Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories

Conference Paper · February 2020

DOI: 10.1145/3328778.3366948

1 author:

Kevin Buffardi
California State University, Chico
40 PUBLICATIONS **437** CITATIONS

Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories

Kevin Buffardi
California State University, Chico
Chico, California, USA
kbuffardi@csuchico.edu

ABSTRACT

Software Engineering courses often incorporate large-scale projects with collaboration between students working in teams. However, it is difficult to objectively assess individual students when their projects are a product of collaborative efforts. This study explores measurements of individuals' contributions to their respective teams.

I analyzed ten Software Engineering team projects (n=42) and evaluations of individual contributions using automated evaluation of the version control system history (Git logs) and user stories completed on their project management (Kanban) boards. Unique insights from meta-data within the Git history and Kanban board user stories reveal complicated relationships between these measurements and traditional assessments, such as peer review and subjective instructor evaluation. From the results, I suggest supplementing and validating traditional assessments with insights from individuals' commit history and user story contributions.

CCS CONCEPTS

• **Social and professional topics** → **Project management techniques**; • **Software and its engineering** → **Software configuration management and version control systems**; *Agile software development*;

KEYWORDS

software engineering, team projects, individual assessment, performance appraisal, git, scrum, kanban, user stories

ACM Reference Format:
Kevin Buffardi. 2020. Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA*. ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.1145/3328778.3366948

1 INTRODUCTION

Association for Computing Machinery (ACM) suggests that computing curricula include a core foundation in *Software Engineering* and *Professional Practices* on top of *Software Development Fundamentals* [2]. Consequently, Software Engineering courses often involve students in larger-scale team projects to more closely resemble realwold software development. However, along with the advantages of more realistic team collaboration come unique challenges for instructors to assess individuals' work with fairness and objectivity.

Different models for team projects have been adopted in Software Engineering curricula, including working on projects for open source software[28][31], humanitarian purposes[5][14][23], industry partners[6][12][16][22][27], and entrepreneurial ventures[9][13]. Each of these models provide valuable experiential learning opportunities for students, but also pose challenges for fair grading strategies. Regardless of the type of software engineering project a team works on, it is a particular challenge to assess each individual's contribution to their team project.

Some instructors choose to grade each team as a whole, where each member of the team receives the same grade. However, it is not necessarily safe to assume that the division of work is split evenly among team members. Individuals often exhibit *social loafing* behaviors in team settings when they lack motivation and accountability. Social loafing is the phenomenon where individuals exert less effort toward a team goal than they would when working alone[19]. As the result of social loafing, some software engineering teams include members who do not contribute their fair share of the work. In these cases, it would be unfair for the other team members to suffer from lower marks, especially if they have tried to engage and motivate the social loafers.

Peer evaluation is a popular assessment method to account for differences in contributions between team members. For example, Fagerholm and Vihavainen developed a framework for students to evaluate themselves and their teammates on interpersonal and other "soft skills"[15]. However, a study of the objectivity in such evaluations found troubling trends of inaccurate self-assessment[18]; the author also warned of potential bias against females and other under-represented minorities in teams. Likewise, there is similar concern for subjective instructor assessments since they are also be susceptible to implicit biases[34].

To the contrary, performance metrics may provide more objective evidence for individual assessment. Additionally, some tools and techniques commonly adopted in Software Engineering courses [7] already capture information about individuals' activities within their teams. Consequently, in this report, I explore measurements of individual students' contributions that can be extracted from popular professional practices.

Specifically, I investigated evidence within teams' version control systems (Git) and project management workflow (Kanban) boards.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366948

Exploratory analysis of ten software engineering teams (n=42) examines methods for assessing individual contributions from records of their committed code as well as from the end-user tasks they implemented. This study analyzes these quantitative measurements and their relationships with traditional qualitative assessment methods. The study's findings prompt a discussion about the potential for subjectivity, biases, and possible unintended consequences from using different assessments.

2 BACKGROUND

A study of how students prefer to be evaluated on team projects found that with more academic experience, their preferences shift toward stronger weights on team (rather than individual) evaluation [33]. However, regardless of student preferences, it is important for individuals to develop soft skills and contribute meaningfully in team settings. Software Engineering courses have a distinct opportunity to assess individuals' contributions to teams while the those students learn to apply professional practices in a collaborative software development environment.

CATME [20] is a widely-adopted team formation and peer evaluation tool that has been studied and validated [26] on various engineering projects for over a decade [25]. CATME provides an online peer rating questionnaire that includes evaluation of each individuals' interactions with their teammates as well as their contributions to the project. Students rate each of their teammates by selecting items on a five-point scale that best describe their interactions with- and contributions to- the team. While it is a useful tool for gathering insights into students' ratings of each other, peer evaluations have inherent limitations.

