Pedagogical Significance of Natural Language Programming in Introductory Programming

Muhammad Shumail Naveed[1,*] and Muhammad Sarim[2]

[1]*Department of Computer Science & Information Technology, University of Balochistan, Pakistan*

[2]*Department of Computer Science, Federal Urdu University of Arts, Science & Technology, Pakistan*

Abstract: Learning programming is hard for novice students. Complicated syntax and semantic of programming languages and lack of previous knowledge are the contributing factors behind the hardness of programming. Natural programming language allows to program in a natural language and thereby ease the programming. In this paper, it is ascertained whether natural programming language is fruitful in learning the elementary programming concepts and supportive in preparing students for introductory programming courses. The discussion included in this paper can be used to design supportive programming languages and formulating effective courses and learning material to ameliorate performance of students' in learning of introductory programming environments.

Keywords: Introductory programming courses, natural language programming, CS0, failure and dropout rates.

1. INTRODUCTION

The need of computing experts in organizations has led to increase the attention of computer science education [1]. The computer science is a large domain and includes a variety of disciplines and programming is one of them. Programming is a central part of a computer science and one of the most demanding areas of computer science [2]. It is estimated that employment of programmers is expected to grow 8 percent from 2012 to 2022 [3]; however, the academic institutes are not producing the enough graduates in computer science. Even the American colleges are not producing adequate graduate to satisfy the need up to 2020 [4].

Programming within computer science is indispensable, and grasp of this skill is essential for novice students to progress. To be victorious in programming the students have to establish consistent mental of the computer executing their program [5].

The majority of novice students faced difficulties in understanding programming as they are coerced to concentrate on concepts as well as on the syntax of programming languages while many of them have no prior knowledge of programming. Similarly, the majority of programming constructs and concepts are intrinsically difficult for beginners.

In [6, 7], it is described that recursion is one of a hard concept for learners. Similarly, in [8], sequence, assignment, iteration and recursion are described as complex concepts for beginners.

In a comprehensive study [9], it is reported that students experienced problems in determining the valid selection structure for the given problems. Problems are also encountered in evaluating the output statements in the control selection structure and evaluating the conditions for selection structures.

Lahtinen *et al*. in [10], categorized the pointers and references, recursion and abstract data types are categorized as the difficult programming concepts for beginners. Similarly, Dale [11] mentioned the arrays, recursion, pointers and control structures as the difficult topics for novice students. Based on these suppositions multiple courses on programming languages are usually offered to students. The selection of programming language for introductory courses is an important indicator of the concepts focused during course instruction [12].

For the mitigation of the inherent complexities of programming an astounding care is taken in the planning of first programming courses. The first programming course has an everlasting effect on the student attitude toward programming and accordingly the first programming language has a strong influence on the programmer's program development capabilities as profound as the impact of our native language on our thought [13].

The first programming courses are usually fundamental courses, but students commonly face problems even on these fundamental courses [14], and at the end of introductory courses, a large number of

*Address correspondence to this author at the Department of Computer Science & Information Technology, University of Balochistan, Quetta, Pakistan; E-mail: mshumailn@gmail.com

ISSN: 1814-8085 / E-ISSN: 1927-5129/18 © 2018 Lifescience Global

students do not know how to program [15], and this causes high failure and dropouts in introductory programming courses [16, 17].

Several solutions like visual languages & microworlds, program visualization & animation tools, flowchart based programming environments and games have been introduced to increase the performance of students in introductory programming courses.

Several studies [18, 19, 20] reveal that the students with no prior experience of programming face difficulties in completing their education, whereas the prior experience of programming positively affect the performance, especially in the first programming course [21]. For this convincing reason, a small programming course usually called CS0 (pre-programming) is intervened before the first programming course. The CS0 course, virtually aims to provide the basic knowledge of programming to students and develop rational thinking before diving in a first programming course. Several notable CS0 courses have already been introduced [22, 23, 24, 25].

Natural language programming is a class of programming which allows to program in natural language. The use of natural languages increased the programming aptitude and logic of programmers, since there is no need to shape the program ideas according to the structure of programming languages. Similarly, the error handling and debugging of programs written in natural language is comparatively simpler than the other programming languages. Natural language programming has proved very effective in different areas including databases, robot programming and question answering systems.

In this paper, we discuss whether the induction of natural language programming, as a precursor of introductory programming course is helpful in learning the programming essentials and fruitful in comprehending first programming language.

The rest of this paper is organized as follows. The second section describes the natural programming languages and third section describes the major natural programming languages. Section four describes the methodology. Discussion and conclusion are included in section five and six respectively.

