Using GitHub Analytics to Assess the Quality of Collaboration in Software Engineering Teams

Quan Le, Kiet Phan, Bowen Hui, and Adara Putri
Department of Computer Science, Mathematics, Physics, and Statistics
University of British Columbia
Kelowna, Canada
minhquanle_2002@yahoo.com, phankiet.at.ubc@gmail.com, bowen.hui@ubc.ca, adarasharmilaputri@gmail.com

***Abstract*—This research-to-practice full paper investigates using team process analytics from GitHub to support team management. Effective teamwork is essential in higher education learning and workplace success. The role of educators in supporting proper team functioning includes helping students learn how to participate actively and communicate effectively in meetings, delegate work fairly, manage high-quality work throughput, and resolve conflicts if problems arise. Problems often emerge when team members have differing visions or individuals do not contribute equally to the work output. These problems are exacerbated in large classes involving many teams. Detecting these potential issues in teams and helping students work through them is important for team success. In software engineering projects, monitoring individual contributions can begin with mining activities on programming platforms such as GitHub, which makes much of the individual contributions more visible and quantifiable. In this work, we propose a framework for fairly assessing teamwork and present the development of team analytics to assist educators in detecting potential issues in team collaboration. We describe a pilot study involving this tool in the context of a software engineering capstone course with 104 students split into 22 teams managed by 4 teaching assistants. Our results show that the reports offer value in guiding the evaluation process and identifying where problems may be, but do not replace the actual repository analysis where needed. We discuss the potential value of using this tool to improve collaboration.**

***Index Terms*—Capstone assessment, collaboration assessment, collaboration analytics, GitHub metrics, GitHub assessments, pull request metrics**

I. Introduction

Effective teamwork is essential in higher education learning and workplace success. The role of educators in supporting proper team functioning includes helping students learn how to participate actively and communicate effectively in meetings, delegate work fairly, manage high-quality work throughput, and resolve conflicts if problems arise. Problems often emerge when team members have differing visions or individuals do not contribute equally to the work output. Detecting these potential issues in teams and helping students work through them is important for team success. In software engineering projects, monitoring individual contributions can begin with mining activities on programming platforms such as GitHub, which makes much of the individual contributions more visible and quantifiable. However, our literature review reveals that advances in code metrics are designed to address code complexity rather than individual work in team contexts. Thus, we turn to other literature for insights into alternative solutions.

Over the past twelve years, our Capstone course grew from an enrollment of 10 graduating computer science students to over 100 students. Project topics varied widely, ranging from web to mobile applications, to database optimization work. A mix of grading strategies has been used to assess individual contribution, team collaboration, and professionalism in client interaction. As the class grew, teaching assistants (TAs) were added to support grading tasks. However, TAs still struggled with identifying the relevant team indicators and often did not grade efficiently and consistently across teams. Problems became more serious in large classes involving multiple TAs. Section III explains the course context in detail.

To help manage large classes, previous attempts in mining code repositories to garner contribution statistics were used but they were not well received. Many projects use different programming languages and frameworks so developing a standard set of meaningful metrics is not easy. To amend these problems, as our first contribution, we propose a framework in Section III-C that considers multiple data sources and positions the human grader as the decision-maker of the assessment. From this perspective, we focus on building analytics tools to assist the human grader in their sense-making process by summarizing the code contributions and simplifying the analytics so that the information can reveal insights about the team's collaboration process.

Coupled with the framework is an analytics report that summarizes interaction and contribution behaviors from code commits, pull requests (PRs), and code reviews. This report serves as our second contribution. Section IV describes the system for building this analytics tool. In addition to simple metrics such as frequency and size information, we developed visualizations and metrics that help human graders differentiate active from inactive participation within teams. We are also interested in differentiating good, systematic collaboration processes that follow industry standards from inexperienced, inconsistent collaboration patterns. To do this, we created an analytics report focused on PR contributions called *PR reports* which are generated for each team on a weekly basis. These reports are designed to assist the human grader in their assessment and decision-making about the team's progress.

Section V describes a study using this tool in a software

engineering capstone course in 2023-2024 with 104 students split into 22 teams managed by 4 TAs and an instructor. We evaluate the utility of the PR reports by gathering survey feedback from the TAs who graded the teams weekly and managed the day-to-day interactions with their teams. Our investigation surrounds the following research questions:

RQ1: What are the administrative gains afforded by the use of PR reports?

RQ2: What are the potential risks of using these PR reports as part of the assessment process?

RQ3: What information should be used in place or in addition to the analytics in the PR reports?

Our findings reveal these analytics afford efficiency gains in the grading process and have a strong potential to detect problematic teams early. However, care must be taken so that TAs do not misinterpret the information presented. Our results also suggest that alternative ways of delivering the PR reports may improve the sense-making process. To develop a more holistic view of team interactions, future work will explore additional collaboration behaviors such as branching, merging, project management activities, and peer evaluation content. Future work also includes giving students access to these reports so we can study to what extent the analytics influence students' collaboration habits.

II. Related Work

A. Team Process Models

Many team process models attempt to explain team performance by viewing teams as a single entity [6], [16], [17], [25], [27]. While these descriptive models are useful in understanding effective teamwork, they do not explain how teams transition from one state to another. More importantly, these models do not explain how individual characteristics and behaviors influence team dynamics. In contrast, a multilevel theory of team effectiveness can capture the characteristics and interactions of individuals, teams, and organizations over time [15]. Unfortunately, most studies in this area have relied on traditional data collection methods, such as self-reported surveys which may not be accurate on their own, and repeated-design experiments which cause participant fatigue [14]. Instead, researchers are recommending the use of digital traces that result naturally from team interactions, such as communication logs, productivity management tools, and online collaborative activities. These types of digital traces can be used to build computational models of teams and individuals. For this reason, we turn to observable collaboration activities as a potential source for modeling team dynamics.

