Evaluating the Effectiveness of Problem Solving Techniques and Tools in Programming

Seyyed Meisam Taheri, Minoru Sasaki
Department of Human and Information Systems Engineering
Gifu University
Yanagido, Gifu-shi, 501-1193, Japan
s3812005@edu.gifu-u.ac.jp

Harrison Thuku Ngetha
Department of Mechanical Engineering
Gifu University
Yanagido, Gifu-shi, 501-1193, Japan
s3812007@edu.gifu-u.ac.jp

***Abstract*—Programming is one of the essential and most difficult skills to learn in the computer field and other disciplines. Programming can seem more troublesome for novices who have not learned programming concepts, usage and other basic programming skills. To become a programmer, there are many things need to be considered in programming such as syntax, pragmatics and semantics. In addition, having a good knowledge about problem-solving techniques is vital. For instance, when some students face a new problem in programming courses and they feel understanding the problem is not easy, difficult tasks which students need to handle as well as, the manner of teaching programming may discourage them to do coding. Mostly, in the first phase of learning programming, they find it difficult, and they will not be inspired to do programming; this is one of the reasons that most students cannot do coding by themselves. An environment in which is a source and guidance to look for commands and explanations would be helpful, and interaction with teachers or friends can assist in overcoming most of the difficulties. This research aims to study about the effectiveness of learning the fundamental of problem solving in programming, as well as, learning these abilities in the early ages. This paper introduces and compares several existing problem-solving tools and shows how visual problem-solving techniques will help students to improve their programming skills. Furthermore, in this paper we analyze the experience of programming, problem solving tools and techniques and the role of social networks in programming based on gathered data which was collected through an online survey.**

Keywords*—Programming; Problem Solving; Learning; Programming Languages; Social network

I. Introduction

Programming defines as a method of writing a source code of computer languages using different languages and integrated development environments (IDE), as well as, testing, debugging and maintaining the code [1]. Programming languages have been developing and growing fast, and the new features have been added to IDEs and compilers, but what has never changed in programming is its nature, fundamental and rules. Various researches has been done in this area which show that those who learn programming from young ages by following problem solving techniques can learn new languages easier. Programming is difficult but learnable skill, and it requires a great depth of knowledge to understand the different aspects of programming (syntax, semantics, pragmatics, problem solving, and etc.) [1]. Programming in a MS-DOS environment was difficult without a user-friendly graphical interface, and programming problems became more complex. The biggest problem of novices is not their understanding of the basic concepts of programming but rather learning how to use the basics in a program. Students' performance in programming is a matter of concern and the basic programming courses remained as a major obstacle for many students with resultant high failure rates. Complexity of modern programming languages and user interfaces of the IDEs is considerable [2]. Hilburn notes that method of coding with simple statements that student introduced to, does not allow them to develop their problem-solving skills, which are vital for a better program development. He also describes this as a "bottom-up" approach; students are expected to identify the logical processes of developing a program ("top-down") which makes programming language a tool to implement the solution to the problem [3]. By developing the methods of teaching and teaching material, as well as, considering teaching the basics of programming and problem solving to young child programmers, it would be possible to encourage future programmer to become more creative and interested in programming. Interaction with classmates, friends and teachers could also help programming learners to meet their needs, and it would be an effective way to learn as well. These types of interactions would help the better understanding of concepts and rectifying errors in the code. In the programming world, programmers usually face compilers' error messages, which are difficult to understand and debugging the errors is not an easy task as well [1]. Some researchers are already developing programs and robots to interact with children and young learners as a tool or game to make them able to learn problem solving and programming techniques, and the result of the researches has shown to be effective. In this study, we analyze a survey to illustrate that how important is to invest on children and novice programmers to learn problem solving techniques and the basic of programming. The effectiveness of techniques and methods will be discussed in this study.

II. Aim

This study aims to review and analyze techniques, tools and online teaching methods to evaluate that these methods and tools are useful for novices to overcome the difficulties in programming, and simplify the learning of programming. Reviewing and analyzing the previous and current studies in this area is the significant objective of this study. It is very important to make in-depth review of each study related to the scope of this study [1]. In this study, we are also analyzing the gathered data through an online survey. The main intention of the qualitative methods used in this study is to gather insights about the nature of problem solving in programming, the difficulties in learning programming and to develop theoretical perspectives about learning programming. The novice programmers' and experts' needs, the use of problem-solving tools and knowledge about the techniques of programming languages will be discussed and evaluated in this study.

