University of Sindh Journal of Information and Communication Technology (USJICT)
Volume 4, Issue 3, October 2020

ISSN-E: 2523-1235, ISSN-P: 2521-5582 © Published by University of Sindh, Jamshoro
Website: http://sujo.usindh.edu.pk/index.php/USJICT/

PFE: A Visual Programming Frame Work for Teaching Programming to Dummies or beginners

Mufti Anees-Ur-Rahman, Rai Sabir, Umer Riaz, Tauseef Rana
Department of Computer Software Engineering, Military College of Signals, NUST
Rawalpindi, Pakistan
Email: muftianees7@gmail.com, raisabir81@gmail.com, umer.riaz@live.com, tauseefrana@mcs.edu.pk

Abstract—Modern human lifestyle has revolutionized and there is a paradigm shift in ways, how humans and machines interact. Everybody is connected and concerned with machines through software and computer applications and SMART is the ongoing style everywhere. With this increased human-machine interaction and the presence of smart machines in their lifestyles, user curiosity and involvement in software logic and its underlying processes has increased the importance and consideration of user understanding of programming manifold. However, programming and developing computer programs have remained a problem specific to software engineers, professional developers, computer geeks and IT professionals, and it has excluded the general public's understandability and involvement. Henceforth this research paper is an endeavor to bring basic programming and its underlying logics to novice learners and non-programmers through a visual programming framework that will enable people with basic computer handling knowledge, to create simple logics and computer programs. Programming for Everyone (PFE) is a visual programming framework that defines basic construction parameters to build visual programming software that enables non-programmer users from outside IT industry to gradually learn and build programming logic and computer applications.

Index Terms—Visual Programming Language, Visual Coding

I. INTRODUCTION

Evolution of IT and software has dramatized the way humans and machines interact. From the simplest one function button machines to complex automated industrial units, software evolution has taken a high tide. In the current wave of technological evolution, a near-future is foreseen where technology is omnipresent, machines predict and anticipate human needs, robotic systems are an integral part of everyday life, and humans' abilities are technologically supported [1]. This increased presence of machines in human life entails that they know each other well, specifically it is and will be perilous for humans if they know less. Computer programming and application development have remained a problem associated with software engineers, professional developers, computer geeks and enthusiasts. Very less of this World of logic and Programming is known to software applications end-user human actors. The increased presence of machines and systems in human life has almost made it a necessity to know IT and this coupled with human curiosity to explore the unknown, have increased their want of knowing logic and programming running behind the systems. However, the knowledge, notations and languages used in software development industry are too complex, technical, syntax and semantic oriented that its understanding for people outside IT domain is a complex and a not to do the task.

Visual Programming techniques and environments have tried to reduce the void of understanding by providing a visual and noncoding mechanism for creating logic and programs but that too by and large had been addressing the requirements of users from within IT domain. From UML to Flow chart diagrams, Xman to extended Xman and so on most of the endeavors are intended to facilitate people from within software industry. This research paper aims at the identification of this void, its importance from HCI point of view and building up of a framework that enables novice users to learn basic logic and programming skills with no coding technique or language to know. Basic research questions that these papers have tried to answer are as follows:

RQ1: Is learning programming and logic important for everyone?

RQ2: Which programming learning approach is better; Text Based Languages (TBLs) or Visual Programming Languages (VPLs)?

RQ3: Does existing VPLs and applications included or considered users from outside IT domain to learn programming and to what extent?

RQ4: What could be a possible framework that can be used to build Visual Programming Applications that could help users from outside IT domain to learn basic programming logics and program construction without having to write any codes?

The framework, Programming for Everyone (PFE), proposed in this paper tries to define the basic parameters to build a visual programming environment or application and logic learning software that can enable users to understand basic programming logics and build simple programs through visual and graphic elements without having any code to write or understand. PFE defines a software program as Buddy that is a virtual companion of our user, created to perform desired functions based on logics and knowledge the user learns and provides. The Knowledge (K) in our framework represents the inclusion of previously build libraries, components, functional modules and services, with a singular package called Knowledge Unit (Ku). Assigning a Ku to our buddy implies that the buddy now knows how to act upon knowledge assigned through a knowledge unit Ku. Technically, all Ku related functioning code and libraries are now available to be used through their respective functions calls or interfaces that shall be provided to the user through visual or graphic interface.

Buddy B, a developed computer program by the user shall be a mirrored representation of the programming knowledge of the user itself. The frame might also consider defining some social networking aspects for users to share their buddy characteristics with other learners and to benefit from each other's learning experiences.

