A statistical evaluation of an interactive elementary programming model – A pedagogical approach

Dr. Payal Gohel, Researcher Scholar,
Saurashtra University, Rajkot, India
Payalgohel.ce@gmail.com

Dr. Kishor Aatkotia,
Professor, Saurashtra University, Rajkot, India

ABSTRACT - Indian's target to achieve a 5 trillion economy by 2024 will require them to train a high skilled labor force. India being the youngest country in the world, it has to use leverage of its top position in computing and software services. In computing education, coding is considered to be the most challenging subject to understand especially for new students. The author conducted literature review focusing on how to overcome this challenge and how to resolve it have been a focused area for the researchers since last decade. The fragmented use of technology and multimedia has hampered realizing the potential of an education sector. Therefore, it's time that appropriate teaching methodology should be developed using pedagogy-based curriculum to ease students' learning difficulty caused by lack of experience. Amongst many, two principal theories, first, theory of Constructivism and second, Cognitive Load theory have been identified and investigated. Then, qualitative and quantitative methods were applied to identify the trends and learning challenges for beginners. An extensive survey was conducted targeting 360 population which helped to identify the current state of coding challenges and what can be done to improve it. Then, the 12 semi-structured interviews were conducted with lecturers to gain more insight into the 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 a visual coding instruction model in the realms of technology, pedagogy and theoretical concepts. Then, the evaluation and assessment of the proposed research model was conducted using quasi-experimental designs to understand the impact of it. Many factors were considered while analyzing the data collected through experiments with 60 odd first time coding students. After the rigorous data quantification, a significant improvement was observed in VProEn users than Turbo C users earlier but in the longer run no significant evidence was detected.

KEYWORDS: Theory of constructivism, Cognitive Load Theory, Computer Science Education, Visualization methodology, Smart Interactive technology, Statistical Analysis.

(Received June 1st, 2020/ Accepted June 11st, 2020)

1. INTRODUCTION

As per recent government data, India's superpower in IT and computing services is responsible for about 3.86 million direct IT employment and about 12 million indirect [1]. 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 (Figure 1). India's software development growth rate is largely attributed, in part, to its educated population size, over a billion with about half the population under 25 years of age and highly ambitious competitive youth. Hence, at a time when India is considered a leader of computing services for the world market, skill in coding is important for the next generation of programmers. Learning programming at the elementary and college level becomes an entry point for most professionals. However, many researchers pointed out that students are facing learning difficulties in the conventional way of teaching. There are many reasons including lack of experience in teaching; complex subject to learn;

difficult to read coding syntax. Even with the advancement of technology, the education sector is still not fully realizing the power of technology.

Therefore, it's high time that we evolve learning and teaching methodologies using interactive approaches to ease students' learning difficulty, caused by lack of experience. The author believed, main issue with the education sector is to only look at the technology and not how it is being used. Smart class doesn't make smart students. Then, the question arises about what tutors need to deliver and how students' should learn in order to appropriately incorporate technology into their teaching has received a great deal of interest in last decade [2], [3], [4]).

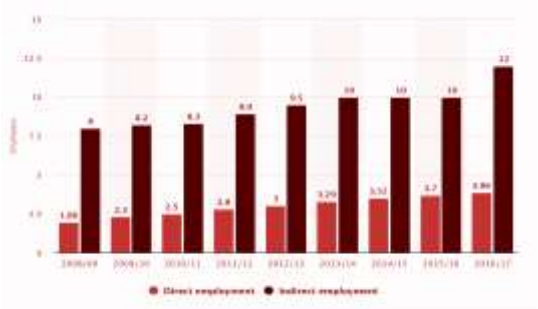

Figure 1: Direct/Indirect IT employment period (2008-09 to 2016-17) - Press Information Bureau, India

The important objectives of the proposed research are (a) the critical exploration of different learning theories to identify main principles for coding tool for beginners, (b) quantitative and qualitative data collection to identify the trend through empirical data, (c) the design and development of a novel framework to demonstrate identified design principles and (d) the evaluation and assessment of the proposed model to verify the enhancement of the learning in new students.