2. NATURAL LANGUAGE PROGRAMMING

Language is a system for communication. A language that we learn from our environment and use to communicate with others is called a natural language [26]. Natural languages are fundamentally the most logical way of communication. Analysis and effective use of natural language in different areas of computer science is of a valuable research area of computer science and computational linguistics.

It is generally argued that it is better to program in natural language rather than in classical programming languages like Java [27]. Programming in natural language is an oldest dream of human, albeit many researchers like Dijkstra [28] recognized this as a foolish idea. However, there are many strong proponents of natural language programming. Natural language programming is a type of programming that allows the development of computer programs by means of natural language [29]. The substantial development in the area of parsing, computational linguistics and natural language processing makes it realizable to develop computer programs in natural language.

In ideal programming, the programmers must spend sufficient amount of time in thinking and rationalizing the definite ideas of a program. But in reality, most of the time spends on trimming their thoughts and logic according to the structure of particular programming languages, and this ultimately affects the creativity of programmers. In this situation, natural language programming helps the programmers to think, rationalize and code in their own languages. Likewise, the time and efforts require to develop programs is dramatically reduced. Similarly, the error handling and debugging of natural language programs are comparatively simpler than the programs in other programming languages.

Several notable studies have been conducted to evaluate the effectiveness of natural language programming. Vadas and Curran [27] defined a system that converts English language instructions into Python programming language code and found it very useful. Cambranes [30] introduced the idea of using visual programming for development of program and generation of side by side description in natural language. The system is provided with a series of input/output examples by crowdsourcing and passed to Programming by Example system. Bruckman and Edwards [31] studied the significance of natural language type syntax in programming languages. More than 35,000 commands on MOOSE crossing were recorded in the online interaction of 16 students and the encountered errors were identified and categorized.

The study reported that for various applications, the use of natural language is a dynamic approach to developing the end user programming languages.

3. MAJOR NATURAL PROGRAMMING LANGUAGES

Natural language programming is useful to ease the programming as no extra effort is required in its learning and consequently novice students can concentrate on concepts rather than confronting with unusual and complicated syntax. Since 1960's several notable natural programming languages have been developed.

Pegasus is a multilingual natural language programming developed at Darmstadt University of Technology [29]. It can read and recognize the natural language and generates the executable programs. Pegasus is able to work on multiple languages, including English and German. The use of natural language makes the programming in Pegasus very simple and understandable. The structure of Pegasus is defined in the form of brain and it has three components: the mind, the long-term memory and the short-term memory. Consider the Pegasus acceptable text.

```
Take the matrix ( [2, 2, 1], [1, 4, 3]).
Print it.
New line
If the first element of the second row of the matrix is smaller than 3 then write "I can understand you!".
```

Figure 1: Pegasus program for matrix [29].

The text illustrated in Figure **1** is highly explicable and easy to recognize. In Pegasus, the program is a collection of statements each of them is ended with a colon or full stop.

NLC (Natural language computation) is another useful system based on natural language [32] and used for matrix operations. It allows the users to define commands in English and observe their execution on panel. NLC provides a feedback to the user about the validity of the program during the computation. NLC system comprised of a scanner, syntax analyzer, semantic processor and matrix computer.

Consider a simple program in NLC:

```
"Choose a row in the matrix."
"Put the average of the first four entries in that row into its last entry."
"Double its fifth entry and add that to the last entry of that row."
"Divide its last entry by 3."
"Repeat for the other rows."
```

Figure 2: NLC Program [33].

The problem domain for NLC is the world of matrices shown in Figure **2**, so all the references are associated with entries, rows and entry columns and therefore no reference are added to the problem notions, such as employees or salary.

Liu and Lieberman [34] introduced the idea of natural language specification as the definition for programs and introduced a system called Metafor which visualized user's typed stories as a code. Metafor automatically generates the scaffolding code as the story typed by the user. The scaffolding code is not directly executable, but can be useful for designer. Parser, Programmatic Interpreter, MetaforLingua, Code Renderer, Introspection, Dialog and User Interface are the main components of Metafor.

