RedWhitepress https://series.gci.or.id

Global Conferences Series:
Sciences and Technology (GCSST), Volume 2, 2019
The 1st International Conference on Education, Sciences and Technology
DOI: https://doi.org/10.32698//tech1315134

Scratching Our Own Itch: Software to Teach Software Programming

M S Tuloli[1], M Latief[1], M Rohandi[1]

[1]Universitas Negeri Gorontalo, Kota Gorontalo, Indonesia

Abstract. Extensive research about the use of software to improve teaching and learning has been done. But this doesn't seem to significantly improve the learning process of the most basic skill in software that is programming skill. This is unfortunate because the nature of programming learning material is full of logic based material and can be automated (and then optimize). Programming is still perceived as a hard subject to learn. Of many approaches that have been taken, this research analyses an adoptable principle, practices, and tools that have been proven to improve programming learning. This research analyses the current state, obstacles, and potential further exploration of technology/software in teaching and learning programming skill.

1. Introduction (The Itch)

Programming course often becomes the highest drop out rate [1] and its hard point out why programming course can be a success or fail (Wiedenbeck et al in [2]). This is becoming enigmatic because the nature of programming teaching material usually are easier to computerized and from this can be optimized [3][4]. We can assume that the programming material can be easily imported into a software-based teaching material so it can be automated and teacher roles can be reduced or even replaced, but this doesn't seem to happen.

Extensive research about the use of software for improvement of teaching and learning is already been done for many subjects. It has been done for language learning [5], art and culture [6], computer graphics [7], software testing [8] or for course supporting [9].

The problem with current approach: long feedback loop of the curriculum [10], the increase of understanding doesn't reflect in the grades [11], most researchers are using new approach just to support traditional curriculum (Sajaniemi & Kittinen in [12]), or may be related to student mental development [13]. Visualization and animation that most used to ease the learning process can sometimes become an unnecessary cognitive load or even generate confusion [14], and prove to be less important than the learning exercises [15]. The difficulties may be caused by programming activities itself that consist of the use of several skill altogether [16], to be taught to student with variety background (some with previous miss understanding) [17] with some them lack of background knowledge (i.e. English and Math) [18], and the teacher itself may not suite for teaching programming [19].

This paper doesn't try to give a complete review about teaching programming, but just get a glimpse of current state of research about the domain. Chapter two will explain approaches in teaching programming, chapter three will propose a principle to be adapt to make a successful teaching programming. Last two chapter explain research and improvement potencies and conclusion.

2. Method

Here we conduct a simple literature survey, by using a manual search on google scholar search engine, the keywords used are "teaching programming". Because of the large number of papers that have appeared, we are starting a search for the latest paper, which is 2019. For papers that do not provide download links, a search is done at ieeexplore.ieee.org. From the collected papers a further study was conducted to produce additional references to a theme which were considered important, for example regarding explaining certain approaches or criticizing the approach. From the collected papers, we analyze and categorize the papers in terms of the approach used, the underlying principles that successfully used, and also the proposed further development.

3. Alternative Method to teach programming (The Scratching)

There are actually many approaches have been explored to improve teaching programming, here we categorize of the most popular.

3.1. Problem-Based Learning

Problem-based learning is by teaching through solving a problem, this approach is able to make students motivated to understand, facilitate multiple levels of difficulty, develop cooperation, and can be easily adjusted to the objectives of the course (B. Dutch in [20]). PBL is a good approach because the most difficult issue in teaching programming is problem-solving strategies [2]. An example of this approach is RoboCode and Online Judge (e.g. Codingbat.com, SPOJ, UVa).

3.2. Gamification and games

Gamification is the adoption of games element like badges, skill tree, leaderboard, level etc to a non-games problem. Gamification has been adapted into many aspects mostly to improve morales and with hope to improve productivity, including in teaching programming [21]. Another approach is to actually makes student play games, and thus makes them understand (or try to) a programming concept by playing it [22].

3.3. Non-programming Activities (Unplugged computing)

Unplugged-computing is a method of teaching programming concepts without using a computer at all. This method has been used to teach concepts such as variables, expressions, looping [23]. Although some (i.e. fruit) do not provide an increase in learning outcomes [24], it still a promising approach to teaching programming.

3.4. Plugged approach

This approach is to invite students to do programming in embedded systems (eg robots), this can make students motivated because they can feel real-world applications from the programming work done. Popular tools and devices used are Lego Mindstorm, Raspberry Pi, Arduino, etc. Another similar approach is the use of virtual robot, the advantage this approach is that students can focus more on programming and not on the physical creation of robots, eliminating errors due to damage of robot parts (sensors, batteries, etc.), no longer requiring to deal with the maintenance and storage of robots [25], and student can practice outside of laboratory/course hour [26]. And because the development environment is the same as the physical robot programming environment, students are also ready to use actual robot programming.