III. Related Works

Research has been carried out in the area of problem solving and techniques, and this study reviews several related works, which are not limited to the following.

A. *The Iconic Programming Tool (B#)*

The Iconic Programming Tool (B#) is a tool designed for novice programmers. The main purpose of designing B# is to focus on difficulties in basic programming courses, including problem-solving techniques and methods, understanding programming languages concepts, and the traditional programming environment. Since the visual environment has been developed, it assists developers in combining teaching methods with iconic and flowchart methods. In fact, iconic programming has aimed to decrease the manual typing and level of accuracy required to make programming easier. Thus, B# is a developed environment that allows programmers to code a program by using iconic flowcharts. In this tool, basic programming features such as assigns, conditions and loops are supported. Automatic generating codes, debugging and program executing are supported by the system [4]. B# objective is focusing on iconic programming to make programming easy to program and understandable. Interacting with an iconic environment can be easy for any user regardless of their age and programming experience.

B. *E-Learning (Visualisation and Dynamic Tool)*

Progranimate is a web-based environment that aims to simplify the issues faced by programmers. This tool concentrates on flowcharts to find solutions for basic programming problems; this program generated the flowcharts and codes based on given information in a visual environment. Due to complex programming IDEs, syntax and non-traceability, because of lack of attention to basic of programming novice programmers find programming very difficult and some researchers have clearly proven that [5]. Flowcharts are not only used in programming, they are used to show the steps of any task which has to be done, because flowcharts are easy to understand and implement. Westphal states, *"Without the use of diagrams or flow charts, it is difficult for beginners, even with pseudo code to communicate the flow of a program*" [6]. According to Ben-Bassat, dynamic animation can develop the flowchart's efficiency to assist novices' in doing algorithmic problem solving and programming coding [7].

C. *Jeliot #*

The Jeliot tool aims to help programming learners to learn object-oriented and procedural programming more easily. Features of Jeliot include semi-automatic visualization and control flows. The main idea of this tool is to allow programmers to see the flow of executing a code and encourages them to see the running code step by step. It helps them better understand the process of a program and lets them develop a mental model about calculation.

When programmers engaged with programming tool to find out more about the abilities of the Jeliot tool, they can learn more about the functionality and concept of programming. BlueJ is one of the first tools which have been developed to teach the introduction of object-oriented programming [8]. The features of this particular tool is the static visualization component which has the same structure as UML diagram to implement and design classes. Javavis is also a tool developed with the same purpose that uses Java Debugging Interface (JDI) to get information about the runtime performance of the program [9]. These tools can be used as a complementary tool together with other tools and teachers to assist novices; however, these could be helpful for experts in programming as well.

D. *The Codewitz project*

Object-oriented approaches, pointers and memory allocations are the most difficult parts in programming. Codewitz is a tool that provides an environment for programmers to visualize their code and the steps and processes of implemented codes can be seen through this tool. Both students and teachers can use the Codewitz tool; it helps them to gain a better understanding of the programming processes and variable memory allocations. The environment has two sections code and visual, which allow users to control and understand their actions and code more easily. This tool is same as Jeliot 3 in functionality and concept, but the variables allocation in the memory is in visual mode, which is slightly different from other tools. This tool is very useful as it can recognize the differences between commands, modules and operations. It also demonstrates operations systematic, memory allocation and how the variables and addresses in the memory which are changing. By using this type of tool and techniques, programming will be easier for novices and other programmers to understand [1].

E. *Raptor: Flowchart based programming Environment*

Despite research showing that approximately 75% and 83% of students are visual learners and thus most students learn faster through visual teaching methods, teachers tend to teach verbally[10][11]. RAPTOR is a flowchart-based programming environment designed to assist students who face syntactic errors by visualizing the algorithms for them. Programmers can trace the code while executing programs visually by following flowcharts. In Raptor, users can design an algorithm by using flowchart icons and combine them together and after running the algorithm, they can see the process step by step. Algorithms and flowcharts are two basic needs in programming, and using this basic problem solving techniques make the tool more effective. Users with different styles of learning can use this tool; some might prefer to use flowcharts, while some others may prefer to use algorithm as a method of implementing or understanding the programming codes.

