ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

An Investigation of Constructivism and Cognitive Load Theory for Computer Programming Tool

Payal Gohel, Dr. Kishor Atkotiya,

PhD Researcher, Saurashtra University, Rajkot, India

Professor, Saurashtra University, Rajkot, India

ABSTRACT: In education, computer programming is considered to be a difficult and challenging subject to learn and understand especially for beginner students. Identification and investigation of such causes and how to resolve have been a focused area for the research community. Even with the advancement of technology, education sector is still not fully utilising the technology. Therefore, it's high time that appropriate teaching materials 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 cognitive approach engaged in educational teaching. Amongst many, two principal theories, specifically theory of Constructivism and theory of Cognitive Load have been identified and investigated. This theory provided a theoretical process framework to design and develop programming instruction guide. The proposed research aims to improve the delivery of teaching for in computer science students from a pedagogical point of view by designing and implementing an innovative visual programming model, verified by students and teaching experts.

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

I. INTRODUCTION

In the past, many researcher have pointed out that most students faced 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. Even with the advancement of technology, education sector is still not fully utilising the technology. Therefore, it's high time that appropriate teaching materials should be developed using interactive teaching curriculum to ease students' learning difficulty caused by lack of experience and necessary understanding. Use of Multimedia in an interactive teaching environment, in particular with the integration of cognitive psychology and 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 cognitive 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 researcher 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).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

- 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.

II. IDENTIFICATION OF LEARNING THEORIES

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, the 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.

III. 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 one, 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.

IV. 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]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

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 loadT 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 study the results that contradicted cognitive load theory's predictions and identified some critical methodical problems.

V. PROPOSED DESIGN PRINCIPLES

The aim of the research is to propose a theoretical conceptual framework to develop an interactive pedagogical multimedia tool for computer science education using Cognitive load and Constructivism theories. The discussion of these learning theories and the identification of critical challenges of computing students in their programming subject, provides the basis to develop a theoretical framework. This framework assist to identify, necessary aforementioned seven principles as vital for the design and development of a tool for young programmers.

1. Native environment: It's a first but important design aspect of the proposed tool. An interactive tool should be able to deliver instructions in multi language to novice users whose first language is not an english.
2. Common syntax semantics: Even the experience programmers face syntax confusion when they are working on different environments. So the proposed tool should present standard PL 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.
3. Visual presentation: Visualisation means visual programming that uses graphical means visual symbols to create programs and that symbols can be in form of flow diagram, icons, tables or forms.
4. Standard set of Abstract commands: Abstraction is a mechanism which focuses 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].
5. User guidelines: 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.
6. Analyse and present error messages: Compiler should have errors 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 novices.
7. An interactive IDE : A visual interaction could make IDEs more engaging to students. Instant response and feedback based graphical interfaces could be used for programming including the editor for declaring variables, defining functions, and the debugger to display errors or any messages[47].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

VI. CONCLUSION AND FUTURE WORK

Many researches 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 papers 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) Native environment, (ii) Common syntax semantics, (iiI) Visual presentation, (iv) Standard set of commands, (v) User guidelines, (vi) Analyse and present error messages and (vii) 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 researcher discussed the set of design principles using aforementioned theories but not many researchers have measured the impact of such educational programming tool empirically. Therefore, this research will propose a design and implementation of a novel methodology as an validation of these principles. Furthermore, this proposed tool will validate the developed model with students and assess its impact on beginners.

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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

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 workedexamples: 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.

BIOGRAPHY

Payal Gohel is a PhD Researcher in Computer Science Department, Saurashtra University. She received Master of Science in Information technology (MSc IT) in 2014 from Saurashtra University, Rajkot, Gujarat, India. Her research interests are Cognitive theory, Educational tools, Visualisation models, Algorithms, Pedagogical approach.