Developing an Institutional Peer Code Review Tool using Software Engineering Principles

1st Tamaike Brown
dept. of Computer Science
State University of New York at Oswego
Oswego, USA
tamaike.brown@oswego.edu

2nd Bastian Tenbergen
dept. of Computer Science
State University of New York at Oswego
Oswego, USA
bastian.tenbergen@oswego.edu

***Abstract*—This work in progress paper reports the software engineering activities involved in designing and implementing an institutional peer code review (IPCR) tool. The platform aims to help novice computer science (CS1) students at a particular institution understand Java introductory programming concepts and improve their programming skills while learning from instructors and peers. Over 200 students enrolled in CS1 programming course experience difficulty understanding basic concept such as loops, methods, and classes. Consequently, students may be forced to repeat CS1 courses or, discouraged from continuing their major resulting in a discontinuation of their CS education. Prior studies show that peer code review(PCR) can help address this problem as a collaborative learning approach. However, the methods used in those studies were not done using a tool that was built to help students with little or no programming experience. The goal of this research is therefore to develop an IPCR tool using software engineering principles that facilitates instructor-to-student and peer-to-peer learning. Students shall be able to access the platform 24/7, submit java code and receive feedback from instructors or peers. This will not only help students understand programming principles and identify common novice programmers' mistakes but will also improve students understanding and programming skills. We report on the progress of our tool.**

I. Introduction

Students enrolled in introductory programming courses for undergraduate CS programs often struggle with understanding basic programming concepts [1] related to, algorithms, iteration, and other concepts traditionally attributed to a "CS1"-type course [2]. Reasons for this lie in students not investing adequate time [3] in understanding programming concepts and the quality of the code they produce since they have little or no prior experience with computational thinking [4]. As a result, they often make programming mistakes that are recurring and common to all students. To assist CS1 students the researchers focus on a student-centered pedagogy method. An example of student-centered pedagogy is to apply peer learning [5]. For CS education, PCR in particular is an effective teaching approach in CS1, CS2, and software engineering (SE) classrooms. PCR have proven to enhance students engagement, improving code quality, and—by extension—students' learning of CS and SE concepts [6]–[8]. PCR involves reading programs develop by others to identify mistakes [9], [10]. Past PCR studies were limited to in-person classroom experiences. Novice programming students require a teaching approach that facilitates deep learning. The goal of this research is therefore to develop an IPCR tool using software engineering principles that facilities instructor-to-students and peer-to-peer learning. Students shall be able to access the platform 24/7 and have their questions answers by instructors or peers, thereby engaging in student-centred peer learning. This will not only help students understand programming principles and identify common novice programmers mistakes, but will also improve students programming skills.

In this paper, we report on initial design results in developing such tool. The particular novelty in this work is that the tool is designed for novice programmers, supports student-to-students feedback, instructor-to-students feedback, points allocation and likes.

The contributions of this research are: 1. To gain insights into the specific requirements for a student-centered IPCR tool; 2. Share the stages involved in developing an IPCR platform where novice students learn to solve programming tasks in a virtual environment; 3. Draw conclusions from these prototypical steps regarding what a IPCR tool needs to accomplish; 4. Draw conclusions regarding non-functional requirement aimed towards increasing learner acceptance of the IPCR tool.

This paper is organized as follows: Section II reviews the related work on peer code reviews and peer learning. Section III briefly overviews the developed prototype. Section IV draws conclusions and general requirements from the IPCR tool prototype. Section V concludes the paper.

II. Background

Code review is one of the first tasks college graduate programmers are expected to do in the software industry. The ability to read and understand programs written by others helps one to understand the program design, address problem solving issues for program maintainability and improve ones own programming skills as a result of knowledge transfer [9], [10]. In 1976 Fagan first formalized a concept of PCR [11], since then PCR is utilized differently in the works of many researchers as an educational tool with positive learning benefits [6]–[8], [12], [13], [13]–[17].

TABLE I
IPCR TOOLS VS OTHER PEER CODE TOOLS:

Comparison	
Tools	Comments
IPCR	automatic points allocation for student participation for gamification, like button installed for gamification, easy to use for novice programmers, student-to-student feedback, instructor-to-student feedback, java programming language friendly, knowledge based for future students, student work is stored on the University owned resources for record keeping, no cost to students.
Github / GitLab	Java friendly, better suited for intermediate to expert programmers.
BitBucket	For professional teams.
Klockwork	Built for enterprise DevOps and DevSecOps.

See Table I for a comparison of the IPCR tool to other Peer Code tools that are most comparable to the IPCR tool. This list exclude tools used in creating and sharing web application.

