Comprehensive review of a pedagogical innovative programming education tool using quantitative and qualitative analysis

Payal Gohel, PhD Researcher, Saurashtra University, Rajkot, India

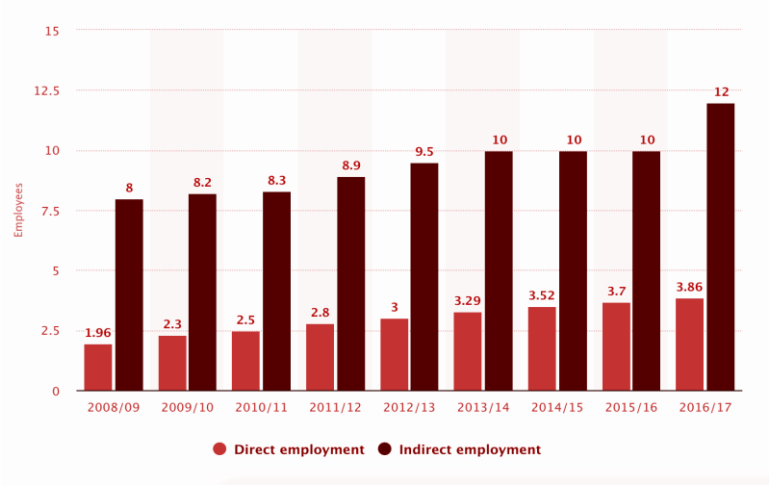
ABSTRACT:

In computer education, computer programming is considered to be the most difficult and challenging subject to learn and understand especially for beginner students. Critical investigation of such causes and how to resolve have been a focused area for the research community since last few decades. Even with the advancement of technology, education sector is still not fully utilising the technology. Therefore, it's time that appropriate teaching methodology should be developed using interactive teaching curriculum to ease students' learning difficulty caused by lack of experience and necessary understanding. There are many theories investigated to enhance knowledge assimilation methodology. This research paper attempted to identify the important design principles by investigating learning theories and identifying the innovative approach engaged in educational teaching. Amongst many, two principal theories, first, theory of Constructivism and second, Cognitive Load theory have been identified and investigated. In addition, a qualitative and quantitative methods were applied to identify the gaps and challenges for beginners. An extensive survey targeting 560 people conducted which helped to identify the trends of current state of programming challenges and what can be done to improve it. 12 open ended interviews were conducted with programming tutors to get more insight into the current state of teaching and difficulties with students. Based on literature and data collection findings, this research proposes a theoretical process framework to design and develop visual programming instruction model.

KEYWORDS: Theory of constructivism, Cognitive Load Theory, Computer Science Education, Visualisation methodology, Smart Interactive education.

1. INTRODUCTION

India has about 3.86 million direct IT employment and about 12 million indirect, related to IT services in 2017[48]. By 2020, India will have 5.2 million developers, a nearly 90% increase, versus 4.5 million in the U.S., a 25% increase through that period. India is set to become Global leader in IT services. India's software development growth rate is attributed, in part, to its population size, 1.2 billion, and relative youth, with about half the population under 25 years of age, and economic growth. So at a time, when India is dubbed as a leader of IT services for the world market, proficiency in programming is pertinent for the next generation of programmer. Learning a programing at the college level becomes an entry point for most professionals However, many researcher have pointed out that student are facing learning difficulties in the conventional way of teaching. There are various reasons cited throughout many research that they don't understand the teacher; they are hesitant to ask any questions; subject is very complicated to learn; it's very boring to sit through entire class and read programming syntax. Even with the advancement of technology, education sector is still not fully utilising the technology. Therefore, it's high time that appropriate learning and teaching methodologies should be developed using interactive teaching mechanism to ease students' learning difficulty, caused by lack of experience and necessary understanding.



The author argued, part of a problem has been a tendency to only look at the technology and not how it is used. Merely introducing technology to the educational process is not enough. The question of what teachers need to know and how students needs to learn in order to appropriately incorporate technology into their teaching has received a great deal of attention in past decade (International Society for Technology in Education, 2000; Zhao, 2003). Our work has been aimed at theoreticians and researchers, as well as practitioners and educators. Use of Multimedia in an teaching environment, in particular with the integration of cognitive load theory has the potential to enhance the student's 'attention span and enable focussed learning. Such methodology has shown tremendous potential for students which help them to guide in the study, minimise the difficulty, and further cut down the mental load. Therefore, this research will focus on investigating the following underlying issues. i) Identify the limitation of conventional way of learning. ii) Identify the use of technology in computer science education. iii) Importance of social cognitive learning in education. iv) Investigate important design principles for pedagogical programming concepts.

Many research has been conducted in educational programming tools to assist the basic learning of computer courses. In literature research, many trends have been summarised and classified into five categories[7].

- Interactive worlds: Development of interactive environments based on manipulating scenes and objects employing basic commands (for example Move, Turn, Back for robots).
- GUI based environments: Coding interface for code creation using visual interaction. Code can be presented in both graphical or textual form (Visual C, VB).
- Abstract environments: Many visual tools that helps to create program with the use of link between different class objects, for example UML, Sequence, Flow charts.
- Object Oriented programming environments: Tool designed for object-oriented programming with visual developmental features. (Visual studio, Eclipse, Netbeans, Android studio).

The primary aim of the proposed research is (a) the critical investigation of different learning theories to outline main principles for coding tool for beginners, (b) the design and development of a novel coding model to demonstrate identified design principles and (c) the verification and validation of the proposed model to assess the enhancement of the learning for young students.. In this paper, the author focuses on the first objective to identify the critical design principles for the effectiveness of the tool to enhance cognitive learning.

