Towards Fostering Code Quality in K-12: Insights from a Literature Review

Elena Starke
elena.starke@tum.de
Computing Education Research Group
Technical University of Munich
Munich, Germany

Tilman Michaeli
tilman.michaeli@tum.de
Computing Education Research Group
Technical University of Munich
Munich, Germany

ABSTRACT

Code quality is a critical aspect of programming education in K-12, as it notably affects students' understanding of code and their ability to modify code. This poster presents a research project aiming at fostering code quality in K-12 education. As a first step, we explore several activities applied in educational contexts, like refactoring, using a linter, or performing a code review. The findings of this short review implicate that active participation in these activities has the potential to foster good programming habits.

CCS CONCEPTS

• **Social and professional topics → K-12 education**.

KEYWORDS

code quality, activities, K-12, computing education, refactoring

ACM Reference Format:
Elena Starke and Tilman Michaeli. 2023. Towards Fostering Code Quality in K-12: Insights from a Literature Review. In *The 18th WiPSCE Conference on Primary and Secondary Computing Education Research (WiPSCE '23), September 27–29, 2023, Cambridge, United Kingdom.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3605468.3609782

1 INTRODUCTION

Learning how to program is a major challenge for students in computing education. As a result, programming novices are often satisfied when their code fulfills the expected functionality and do not consider the (code) quality of their work [13]. Code quality is a multifaceted aspect of software quality that lacks a precise definition and encompasses various interpretations. In our perspective, code quality refers to the evaluation of the static characteristics of a program directly from its source code, which should be given due consideration after the initial programming phase [15]. Code smells, serving as indicators of non-functional aspects of code that require improvement rather than bugs or errors, are often associated with poor code quality [6]. Regarding K-12, bad code quality is a problem because it negatively affects students' ability to understand and modify code [9] and consequently their capacity to comply with the required functionality. Except for the knowledge that this problem exists, there is a lack of research on how we could address this in K-12. Therefore, we present a research project that aims to address code quality in K-12. As a first step, in this poster, we provide an initial review of the existing literature on activities designed to foster code quality in educational settings.

2 RELATED WORK

The existing body of research on code quality in educational settings has predominantly focused on text-based environments at the university level. In addition, there are suggestions as to what aspects of code quality are considered relevant in this context [20]. In contrast, studies in K-12 settings have primarily centered around block-based programming languages. With regard to K-12 students, despite demonstrating an understanding of code quality concepts [10], their code frequently exhibits code smells [9]. These problems often stem from poor programming habits, such as extremely fine-grained programming or extensive bottom-up programming, often seen among students working with block-based languages [17]. Additionally, code smells have been identified to have a detrimental effect on the ability to understand the intricacies of the system and the capacity to make changes to it [9]. This is consistent with the finding that code free of code smells is perceived as being more readable [16] and therefore easier to comprehend, which is an essential prerequisite for "debugging, refactoring, and extending the functionality" [11]. Furthermore, code quality issues can also be found in the code produced by university students. Also, in this context, this is an acknowledged problem [13] which needs explicit support in the classroom [4]. These results indicate that the high occurrence and the negative impact of code smells is a challenge for students, especially regarding bug and error identification.

3 ACTIVITIES TO FOSTER CODE QUALITY

Within professional software development, various activities are used to enhance code quality. These interventions have also found their way into educational contexts, where they are adapted and utilized to help students to improve code quality. Building upon this premise, our research project is grounded in the hypothesis that these activities contribute to improving problem-solving skills and facilitate the acquisition of a deeper understanding of good programming practices among novice students. In the following, we present a short literature review on existing findings regarding activities fostering code quality in educational settings. Therefore, we have identified research papers that not only propose interventions but also provide a brief evaluation of their implementation in the classroom setting.

3.1 Refactoring

Refactoring is a step-by-step process of improving the internal structure of code without changing its functionality [6]. Refactoring activities have been implemented, for example, within specific tools for educational purposes [14] or explicit course offers [2]. However, these teaching activities address university students, and there is currently a lack of equivalent studies specifically tailored for K-12 students. The results show that refactoring poses several challenges for students. To address these challenges, students can benefit from applying an approach that involves working in small steps and with code that is easy to comprehend. By adapting these strategies, students can reap the benefits of developing valuable problem-solving and analytical skills [2, 14].

3.2 Linter Usage

Linters are tools used for static code analysis to identify poor programming practices and prove valuable in professional software development for enhancing code quality as well as in educational settings [1]. For K-12 settings, there are specific linters for Scratch, such as the *LitterBox* [7] and *Hairball* [3]. They are designed to assist students in their programming endeavors. To this end, good and bad patterns are highlighted, and feedback is given. These patterns can indicate either a lack of understanding of programming concepts or a beneficial programming structure [7, 18]. Additionally, these tools aim to improve the feedback process for Scratch projects by automating certain aspects and providing valuable feedback to both students and teachers [3]. Considering text-based languages, the results indicate that resolving issues concerning design and best practice rules is more time intensive than rules related to code style and documentation. According to student feedback, the use of a linter, in general, was rated helpful in their programming process, with some students even noting an improvement in the readability of their code [1].

3.3 Code Review

Code Review, a manual inspection of source code to identify bugs and errors, is an important practice in preventing incorrect behavior and unexpected results [5]. By using checklists, reviewers can systematically examine code and recognize defects through clever questioning. Considering the use of such checklists in novice educational settings, it only leads to a marginal increase in the removal of defects [19]. Conversely, experienced students demonstrate the ability to generate appropriate review questions most of the time. Only in a few cases the questions are unclear, irrelevant, or considered testing or static analysis [5].