B. Collaborative Work and Code Metrics

Computer-supported cooperative work investigates how people use technology to collaborate on a common goal. Much work has been dedicated to studying team formation, collaboration dynamics, and tool design to facilitate collaboration [29]. However, the context of collaboration is often limited to groups that work on isolated tasks together without necessarily having to deal with long-term team dynamics and fair assessments that are central to capstone courses.

In software engineering, many researchers studied code complexity and developed a variety of metrics for source code analysis [20]. Many of these metrics have also been applied to investigating the difficulty of programming concepts and code complexity in student submissions (see [13], [21] for examples). Unfortunately, this approach is challenging to apply to a capstone context because teams typically work on different technology stacks, and parsing different programming languages and frameworks would pose too many obstacles. While mining code repositories is feasible, we are unaware of meaningful code metrics that allow for general comparisons across technology languages and frameworks that enable collaboration insights about the team members.

C. Capstone Collaboration Assessments

Many universities use a capstone course to help students pool the skills gained from their degree and apply them to open-ended projects. Typically, the capstone is structured to allow students to work with external partners in an authentic, project-based learning context [10]. For reasons mentioned earlier, it is becoming increasingly difficult to fairly assess individual contributions in a collaborative setting [11]. Studies on capstone assessments highlight the importance of fairness in group work but do not discuss how student collaboration or team processes might be measured. In particular, we found several reports on using rubrics to create transparent capstone assessments but they did not use collaboration as a criteria – some based their rubric criteria on the quality of the project deliverables [9], or project and course outcomes [23] [28], or discussed the assessment roles that different stakeholders had [4]. Three papers mentioned using teamwork as a criterion in their rubrics but no details were provided [1] [30] [19].

Some educators use peer evaluations to make assessments more fair in cases where individual contributions are unevenly distributed. For example, some educators use peer evaluations to identify when problems arise in teams but do not necessarily use peer evaluations to modify individual grades [2] [3]. Some instructors collect peer evaluation data to create estimates of individual contributions, then use them as an adjustment factor on the team deliverable [24] [11]. In another study, the researchers mention a 4-year study designed to train students to fill out accurate peer evaluation assessments [5]. Since peer evaluation scores are often biased and inflated, educators must be careful with how they are used to modify grades.

D. Weekly Assessments

Rather than conducting peer evaluations at the end of each milestone which may be susceptible to recency bias, we found several studies that used weekly peer evaluations beyond the capstone context. One of the earliest uses of weekly peer assessments was reported in 2006 where students distributed a percentage of the grade to each team member, along with a 5-point Likert scale question and an open-ended question [26]. A key strength in this type of assessment is the use

of a distribution question which enables students to focus on the amount and quality of work completed rather than the perceived competency of the teammates. Other researchers have used weekly assessments involving quantitative questions, but the question wording varied. In one case, students distributed percentages based on work performed, creativity in solving problems, and positive impact on group dynamics [18]. In another study, students completed weekly surveys with checkbox questions, Likert scale questions, and open-ended questions to elicit students' perceptions of work performed for early identification of dysfunctional teams [22]. Other researchers built a weekly survey to assess *team harmony* using multiple choice questions about tasks performed, amount of speaking time, and task assignment decisions and Likert scale questions about their sense of belonging, feeling valued, and perceived team functioning [12]. We adopt a mix of these approaches to develop our weekly peer evaluations.

III. Course Context

In our four-year undergraduate computer science program, we run a final-year software engineering capstone course that spans two semesters between September and April. In the first few weeks of the course, students spend time understanding course logistics, developing teamwork, establishing a common collaboration process, and writing a project charter that defines the requirements. By the end of October, the teams will have developed design mocks, set up their technical stack, designed some database relationships, and started implementing their software solution. Generally, the teams are expected to have completed a minimal working prototype by December so they can focus on adding features in January. In the second semester, students work on extending the prototype, gathering peer feedback on their systems, and fixing bugs. By April, each team will have a fully developed project to showcase. In some cases, projects have known bugs documented in the handover document. Many projects also get deployed publicly, such as websites and mobile apps.

The course has three evaluation components: a team component, an individual component, and a client component. Their details have been refined over the years. Students are required to pass all three components to pass the course.

A. Past Assessment Challenges

Common issues observed in previous course offerings include higher performing students carrying the team, certain students assigned to all the non-technical work, and students generally reluctant to do written and reflective work. When class sizes increased, close team monitoring became difficult and we attempted to mine Github repositories to garner code contribution insights. However, students disliked having their work summarized in terms of common code metrics such as lines of code, cyclomatic complexity, or the number of dependencies. Students who worked on projects that required heavy system administration felt such code metrics could not capture their effort accurately. Teams that used frameworks with auto-generated code also expressed a misrepresentation of contributions for the individual who happened to set up the project and committed those lines of code. With varying technology stacks and development environments, students felt that progress should be evaluated differently because some situations require a steeper learning curve.

Assessing individual contributions was challenging because some students worked on "invisible tasks" like project management and client liaison. This is a well-known problem in the literature [8]. While we tried to encourage students to divide all the work fairly and evenly, some students are consistently left with less technical tasks.

B. General Assessment Goals

While we acknowledge that not all aspects of collaboration can be automated, we wish to quantify common tasks so we know how students spend their time. Specific to software projects on the GitHub platform, we wish to identify metrics that help unveil whether students are following the code collaboration and review processes established within their teams. We will use these two objectives to guide the design of our assessment approach described below.

