Harnessing Rule-Based Chatbots to Support Teaching Python Programming Best Practices

Juan Carlos Farah[1,2], Basile Spaenlehauer[1], Sandy Ingram[2], Aditya K. Purohit[3], Adrian Holzer[4], and Denis Gillet[1]

[1] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland,
{juancarlos.farah,basile.spaenlehauer,denis.gillet}@epfl.ch
[2] University of Applied Sciences (HES-SO), Fribourg, Switzerland,
{juancarlos.farah,sandy.ingram}@hefr.ch
[3] Radboud University, Nijmegen, The Netherlands,
aditya.purohit@ru.nl
[4] University of Neuchâtel, Neuchâtel, Switzerland,
adrian.holzer@unine.ch

Abstract. In recent years, the use of chatbots in education has been driven by advances in natural language processing and the increasing availability of digital education platforms. Although the added value of educational chatbots appears promising, researchers have noted that there is a need for empirical studies that explore the effects of incorporating chatbots into different learning scenarios. In this paper, we report on the integration of a rule-based chatbot into an information technology course. We conducted a controlled experiment in which half of the students were able to interact with the chatbot during Python lab sessions while the other half completed the sessions without the chatbot. Our results suggest that educational chatbots powered by short, simple, interactive scripts could have a positive impact on the user experience offered by learning technologies and could be pertinent to educators looking to integrate chatbots into their practice.

Keywords: educational chatbots, digital education, Python, empirical study, programming best practices

1 Introduction

Chatbots have been poised to disrupt educational technologies for well over a decade [30]. Nevertheless, researchers have found that there is still a lack of empirical studies on the use of chatbots in education [15]. In this study, we present results from a controlled experiment conducted within a university-level information technology course. Our study aimed to shed light on how rule-based chatbots could be harnessed to support students learning Python programming best practices as defined by the PEP-8 standard [24]. To that end, we configured a chatbot to follow a predefined script designed to show students that a particular code style guideline was useful in practice. The results from our case study contribute to the growing body of research on educational chatbots and may be relevant to practitioners looking to integrate these chatbots into their courses.

2 Background and Related Work

Recent reviews of the literature have highlighted that chatbots are often used to teach computer-related topics [18, 31]. In software engineering education, educational chatbots have been specifically harnessed in database, programming, computer networks, and compiler courses [18]. For example, Coronado *et al.* proposed a personal agent to support students in learning the Java programming language [5]. Their empirical evaluation, which relied on both objective and subjective metrics, showed that incorporating social dialog through question-answering agents increased user satisfaction and engagement with the system in which the agents were deployed. Mad Daud *et al.* also proposed a chatbot for learning Java [22]. By generating different code control structures, the proposed *e-Java* chatbot helped students in learning different ways of coding solutions for the same problem. Custom surveys were used as evaluation metrics to assess the perceived usefulness of the proposed chatbot. Going beyond self-reported metrics, an empirical study conducted by Winkler *et al.* evaluated information retention and transfer ability using different types of conversational agents (e.g., scaffolding vs. non-scaffolding, text vs. voice-based) [29]. Their evaluation using introductory Python programming tutorials achieved positive results.

Our work adds to this growing body of research by addressing the use of chatbots to support conducting code reviews in educational contexts. Conducting peer reviews for software verification and validation is one of the topics that the IEEE Computer Society and the Association for Computing Machinery recommend for computer science curricula [16]. Indeed, code reviews are an integral part of the software development process [28] and while there are several tools to support code reviews [2, 13], most collaborative software development involving the code review process currently takes place on social coding platforms. These platforms feature interfaces for reviewing and annotating code, providing a backdrop for discussions between developers. Chatbots can help with this process and have thus become a common feature of the social software development experience [20], helping to reduce manual labor, improve code quality, and increase productivity [26, 27]. However, very few studies have focused on the use of chatbots to support the code review process in formal education settings. We build on a previous Wizard of Oz [6] experiment [10] and a pilot study [8] to explore the impact that rule-based chatbots supporting code review exercises could have on the learning experience. In the following section, we present our guiding research question and the methodology we followed for our evaluation.

