Toward Code Review Notebooks

Juan Carlos Farah*, Basile Spaenlehauer*, María Jesús Rodríguez-Triana†, Sandy Ingram‡, and Denis Gillet*
*School of Engineering, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
Email: {juancarlos.farah,basile.spaenlehauer,denis.gillet}@epfl.ch
†School of Digital Technologies, Tallinn University, Tallinn, Estonia
Email: mjrt@tlu.ee
‡School of Engineering and Architecture of Fribourg, University of Applied Sciences (HES-SO), Fribourg, Switzerland
Email: sandy.ingram@hefr.ch

***Abstract*—Peer code review has proven to be a valuable tool in software engineering. However, integrating code reviews into educational contexts is particularly challenging due to the complexity of both the process and popular code review tools. We propose to address this challenge by designing a code review application (CRA) aimed at teaching the code review process directly within existing online learning platforms. Using the CRA, instructors can scaffold online lessons that introduce the code review process to students through code snippets, following a format resembling computational notebooks. We refer to this online lesson format as the code review notebook format. Through a case study comprising an online lesson on code quality standards completed by 23 university students, we evaluated the usability of the CRA and the code review notebook format, obtaining positive results for both. These results are a first step toward integrating code review notebooks into software engineering education.**

***Index Terms*—code review, software engineering education, code quality, online learning, computational notebooks**

I. Introduction

The importance of code reviews in software engineering education has long been recognized [1]. Nevertheless, teaching the code review process in an educational setting can be challenging, given the complexity associated with tools supporting code reviews. Social coding platforms, for example, have steep learning curves that can be overwhelming for entry-level students [2]. Moreover, code reviews are usually integrated into educational contexts as a peer review exercise through the use of social coding platforms or bespoke tools [3], [4] specifically designed for *peer* code review. Indeed, most tools used to support code review in education focus on the peer code review use case, providing features such as automatic review assignment and anonymous reviews [5]. While peer code review is particularly useful for students to get practical experience, fewer tools focus on providing individual students with a demonstration of the code review process. Examples include work by (i) Ardıç et al., who proposed a serious game in which players review predefined code snippets, identifying lines containing defects in order to advance to the next level [6], and (ii) Song et al., who developed a tool for peer code review that could also be used to explain the code review process by having students review code that had already been graded [7]. Nevertheless, these tools are often standalone applications that are not designed to be embedded directly within an online lesson or lecture. This lack of pedagogical code review tools that seamlessly integrate with existing online learning platforms motivates our work.

To address these challenges, we propose a code review application (CRA) focused on introducing students to the code review process. This CRA integrates with current online learning platforms to support lesson formats that have proven successful in software engineering education. Computational notebooks, for example, interpolate rich explanations with short snippets of code and are widely used in educational settings [8]. In this paper, we present the design and evaluation of both our CRA and its use inside an online lesson resembling a computational notebook. We refer to online lessons following this format as *code review notebooks*.

II. Design

The CRA's main interface consists of a static code snippet that can be annotated with comments (see Fig. 1). Through this interface, the CRA can support two different use cases, which are based on four contexts in which computational notebooks can support education: (i) lectures, (ii) flipped classroom settings, (iii) homework, and (iv) exams [8]. The first use case is for *explanatory* purposes, which corresponds to the use of computational notebooks for lectures and flipped classroom settings. Instructors can use the CRA to demonstrate the code review process. This is done by configuring the CRA so that it displays a code snippet with issues that the instructor wants to illustrate. These issues can then be explained by the instructor, who can (i) annotate the code snippet with comments before sharing the CRA with students—providing students with a complete example of a code review—or (ii) create those comments during a lecture, showing students how to complete the code review live. The second use case aims to provide students with *practical* experience reviewing code and corresponds to the use of computational notebooks for homework and exams. Instructors can configure the application with snippets containing issues that students are then required to identify. If presented as an unmarked exercise, this configuration is a way to provide students with practical experience. The same configuration can also be presented as a quiz or assessment to evaluate how well students understand the code review process.