Consider the Figure **3** for the Metafor program and its visualization:

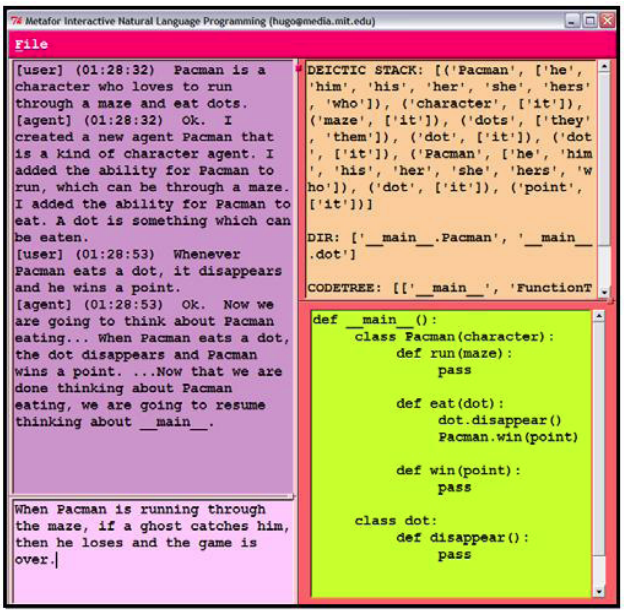

Figure 3: Screenshot of Metafor [34].

The parser of Metafor is based on huge knowledge base of common sense knowledge, but cannot recognize every grammatically correct construction.

Price *et al*. in [35], introduced a natural language interface called NaturalJava for developing Java programs. The restricted natural language is used in the languages, but the pattern of statements and organization of programs is still very imperative. Sundance, PRISM and TreeFace are the main components of NaturalJava.

Consider the Figure **4** for a tiny program of NaturalJava:

```
I would like to define a public method this is named deq and that returns a Comparable.
Declare an int variable name i that is initialized to 1.
Declare an integer variable named minIndex that has an initial value of 0
Add a Comparable variable named minValue which is equal to elements' firstElement but that is cast to a
Comparable
Declare a loop and have it iterate while i < elements' size.
Add a Comparable name c, initialize it to elements' elementAt, pass in i, and cast to a Comparable.
If c's  le when passed minvalue.
minIndex gets i.
minValue gets c.
Exit the loop.
Call elements' removeElementAt and pass it minIndex
```

Figure 4: Program in NaturalJava [35].

NaturalJava supports a huge but restricted subset of Java features. It does not allow nested classes and similarly does not permit array declaration.

sEnglish [36] is a tool used for developing MATLAB programs in natural language. It can generate Latex and HTML document of sEnglish program written in natural language. It is appropriate for the programming of handheld devices, robots and other complex systems.

As an illustration, consider the sEnglish sentences in Figure **5** for the database handling:

```
Amend the record of 'George Xia' with 'mobile phone, email' as '07738675221,g.xia@comp.uk' in contact
database Smvc.

Available things are: N_(quote) , Prs_( quote) , Vals_( quote) , Smvc(contact database) .

Create new record of 'George Xia' with ''mobile phone, email' as '07738675221' in contact database Smvc.

Available things are: N_(quote) , Prs_( quote) , Vals_( quote) , Smvc(contact database) .
```

Figure 5: sEnglish sentences [36].

sEnglish is available for multiple platforms and available in combination with Octave and Python.

SNAP (A Stylized NAtural Procedural language) is a natural procedural language developed for nonscientists [37]. The SNAP procedure consists of English sentences like statements. It allows very plain and straightforward statements for the programming.

Consider a small segment of SNAP program in Figure **6**.

```
CALL "JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST,
SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER" THE MONTH LIST.
(LOOP START) READ A RECORD. SET M TO THE 25-TH THROUGH
26-TH CHARACTERS OF THE RECORD. CALL THE 1-ST CHARACTER OF THE
RECORD THE TENS. IF THE TENS IS " " SET I TO 2, OTHERWISE
SET I TO 1. CALL "-TH" THE END. IF THE TENS IS "1"
CONTINUE WITH THE OUTPUT ACTION, OTHERWISE CONTINUE AS FOLLOWS.
CALL THE 2-ND CHARACTER OF THE RECORD THE UNITS. IF THE UNITS
IS "1" CALL "-ST" THE END. IF THE UNITS IS "2" CALL "-ND" THE
END. IF THE UNITS IS "3" CALL "-RD" THE END. (OUTPUT ACTION)
PRINT "THE " THEN THE I-TH THROUGH 2-ND CHARACTERS OF THE
RECORD THEN THE END THEN " PRESIDENT OF THE U.S. WAS BORN ON "
THEN THE M-TH MONTH THEN " " THEN THE 23-RD THROUGH 24-TH
CHARACTERS OF THE RECORD THEN ".". REPEAT FROM THE LOOP START.
EXECUTE.
```

Figure 6: SNAP Program [38].

The statements' in SNAP are in simple English yet too imperative. The statements generally begin with an imperative verb.

There are many programming languages like Learners Programming Language [39, 40], HyperTalk, Resume, AppleScript, xTalk and Lingo that use natural language in their syntax, but the semantics are towards synthesized programming and therefore do not categorize as natural language programming, but as natural language complemented programming [41].