2. IDENTIFICATION OF LEARNING THEORIES AND THEORETICAL REVIEW

In the context of this research, learning theories are conceptual frameworks describing how the coding skill is acquired, understood, and maintained during learning stage. From various learning theories, propose research focussed on two critical theories, i) Theory of Constructivism and, ii) Theory of Cognitive Load. Theory of Constructivism focuses on

understanding as an running process of knowledge construction. Therefore, in the context of this research and within the education sector, a student can model their knowledge of a particular area. In the area of programming, the proper guidance should be included by designing a focussed computer education model for coding commands and set of instructions. Furthermore, design principles for computer education model can be reviewed via theory of constructivism. For example: Logo programming environment (1980), one of the most popular programming languages for students, was developed using constructivism. On the other side, Theory of Cognitive Load analyse how student process information and presents a set of design guidelines for organising this information for successful learning for beginners. Therefore, guidelines deduced from the above theories could result in the enhancement of the learnings of these models.

2.1 REVIEW OF CONSTRUCTIVISM THEORY

Constructivism is a theory of knowledge and how can it be achieved by people. Many research studies have referred to constructivism in educational tool. However, one of the first research to study of the implications of applying constructivism was conducted by [10][11]. The author identified the learning difficulties in students when they used a what-you-see-is- what-you-get (WYSIWYG) word processor. The author found that, CS students lacked a cognitive framework to guide them, in order to gain focussed knowledge from their regular interaction with a computer. Second the computer presents an accessible ontological reality. A number of researchers have focused on this area. The InSTEP [12] was developed to provide a constructivist learning experience for computer engineering students as an introductory course. His work demonstrated that students who received feedback from the InSTEP system needed minor help from teachers in learnings than the students who had no feedback from the system. [8] developed a constructivist approach in creating teaching material and guidelines for basic coding classes. He echoed that constructivism theory enhanced students' understanding of the subject material. Then, A pedagogical approach based on a constructivism was presented by [13] for teaching object-oriented concepts for students. The research indicated that students improved their problem solving skills and theoretical understanding of coding concepts.[14], They conducted three case studies on how real life date can be used from constructivism to teach the sorting algorithms, solve puzzles and recognize groups from their multiplication tables. [15] applied constructivism to demonstrate the concept of 'static' in Java program and why it is often hard to understand. Another graphical environment was presented by [16] to guide teachers programmatically produce their teaching material using constructivism theory.

2.2 REVIEW OF COGNITIVE LOAD THEORY

This theory aims to create a conceptual framework of how information is understood intellectually by an individual to achieve greater learning outcomes. Many researchers have undertook this area of research in teaching tools. [20] presented a 4C/ID model for developing instructional programmes for complex skills acquisition. [23][24] presented how cognitive load theory can assist multimedia learning and the design of such software. One of the experiment was conducted using text and then, images and text both at the same time. [25] developed a pedagogical design using cognitive load theory and other theories for teaching Object Oriented Programming concepts. [28] investigated the effect of various strategies on the different learning measurements for cognitive load theory to acquire programming-knowledge especially loops acquired. [26] presented case-study for particular programming concepts. [27]The model was developed using the Cognitive load and Human computer interaction principles. Their review showed that there is commonality between aforementioned principles, namely the reduction of unnecessary load in users' mind. Many years of research in the field of cognitive load theory in various disciplines have been undertaken where researchers have demonstrated some important techniques that minimise cognitive load. These techniques are: the modality, the worked-example, the expertise reversal, the redundancy,the goal-free, the split-attention effect and, the completion problem effect [29][30][31][32]. However, not every researcher found cognitive load theory useful in enhancing learning [33][34][35],there have been some criticism. [36] raised some concerns regarding the effectiveness of

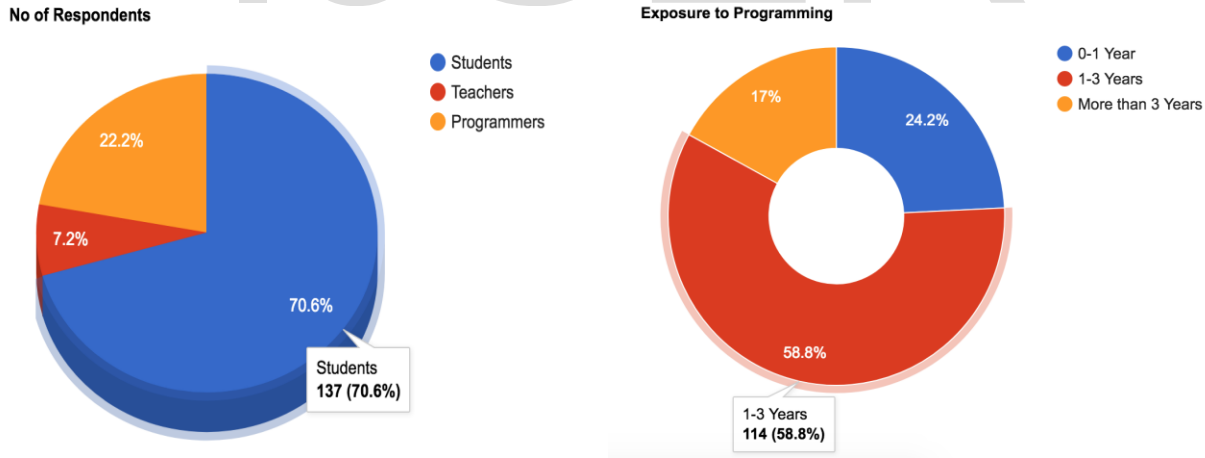
cognitive load theory theory in practical environment. A number of researcher studied the results that contradicted cognitive load theory's predictions and identified some critical methodical problems.