One potential threat to objectivity of peer evaluation is the *Halo Effect*. The Halo Effect describes a bias in performance appraisal where the evaluator's assessment of an individual is unduly influenced by a single, sometimes unrelated, characteristic. For example, an evaluator who appreciates an individual's agreeableness may rate that individual higher on technical skills–even when lacking–because of the favorable impression made by their personality[30].

Concerns about the Halo Effect extend beyond peer evaluations since teachers have also been found to be susceptible to the bias when evaluating students [1]. Moreover, Herbert [18] also raises concerns over how students' ethnicity and gender might unfairly influence peer evaluations in teams due to explicit or implicit biases. Broader studies in education have found that implicit biases impact an achievement gap between students of different ethnicities even when self-reported biases did not [34]. Computing fields already have a lack of diverse representation. Negative impacts of implicit bias could compound the problem.

Meanwhile, project-based Software Engineering courses often teach and engage students in professional tools and practices that create records of their work. Software configuration management and version control is fundamental to software development and is pervasive in Software Engineering curricula [2]. *Git* is a popular version control system that has been widely adopted in Software Engineering classes [3] and is emerging as a common tool throughout computing curricula [29].

Git maintains a log of contributions (or "commits"), including their commit messages (a description of the revision), time, author, and how many lines have changed (including both insertions and deletions). Instructors can find evidence of students' contributions to a project by identifying commits in the log that identify the student as the author. Other version control systems support similar capabilities, but due to its popularity, I concentrate on Git specifically in this report.

Project management tools have also grown in popularity, especially in support of *Agile Software Development* frameworks. Agile emphasizes "Deliver[ing] working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale" and promotes sustaining a constant pace of development [4]. To plan and track progress of development in short intervals, software engineering teams often adopt Kanban boards. Kanban boards keep track of tasks with columns representing their statuses: *To Do*, *In progress*, or *Done*. Kanban boards are specifically integrated into some Agile frameworks [24] and built into online tools including GitHub [17], the popular Git repository host.

While Kanban boards generally dedicate a card to a particular task that needs to be completed, some Agile frameworks advocate for cards to be written as *User Stories*. User Stories describe software features from the perspective of the customer (or end-user) so that it explains a goal they want to achieve; stories usually follow the format: "As a *<role or persona>*, I want to *<action>* so that *<goal>*." Teams prioritize and plan which stories should be completed during an iteration in order to deliver the most value to the customer for the least effort. The team also assigns *points* to represent how difficult it will be to develop, relative to other stories. These points can be used to monitor pace of development [21].

As stories are implemented and completed, they are moved along the Kanban board to indicate their progress. GitHub Projects supports creation of Kanban boards dedicated to a project's repository. In particular, it provides features to assign developers to specific cards and story points can be indicated on the card's description or applied as a label. Additionally, GitHub Projects provide automation so that when a code commit satisfies the requirements for a user story, the commit message can use keywords (tenses of *fix*, *resolve*, and *close*) paired with the card number to automatically move the corresponding user story card to the *Done* column on the automated Kanban board[17]. **Figure 1** illustrates an example of user stories on an automated Kanban board using GitHub.

With *user story* and *Git log* artifacts, instructors gain access to quantitative measurements of individuals' contributions. However, Campbell observes that "The more any quantitative social indicator is used for social decision-making, the more subject it will be to corruption pressures and the more apt it will be to distort and corrupt the social processes it is intended to monitor" [11]. As applied to software engineering teams, *Campbell's Law* suggests that the more quantitative measurements are emphasized in assessment, the more likely the team members will be to change their behaviors with the goal of improving the measurements–even if those behaviors are not in the best interest of the software product's development. Consequently, this report also carefully considers ways in which these quantitative measurements can be manipulated as well as their potential unintended consequences.

