Online Community Members as Mentors for Novice Programmers Position Statement

Michelle Ichinco and Caitlin Kelleher
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, USA
{michelle.ichinco, ckelleher}@wustl.edu

***Abstract*— While online communities exist surrounding blocks programming environments, they do not support the type of feedback a novice programmer might receive in a classroom setting. We propose a feedback system in which experienced programmers author feedback and programmatic rules to distribute feedback to novice programmers. Finally, we outline three main considerations for designing systems for online community feedback for novice programmers: community members, the format of the feedback, and how the feedback is distributed at a large scale.**

Keywords—novice programming; online communities; block programming

I. INTRODUCTION

Current online communities for block programming languages allow users to share their content, provide basic feedback such as "likes" and comments, and reuse programs. Many programmers involved in these communities learn to program outside of a classroom, so they only receive these basic forms of feedback on their programs. However, feedback is critical to improving a new skill, especially programming. Furthermore, in some online communities, content rarely even receives likes or comments, yet feedback is directly linked to participation. Feedback, especially when it is specific and personalized, can be critical in boosting motivation and learning for novices.

We believe that online communities for block programming languages can be leveraged to provide more specific and useful feedback to programmers. One such system involves experienced programmers who view content contributed by friends or family members. These experienced programmers may have valuable feedback that they can provide to a novice programmer they know, which can also extend to other novice programmers. This work describes a system for leveraging experienced programmers as mentors for novices. We then present open questions about the possible types of community members, types of feedback, and ways to scale online community feedback.

II. EXISTING ONLINE COMMUNITIES

A variety of current online communities exist to support users of blocks based programming languages. However, none support detailed feedback aimed at motivating novices or helping them improve their skills. For example, Scratch has a community of users who share projects, collaborate, comment, and reuse [1]. Looking Glass also has an online community where users can share, reuse, and comment on projects [2]. Similarly, TouchDevelop [3] has a showcase where users can post their projects and comment on other users' projects. Kodu [4] and App Inventor [5] have forums where users can ask questions or discuss a variety of topics. However, these communities do not provide support for members to give in-depth feedback. In order to harness the power of online communities for blocks programming environments, we need community members to be able to act as mentors for novice programmers and also ensure that feedback can be extended to a large population of novice programmers.

III. EXPERIENCED PROGRAMMER MENTORS

We propose one way to utilize an online community: to have experienced programmers act as "mentors", contributing feedback for novices. Some novice programmers will likely have a friend or family member who has programming experience and motivation to help the novice improve. If we can leverage the feedback of a small number of mentors from an online community by having them author rule scripts that distribute the feedback to other novices, we can provide personalized valuable feedback to novice programmers at a large scale. In this system, experienced programmers have three main roles: A) feedback creation, B) rule authoring, and C) feedback and rule vetting.

A. Feedback creation

We propose that experienced programmers can identify issues in programs and improve novice code with more complex concepts. An exploratory study showed that experienced programmers often made valuable technical suggestions, as well as suggestions based on program content [6]. We hypothesized that a system could present example code for a novice programmer to use in improving their program, but many challenges still exist in novice programmer use of examples [7]. It is also possible that better forms of feedback exist that could be more motivating or effective than example code. Regardless, the feedback should not show the novice programmer exactly what they need to do, but should demonstrate skills that the novice programmers has not yet mastered.

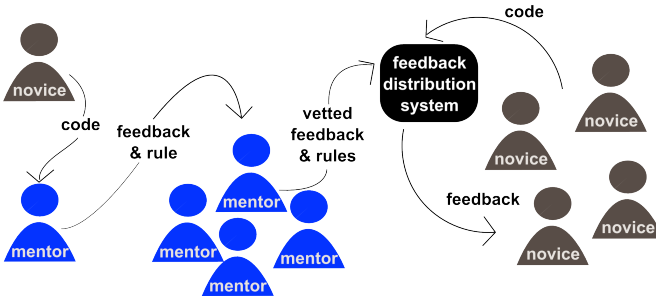

Fig 1 Crowdsourced feedback system

Once the mentor improves a part of a program, the system can then present feedback to novices to improve their code the same way, or to learn that skill. The system can determine which novices should receive which feedback using rule scripts authored by the mentors.

B. Rule Authoring

Many experienced programmers can author a short script, or a rule, that can check whether a novice program contains a certain issue. We ran an exploratory study that showed that experienced programmers could often write a pseudocode rule to check a program, like the example shown in Fig. 2 [6]. However, early work on the design of a system for rule authoring showed that only highly skilled programmers can actually author working rules. One reason for this might be that experienced programmers do not often work with block programming languages, so even with API support, authoring a rule to check the contents of a block program was challenging. However, another issue may be varying "experience" levels in the community of mentors. Some programmers noticed right away by reading the instructions for rule authoring that it was beyond their skills. Others attempted to author rules, but they were not all successful. Due to the differing skill levels and possible quality issues related to crowd work, this system needs a way to ensure that feedback and rules actually provide valuable information to novice programmers.