3.5. Reduce programming languages difficulty

This can be done by trying to use less complicated programming language [27] or using pseudo-code [28]. It aims to make it easier for students to focusing more on improving the ability of problem-solving without being distracted by the syntactic complexities of programming languages.

3.6. Visual Block-Based programming

To eliminate the syntactic difficulties of programming some approach uses block-based programming, this has the potential to direct teaching more to problem-solving ability. Block-based programming is an approach to create a program only by using already defined block. This approach is very easy to use but some result shows that it doesn't improve problem-solving skill [29]. This is because to understand programming visually takes some level of expertise (Petre in [14]).

Example of block-based programming is Scratch, Appinventor, and Blocky [30]. Some tools already combined with 3D visualization [31]. This approach is not without a caveat, it's proven that it may instill a bad programming habit, for example, it may encourage a bottom-up approach in solving problem, and create fine-grained programming [32]. As many of these approaches are for teaching introductory programming for children, it seems apart from the programming languages teaches in higher education [18].

Some of the approaches are a combination of these approaches (i.e. Problem based learning and gamification, and already given a promising result [17].

4. **Principles**

The main principle that has been mentioned in the previous chapter are strengthening algorithm weakening languages [33] and more focus on programming strategies than languages related knowledge [1]. While a program/software feature that should be considered when designing an approach is detailed in [34], we argue that some more principle should be considered to increase method effectiveness:

4.1. Teach less practice more

The road to learning programming skill often takes quite a time (Wei in [35]), it takes more than class learning times, it needs student to do self-learn. So we need a strategy to accompany student through boredom, some ideas like incremental-practices and reduces teach has proven to success [36].

4.2. Instantaneous feedback

Many of the positive results, points out that instantaneous feedback is a successful strategy. Because it proves to improves student engagement [37] and easy to integrate optimization [4].

4.3. Deliberate Practices

Increase focus on difficult concept [38], Identifying threshold-concept in programming [39], because in these concept student can have misconception [40] that caused by previous self-taught experience [17].

4.4. Integrating mobile learning

Because of the popularity of the use of mobile applications by students, teaching should be able to take advantage of this, besides of being able to provide a learning experience "anytime, anywhere" mobile learning has been proven to improve: student engagement, authentic learning activities, and informal learning [41].

4.5. Teacher Involvement

The last but not least principle is the awareness and concern of the teacher to improve his/her abilities, experience in programming, and openness to integrated a successful practice into its teaching method. This is because has been shown that teacher sometimes is untrained, inexperienced, or just lack confidence [19].

5. **Research Potencies**

Actually, every approach that has been described each can still be developed to provide better results or to provide new insights into the use of methods/tools in new environments/ways. But we think most of a good and up-to-date research direction has been defined by Medeiros et al [18]:

5.1. Improvement in Research Construct

As has been emphasized in the previous chapter about the importance of focus to be more on problem-solving abilities than on mastery of programming syntax. There is still no consistent definition of problem-solving capabilities, without a clear definition, making the right strategy and steps will be difficult to do. A clear definition of problem-solving capabilities is obviously needed to be explored.
There is a high variation of methods, objectives, targets, etc of the research carried out in this domain. This makes it difficult to provide a more accurate comparison of the effectiveness of an approach to other approaches. It is also necessary to emphasize the use of empirical data (not just observations) to provide strong support for the results given. Exploration in this aspect may also help the selection of method/tools to reach learning objective [42].

5.2. Improving Background Knowledge,

The actual problem faced by the instructors of programming is the lack of student background knowledge. Capabilities such as mathematics are often needed to design algorithms. Another important ability is mastery of English, this will limits learning process, because many of the materials and tools for teaching that need English capabilities. So the adoption of these materials/tools can even reduce the effectiveness of teaching and learning. Efforts that have been made to reduce this gap (eg teaching introduction to programming in children) need to be done more, it is also necessary to examine methods and metrics to measure the effectiveness of the approach used.

5.3. Development of Tools and Method,

The challenge faced by students in learning programming (i.e. working on programming problems) can be divided into two type, formalizing problems and expressing solutions to problem. This is the reason for developing specific tools that only focus on one aspect (e.g. block-based programming). Research and development are still needed to develop methods and tools that focus on one of these aspects. Tools also need to adapt to new technology and trends (e.g. using augmented reality [43]),

5.4. Motivation,

Actually, almost all approaches that have been developed will be influenced by motivational factors of students to carry out independent exploration both in the classroom or even better outside the classroom. Broader exploration needs to be done in attempt to generate student internal motivation in using approaches/teaching aids in programming.