3. SURVEY METHODOLOGY

This survey focused on computer science students, teachers and programmers who have different degree of exposures to programming. The students were identified from wide spectrum, starting from primary level who get exposed to command based programming to undergraduate students who learns procedural, scripting and object oriented programming in their first year itself. Being a beginner, it can be quite difficult to understand the programming if they didn't learn the basics correctly. Hence,The purpose of the survey presented here is to understand the current state of programming teaching for beginners and identifying the underlying pertinent issues with programming tools PL/IDE and what kind of features required for interactive, cognitive and intuitive IDE for beginners. To achieve this purpose, the survey looked at the current usage of programming languages, the complexity and challenges of the current languages, advantages and disadvantages of the available programming tools from students, teachers and programmer's perspective. This survey also includes the view of teachers who are the representative of delivering programming courses and to receive their view about what they think about the current mode of learning. The results will provide directions in research, development, training, and strategies that will respond to the needs of this industry and academic studies. The survey was sent by e-mail, posted on social network and social messaging through whatsapp to a statistical sample of 580 people. This sample of group was split in four categories: academic researchers, teachers, computing students, and programmers. The sample was selected at random in order to ascertain reliability about the population. The prior consent was taken and the answers provided were kept confidential and used for statistical purposes and released in aggregate form only.

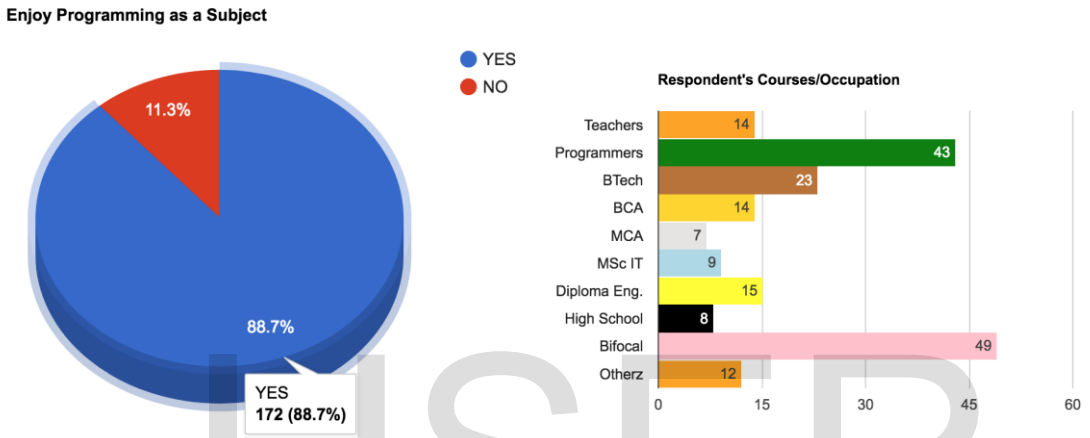
3.1 Quantitative results and discussions

Most conducted survey indicates, the return rates for Internal surveys will generally receive a 30-40% response rate on average, compared to an average 10-15% response rate for external surveys.

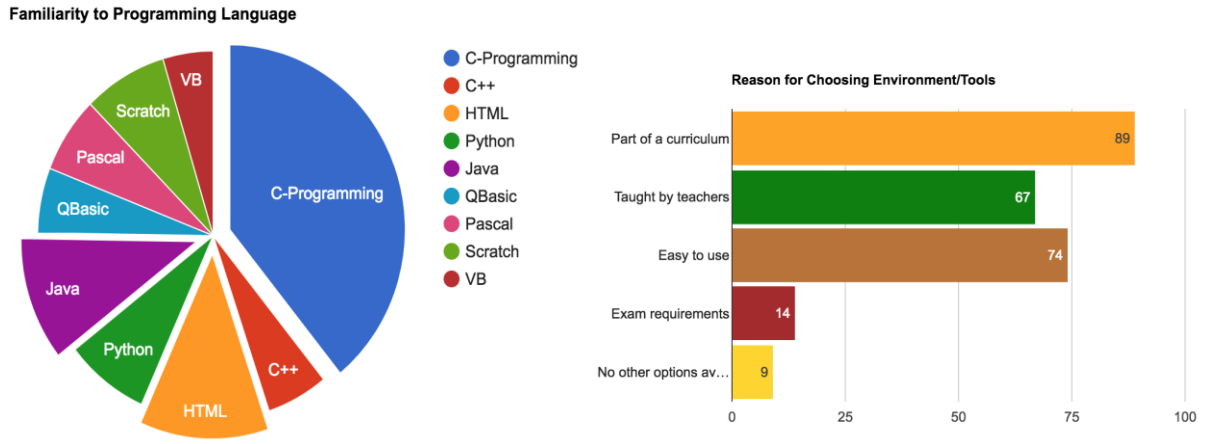


It also depends on the selected demographics, enthusiasm towards the topic, focused participations etc; The findings presented here are based on an overall 34.6% return rate, 194 respondents. Even though an average response was obtained, the findings of the survey presented a clear trend towards the current programming tools and its complexity while learning for beginners, useful information about the respondent's perspective and addressed specific concerns while learning programming. The results of the survey are presented within the following four topics: Exposure to programming, List of current PL/IDE usage, Advantages and disadvantages of current programming tools; and research directions. The distribution of responses with respect to the three categories of targeted population is shown in Figure 1.

