EdCode: Towards Personalized Support at Scale for Remote Assistance in CS Education

Yan Chen[1], Jaylin Herskovitz[1], Gabriel Matute[1], April Wang[1], Sang Won Lee[2], Walter S. Lasecki[1], Steve Oney[1]

[1]*University of Michigan, Ann Arbor, United States, {yanchenm,jayhersk,gmatute,aprilww,wlasecki,soney}@umich.edu*
[2]*Virginia Polytechnic Institute and State University, Blacksburg, United States, sangwonlee@vt.edu*

***Abstract*—Programming support methods, like discussion forums and office hours, are important in CS education, but difficult to scale. In this paper, we introduce EdCode, a system that allows students to seek remote instructional support within their IDE in a way that resembles in-person support. It also allows instructors to provide contextualized responses by referencing students' code, and curate and publish their answers for an entire class by selecting only the relevant part of the code referenced, thereby helping to avoid plagiarism. We evaluated EdCode with a series of usability studies and identified benefits and challenges for its use in programming courses. Students found that the perceived quality of support from EdCode was comparable to that of support from in-person office hours, and both students and instructors found publishing and viewing other students' answers helpful.**

***Index Terms*—Programming Education, Remote Assistance, Scalable Support**

I. Introduction

The enormous growth of software development jobs has led to a rise in the desire to learn to code [1], [2]. As a result, demand for computer science (CS) courses has swelled and instructional resources have struggled to keep up with this demand. Prior work has shown that instructional resources, such as in-person support, can help students improve their performance in programming courses [3], [4]. In-person support makes it easy for instructors to access students' code, allows instructors to proactively help, and is personalized [5]. However, instructional support can be hard to scale in courses that have high student-teacher ratios, as many CS courses do.

Many courses use Q&A forums (e.g., Piazza) to scale support for students but forum participation is often low due to their public nature and the inability to have a natural conversation [6]–[11]. Several systems, including Codeon [12], have been shown to be effective at scaling remote programming assistance in work settings. However, programming support in educational settings is different than work settings in terms of goals, stakeholders' expertise, and collaboration structure. For instance, students often lack sufficient knowledge and understanding to phrase a question correctly compared to professional programmers. Instructors would prefer to guide students with hints and questions rather than giving away the solutions, whereas professional programmers tend to offer straightforward solutions that can be replicated by others.

978-1-7281-6901-9/20/$31.00 ©2020 IEEE

In this paper, we introduce EdCode, a remote support system that allows instructors to provide personalized assistance to students and publicly share their support with coding questions in programming classes. We conducted two needfinding studies and hypothesized that 1) supporting both asynchronous and synchronous interaction could help instructors better guide students' learning processes, 2) allowing instructors to refer to portions of code in their responses could make their explanations clearer, and 3) allowing instructors to select relevant code in students' questions to share with the entire class would help scale their support. EdCode instantiates these ideas by adding three new features on top of Codeon—**contextualized explanations**, which allows students to make text-based help requests with code context (by highlighting relevant code snippets) from their working context (IDE state) in asynchronous fashion; a **chat tool** that allows instructors to converse with students within each question; and **selective code publicity**, which allows instructors to publish students' questions and answers by selecting code regions that are relevant to the question while concealing the rest of the solution. In this way, other students could review questions and answers curated by the instructors, leading to fewer duplicate questions and saving time for instructors and students alike.

We conducted a virtual office hour study in an introductory programming course. Both student and instructor participants reported that the quality of answers from EdCode was better than their existing Q&A forum (Facebook Group). Students also found the answers were comparable to those obtained from in-person office hours, and they could understand the questions and answers with only the curated version of the original code visible. Our contributions are:

- study-based insights into the needs and challenges of using existing asynchronous remote support tools in programming classes;
- a system (EdCode) that addresses these needs and challenges, used as a 'technology probe'; and
- evidence that EdCode has the potential to be useful in a classroom setting.

II. Related Work

Community question-answering websites (e.g,. Stack Overflow) are common help-seeking systems, but they often lack personalized support for programmers, especially novices [13]–[15]. To scale personalized support, prior work has proposed

automated techniques to repair bugs [16], construct in-situ explanations for code examples [17], and provide intelligent tutoring [18]. While results are promising, they often require expertise to develop and can be difficult to adapt for programming courses [19], [20]. Codeopticon [21] allows teachers to monitor students' behavior and provide support by detecting when students need help. However, instructors must be proactive, and the help sessions are not archived or reusable. CrowdCode [22] enables code context capturing for request composition, yet it focuses on improving programmers' productivity, while we aim to help students receive more guided support to improve their learning experience.

