Recommendations to Improve Programming Skills of Students of Computer Science

Fatmah Yousef Assiri
Faculty of Computing and Information Technology
King Abdulaziz University
Jeddah, KSA 21589
Email: fassiri@kau.edu.sa

***Abstract*—Computer programming is one of the main skills that students gain when they graduate from computer science programs. However, students often have significant difficulties with their programming skills. This paper is based on a survey study that summarizes the primary reasons for students difficulties in this area. In addition, methodologies, such as seeing the software big picture and pair programming, are suggested by which students can improve their programming skills. These methods are adapted from software engineering subdisciplines.**

Keywords*—*Programming skills; Students; difficulties; Pair programming; Game development, Practical examples; Practice.

I. Introduction

Computer programming is the process of solving problems by creating executable programs, which are also known as computer software. To develop a specific type of software, developers study software requirements and implement them through the use of programming languages, such as Java, C#, Visual Basic .NET (VB.NET), etc.

Computer programming is challenging, and students struggle to be proficient in this skill. Many factors can negatively impact the degree to which students are interested in the topic and their ability to obtain the required knowledge. Researchers have investigated factors that cause students to have difficulty when learning programming subjects. Some relate the difficulties to the required wide-ranging knowledge of different areas, such as problem-solving skills and deep understanding of different programming languages. Others relate the drop out rate in programming subjects to lack of student understanding of program execution that increases the difculty in comprehending programming subjects.

We found that students' lack of understanding of the problem domain, programming languages, and/or development environments is among the main reasons for student difficulties with programming subjects. Many suggestions were proposed in the literature to improve students programming skills, such as game development and practical exercises. However, no studies propose to use software engineering methodologies as a way to improve student performance in programming courses.

The present research is a survey study to: (1) summarize some of the factors that can negatively influence student ability to become proficient in programming subjects (Section I), and (2) propose methods using software engineering subdisciplines, such as software modeling, software debugging, and pair programming, along with other methods, to improve students' understanding of programming subjects, and thereby, increase their programming skills (Section II).

sectionStudent Difficulties with Programming

Students face many difficulties when learning computer programming. In an investigation into the reasons for these difficulties, Mow [1] suggested that these issues may be related to difficult programming languages and/or development environments. Programming languages' syntax and semantics are very different; students/programmers need to understand the details of programming languages to be proficient in them. In addition, students need expertise with different development environments to use the compiler, read error messages, and understand memory usage. Mow also mentioned the difficulty students have in differentiating between a problem and a solution; problems need to be understood before solutions are designed.

Tan et al. [2] investigated the difficulties of understanding programming subjects among 182 undergraduate students. They found the main difficulty for students is understanding how programs are executed. These results are similar to those found by Milne and Rowe [3]. Moser [4] relates the difficultes to the way programming langauges are explained to the students; explaining programming languages in a single context may negatively impact their ability to learn other programming languages [4].

There are other factors can impact students' ability to learn. For example, many students do not have previous experience in programming. Therefore, student struggles might be due to course contents that do not cover related background materials, assuming they were covered in previous classes. In addition, different levels of student skills in one classroom might intimidate them learning the subjects [1].

II. Improving Programming Skills Among Students

The education process can help students strengthen their programming skills. Therefore, this paper presents specific methodologies with which educators can help their students develop skills in this area.

A. Seeing the Big Picture

Problem solving, which is the process of understanding the problem domain, identifying solutions, and transferring the solutions into the computer using programming languages, is a

crucial skill for developers. Students need to be trained to gain a thorough understanding of the problem before beginning to determine a solution.

Training in software modeling helps students gain this understanding of the problem. This is the first step in software development cycles after collecting requirements. Software modeling involves gaining the required background knowledge of the problem, studying the requirements that are necessary to solve the problem, and using modeling languages, such as the Unified Modeling Language (UML), to model a system's structure. UML models a system in terms of classes, attributes, operations, and relationships between classes [5], [6]. It provides the overall picture of a system's different parts and the connections between these parts. Figure 1 is a UML class diagram for a system that models phone call originating and billing. The system consists of eight classes (PhoneBill, PhoneCall, Phone, Origin, ModileOrigin, MobileCall, CellPhone, and FixedPhone) and the relationships. A UML class diagram can help students understand a system's architecture before proceeding with implementation in detail.

There are other UML diagams that can be used to model different perspective of the systems. For example, use case diagrams describe the relationships between the user and the system and sequence diagrams describe the relationships of objects using ordered sequences of messages [7]. Researchers have investigated the benefits of using UML modeling to improve developers' understanding of the system and software quality [8], [9]. Using UML improved system correctness significantly (54% increase in program correctness) [9]. There has been no empricial study that investigated the benefits of using UML modeling to improve students' understanding of the problem and thus thier programming skills; we plan to study this in future research.