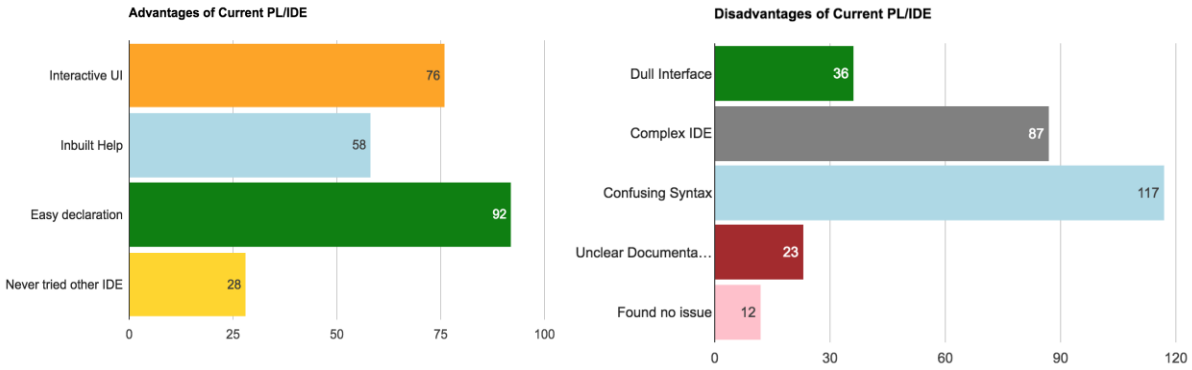
As it shows, students participated overwhelmingly amounting to 70.6% that gave a clear trends of their difficulties and challenges in learning programming in existing teaching methodologies. There were also 22.2% programmers who gave us some interesting insight about present IDEs and what can be done to improve the current state. Among those respondents, more than 58% population were exposed to programming between 1-3 years that shows most of the respondents were either familiar with the programming or had some knowledge about it. Next chart spell out the languages and environment they were exposed for the first time in their life. This says something about today's teaching curriculum that are exposing the students to complex programming structure without building their base and creating a basic understanding of natural programming language. Majority of the respondent enjoyed programming as a subject, almost 89% but found it boring while learning in class .



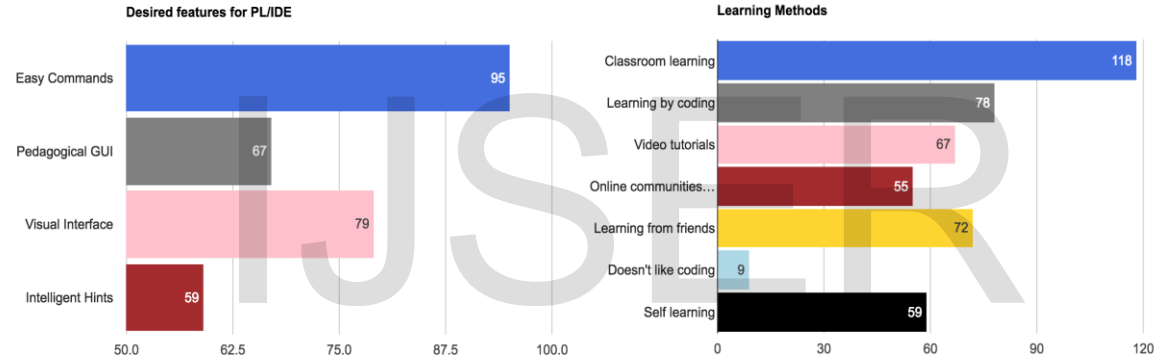
While selecting the population, authors decided to select students from School to diploma, bachelors and master to teachers and programmers. It gave a wider perspective about how they see programming as a subject and how to develop an effective methodology for beginners. Surprisingly maximum respondent cited part of a curriculum as their reason for learning programming, some said they are using particular IDE because their teachers said so while some minority suggested they are learning only for exams.



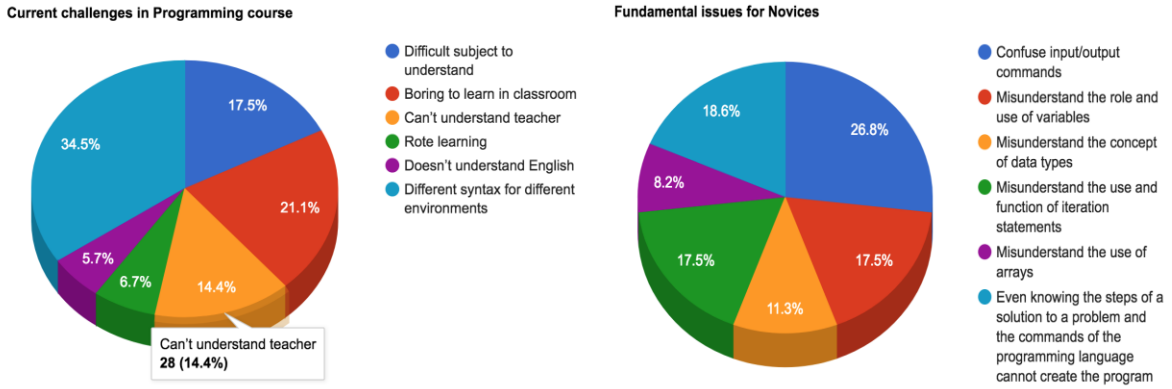
However, there were majority of the respondent who enjoyed programming as a subject and found Interactive UI and Easy declaration one of the important features for using any specific PL/IDE. One of the major revelation was, more than 60% people found syntax semantics in different environment confusing including complex IDE and dull interface. They thought aforementioned are the main disadvantages..