3 Methodology

Our evaluation aimed to address the following research question: *What are the effects of a rule-based chatbot designed to support Python programming lessons on students' learning experiences?* We addressed this research question by conducting a between-subjects controlled experiment comprising one control and one treatment. In both conditions, students were presented with the same lesson. The conditions differed only in the way we explained the code style issues

illustrated by the example code snippets. To frame our evaluation, we focused specifically on four aspects of the learning experience: (i) learning gains achieved, (ii) perceived usefulness of the material, (iii) user experience of the lesson, and (iv) feedback. In this section, we explain the methodology of our evaluation in detail.

3.1 Pedagogical Scenario

The main evaluation took place within the five-week Python programming component of a 14-week information technology course tailored for students completing a bachelor's degree in economics and business at the University of Neuchâtel, Switzerland. A total of 97 students were enrolled in the course. Before the beginning of the course, students were randomly assigned to the treatment (chatbot) or control (no chatbot) group. As outlined in Table 1, each week, students took part in an in-class lecture. The same week, they were able to attend a lab session in which the topics covered in the lecture were reviewed and a code style exercise was presented. Participation, however, was not mandatory. The course was conducted in French and all the material—including the chatbot scripts—was presented in French. For convenience, scripts, screenshots, student responses, and other material presented herein have been translated into English by the authors.

Table 1: Weekly lecture topics included in the Python programming component of the course alongside their corresponding lab session and code style exercises.

Week	Lecture	Lab	Code Style Exercise
1	Conditions	Conditions	Pre-Test
2	Loops	Loops	Indentation, Whitespace, Constants
3	Lists	Lists	Comparisons, Negating, Comparing Booleans
4	Functions	Functions	Max Length, Function Names, Descriptive Names
5	—	Review	Post-Test

Each lab session was conducted in French and included an exercise on Python code style guidelines based on the PEP-8 standard. These exercises were structured as code review notebooks [9] and followed the *Fixer Upper* pedagogical pattern both for explanation and evaluation [3]. That is, students were presented with code snippets that included code styling violations and were shown how to correct them (explanation) or asked to identify the issues present (evaluation). The first lab introduced students to the exercises and included a short activity that served as a pre-test. The second, third, and fourth labs each included explanations covering code layout, coding standards, and naming standards, respectively. In the final session, students completed a second activity that served as a post-test. Sessions were supported by the *Graasp* learning experience plat-

form [11] and *Code Review*, an application that allows students to annotate code and supports dialogs with chatbots [9].