2. EXPLORE LEARNING THEORIES

This research work has been aimed at theoreticians and researchers, as well as practitioners and educators. Use of interactive pedagogy-based teaching environments, in particular with the integration of learning theories, have the potential to increase the student's attention span and enable focused learning. From various learning theories, the proposed research focused on two important theories, i) Constructivism and, ii) Cognitive Load Theory [5]. Author believes integration of such learning concepts in modelling could show tremendous potential for students which help them while self-study, and further cut down the mental load.

2.1 CONSTRUCTIVISM THEORY

Constructivism is a theory of knowledge and how it can be achieved by people. Many research studies have referred to constructivism as an educational tool. However, one of the first research to study of the implications of applying constructivism was conducted by [7][8]. 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 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 [9] 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 learning than the students who had no feedback from the system. [10] 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 constructivism was presented by [11] for teaching object-oriented concepts for students. The research indicated that students improved their problem-solving skills and theoretical understanding of coding concepts.[12], They conducted three case studies on how real-life data can be used from constructivism to teach the sorting algorithms, solve puzzles and recognize groups from their multiplication tables. [13] applied constructivism to demonstrate the concept of 'static' in Java programs and why it is often hard to understand. Another graphical environment was presented by [14] to guide teachers programmatically produce their teaching material using constructivism theory.

2.2 COGNITIVE LOAD THEORY

This theory aims to create a conceptual framework of how information is understood mentally by an individual to achieve greater learning outcomes. [15] presented a 4C/ID model for developing instructional programs for complex skills acquisition. [16][17] presented how cognitive load theory can assist multimedia learning and the design of such software. [18] developed a pedagogical design using cognitive load theory and other theories for teaching Object Oriented Coding concepts. [19] investigated the effect of various strategies on the different learning measurements for cognitive load theory to acquire

coding-knowledge especially loops acquired. [20] presented case-study for particular coding concepts. [21] The model was developed using the Cognitive load and Human computer interaction principles. However, not every researcher found cognitive load theory useful in enhancing learning [22][23][24], there has been some criticism. [25] raised some concerns regarding the effectiveness of cognitive load theory in a practical environment.

3. SURVEY METHODOLOGY

The aim of the survey is to understand the current state of coding learning for beginners and identifying the underlying pertinent issues with coding tools PL/IDE and what kind of features required for interactive, cognitive and intuitive IDE for beginners [26]. The results will provide directions in research, development, training, and strategies that will respond to the needs of the academic courses. The survey was sent by email, posted on social network and group messaging through WhatsApp to a statistical sample of 380 people. The sample was selected at random 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 DATA QUANTIFICATION OF SURVEY

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 coding tools and its complexity while learning for beginners, useful information about the respondent's first hand view and identified specific concerns while learning coding. The results of the survey are classified within the following four topics: Exposure to coding, List of current PL/IDE usage, Advantages and disadvantages of current coding tools; and research directions.

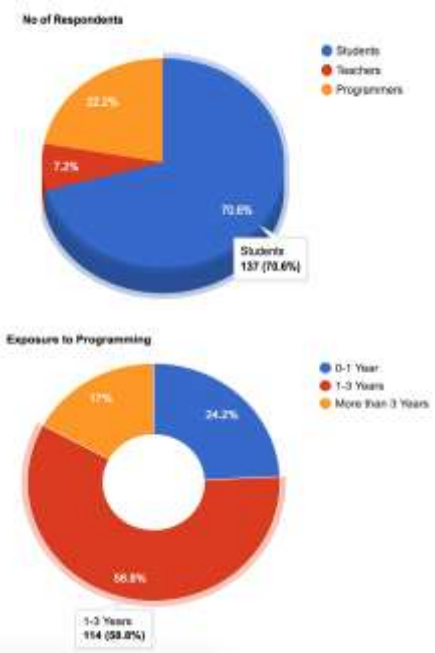

Figure 2 : Questionnaire Population Classification

As it presents (refer to figure 2), students participated overwhelmingly amounting to 70% that gave a clear trend of their difficulties and challenges in learning coding in existing teaching methodologies. There were also 22.2% programmers who provided us some insight about present IDEs and what can be done to improve the current state. Among those respondents, more than 58% of the population were exposed to coding between 1-3 years that shows most of the respondents were either familiar with the coding or had some knowledge about it.