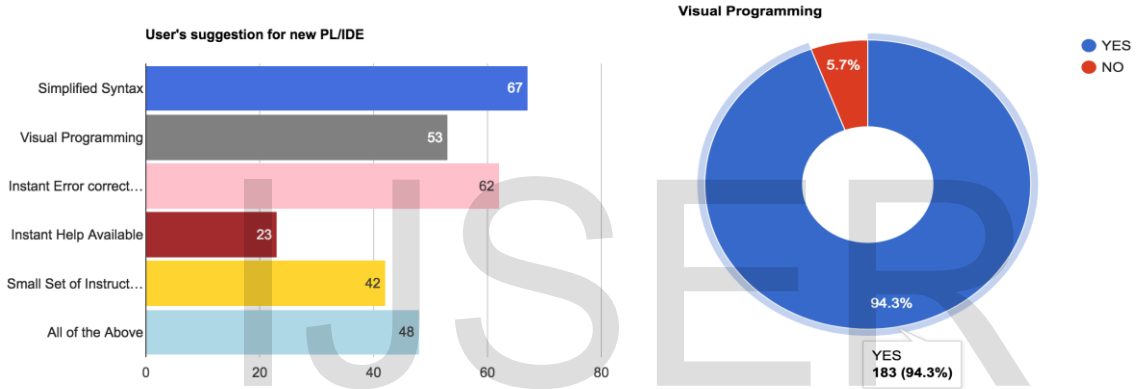
When the author asked for desired features for proposed PL/IDE, majority suggested to have natural language easy commands, step by step instruction to write program and visual interface and some desired intelligent hints while writing codes. When the author asked about their current mode of teaching majority answered, they are learning in traditional classroom setup or in labs or learning from friends. Majority of the programmers said they prefer to learn from video tutorials or getting help from online communities. They were quite uptodate with their knowledge and wanted instantaneous help.



In another question, respondents addressed their challenges in current course. Different syntax for different programming environment was major complain from almost 35% respondent while 21% opined learning in classroom is the most boring way to learn programming. There were 15% who said, they don't understand teacher at all. Most beginners have difficulty in learning a programming in C and Java environment especially when they are beginning to learn. Most students found difficulty in basic understanding of difference between data types and variable. Hence, having a instruction in natural language for the beginners can be easy to understand and program. Many participants said they knew the entire logic and steps for the program but still they couldn't create the program. They were missing a basic understanding to code.



When the author asked for important features they would like to have in proposed PL/IDE, majority responses were simplified syntax, instant error detection and visual way of coding. It seems they were quite clear about what they would prefer in a programming environment. Almost unanimously all participant agreed that a Icon based visual programming is a better way to get exposed to programming as it is cognitive and constructive way to learn it.



4. QUALITATIVE RESULTS AND DISCUSSIONS

An extensive literature review helped in identifying the design principles. However, little information is available in the international literature about the current status of introductory programming education in India. The author interviewed 12 teachers and lecturers from school and colleges who were teaching programming to beginners. As part of this research methodology, along with survey open ended interviews were conducted to get insight into the teacher's challenges and took their opinion about why it exist in today's techno savvy time and what should be done to resolve this issue. Teachers were informed about the aim of the research. During these interviews, author was focussed specific issues i) the software tool that participants have used to teach programming and their opinion on its/their effectiveness; ii) the problems that beginners face during introductory programming courses, according to participants' experience; and iii) the desired features for an educational programming tool for teaching and learning introductory programming. The answers of all respondents were categorised per question. Each category was created based on the answers of the respondents. More specifically, analysis was conducted either by identifying common answers and emerging patterns or by synthesising all responses to produce an aggregated/combined response for a category.

During the discussion, majority teachers said that a set of general-purpose as well as educational programming languages used were QBasic, Fortran, C, Visual Basic and Java. The main disadvantages of these languages are that the majority of these programming languages (i) are too complex and (ii) provide features beyond those that are required for pedagogical purposes. In particular, they lack a user- friendly environment, they are in the English

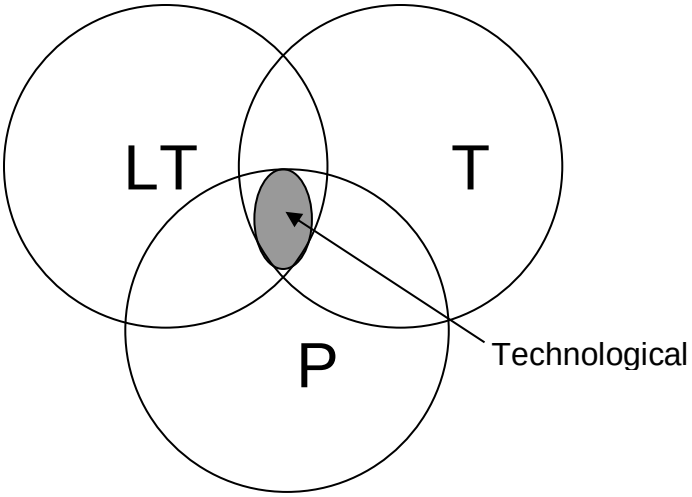
language and their set of instructions is too large. Thus, an unnecessary complexity is imposed by the use of these tools. Many tutors said even when beginners know the steps of a solution and the commands of a programming language they are unable to structure the final program. They think this happens because beginners seem to lack the knowledge and experience of using programming commands to translate the steps of the solution into a program. Other major problems that beginners have are distinguishing the commands they should use in a program and lacking knowledge of how data types, arrays and functions should be used. Most of the interviewees commented that knowledge of mathematics and computer architecture could lead novices towards a more analytical and structured way of thinking, which could help them to conceptualise and create their own programs. According to the interviewees, an educational programming language suitable for teaching introductory programming should have a set of desirable characteristics in order to serve its educational purpose. The results of the interviews revealed that this set should include a visual environment, which would be supportive and simple to learn and use, a small and simple set of instructions, easy syntax close to natural language, and a visual representation of programming elements. Concisely, a new programming language and integrated development environment should support the following features:

- Graphical development environment, which will help and guide students during the creation of programs;
- Simple syntax;
- Semantics that can be easily understood by non-programmers;
- Visualisation of basic programming structures and examples;
- A small set of commands; and
- Documentation of the set of instructions and supported features by the programming language and development environment.