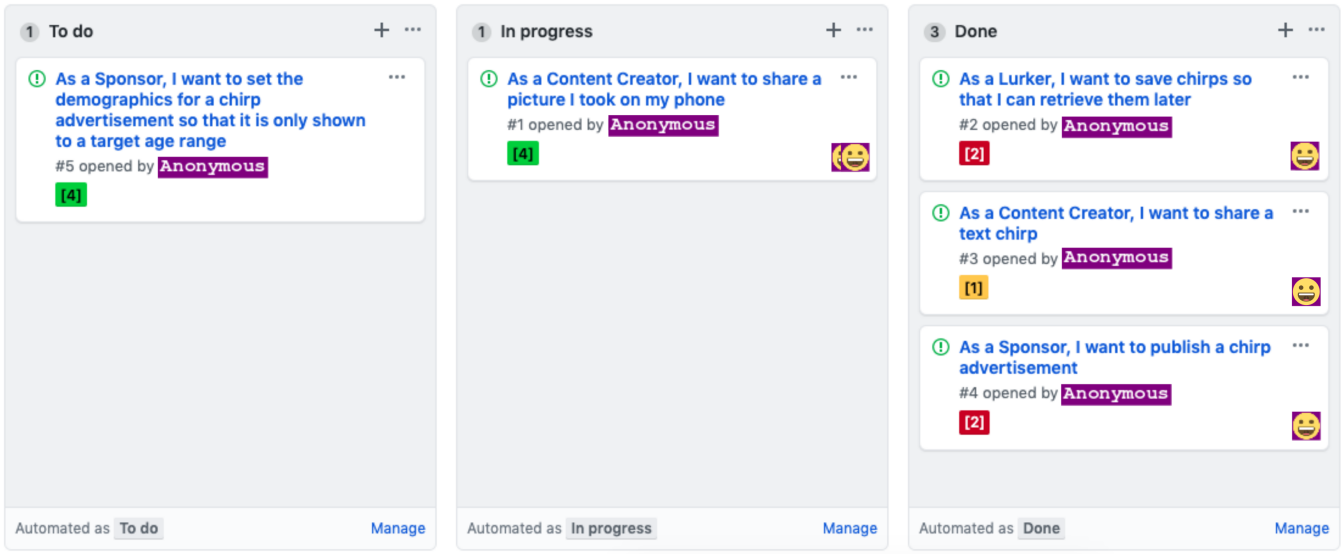

Figure 1: GitHub Projects automated Kanban board featuring user stories along with assigned developers (represented by an avatar) and points (as labels in square brackets)

Sutherland, a founder of *Scrum*, also warns against performance appraisals in industry as "counterproductive" and "demotivating" [32]. Nevertheless, as long as team projects are adopted in software engineering education, there will be an onus on teachers to grade individuals' contributions fairly and objectively. This study explores meta-data from software engineering artifacts to consider their potential for fairness and objectivity in grading.

3 METHOD

In the 2018-2019 academic year, the upper-division Software Engineering course at *California State University, Chico* adopted the *Tech Startup model* [9][10] for small teams (three to five Computer Science undergraduates) to work on semester-long software projects. First, students identified the projects that most interested them as lectures introduced them to Scrum (an Agile framework), collaborative version control with Git and GitHub, and user stories created on GitHub Projects automated Kanban boards. By the third week of the semester, the teams had formed, met with their external clients, and planned their first sprint (two-week Scrum interval).

At the beginning of their first sprint, a graded assignment required each team to have an updated Kanban board that reflected the user stories they planned to accomplish during the sprint. For the remainder of the semester, the course included two hour-per-week lessons on developer operations, design patterns, design and code quality metrics, and unit testing. Periodically, teams were asked to demonstrate their software's working functionality and explain what value it provides to their customers. In coordination with each of the demonstrations (weeks five, ten, and fifteen), each team member completed peer evaluations of each of their teammates using CATME. Peer evaluations were anonymous but each student received aggregate scores of their *interaction* and *contribution* ratings the following week. In addition, students had two hour-per-week lab periods that were dedicated to project teamwork.

Team projects accounted for sixty percent of each student's final marks. The projects were evaluated based on a final presentation and deliverable to demonstrate the project's *usefulness* (30%, including demonstrated adoption of Agile principles to adapt to changing requirements), *software design* (35%), and *testing* (35%). The instructor adjusted individuals' scores as judged by their professionalism and perceived contributions. Neither the peer evaluation scores nor any of the other metrics discussed in this paper directly affected students' project scores, although the instructor had observed team interactions and had access to both the CATME ratings and the teams' Kanban boards.

To consider students' technical skills as a potential confound in our analysis, I gathered each student's scores on four quizzes completed during the semester (weighted 18% of their final marks). The quizzes were all individual assignments completely separated from the team projects and evaluated students on their ability to apply technical development skills taught in class.

I gathered data for analysis from all ten teams (n=42) after the conclusion of the semester and the posting of final marks. I collected each student's quiz and project grades from the gradebook and merged those scores in a spreadsheet with each student's average *interaction* and *contribution* CATME ratings.

To gather data about students' code commits, I cloned the each team's Git repository and executed a python script that extracted each commit's meta-data from the Git log and aggregated them into a spreadsheet [8]. Finally, I manually reviewed each team's Kanban boards and recorded each user story in the *Done* column, along with its points and student(s) assigned. It should be noted that Kanban board data is not downloaded when the repository is cloned and there is no current feature for exporting GitHub Project Kanban boards. However, while I manually collected user story data for this study, a web scraper could automate the process.

4 EXPLORATORY ANALYSIS

I first analyzed the data to observe overall individual and team trends. Total commits by team varied widely from 39 to 179 (M=90.1, sd=40.7). I also recognized that several teams had very large line changes on their first commits (as many as 276,844 lines inserted). Upon further investigation, I found that these initial large commits represented only files from web or mobile frameworks (e.g. Android Studio mobile application projects) and not from student-written code. Since these commits would drastically distort the counts of line changes by individuals, I excluded them from analysis.