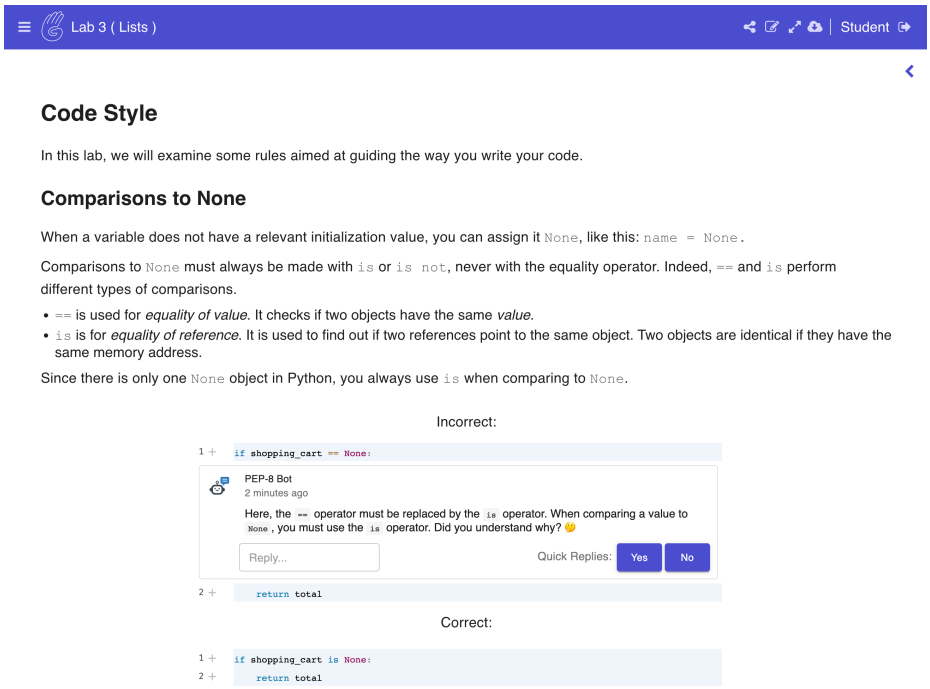

Fig. 1: Our chatbot was integrated into an application that was embedded in a code review notebook aimed at teaching Python programming.

Chatbot For this case study, we equipped Code Review with *PEP-8 Bot*. This chatbot was used to annotate the lines of code that contained potential code style issues within a series of Python code snippets used in Labs 2–5. The chatbot was configured to engage students by asking them if they understood and agreed with the logic behind the code style issue at hand, providing explanations of Python programming best practices, and motivating the reasons behind those best practices. An example of the chatbot embedded in the code review notebook is shown in Figure 1. Furthermore, we included interactions featuring emoji and animated gifs, taking advantage of the Markdown [12] support provided by Code Review. This allowed our chatbot to express—among other emotions—humor and confusion, as shown in Figure 2.

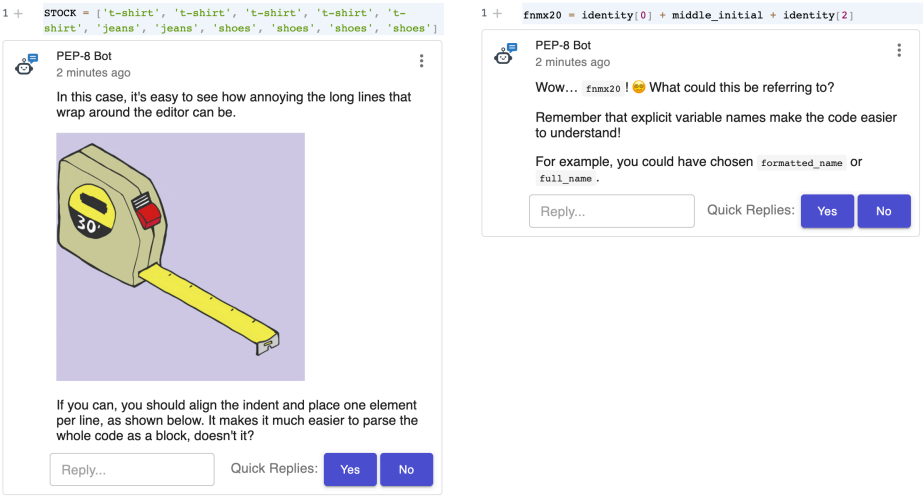

Fig. 2: Some of the chatbot's comments featured animated gifs (left) and emojis (right) to elicit humor and express confusion, respectively.

3.2 Procedure

At the beginning of each lab session, students were asked to complete a short code style exercise. Each exercise was meant to take approximately five minutes to complete. Lab 1 included the pre-test and was identical for all students. In Labs 2–4, each code style exercise covered three guidelines, spanning a total of nine guidelines (see Table 1). Finally, in Lab 5, students completed the post-test, after which they were shown the solutions to the test. For each guideline—and for the solutions to the post-test—students were presented with a short explanation followed by a code snippet containing an issue (incorrect snippet) and a code snippet with a correction of the issue (correct snippet). In the incorrect snippet, on the line containing the issue, students in the treatment group were also shown a comment from PEP-8 Bot that provided a further explanation and prompted the student to start a dialog about the usefulness of the guideline. Students in the control group did not see this comment.