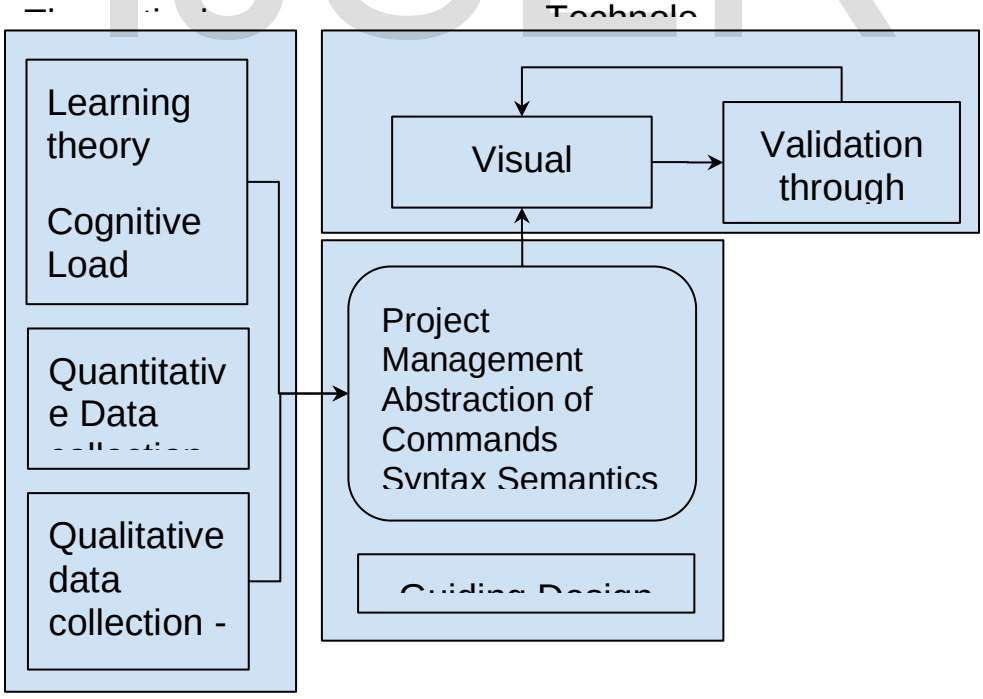
Overall, It took author 14-16 weeks to complete this process and gather 194 participants data that helped to identify the underlying issues with the current mode of learning especially for beginners. Most respondent found traditional way of learning tedious and confusing and desired to have a tool that would help the coding visually in simplified commands and instant error detection. The results from Surveying and Open ended interviews echoed each other's sentiment and identified the root causes.

5. PROPOSED THEORETICAL PROCESS FRAMEWORK

The basis of our framework is the understanding that learning and teaching is a highly complex activity that draws on many kinds of knowledge (Chavada R. et al, 2012). In this particular context, the implications of developing a framework go beyond a coherent way of thinking about learning theories, multimedia and pedagogy integration. Many scholars have argued that knowledge about technology cannot be treated as context-free and that good methodology requires an understanding of how technology relates to the pedagogy and theoretical contents (Hughes, 2005; Lundeberg, Bergland, Klyczek, & Hoffman, 2003; Margerum- Leys & Marx, 2002; Neiss, 2005; Zhao, 2003).



What sets our approach apart is the specificity of our articulation of these relationships between content, pedagogy, and technology. In practical terms, this means that apart from looking at each of these components in isolation, we also need to look at them in pairs: pedagogical content knowledge (PCK), technological content knowledge (TCK), technological pedagogical knowledge (TPK), and all three taken together as technological pedagogical content knowledge (TPCK). This is similar to the move made by Shulman, in which he considered the relationship between content and pedagogy and labeled it pedagogical content knowledge. In our case, a similar consideration leads us to three pairs of knowledge intersection and one triad. One of the pairs, pedagogical content knowledge, was introduced and articulated by Shulman, but we introduce two new pairs and one new triad. Thus, the following elements and relationship are important in the framework we propose.



Authors argued that a conceptually based theoretical framework about the relationship between technology and pedagogical learning can transform the conceptualization and the practice of learning and teaching. It can also have a

significant impact on the kinds of research questions that we explore. How this framework has guided our research and analysis of the effectiveness of our pedagogical approach. The aim of this research is to propose a theoretical conceptual framework to develop an interactive pedagogical multimedia tool for beginners to learn programming using Cognitive load and Constructivism theories. An additional objective of this work to offer a theoretical framework that integrates the pragmatic, technological and the theoretical components together. The discussion of these learning theories and the identification of critical learning challenges of computing students in their programming subject via quantitative and qualitative survey provides the basis to develop a conceptual process framework. Figure X present the process framework. This framework assists to identify, employing seven guiding design principles:

Project Management	Organising the project structure according to programmers needs; creating and editing source files, providing structure views for quick and easy navigation, and so on; refactoring to enable fast and reliable code restructuring operations; compile and run process allowing the execution of application within the IDE;
Abstract programming commands	Abstraction is a mechanism which focuses only on relevant information for end users without understanding the unnecessary details. Programmers use abstract definition while create programs, for example: Programming use abstraction to present coding with a certain level of details[44].
Common syntax semantics	Even the experience programmers face syntax confusion when they are working on different environments. So the proposed tool should present natural language syntax which is easier to follow and code [38][39][40][41]. The syntax of a programming language is a structure or the grammar of the language. It is a set of standard instruction on how to write code including correct spelling, order of commands and punctuation marks[42]. Semantics is the meaning of PLs sentences [43]. Semantic error occurs because novices fails to understand the concept or functionality of program statements.
Visual representation	Visualisation means IDE that uses visual symbols to create programs and that symbols can be in form of flow diagram, icons, tables or forms.
Intelligent assistant mechanism	The most software provides a small set of guidelines or programming instructions to kick-start the learning [45]. An educational PL could have set of instructions either in mini or a sub language. Former supports only the fundamental programming constructs while later supports only the commands that define basic programming constructs like variables, output, looping and branching.
Analyse and present error messages	Compiler should have error checkers that can provide understandable and informative error message before execution. [46] suggested that error messages should be specific, user friendly and easy to understand for beginners.
An interactive Pedagogical IDE	It will be a visual expression for programming. A visual interaction could make IDEs more engaging to students. Instant response and feedback based graphical interfaces could be used for programming including pedagogical interface, the editor for declaring variables, functions, and the debugger to display errors or any messages [47].