C. Framework for Assessing Work Contributions

We propose a triangulation framework with three sets of data that offer a more accurate picture of individual contributions. This framework is illustrated in Figure 1 where each data source and each validation iteration help to unravel an individual's involvement. The three data sources are:

1) **Self-Reported:** Logs and peer evaluations
2) **Observed:** Code repository contributions
3) **Meetings:** In-class check-ins

Data is collected weekly and this process repeats throughout the project to enable the teaching staff to become familiar with the members of each team.

The first data source uses students' self-reported progress logs and peer evaluation information to reveal the amount of work done across the team. Since these team and individual logs are pushed to the team repository, teammates can validate the truthfulness of each other's claims. Students also completed several peer evaluation questions to help us assess team dynamics. We asked students to distribute a percentage across all the members to represent the amount of work, talking, and decision-making each student did in the past week. Students also completed a checkbox question by selecting the

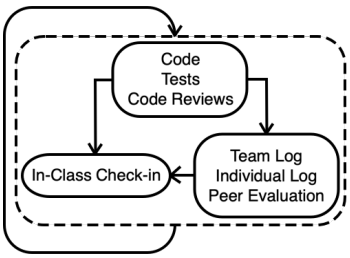

Fig. 1. The triangulation framework for assessing individual contributions. Nodes represent data sources and arrows represent direction of flow.

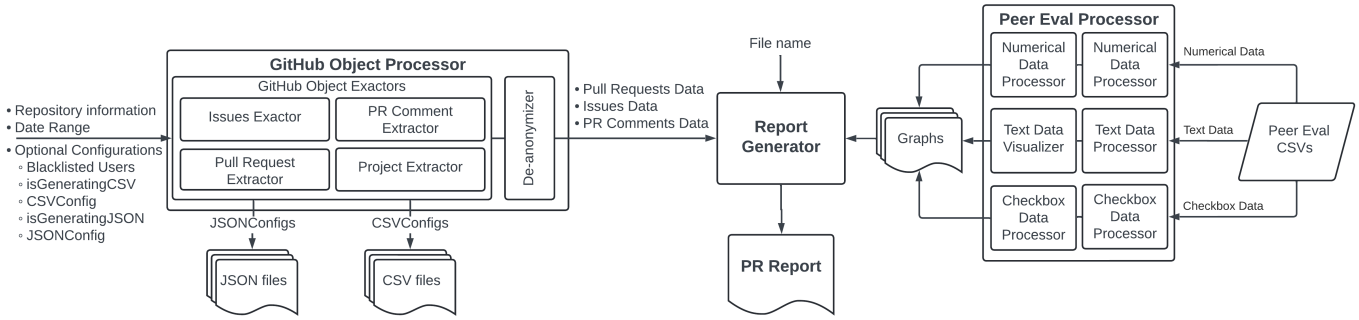

Fig. 2. System Architecture Diagram

activities they spent time on. These tasks are designed to encompass both technical (e.g., programming, testing, system administration) and non-technical work (e.g., report writing, presentations, diagram construction, interface design).

The second data source comes from the code repository, where we assess code contributions, tests, and code reviews. Although this data is purely objective, the quality and the relative difficulty of the contributions are subjectively measured by the teaching staff.

The third data source is an in-class meeting between the student team and the TA (or instructor) to review project progress, potential problems, and disputes on previous grades. Before each meeting, TAs should ideally review all the repositories, logs, and peer evaluations from the previous week. Since there is a lot of data to synthesize, TAs did not always come prepared in practice. To facilitate these meetings, we summarized the pull request activities for each team in a *PR report* and generated one report per team. These reports also included an overview of the peer evaluation data designed to help the TAs quickly identify anomaly patterns. The next section presents the system designed to create these reports.

IV. System Architecture

Figure 2 shows the system architecture for generating a PR report. The system has three main components: *GitHub Object Processor*, *Peer Evaluation Processor*, and *Report Generator*. The two processors are responsible for parsing the relevant input data based on the specified parameters such as date range and output format. The output of the GitHub Object Processor is a set of GitHub pull requests, issues, and pull request code review comments. The output of the Peer Evaluation Processor is a set of graphs that visualize different patterns in the peer evaluation data. The Report Generator merges the outputs from these two components into a readable format. This component generates a PR report for each team based on their repository and peer evaluation data. Each report provides overall statistics and visualizations to give the reader an overview of the activities that took place in the past week. The reports also have accompanying details about the pull requests to facilitate a deeper diagnosis without having to dive into each code repository. Example details of a PR report are presented in the remainder of this section.

A. Collaboration Analytics

The PR report is designed to assist team management by gathering different data sources in one place and showing an overview of that information. First of all, the report presents a broad overview of productivity at the individual and team levels as shown in Figure 3. This table presents information about pull request statuses, the number of commits, lines, and the number of files changed related to these PRs for each member of the team and for the whole team.

A key aspect of the PR report is to enable the TA to glean insights about the team's code development processes so they can help the team improve their collaboration habits. For example, Figure 3 shows GitHub User C has 3 small unmerged PRs. This suggests the user may be lost or have trouble finishing tasks. Comparing GitHub User A and B, we see they merged two PRs but the first involved 11 file changes while the second involved only 4 file changes, suggesting that the first user's tasks were more complex. Also, GitHub User A made a total of 6 commits involving 280 line additions and 26 line deletions. One may argue that more commits should be made in this case to practice better version control habits. For users without merged PRs, they may need help scoping their PRs and sizing their features. Many students feel small PRs are insignificant and prefer to spend time building more code. However, if the PRs are too big, students may have more code conflicts and unwanted code which cause future complications. Lastly, we see GitHub User D has 1 commit but no PRs. This means the user made the commit on someone else's branch, and a deeper conversation around those activities is warranted.