3.3 Participants

There were 97 students enrolled in the course (44 female, 53 male). A total of 89 students accessed at least one of the exercise sessions and 65 students accessed all the sessions.

3.4 Instruments

We operationalized the four aspects of the learning experience as follows. Learning gains were calculated by taking the difference between students' scores on

the pre-test and the post-test. This yielded a learning gain that could range from -100% to 100%. Usefulness was measured using a seven-point Likert scale. Students were asked to rate how useful they found each individual guideline on a scale of 1 (*not useful at all*) to 7 (*very useful*). After the final exercise, using the same scale, they were asked to rate the overall usefulness of the code style exercises as a whole. User experience was captured with the *User Experience Questionnaire* (UEQ), a standard instrument that measures user experience across six dimensions [19]. Finally, feedback was assessed using the following question: "*Do you have any suggestions or comments on the parts of the labs that dealt with code style guidelines?*"

3.5 Data Analysis

We analyzed quantitative data using descriptive and inferential statistics, reporting sample means ($\bar{x}$), medians ($\tilde{x}$), and standard deviations (s_x), as well as results from two-sample t-tests, where applicable. Qualitative feedback was analyzed using line-by-line data coding [4].

4 Results

In this section, we highlight our results with respect to each aspect considered.

4.1 Learning Gains

A total of 25 students (10 control, 15 treatment) completed the post-test required to calculate learning gains. In both groups, as shown in Figure 3, learning gains were positive. The students in the control group achieved a mean learning gain of 36.0% ($s_x = 31.3\%$), while those in the treatment group achieved a mean learning gain of 36.7% ($s_x = 30.6\%$). As expected, the results of a two-sample t-test did not yield significant results ($p = 0.958$).

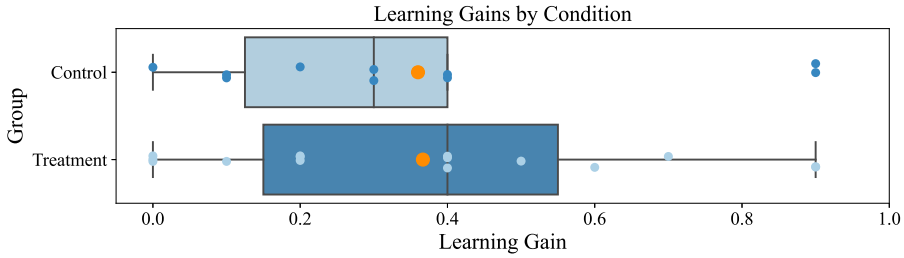

Fig. 3: Learning gains were positive for both groups, but there were no significant differences across conditions.

4.2 Perceived Usefulness

Regarding perceived usefulness, 46 students (26 control, 20 treatment) provided a total of 232 ratings distributed across the nine different guidelines, while 14 students (5 control, 9 treatment) rated the overall usefulness of the code style exercises. As shown in Figure 4, ratings were on average positive (above the median rating of 4), both overall and across individual guidelines. Nevertheless, two-sample t-tests did not yield significant differences across the groups. Furthermore, the number of students who provided the ratings decreased over time. While 34 students rated the first guideline, only 17 students rated the last one. It is important to note, however, that there were no significant differences in how this diminishing trend manifested itself in each of the two conditions.