While different variations of a peer code review have been used by CS educators, these studies lack the usage of a student-centred institutional tool developed by students, that supports students-to-students feedback, instructor-to- student feedback learning, self-paced peer-to-peer learning, engaged critical thinking, accessibility, and an effective way of reaching students.

In this work, we aim at developing a IPCR tool that promotes instructor to student learning and peer-to-peer learning for novice programmers from all STEM discipline enrolled in the CS1 course at University of Oswego.

III. STUDENT-DESIGNED PCR PROTOTYPE

A team of five students volunteered for this project, see project description[1]. In this section, we present students solutions for developing a prototype for the PCR tool. We draw conclusions from their work in Section IV.

A. Project Specification

Herein, we describe how the team conceived the IPCR tool [2].

1) Requirements Engineering: During this phase of the software development life cycle student developers gathered initial requirements from the stakeholders and gain an overview of the project goals, relevant materials, systems, and constraints.

The team developed the following *Key Features*: 1. Offer a visual text area with Java syntax highlighting; 2. Allow users to create, update, delete code in "edit mode"; 3. Allow storing of code files; 4. Allow users to share files with other users; 5. Allow users to open files that have been shared in a "feedback mode"; 6. Provide feedback by posting comments on existing code

2) System Architecture: Students devised an high-level ontological relationship architecture as shown in Figure 1. Subsequently, the team refined this high-level depiction into a database schema and middleware achitecture, as shown in Figure 2 and 3. The main components of the system shown in Figure 3 are a "PostHandler", "LanguageFilter", and a database component.

3) System Behavior and User Interface: The key system functions include (1) create, edit, and remove comments as well as posts, (2) flag posts for review by instructors, and (3) filter course access by instructor-provided access list.

B. Construction & Quality Assurance

In this section, we briefly overview the construction process, with particular focus on the team's choice of tools and technologies used in the development of the IPCR tool.

Technology choices for the IPCR tool include: MongoDB as the Database service; Spring Boot web framework to serve the Java middleware for post management and commenting, and; React JavaScript for front end implementation.

Auxiliary libraries to facilitate functionality include: Axios, an HTTP request library for making backend API calls; CSV-toJSON , a library for converting CSV files to JSON arrays used for class roster management; React-Syntax-Highlighter – an open-source library for colorizing code and highlighting programming language specific keywords; React-Google-Login – an open-source library for rendering a Google popup window for users to authenticate Google credentials; GSON, Google's open-source library for handling JSON; Firebase, used for managing Google authentication; Material-UI, a React library that provides styling for React components

C. Deployment and Completion

The IPCR tool was deployed to Amazon WS as well as Herokuapp.

IV. IPCR TOOL REQUIREMENTS & EVALUATION

In this section, we draw conclusions based on the students implementation regarding what programming novices expect from a peer code review platform. We review the key concerns and main requirements students documented in their specification document (see Section III for a link) as shown in Table II.

Active and deep learning is promoted by making a post and passive learning is promoted by browsing and reading through peers contributions (see Figures 4 and 5). The development team suggested adding a functionality to allow students to read others posts after having contributed their own solution, e.g., to an assignment.

the IPCR tool should be tailored to the course students are enrolled in by the instructor. This includes that the IPCR tool must be easy to deploy and access for students, e.g.,

[1]Available at: https://bit.ly/3wbcngx

[2]The complete specification can be found at: https://drive.google.com/file/d/1-1QslkaIbkr3Ggdg124IDl0qF6-16gTB/view?usp=sharing