Figure 4 shows the team's line edits visually. This bar graph is designed to give the reader a quick comparison of the

Contributor	PRs (Merged / Not Merged)	Commits	Lines added	Lines deleted	Lines contributed	Files changed
GitHub User A	2 (2 / 0)	6 (3.0/PR)	280 (140.0/PR)	26 (13.0/PR)	254 (127.0/PR)	11 (5.5/PR)
GitHub User B	2 (2 / 0)	4 (2.0/PR)	72 (36.0/PR)	1 (0.5/PR)	71 (35.5/PR)	4 (2.0/PR)
GitHub User C	3 (0 / 3)	8 (2.7/PR)	217 (72.3/PR)	125 (41.7/PR)	92 (30.7/PR)	8 (2.7/PR)
GitHub User D	0 (0 / 0)	1 (0/PR)	0 (0/PR)	0 (0/PR)	0 (0/PR)	0 (0/PR)
GitHub User E	2 (0 / 2)	32 (16.0/PR)	1,236 (618.0/PR)	957 (478.5/PR)	279 (139.5/PR)	5 (2.5/PR)
Total	**9 (4 / 5)**	**51 (5.7/PR)**	**1,805 (200.6/PR)**	**1,109 (123.2/PR)**	**696 (77.3/PR)**	**28 (3.1/PR)**

Fig. 3. Overview table of PR activities.

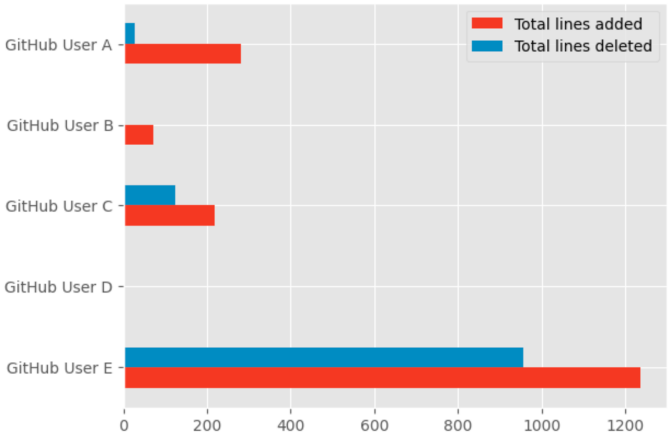

Fig. 4. Bar graph of line contributions.

relative contributions made by the team members. Although GitHub User E made a lot of line additions and deletions, the relative code contribution is similar to that of GitHub User A.

The next part of the report displays PR details. These details are meant to give the reader more information about the PRs by presenting the header descriptions as shown in Figure 5 and commit messages as shown in Figure 6. The idea is that this information should give the reader a sense of what the PR is trying to accomplish without reading the code. If the header description is missing or poorly written, the TA should help train the student to write more meaningful descriptions. Based on the commit messages shown in Figure 6, we see that this user does not provide much detail at all. Which bug was fixed? What did the tests handle? Which comment was resolved? The lack of details in these commit messages often makes the review process slower, especially if the team members are not familiar with each other's work. In order to better understand what was in this PR, the reviewers and the TA would need to read the code more closely. Ideally, students should be directed to produce more details in their messages so that the TA management process can operate more efficiently.

Lastly, we have a table summarizing code review activities as shown in Figure 7. It includes data such as the number of

PR Title: Browse screen reserve

PR Author: GitHub User B

Description: Removed the reserved post from browse screen and book mark screen

Status: Open
Number of commits: 6
Lines added: 13
Lines deleted: 6
Lines contributed: 7
Files changed: 2
Reviewers: GitHub User C , GitHub User D , GitHub User B
Created at: 2024-02-18T17:50:52Z
Closed at: Still Open

Fig. 5. Header details for a given PR.

Commit History
@GitHub User B - Fix bug
@GitHub User B - Remove whitelist after unsubmit
@GitHub User B - Add tests
@GitHub User B - resolve comment
@GitHub User B - Resolve comment

Fig. 6. Commit history for a given PR.

review comments made (including approval messages, request changes, and general comments), the number of review replies made, the average number of words, and the total number of reviews made at the individual and team level. This table provides a concise summary of PRs that are not reviewed which should be alerted to the TA. It also provides an estimation of the depth of the written comments. For example, the comment "looks good to me" is quite shallow and only counts as 4 words. Also, the number of code review comments made by an individual indicates their level of involvement with other member's code in the repository, which reflects the interdependencies of the team based on code review activities. Overall, the activities reported in Figure 7 suggest this team does not have a strong code review process because they did not have any discussions in the PRs (as indicated by 0 review replies) and the approvals are usually shallow (the number of total reviews is more than the number of comments).

B. Integration with Other Data Sources

The PR report also includes data from weekly peer evaluations that report individual tasks, student perceptions of their teammates, and written acknowledgments or complaints. This information was only added to the PR report at the end of the year so the TAs only saw these graphs in the last week. A common scenario where students may be involved in programming but the work is not visible is when they are pair programming with another person. To account for this invisible work, we ask students to explicitly acknowledge other teammates when they receive help. Based on this information, we create an acknowledgement graph as shown in Figure 8. The system extracts team member names from the peer evaluation data and plots a histogram based on the number of times those names were mentioned in a given week. For example, we see Student A is a frequent helper in Figure 8. Based on the number of acknowledgments shown in this graph, this team likely has good interaction and are able to work closely together.