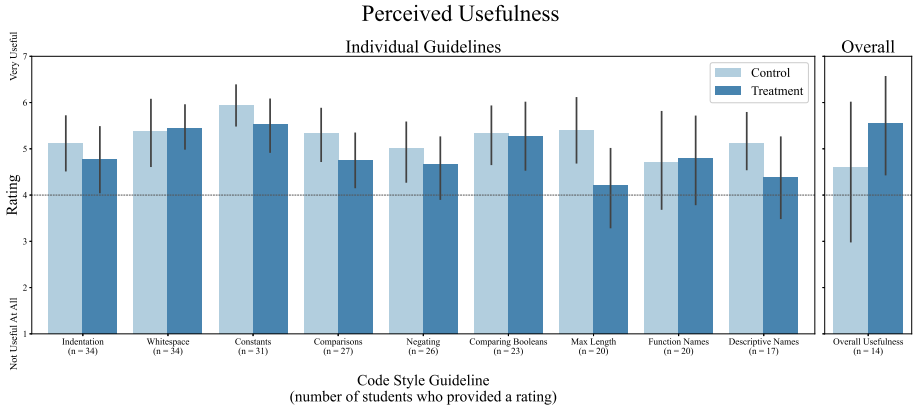

Fig. 4: On average, students rated how useful they found each guideline and the overall usefulness of the code style exercises positively (above the median rating of 4), but there were no significant differences across groups.

4.3 User Experience

A total of 27 students (14 control, 13 treatment) completed the UEQ. On average, students in the control group provided negative ratings for four of the six dimensions, while students in the treatment group provided positive ratings across all dimensions (see Figure 5). This difference was more pronounced in the efficiency ($p = 0.0127$), dependability ($p = 0.0565$), and perspicuity ($p = 0.0807$) dimensions.

4.4 Feedback

A total of 10 students (5 control, 5 treatment) provided qualitative feedback in the form of short open-ended responses. Four themes emerged in the responses:

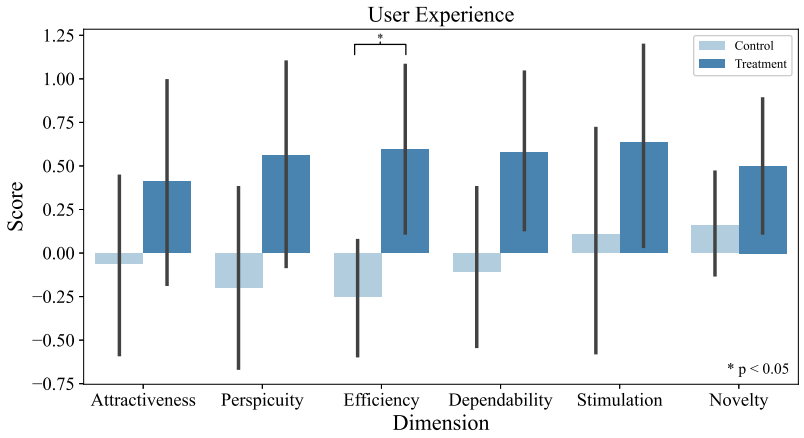

Fig. 5: User Experience Questionnaire (UEQ) ratings were consistently higher for the treatment group, especially for the efficiency ($p = 0.0127$), dependability ($p = 0.0565$), and perspicuity ($p = 0.0807$) dimensions.

(i) *more support*, (ii) *more exercises*, (iii) *useful*, and (iv) *useless*. First, *more support* was requested by four students (one control, three treatment), who suggested that instructors help clarify doubts and provide more information on the exercises. These requests are exemplified by a comment from a student in the treatment condition, who underlined that a "human touch would be nice" when referring to the lesson. Second, two students in the control group requested *more exercises* as a way to better assimilate the material. Third, two students in the treatment condition expressed that the exercises were *useful*, with one student providing the following comment:

> "Guidelines that addressed code style that made code more readable (layout, balance between upper and lower case, convention in writing style) were very helpful. The little touches of humor in [Lab 4] made the [lab] much more interesting. In general, the practical guidelines of the code allowed the course to be more complete."

Finally, two students in the control group referred to the exercise as *useless*, with one explicitly saying that they were "quite useless and boring compared to normal lab exercises".

5 Discussion

The results of our evaluation show that although our chatbot integration did not affect learning gains or the perceived usefulness of the material, it did have an impact on the user experience of the lesson. On the one hand, results regarding

learning gains could be explained by the fact that the chatbot interaction was primarily designed to reiterate the explanation that was already present in the text and—if needed—persuade the student that the guideline in question was useful in practice. On the other hand, for the same design reason, we would have expected students who were exposed to the chatbot to perceive the code style guidelines as more useful than students who did not hold those interactions, which was not the case. However, as evidenced by the decreasing number of students who provided ratings as the course progressed, this result could have been influenced by a diminishing novelty effect that curbed student interest in the code style exercises.