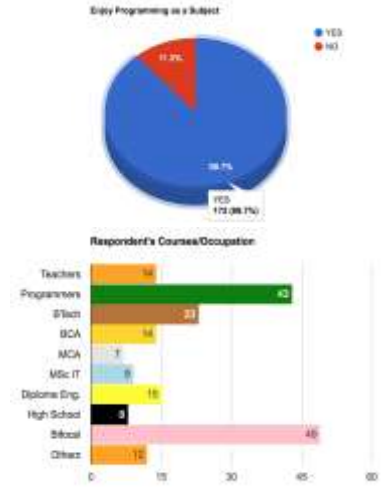

Figure 3 : i) Coding as a Subject, and ii) Respondent's domain

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 coding as a subject and how to develop an effective methodology for beginners (figure 3). Next chart spells out the languages and environment they were exposed to for the first time in their life. It explains today's teaching curriculum that is exposing the students to complex coding structures without building their base and developing a basic understanding using natural coding language. Majority of the respondents were excited to learn coding, almost 89% but found it boring while learning in class (figure 4). Surprisingly maximum respondents cited part of a curriculum as their reason for learning coding, some said they are using particular IDE because their teachers said so while some minority suggested they are learning only for exams

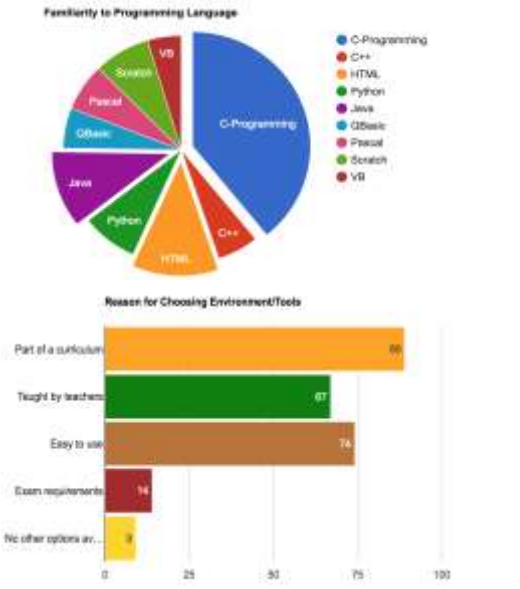

Figure 4 : i) Exposure to Coding and, ii) Reason for choosing Environment/Tools

However, there were the majority of the respondents who enjoyed coding 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 revelations was, more than 60% people found syntax semantics in different environments confusing including complex IDE and dull interface (figure 5). They thought aforementioned are the main disadvantages for the beginners.