In programming communication, relevant context (e.g., specific code lines) is important but challenging to capture. Chen et al. [23] showed that presenting context could improve the efficiency of help sessions, as helpers can provide feedback that is specific to a requester's code. Chat.codes [24] and Callisto [25] developed deictic code references for explicitly specifying a message's context. Inspired by these studies, we designed EdCode to automatically capture students' codebases and context, along with the request description.

III. Formative Studies

We conducted two studies to better understand the tradeoff between personalization and scalability for support in programming education. We conducted these studies with Codeon [12], an asynchronous remote programming support tool, to explore potential issues in the problem space. We chose Codeon because it uses an asynchronous communication paradigm (which can be more scalable than synchronous support) while enabling easy context sharing (which can make it easier to provide personalized support). Codeon has also been shown to improve the productivity of software programmers.

A. Method

We first interviewed three students and two instructors from an introductory programming class at the authors' university. We asked about their experiences with different programming support methods, including online forums and office hours. We then evaluated the usability of Codeon [12] in an educational setting with 11 different students (5 female, 6 male) and the two interviewed instructors from the same class by asking them to make and review questions they had previously asked with the assigned support tool (the course's Facebook Group). The instructor and student participants completed the study at different times. The first author (who was a teaching assistant in the same class) responded to students' questions using Codeon in real time. We then interviewed participants about their experiences using Codeon compared to other help-seeking methods.

B. Results

We observed three common problems across these studies. First, the responses that our researcher instructor gave through Codeon were difficult for students to interpret as students lacked the ability to efficiently switch between deciphering a textual explanation and parsing references to code, which were often included in responses. This is because Codeon was designed to help programmers complete tasks rather than guiding students to learn, and thus prioritizes allowing users to quickly integrate code snippets over explaining or annotating lines of code. Second, both the instructor and student participants mentioned that they sometimes misunderstood a student's question or the instructor's response respectively. Part of this miscommunication resulted from the instructor assuming prerequisite knowledge in their answer that the student actually lacked. Neither existing support tools nor Codeon are able to handle follow up questions that have code context within an existing request. Lastly, to scale their support, the instructor participants often wanted to publish previous responses for the entire classes to avoid redundant effort. Meanwhile, we found students would often include code context (e.g., screenshots) along with their questions on the course's Facebook group. This could encourage other students to plagiarize their code, which instructors prevented by removing their request. Students, in turn, were uncertain about what code they can and should share in their questions. Based on these findings, we designed three features in EdCode that can support efficient communication between instructors and students.

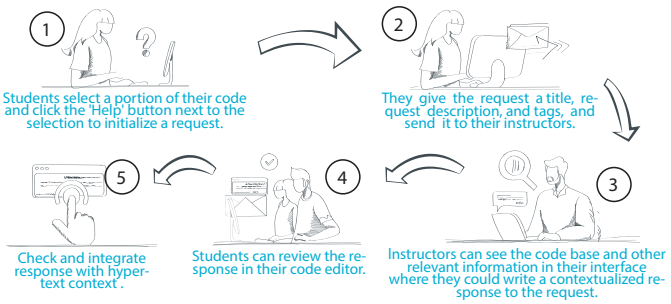

Fig. 1. A diagram shows the process of using EdCode to make a request, write a response, and integrate a response. (1) Students select a portion of their code and click the 'Help' button next to the selection to initialize a request. (2) Students attach a title, description, and tags to the request and send it to instructors. (3) Instructors can see the code base and other relevant information in their interface where they could write a contextualized response to the request. (4) Students can review the response in their code editor, and (5) check response context by clicking the hypertext.

IV. EdCode: Feature Design

A. Contextualized Explanations: Referring to Code with Hypertext

As we found in our formative studies, there is a high level of interdependency between natural language explanations and the referenced code. This can make it difficult for novice programmers to follow the text responses, as their mental model for programming is not robust enough to make connections between the different pieces of information they receive. Borrowing from the idea of hyperlinks, which are pervasive in web documents, we created a contextualized explanation feature that allows instructors to link a selection of code to textual explanations (Fig. 2). In students' code editor, they can click