B. Practice

Learning by doing improves student programming skills [1], [2], [10]. With this approach, rather than exclusively listening to lectures, students have more time to practice coding. Practice can be completed in laboratory sessions and when independently completing coursework. When students practice in a lab, the educator acts as a guide to help students solve problems. In addition, students can work collaboratively in groups of two using a pair programming approach. Pair programming is a development technique in which two programmers work together in same design, code, or algorithm; one is the code writer, and the other is the observer [11].

Research shows that pair programming improves the quality of generated code within a shorter time period compared to individual programming in economic models [12], [13], [14]. Other studies have been conducted on the use of pair programming in computer science courses [12], [15], [16], [17]. Nagappan et al. [15] investigated about 700 students in two consecutive semesters, and they found that students groups that pair programmed performed better (higher scores) on exams and projects. Mendes [17] studied the impact of pair programming for 300 second-year computer science students at the University of Auckland in 2004. Pair programming improved the success percentage, which was defined as the percentage of students who passed the class with a grade of C or higher. In Cockburn [13], 74% of students reported that pair programming helped them to answer all their questions, and 84% learned new technology faster and better.

These findings establish pair programming as an effective learning technique, because it helps students perform better on projects and exams. More controlled empirical studies are needed to study the impact of pair programming as a method to improve student programming skills.

C. Practical Examples

Research identifies the use of examples as the best option for educators to help their students learn [1], [2], [10]. The use of examples, especially practical examples, can simplify programming subjects for students. For example, the physical space of the classroom can be used to represent a class in Object-Oriented Programming (OOP). Students can represent the objects. In addition, information from each student (e.g., name, age, and major) can act as the class attributes, and different values of these attributes can be used as object states. Using this method, the concept of OOP can become a logical and approachable topic for students. In addition, educators can incorporate examples into lectures. These examples can be made engaging to students by targeting their interests, such as building robots or creating sustainable food systems.

D. Game Development

Students can be encouraged and motivated to learn programming languages. Therefore, it is important for educators to pay close attention to the coursework that they design, including projects, assignments, and lab sessions. Students will become more engaged when solving and implementing computer games. A study by Cagiltay [18], [19] reveals that developing computer games strengthens problem-solving skills in students, helps them to learn independently, and refines their academic performance in general. Akcaoglu [20] evaluated problem-solving skills for middle school students who attended a summer program and practiced programming through a game development. The results showed a significant improvement in students' problem-solving skills: system analysis, design, decision-making, and troubleshooting.

Game development helps students improve their problem-solving skills or thinking skills, since they have to gain a full understanding of the game techniques before implementing. Thus, they will learn the problem domain (how the game works), propose a solution (how to implement it), and then implement the solution (write the game code).

E. Programming Environments

Programming environments, also known as integrated development environments (IDE), enable developers to program in different programming languages. It consist of source code editors and a set of tools to compile and execute code. They also offer debugging tools, which are used to track source code behavior by moving through the code and checking variable values at each step. Debugging is a rich process that can be used to help students understand a program's behavior [1]. Therefore, educators can help students by introducing them to and encouraging them to use programs and tools, including debugging, to understand code execution. As a result, their programming skills can be improved.

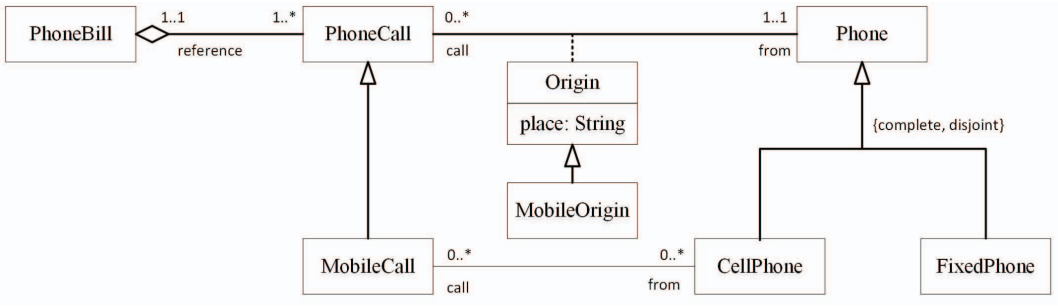

Fig. 1: UML class diagram for a phone call system.

F. Psychological Effects

Past experience can have a major impact on a student's ability to learn and practice programming. The range of programming capabilities that students have can be intimidating to students [1]. It is the educator's responsibility to encourage students to write code, to allow them to make mistakes, and to help them understand error messages. Educators may find it helpful to share their past experiences in programming with their students. This will also let students know that the educator once experienced the same difficulties that the students face, thereby helping students see their potential to become skilled programmers.