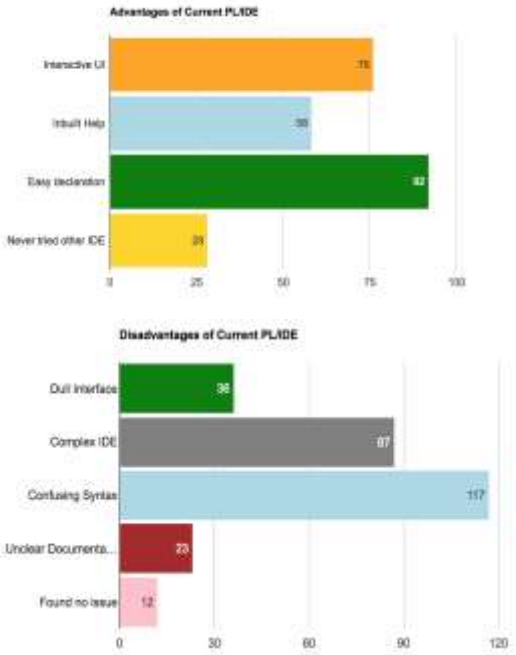

Figure 5 : Advantages and Disadvantage of Current PL/IDE

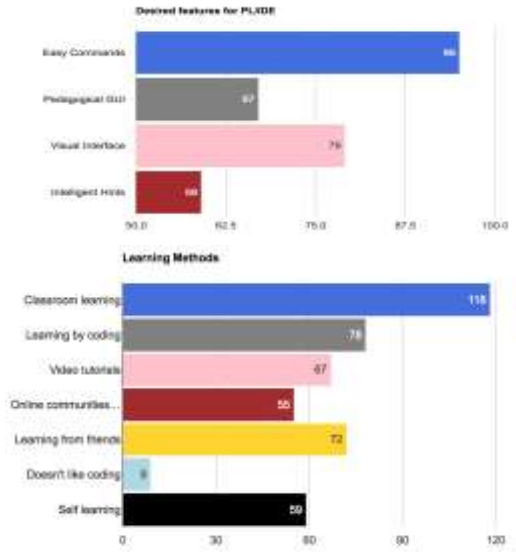

Figure 6 : i) Desired features of the PL/IDE and ii) Learning Methods for coding students

When the author asked for desired features for proposed PL/IDE, majority suggested having natural language easy commands, step by step instruction to write program and visual interface and some desired intelligent hints while writing codes (refer to figure 6). When the author asked about their current mode of teaching, the 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 up to date with their knowledge and wanted instantaneous help.

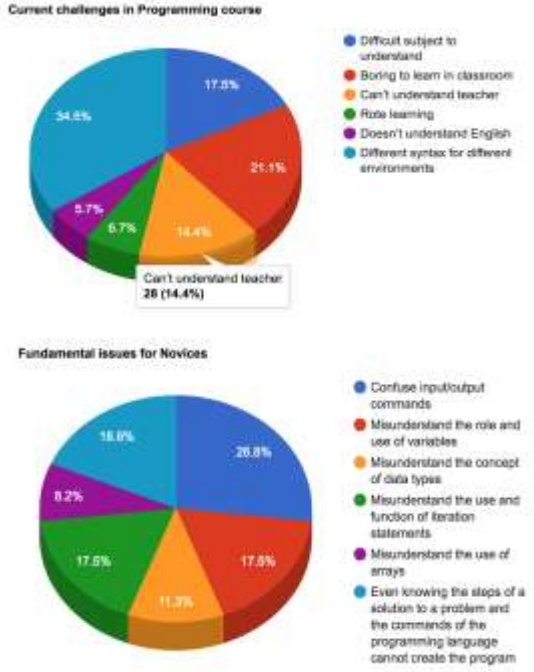

Figure 7 : i) Current Challenges in Coding courses and ii) Fundamental issues for novices

In another question, respondents addressed their challenges in the current course. Different syntax for different coding environments was a major complaint from almost 35% respondents while 21% opined learning in the classroom is the most boring way to learn coding. There were 15% who said they didn't understand the teacher at all. Most beginners have difficulty in learning coding in C and Java environments, especially when they are beginning to learn (refer to figure 7). Most students found difficulty in basic understanding of the difference between data types and variables. Hence, having an 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 of code.

When the author asked for important features they would like to have in the proposed PL/IDE, majority responses were simplified syntax, instant error detection and visual way of coding (refer to figure 8). It seems they were quite clear about what they would prefer in a coding environment. Almost unanimously all participants agreed that a Icon based visual coding is a better way to get exposed to coding as it is a cognitive and constructive way to learn it.

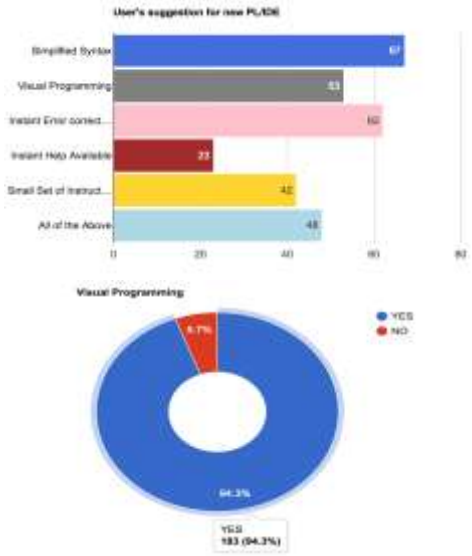

Figure 8 : i) Respondent's suggestion for new PL/IDE and, ii) User's preference for Visual Coding