4. METHODOLOGY & RESULTS

Learning to program is recognized to be hard and challenging task for a significant number of beginners. The main difficulty in learning programming is to obtain a different collection of skills at the same time. Beginners have to learn both the complicated syntax and unusual semantics of a programming language while growing problem-solving abilities.

Natural programming language allows writing program with simple and understandable statements of natural language and thereby relaxing the user from complicated syntax and unusual semantics of programming languages. The effectiveness of natural language programming to ease the programming is an open secret; but, its appropriateness in introductory programming environments by helping novice students to learn elementary programming and comprehend contemporary programming languages are still a less studied area and therefore addressed in this paper.

The Pegasus programs comprised of natural language statements which can be grouped into sections through indentation. This feature is visibly simple, but unavailable in contemporary programming languages like C, C++, Java and C#. In classical languages, the group of statements is explicitly marked with special delimiters which identify the boundary of code sections. This information is essential for novice students, but indentation for grouping the section would be confusing for beginners especially when transitioning to the actual programming language.

The style of programming in Pegasus seems simple but its pattern is far different from contemporary programming languages. Implicit referencing and context dependency are extensively followed in Pegasus programs and pronouns like "it" or "that" are frequently used in Pegasus but not available in conventional programming languages. As an example,

consider the following segment of code for socioeconomic simulation.

> "A society consists of many citizens. The citizens have an income and pay taxes to the state. If a citizen has no job, he is unemployed. Two citizens of different gender can marry"

Implicit referencing and context dependency reduced the size of the program, but these features are not available in classical programming languages. In Pegasus, there is no explicit mechanism for the declaration of typed variables; the notion of selection is absolutely different from the methods available in imperative programming languages. The expressions for the compression like "for all" or "vice versa" are permissible in Pegasus; however, unavailable in classical programming languages and no explicit method of Pegasus is directly traceable to the comprehension of iteration structures in imperative programming.

NLC is based on natural language and its problem domain is the world of matrices, so all the references are associated with entries, rows and entry columns, therefore no reference is added to the problem notions. The selection and iteration in NLC are purely defined in natural style and their structure is entirely different from traditional programming languages and consequently not obliging in understanding the conditional and iteration structures of programming languages. Broadly, no concepts of abstract data type, recursion, pointers and procedure are available in NLC.

Like Pegasus the pronoun like "that" and the expression like "for all" are allowed in NLC. The NLC programs are extensively based on the implicit referencing, context dependency and compression. As an example, consider the following statements:

> "Subtract 4 from THAT entry.
> Double THOSE rows.
> Multiply the last row by ITSELF.
> Add the odd entries to THEMSELVES.
> Square the NEXT entry.
> Triple the OTHER numbers in row 5."

The majority of elementary programming concepts like variable, data type, function, selection and repetition structure are directly not available in NLC.

Metafor is another important programming system based on natural language. The central theme of Metafor is very unique and helpful in mitigating the complexity of programming. However, the essential features of introductory programming like variable, repetition structure, pointers, selection structure, recursions and functions are directly not supported in Metafor. The side-by-side code generated Metafor has been supportive in comprehending programming languages, yet some perquisite is required to comprehend the code. Like other natural language programming the Metafor support implicit referencing, context dependency and syntax compression. These traits significantly reduced the size of programs, but provide no support in learning programming fundamentals.

NaturalJava is an interface based on restricted natural language. In NaturalJava, the elementary concepts of introductory imperative programming like variables, data types, methods, selection and repetition structures are explicitly allowed in the restricted natural language and the statements are highly motivated from high-level languages. Some unusual statements which are uncommon in other contemporary programming languages are also allowed in NaturalJava like "I would like" and "please return". The pronoun like "it" is allowed in NaturalJava. Working with NaturalJava may help the beginners to understand the imperative concepts of programming like variables, data types, selection and repetition; however, it is typically designed to develop Java programs so based on methods, attributes and classes.

sEnglish is one of an important development in the area of natural language programming. It is particularly developed for engineers and scientists so it's no primitive features typically address any concept of introductory programming. The subset of English used in sEnglish is quite simple, but the overall structure of statements is still unusual and only technical users with some knowledge of MATLAB can program in sEnglish. More importantly the users who are proficient in sEnglish gain no obvious knowledge of introductory programming.

The definition of variable in sEnglish is descriptive, but complex in that multiple properties of a variable is defined in a single statement which is different from traditional programming languages. The natural language representation of assignments, calling of functions and passing of parameters to functions in sEgnlish is still technically and cannot be helpful in visualizing the equivalent concepts of programming languages. Similarly, the implementation of selection

and repetition structures are far different from classical programming languages.