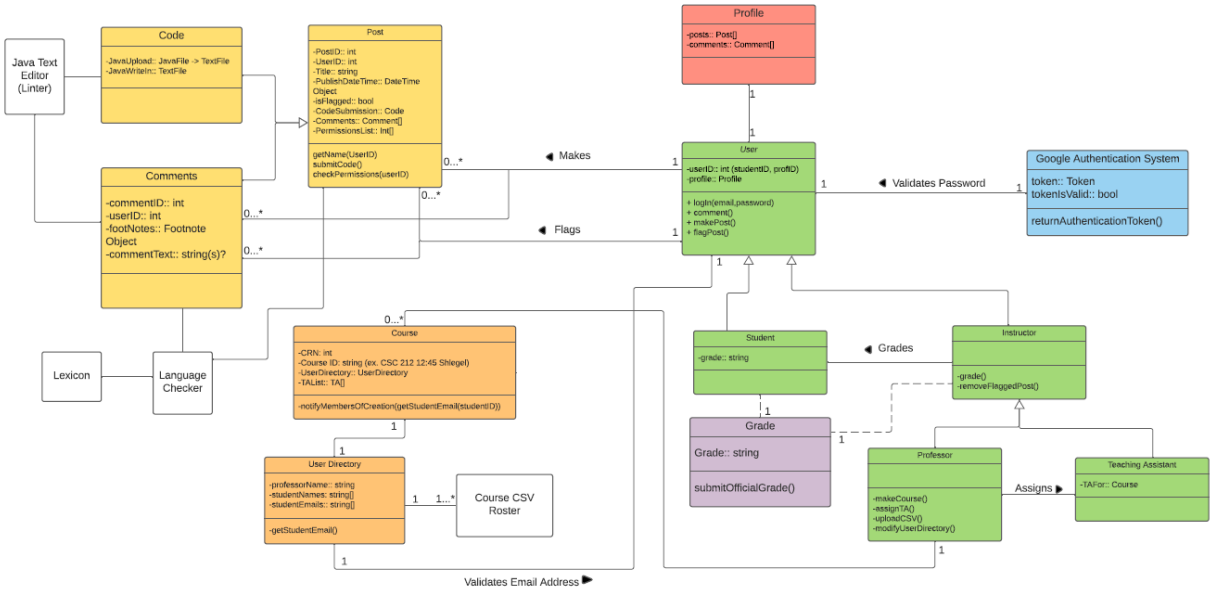

Fig. 1. Class Diagram showing Ontological Relationships of the Problem Domain

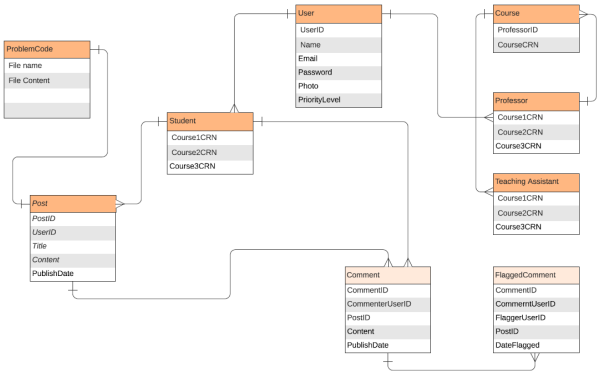

Fig. 2. Database Schema of the PCR Tool

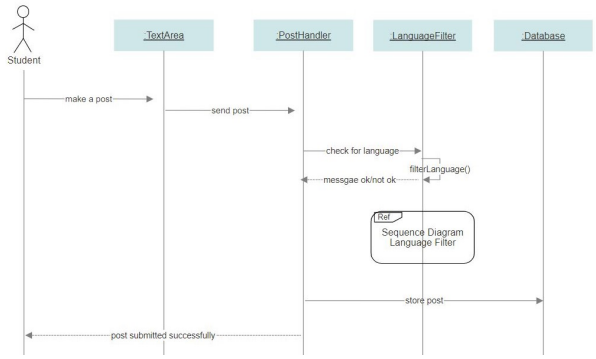

Fig. 3. PCR Tool Components and Interactions

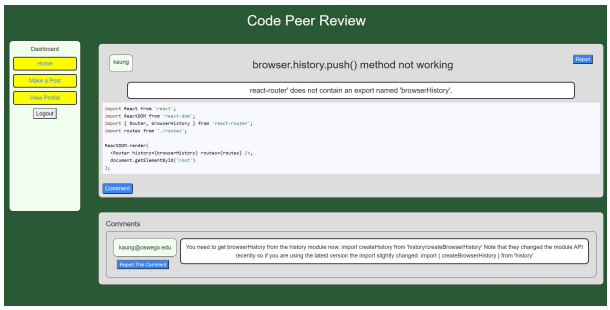

Fig. 4. User Interface Example

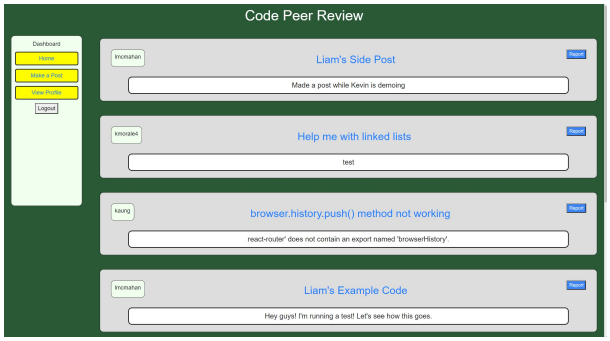

Fig. 5. Adding Post Comments to Code Submissions

through programming environment integration or a dedicated link specific to the course they are enrolled in. A key factor here is that posting a question when students struggle with a programming assignment must be as effortless as possible, minimizing the need to log into the "right" system and without having to observe, e.g., syntax rules of the programming language while posting.