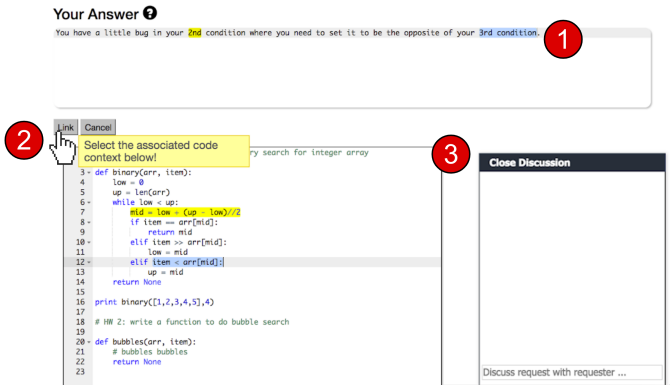

Fig. 2. The contextualized explanation feature and the chat tool. To make their answers more contextualized, instructors can (1) select a portion of the text in their natural language answer and the relevant code context; (2) click the 'link' button to create the references, which will highlight the relevant code (hypertext) when clicking the references in the text. Instructors can also chat with students via (3).

highlighted text in an answer to see the code associated with it (4, 5 in Fig. 1). This feature helps to lower the cognitive cost for novices by dereferencing and removing navigation effort of finding the relevant code [24]. It also helps instructors avoid the effort of clarifying each piece of code context in composing an answer (e.g., providing the line number or copying segments of code into the explanation).

B. Chat Tool: Supporting Hybrid Interaction

To allow for quick follow-up and clarification while still maintaining the Q&A structure, we created a chat feature (Fig. 2.3). The formative study showed that unlike face-to-face support, remote and asynchronous support available in Codeon (or online forums) lacks the ability for one party to interrupt another to immediately clarify a question or response. We aim to add a feature that can facilitate light-weight and real-time communication as if it were an in-person office hour. However, we did not change the model entirely to synchronous collaboration. We instead made the chat tool a supplementary communication channel to preserve the benefits of asynchronous communication, as both parties often need to engage in other activities without interruption (i.e., instructors need to work on other requests, while students need to fix other parts of their code). For this reason, we took a hybrid approach—supporting real-time interaction within the model of asynchronous support.

The chat tool is situated per-request, meaning each request will have its own chat box associated with the question and its code context so that the conversation will be centered around the question [26]. This is different from common instant chat messengers in which a conversation is displayed *separate* from the working context. Instructors are still encouraged to compose an answer using the contextualized explanation feature to close and publish the request later. This light-weight communication is not the primary way of composing an answer, but efficient for quickly resolving any discrepancies in understanding the question or the response (Fig. 2).

C. Code Publicity: Sharing Code Selectively

We allowed instructors to control what parts of students' code are made public for each request by selecting portions of code and making them visible to all students (see Fig. 3). The instructor can easily edit out sensitive information, eliminating guesswork on the part of the student as to what information should not be shared. Because students are able to send private requests to instructors with all of their code included, they do not have to determine what context is appropriate to include with their request; the context is captured automatically. Furthermore, instructors still benefit from being able to access the complete context of the question. This allows instructors to examine the code in depth and provide more personalized feedback, as they can view and execute the code without worrying about plagiarism and share a curated version of their answer with the rest of the class.

V. Evaluation

We conducted two studies: a virtual office hour study to evaluate the usability of EdCode and a comparison study for the contextualized explanation feature and the code publicity feature in EdCode.

A. Study 1: Virtual Office Hours

1) Study Setup: We recruited two instructors (I), and seven students (S) (3 female, 4 male) from an introductory Python class at the authors' university. Each instructor hosted an virtual office hour session for two hours using EdCode to provide remote support to the students: two in the first session and five in the second (unbalanced due to availability). The students were allowed to ask any questions relevant to the course assignments or coding exercises. This setup encouraged students to work at their own pace, giving them flexibility in how they used EdCode. Through a survey and interviews, we then compared their experience using EdCode to other support methods used in class: a course's Facebook Group and in-person office hours.

2) Results and Discussion: The students asked 30 questions in total, and instructors reported the questions as being realistic. **Usability:** All seven students reported an overall preference for EdCode, as it was *"convenient"* (S1, S3–S6), *"effortless"* (S2),

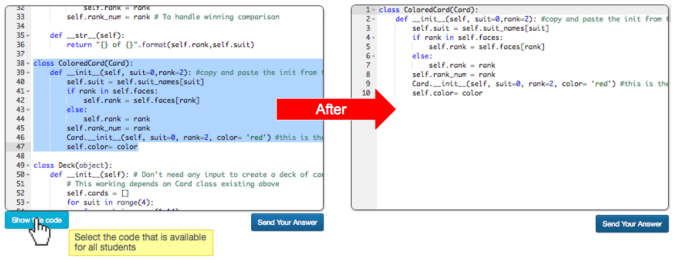

Fig. 3. The code publicity feature. Instructors first click the 'Show this code' button to select a portion of the code in original request that is necessary to understand the question for other students. Then they click 'Send Your Answer' button to publish the question and the answer to the entire class.

or *"localized to my text editor"* (S7). The flexibility between synchronous and asynchronous communication in EdCode allows students to *"easily follow up with the chat"* (S2) and *"ask questions wherever"* (S1). EdCode also encouraged one student to ask more questions: *"I felt better asking questions [using EdCode] than I would at office hour"* (S2).