5.5. Wider use of methods (Disabilities, children, and woman)

The lack of alternative non-visual learning aids can discourage people with disabilities from learning to program, more initiatives such as those on code.org (*http://code.org*) are needed to facilitate this. The use of robots to teach kindergarten and pre-school is still less explored by researchers [44]. Exploration also needs to be done to motivate female students in learning programming [45].

6. Threat to Validity

Because the purpose of the research is to get a general picture of this field, there are still many ideas that might be missed (e.g. because of the exclusion of non-English papers), and vice versa, because varying quality of the papers we used (i.e. journals, conference, and article) may affect the validity of our result. Likewise, here we still have yet conducted a deeper analysis of the differences in objectives, student targets, and measurement methods of each paper, because it will take more spaces to describe it (e.g. papers about teaching children are very different to teach higher education).

7. Conclusion

Teaching programming has many problems, and there have been many approaches that have been tried at various levels (pre-school to college level). The approaches and tools used have their advantages and disadvantages so they need to be adapted to support learning objectives. The development of a new approach needs to accommodate principles that have proven successful. Further exploration needs to be done to further enhance effectiveness or design new approaches to teaching and learning programming.

8. References

[1] Robins A, Rountree J and Rountree N 2003 Learning and Teaching Programming: A Review and Discussion *Comput. Sci. Educ.* **13** 137–72

[2] Malik S I and Coldwell-Neilson J 2017 A model for teaching an introductory programming course using ADRI *Educ. Inf. Technol.* **22** 1089–120

[3] Lin C, Liu Z, Chang C and Lin Y 2018 A Genetic Algorithm-Based Personalized Remedial Learning System for Learning Object-Oriented Concepts of Java 1–9

[4] Howell R and Wong S H S 2018 Making the Most of Repetitive Mistakes : An Investigation into Heuristics for Selecting and Applying Feedback to Programming Coursework *2018 IEEE Int. Conf. Teaching, Assessment, Learn. Eng.* 286–93

[5] Elaish M M, Ghani N A, Shuib L and Al-haiqi A 2019 Development of a Mobile Game Application to Boost Students ' Motivation in Learning English Vocabulary *IEEE Access* **7** 13326–37

[6] Michala M and Alexakos C Mobile Applications and Games for a Digital Educational Program on Art and Culture in Secondary School *2018 9th Int. Conf. Information, Intell. Syst. Appl.* 1–6

[7] Suselo T and Luxton-reilly A 2019 Technologies and Tools to Support Teaching and Learning Computer Graphics A Literature Review 96–105

[8] Fraser G, Gambi A, Kreis M and Rojas J M 2019 Gamifying a Software Testing Course with Code Defenders

[9] Vial P J, Russel T J, Stirling D, Ros M, Prashan P, Tran L C and Nikolic S 2018 A Java Program for Automatic Team Allocation in Project-Based Course Work **6** 185–92

[10] Paillard B 2019 Teaching Programming is Hard How to disrupt traditional universities and apply rapid iteration to build the best tech curriculum . **2016** 1–7

[11] Sajaniemi J and Kuittinen M 2007 An Experiment on Using Roles of Variables in Teaching Introductory Programming *Comput. Sci. Educ.* **15** 59–82

[12] de Raadt M 2008 *Teaching programming strategies explicitly to novice programmers*

[13] Ko A J, Latoza T D, Hull S, Ko E A, Kwok W and Quichocho J Teaching Explicit Programming Strategies to Adolescents

[14] Moreno A, Sutinen E and Joy M 2014 Defining and evaluating conflictive animations for programming education 629–34

[15] D H, Douglas S A and Stasko J T 2002 A Meta-Study of Algorithm Visualization Effectiveness

[16] Juárez-ramírez R, Christian X and Macías-olvera R 2018 What is Programming ? Putting all together – A set of skills required 11–20

[17] Gordon N, Brayshaw M and Grey S 2019 A Flexible Approach to Introductory Programming: Engaging and motivating students *Proc. 3rd Conf. Comput. Educ. Pract.* 15

[18] Medeiros R P, Ramalho G L and Falcão T P 2018 A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education 1–14

[19] Ohashi Y 2018 Readiness of Japanese Elementary School Teachers to Begin Computer-Programming Education *2018 IEEE Int. Conf. Teaching, Assessment, Learn. Eng.* 807–10

[20] Kelly J O and Gibson J P 2006 A non-prescriptive approach to teaching programming *ITiCSE* 217–21

[21] Morrison B B and DiSalvo B 2014 Khan academy gamifies computer science 39–44