3.2 OPEN-ENDED INTERVIEW DISCUSSIONS

During the research, the author found little information was available in the international literature about the current status of introductory programming education in India. India being the largest software developer community, author strongly believes that it is pertinent to take the stock of the situation in India regarding programming. The author interviewed 12 lecturers who were teaching coding to beginners. As part of this research methodology, open ended interviews were conducted to get insight into the teacher's challenges and took their opinion about why it exists 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 focused on specific issues:

i) The software tool that participants have used to teach coding and their opinion on its effectiveness;
ii) The problems that beginners face during introductory coding courses, as per their experience;
iii) The desired features for an educational coding tool for teaching and learning introductory coding.

The answers of all respondents were categorized per question. More specifically, analysis was conducted either by identifying common answers and emerging patterns or by synthesizing all responses to produce an aggregated response for a category.

During the discussion, majority teachers said that educational coding languages used were QBasic, C, Visual Basic and Java. The main disadvantages of these languages are that the majority of these coding 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, and their set of instructions is too large. Thus, an unnecessary complexity is imposed by the use of these tools. Many tutors said that many students know the steps of a solution and the commands, still they are unable to structure the final program. They think this happens because beginners seem to lack the knowledge and experience of using coding 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. According to the interviewees, an educational coding language suitable for teaching introductory coding 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 simple to learn and use, a small and simple set of instructions, easy syntax close to natural language, and a visual representation of coding elements. Concisely, a new coding language and integrated development environment should support the following features:

- GUI Environment, which will help and guide students during the creation of programs;
- Simplified coding syntax;
- Structured visualization of basic coding structures and examples;
- A small set of commands; and
- Documentation of the set of instructions and supported features by the coding language and development environment.

Overall, it took the author 14-16 weeks to complete this process and collect 194 participants data that helped to identify the aforementioned issues with the current mode of learning especially for beginners. Most respondents found traditional ways of learning tedious and confusing and desired to have a tool that would help the coding interactively 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.

4. 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 individual perception and knowledge. In the realm of this research, the importance of developing a framework goes well beyond a coherent way of thinking about theoretical learning concepts, Technological and pedagogical integration [27]. Many scholars have argued that knowledge about technology cannot be treated as context-free and that a robust methodology requires an in-depth view of how technology relates to the pedagogy and theoretical contents [28] [29] [30]. While discussing the framework components and the relationships between its contents, this research proposes a framework that is based on Technological, Theoretical and Pedagogical (refer to figure 9).

This framework focuses on integration of these components that would develop a robust and pragmatic conceptual framework to build a strong research base. The author argued that a proposed conceptual framework is, based on the relationship between technology, theory 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 need to be explored. How this framework has guided the proposed research and analysis of the effectiveness in this pedagogical approach.

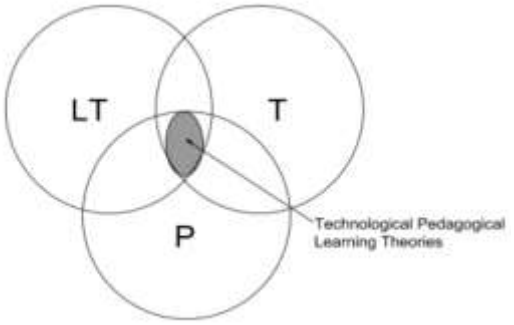

Figure 9 : Union of Theoretical, pedagogical and Technological approach

The aim of this research is to present a theoretical conceptual framework to develop an interactive pedagogical interface for beginners to learn coding using cognitive load and constructivism theories (refer to figure 10).