SNAP is a natural language based system and essentially designed for teaching, but its string handling techniques are more appropriate for text processing systems. The statements in SNAP generally begin with an imperative verb like *READ*, *PRINT*, *APPEND* and *DELETE*, and many of them are very similar to their equivalents in high-level languages. For instance, the print statement consists of a verb *PRINT*, a quotation and a period. The input statement consists of verb *READ*, name and a period.

Declaration of the procedure is allowed in SNAP and very much similar to the procedures of high-level languages. A mechanism for transforming the control is available in SNAP and the *EXECUTE* statement is available for transferring control from the loader to the processor. Conditional statements in SNAP starts with *IF* and includes the conditions with the possible actions.

Unlike conventional programming the condensing of statements is allowed in SNAP:

{INCREASE, DECREASE} e BY f.

The repetition is allowed in SNAP and *CONTINUE WITH* is used for going forward and *REPEAT FROM* for going back. Control is return to monitor with *TERMINATE* statement. The arithmetic operations are originally allowed in linguistic form, but the general arithmetic instructions have been introduced.

Most of the SNAP list operations are not very useful in learning introductory programming. The statements of SNAP are in computable style yet too verbose and would not be a very cogent tool for introductory programming environments; however, it would be supportive in the basic course of data processing.

All the major natural programming languages are based on the same philosophy, so their main features are summarized in Table **1**.

The natural programming languages support familiar lexicons with simple syntax and semantics. However, the primitive data types which are available in contemporary programming languages are mainly not available in natural programming languages. Similarly, the concept of variable declaration and delimitation are also missing in natural programming languages.

The imperative style of selection and iteration structures are directly not supported in natural programming languages, and similarly the generation of high-level programming code is only available in few natural programming languages. In any natural programming language, there is no predefined support which helps the users while transitioning to the traditional programming languages. Implicit

Table 1: Main Features of Natural Language Programming

Features	Natural Programming Languages					
	Pegasus	**NLC**	**Metafor**	**NaturalJava**	**sEnglish**	**SNAP**
Use of common lexicons	Yes	Yes	Yes	Yes	Yes	Yes
Simple syntax	Yes	Yes	Yes	Yes	Yes	Yes
Plain semantics	Yes	Yes	Yes	Yes	Yes	Yes
Direct support of major data types	Limited	No	Limited	Yes	Limited	Limited
Explicit declaration of typed variables	No	No	No	Yes	No	No
Explicit construct/marks for delimiters	No	No	No	No	No	No
Support of elementary programming features	Limited	Limited	Limited	Limited	Limited	Limited
Direction support of programming selection structure	Limited	Limited	Limited	Yes	Limited	Limited
Direction support of programming iteration structure	Limited	Limited	Limited	Yes	Limited	Limited
Generation of high level programming code	Limited	No	Limited	Yes	Limited	No
Support in transition to other programming languages	No	No	No	No	No	No
Implicit referencing	Yes	Yes	Yes	Yes	Yes	Yes
Context dependency	Yes	Yes	Yes	Yes	Yes	Yes
Compression	Yes	Yes	Yes	Yes	Yes	Yes

referencing, context dependency and compression are the real traits of natural language programming and make their program concise.

The prominence of any programming language can be evaluated with its popularity and natural programming languages are no exception. The TIOBE programming community index is of an important and widely used measure of the popularity of programming languages. The TIOBE index is updated once a month. The statistics of popular languages for October 2017 are illustrated in Figure **7** [42].

Oct 2017	Programming Language	Ratings
1	Java	12.431%
2	C	8.374%
3	C++	5.007%
4	C#	3.858%
5	Python	3.803%
6	JavaScript	3.010%
7	PHP	2.790%
8	Visual Basic .NET	2.735%
9	Assembly language	2.374%
10	Ruby	2.324%
11	Delphi/Object Pascal	2.180%
12	Perl	1.963%
13	MATLAB	1.880%
14	Scratch	1.819%
15	R	1.684%
16	Swift	1.668%
17	Objective-C	1.513%
18	Visual Basic	1.420%
19	PL/SQL	1.408%
20	Go	1.357%

Figure 7: TIOBE Index for October 2017.

Figure **7** shows that none of any natural programming language is included in the top twenty popular programming languages. More detailed information maintained by TIOBE community on popular programming languages from 2002 to 2016 is shown in Figure **8**.

Popularity index of programming languages illustrated in Figure **8** shows that from 2002 to 2016 none of any natural programming language is recognized as a popular language.