C. Feedback and Rule Vetting

We propose that having experienced programmers create feedback and then vet or improve each others' feedback can produce high quality feedback. While experienced programmers likely are not trained in creating educational content, we hypothesize that a community of mentors can create valuable educational material through an iterative process of improving and filtering the feedback. Furthermore, some experienced programmers who do not necessarily have the skills to author rules or even create feedback may have enough knowledge to determine the quality of the feedback. Thus, this system can leverage a variety of different mentor skill levels to produce valuable feedback for novices at a large scale.

```
for each doTogether in program:
        if doTogether.size()<2:
                return True
```

Fig 2 An example of a pseudocode rule. In this rule, a program receives feedback if it has a parallel execution block with only one statement inside, since that might indicate a misunderstanding of the parallel execution concept.

IV. Design Features of Online Community Feedback

Based on the findings and issues with having experienced programmers provide feedback to novices, three important questions are: which community members can and will provide valuable feedback, what format is best for providing valuable and motivating feedback, and what is the most accurate and efficient way to provide feedback at a large scale?

A. Community Mentors

There are a variety of different types of online community members who could fulfill the role of a mentor, such as experienced programmers and novice programmers who have started to gain skills.

1) Experienced programmers

Experienced programmers will likely program as a part of their career and have a friend or family member using a blocks based programming language. Their motivation would stem from their relationship with a specific novice programmer who they would like to help. The benefit of involving these experienced programmers is they may be motivated to continue providing feedback as long as the novice programmer they know continues programming. We have also found that some experienced programmers generally want to "give back", by sharing their programming skills. However, these "experienced" programmers may have a variety of different skill levels and often do not have training in education. Thus, having experienced programmers provide feedback requires that the feedback be checked for quality and generalizability.

2) Novice programmers with skills

"Novice programmers" who have learned certain skills may also be able to provide feedback to newer programmers. These experienced novice programmers have the potential to generate highly motivating content, as they have more experience with the blocks based programming language and features. They also might generate feedback that is closer to the level of the novice programmer, since they just learned those skills. Like the experienced programmer feedback, this feedback would likely also need vetting.

B. Feedback format

The feedback novice programmers receive should motivate them to continue programming, in addition to helping them improve their programming skills. In the context of blocks based programming, users often choose to program a specific project and may not want to be interrupted with suggestions. Instead, the feedback could be an alternative programming activity that grabs their attention and gives them new ideas and direction. Ideally, the feedback novice programmers receive can both motivate them and educate them, by presenting new skills that novices can use to create exciting new content. Two possible ways to do this are to provide extra motivation to use feedback, such as awards for completing certain tasks [8], or by presenting feedback in a highly motivating way, such as a game or puzzle. Likely, other motivating and effective feedback forms also exist.

C. Feedback distribution

Since it is unlikely that online community members can generate feedback for each program created, we need a way to distribute feedback to a large number of programmers who could also benefit from it, such as by programmatic rules, machine learning, or a curriculum. However, mentor rules may not always correctly generalize an issue, while machine learning can also misclassify a programmer's skill level and knowledge gaps. With a curriculum, the system would also need to check where a programmer is within the curriculum, and it may not be personalized to the programmer's needs. Regardless of the method, we also need to be able to capture whether the distribution of feedback accurately captures the needs of the programmers.

V. Conclusion

In this position paper, we argue that online community members can help to provide detailed feedback to novice programmers at a large scale. We have shown that experienced programmers have the capabilities to fulfill this role, by creating feedback and authoring rules to distribute the feedback. However, another possible direction is to have novice programmers create feedback for others. We believe there is still more work to be done to discover the best way to present feedback to novices in blocks programming environments, as well as in how to best distribute feedback at a large scale.

VI. References

[1] "Scratch | Home | imagine, program, share." [Online]. Available: http://scratch.mit.edu/. [Accessed: 19-Sep-2012].

[2] "Looking Glass Community." [Online]. Available: https://lookingglass.wustl.edu/. [Accessed: 24-Feb-2013].

[3] "TouchDevelop - create apps everywhere, on all your devices!," *TouchDevelop*. [Online]. Available: https://www.touchdevelop.com/. [Accessed: 21-Jul-2015].

[4] "Kodu | Home." [Online]. Available: http://www.kodugamelab.com/. [Accessed: 21-Jul-2015].

[5] "MIT App Inventor | Explore MIT App Inventor." [Online]. Available: http://appinventor.mit.edu/explore/front.html. [Accessed: 21-Jul-2015].

[6] M. Ichinco, A. Zemach, and C. Kelleher, "Towards generalizing expert programmers' suggestions for novice programmers," in *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*, 2013, pp. 143–150.

[7] M. Ichinco and C. Kelleher, "Exploring Novice Programmer Example Use," *IEEE Symp. Vis. Lang. Hum.-Centric Comput.*, To Appear 2015.

[8] A. Zemach, "The Effects of Gamifying Optional Lessons on Motivation," Master's Thesis, Washington University in St. Louis, 2014.