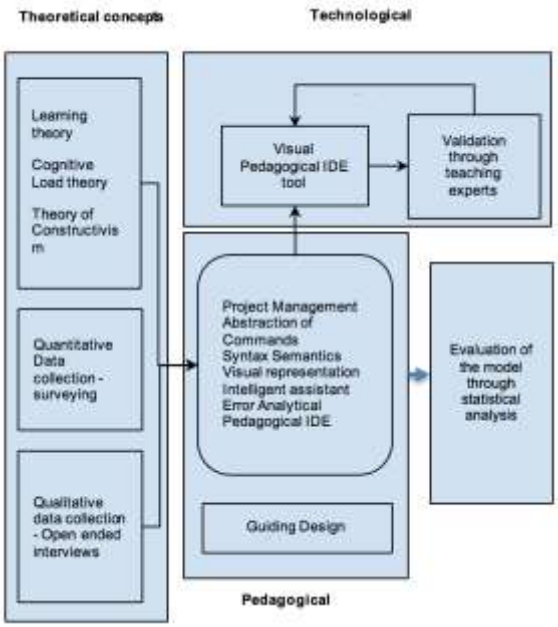

Figure 10 : Top level Process framework

Figure 11 briefly explains the meaning of each guiding design principles. This framework assists to identify, employing seven guiding design principles.

Figure 11 : Design Principles

5. EVALUATION OF THE VproEn

VProEn (Visualized Coding Environment) was designed as a coding interface fulfilling all the above-mentioned guiding principles (refer to figure 11).

Figure 12 presents this basic research design for statistical evaluation.

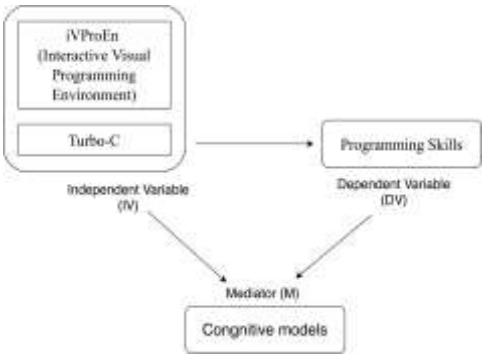

Figure 12: Research experiment design

i) the VProEn model evaluation, ii) the impact of such research on acquisition of a coding and level of skills quality on new students and, iii) The comparative study of proposed design model and partially complied tool (refer to table 1).

	Natural Language IDE	Syntax and Semantics	Visualization	Level of Abstraction	Set of Commands	Provision of Error messages	Interactive IDE
Turbo C	X	X	X	√	X	X	X
VProEn	√	√	√	√	√	√	√

Table 1: Key design for the two coding interfaces used in the Quasi-Experimental Designs

The proposed research conducted a quasi-experimental study to evaluate the efficacy of the model. In the experiment, the first-year students were divided into three groups S1 and S2 sizing approximately 20 students each. To conduct the experiment in class, all groups were asked to use Turbo C and the VProEn as their coding interface. The author distributed the teaching material, classroom examples, homework and teaching herself with the help of college. Moreover, the author was actively involved in the process and monitored all groups to gather unbiased data throughout the experiment session. The coursework to learn coding included First Theoretical notes about different coding concepts, second illustration of coding concepts in coding interface, Third Coding examples for learning to apply in coding concepts and Fourth, An exercise (refer to figure 12). In each coursework, students have the option to learn coding either using a nature language or the commands that were provided in the coding interface.
First few commands were provided to them to encourage and make them interested in learning by giving easy exercise.

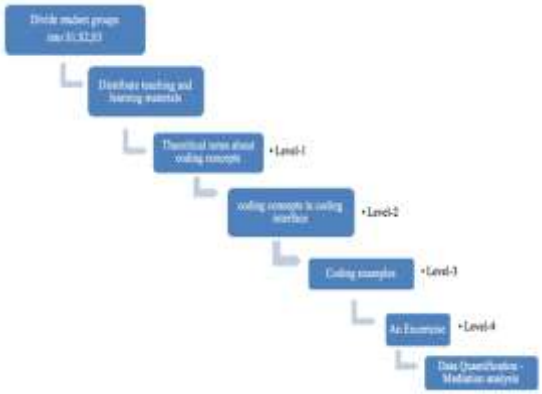

Figure 13 : Design of a Quasi-experimental group study

The author had to keep in mind that this was the first time that all his target population were learning to code. At the same time, they may not have access to software at their home. So, their productivity and focus were very crucial. At the end of each session, the author asked to return all the assignments and made sure that they finished their exercise in the classroom itself. There were two reasons for it, first students can finish their coursework while learning it and second do exercise while the concepts are fresh. After collecting the data from the designed experiment, data quantification was carried out.