I also found wide variance between teams in how many stories (M=14.0, sd=6.5) and the respective number of total points completed (M=23.20, sd=25.65). Unfortunately, two teams did not record which team member was assigned to every story on the Kanban board and third team only had one member assigned to stories (and the rest of the stories had no one listed, although "Done"). All other teams had at least one team member assigned to each completed user story, along with points assigned to stories.

Comparing points between teams would be misleading because each team separately assigned points, only relative to *their team's* other stories. Similarly, different teams used different programming languages, frameworks, and tools so comparing line changes across teams would be inappropriate. Consequently, I adjusted each of the performance metrics for individual students to compare relative to their team. An individual's **relative commit share** was calculated by dividing their total commits by the expected number of commits. Since number of team members also varied between teams, the expected number of commits was calculated as the total number of team commits divided by number of team members.

Likewise, an individual's **relative change share** was calculated by the total number of line changes in their commits divided by the number of expected line changes–the total number of team line changes divided by the number of team members. For both measures, if each team member contributed exactly the same number of commits and line changes, each of their commit share and change share scores would be 1. **Figure 2** shows the distribution of individuals' relative commit shares across the ten teams and **Figure 3** shows a corresponding distribution for the teams' relative line changes.

Some teams (such as the fourth and ninth teams from the left) demonstrated a mean close to 1 for both scores. However, both figures show outliers (round points in the box plots) who contributed considerably more or less than what would be expected as their fair share. In an extreme case, we found one team (far right on Figure 3) where one student contributed nearly five times the expected line change for their team while the rest of the team contributed close to no line changes.

Similarly, I investigated the distributions of individuals' contributions within their teams, as indicated by their user stories and points completed. Individuals' **relative story share** were calculated relative to the total number of team user stories, divided by the number of team members. However, I found that teams commonly assigned more than one person to a user story so it is possible for an entire team to earn relative story shares above 1, as illustrated by two teams (second and third from left) in **Figure 4**.

Individuals' **relative point share** was calculated relative to the total number of points the team completed, divided by number of

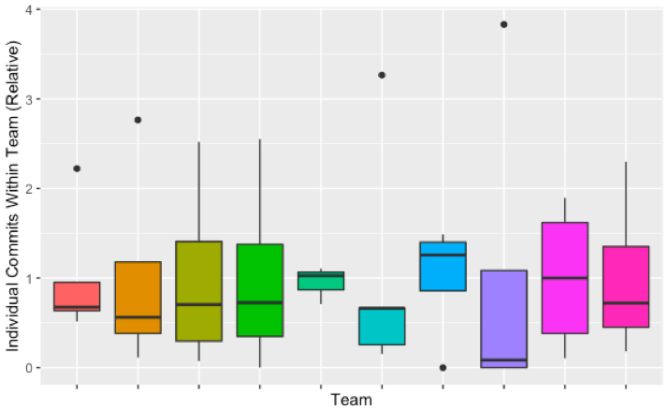

Figure 2: Distributions of individuals' commits, relative to their teams

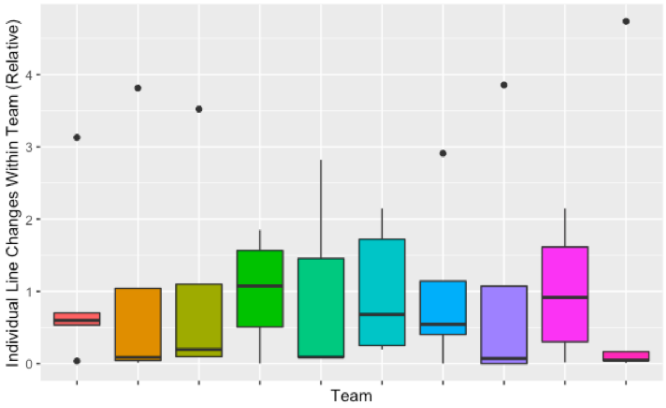

Figure 3: Distributions of individuals' line changes, relative to their teams

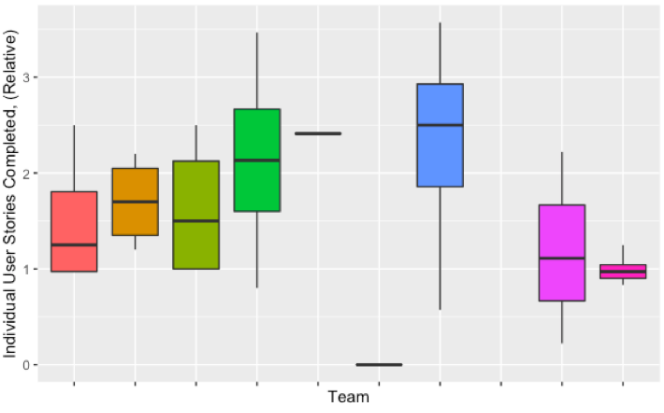

Figure 4: Distributions of individuals' stories completed, relative to their teams

team members. However, as Figure 4 shows, many stories were assigned to multiple students. In those cases, the points for a story were divided evenly among the assigned students. For example, a three-point story with two assigned contributors earned 1.5 points for each contributor. **Figure 5** shows the relative point share distribution with each team. It should be noted that teams are listed in the same order with corresponding colors for **Figures 2-5**. Teams with incomplete (or completely absent) story assignments have incomplete box plots or empty columns in **Figures 4-5**.