II. LITERATURE REVIEW

A group of 32 experts from HCI domain carried out empirical research to identify grand challenges being faced by the Human-Computer Interaction domain. This effort was supported by HCII Conference series that concluded at HCII conference at Las Vegas, USA in 2018. Later the research was published by the International Journal of Human-Computer Interaction, Taylor and Francis in Jul 2019. The research identified 7 grand HCI challenges as Human-Technology Symbiosis, Human-Environment Interaction, Ethics Privacy and security, Well Being Health and Eudemonia, Accessibility and Universal Access, Learning and Creativity and Social Organization and Democracy [1].

Further exploring the individual challenges, the research explains the challenge of learning and creativity as "As technologies continue to mature, new opportunities for fostering individual growth through multi-modal stimulation of how humans learn and apply creativity will emerge. People with diverse backgrounds, skills, and interests will be able to collaborate to solve challenging problems, by cooperatively learning and creating knowledge together. In this new era, technology will support and promote new learning styles, multi-modal learning affordances, as well as lifelong learning"

[1]. Therefore, to embrace the exponential technology growth together and to accrue maximum collective benefits, it is imperative to create programming learning environments for everyone. Recognizing this importance, some futuristic leaps have been taken recently to even teach computer programming to people right from the young age. As a result of these studies and analysis, many countries have introduced programing language as integral part of curriculum for children right from early ages. For an example in Japan, a new course of study shall be followed from 2020 onwards and computer programming has been added as a mandatory subject. Children attending primary schools shall be required to study programming in different disciplines throughout their curriculum. The UK government realizing the importance of knowing programming and logic has also included computer science education right from the beginning. In their new computing curriculum, the children aged between 11 and 14 are required to use two or more programming languages [3].

Apropos, we can conclude that learning programming and computer sciences is an imperative requirement now and it has been realized and efforts are in hand everywhere to foster the upcoming generations with this knowledge right from the start. Those already out of schools and learning process are likely to be left out, if they don't somehow get into learning computer programs and logics through easily understandable, non-coding and ways best suited to them.

III. TEXT BASED LANGUAGES (TBLS) VS VISUAL PROGRAMMING LANGUAGES (VPLS)

The history of formally recognized programming languages dates back to early fifties where computer scientists started to build set of characters and rules for combining them so as machine code knowledge becomes unnecessary and programmers could talk to computers in a language near to real languages. The first major meeting held solely to discuss higher-level languages or automatic coding as the subject was referred to then was at the Franklin institute in 1956 [4].

Programming Languages alongside technology evolution have also been revolutionized and comprehensive rules and mechanisms have been built both as textual programming languages (TBLs) and Visual Programming Languages (VPLs). VPLs though smaller in age than TPLs are considered more appropriate especially for teaching programming to new and novice programmers. A VPL or environments are built so as to allow users to develop programs and applications without writing the program codes textually. Rather they allow users to manipulate various program codes and blocks graphically through an interactive mechanism involving drag drop or click functions. [5].

In a research in Japan, textual programming languages and visual programming languages were compared from the aspect of learning motivation. It was carried out by selecting two classes of primary students to work on VPLs and the other one with TPLs. After the experiment, results concluded that from Motivation of Users point of view, the class that worked on VPLs showed better scores as the course progressed while those with TPLs didn't show any significance increase in their motivational score. Hence it can be deduced that VPLs are a better choice than TPLs keeping in view the learning motivational scores, for teaching programing to school children and novices. [6].

Visual Programming Languages and environments are built to enable users to be able to create computer programs without having to write any lines of code or programming language. They use graphical menus and interactions instead and programming commands and options are displayed in the form of blocks with a pre-defined color scheme associated with each type and category of block. These blocks are joined together through visual options and resultantly required codes are generated for the program. Therefore, we conclude that since VPLs have lesser requirements from user in terms of language syntax and semantics, hence the chances of user errors are much lesser as compared to TPLs. Apropos, VPLs are a more suitable choice than TPLs for learning programming and teaching programing for educational purpose. [7]. Therefore, we conclude that VPLs are a better choice while developing program learning environments for the novice, young and new programming learners. Researchers have considered VPLs to be closer to the needs of new learners. Since technology, its evolution and presence in human lives is a reality and an inevitable fact, this omnipresence of technology brings a major challenge that is to find structured and comprehensive mechanisms to pass on the knowledge, through learning and training, to diverse nature of the audience. Therefore, there is a need to understand the influences of human factors to design digital learning environments that best suit every learner [8].