Next, the system extracts the percentage of work completed from the student evaluations and graphs them longitudinally.

Contributor	Comments	Review Replies	Words per Comment	Reviews
GitHub User A	2	0	9.0	4
GitHub User B	0	0	0	1
GitHub User C	1	0	12.0	1
GitHub User D	2	0	16.5	3
GitHub User E	2	0	12.5	2
Total	**7**	**0**	**12.6**	**11**

Fig. 7. Overview table of code reviews associated with the PRs.

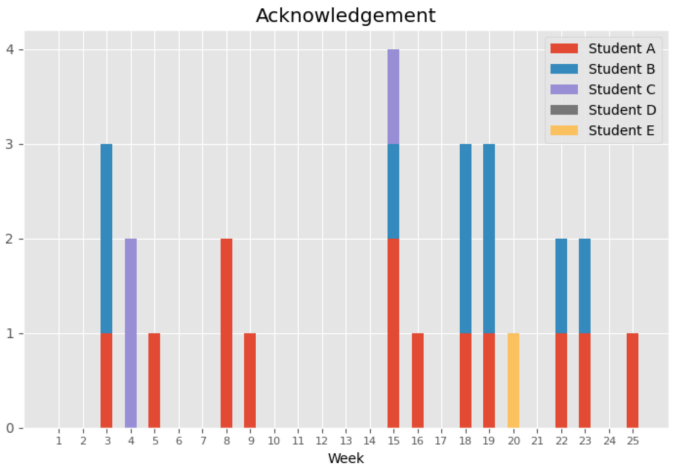

Fig. 8. Acknowledgement graph displayed for the whole project.

Figure 9 shows an example of this longitudinal work graph for a team of 5 members. We see quickly that on average, most data points hover around the 20% mark, indicating the team generally felt their work was evenly distributed. Although not presented in this paper, we also have similar peer evaluation data asking students to report on the percentage of people talking and making decisions. Analogous graphs are also generated to show the dominance relationships within a team.

Recall the peer evaluation checkbox question listing the categories of tasks commonly needed in capstone projects. Our system took this information, weighed it using the average percentage of work done calculated by their teammates' responses, and constructed a heatmap as shown in Figure 10 with the list of tasks displayed on the left and the weeks at the bottom. Note that the darker the colored box is the more time is spent doing that task as perceived by the teammates. If the team collectively reports that one member did not do any work in a given week, the average percentage of work done for that student will be 0% and the column of tasks for that week will remain uncolored even if the individual claims

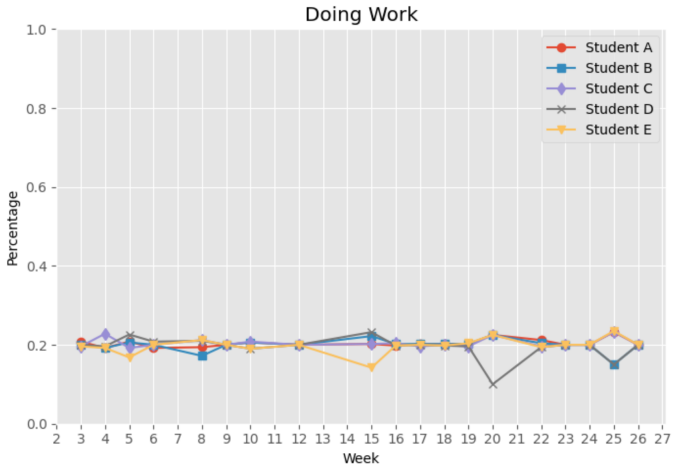

Fig. 9. Average percentage of work done perceived by other teammates.

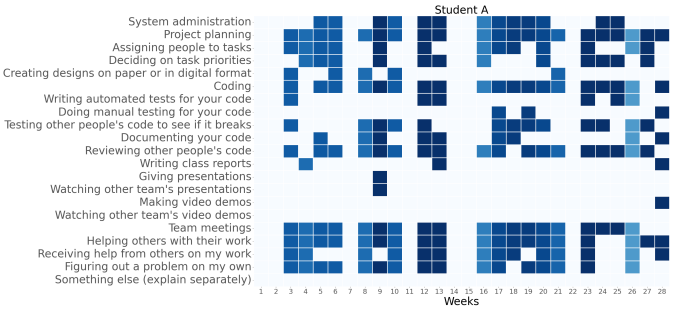

Fig. 10. Self-Assigned Tasks Done Graph

to have done work on certain tasks. From Figure 10, we see Student A did not work in weeks 1, 2, 7, 13, 14, and 21.

Overall, the PR report can also be used as a tool for assessing the truthfulness of the amount of work that students claim to have done. The report enables the reader to quickly cross-reference information that would have otherwise taken a long time to check across multiple data sources.

V. Pilot Study

A. Participants

The primary participants of the study were 4 TAs of the Capstone course. Among them, there were 2 males and 2 females, all 4 of whom were racial minorities. With the 104 students grouped into 22 teams, 2 TAs managed 4 teams and the other 2 TAs managed 7 teams.

B. Procedure

At the end of the course, we distributed a Qualtrics survey to the TAs to provide anonymous feedback about the PR reports. Everyone was given 2 weeks to complete the survey.

C. Materials

The TA survey had 20 questions about the PR process, with 10 structured questions and 10 open-ended questions. The structured questions had 4 Yes/No questions about the impact the PR report had on the ease of evaluation, the guidance it offers, and new information to include. There were also 6 even-point Likert questions about their accuracy in assessment with and without the PR reports and the helpfulness of the PR reports. The open-ended questions asked the TAs to provide explanations and suggestions for improvements.