Opposed to the quantitative performance metrics, peer evaluation, project, and quiz scores do not have to be adjusted to account for differences between teams because each belongs on an absolute scale. **Interaction** and **contribution** ratings ranged from 1 (poorest) to 5 (best), while quizzes were graded on an 100 point scale. Overall, students received mostly positive ratings for their **interactions** (M=4.31, sd=0.43) and **contributions** (M=4.14, sd=0.59). Similarly, the students received high **instructor assessment** scores (M=90.02, sd=5.43). Students' quiz scores were lower (M=81.22, sd=9.03).

4.1 Relationships between Assessments

Individual students' relative shares of commits, line changes, stories, and points should all indicate the degree of contributions they made to their respective teams. Consequently, I explored possible correlations with students' contribution ratings. I used Spearman's rank-order correlation and lowered the critical value for statistical significance (α=0.0125) using the Bonferonni correction because of increased likelihood of observing correlations with multiple comparisons.

I found that **relative commit shares** (M=1, sd=0.95) had a moderately positive correlation (ρ=0.42, p<.0125) with **contribution ratings** (M=4.14, sd=0.59). However, contribution ratings were not significantly correlated with line change share (M=1, sd=1.32, ρ=0.28, p=.08), story share (M=1.62, sd=0.88, ρ=0.26, p=.15), nor point share (M=0.92, sd=0.45, ρ=0.17, p=.35).

To investigate predictive value in the relative shares of commits, line changes, stories, and points, I modeled a multiple linear regression to explain the peer-evaluated contribution rating. The initial model also included students' quiz averages as a measurement–external to the team projects–that could account for students' technical skills. However, the multiple linear regression found that only the relative commit share was a significant predictor (p<.05) of contribution ratings. After removing the non-significant factors, a simple linear regression found a significant regression equation (F(1,40)=11.05, p<.01), with an R^2 of 0.22. Students' predicted **peer evaluation contribution rating** is 3.858 + 0.287*__relative commit share__.

Similarly, I investigated possible relationships between the quantitative performance metrics and the **instructor assessment** for each student. Using the Spearman correlation with adjusted critical value (α=0.0125), we found that only **relative story share** had a significant, moderately positive correlation with instructor assessment (ρ=0.51, p<.0125). Commits (ρ=0.21, p=.18), line changes (ρ=0.21, p=.19), and points (ρ=0.37, p=.04) were not significantly correlated with instructor assessment. We modeled a multiple linear regression to explain the instructor assessments for students' projects considering the quantitative performance metrics, peer evaluation ratings, and the quiz scores. Only the peer evaluation **contribution** rating was a significant predictor (p<.05) and after excluding non-significant variables, a simple linear regression found a significant regression equation (F(1,40)=41.08, p<.01) with R^2 of 0.507. Predicted **instructor assessment** is 62.872 + 6.552*__contribution rating__.

Both peer and instructor assessments of a student's project contributions could be susceptible to the Halo Effect. I investigated a possible relationship between peer evaluation **contribution** and **interaction** ratings as well as between **instructor assessment** and **quizzes** (also graded by the instructor). Spearman's correlation found a significant, strong, positive correlation (ρ=0.75, p<.001) between peer evaluation interaction and contribution ratings. **Figure 6** shows the strong correlation between the two ratings. There was also a significant, moderately weak correlation (ρ=0.38, p<.05) between instructor assessment of the students' projects and quizzes.

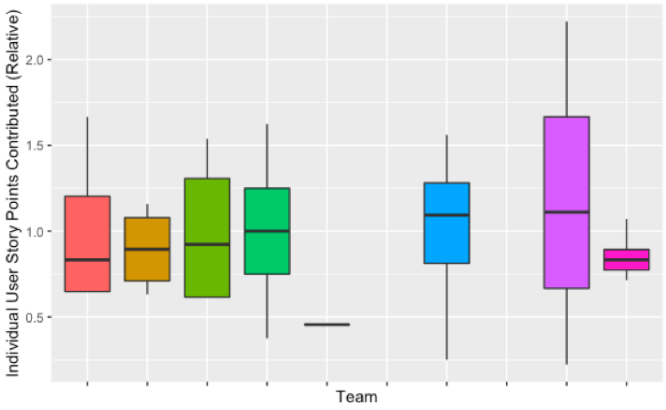

Figure 5: Distributions of individuals' points completed, relative to their teams