F. Teaching using Massive Open Online Course (MOOC)

Distance learning is a learning option for those who do not have access to teachers physically during their learning period, and it is by no means a new phenomenon [12]. Information and knowledge have been transferred through e-mail, social media or even TV programs, and nowadays mostly through Internet. Distance learning provides an environment for both teachers and students to interact for collaboration among students themselves. Distance learning has been considered as an extra or alternative method of education, yet it has not become a major segment of education and the physical attendance of teachers is required. However, this may be quickly changing now, with MOOCs offering openly online courses free to students anywhere in the world [13]. The first MOOC was offered by the University of Manitoba in 2008 [14], however MOOCs have grown too fast for higher education needs due to the growing world population and availability of technology, and internet access has made these courses more accessible and possible. When an Artificial Intelligent MOOC course was offered by Stanford University in 2011, 160,000 students from all over the world registered and 23,000 of them successfully completed the course. The foundation of MOOCs were based on computer science and engineering courses, but researchers and instructors have considered whether MOOCs can be used in teaching skills such as critical thinking, problem solving, communication, and entrepreneurship[15].

IV. Review and Analysis Collected Data

To be a good programmer means to be creative and have invention, and specific knowledge within programming is something that can be learned based on each programmer’s needs. Problem solving techniques use in an array of fields; hence, it is useful to have some basic familiarity with problem solving regardless of one field of work or study. Designing complex systems is much like designing code; understanding how smaller parts of code create a bigger functional unit could help in understanding how one could build real-world complex systems from simpler parts (e.g. engines, buildings, power plants). Problem solving skills are important, it is simply because, understanding what questions to ask is as important as being able to search for the answers.

Many people think knowing a computer language and doing programming is enough. Knowing how the pieces move on a chessboard does not mean you know how to play chess; memorizing many words in a foreign language does not qualify you to write a poem or novel in that language. It is a need to teach actual programming in schools, i.e. understanding how to translate the problems and solution into a working set of instructions. Teachers should encourage students to try, and find another point of view about topics or problems; otherwise, they becoming parrot-like learners who mimic the same things as their teachers. Watching videos about programming can be beneficial and group work may be useful provided everyone in the group understands the problem is, they also can cooperate and motivate each other to accomplish an assignment. When you program a code, you are converting your text and logic into an automated program. In doing so, you are solving a problem that is essentially "what should I do to make this something efficient, productive, and useful". Problem solving is used every step of the way to reach the desired result. If problem solving had never been used in programming, no program could ever been created. Everything from a single click to vast amount and range of programs, all is possible because of programming. The science and technology world is expanding and it is expanding an extremely rapid pace.

Teaching programming and failure rates in schools are a concern today, it would be better if teaching problem solving and programming become a point of consideration in our society. Children should learn the basics first before expanding into programming. For instance, in bit manipulation, a child could have a firm grasp of Boolean algebra before working on embedded technologies. Children also should practice reading and writing and working in sound logical techniques before having permission to begin programming on a computer. The age of chaotic incoherent code exists because many companies have used developers who have not been educated or classically trained in classical education or basic IT studies. Programming has a lot in common with natural sciences and math, hence programming is a practical and "visual" way of learning problem solving. Not everyone will become a programmer but even knowing the basics of programming will promote algorithmic thinking and creative problem solving. The aim of programming should not be too much on the language but problem solving.

Respondents in this study also believe that teaching and observing children's problem solving skills at a young age is a better measure of potential talent than actual programming classes at a young age. It is not about programming specifics, yet more about high-level understanding of problem solving, logic and reason. Teaching programming languages to children is a difficult task but important; however, having programming as a subject or optional choice at an earlier stage in the school system would be beneficial. There are some researches has been done with some researchers in this area and they have develop a robot which is able to be programmed by children, for instance Dash & Dot.