The lack of significant differences in learning gains between conditions could be interpreted positively. In line with a suggestion made by Hobert [14], this result bolsters the idea that educational chatbots could support learners when teaching staff are not available or in scenarios with large numbers of students. Given that learning is not impacted, educational chatbots could serve as an additional layer of interaction for learners seeking more information.

The differences in user experience, however, were evident—albeit not always significant—across all dimensions of the UEQ, and are very promising. Results from the UEQ suggest that incorporating the chatbot into the interface improved student perception of the user experience of the lesson, especially in terms of efficiency, dependability, and perspicuity. This improvement could have been mediated by the fact that including the chatbot added interactivity to a lesson that was otherwise primarily explanatory.

Improvements in user experience that are not accompanied by improvements in learning gains have been observed in the literature. As discussed by Davids *et al.*, this lack of correlation could be due to the type of learners that are participating in the learning activity or the interface that is being optimized [7]. In the case of Davids *et al.*, their study was conducted with practicing clinicians who the authors describe as possibly being highly motivated and therefore less affected by the user experience improvements the authors were testing. In our case, the fact that the exercises were not mandatory might have led to a selection bias, where the most motivated students took part in code style exercises included as part of their lab work, possibly leading to a similar effect as the one observed by Davids *et al.* [7]. Nevertheless, an improvement in perceived user experience can have an impact on other dimensions, such as task completion rate [17], self-regulation [21], and motivation [32].

Further insights were provided by the qualitative feedback. Positive comments regarding the *usefulness* of the exercises were present exclusively in the treatment condition, while negative comments about the *uselessness* of the exercise were present exclusively in the control condition. This contrast suggests that including the educational chatbot made the exercises more meaningful, possibly shedding some light on why perceived user experience was higher in the treatment condition. Nonetheless, the fact that more support was requested by four students, including three in the treatment condition, indicates that even with

the inclusion of a chatbot, students still require a "human touch" in the learning process.

6 Conclusion, Limitations, and Future Work

In this paper, we presented results from an empirical case study assessing the effects of integrating educational chatbots into blended learning scenarios aimed at teaching Python programming best practices. The findings of our controlled experiment show that there were no significant differences in the learning gains achieved and the perceived usefulness of the lessons between students who completed the exercises alone and those who completed them with support from the chatbot. However, students who had access to the chatbot rated the user experience of the lesson more positively, particularly in the efficiency, dependability, and perspicuity dimensions of the UEQ. This improved user experience could motivate the integration of chatbots into the type of lessons used in this study.

It is important to note that this study has some limitations worth considering. First, given that the exercises were not mandatory, there could have been some selection bias in our sample, as possibly only the most motivated students interacted with the lesson. Second, while the rule-based scripts ensured that student interaction with the chatbot was on topic and pertinent to the lessons, the limited scope of these exchanges could have diminished how natural they appeared to the student and, therefore, discouraged students from interacting with the chatbot. These limitations could be addressed by (i) ensuring that all students are exposed to the exercises and (ii) equipping the chatbot with a generative language model. We will address these limitations and explore possible improvements through the use of large language models—such as those powering ChatGPT [23]—in future work.

Acknowledgments

Images used in this study include icons made by Vector Stall [25] and Bad Arithmetic [1].