IV. EXISTING FRAMEWORKS

In [9] a few programmers were also given a task which they needed to code using 2 programming languages. One of the languages was known to them the other was unknown. This was done to get their opinions on the differences, which shows us that it is not difficult for even the older veteran programmers to adopt a new language. Another experiment was carried out in [10] which included 60 students who all knew a procedural programming language and were divided into 3 groups. They had to use a Visual Programming language called VEDILS to control a robot using a Bluetooth connection with an android device. After the series of experiments, the results suggested that the visual programming language made understanding of the problems easier for the student. They had a better idea of the system and they also enjoyed their work using the Visual language rather than the textual interface. they mainly used Block Languages for this. Similarly, another study was also carried out in [11]. This time using Blockly.

Blockly is a Block language Developed by Google. PBL-VP (Problem Based Learning-Visual Programming) was used in this scenario. the PBL was used to guide students to analyze problems, provide solutions and establish scientific logic thinking. In the end, the effectiveness of this PBL was tested and it came out more than the normal teaching methods that are used by us which is textual Programming.

One slightly different approach to this matter was made when the use of "recognition over recall" method was used in [12]. the aim of this was rather than focusing on the syntax of the programming language students should focus on programming logic. This was done while using a Visual programming language called "Block-C". This not only provided the evidence that this was better for beginners but also had an advantage of eliminating Syntax errors while using the graphical coding method. Similarly, there is another Visual Programming Language named "Milo". It uses Graphical blocks to represent Machine learning and Data science concepts. its code is of the same "level" as JavaScript. Again, this proved to be very successful for novices and beginners and had a positive effect [13]. one other interesting implementation of this is carried out by using a "Flow Model" to code using visual programming. In this experiment, no existing visual programming language was used for this pilot program. A benefit to this was that there was no programming enforced rule sin this methodology [14].

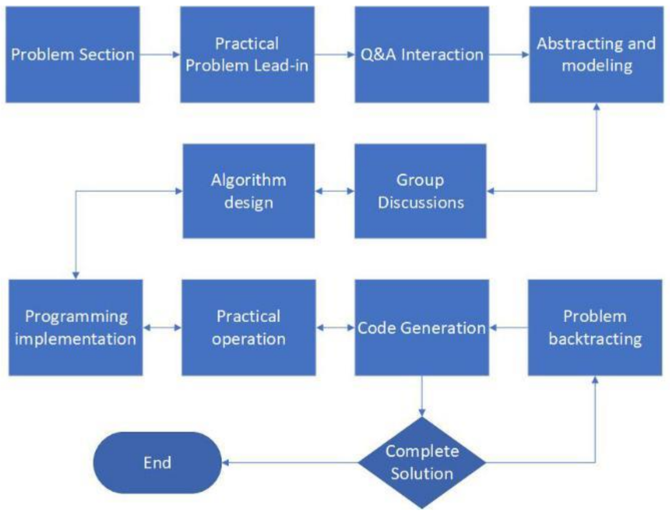

Figure 1. Flowchart of PBL-VP teaching Process

V. PROGRAMMING FOR EVERYONE (PFE)

Programming for Everyone is a concept and framework that endeavors to bring computer programming and logic understanding to novice dumb programmers. The acquired logic and knowledge shall gradually be built through a systematic and progressive mechanism. Users shall be able to explore computer logics and programming as per their own interests and needs.

A. Fundamental concepts of PFE Buddy

PFE considers user learning process and experience as the foremost factor and we have named the user developed computer program or application as Buddy (Bdy) representing user knowledge and expertise about computer programming and logic. This buddy shall constantly remain as an interface between user and computer logic and programming knowledge base available locally or through cloud resources.

Users shall interact with the system using the user interface of Buddy and programming logic resources shall interface with Buddy using appl interface as shown in fig1. A buddy shall be created for each user at the beginning of user programming learning experience and shall continue throughout hi learning life cycle. Buddy can be represented with a graphical representation like a pic, icon, or any other representation user wishes to select suiting his cognitive and imaginative behavior.

B. The Knowledge K

The learned programming and computer logic by a user shall be represented by Knowledge K that shall provide an insight into functionalities and capabilities acquired by Buddy during the learning process. Knowledge shall consist of smaller programming packages called Knowledge Unit (Ku) which can consist of user developed program or an already developed component available in the form of coding libraries and packages. User shall keep increasing the knowledge of his program by adding knowledge units to the system. Adding knowledge unit in the form of external libraries or packages shall require a graphical visual interaction of the user with a list of available knowledge units depending upon the underlying programming language and packages or libraries offered. User shall be able to add ku to his buddy knowledge by just dragging and dropping knowledge unit from the list. Each knowledge unit on addition by user, shall automatically provide a list of functionalities it offers to user in a graphical menu on user action like mouse click or mouse over function through Buddy user interface as shown in fig 3.