TABLE I. Survey Questions

	Question	[a] 1	2	3	4	5
Q1	level of programming	1	5	38	59	56
Q2	The use of programming	1	0	5	18	135
Q3	Problem solving familiarity	5	16	29	44	65
Q4	Using Problem solving techniques	7	17	32	35	68
Q5	Effectiveness of problem solving and basic of programming	3	2	18	39	97
Q6	Effectiveness of learning from schools	21	29	48	28	33
Q7	Knowledge of programming	3	11	30	53	62
Q8	Study in a group	2	24	32	50	51
Q9	Teaching programming to young ages	8	10	38	41	62

[a] (1 to 5 scales are from negative to positive answers)

Table1 shows that the programming level of most respondents is high and most of them have programming background, however approximately 4% do not have computer programming knowledge or have very limited knowledge. Thirty-eight respondents know about programming or they are novices in programming, approximately 37% of them know about programming, and about 35% of them are using programming in their profession or are experts in programming. Furthermore, in this study we have collected information from the variety of users with broad backgrounds and knowledge through an online survey. The ages of respondents range between 12 to 76 years old, and 10 respondents are female with the remaining of them being male.

Respondents have studied or done programming for at least 2 months, and the longest period time of doing programming of all participants in the survey is 44 years. Most respondents are studying in one of the computer science fields; however, there are also respondents who have studied in other fields such as biology, physics. As the results show, computer programming is not limited to computer field specifically, as and as previously discussed, programming is necessary in different disciplines these days. One respondent does not find programming useful in their field of study, and five think it is not really necessary to have programming knowledge in their field. Eighteen respondents found that programming is related to their field and useful, but they can carry out and support their study even without programming. 135 of respondents believe that programming is essential in their field of study and having knowledge about how to do it will help them in their study and work.

Approximately 96% of respondents know about problem solving techniques, and this large number could be very helpful in this study. Some of them believe that problem solving is necessary in all aspects of life and it is not a concept specifically to programming. Other than 4.4 % of participants who did not use problem-solving techniques in programming or do not have a lot of knowledge about it, the remainder of respondents use problem solving techniques while they are doing programming.

The data also shows that some of the respondents who do not know about problem solving techniques also do not believe that problem solving can be useful in gaining a better understanding of programming. However, most believe that by teaching novices problem solving techniques and improving their problem solving abilities, it might become easier for them to understand coding or the concept of programming.

There were respondents who believe that programming courses in school and universities are useful and those can be relied on to learn programming, while others believe that education materials are not sufficient, or that the teachers are not covering all programming areas and some parts are being omitted. In addition to this, some of them believe that the teaching methods are not good enough to encourage students to take the course that more serious, and students will tire of programming after a short time. It can be one reason of high failure rates in programming as we mentioned in this study before.

Although, most respondents also believe that by doing programming as a team, they can share ideas to solve problem much easier, and that teamwork in programming can be effective. Only two respondents in this study believe that study in a group is not useful.

In regards to problem solving, most participants believe problem solving is necessary in all disciplines, particularly in mathematics and computer fields. They believe that teaching problem solving techniques from school age can prepare students for the next step, which in this case is learning the concept of programming. However, it does not mean everyone needs to be problem solver or programmer; however, it is a good way to discover students' abilities and creativity, students also can understand more about programming and decide whether to take or not take this course in future. As most of respondents have learnt problem solving in programming and used it in their professions, they think it may be beneficial skill for novices of young age.

As the previous table showed, most of respondents believe that problem solving is necessary and should be a mandatory course in schools, and on the other hand, some of them think that teaching programming to young children is difficult task for them and it makes the children grow tired of programming. Some respondents suggest that programming would be difficult for children, however if it is introduced as a game it would be both fun to play and educational. Teaching programming and problem solving to children can open their minds so they can see their future and world differently.

As previously discussed, 23% of respondents think that teaching programming to children will help them, and about 11% of them think maybe it is not necessary or not effective method to teach children programming and problem solving.

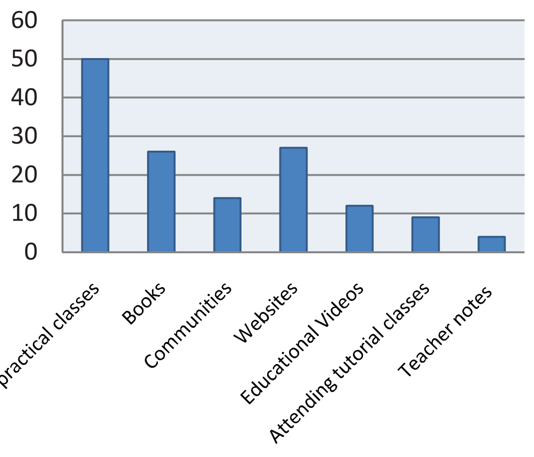

Fig. 1. Learning Materials

As the figure1 shows, most respondents believe that by joining practical classes the rate of learning would be higher, and self-study through reading books and the use of websites can be effective ways to learn more about programming. As the result shows, programming is not easy to learn and understand for students if they just join tutorial classes, understanding programming by doing coding or visual courses would be a better choice for them to learn programming.