References

1. Bad Arithmetic: Measuring Tape Measure Up (2017). URL https://giphy.com/embed/3og0IQttlo3NfcsIiQ
2. Baum, T., Schneider, K.: On the Need for a New Generation of Code Review Tools. In: P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, T. Mikkonen (eds.) Product-Focused Software Process Improvement, vol. 10027, pp. 301–308. Springer, Cham, Switzerland (2016). DOI 10.1007/978-3-319-49094-6_19
3. Bergin, J.: Fourteen Pedagogical Patterns. In: M. Devos, A. Rüping (eds.) Proceedings of the 5th European Conference on Pattern Languages of Programs (EuroPLoP 2000). Universitaetsverlag Konstanz, Irsee, Germany (2000)

4. Charmaz, K.: Constructing Grounded Theory: A Practical Guide through Qualitative Analysis. Sage, London, UK (2006)
5. Coronado, M., Iglesias, C.A., Carrera, Á., Mardomingo, A.: A Cognitive Assistant for Learning Java Featuring Social Dialogue. International Journal of Human-Computer Studies **117**, 55–67 (2018). DOI 10.1016/j.ijhcs.2018.02.004
6. Dahlbäck, N., Jönsson, A., Ahrenberg, L.: Wizard of Oz Studies—Why and How. Knowledge-Based Systems **6**(4), 258–266 (1993)
7. Davids, M.R., Chikte, U.M.E., Halperin, M.L.: Effect of Improving the Usability of an E-Learning Resource: A Randomized Trial. Advances in Physiology Education **38**(2), 155–160 (2014). DOI 10.1152/advan.00119.2013
8. Farah, J.C., Spaenlehauer, B., Bergram, K., Holzer, A., Gillet, D.: Challenges and Opportunities in Integrating Interactive Chatbots into Code Review Exercises: A Pilot Case Study. In: EDULEARN22 Proceedings, pp. 3816–3825. IATED, Valencia, Spain (2022). DOI 10.21125/edulearn.2022.0932
9. Farah, J.C., Spaenlehauer, B., Rodríguez-Triana, M.J., Ingram, S., Gillet, D.: Toward Code Review Notebooks. In: 2022 International Conference on Advanced Learning Technologies (ICALT), pp. 209–211. IEEE, New York, NY, USA (2022). DOI 10.1109/ICALT55010.2022.00068
10. Farah, J.C., Spaenlehauer, B., Sharma, V., Rodríguez-Triana, M.J., Ingram, S., Gillet, D.: Impersonating Chatbots in a Code Review Exercise to Teach Software Engineering Best Practices. In: 2022 IEEE Global Engineering Education Conference (EDUCON), pp. 1634–1642. IEEE, New York, NY, USA (2022). DOI 10.1109/EDUCON52537.2022.9766793
11. Gillet, D., Vonèche-Cardia, I., Farah, J.C., Phan Hoang, K.L., Rodríguez-Triana, M.J.: Integrated Model for Comprehensive Digital Education Platforms. In: 2022 IEEE Global Engineering Education Conference (EDUCON), IEEE Global Engineering Education Conference, pp. 1586–1592. IEEE, New York, NY, USA (2022). DOI 10.1109/EDUCON52537.2022.9766795
12. Gruber, J., Swartz, A.: Markdown (2004). URL daringfireball.net/projects/markdown/
13. Hedberg, H.: Introducing the Next Generation of Software Inspection Tools. In: T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, O. Nierstrasz, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, M.Y. Vardi, F. Bomarius, H. Iida (eds.) Product Focused Software Process Improvement, vol. 3009, pp. 234–247. Springer, Berlin, Germany (2004). DOI 10.1007/978-3-540-24659-6_17
14. Hobert, S.: Say Hello to 'Coding Tutor'! Design and Evaluation of a Chatbot-based Learning System Supporting Students to Learn to Program. In: 40th International Conference on Information Systems (ICIS 2019), vol. 3, pp. 1776–1792. Curran Associates, Inc., Red Hook, NY, United States (2020)
15. Hwang, G.J., Chang, C.Y.: A Review of Opportunities and Challenges of Chatbots in Education. Interactive Learning Environments (2021). DOI 10.1080/10494820.2021.1952615
16. Joint Task Force on Computing Curricula: Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Tech. rep., IEEE & ACM (2015)
17. Kanuka, H., Szabo, M.: Conducting Research on Visual Design and Learning: Pitfalls and Promises. Canadian Journal of Learning and Technology / La revue canadienne de l'apprentissage et de la technologie **27**(2) (1999). DOI 10.21432/T2SW37
18. Kuhail, M.A., Alturki, N., Alramlawi, S., Alhejori, K.: Interacting with Educational Chatbots: A Systematic Review. Education and Information Technologies **28**(1), 973–1018 (2023). DOI 10.1007/s10639-022-11177-3