To prototype our proposed code review notebook format, we integrated our CRA with Graasp, an open digital education

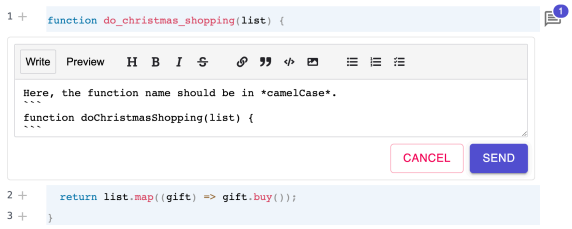

Fig. 1. The code snippet can be annotated with rich text and images.

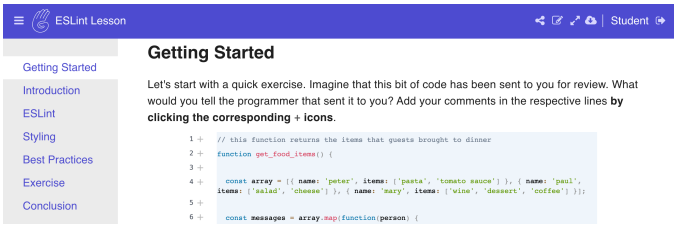

Fig. 2. The code review notebook comprised a lesson on code linting.

platform that allows instructors to create online lessons that support interactive applications [9]. Using Graasp, instructors can scaffold the CRA within an online lesson comprising multiple phases that a student can explore. Each phase can include text, images, and other media that serve to support the code review activity. Some phases can provide context (e.g., a phase introducing the lesson), while others can feature the CRA for its explanatory use case (e.g., a phase showing examples alongside explanations) or for its practical use case (e.g., an exercise). Once the instructors have set up the code review notebook, they can share it with students via a link.

III. Evaluation

As students are often the main actors in educational code review exercises, our focus was on evaluating the CRA's student interface and how well it can support software engineering education within an online lesson. To frame our evaluation, we conducted an instrumental case study [10]. Specifically, we assessed the *usability* of our CRA and the code review notebook format. An additional exploratory aspect consisted of *feature requests* to identify the functionalities that students would like to see in the CRA.

A. Context

To provide an ecologically valid environment for our case study, we created an online lesson following our proposed code review notebook format. The lesson introduced students to code linting in JavaScript. Code linting refers to the use of tools to detect bugs and other issues in software [11] and is often used to enforce code quality standards. Linting is thus a particularly fitting subject for our case study, as code review has been suggested as an effective pedagogical tool to teach code quality standards [12].

The lesson (shown in Fig. 2) was structured into seven phases. Two phases served to explain code linting. Three phases exploited the explanatory use case of the CRA to walk students through how code reviews work in practice. Finally, two phases focused on the practical use case. Following the *Fixer Upper* pedagogical pattern [13], students were presented with two exercises. Each exercise comprised a code snippet containing code quality issues that students were asked to identify. Issues included (i) not following naming conventions, (ii) mixing spaces and tabs, (iii) not declaring constants with `const`, and other violations of JavaScript best practices.

B. Methodology

We distributed our online lesson in January 2022 as an optional, 30-minute, ungraded exercise to students currently pursuing a degree at the École Polytechnique Fédérale de Lausanne and the School of Engineering and Architecture of Fribourg in Switzerland. Students contacted were asked to share the lesson with their peers. A total of 23 students consented to participate, completed the lesson presented in Section III-A, and responded to a post-questionnaire asking them to report on the usability of both the CRA and the code review notebook format, as well as to rate a list of features that they would like to see implemented in the CRA, and optionally provide demographic information about themselves.[1] Usability was measured using two standard instruments. For the CRA, we used the short version of the User Experience Questionnaire (UEQ-S) [14]. For the usability of the code review notebook format, we used the System Usability Scale (SUS) [15]. For the exploratory aspect we asked students to rate 10 features that could be added to the CRA on a scale of one (not interested) to five (very interested). Features were selected based on functionalities available on computational notebooks (e.g., *Executing Code*), social coding platforms (e.g., *Bots*), and intelligent tutoring systems (e.g., *Hints*).