Answer Quality: Students reported the answers from EdCode as being *"more direct"* (S4) and *"more in depth"* (S6) than those from the course's Facebook Group, and five reported the quality as close to (S7) or the same (S1,3,5,6) as that of in-person office hours. Some of the reasons that students gave were: the code in the response is editable and built on code from students' original questions; the answer format is more contextualized and guided than that of the course's Facebook Group, as *"it provides visual feedback on my code [with the contextualized explanation]"* (S6); they were *"able to follow up with [the instructors] with the chat feature"* (S7); *"instructors could add comments to my code to help me"* (S3). These results suggest that instructors can use EdCode to not only provide high-quality remote assistance without some of the limitations of in-person office hours, but also simplify the interaction by splitting the controls and letting them focus more on the feedback than the interaction with code on another machine.

Instructors' Perception of Code Publicity: In the survey, one instructor reported that *"I appreciated having the ability to write out the code itself in-line, reference it in my answer by linking it to the word 'here,' and comment some additional notes in-line"* (I2), showing the versatility of the tool, and that whether or not instructors choose to capture the code context as a part of the explanation *"depend[s] on the quetions"* (I1). One instructor (I1) proposed a feature that would suggest similar questions that were previously answered, anticipating that the code publicity feature could be helpful in reducing duplicate questions in the first place. The same instructor (I1) commented on the issue of plagiarism on current discussion forums: *"[the code] should definitely be controlled because a lot of people post their full answers and said like 'is this correct?"'* This instructor (I1) found the code publicity feature useful, *"it removes the problem of other students being able to see a student's full code and copying their answers, which is a big problem with the Facebook Group."*

Chat Tool: All requests received at least one response; 27 were answered with a textual explanation, and three were answered only through chat messages. In eight cases, participants used at least one round of back-and-forth interaction using the chat tool, either in real time or leaving messages for asynchronous interaction. The messages left in the chat tool were often short rather than self-contained, which invite quick clarifications and light-weight conversation. Because the chat tool was situated within a request, an instructor can focus on one request. This is in contrast with in-person communication; instructors in office hours may feel pressured due to the student waiting, and they may even be interrupted. We anticipate that the instructor's attention can be more focused to those who are available at the moment and thus provide more assistance (as opposed to first-come-first-served basis).

B. Study 2: Contextualized Response and Code Publicity

We further evaluate EdCode's response quality and its Code Publicity feature. To compare responses, we recruited one instructor from the same class and asked him to write answers to six questions with three formats: (1) plain text (PT) (2) contextualized explanation (CE) with EdCode's hypertext feature; (3) write contextualized explanation first and then use the code publicity (CP) feature to censor any code that was irrelevant or that other students should not see. We recruited four different students to first rate the the comprehensibility of answers in the PT, CE format on a scale of 1 to 9, and then review both the questions and answers for the CP format.

Results and Discussion: The answer comprehensibility ratings for contextualized explanation with code (CE) (μ=7.4, s.d.=1.82) and plain-text explanations (PT) (μ=8.0, s.d.=1.58) were comparable. Students reported that the contextualized responses (CE) allowed them to easily see *"what my code should look like"* (S9). When students were asked to give integration strategies, they were able to provide the correct line number and the location to insert the given code snippets for responses in both CE and PT format. All students were able to comprehend both the questions and answers in the CP format where only the portion of the code was provided. When asked to explain the given questions and answers, students had the option to provide their own answer to the question. We annotated their answers and found alignment with the instructors', showing that even with some context censored, the questions were still fully understandable by students.

VI. Conclusion

In this paper, we explored ways to scale personalized support for students in introductory programming classes. We designed and studied EdCode, a system that allows students to seek remote instructional support within their IDE in a way that resembles in-person support. Through our user study, we find that the support provided by EdCode can be on par with traditional office hours in terms of the comprehensibility of answers, while also allowing for much more flexibility in communication formats (asynchronous and synchronous, with-in context Q&A). Additionally, it facilitates scaling the personalized support without risking the learning opportunities coming from potential plagiarism that exists in traditional Q&A forums. Follow-up questions using the chat tool also allowed for more immediate feedback and interaction than these forums. Our findings provide guidance for designing help-seeking tools in other subjects to scale high quality support for students.

