On-Demand Collaboration in Programming

YAN CHEN, University of Michigan, Ann Arbor
JASMINE JONES, Berea College, Berea
STEVE ONEY, University of Michigan, Ann Arbor

Many teams have shifted to online remote collaboration as a result of COVID-19, from professional development teams to programming classes to computing-related workspaces (e.g., makerspaces). This paper explores on-demand remote help seeking in programming, a type of collaboration that occurs when developers seek online support for their tasks as needed, and argues that a key challenge in scaling remote on-demand collaboration in programming is to facilitate effective context capturing and workforce coordination. Traditionally, this collaboration happens within teams and organizations where people are familiar with the context of the tasks. Recently, this collaboration has become ubiquitous due to the success of paid online crowdsourcing marketplaces (e.g., Upwork) and Q&A sites (e.g., Stack Overflow). We discuss prior work on on-demand collaboration in programming, analyze how the idea can be tested in physical computing as well, and examine existing and new challenges that should be further explored.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: on-demand support; programming collaboration; crowdsourcing

ACM Reference Format:
Yan Chen, Jasmine Jones, and Steve Oney. 2018. On-Demand Collaboration in Programming. In *New Future of Work '20: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Technology companies, schools, bootcamps, and makerspaces have shifted to prolonged online work as a result of COVID-19. This change in working environment directly affects one of the most common human behaviors at work—asking for support from others during program development. With programmers working from a relaxed environment at home, help-seeking behavior has shifted from synchronous and in-person to asynchronous and remote. Although more flexible in terms of location and time, overhead coordination costs and ineffective communication (e.g., insufficient context) often delay the interaction, reducing overall productivity. To address these issues, this position statement draws lessons from prior work in the field of communication and collaboration in programming, analyzes the effective practices in the relevant sub-fields, and discusses ways we can integrate the benefits into current work environments.

We envision a new collaboration mode in which developers can easily make comprehensive request on-demand that include appropriate context, and remote helpers can provide appropriate and high-quality assistance quickly. This will enable more personalized question-answering, or task hand-off type of assistance and more efficient allocation of expertise than current techniques. In addition, this paper lists potential research gaps in the field of on-demand programming assistance that might be worth exploring to help address some of the current global work conditions.

Manuscript submitted to ACM

1.1 Related Work

We argue that a key challenge in scaling remote on-demand collaboration in programming is to facilitate effective context capturing and workforce coordination. In this section, we review prior work in programming team communication and programming assistance in embedded system development to support this argument.

1.2 Programming Team Communication

Despite their reputation for preferring solitude, software developers who work in teams can spend up to half of their time communicating [10]. Team organization and communication mediums play an important role in the effectiveness of communication between developers. Our vision is related to Harlon Mills' idea of a "surgical team" development model where the developer with the most experience with a particular task delegates more mundane tasks to other developers [2, 17]. Prior work done by the first author has studied a task delegation model for developers and found multiple needs and challenges that developers face when using existing help-seeking tools [4]. He then developed a system to instantiate this model, enabling more effective task hand-offs between software developers and remote helpers, which reduced the coordination costs compared to existing methods (e.g., one-on-one help sessions) [3]. While promising, this is only a first step and **more work needs to be done in the space of task delegation during programming with regard to concepts such as team trust, communication preferences, and context sharing.**

1.2.1 Pair Programming. Another common form of collaboration at work is pair programming, a method in which two people work together side by side on one computer [6]. Pair programming has been shown to yield better design, more compact code, and fewer defects for roughly equivalent person-hours as solo programming [23], but it requires the collaborators to be available for long sessions at the same place, even when minimally needed, which is inefficient. Distributed pair programming is a derived version of pair programming that uses a dedicated IDE to allow every participant to edit the same code locally [1, 19]. Although the distributed pair programming approach removes the issues of distance work and dispersed teams [18], **best practices for coordinating work and maintaining both old and new participants' understanding of the task context still remain largely unexplored.**