ACKNOWLEDGMENT

The author would like to express his deep gratitude to professors Sasaki for his help, support, and encouragement. Special thanks extended to the staffs of the Gifu University of Japan.

REFERENCES

[1] Seyyed Meisam Taheri, Yamamoto Hidehiko, Hrudaya Kumar Tripathy, Novel Assessment of Different Intelligent Tools for Problem Solving, *Computer Science and Engineering*, Vol. 3 No. 3, 2013, pp. 67-75.

[2] Govender, I. & Grayson, D. (2006). Learning to program and learning to teach programming: A closer look. In E. Pearson & P. Bohman (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2006* (pp. 1687-1693). Chesapeake, VA: AACE.

[3] Hilburn, T.B. (1993). A top-down approach to teaching an introductory computer science course. 24th SIGSCE Technical Symposium of Computer Science Education. Indianapolis, USA, 1993.

[4] Greyling, J.H.; Cilliers, C.B.; Calitz, A.P.; , "B#: The Development and Assessment of an Iconic Programming Tool for Novice Programmers," Information Technology Based Higher Education and Training, 2006. ITHET '06. 7th International Conference on , vol., no., pp.367-375, 10-13 July 2006.

[5] Scott, A.; Watkins, M.; McPhee, D.; , "E-Learning For Novice Programmers; A Dynamic Visualisation and Problem Solving Tool," Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on , vol., no., pp.1-6, 7-11 April 2008.

[6] Westphal B, Harris F and Fadali M, “Graphical Programming: A Vehicle for Teaching Computer Problem Solving”, 33rd ASEE/IEEE Frontiers in Education Conference, IEEE, Boulder Colorado, 2003, pp: 19-23.

[7] Ben-Bassat Levy R, Ben Ari M and Uronen P, “An Extended Experiment with Jeliot 2000”, In Proceedings of the First International Program Visualization Workshop, University of Joensuu Press, Porvoo Finland, 2001, pp: 131-140

[8] D. J. Barnes and M. Kölling. Objects First with Java – A Practical Introduction using BlueJ. Prentice Hall/Pearson Education, Reading, Massachusetts, USA, 2003.

[9] R. Oechsle and T. Schmitt. JAVAVIS: Automatic Program Visualization with Object and Sequence Diagrams Using the Java Debug Interface (JDI). In S. Diehl, editor, Software Visualization, volume 2269 of Lecture Notes in Computer Science, pages 176–190. Springer-Verlag, 2002.

[10] Fowler, L., Allen, M., Armarego, J., and Mackenzie, J. Learning styles and CASE tools in Software Engineering. In A. Herrmann and M.M. Kulski (eds), Flexible Futures in Tertiary Teaching. Proceedings of the 9th Annual Teaching Learning Forum, February 2000.

[11] Thomas, L., Ratcliffe, M., Woodbury, J. and Jarman, E. Learning Styles and Performance in the Introductory Programming Sequence. Proceedings of the 33rd SIGCSE Symposium (March 2002), 33-42.

[12] Barker, R. T., & Holley, C. L. (1996). Interactive distance learnig:Perspective and thoughts. Business Communication Quarterly, 59 (4), 88-97.

[13] Kop, R., Fournier, H., & Mak, J. S. F. (2011). A pedagogy of abundance or a pedagogy to support human beings? Participant support on massive open online courses. International Review of Research in Open and Distance Learning, 12(7), 74-93.

[14] Fini, A. (2009). The Technological Dimension of a Massive Open Online Course: The Case of the CCK08 Course Tools. The International Review of Research in Open and Distance Learning, 10(5). Mushtak Al-Atabi, Jennifer DeBoer, Teaching entrepreneurship using Massive Open Online Course (MOOC), Technovation, Volume 34, Issue 4, April 2014, Pages 261-264.