3.4 Further Approaches

Other approaches reported in the literature to improve students' code quality include providing feedback on the program structure through control flow graphs [12]. Additionally, a personalized quiz platform called *Foobaz* has been developed to provide feedback on variable names [8].

4 CONCLUSION AND FUTURE WORK

This review provides first insights into several approaches to enhancing students' code quality in educational settings. The findings implicate that integrating activities to improve code quality may help students tackle quality issues that hinder bug and error identification. Furthermore, we hypothesize that active engagement in code quality improvement activities promotes a deeper understanding of good programming practices and supports students in their learning process. While many of the explored activities in the literature focus primarily on university students, it is important to recognize that they can not be mapped directly to K-12 education. This is because this age group has its own unique requirements and educational goals. Currently, we are conducting a systematic literature review on code quality in educational settings to gain a comprehensive overview of what topics have been explored so far. Furthermore, we aim to investigate the specific aspects of code quality that are relevant to K-12 students and to explore effective strategies for teaching code quality concepts to this particular audience while also evaluating the effectiveness of these activities with K-12 students.

REFERENCES

[1] E. Abdullah AlOmar, S. Abdullah AlOmar, and M. Wiem Mkaouer. 2023. On the Use of Static Analysis to Engage Students with Software Quality Improvement: An Experience with PMD.

[2] C. Bezerra, H. Damasceno, and J. Teixeira. 2022. Perceptions and Difficulties of Software Engineering Students in Code Smells Refactoring. In *VEM*. SBC, Brazil, 41–45.

[3] B. Boe, C. Hill, M. Len, G. Dreschler, P. Conrad, and D. Franklin. 2013. Hairball: Lint-inspired Static Analysis of Scratch Programs. In *SIGCSE*. ACM, New York, NY, USA, 215–220.

[4] D. Breuker, J. Derriks, and J. Brunekreef. 2011. Measuring static quality of student code. In *ITiCSE*. ACM, New York, NY, USA, 13–17.

[5] C. Chong, P. Thongtanunam, and C. Tantithamthavorn. 2021. Assessing the Students' Understanding and their Mistakes in Code Review Checklists: An Experience Report of 1,791 Code Review Checklist Questions from 394 Students. In *ICSE-SEET*. IEEE, Madrid, 20–29.

[6] M. Fowler. 2019. *Refactoring: Improving the design of existing code* (second edition ed.). Addison-Wesley, Boston.

[7] G. Fraser, U. Heuer, N. Korber, F. Obermuller, and E. Wasmeier. 2021. LitterBox: A Linter for Scratch Programs. In *ICSE-SEET*. IEEE, Madrid, 183–188.

[8] E. Glassman, L. Fischer, J. Scott, and R. Miller. 2015. Foobaz: Variable Name Feedback for Student Code at Scale. In *UIST*. ACM, New York, NY, USA, 609–617.

[9] F. Hermans and E. Aivaloglou. 2016. Do code smells hamper novice programming? A controlled experiment on Scratch programs. In *ICPC*. IEEE, Austin, Texas, 1–10.

[10] F. Hermans and E. Aivaloglou. 2017. Teaching Software Engineering Principles to K-12 Students: A MOOC on Scratch. In *ICSE-SEET*. IEEE, Buenos Aires, 13–22.

[11] C. Izu, C. Schulte, A. Aggarwal, Q. Cutts, R. Duran, M. Gutica, B. Heinemann, E. Kraemer, V. Lonati, C. Mirolo, and R. Weeda. 2019. Fostering Program Comprehension in Novice Programmers - Learning Activities and Learning Trajectories. In *ITiCSE-WGR*. ACM, New York, NY, USA, 27–52.

[12] L. Jiang, R. Rewcastle, P. Denny, and E. Tempero. 2020. CompareCFG: Providing Visual Feedback on Code Quality Using Control Flow Graphs. In *ITiCSE*. ACM, New York, NY, USA, 493–499.

[13] H. Keuning, B. Heeren, and J. Jeuring. 2017. Code Quality Issues in Student Programs. In *ITiCSE*. ACM, New York, NY, USA, 110–115.

[14] H. Keuning, B. Heeren, and J. Jeuring. 2020. Student Refactoring Behaviour in a Programming Tutor. In *Koli Calling*. ACM, New York, NY, USA, 1–10.

[15] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2023. A Systematic Mapping Study of Code Quality in Education. In *ITiCSE*. ACM, New York, NY, USA, 5–11.

[16] U. Mannan, I. Ahmed, and A. Sarma. 2018. Towards understanding code readability and its impact on design quality. In *NL4SE*. ACM, New York, NY, USA, 18–21.

[17] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari. 2011. Habits of programming in scratch. In *ITiCSE*. ACM, New York, NY, USA, 168–172.

[18] F. Obermüller, L. Bloch, L. Greifenstein, U. Heuer, and G. Fraser. 2021. Code Perfumes: Reporting Good Code to Encourage Learners. In *WiPSCE*. ACM, New York, NY, USA, 1–10.

[19] G. Rong, J. Li, M. Xie, and T. Zheng. 2012. The Effect of Checklist in Code Review for Inexperienced Students: An Empirical Study. In *CSEE&T*. IEEE, Nanjing, China, 120–124.

[20] M. Stegeman, E. Barendsen, and S. Smetsers. 2016. Designing a rubric for feedback on code quality in programming courses. In *Koli Calling*. ACM, New York, NY, USA, 160–164.