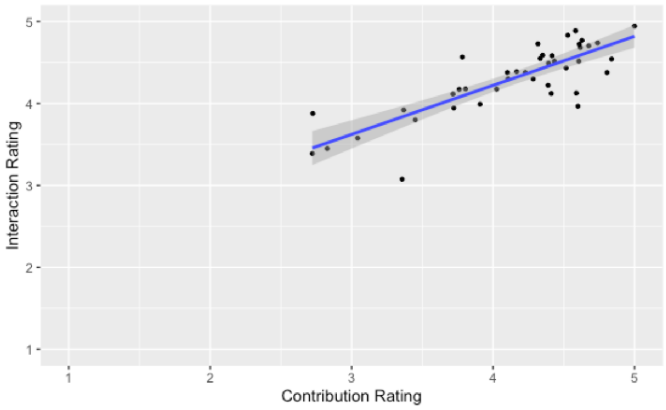

Figure 6: Strong, positive correlation between peer evaluations with regression line and shaded confidence interval

5 DISCUSSION

The strong, positive correlation between peer evaluation ratings for students' **contributions** and **interactions** exposes a likely Halo Effect that influences how teammates rate each other. Anecdotally, the instructor observed that all of the students communicated respectfully and acted with a degree of professionalism consistent with unanimous interaction peer evaluation ratings above 3.0.

On the other hand, only three students received average ratings below neutral (3) from their peers for their contributions. To the contrary, several teams showed wide disparities between their team members' concrete, quantitative contributions. For example, 16 students (38%) committed less than half of their expected commit share (<0.5, total team commits divided by number of team members) and 21 students (50%) made less than half of their expected line change share (<0.5, total line changes divided by number of team members).

While it is possible to make contributions to a team without committing code–such as eliciting feedback from customers and managing the Kanban board–it should also be noted that teams' documentation and testing were also committed to their repositories (and those line changes also counted). There is a clear inconsistency between peer evaluations and quantitative evidence of actual contributions. The inconsistency may actually reveal a strength in the objectivity of quantitative measurements when peer evaluations can be influenced by personal biases.

Nevertheless, there are no perfect measurements of the *value* of contributions. There are some behaviors that can skew counts of commits, line changes, user stories, and points. Foremost, git only records a single author for each commit (and its associated line changes). Effective teams are collaborative, either formally through practices like pair programming, or informally by helping each other debug or review code. Therefore, some meaningful contributions are not reflected in Git log meta-data.

On the other hand, Kanban boards reflect which team members assigned themselves to contribute to each user story. In these cases, aforementioned "hidden" contributions should be reflected in individuals' user stories. Moreover, user story points represent an estimation of how difficult it is to develop. While the value of two different lines of code cannot be assumed to be equivalent, points can provide insight into *quality* of contributions rather than just the *quantity* or volume of code contributions.

In this experience report, none of the quantitative measurements were immediately accessible nor were they directly calculated as part of students' project scores. It may be safe to assume that students did not try to manipulate the measurements. However, if these measurements were to be included in assessment–and especially if that is transparent to the students–*Campbell's Law* [11] leads us to predict that they might change their behaviors to manipulate the measurements in their favor. Students guiding their behavior to satisfy measurements of code volume could lead to competing interests with software quality. For example, incentivizing high line changes could influence students to purposely write less elegant and less maintainable code.

Students could also manipulate their user story assignments and points to reflect their desired grade outcome rather than using the Kanban board to accurately reflect the team's planning and progress. However, as long as user stories and points contributed are calculated relative to the individuals within the same team, those measurements should be less prone to manipulation by single team members. The Kanban board is maintained by the whole team and calculating contributions is a zero-sum game: if one student increased the points on only the user stories to which they were assigned, the relative point share of the other students would decrease. Consequently, teams might be able to self-regulate and protect the Kanban board from manipulation. Additionally, if teams make use of the Kanban automation feature (which links a "Done" user story with the commit that resolved the feature), instructors can manually verify the accuracy of the user stories and review their associated code contributions.

6 CONCLUSIONS

Assessing individual contributions to unique team projects with fairness and objectivity is difficult. This experience report examined ten Software Engineering teams (n=42) and explored both subjective and quantitative assessments of individual team members. I found that peer evaluations of interactions and contributions (using CATME [20]) were strongly correlated and at odds with concrete evidence of individuals' contributions, suggesting likelihood of Halo Effect bias [30].

A regression model revealed that peer evaluation contribution ratings were significant predictors of the corresponding student's instructor assessments. While consistency between the peer and instructor assessments of individual contributors may be reassuring, it may also indicate subjectivity of both ratings as well as potential influence of peer evaluations on the instructor's evaluations. However, the study only observed a single instructor's evaluations, which is a threat to the validity of the results.

