. and Conway, D.M. (1996). "Seven deadly sins of introductory programming language design." Proceedings of Software Engineering: Education and Practice (SE:E&P'96), Dunedin, NZ, pp.309-316.

[29] Merritt, S. M. (1980). "On the importance of teaching Pascal in the IS curriculum." ACM SIGCSE Bulletin , Proceedings of the eleventh IGCSE technical symposium on Computer science education SIGCSE '80 12(1)

[30] Perlis, A. J. (1981). The American Side of the Development of Algol. In R. L. Wexelblat (Ed.), History of programming languages I (pp. 25-74): ACM.

[31] Parker, K.R., Ottaway, T.A. and Chao, J.T. (2006a). "Criteria for the selection of a programming language for introductory courses." International Journal of Knowledge and Learning 2 (1/2)119-139.

[32] Parker, K.R., Chao, J.T., Ottaway, T.A., and Chang, J .(2006b). "A formal language selection process for introductory programming courses. Journal of Information Technology Education, 5, 133-151.

[33] Riehle, R. (2003). "SEPR and programming language selection." CrossTalk – The Journal of Defense Software Engineering 16(2): 13-17, http://vkfww.stsc.hill.af mil/crosstalk/2003/02/Riehle.html

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

[34] Roberts, E. (2004). "Resources to support the use of java in introductory computer science." Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, Virginia, pp.233–234

[35] Sammet, J. E. (1972). "Programming languages: History and future." Communications of the ACM 15(7): 601.

[36] Sammet, J. E. (1981). The Early History of COBOL. In R. L. Wexelblat (Ed.), History of programming languages I: ACM.

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

[38] Sime, M.E., Green, T.R.G., and Guest, D.J. (1973). "Psychological evaluation of two conditional constructions used in computer languages." International Journal of ManMachine Studies 5 (1), 105–113.

[39] Smillie, K. (2004). "People, languages, and computers: A short memoir." IEEE Annals of the History of Computing (April-June): 60-73.

[40] Smith, C. and Rickman, J. (1976). "Selecting languages for pedagogical tools in the computer science curriculum." Proceedings of the 6th SIGCSE technical symposium on Computer science education, 39-47