[22] Miljanovic M A and Bradbury J S 2013 A review of serious games for Programming *Diabetes Technol. Ther.* **15** A109
[23] Grover S, Jackiw N, Lundh P and Grover S 2019 Concepts before coding : non-programming interactives to advance learning of introductory programming concepts in middle school concepts in middle school *Comput. Sci. Educ.* **00** 1–30
[24] Pérez-Marín D, Raquel H-N, Romero A and Cruz S 2019 Is the use of Makey Makey Helpful to Teach Programming Concepts to Primary Education Students? *Int. J. Online Pedagog. Course Des.* **9** 15
[25] Majherová J and Králík V 2017 Innovative Methods in Teaching Programming for Future Informatics Teachers *Eur. J. Contemp. Educ.* **6** 390–400
[26] Haak V, Abke J and Borgeest K 2018 Conception of a Lego Mindstorms EV3 simulation for teaching C in computer science courses *IEEE Glob. Eng. Educ. Conf. EDUCON* **2018**–**April** 478–83
[27] Koulouri T, Lauria S and Macredie R D 2015 Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches *ACM Trans. Comput. Educ.* **14** 26:1–26:28
[28] Malik S I, Mathew R and Hammood M M 2019 *PROBSOL: A Web-Based Application to Develop Problem Solving Skills in in Introductory Programming* (Springer International Publishing)
[29] Kalelioğlu F and Gülbahar Y 2014 The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective *Informatics Educ.* **13** 33–50
[30] Papadakis S, Kalogiannakis M, Orfanakis V and Zaranis N 2017 The Appropriateness of Scratch and App Inventor as Educational Environments for Teaching Introductory Programming in Primary and Secondary Education *Int. J. Web-Based Learn. Teach. Technol.* **12** 58–77
[31] Pausch R and Dann W 2000 Alice : a 3-D Tool for Introductory Programming Concepts *J. Comput. Sci. Coll.* **15** 107–16
[32] Meerbaum-salant O, Israel R and Ben-ari M 2011 Habits of Programming in Scratch 168–72
[33] Yu L, Zhang L, Su X and Liu X 2019 Research on Case Teaching Mode of Programming Course Based on Interdisciplinarity **286** 385–8
[34] Talib M A, Einea O, Nasir Q, Mowakeh M F and Eltawil M 2019 Enhancing computing studies in high schools: A systematic literature review & UAE case study *Heliyon* **5** e01235
[35] Khaleel F L, Ashaari N S, Wook T S M T and Ismail A 2017 Programming learning requirements based on multi perspectives *Int. J. Electr. Comput. Eng.* **7** 1299–307
[36] Vihavainen A, Paksula M and Luukkainen M 2011 Extreme apprenticeship method in teaching programming for beginners 93
[37] Rattadilok P and Roadknight C 2018 Improving Student's Engagement Through the Use of Learning Modules, Instantaneous Feedback and Automated Marking *2018 IEEE Int. Conf. Teaching, Assessment, Learn. Eng.* 802–6
[38] Hamzah N H, Azha N and Shaari M 2019 Undergraduate Computer Science Students ' Perception and Motivation : A Feasibility Study and a Proposed Technique for Multimedia Approach in Teaching and Learning Introductory Programming (Springer Singapore)
[39] Yeomans L, Zschaler S and Coate K 2018 Transformative and Troublesome ? Students ' and professional programmers ' perspectives on di difficult concepts in programming **1** 1–27
[40] Diego S, Kaczmarczyk L C, East J P, Petrick E R and Herman G L 2010 Identifying Student Misconceptions of Programming 2–6
[41] Yong S, Noum E, Sivanesan S K, Pei M, Tay L, Namasivayam S N, Fouladi M H and Loong T H 2018 Integrating Mobile Learning into the Foundation in Engineering Programme *2018 IEEE 10th Int. Conf. Eng. Educ.* 196–201
[42] Varvara G, Giannakos M N and Chorianopoulos K 2015 Computing education in K-12 schools: A review of the literature *IEEE Global Engineering Conference (EDUCON)* pp 543–51
[43] Jedc L H, Jc I D, Bdg I D, Egd E I H, Dk D C, Dcid H, Ild I D, Ig D G, Edh G Y H, Hj J, Id I,

Dc H D, Egdk A A, Egdk G A N, Hij D G, Bda I H, Dbeji D G, Dbeji D C, Hij G H I, Jc G N I D, Jh C I, Egd D G, Hij A A N, Bjhi C I H, Hdak B H I D, Id B H, Ldg I D, Ldg A N and He C 2018 The Potential of Augmented Reality for Computer Science Education 350–6

[44] Khine M S 2017 Robotics in STEM Education: Redesigning the Learning Experience *Robot. STEM Educ. Redesigning Learn. Exp.* 1–262

[45] Chetty J and Barlow-Jones G 2018 CODING FOR GIRLS : DISMISSING THE BOYS CLUB MYTH Jacqui Chetty and Glenda Barlow- - Jones University of Johannesburg South Africa *ICICTE* pp 324–35