III. Recommendations

The following recommendations will help students strengthen their programming skills:

- It is highly recommended to introduce students to software modeling before they complete computer programming courses to help them formalize a problem and understand a system's structure prior to proceeding with coding.
- Students need to refine their problem-solving skills, which involves understanding the problem domain, analyzing and solving the problem that is under study, and translating the solution into executable code.
- Debugging tools are rich material that can help students to understand program execution by tracking variables and methods. It is highly beneficial to introduce students to debugging tools along with programming subjects.
- Educators can challenge students through lab work and assignments that are completed outside of class to improve their programming performance.

IV. Conclusion

This paper suggests that the education process is the main factor leading to the difficulties that students have with programming. Bridging the gap between the real world and programming subjects through the use of practical examples and game development can improve student understanding of programming concepts and make the topic more enjoyable. Students can be introduced to modeling and debugging tools to improve their knowledge of the system under study before they begin to code. In future research, I plan to study the difficulties that students have with programming at King AbdulAziz University and to develop course curricula that overcome the main difficulties students have with programming subjects.

References

[1] I. Mow, "Issues and difficulties in teaching novice computer programming," in *Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education*, M. Iskander, Ed. Springer Netherlands, 2008, pp. 199–204.

[2] P.-H. Tan, C.-Y. Ting, and S.-W. Ling, "Learning difficulties in programming courses: Undergraduates' perspective and perception," in *Computer Technology and Development, 2009. ICCTD '09. International Conference on*, vol. 1, Nov 2009, pp. 42–46.

[3] I. Milne and G. Rowe, "Difficulties in learning and teaching programmingviews of students and tutors," *Education and Information technologies*, vol. 7, no. 1, pp. 55–66, 2002.

[4] R. Moser, "A fantasy adventure game as a learning environment: why learning to program is so difficult and what can be done about it," in *ACM SIGCSE Bulletin*, vol. 29, no. 3. ACM, 1997, pp. 114–116.

[5] S. S. Alhir, *Unified Modeling Language (UML)*. John Wiley & Sons, Inc., 2002. [Online]. Available: http://dx.doi.org/10.1002/0471028959.sof365

[6] J. B. Warmer and A. G. Kleppe, "The object constraint language: Precise modeling with uml (addison-wesley object technology series)," 1998.

[7] G. Booch, *The unified modeling language user guide*. Pearson Education India, 2005.

[8] H.-E. Eriksson and M. Penker, "Business modeling with uml," *Business Patterns at Work, John Wiley & Sons, New York, USA*, 2000.

[9] W. Dzidek, E. Arisholm, and L. Briand, "A realistic empirical evaluation of the costs and benefits of uml in software maintenance," *Software Engineering, IEEE Transactions on*, vol. 34, no. 3, pp. 407–432, May 2008.

[10] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," in *ACM SIGCSE Bulletin*, vol. 37, no. 3. ACM, 2005, pp. 14–18.

[11] L. Williams and R. Kessler, *Pair programming illuminated*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[12] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," *IEEE software*, no. 4, pp. 19–25, 2000.

[13] A. Cockburn and L. Williams, "The costs and benefits of pair programming," *Extreme programming examined*, pp. 223–247, 2000.

[14] L. Williams and R. L. Upchurch, "In support of student pair-programming," in *ACM SIGCSE Bulletin*, vol. 33, no. 1. ACM, 2001, pp. 327–331.

[15] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik, "Improving the cs1 experience with pair programming," in *ACM SIGCSE Bulletin*, vol. 35, no. 1. ACM, 2003, pp. 359–362.

[16] A. Begel and N. Nagappan, "Pair programming: what's in it for me?" in *Proceedings of the Second ACM-IEEE international symposium on*

Empirical software engineering and measurement. ACM, 2008, pp. 120–128.

[17] E. Mendes, L. B. Al-Fakhri, and A. Luxton-Reilly, "Investigating pair-programming in a 2 nd-year software development and design computer science course," in *ACM SIGCSE Bulletin*, vol. 37, no. 3. ACM, 2005, pp. 296–300.

[18] N. E. Cagiltay, "Teaching software engineering by means of computer-game development: Challenges and opportunities," *British Journal of Educational Technology*, vol. 38, no. 3, pp. 405–415, 2007.

[19] M. Feldgen and O. Clua, "Games as a motivation for freshman students learn programming," in *Frontiers in Education, 2004. FIE 2004. 34th Annual*, Oct 2004.

[20] M. Akcaoglu, "Learning problem-solving through making games at the game design and learning summer program," *Educational Technology Research and Development*, vol. 62, pp. 583–600, 2014.