For example if a user wants to work in C++ language and learn mathematical functions, he will just drag drop maths library from C++ language package being provided in the buddy menu, the math.h library shall be included in knowledge, providing list of math functions the library can handle, through buddy interface Fk where Fk is the consolidated functions list of all functionalities included in knowledge through respective Knowledge units.

At any point of time the list of functions fK of knowledge K is the sum or consolidation of all functions being provided by individual knowledge units Ku. Mathematically, list of knowledge functions fk at any time t is

$$(fK)t= (SfKu)n$$

This list shall be updated upon the addition of each new knowledge unit by the user as shown in figure 2. For every new addition, the fK is updated as:

$$fK = fK + fKU \text{ (new list)}$$

With this mechanism, a user can always have a track of the available list of functions and capabilities with the buddy or in retrospect user himself.

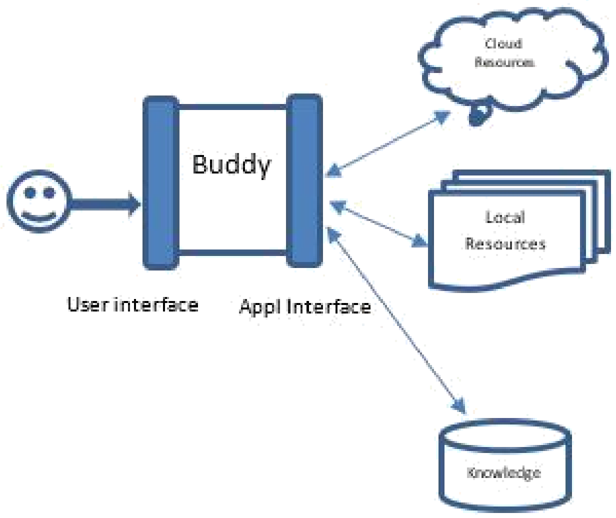

Figure 2. An overview of the Language Buddy

C. Buddy Classification

The buddy of a user at any point of time shall depict his learned programming capability through a classification mechanism of user buddy. For a buddy or user to be qualified in a knowledge unit with a list of functions fKu, he must have used at least 75% of the functions in his own created program. Based upon consolidated qualification credentials an overall expertise classification of buddy can be done. The Buddy can be classified into following 6 categories: -

1) Beginner
2) Basic Programmer
3) Programmer
4) Good Programmer
5) Expert Programmer
6) Legendary Progammer

This no of classifications can be increased or decreased as found appropriate during implementation or development process of the concept. Moreover, the classification criterion and algorithm shall also be worked in more detail in subsequent phases of the research.

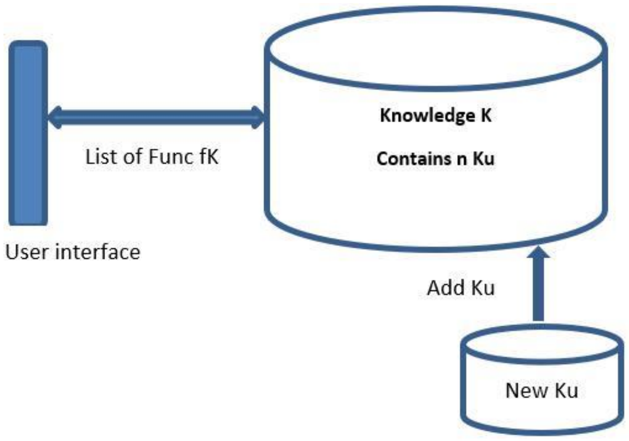

Figure 3. Interface interaction

D. User learning procedure

According to PFE framework, the system shall provide user creation module where the user shall be prompted to give a name to his Buddy, select its graphical representation and some other basic profile details (optional) like age, email, contact no, qualification etc. This shall provide him with a personalized experience while learning programming and also shall help him in displaying his credentials for online forums for knowledge and experience sharing. This unique buddy for each user shall keep on updating itself based on user learning and programming process. User shall be required to use functionalities of knowledge units graphically. For example, if a user has added maths library to knowledge, a maths Tab shall appear on buddy menu, having list of all maths functions the user can use. When a user selects a particular function its required inputs and outputs shall automatically drop down into programming screen of the user i.e upon selection of add or sum function the containers for input nos and result shall automatically appear on screen for user to configure how many input nos he wants to use and how he wants them to be provided to user (manual prompted input from user or from a list or data base of numbers) and finally what to with the calculated result. The system shall also keep a track or record of functions of a particular library or component being used by the user in his program and this info shall be used in classification of user as qualified or not, in a particular component and further will add to overall classification of a user buddy.