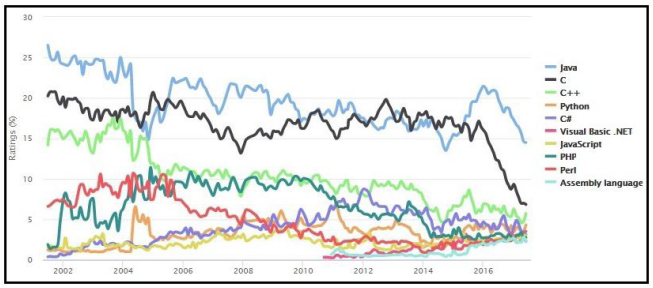

Figure 8: TIOBE Programming Community Index [42].

5. DISCUSSION

The use of natural language in computer programming has had a considerable impact on simplifying the programming. The majority of natural programming languages comprised of simple lexicons with obvious syntax and semantics. The concept of data types is somewhat present in natural language programming, yet the conventional data types of programming languages are almost unavailable in natural language programming. The absence of conventional data types in natural programming languages eases the programming, but would not help the students in learning conventional programming languages.

The explicit declaration of variable is one of a central concept of elementary programming and thereby available in most of the classical programming languages including C, C++ and Java. This feature is almost unavailable in natural language programming and the absence of this feature obviously eases the pattern of programming and reduced the size of program code. However, the experience of working with implicitly declared variables in natural programming language is problematic for novice students of traditional programming language because these languages are based on the explicit declaration of variables.

The concept of explicit delimitation is unavailable in natural programming languages, but extensively used in contemporary programming languages. So, working with natural programming languages provides no support for novice students in learning explicit delimitation of contemporary programming languages. Similarly, the pattern of selection structure and iteration structures of natural programming language is far different from classical programming languages and hence not very helpful for classical programming environments.

The majority of natural programming languages does not support the generation of pure high-level code

from the input source program. So, it would very difficult to learn the high-level code of classical programming languages from natural programming code. More importantly, no feature is available in any natural programming language that would be helpful for their users transitioning to common languages of introductory programming environment.

The implicit referencing, context dependency and compression are the pivotal traits of natural programming languages, and helpful in reducing redundancy, but absent in classical programming languages. In most of the introductory programming languages all the programming elements like variables, constants and functions are explicitly referenced whereas implicitly referenced in natural language programming. In the same way, syntax and semantic compressions are almost unavailable in high-level programming languages, but frequently used in natural language programming. So, anyone who is proficient in natural language programming should naturally face problems in learning elementary programming languages because these languages are based on implicit referencing with limited or no support of context dependency and compression.

The design philosophy observed in natural language programming revealed that its' sole objective is simplifying the intrinsic nature of programming and helping end users to develop programs without learning and understanding the fundamental concepts of programming. However, its different structure and pattern of programming makes it different from elementary programming languages and therefore it is not very helpful in introductory programming environments.

6. CONCLUSION

Programming learning is a meticulous task and naturally more complex for many students. Complicated syntax and semantic of programming languages and lack of previous knowledge are the contributing factors behind the intricacies of introductory programming courses. Natural programming language is one of the prominent systems that ease the programming by allowing simple, plain and understandable notation of a natural language. Use of natural language makes the programming simple for many users; however its general philosophy, programming style and supported features are far different from the conventional programming languages. Implicit referencing, context dependency, compression, implicit declaration of variables and delimitation are extensively exercised in natural language programming. These features manifestly improve the readability and ease the programming, but these features are almost absent in contemporary programming and therefore the notion of natural language programming is not very helpful for learning the fundamental of programming and topical languages are thereby not very supportive for the introductory programming environment.

ACKNOWLEDGEMENTS

The authors greatly acknowledge the support from Muhammad Tahaam and Muhammad Aayaan for motivating and providing with the material for this study and for giving valuable feedback and comments.

REFERENCES

[1] Qian Y, Lehman JD. Correlates of Success in Introductory Programming: A Study with Middle School Students. Journal of Education and Learning 2016; 5(2): 73-83. https://doi.org/10.5539/jel.v5n2p73

[2] Reardon S, Tangney B. Smartphones, studio-based learning, and scaffolding: Helping novice learn to program. Transaction of Computer Education 2014; 14(4): 1-15. https://doi.org/10.1145/2677089

[3] Bureau of Labor Statistics, U.S. Department of Labor, [homepage on the Internet]. Occupational outlook handbook, 2014-15 Edition, Computer Programmers. [cited: 2016 Sep 18]: Available from: from http://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm

[4] Microsoft Corporation, 2012. [homepage on the Internet]. A national talent strategy: Ideas for securing U.S. competitiveness and economic growth, 2012. [cited: 2013 Jul 22]: Available from http://www.microsoft.com/en-us/news/download/presskits/citizenship/MSNTS.pdf.