1.2.2 Team Information Needs. A number of researchers have also categorized the types of questions programmers ask in different contexts. Sillito et al. described 44 types of questions programmers ask when evolving a large code base [20]. Ko et al. categorized six types of learning barriers beginners face when programming systems, proposed possible solutions for programming system related issues [15], and documented communication amongst co-located development teams [14]. Guzzi et al. analyzed IDE support for collaboration and evaluated an IDE extension to improve team communication [9]. These studies of team information needs focused on existing team structures when groups are in a shared location. We believe that **new challenges evolve when the collaboration shifts to remote work [5].**

1.2.3 Collaborative Development. Systems like Codeopticon [7] and Codechella [8] provide methods that allow helpers (i.e., tutors/peers) to efficiently monitor multiple learners' behavior and provide proactive, on-demand support. Commercial IDE tools such as Koding [21], Codenvy [11], and Cloud9 [12] enable users to code collaboratively online in real time. Although these systems reduce many of the barriers developers face when working at a distance [18] and time spent on environment configuration, **they do not support cases in which developers actively seek help [22].**

Altogether, these prior studies suggest that more work needs to be done in the space of remote collaboration in programming in order to lower the cost of supporting context capture and presentation, notifying team members with the right expertise, and easing the integration of responses into developers' working artifacts.

1.3 Programming Assistance in Embedded System Development

The rise of the Maker movement has led to a growing number of developers who prototype and program embedded systems (e.g., Arduino). However, with makerspaces and laboratories temporarily shutting down, developers must collaborate remotely to work on their projects, which can be problematic. Specifically, there are two challenges. **First**, capturing the context of physical devices can be daunting. Typically, development history such as prior wire connections and hardware states is not automatically captured. This makes communication inefficient, as helpers might ask requesters to try previously attempted solutions or request historical information (e.g., prior wire connections). **Second**, remote helpers lack access to inspect and manipulate developers' physical devices, which limits them to providing only suggestions or guidance rather than contributing via physical changes made to the devices. Thus, in spite of the collaboration, only end-user developers have the ability to carry out the planned tasks.

To address these challenges, existing tools such as Bifröst allow developers to more efficiently debug embedded systems [16]; likewise, Heimdall allows instructors to remotely inspect, measure, and rewire students' circuits [13]. However, both tools have drawbacks: use of Bifröst is limited to specific tasks, and developers must still rely on their own capacity to complete their tasks. Heimdall requires students to build their circuits on a specialized workbench, limiting the size of students' projects and prohibiting them from working in parallel when seeking help [13]. Therefore, this prior work leads to the following research question: **How can we facilitate effective remote on-demand help seeking for embedded system programmers to receive personalized expert assistance?**

2 CONCLUSION AND BROADER IMPACTS

This position statement envisions a new collaboration mode in which developers can easily make comprehensive request on-demand that include appropriate context, and remote helpers can provide appropriate and high-quality assistance quickly. This will enable more personalized question-answering, or task hand-off type of assistance and more efficient allocation of expertise than current techniques. It also advocates facilitating effective context capturing and workforce coordination to help scale the current remote work situation. By working towards this vision, we will better understand what questions developers need answered remotely, how they ask them, and how the existing question-answering systems can support their requests. It will enable new types of programming collaboration and teamwork by using a combination of human and machine intelligence to complete programming tasks. The resulting design implications could provide guidance for future system designers to build better support tools that allow software and embedded system developers to receive on-demand assistance. The resulting system will be, for example, the first to enable developers to hand off request-based physical computing tasks on demand without relinquishing control of the devices.

More broadly, the produced tools and techniques will make developers more efficient and thus more productive overall. The resulting techniques can also help expand participation in physical computing design and development in educational and collaborative work as well as DIY settings. Since shifting to an online setting, for example, programming courses across the country have had to adjust the difficulty level of the coursework or even reduce the number of collaborative projects to allow for remote study. Our work envisions a future in which students could more easily collaborate and instructors could more easily track and facilitate their progress through resulting tools. With more efficient remote support, instructors can host virtual office hours instead of in-person hours and more easily understand comprehension gaps by viewing learners' past attempts, thus reducing the interaction delay. Additionally, the produced

tools could enable more collaboration, from peer feedback to team projects, than digital-only communication techniques between learners, thus helping to increase learner engagement.

ACKNOWLEDGMENTS

We thank Björn Hartmann for his feedback on this work.