E. System Upgradation Process

The PFE framework also explains a mechanism for upgrading the system by the addition of new language supported and their libraries. This procedure will involve a few step configurations of the system by developers and release of new version with added support of new language. This language inclusion package shall include:

1) Name of Language i.e java, C++
2) List of libraries or packages being offered as knowledge units Ku.
3) List of functions fKu for each Ku.
4) Integration mechanism for language compiler.
5) Interoperability requirements for interaction with other language packages or functionalities.

Details of this mechanism shall be explained or explored in future research

F. User interaction through online form

The program learning experience through PFE shall also provide users with an online forum or application to interact with other learners and share learning experiences. User buddy profile can be used as a basic identification tool for this online interactive mechanism. This interconnected learning shall increase learning motivation of users, provide a centralized problem sharing platform, provide users with an opportunity to interactively learn and share experiences and build an inclusive community of learners. This community may also be used for finding or hiring programmers with a required set of expertise employers are looking for.

VI. FUTURE WORK

The following shall be carried out in further research of the project: -

1) Defining user interface design and working prototype model.
2) Defining language integration process.
3) Defining buddy classification algorithm.
4) Defining mechanisms for online interactive forum or networking system.

VII. CONCLUSION

Programming has been exclusive in nature as it has remained more related to computer experts and IT field people. However, the expansion of IT in day-to-day human lives have increased the want and requirement of learning logic and programming by everyone. PFE hence provides a basic idea of how learning of programming and logic can be brought to an inclusive set of users without getting them into language syntax and semantics. The gradual visual learning environment shall motivate users and enable them to learn logic and programming as per their personalized interests and requirements hence shall decrease the chances of people quitting learning programming. Moreover, it will also provide an online platform to converge the vast community of learners for sharing their experiences and enable the professional IT industry to hunt for good programmers for professional use.

REFERENCES

[1] C. C. Stephanidis and G. Salvendy, "Seven HCI Grand Challenges", International Journal of Human-Computer Interaction, vol. 35, Number 14, 2019 pp. 1229-1269.

[2] H. Tsukamoto, Y. Oomori, H. Nagumo, Y. Takemura, A. Monden and K. Matsumoto, "Evaluating algorithmic thinking ability of primary schoolchildren who learn computer programming," 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, 2017, pp. 1-8.

[3] "Statutory guidance National curriculum in Eng-land: computing programmes of study," [Available] https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/n.

[4] J. E. Sammet, "Programming Languages: History and Future" , Com-munications of the ACM, vol. 15, Number 7, 1972, pp. 601-610.

[5] N. Bak, B. Chang and K. Choi, "Smart Block: A Visual Programming Environment for SmartThings," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, 2018, pp. 32-37.

[6] H. Tsukamoto et al., "Textual vs. visual programming languages in pro-gramming education for primary schoolchildren," 2016 IEEE Frontiers in Education Conference (FIE), Erie, PA, USA, 2016, pp. 1-7.

[7] T. Karvounidis, I. Argyriou, A. Ladias and C. Douligeris, "A design and evaluation framework for visual programming codes," 2017 IEEE Global Engineering Education Conference (EDUCON), Athens, 2017, pp. 999-1007.

[8] Chen and Wang "VIPLE: Visual IoT/Robotics Programming Language Environment for Computer Science Education",2019.

[9] B. Frey, Moving from the Known to the Unknown to Measure the Initial Learnability of Programming Languages,IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) 2017.

[10] J. María Rodríguez Corral, "A Study on the Suitability of Visual Languages for Non-Expert Robot Programmers", Received November 25, 2018, accepted January 23, 2019, date of publication January 29, 2019, date of current version February 14, 2019.

[11] P. Gao, "A New Teaching Pattern Based on PBL and Visual Pro-gramming in Computational Thinking Course", The 14th International Conference on Computer Science and Education (ICCSE 2019) August 19-21, 2019. Toronto, Canada,2019

[12] C. Kyfonidis, "Block C: A block based programming teaching tool to facilitate introductory C programming courses", 2017 IEEE Global Engineering Education Conference (EDUCON).

[13] A. Rao, "Milo: A visual programming environment for Data Science Education", 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).

[14] B.J. Smith, Harry S. Delugach, "Work In Progress - Using a Vi-sual Programming Language to Bridge the Cognitive Gap Between a Novice's Mental Model and Program Code", 2010, Washington, DC 40 the ASEE/IEEE Frontiers in Education Conference