6. CONCLUSION AND FUTURE WORK

Many researchers suggested that learning programing for novices has always been demanding and frustrating. There are various reasons identified throughout many researches including sometimes they don't understand the teacher; they are hesitant to ask any questions; subject is very complicated to learn; it's very boring to sit through entire class.

Therefore, to investigate such issues, this paper conducted an extensive review of two important learning theories that have a greater impact on academic learning in computer programming, i) theory of constructivism and, ii) Constructive Load Theory. After the thorough and critical review, this research identified and presented seven design principles for programming tool for beginners. As discussed, and described, these principles are (i) Project management, (ii) Abstraction of commands, iii) Common syntax semantics, (iv) Visual presentation, vi) User guidelines, (vii) Analyse and present error messages and (viii) An interactive IDE. From the review of the past literature, some vital gaps in knowledge are identified. First, many researchers have employed constructivism and cognitive load theory into their research, but this research proposes to integrate both the learning theories for the design of educational programming tool for beginner students. Second, many researchers discussed the set of design principles using aforementioned theories but not many researchers have measured the impact of such educational programming tool empirically. Then, to gather quantitative data survey was conducted, targeting 560 population and to collect qualitative data open ended interviews were conducted from 12 tutors. The results of the interviews revealed that this set should include a visual environment, which would be supportive and simple to learn and use, a small and simple set of instructions, easy syntax close to natural language, and a visual representation of programming elements. Based on the qualitative and quantitative data collection and critical investigation of no of research, this paper proposes a theoretical process framework of a novel methodology as a validation of these proposed principles. Furthermore, the future work will contribute towards the development of the programming tool imbibing proposed design principles and test the model using theory of constructivism. verification of the model will be done through teaching expert to develop a robust and novel methodology in line with the proposed principles.

REFERENCES

1. Murnane, J. S. (1993). The psychology of computer languages for introductory programming courses. New Ideas in Psychology, 11(2), 213-228.
2. Goldweber, M., Bergin, J., Lister, R. & McNally, M. (2006). A comparison of different approaches to the introductory programming course. 8th Australasian Conference on Computing Education - Volume 52, Hobart, Australia.
3. Powers, K. (2004). Teaching computer architecture in introductory computing: Why?And how? 6th Australasian Conference on Computing Education - Volume 30, Dunedin, New Zealand
4. McIver, L. & Conway, D. (1996). Seven deadly sins of introductory programming language design. 1996 International Conference on Software Engineering: Education and Practice (SE:EP '96).
5. Kelleher, C. & 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
6. Georgatos, F. (2002). How applicable is Python as first computer language for teaching programming in a pre-university educational environment, from a teacher's point of view? MSc thesis, Universiteit van Amsterdam, Amsterdam.
7. Gross, P. & Powers, K. (2005b). Work in progress - a meta-study of software tools for introductory programming. 35th ASEE/IEEE Frontiers in Education, Indianapolis, IN.
8. Gonzalez, G. (2004). Constructivism in an introduction to programming course.Journal of Computing Sciences in Colleges, 19(4), 299-305
9. Mannila, L. & de Raadt, M. (2006). An objective comparison of languages for teaching introductory programming. 6th Baltic Sea conference on Computing education research: Koli Calling 2006, Uppsala, Sweden.
10. Ben-Ari, M. (1998). Constructivism in computer science education. ACM SIGCSE Bulletin 30(1), 257-261.
11. Ben-Ari, M. (2001). Constructivism in computer science education. Journal of Computers in Mathematics and Science Teaching, 20(1), 45-73.
12. Odekirk-Hash, E. & Zachary, J. L. (2001). Automated feedback on programs means students need less help from teachers. SIGCSE Bulletin, 33(1), 55-59.
13. Hadjerrouit, S. (2005). Constructivism as guiding philosophy for software engineering education. SIGCSE Bulletin, 37(4), 45-49.
14. Beynon, M. (2009). Constructivist computer science education reconstructed.Innovation in Teaching And Learning in Information and Computer Sciences,8(2), 73-90.