REFERENCES

[1] Prashant Baheti, Edward Gehringer, and David Stotts. 2002. Exploring the efficacy of distributed pair programming. In *Extreme Programming and Agile Methods—XP/Agile Universe 2002*. Springer, 208–220.

[2] Frederick P Brooks. 1975. *The mythical man-month*. Vol. 1995. Addison-Wesley Reading, MA.

[3] Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S Lasecki, and Steve Oney. 2017. Codeon: On-demand software development assistance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 6220–6231.

[4] Yan Chen, Steve Oney, and Walter S Lasecki. 2016. Towards providing on-demand expert support for software developers. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. ACM, 3192–3203.

[5] Yan Chen, Maulishree Pandey, Jean Y Song, Walter S Lasecki, and Steve Oney. 2020. Improving Crowd-Supported GUI Testing with Structural Guidance. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[6] Alistair Cockburn and Laurie Williams. 2000. The costs and benefits of pair programming. *Extreme programming examined* (2000), 223–247.

[7] Philip J Guo. 2015. Codeopticon: Real-Time, One-To-Many Human Tutoring for Computer Programming. In *Proceedings of the 28th annual ACM symposium on User interface software and technology*. ACM.

[8] Philip J Guo, Jeffery White, and Renan Zanelatto. 2015. Codechella: Multi-User Program Visualizations for Real-Time Tutoring and Collaborative Learning. In *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on*. IEEE.

[9] Anja Guzzi, Alberto Bacchelli, Yann Riche, and Arie van Deursen. 2015. Supporting Developers' Coordination in the IDE. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 518–532.

[10] James D Herbsleb, Helen Klein, Gary M Olson, Hans Brunner, Judith S Olson, and Joe Harding. 1995. Object-oriented analysis and design in software project teams. *Human–Computer Interaction* 10, 2-3 (1995), 249–292.

[11] Codenvy Inc. 2012. *Codenvy*. https://codenvy.com

[12] Cloud9 IDE Inc. 2010. *Cloud9 IDE*. https://c9.io

[13] Mitchell Karchemsky, JD Zamfirescu-Pereira, Kuan-Ju Wu, François Guimbretière, and Bjoern Hartmann. 2019. Heimdall: A Remotely Controlled Inspection Workbench For Debugging Microcontroller Projects. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[14] Amy J Ko, Robert DeLine, and Gina Venolia. 2007. Information needs in collocated software development teams. In *Proceedings of the 29th international conference on Software Engineering*. IEEE Computer Society, 344–353.

[15] Amy J Ko, Brad Myers, Htet Htet Aung, et al. 2004. Six learning barriers in end-user programming systems. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*. IEEE, 199–206.

[16] Will McGrath, Daniel Drew, Jeremy Warner, Majeed Kazemitabaar, Mitchell Karchemsky, David Mellis, and Björn Hartmann. 2017. Bifröst: Visualizing and checking behavior of embedded systems across hardware and software. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 299–310.

[17] Harlan D Mills. 1980. Software engineering education. *Proc. IEEE* 68, 9 (1980), 1158–1162.

[18] Gary M. Olson and Judith S. Olson. 2000. Distance Matters. *Human-computer interaction* 15, 2 (2000), 139–178.

[19] Julia Schenk, Lutz Prechelt, and Stephan Salinger. 2014. Distributed-Pair Programming can work well and is not just Distributed Pair-Programming. In *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 74–83.

[20] Jonathan Sillito, Gail C Murphy, and Kris De Volder. 2008. Asking and answering questions during a programming change task. *Software Engineering, IEEE Transactions on* 34, 4 (2008), 434–451.

[21] Devrim Yasar Sinan Yasar. 2012. *Koding*. https://koding.com

[22] Igor Steinmacher, Marco Aurélio Graciotto Silva, and Marco Aurélio Gerosa. 2014. Barriers faced by newcomers to open source projects: a systematic review. In *Open Source Software: Mobile Open Source Technologies*. Springer, 153–163.

[23] Laurie Williams and Robert Kessler. 2002. *Pair programming illuminated*. Addison-Wesley Longman Publishing Co., Inc.