V. Conclusion

In this paper, we presented findings from a learner's perspective gathered through the process of developing a pro-

TABLE II
NON-FUNCTIONAL STUDENT REQUIREMENTS FOR THE PCR TOOL.

Criteria	
Attributes	Comments
Repudiation	The IPCR tool shall maintain records of user contributions, yet without making students feel exposed for uploading poor quality code (e.g., allow restrictive post sharing, deleting or editing posts, or anonymous posting).
Reliability	Code shall be posted "as is", without enforcing syntax rules.
Usability	Posting code shall resemble comment discussions without oppressing tool-specific functionality.
Efficiency	Posting code must be effortless on the user, ideally through integration into a programming environment.
Portability	The IPCR tool must be compatible with different classes, but offer a singular user experience for the student.
Maintainability	Instructors must be able to easily set up the IPCR tool for each new course.

totype for a IPCR tool. The IPCR tool will not only teach students how to do code review, it also educated students on a particular topic while encouraging critical thinking. Requirements as developed through a software engineering course project include (1) submitting code for review by peers; (2) instructor-assigned teams to review submitted code using a formal review checklist; and (3) reviewer-provided review ratings and feedback on the usefulness of the review.

Furthermore, we investigated students' non-functional requirements to increase acceptance of the tool. We also seek avenues to make the peer code review tool more engaging by incorporating games design elements into the code review process to improve students' learning experience.

In future work, we seek to continue developing the IPCR tool and empirically evaluate the effectiveness of the IPCR platform on students learning in CS1/2 courses at different level of the curriculum.

REFERENCES

[1] H. M. Walker, "Retention of students in introductory computing courses: Preliminary plans—acm retention committee," *ACM Inroads*, vol. 8, p. 12, Oct. 2017.

[2] M. Hertz, "What do "cs1" and "cs2" mean? investigating differences in the early courses," in *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, (New York, NY, USA), p. 199–203, Association for Computing Machinery, 2010.

[3] N. R. Council, *How Students Learn: History, Mathematics, and Science in the Classroom*. Washington, DC: The National Academies Press, 2005.

[4] J. A. Qualls and L. B. Sherrell, "Why computational thinking should be integrated into the curriculum," *J. Comput. Sci. Coll.*, vol. 25, p. 66–71, May 2010.

[5] K. J. Topping, "Trends in peer learning," *Educational Psychology*, vol. 25, no. 6, pp. 631–645, 2005.

[6] T. Brown, "Guided peer code reviews in cs1/cs2 and se curricula," *IEEE Conference on Frontiers in Education*, 2019.

[7] T. Brown, G. Walia, M. Nassareddygari, and A. Radermacher, "Using guided peer code review to support learning of programming concepts in cs2 course: A pilot study," *ASEE Annual Conference & Exposition*, 2020.

[8] T. Brown, M. Nassareddygari, M. Singh, and G. Walia, "Using code review to improve programming skills of students in an introductory cs course," *IEEE Conference on Frontiers in Education*, 2019.

[9] T. Baum, K. Liskin, K. Niklas, and K. Schneider, "A faceted classification scheme for change-based industrial code review processes". software quality, reliability and security (qrs)," *IEEE International Conference on, Proceedings*, 2016.

[10] A. Kolawa and D. Huizinga, "Automated defect prevention: Best practices in software management," *Wiley-IEEE Computer Society Press*, 2007.

[11] M. Fagan, "Design and code inspections to reduce. errors in program development," *IBM Systems Journal*, vol. 15(3), 1976.

[12] D. Trytten, "A design for team peer code review," *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, 2005.

[13] W. Yan-qing, L. Yi-jun, M. Collins, and L. Pei-jie, "Process improvement of peer code review and behavior analysis of its participants," *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 2008.

[14] S. Sripada, Y. Reddy, M. Singh, and A. Sureka, "In support of peer code review and inspection in an undergraduate software engineering course," *IEEE 28th Conference on Software Engineering Education and Training (CSEET)*, 2015.

[15] E. Yourdon, *Structured Walkthroughs*. Prentice Hall, 1989.

[16] N. Clark, "Peer testing in software engineering projects," *Sixth Australasian Computing Education Conference (ACE2004), Dunedin, New Zealand, 2004*, vol. 30, 2004.

[17] A. Hundhausen, A. Agrawal, D. Fairbrother, and M. Trevisan, "Integrating pedagogical code reviews in cs1 course: An empirical investigation," *SIGSCE '09 Proceeding of the 40th ACM technical symposium on Computer Science education*, pp. 219–295, 2009.