15. Milner, W. W. (2010). Concept development in novice programmers learning Java. PhD thesis, The University of Birmingham, Birmingham.
16. Lee, Y.-J. (2011). Empowering teachers to create educational software: A constructivist approach utilizing etoys, pair programming and cognitive apprenticeship. Computers & Education, 56(2), 527-538
17. Van Merriënboer, J. J. G. (1990a). Instructional strategies for teaching computer programming: Interactions with the cognitive style reflection-impulsivity. Journal of Research on Computing in Education, 23(1), 45-53.
18. Van Merriënboer, J. J. G. & Krammer, H. P. M. (1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. Instructional Science, 16(3), 251 -285.
19. van Merriënboer, J. J. G. & Paas, F. (1990). Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. Computers in Human Behavior, 6(3), 273-289
20. Van Merriënboer, J. J. G. & Kirschner, P. A. (2007). Ten steps to complex learning: A systematic approach to four-component instructional design. Mahaw, New Jersey: Erlbaum.
21. Garner, S. (2002a). Colors for programming: A system to support the learning of programming. Informing Science & IT Education Conference (InSITE), Cork, Ireland.
22. Garner, S. (2002b). Reducing the cognitive load on novice programmers. World Conference on Educational Multimedia, Hypermedia and Telecommunications 2002, Denver, Colorado, USA.
23. Mayer, R. E. & Moreno, R. (2002). Aids to computer-based multimedia learning. Learning and Instruction, 12(1), 107-119.
24. Mayer, R. E. & Moreno, R. (2003). Nine ways to reduce cognitive load in multimedia learning. Educational Psychologist, 28(1), 43-52.
25. Caspersen, M. E. & Bennedsen, J. (2007). Instructional design of a programming course: A learning theoretic approach. 3rd international workshop on Computing education research, Atlanta, Georgia, USA.
26. Gray, S., Clair, C. S., James, R. & Mead, J. (2007). Suggestions for graduated exposure to programming concepts using fading worked examples. 3rd international workshop on Computing education research, Atlanta, Georgia, USA.
27. Hollender, N., Hofmann, C., Deneke, M. & Schmitz, B. (2010). Integrating cognitive load theory and concepts of human-computer interaction. Computers in Human Behavior, 26(6), 1278-1288.
28. Abdul-Rahman, S.-S. & du Boulay, B. (2014). Learning programming via worked examples: Relation of learning styles to cognitive load. Computers in Human Behavior, 30(0), 286-298.
29. Chong, T. S. (2005). Recent advances in cognitive load theory research:Implications for instructional designers. Malaysian Online Journal of Instructional Technology, 2(3), 106-117.
30. Sweller, J., van Merriënboer, J. J. G. & Paas, F. (1998). Cognitive architecture and instructional design. Educational Psychology Review 10(3), 251 -296.
31. van Merriënboer, J. J. G. & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. Educational Psychology Review, 17(2), 147-178.
32. van Mierlo, C., Jarodzka, H., Kirschner, F. & Kirschner, P. A. (2011). Cognitive load theory and e-learning. In Z. Yan (Ed.), Encyclopedia of Cyberbehavior: IGI Global.
33. Ayres, P. & van Gog, T. (2009). State of the art research into cognitive load theory. Computers in Human Behavior, 25(2), 253-257. Baddeley, A. (1992). Working memory.
34. Paas, F., Van Gog, T. & Sweller, J. (2010). Cognitive load theory: New conceptualizations,specifications, and integrated research perspectives. Educational Psychology Review, 22(2), 115-121.
35. Verhoeven, L., Schnotz, W. & Paas, F. (2009). Cognitive load in interactive knowledge construction. Learning and Instruction, 19(5), 369-375.
36. Moreno, R. (2006). When worked examples don't work: Is cognitive load theory at an impasse? Learning and Instruction, 16(2), 170-181.
37. Schnotz, W. & Kürschner, C. (2007). A reconsideration of cognitive load theory. Educational Psychology Review, 19(4), 496-508.
38. Gomes, A. & Mendes, A. J. (2007). Learning to program - difficulties and solutions. International Conference on Engineering Education – ICEE 2007, Coimbra, Portugal.
39. Guibert, N., Girard, P. & Guittet, L. (2004). Example-based programming: A pertinent visual approach for learning to program. Working conference on Advanced visual interfaces, Gallipoli, Italy.
40. Simon, B., Fitzgerald, S., McCauley, R., Haller, S., Hamer, J., Hanks, B. et al. (2007). Debugging assistance for novices: A video repository. ITiCSE on Innovation and technology in computer science education, Dundee, Scotland.
41. Teague, D. & Roe, P. (2008). Collaborative learning: Towards a solution for novice programmers. 10th conference on Australasian computing education Volume 78, Wollongong, NSW, Australia.

42. Hristova, M., Misra, A., Rutter, M. & Mercuri, R. (2003). Identifying and correcting Java programming errors for introductory computer science students.SIGCSE Bulletin, 35(1), 153-156.
43. Wiedenbeck, S., Ramalingam, V., Sarasamma, S. & Corritore, C. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. Interacting with Computers, 11(3), 255-282.
44. Pane, J. F. & Myers, B. A. (1996). Usability issues in the design of novice programming systems (school of computer science technical report cmu-CS- 96-132). Pittsburgh, PA: Carnegie Mellon University.
45. Dagdilelis, V., Satratzemi, M. & Evangelidis, G. (2004). Introducing secondary education students to algorithms and programming. Education and Information Technologies, 9(2), 159-173.
46. raver, V. J. (2010). On compiler error messages: What they say and what they mean. Advances in Human-Computer Interaction, 2010, 1 -26.
47. Romero, P., du Boulay, B., Robertson, J., Good, J. & Howland, K. (2009). Is embodied interaction beneficial when learning programming? 3rd International Conference on Virtual and Mixed Reality: Held as Part of HCI International 2009, San Diego, CA.
48. Press Information Bureau, Ministry of Electronics & IT, Government of India (2017), http://pib.nic.in/newsite/PrintRelease.aspx?relid=162046
49. International Society for Technology in Education. (2000). National educational technology standards for students: Connecting curriculum and technology. Eugene, OR: Author
50. Zhao, Y. (Ed.). (2003). What teachers should know about technology: Perspectives and practices. Greenwich, CT: Information Age.
51. Hughes, J. (2005). The role of teacher knowledge and learning experiences in forming technology-integrated pedagogy. Journal of Technology and Teacher Education, 13(2), 277–302.
52. Lundeberg, M. A., Bergland, M., Klyczek, K., & Hoffman, D. (2003). Using action research to develop preservice teachers' beliefs, knowledge and confidence about technology [Electronic version]. Journal of Interactive Online Learning, 1(4). Retrieved June 29, 2004, from http://ncolr.uidaho.com/jiol/archives/2003/spring/toc.asp
53. Neiss, M. L. (2005). Preparing teachers to teach science and mathematics with technology: Developing a technology pedagogical content knowledge. Teaching and Teacher Education, 21(5), 509–523.
54. Rajiv Chavada, Nashwan Dawood, Mohamad Kassem (2012). Construction workspace management: the development and application of a novel nD planning approach and tool, ITcon Vol. 17, pg. 213-236, http://www.itcon.org/2012/13