Meanwhile, Software Engineering artifacts may provide more objective evidence for individual contributions. I investigated quantitative measurements of code contributions (Git commits) and line changes (insertions and deletions), as well as the number of user stories completed and their respective point values. The number of a student's commits–relative to total team commits–is associated with peer evaluations of their contributions. However, the other measurements were not significant predictors of either peer evaluations or instructor assessments. The lack of strong associations between quantitative and qualitative assessments may indicate inconsistencies in evaluating value of contributions; alternatively, it may demonstrate the objectivity of quantitative metrics that are immune to subjective biases. Future work is necessary to evaluate assessments independently and should also consider the consistency of contribution assessments over time.

This paper introduces quantitative metrics that may potentially provide more objective assessments of individual students' contributions to team projects. However, there are limitations of these measurements and students may stray from professional practices to manipulate their grades if calculated directly from the commits, line changes, user stories, and points. For the time being, I recommend that instructors gather evidence from these quantitative metrics to supplement and validate their traditional, subjective assessments.

REFERENCES

[1] Howard Abikoff, Mary Courtney, William E. Pelham, and Harold S. Koplewicz. 1993. Teachers' ratings of disruptive behaviors: The influence of halo effects. *Journal of Abnormal Child Psychology* 21, 5 (01 Oct 1993), 519–533. https://doi.org/10.1007/BF00916317

[2] ACM. 2013. Computer Science 2013: Curriculum Guidelines for Undergraduate Programs in Computer Science. http://www.acm.org/education/curricula-recommendations

[3] Francesca Arcelli Fontana and Claudia Raibulet. 2017. Students' Feedback in Using GitHub in a Project Development for a Software Engineering Course. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, USA, 380–380. https://doi.org/10.1145/3059009.3072984

[4] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. 2001. Agile Manifesto. http://agilemanifesto.org/

[5] Grant Braught, John Maccormick, James Bowring, Quinn Burke, Barbara Cutler, David Goldschmidt, Mukkai Krishnamoorthy, Wesley Turner, Steven Huss-Lederman, Bonnie Mackellar, and Allen Tucker. 2018. A Multi-Institutional Perspective on H/FOSS Projects in the Computing Curriculum. *ACM Trans. Comput. Educ.* 18, 2, Article 7 (July 2018), 31 pages. https://doi.org/10.1145/3145476

[6] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. 2015. Software Engineering Project Courses with Industrial Clients. *Trans. Comput. Educ.* 15, 4, Article 17 (Dec. 2015), 31 pages. https://doi.org/10.1145/2732155

[7] Kevin Buffardi. 2018. Tech Startup Learning Activities: A Formative Evaluation. In *IEEE/ACM International Workshop on Software Engineering Education for Millennials (SEEM)*. 24–31.

[8] Kevin Buffardi. 2019. gitlog2csv. https://github.com/kbuffardi/gitlog2csv

[9] Kevin Buffardi, Colleen Robb, and David Rahn. 2017. Learning Agile with Tech Startup Software Engineering Projects. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, USA, 28–33. https://doi.org/10.1145/3059009.3059063

[10] Kevin Buffardi, Colleen Robb, and David Rahn. 2017. Tech Startups: Realistic Software Engineering Projects with Interdisciplinary Collaboration. *J. Comput. Sci. Coll.* 32, 4 (April 2017), 93–98. http://dl.acm.org/citation.cfm?id=3055338.3055355

[11] Donald T. Campbell. 1979. Assessing the impact of planned social change. *Evaluation and Program Planning* 2, 1 (1979), 67 – 90. https://doi.org/10.1016/0149-7189(79)90048-X

[12] Nicole Clark. 2005. Evaluating Student Teams Developing Unique Industry Projects. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42 (ACE '05)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 21–30. http://dl.acm.org/citation.cfm?id=1082424.1082428

[13] N. M. Devadiga. 2017. Software Engineering Education: Converging with the Startup Industry. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T)*. 192–196. https://doi.org/10.1109/CSEET.2017.38

[14] Heidi J.C. Ellis and Gregory W. Hislop. 2016. Pathways to Student Learning Within HFOSS. In *Proceedings of the 17th Annual Conference on Information Technology Education (SIGITE '16)*. ACM, New York, NY, USA, 168–168. https://doi.org/10.1145/2978192.2978242

[15] F. Fagerholm and A. Vihavainen. 2013. Peer assessment in experiential learning Assessing tacit and explicit skills in agile software engineering capstone projects. In *2013 IEEE Frontiers in Education Conference (FIE)*. 1723–1729. https://doi.org/10.1109/FIE.2013.6685132

[16] Vahid Garousi, Kai Petersen, and Baris Ozkan. 2016. Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review. *Information and Software Technology* 79 (2016), 106 – 127. https://doi.org/10.1016/j.infsof.2016.07.006

[17] GitHub. 2019. The world's leading software development platform - GitHub. https://github.com/ Accessed on August 2019.