The important objective for quantifying the data was to verify the mediation analysis. Therefore, the results gathered from quantifying the data were to identify: i) coding interface can have impact on participant's skill, ii) Changes in the significant percentage of variance. The author has decided to use following test that can present the results are: (a) analysis of variance (ANOVA), and (b) analysis of covariance (ANCOVA) and, (c) hierarchical multiple regression (HMR) when covariates should be included (Field, 2009). A regression analysis was conducted for all covariates as the internal parameter and the imperative-coding scores as the external parameter, to identify the important covariates. The only common significant predictor in the three quantified data was Participant's lack of experience. when the coding interface was included in the regression model of each quantification, all covariates that were significant before the inclusion ceased to be, with the exception of Participant's lack of experience (Refer to table 2).

Table 2: Regression Analysis of imperative-coding

internal parameter	external parameter	Statistical test	Covariates
Coding interface	Scores of the first imperative knowledge test	HMR	lack of Computing experience

The mediation quantification establishes whether the independent variable (IV) has significant impacts on the dependent variable (DV). The skill level of imperative cognizance was quantified using a number of parameters that was gathered from data collection from participants' ability to predict the outcome of codes, complete missing codes and change existing programs. Before testing the user's ability, all the concepts were taught to them and they were already aware of each of these commands. The difference in the group scores between VProEn and Turbo C was small to medium (refer to table 3).

Table 3: Descriptive Statistics of the sample data quantification

Group	n	Mean	SD	SE
VProEn	35	10.34	3.58	0.76
Turbo C	27	5.27	5.41	1.18

Note: Because not all students attended the test session, sample size was reduced in VProEn group

Moreover, data quantification of the test scores indicated that VProEn users appeared to perform better than users of Turbo C in the following coding aspects: i) Declaration of variables and use of input and output commands, ii) Use of correct semantics, iii) Identify the correct order and scope of the code, iv) Executability of the code and correctness of program execution. In nutshell, a significant improvement was observed in VProEn users than Turbo C users earlier but in the longer run no significant evidence was found. The outcomes of statistical assessment of collected data revealed that, at the end of the study, first timer coders who used VProEn tool fulfilling proposed design concepts seems to have augmented neither a higher level of coding skills nor richer logical models in introductory coding than first timer coders who used Turbo C fulfilling partial set of design concepts.

6. CONCLUSION

Many researchers suggested that learning programming for novices has always been demanding and frustrating. There were many reasons identified in many researches including sometimes lack understanding by the teacher, hesitant to ask any questions; the subject is very complicated to

understand and, at times, it's very boring to sit through entire class if you don't know how to read and write the syntax of the code. Therefore, to investigate such issues, this paper conducted an extensive review of two important learning theories that have a greater impact on learning while coding, i) theory of constructivism and, ii) Constructive Load Theory. After the thorough and critical review, this research identified and presented seven guiding principles for coding tool. 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) Error detector and (viii) An interactive IDE. From the review of the past literature, some vital gaps in knowledge were 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 coding tools. Second, many researchers discussed the set of design principles using aforementioned theories but not many researchers have measured the impact of such educational coding tools empirically. Then, to gather quantitative data through survey, targeting 360 population and collect qualitative data using open ended interviews from 12 lecturers. 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 coding elements. Based on the data quantification 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. The framework defines the relationships between theoretical, pedagogical and technological concepts. Data quantification of the test scores indicated that VProEn users appeared to perform better than users of Turbo C in the following coding aspects: i) Declaration of variables and use of input and output commands, ii) Use of correct semantics, iii) Identify the correct order and scope of the code, iv) Executability of the code and correctness of program execution. In nutshell, a significant improvement was observed in VProEn users than Turbo C users earlier but in the longer run no significant evidence was found. Furthermore, the future work will contribute towards the development of the coding tool to support different regional languages. Evaluation of the model was carried out using regression analysis. It will be done using real-life data to develop a robust and novel methodology in line with the proposed principles.