D. Analysis

The structured questions are summarized numerically. To analyze the open-ended responses from the TA survey, we conducted a thematic analysis and developed codes discovered in the responses [7]. The responses were segmented by sentences and further by phrases if a conjunction was used to separate out multiple ideas. Specifically, two raters independently analyzed all of the qualitative responses from these surveys based on the codebook established from the initial round of inductive familiarization, followed by two rounds of deductive categorization. In total, the raters completed two passes to reach an

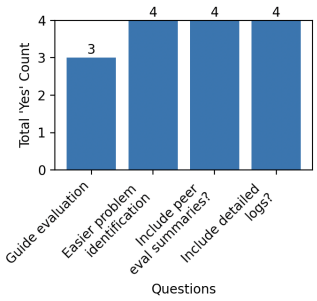

Fig. 11. Total counts from Yes/No questions based on TA responses.

acceptable intercoder reliability level of $\alpha = 0.88$ for the TA data. The final codebook is provided in Table I.

E. Results

We developed 13 codes to capture the TA responses: New Information (20), Insightful (16), Supplemental (11), Generic (11), Positive (7), Neutral (7), Efficient (5), Misleading (5), Access (4), Unclear (4), Excessive (3), Easy (2), Inaccessible (1). We explain these codes below. In two Yes/No questions shown in the first and second bars of Figure 11, we see the TAs generally felt the PR reports were useful in guiding their weekly evaluations and made it easier for them to identify where problems occurred.

New Information, where participants suggest new information to be included in the PR reports, have the highest frequency count. On average, each TA made 5 suggestions. There were 3 categories of suggestions: filtering data (e.g., based on dates and PR status), providing more data (e.g., list of review comments, dates for commit activities), and general improvements (e.g., definitions where terminology may be unclear). The TAs also unanimously agreed that peer evaluation and weekly log details should be added to the PR reports, as shown in the third and fourth bars of Figure 11.

TAs felt the PR reports were *insightful* and this code appeared 16 times overall. On average, the TAs mentioned these reports were insightful 4 times per person, stating comments such as "[the PR report] can give a quick insight about the individual contributions and the team contributions". Figure 12 supports this finding, showing that TAs generally strongly agreed the reports provided insights into assessing both individual contributions and team progress.

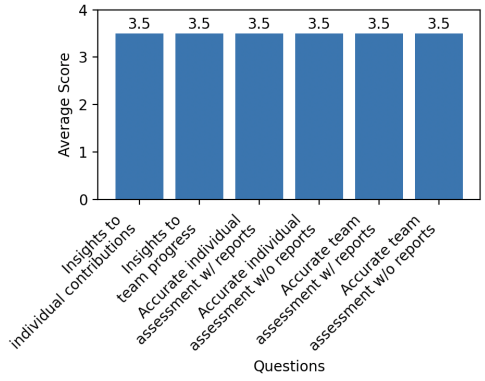

Fig. 12. Average scores reported from 4-point Likert scale questions (1=Strongly Disagree; 4=Strongly Agree) based on TA responses.

We were curious whether the TAs felt confident in accurately evaluating individual and team assessments using the PR reports. As seen in Figure 12, the TAs generally strongly agreed in these respects. However, we asked them these questions in the context of using the PR reports (most of the semester) and without using the PR reports (at the very beginning of the semester). We see no change in their agreement. This suggests that, with or without the PR reports, the TAs were confident in their skill to assess the teams. Thus, the value offered for the TAs may be limited to efficiency only.

The TAs mentioned using the PR reports as a *supplemental* tool to assess student work almost 3 times per person on average. This suggests that the PR reports do not replace the actual repository inspection done by the TAs. As one TA commented, "I used the PR reports but mainly relied on going through the actual commits and the individual logs".

The remaining codes appeared about once per TA, or less. All the TAs found the PR reports made the assessment more *efficient*. Most of the responses refer to a quicker process using PR reports than not using them, such as "I could point directly to the report and ask them questions, which sped things up" or "spending less time overall hunting down work that had been done". This suggests that PR reports improve efficiency specifically in measuring the amount of work done by individual team members. Related to efficiency, 2 comments were coded as *easy*, which denotes the PR reports were easy to use or made the assessment process less overwhelming. The PR reports seem to be beneficial for assessing students, as "they made things WAY easier and clearer". Recall from Figure 11 that most TAs found the PR reports helped guide the evaluation process and locate the problems.

The TAs also felt that the PR reports made the discovery and access to information about the student activity easily *accessible* because "everything [is] neatly in one place". However, one response was coded as *inaccessible*, indicating that tracking individual contributions was made more difficult. This could be related to the problems of information being *unclear* and *excessive*. We believe some text definitions can be used to clarify potential misconceptions of the presented data. Perhaps a better organization of the information or an interactive version of the PR reports where TAs can select the desired graphs and details would be preferable.

Unfortunately, the TAs also found some aspects of the reports as *misleading*, because they gave inaccurate data. For instance, the report shows PR activity simply because its status is open even if the activities took place in previous weeks. Unfortunately, this is a limitation of the GitHub API that we encountered. Another TA stated that "PRs can be very skewed and the charts [can] lose meaning". This means, if one member has tens of thousands of auto-generated lines of code added, then the rest of the contributions would not be visible in the graph. These responses suggest the system needs better data

TABLE I
CODEBOOK USED IN THE ANALYSIS OF TA DATA, SHOWN WITH CODING FREQUENCIES.