[18] Nicole Herbert. 2007. Quantitative Peer Assessment: Can Students Be Objective?. In *Proceedings of the Ninth Australasian Conference on Computing Education - Volume 66 (ACE '07)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 63–71. http://dl.acm.org/citation.cfm?id=1273672.1273680

[19] Bibb Latane, Kipling Williams, and Stephen Harkins. 1979. Many hands make light the work: The causes and consequences of social loafing. *Journal of Personality and Social Psychology* 37 (1979), 822–832. Issue 6. https://doi.org/10.1037/0022-3514.37.6.822

[20] Richard A. Layton, Matthew W. Ohland, and Hal Pomeranz. 2007. Software For Student Team Formation And Peer Evaluation: Catme Incorporates Team Maker.

[21] Robert C. Martin. 2006. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall.

[22] Nancy R. Mead. 2015. Industry/University Collaboration in Software Engineering Education: Refreshing and Retuning Our Strategies. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2 (ICSE '15)*. IEEE Press, Piscataway, NJ, USA, 273–275. http://dl.acm.org/citation.cfm?id=2819009.2819050

[23] Christian Murphy, Kevin Buffardi, Josh Dehlinger, Lynn Lambert, and Nanette Veilleux. 2017. Community Engagement with Free and Open Source Software. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 669–670. https://doi.org/10.1145/3017680.3017682

[24] Natalja Nikitina, Mira Kajko-Mattsson, and Magnus Ståle. 2012. From Scrum to Scrumban: A Case Study of a Process Transition. In *Proceedings of the International Conference on Software and System Process (ICSSP '12)*. IEEE Press, Piscataway, NJ, USA, 140–149. http://dl.acm.org/citation.cfm?id=2664360.2664382

[25] Matthew Ohland, Misty Loughry, Rufus L. Carter, Lisa Bullard, Richard Felder, Cynthia Finelli, Richard Layton, and Douglas Schmucker. 2005. Developing a Peer Evaluation Instrument that is Simple, Reliable, and Valid. *Proceedings 2005 American Society of Engineering Education Conference and Exposition Portland Oregon* (01 2005).

[26] Matthew W. Ohland, Misty L. Loughry, David J. Woehr, Lisa G. Bullard, Richard M. Felder, Cynthia J. Finelli, Richard A. Layton, Hal R. Pomeranz, and Douglas G. Schmucker. 2012. The Comprehensive Assessment of Team Member Effectiveness: Development of a Behaviorally Anchored Rating Scale for Self- and Peer Evaluation. *Academy of Management Learning & Education* 11, 4 (2012), 609–630. https://doi.org/10.5465/amle.2010.0177

[27] Helen Parker and Mike Holcombe. 1999. Campus-based Industrial Software Projects: Risks and Rewards. *SIGCSE Bull.* 31, 3 (June 1999), 189–. https://doi.org/10.1145/384267.305935

[28] Gustavo Pinto, Clarice Ferreira, Cleice Souza, Igor Steinmacher, and Paulo Meirelles. 2019. Training Software Engineers Using Open-source Software: The Students' Perspective. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '19)*. IEEE Press, Piscataway, NJ, USA, 147–157. https://doi.org/10.1109/ICSE-SEET.2019.00024

[29] Arnold Rosenbloom, Sadia Sharmin, and Andrew Wang. 2017. GIT: Pedagogy, Use and Administration in Undergraduate CS. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, USA, 82–83. https://doi.org/10.1145/3059009.3072980

[30] Frank W. Schneider, Jamie A. Gruman, and Larry M. Coutts. 2012. *Applied Social Psychology: Understanding and Addressing Social and Practical Problems*. SAGE.

[31] Therese Mary Smith, Robert McCartney, Swapna S. Gokhale, and Lisa C. Kaczmarczyk. 2014. Selecting Open Source Software Projects to Teach Software Engineering. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 397–402. https://doi.org/10.1145/2538862.2538932

[32] Jeff Sutherland. 2018. Impediments of Performance Appraisals. https://www.scruminc.com/performance-appraisals/ Accessed November 2019.

[33] Anya Tafliovich, Andrew Petersen, and Jennifer Campbell. 2015. On the Evaluation of Student Team Software Development Projects. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 494–499. https://doi.org/10.1145/2676723.2677223

[34] Linda van den Bergh, Eddie Denessen, Lisette Hornstra, Marinus Voeten, and Rob W. Holland. 2010. The Implicit Prejudiced Attitudes of Teachers: Relations to Teacher Expectations and the Ethnic Achievement Gap. *American Educational Research Journal* 47, 2 (2010), 497–527. https://doi.org/10.3102/0002831209353594 arXiv:https://doi.org/10.3102/0002831209353594