19. Laugwitz, B., Held, T., Schrepp, M.: Construction and Evaluation of a User Experience Questionnaire. In: A. Holzinger (ed.) HCI and Usability for Education and Work, *Lecture Notes in Computer Science*, vol. 5298, pp. 63–76. Springer, Berlin, Germany (2008). DOI 10.1007/978-3-540-89350-9_6
20. Lebeuf, C., Storey, M.A., Zagalsky, A.: Software Bots. IEEE Software **35**(1), 18–23 (2018). DOI 10.1109/MS.2017.4541027
21. Liaw, S.S., Huang, H.M.: Perceived Satisfaction, Perceived Usefulness and Interactive Learning Environments as Predictors to Self-Regulation in e-Learning Environments. Computers & Education **60**(1), 14–24 (2013). DOI 10.1016/j.compedu.2012.07.015
22. Mad Daud, S.H., Ibrahim Teo, N.H., Mat Zain, N.H.: E-JAVA Chatbot for Learning Programming Language: A Post-Pandemic Alternative Virtual Tutor. International Journal of Emerging Trends in Engineering Research **8**(7), 3290–3298 (2020). DOI 10.30534/ijeter/2020/67872020
23. OpenAI: Introducing ChatGPT (2022). URL https://openai.com/blog/chatgpt
24. van Rossum, G., Warsaw, B., Coghlan, N.: Style Guide for Python Code. PEP 8, Python Software Foundation (2001). URL https://www.python.org/dev/peps/pep-0008/
25. Vector Stall: Assistant Free Icons. URL flaticon.com/free-icon/assistant_4818971
26. Wessel, M., Serebrenik, A., Wiese, I., Steinmacher, I., Gerosa, M.A.: Effects of Adopting Code Review Bots on Pull Requests to OSS Projects. In: Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, Adelaide, Australia (2020). DOI 10.1109/ICSME46990.2020.00011
27. Wessel, M., Serebrenik, A., Wiese, I., Steinmacher, I., Gerosa, M.A.: What to Expect from Code Review Bots on GitHub?: A Survey with OSS Maintainers. In: Proceedings of the 34th Brazilian Symposium on Software Engineering, pp. 457–462. ACM, Natal, Brazil (2020). DOI 10.1145/3422392.3422459
28. Wiegers, K.E.: Peer Reviews in Software: A Practical Guide. Addison-Wesley, Boston, MA, USA (2002)
29. Winkler, R., Hobert, S., Salovaara, A., Söllner, M., Leimeister, J.M.: Sara, the Lecturer: Improving Learning in Online Education with a Scaffolding-Based Conversational Agent. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. ACM, Honolulu, HI, USA (2020). DOI 10.1145/3313831.3376781
30. Winkler, R., Soellner, M.: Unleashing the Potential of Chatbots in Education: A State-Of-The-Art Analysis. Academy of Management Proceedings **2018**(1), 15903 (2018). DOI 10.5465/AMBPP.2018.15903abstract
31. Wollny, S., Schneider, J., Di Mitri, D., Weidlich, J., Rittberger, M., Drachsler, H.: Are We There Yet? - A Systematic Literature Review on Chatbots in Education. Frontiers in Artificial Intelligence **4**, 654924 (2021). DOI 10.3389/frai.2021.654924
32. Zaharias, P., Poylymenakou, A.: Developing a Usability Evaluation Method for e-Learning Applications: Beyond Functional Usability. International Journal of Human-Computer Interaction **25**(1), 75–98 (2009). DOI 10.1080/10447310802546716