Code	Frequency	Explanation
New Information	20	Suggestions of new types of info to include, such as filter by week, updated timestamp, actual review messages E.g. "Having the date it was updated might help us determine whether a PR is just sitting open, or whether it's actually being worked on."
Insightful	16	PR report gives insights that would otherwise not have been discovered; problems and issues a team experienced as derived from their report E.g. "I probably wouldn't have noticed the branching strategy issue as quickly as I did without the reports"
Supplemental	11	Old methods of assessing performance is still found to be more effective. E.g. "I still heavily rely on the github repo for details, the report helps me to see if there's anything I might have missed."
Generic	11	Anything not relevant or nothing to add E.g. "I cannot think of anything new"
Positive	7	General positive opinion with no specific information or reason, generally helpful E.g. "The PR reports were wonderful."
Neutral	7	Opinion is neutral E.g. "Further, I'd often see commits from multiple different people on the same branch."
Efficient	5	PR report makes assessments faster than before (requires less time) E.g. "This meant that I was spending less time overall hunting down work that had been done, because now I knew where to look."
Misleading	5	PR report gives inaccurate data that leads to false conclusion E.g. "Some things on the PR reports may be misleading on a glance, for instance number of PRs per student include PRs for merging into branches."
Access	4	Easy to locate and access information about student activity E.g. "The PR reports showed me the aggregated total of comments, which helped me to search them out on GitHub."
Unclear	4	PR report requires additional information to allow for understanding, for example, adding time stamps E.g. "The amount of code lines does not directly reflect the amount of work done so it was necessary to go through it."
Excessive	3	When too much information becomes a bottleneck E.g. "Some PRs that remain open are not dealt with show up every week."
Easy	2	PR report is easy to use, or a way to make comparisons easier, or makes part of a process easier E.g. "It also makes it easier to track which PRs I need to look at on the repo."
Inaccessible	1	Difficult to locate and access information in the report, easier to find directly in the repo E.g. "This also make the contributions of persons who did those hard to track."

filtering and cleaning to remove ambiguity and potential bias.

Finally, 25 comments were coded as *Generic*, *Positive*, or *Neutral* because they expressed general sentiments.

VI. DISCUSSION

A. Threats to Validity

We note that this pilot study primarily involves the self-reported experience of 4 TAs in one course. While we tried to gather more details from them to garner deeper insights, we acknowledge the data obtained from this study is limited.

B. RQ1: Administrative Gains

Our findings reveal that TAs find value in using the PR reports as part of their evaluation process. While it is clear that these reports do not replace repository analysis, they all believe that the information in the PR reports helps guide their assessment and locate where problematic areas are. Thus, PR reports help create better conversations between students and their TAs, providing a broad overview of repository activities and team dynamics. All the TAs indicate using PR reports improves their grading efficiency.

C. RQ2: Risks and Drawbacks

As mentioned in the results, some TAs felt the information was not labeled correctly or presented in a misleading way. Fortunately, when the students raised concerns during the in-class checkin's, the TAs were able to verify what was going on by checking the detailed activities in the code repositories. On the other hand, if the TA relied solely on the information in the PR report and ignored the student's concerns, the problem would have escalated further.

If used on their own, PR reports do not suffice in encapsulating accurate details about individual contributions done outside of the code repository. However, the TAs knew that code contributions constitute one of the data sources in the assessment triangulation framework presented in Section III-C, so this was not a problem.

Although our goal in designing these PR reports is to help students improve their collaboration habits, it is not clear whether the TAs used the reports this way and whether the students' collaboration processes improved. Future work needs to control this aspect more carefully to encourage better TA guidance. Alternatively, another study could involve students accessing the reports directly and evaluating the impact on their collaboration behavior.

D. RQ3: New Information and Features

The TAs suggested report improvements by including more details in certain areas and making the content more clear. Another suggestion was to provide different ways for filtering the data. However, this might create a cognitive overload on the reader when there was already some feedback about the

reports being overly excessive. A possible solution is to create an interactive report where data is dynamically filtered.

VII. Conclusions and Future Work

This work presented a novel system for analyzing GitHub repositories and peer evaluation data to generate PR reports that provide context about individual contributions and team collaboration. The purpose of this work is to assist the teaching staff in better managing their student teams in large software development classes. Previous automation attempts that provided superficial code metrics did not measure work output accurately and fairly. With PR reports, the focus of the assessments now shifts from measuring individual code contributions to understanding the individual's role relative to the team's work and their collaboration process. It is important to note that these collaboration analytics should be used holistically in the broader assessment framework, such as the triangulation framework proposed in this paper, while considering the specific context of the project and the expectations set for the team. Individual contributions are multifaceted, and a combination of these metrics provides a more comprehensive understanding of a team member's impact and effectiveness within the collaborative environment.

Our next step is to run another study and emphasize collaboration habits. Future work includes detecting tests, computing test coverage, and incorporating chore activities available in GitHub projects such as issue creation and assignments.

References

[1] F. Abu Salem, I. Damaj, L. Hamandi, and R. Zantout. Effective assessment of computer science capstone projects and student outcomes. *International Journal of Engineering Pedagogy (iJEP)*, 10:pp. 72–93, 2020.

[2] R. Adams and C. Kleiner. Collaboration support in an international computer science capstone course. In G. Meiselwitz, editor, *Social Computing and Social Media*, pages 313–323, 2016.

[3] G. I. Allen. Experiential learning in data science: Developing an interdisciplinary, client-sponsored capstone program. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, page 516–522, 2021.

[4] M. C. Bastarrica Piñeyro, D. Perovich Gerosa, and M. Marques Samary. What can students get from a software engineering capstone course? In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track*, 2017.