1) *Automated Feedback*: Provide information about code quality.
2) *Peer Code Review*: Comment and edit other students' code.
3) *Bots*: Interact with bots within the application to get feedback.
4) *Code Editing*: Edit, save, and compare code versions.
5) *Hints*: Provide hints without giving away the answer.
6) *Heat Map*: Show which lines received comments from others.
7) *Executing Code*: Execute code within the browser.
8) *Fill In The Blanks*: Finish an incomplete code snippet.
9) *Dashboard*: Visualize one's activity against that of one's peers.
10) *Overview*: Summarize activity across all snippets in a lesson.

A total of 22 students (95.7%) responded to the demographic questions. Of these students, 5 (22.7%) were completing a bachelor's degree and 17 (77.3%) were completing a master's, while 9 (40.9%) were female and 13 (59.1%) were male. Furthermore, the mean self-reported overall programming experience on a scale of 1 to 5—with 1 being *Beginner* and 5 being *Expert*—was $\bar{x} = 3.18$ (Mode $= 4$). For JavaScript specifically, the mean was $\bar{x} = 1.95$ (Mode $= 1$).

Data were analyzed using descriptive statistics. Specifically, we report the sample mean ($\bar{x}$), median ($\tilde{x}$), and standard deviation (s). Responses from the UEQ-S were further processed

[1]The lesson and the questionnaire are available here: bit.ly/3kJwZbM.

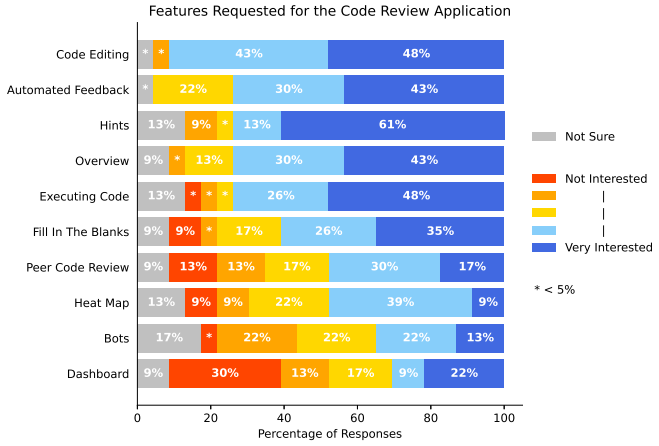

Fig. 3. Features requested for the CRA ordered by mean score (top to bottom).

using its standard data analysis tool, which compares results to benchmark data [14].

C. Results

The code review notebook format was evaluated using the SUS and received a mean score of $\bar{x} = 84.35$ ($\tilde{x} = 85, s = 12.46$), a usability score that can be described as *excellent* [16]. The CRA—evaluated using the UEQ-S—received a mean overall score of 1.29 ($\tilde{x} = 1.25, s = 0.81$), which is considered *positive* and *above average* (25% of results better, 50% of results worse) when compared to the benchmark. When considering pragmatic and hedonic qualities separately, the CRA achieved mean scores of $\bar{x} = 1.76$ ($\tilde{x} = 2.00, s = 0.91$) and $\bar{x} = 0.82$ ($\tilde{x} = 0.75, s = 1.01$), respectively. While both of these results are considered *positive*, the pragmatic score corresponds to an *excellent* result (top 10%) when compared to the benchmark, while the hedonic score is *below average* (50% of results better, 25% of results worse).

The three most requested features were *Code Editing* ($\bar{x} = 4.22, \tilde{x} = 4, s = 1.17$), *Automated Feedback* ($\bar{x} = 4.04, \tilde{x} = 4, s = 1.19$), and *Hints* ($\bar{x} = 3.87, \tilde{x} = 5, s = 1.79$). Results for the full set of features requested are shown in Fig. 3.

IV. Discussion

The results of our study show that the overall usability scores for both the CRA and the code review notebook format were excellent and above average, respectively. These scores suggest that the CRA and the code review notebook format lead to positive user experiences (UXs) within an educational context. Additionally, the CRA received an excellent score—top 10% when compared to the benchmark—with respect to its pragmatic qualities. That is, users perceive its design as well-suited for a task-oriented UX. Results for the feature request aspect of our study provide key insights into how we can improve our CRA to support functionalities that students will find useful. The three most requested features (*Code Editing*, *Automated Feedback*, and *Hints*) illustrate how students perceive the CRA's potential. *Code Editing*, on the one hand, would enhance the CRA to more closely resemble the development process supported by social coding platforms. *Automated Feedback* and *Hints*, on the other hand, focus on the educational aspect of the CRA, expanding support for more pedagogical scenarios, such as those typically supported by intelligent tutoring systems intended for programming education [17]. Integrating these features could also make the CRA a more novel and engaging tool, thus improving its usability score with respect to its hedonic qualities, which were rated as below average in the current implementation.

