GitCanary: A Tool for Analyzing Student Contributions in Group Programming Assignments

Jan Jaap Sandee  
Open University of the Netherlands  
Saxon University of Applied Sciences  
j.m.j.sandee@saxon.nl

Efthimia Aivaloglou  
Leiden Institute of Advanced Computer Science  
Open University of the Netherlands  
fai@ou.nl

ABSTRACT
Courses in computer science curricula often involve group programming assignments. In this paper, we present the GitCanary tool for monitoring project progress and member contributions using development productivity metrics. The tool is developed to support teachers and students with an overview that they can use during their guidance sessions. Feedback on the tool and its metrics was collected from students and teachers using interviews and questionnaires after utilizing it in a software development course. The results that the tool provides were found to be valuable in project guidance and to promote transparency in work distribution.

CCS CONCEPTS
- Social and professional topics → Computing education.

KEYWORDS
Group programming assignments, Git, Productivity, Metrics

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:

1 INTRODUCTION
Group programming assignments are a common aspect of software development courses in higher education. These projects enable students to practice and improve both their programming and their collaboration skills. However, students might not participate equally or contribute to programming aspects. Related work shows that group work is often not divided equally [2, 3], while students do not always inform their teacher of this problem. It has also been found that work equity concerns negatively impact students’ attitudes toward group work, and that the role of the teacher is important to help students view group work positively [2].

In order for teachers to monitor and guide student groups, they would have to inspect the code written by each team member. One advantage that software projects have is that they are commonly maintained using a version management system like Git, which makes student activities transparent and enables continuous monitoring of student contributions over the course of their assignments [1, 9]. Automated tools have been proposed for monitoring student collaboration. The Teamwork Dashboard [8] analyzes online team discussion transcripts to visualize team mood, role distribution and emotional climate. TeamAnalytics [5] utilizes students’ wikis and software version control system (svn) repositories, from which it extracts summaries of the number of files added, modified or deleted by each student. At the same time, in the industry and the open source software development world, various more fine-grained productivity metrics have been proposed for quantifying developer contributions using the source code repositories [4, 6].

In this paper, we present the GitCanary tool for monitoring project progress and contributions using development productivity metrics. Our aim for developing GitCanary is to support teachers and students with an overview they can use to assist in their guidance sessions. This tool was experimentally evaluated during a 1st year software development course in the spring semester of 2020. The 147 computer science students of the course were assigned to develop a game for Android in groups of an average of four students during three Scrum sprints of two weeks per sprint. Seven teachers were involved in guiding these groups and using GitCanary. At the end of the course, we explored teacher perceptions about the tool through six online interviews, and student perceptions using an anonymous questionnaire which was filled in by 93 students.

2 GITCANARY
The GitCanary tool was developed iteratively, using meetings with the course teaching team to inform its design decisions. GitCanary utilizes PyDriller [7] to analyze Git repositories and generate a JSON report. This report is stored in a Django-based REST server. A front-end system built in VueJS and ChartJS is used to display this data using charts.

GitCanary allows users to select a date range and specific branch for examining project activity. GitCanary gives an overview of the following metrics, in order of display in Figure 1:

- **Lines of Code (LOC)** Cumulative LOC added, modified, or deleted.
- **Comments** Cumulative comment lines added, modified, or deleted.
- **Documentation** Cumulative lines added, modified, or deleted in .md files in the project.
- **Git Blame** Who last modified each line at the master version.
- **Complexity** Cumulative number of methods someone introduced or modified with complexity n.

The source code of GitCanary is available at https://gitlab.com/gitcanary.
were also students who drew motivation from seeing the numbers go up and getting competitive within their team.

An additional advantage that students indicated in their open text responses was that they did not have to point out slacking members to be allocated to the other two students in the later sprints. Conversely, the Git Blame metric turned out to be not as useful because it was found hard to understand and utilize. Finally, several students indicated that the produced results across metrics are not representative when they practice pair programming or switch accounts.

4 LIMITATIONS AND FUTURE WORK

The threats to the validity of the feedback received for the tool include that it was collected from one run of a single course, which was also given remotely due to the Corona pandemic. Moreover, the feedback might have been biased due to interviewees being co-workers of the first author and the students being asked to fill out the survey during class activities.

As future work, we would like to enrich the tool with code quality metrics and to explore in more depth its application to later courses where students have more programming experience. Moreover, experiments on different setups for the use of the tool in courses could provide more insight on its effect on student behavior.

REFERENCES