[5] F. C. Berry, W. Huang, and M. Exter. Improving accuracy of self-and-peer assessment in engineering technology capstone. *IEEE Transactions on Education*, 66:174–184, 2023.

[6] C. Bowers, R. Oser, E. Salas, and J. Cannon-Bowers. Team performance in automated systems. In R. Parasuraman and M. Mouloua, editors, *Automation and Human Performance: Theory and Applications*, chapter 12. CRC Press, Boca Raton, 1996.

[7] V. Braun and V. Clarke. Thematic analysis. In H. Cooper, P. Camic, D. Long, A. Panter, D. Rindskopf, and K. Sher, editors, *APA handbook of research methods in psychology: Research designs: Quantitative, qualitative, neuropsychological, and biological*, page 57–71. American Psychological Association, 2012.

[8] R. Cross, S. Borgatti, and A. Parker. Making invisible work visible: Using social network analysis to support strategic collaboration. *California Management Review*, 44(2):25–46, 2002.

[9] N. Deepamala and G. Shobha. Effective approach in making capstone project a holistic learning experience to students of undergraduate computer science engineering program. *JOTSE: Journal of Technology and Science Education*, 8:420–438, 2018.

[10] R. F. Dugan. A survey of computer science capstone course literature. *Computer Science Education*, 21(3):201–267, 2011.

[11] V. Farrell, G. Ravalli, G. Farrell, P. Kindler, and D. Hall. Capstone project: Fair, just and accountable assessment. In *Proceedings of ITiCSE'12*, pages 168–173, 2012.

[12] N. Heyl, E. Baniassad, and O. Ola. Team harmony before, during, and after covid-19. In *Proceedings of the 2022 ACM SIGPLAN International Symposium on SPLASH-E*, page 52–61, 2022.

[13] P. Ihantola and A. Petersen. Code complexity in introductory programming courses. In *Proceedings of the Hawaii International Conference on System Sciences*, pages 7662–7670, 2019.

[14] S. Kozlowski and G. Chao. Unpacking team process dynamics and emergent phenomena: Challenges, conceptual advances, and innovative methods. *American Psychologist*, 73(4):576–592, 2018.

[15] S. Kozlowski and K. Klein. A multilevel theory, research, and methods in organizations: Foundations, extensions, and new directions. In S. Kozlowski and K. Klein, editors, *A multilevel approach to theory and research in organizations: Contextual, temporal, and emergent processes*, pages 3–90. Jossey-Bass, San Francisco, CA, 2000.

[16] P. Lencioni and C. Stransky. *The five dysfunctions of a team*. Random House, Inc, 2002.

[17] W. M. Marston. *Emotions of normal people. by William Moulton Marston with a new introd. by S.H. Irvine*. Thomas Lyster, 1989.

[18] M. Modell. Iterating alone over a method and tool to facilitate equitable assessment of group work. *International Journal of Designs for Learning*, 4(1):39–53, 2013.

[19] A. Neyem, J. Diaz-Mosquera, J. Munoz-Gama, and J. Navon. Understanding student interactions in capstone courses to improve learning experiences. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, page 423–428, 2017.

[20] A. N. nez Varela, H. Pérez-Gonzalez, F. Martínez-Perez, and C. Soubervielle-Montalvo. Source code metrics: A systematic mapping study. *Journal of Systems and Software*, 128:164–197, 2017.

[21] H. Nguyen, M. Lim, S. Moore, E. Nyberg, M. Sakr, and J. Stamper. Exploring metrics for the analysis of code submissions in an introductory data science course. In *Proceedings of the International Learning Analytics and Knowledge Conference*, page 632–638, 2021.

[22] K. Presler-Marshall, S. Heckman, and K. Stolee. Identifying struggling teams in software engineering courses through weekly surveys. In *Proceedings of the ACM Technical Symposium on Computer Science Education (SIGCSE)*, page 126–132, 2022.

[23] T. Sasipraba, R. K. B. Navas, N. M. Nandhitha, S. Prakash, J. Jayaprabakar, et al. Assessment tools and rubrics for evaluating the capstone projects in outcome based education. In *Proceedings of the 9th World Engineering Education Forum (WEEF)*, pages 296–301, 2020.

[24] C. C. Tappert, A. M. Leider, and S. Li. Student assessment in a capstone computing course. In *Proceedings of the 2019 Southeast Decision Sciences Institute (SEDSI) Conference*, pages 1–7, 2019.

[25] K. W. Thomas and R. H. Kilmann. Comparison of four instruments measuring conflict behavior. *Psychological Reports*, 42:1139 – 1145, 1978.

[26] R. Tucker and C. Reynolds. The need for a culturally inclusive andragogy for collaborative design learning. In *Proceedings of the Australian University Building Educators Association Annual Conference*, pages 1–10, 2006.

[27] B. W. Tuckman. Developmental sequence in small groups. *Psychological Bulletin*, 63(6):384–399, 1965.

[28] M. Vijayalakshmi, P. D. Desai, and G. H. Joshi. Outcome based education performance evaluation of capstone project using assessment rubrics and matrix. In *Proceedings of the IEEE International Conference on MOOC, Innovation and Technology in Education (MITE)*, pages 6–10, 2013.

[29] J. Wallace, S. Oji, and C. Anslow. Technologies, methods, and values: Changes in empirical research at CSCW 1990 - 2015. In *Proceedings of the ACM on Human-Computer Interaction*, pages 1–18, 2017.

[30] J. Yousafzai, I. Damaj, and M. El Abd. A unified approach for assessing capstone design projects and student outcomes in computer engineering programs. In *Proceedings of the IEEE Global Engineering Education Conference (EDUCON)*, pages 333–339, 2015.