REFERENCES

1. Press Information Bureau, Ministry of Electronics & IT, Government of India (2017), http://pib.nic.in/newsite/PrintRelease.aspx?relid=162046
2. International Society for Technology in Education. (2000). National educational technology standards for students: Connecting curriculum and technology. Eugen
3. Zhao, Y. (Ed.). (2003). What teachers should know about technology: Perspectives and practices. Greenwich, CT: Information Age.
4. PRAZERES, Cássio V. S.; PEIXOTO, Maycon L. M (2013), "A Multimodal Interface for the Discovery and Invocation of Web Services". INFOCOMP Journal of Computer Science, Volume 12, Issue 2, p.p. 23-31. Available at: <http://www.dcc.ufla.br/infocomp/index.php/INFOCOMP/article/view/24>
5. Payal Gohel and Dr. Kishor Atkotiya (2017), "An Investigation of Constructivism and Cognitive Load Theory for Computer Programming Tool", International Journal of Innovative Research in Computer and Communication Engineering, Volume 5, Issue 9;
6. Ben-Ari, M. (1998). Constructivism in computer science education. ACM SIGCSE Bulletin 30(1), 257-261.
7. Ben-Ari, M. (2001). Constructivism in computer science education. Journal of Computers in Mathematics and Science Teaching, 20(1), 45-73.
8. Odekirk-Hash, E. & Zachary, J. L. (2001). Automated feedback on programs means students need less help from teachers. SIGCSE Bulletin, 33(1), 55-59.
9. Hadjerrouit, S. (2005). Constructivism as guiding philosophy for software engineering education. SIGCSE Bulletin, 37(4), 45-49.
10. Gonzalez, G. (2004). Constructivism in an introduction to coding course.Journal of Computing Sciences in Colleges, 19(4), 299-305
11. Beynon, M. (2009). Constructivist computer science education reconstructed.Innovation in Teaching And Learning in Information and Computer Sciences,8(2), 73-90.
12. Milner, W. W. (2010). Concept development in novice programmers learning Java. PhD thesis, The University of Birmingham, Birmingham.
13. Lee, Y.-J. (2011). Empowering teachers to create educational software: A constructivist approach utilizing etoys, pair coding and

cognitive apprenticeship. Computers & Education, 56(2), 527-538
14. Van Merriënboer, J. J. G. (1990a). Instructional strategies for teaching computer coding: Interactions with the cognitive style reflection-impulsivity. Journal of Research on Computing in Education, 23(1), 45-53.
15. 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.
16. Mayer, R. E. & Moreno, R. (2002). Aids to computer-based multimedia learning. Learning and Instruction, 12(1), 107-119.
17. Mayer, R. E. & Moreno, R. (2003). Nine ways to reduce cognitive load in multimedia learning. Educational Psychologist, 28(1), 43-52.
18. Caspersen, M. E. & Bennedsen, J. (2007). Instructional design of a coding course: A learning theoretic approach. 3rd international workshop on Computing education research, Atlanta, Georgia, USA.
19. Abdul-Rahman, S.-S. & du Boulay, B. (2014). Learning coding via worked examples: Relation of learning styles to cognitive load. Computers in Human Behavior, 30(0), 286-298.
20. Gray, S., Clair, C. S., James, R. & Mead, J. (2007). Suggestions for graduated exposure to coding concepts using fading worked examples. 3rd international workshop on Computing education research, Atlanta, Georgia, USA.
21. 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.
22. 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.
23. Teague, D. & Roe, P. (2008). Collaborative learning: Towards a solution for novice programmers. 10th conference on Australasian computing education Volume 78, Wollongong, NSW, Australia.
24. Hristova, M., Misra, A., Rutter, M. & Mercuri, R. (2003). Identifying and correcting Java coding errors for introductory computer science students.SIGCSE Bulletin, 35(1), 153-156.
25. 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.
26. Pane, J. F. & Myers, B. A. (1996). Usability issues in the design of novice coding systems (school of computer science technical report cmu-CS- 96-132). Pittsburgh, PA: Carnegie Mellon University.
27. 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
28. 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.
29. 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
30. 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.