[5] Berry M, Kölling M. Novis: A Notional Machine Implementation for Teaching Introductory Programming. Proceedings of the 2016 International Conference on Learning and Teaching in Computing and Engineering; 2016; 31 Mar – 3 Apr; Mumbai, India; 2016; p. 54-59. https://doi.org/10.1109/LaTiCE.2016.5

[6] Haberman B, Averbuch H. The Case of Base Cases: Why Are They So Difficult to Recognize? Student Difficulties with Recursion. ACM SIGCSE Bulletin 2002; 34(3): 84-88. https://doi.org/10.1145/637610.544441

[7] Lewis CM. Exploring Variation in Students' Correct Traces of Linear Recursion. Proceedings of the Tenth Annual Conference on International Computing Education Research; 2014: Aug 11-13; Glasgow, Scotland, United Kingdom; 2014; p. 67-74. https://doi.org/10.1145/2632320.2632355

[8] Vujoševic-Janicic M, Tošic D. The role of programming paradigms in the first programming courses. THE TEACHING OF MATHEMATICS 2008; XI(2): 63-83.

[9] Gobil AR, Shukor Z, Mohtar IA. A. Novice Difficulties in Selection Structure. Proceedings of International Conference on Electrical Engineering and Informatics; 2009: Aug 5-7; Selangor, Malaysia; 2009; p. 351-356. https://doi.org/10.1109/ICEEI.2009.5254715

[10] Lahtinen E, Ala-Mutka K, Jarvinen H. A Study of the Difficulties of Novice Programmers. ACM SIGCSE Bulletin 2005; 37(3): 14-18. https://doi.org/10.1145/1151954.1067453

[11] Dale NB. Most Difficult Topics in CS1: Results of an Online Survey of Educators. ACM SIGCSE Bulletin 2006; 38(2): 49-53. https://doi.org/10.1145/1138403.1138432

[12] McMaster K, Sambasivam S, Rague B, Wolthuis S. Java vs. Python Coverage of Introductory Programming Concepts: A Textbook Analysis. Information Systems Education Journal 2017; 15(3): 4-13.

[13] Wexelblat RL. 1998. The consequences of one's first programming language. Software: Practice and Experience 1981; 11(7): 733-740. https://doi.org/10.1002/spe.4380110709

[14] Amarathunga ML. Supporting Tool for Introductory Programming Labs. 5th Annual UCSC Research Symposium; 2012; p. 37-41.

[15] McCracken M, Almstrum V, Diaz D. Guzdial M. Hagan D. Kolikant YB. Laxer C. Thomas L. Utting I. Wilusz T. A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students. ACM SIGCSE Bulletin 2001; 33(4): 25-180. https://doi.org/10.1145/572139.572181

[16] Sloan RH, Troy P. CS 0.5: A Better Approach to Introductory Science for Majors. ACM SIGCSE Bulletin 2008; 40(1): 271-275. https://doi.org/10.1145/1352322.1352230

[17] Herrmann N, Popyack JL, Char B, Zoski P, Cera CD, Lass RN. Nanjappa A. Redesigning Introductory Computer Programming Using Multi-level Online Modules for a Mixed Audience. ACM SIGCSE Bulletin 2003; 35(1): 196-200. https://doi.org/10.1145/792548.611967

[18] Davy JR, Audin K, Barkham M, Joyner C. Student Well-being in a Computing Department. ACM SIGCSE Bulletin 2000; 32(3): 136-139. https://doi.org/10.1145/353519.343145

[19] Hagan D, Markham S. Does It Help to Have Some Programming Experience Before Beginning a Computing Degree Program. ACM SIGCSE Bulletin 2000; 32(3): 25-28. https://doi.org/10.1145/353519.343063

[20] Morrison M, Newman TS. A Study of the Impact of Student Background and Preparedness on Outcomes in CS1. ACM SIGCSE Bulletin 2001; 33(1): 179-183. https://doi.org/10.1145/366413.364580

[21] Holden E, Weeden E. The Impact of Prior Experience in an Information Technology Programming Course Sequence. Proceedings of the 4th Conference on Information Technology Curriculum; 2003; Oct 16-18; Indiana, USA; 2003; p.41-46. https://doi.org/10.1145/947121.947131

[22] McIver L, Conway D. GRAIL: A Zeroth programming language. Proceedings of seventh International Conference on Computing in Education; Amsterdam Netherlands; 1999; p. 43-50.

[23] Panitz M, Sung K, Rosenberg R. Game Programming in CS0: A Scaffolded Approach. Journal of Computing Sciences in Colleges 2010; 26(1): 126-132.