V. Conclusion, Limitations, and Future Work

In this study, we presented the design and evaluation of a CRA aimed at integrating the code review process into software engineering education. Nevertheless, our study has limitations worth addressing. Specifically, even though our use of Graasp is suitable for a proof-of-concept, it is important to show how the CRA can be incorporated into other online learning platforms to highlight its portability. Furthermore, we plan to complement these results by evaluating the impact the CRA has on learning gains and student engagement. We aim to address these limitations and integrate the results of our exploratory findings in future work.

References

[1] M. Towhidnejad and A. Salimi, "Incorporating a Disciplined Software Development Process in to Introductory Computer Science Programming Courses: Initial Results," in *FIE 1996*, vol. 2. IEEE, 1996, pp. 497–500.
[2] A. Zagalsky, J. Feliciano, M.-A. Storey, Y. Zhao, and W. Wang, "The Emergence of GitHub as a Collaborative Platform for Education," in *CSCW 2015*. ACM, 2015, pp. 1906–1917.
[3] J. G. Politz, S. Krishnamurthi, and K. Fisler, "CaptainTeach: A Platform for In-Flow Peer Review of Programming Assignments," in *ITiCSE 2014)*. ACM, 2014, p. 332.
[4] M. Tang, "Caesar: A Social Code Review Tool for Programming Education," Master's Thesis, MIT, 2011.
[5] T. D. Indriasari, A. Luxton-Reilly, and P. Denny, "A Review of Peer Code Review in Higher Education," *ACM Transactions on Computing Education*, vol. 20, no. 3, 2020.
[6] B. Ardıç, İ. Yurdakul, and E. Tüzün, "Creation of a Serious Game for Teaching Code Review: An Experience Report," in *CSEE&T 2020*. IEEE, 2020, pp. 204–208.
[7] X. Song, S. C. Goldstein, and M. Sakr, "Using Peer Code Review as an Educational Tool," in *ITiCSE 2020*. ACM, 2020, pp. 173–179.
[8] K. J. O'Hara, D. Blank, and J. Marshall, "Computational Notebooks for AI Education," in *FLAIRS 2015*. AAAI, 2015, pp. 263–268.
[9] D. Gillet, I. Vonèche-Cardia, J. C. Farah, K. L. Phan Hoang, and M. J. Rodríguez-Triana, "Integrated Model for Comprehensive Digital Education Platforms," in *EDUCON 2022*. IEEE, 2022, pp. 1586–1592.
[10] R. E. Stake, *The Art of Case Study Research*. Sage, 1995.
[11] S. C. Johnson, "Lint, A C Program Checker," 1978.
[12] X. Li and C. Prasad, "Effectively Teaching Coding Standards in Programming," in *SIGITE 2005*. ACM Press, 2005, p. 239.
[13] J. Bergin, "Fourteen Pedagogical Patterns," in *EuroPLoP 2000*, M. Devos and A. Rüping, Eds. Universitaetsverlag Konstanz, 2000, pp. 1–19.
[14] M. Schrepp, A. Hinderks, and J. Thomaschewski, "Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S)," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 6, pp. 103–108, 2017.
[15] J. Brooke, "SUS: A 'Quick and Dirty' Usability Scale," in *Usability Evaluation In Industry*, 1st ed. CRC Press, 1996.
[16] A. Bangor, P. Kortum, and J. Miller, "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale," *Journal of Usability Studies*, vol. 4, no. 3, pp. 114–123, 2009.
[17] T. Crow, A. Luxton-Reilly, and B. Wuensche, "Intelligent Tutoring Systems for Programming Education: A Systematic Review," in *ACE 2018*. ACM, 2018, pp. 53–62.