VII. Acknowledgements

We thank our reviewers for their helpful suggestions on this work, and our study participants for their time.

References

[1] B. of Labor Statistics. U.S., "Occupational outlook handbook, 2014-15 edition, software developers. retrieved december 10, 2016," 2016. [Online]. Available: http://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm

[2] P. K. Chilana, C. Alcock, S. Dembla, A. Ho, A. Hurst, B. Armstrong, and P. J. Guo, "Perceptions of non-cs majors in intro programming: The rise of the conversational programmer," in *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 251–259.

[3] B. S. Bloom, "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring," *Educational researcher*, vol. 13, no. 6, pp. 4–16, 1984.

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

[5] S. A. Ambrose, M. W. Bridges, M. DiPietro, M. C. Lovett, and M. K. Norman, *How learning works: Seven research-based principles for smart teaching*. John Wiley & Sons, 2010.

[6] L. Breslow, D. E. Pritchard, J. DeBoer, G. S. Stump, A. D. Ho, and D. T. Seaton, "Studying learning in the worldwide classroom: Research into edx's first mooc," *Research & Practice in Assessment*, vol. 8, 2013.

[7] J. Manning and M. Sanders, "How widely used are mooc forums? a first look," *Signals: Thoughts on Online Learning*, 2013.

[8] J. Huang, A. Dasgupta, A. Ghosh, J. Manning, and M. Sanders, "Superposter behavior in mooc forums," in *Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 2014, pp. 117–126.

[9] R. McGuire, "Building a sense of community in moocs," *Campus Technology*, vol. 26, no. 12, pp. 31–33, 2013.

[10] M. J. Thomas, "Learning within incoherent structures: The space of online discussion forums," *Journal of Computer Assisted Learning*, vol. 18, no. 3, pp. 351–366, 2002.

[11] A. Pincas, "Successful online course design: Virtual frameworks for discourse construction," *Educational Technology & Society*, vol. 1, no. 1, p. 15, 1998.

[12] Y. Chen, S. W. Lee, Y. Xie, Y. Yang, W. S. Lasecki, and S. Oney, "Codeon: On-demand software development assistance," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017.

[13] J. Zhu, J. Warner, M. Gordon, J. White, R. Zanelatto, and P. J. Guo, "Toward a domain-specific visual discussion forum for learning computer programming: An empirical study of a popular mooc forum," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2015, pp. 101–109.

[14] R. Slag, M. de Waard, and A. Bacchelli, "One-day flies on stackoverflow-why the vast majority of stackoverflow users only posts once," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 458–461.

[15] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest q&a site in the west," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2011, pp. 2857–2866.

[16] X. Liu and H. Zhong, "Mining stackoverflow for program repair," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2018, pp. 118–129.

[17] A. Head, C. Appachu, M. A. Hearst, and B. Hartmann, "Tutorons: Generating context-relevant, on-demand explanations and demonstrations of online code," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2015, pp. 3–12.

[18] J. R. Anderson and E. Skwarecki, "The automated tutoring of introductory computer programming," *Communications of the ACM*, vol. 29, no. 9, pp. 842–849, 1986.

[19] K. Rivers and K. R. Koedinger, "Data-driven hint generation in vast solution spaces: a self-improving python programming tutor," *International Journal of Artificial Intelligence in Education*, vol. 27, no. 1, pp. 37–64, 2017.

[20] B. P. Woolf, *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, 2010.

[21] P. J. Guo, "Codeopticon: Real-time, one-to-many human tutoring for computer programming," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 2015, pp. 599–608.

[22] T. D. LaToza, W. B. Towne, C. M. Adriano, and A. van der Hoek, "Microtask programming: Building software with a crowd," in *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 2014, pp. 43–54.

[23] Y. Chen, S. Oney, and W. Lasecki, "Towards providing on-demand expert support for software developers," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2016.

[24] S. Oney, C. Brooks, and P. Resnick, "Creating guided code explanations with chat. codes," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–20, 2018.

[25] A. Y. Wang, Z. Wu, C. Brooks, and S. Oney, "Callisto: Capturing the "why" by connecting conversations with computational narratives," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20. ACM, 2020.

[26] C. Gutwin and S. Greenberg, "A descriptive framework of workspace awareness for real-time groupware," *Computer Supported Cooperative Work (CSCW)*, vol. 11, no. 3-4, pp. 411–446, 2002.