[24] McFarland RD. Development of a CS0 Course at Western New Mexico University. Journal of Computing Sciences in Colleges 2004; 20(1): 308-313.

[25] Ernest JC, Bowser AS. Ghule S. Sudireddy S. Porter J.P. Talbert DA. Kosa MJ. Weathering MindStorms with Drizzle and DIODE in CS0. ACM SIGCSE Bulletin 2005; 37(3): 353-353. https://doi.org/10.1145/1151954.1067552

[26] Harris MD. Introduction to Natural Language Processing. Reston: Reston Publishing Company; 1985.

[27] Vadas D, Curran JR. Programming With Unrestricted Natural Language. Proceedings of the Australasian Language Technology Workshop; 2005: Dec 10 -11; Sydney, Australia; 2005; p. 191-199.

[28] Dijkstra EW. On the foolishness of "natural language programming". Program Construction, Lecture Notes in Computer Science 1979; 69: 51-53. https://doi.org/10.1007/BFb0014656

[29] Knol R, Mezini M. Pegasus: First Steps Toward a Naturalistic Programming Language. Proceedings of companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications; 2006; Oct 22 – 26; Portland, Oregon, USA; 2006; p. 542 - 559.

[30] Cambranes E. Using Natural Language Descriptions of Algorithms in the Early Stage of Programming. Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing; 2012: 30 Sept - 4 Oct; Innsbruck, Austria; 2012; p. 217-218. https://doi.org/10.1109/VLHCC.2012.6344521

[31] Bruckman A. Edwards E. Should We Leverage Natural-language Knowledge? An Analysis of User Errors in a Natural-language- style Programming Language. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 1990; May 15-20; Pittsburgh, Pennsylvania, USA; 1990; p. 207-214.

[32] Ballard BW, Biermann AW. Programming in Natural Language: "NLC" As a Prototype. Proceedings of the ACM 1979 Annual Conference; 1979; p. 228-237. https://doi.org/10.1145/800177.810072

[33] Biermann AW, Ballard BW, Holler AM. A System for Natural Language Computation. ACM SIGLASH Newsletter 1979; 12(1): 6-16. https://doi.org/10.1145/1041361.1041362

[34] Liu H, Lieberman H. 2005. Metafor: Visualizing Stories as Code. Proceedings of the 10th International Conference on Intelligent User Interfaces; 2005; Jan 10 - 13; San Diego, California, USA; 2005; p. 305-307. https://doi.org/10.1145/1040830.1040908

[35] Price D, Rilofff E, Zachary J, Harvey B. NaturalJava: A Natural Language Interface for Programming in Java. Proceedings of the 5th International Conference on Intelligent User Interfaces; 2000: Jan 09-12; New Orleans, Louisiana, USA; 2000; p. 207-211. https://doi.org/10.1145/325737.325845

[36] System English [homepage on the Internet]. 2004 [cited 2015 Dec 03]: Available from: http://www.system-english.com/

[37] Barnett MP, Ruhsam WM. SNAP: An Experiment in Natural Language Programming. Proceedings of the Spring Joint Computer Conference; 1969: May 14-16; Boston, Massachusetts, USA, 1969; p. 75-87. https://doi.org/10.1145/1476793.1476815

[38] Barnett MP, Ruhsam WM. A Natural Language Programming System for Text Processing. IEEE Transactions on Engineering Writing and Speech 1968; 11(2): 45-52. https://doi.org/10.1109/TEWS.1968.4322334

[39] Naveed MS, Sarim M, Ahsan K. Learners Programming Language a Helping System for Introductory Programming Courses. Mehran University Research Journal of Engineering & Technology 2016; 35(3): 347-358.

[40] Naveed MS, Sarim M, Ahsan K. On the Felicitous Applications of Natural Language. Science International 2015; 27(23): 2643-2646.

[41] Knoll R, Gasiunas V, Mezini M. Naturalistic Types. Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software; 2011; Oct 22-27; Portland, Oregon, USA, 2011; p. 33-48. https://doi.org/10.1145/2048237.2048243

[42] TIOBE [homepage on the Internet]. TIOBE Index for October 2017; [updated 2017 Oct; cited 2017 Oct 9]: Available from: https://www.tiobe.com/tiobe-index/.

Received on 10-03-2018 Accepted on 20-03-2018 Published on 06-04-2018

https://doi.org/10.6000/1927-5129.2018.14.09

© 2018 Naveed and Sarim; Licensee Lifescience Global.
This is an open access article licensed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted, non-commercial use, distribution and reproduction in any medium, provided the work is properly cited.