Combining GitHub, Chat, and Peer Evaluation Data to Assess Individual Contributions to Team Software Development Projects

CHRISTOPHER HUNDHAUSEN, Oregon State University
PHILL CONRAD, University of California, Santa Barbara
OLUSOLA ADESOPE, Washington State University
AHSUN TARIQ, Oregon State University

Assessing team software development projects is notoriously difficult and typically based on subjective metrics. To help make assessments more rigorous, we conducted an empirical study to explore relationships between subjective metrics based on peer and instructor assessments, and objective metrics based on GitHub and chat data. We studied 23 undergraduate software teams (n = 117 students) from two undergraduate computing courses at two North American research universities. We collected data on teams' (a) commits and issues from their GitHub code repositories, (b) chat messages from their Slack and Microsoft Teams channels, (c) peer evaluation ratings from the CATME peer evaluation system, and (d) individual assignment grades from the courses. We derived metrics from (a) and (b) to measure both individual team members' *contributions* to the team, and the *equality* of team members' contributions. We then performed Pearson analyses to identify correlations among the metrics, peer evaluation ratings, and individual grades. We found significant positive correlations between team members' GitHub contributions, chat contributions, and peer evaluation ratings. In addition, the equality of teams' GitHub contributions was positively correlated with teams' average peer evaluation ratings and negatively correlated with the variance in those ratings. However, no such positive correlations were detected between the equality of teams' chat contributions and their peer evaluation ratings. Our study extends previous research results by providing evidence that (a) team members' chat contributions, like their GitHub contributions, are positively correlated with their peer evaluation ratings; (b) team members' chat contributions are positively correlated with their GitHub contributions; and (c) the equality of team' GitHub contributions is positively correlated with their peer evaluation ratings. These results lend further support to the idea that combining objective and subjective metrics can make the assessment of team software projects more comprehensive and rigorous.

CCS Concepts: • **Software and its engineering-Programming teams**; • **Social and professional topics-Software engineering education**;

Additional Key Words and Phrases: Software engineering education, collaborative software development, assessment, peer evaluation, Covid-19, online chat communication, GitHub, Slack, Microsoft Teams, CATME

This work was supported by the National Science Foundation under grants 1915196 and 1915198.
Authors' addresses: C. Hundhausen (corresponding author) and A. Tariq, 2113 Kelley Engineering Center, Oregon State University, Corvallis, OR 97331; emails: {chris.hundhausen, tariqa}@oregonstate.edu; P. Conrad, University of California, Santa Barbara, Santa Barbara, CA; email: phtcon@ucsb.edu; O. Adesope, Washington State University, Pullman, WA; email: olusola.adesope@wsu.edu.

1946-6226/2023/07-ART33 $15.00
https://doi.org/10.1145/3593592

ACM Reference format:
Christopher Hundhausen, Phill Conrad, Olusola Adesope, and Ahsun Tariq. 2023. Combining GitHub, Chat, and Peer Evaluation Data to Assess Individual Contributions to Team Software Development Projects. *ACM Trans. Comput. Educ.* 23, 3, Article 33 (July 2023), 23 pages.
https://doi.org/10.1145/3593592

Author's Note:

- This paper uses data that was also used as the basis of two previous publications. This paper differs from [28] in that we broaden the analysis in two respects. First, we include in our analysis a new set of data on teams' online team discussions. Second, whereas [28] considers only individual metrics of contribution, this paper considers a team metric of the equality of team members' contributions. This paper differs from [27] in that, whereas [27] performs a content analysis on teams' online discussions for evidence of reflection, this paper considers only the quantity of posts, words, and characters in teams' online discussions.
- The study was conducted while the first author was a faculty member at Washington State University; he is now a facuty member at Oregon State University.
- The corresponding author may be reached at chris.hundhausen@oregonstate.edu.

1 INTRODUCTION

Team projects are the centerpiece of undergraduate capstone design courses (e.g., [24, 41]).They are also featured in a variety of upper- and lower-division undergraduate computing courses that involve software development, ranging from web development and software engineering to security and architecture (e.g., [50]). Team software development projects are widely regarded to be educationally valuable because they provide students with authentic experiences that align with real-world practice.

Despite their value in undergraduate computing education, team software development projects prove difficult to assess, both at the individual and team level. In this article, we evaluate how the use of data from a version control system (GitHub) and from online communications tools (e.g., Slack, Microsoft Teams) may be helpful in these efforts.

1.1 Research Problem and Questions

In academic team-based software projects, assessing individual contributions is crucial for at least two reasons: (a) students need to be assigned individual grades, and (b) instructors and students need to assess the extent to which team members are contributing equitably to the team's efforts, so that adjustments can be made to improve the functioning of the team.

Although previous work [7, 28] indicates that objective metrics derived from students' activities in GitHub can be combined with subjective instructor and peer evaluations to provide a clearer picture of students' individual contributions to team software development projects, they also raise follow-up research questions about the assessment of team projects. One such question relates to the fact that software teams frequently collaborate through online chat channels in, for example, Slack or Microsoft Teams:

> *RQ1: How do students' individual contributions to their teams' online chats correlate with the subjective and objective metrics considered in prior work?*

A second question relates to the need to assess the *equality* of individual contributions to team software development projects—that is, the extent to which team members are contributing equally to their teams' efforts:

RQ2: How does the equality of individual contributions within a team correlate with the subjective metrics considered in prior work?

1.2 Aims, Objectives, and Hypotheses

This article builds on prior work by presenting an expanded analysis of data collected in a previous study of student software project teams enrolled in undergraduate courses at North American research universities [28]. To address RQ1, we integrate counts of student chat channel contributions into our prior correlational analysis. To address RQ2, we leverage the Gini Index [11, 40], a measure of income inequality that has been previously applied to gauge contribution inequality in collaborative software engineering [49]. We use the Gini Index to assess the inequality of individual GitHub and chat contributions within team software development projects. We then determine whether this inequality measure correlates with peer assessments of team contributions. Our overall aim is twofold: first, to help make the assessment of team software development projects more comprehensive and rigorous by applying a range of subjective and objective measures, and second, to provide computing educators with practical guidance on how to assess team projects through a range of subjective and objective measures.

We explore the following hypothesis regarding the relationships between students' chat contributions, code contributions, and peer evaluation ratings:

H1: Measures of students' chat channel contributions are positively correlated with measures of code contributions and peer evaluation ratings.

H1 states that the quantity of a student's chat messages is positively associated with the quantity of the student's code contributions, as well as the magnitude of the student's peer evaluation ratings. Contributing to team chats is one way a student can demonstrate active participation in a team project. It seems reasonable that the contributions of a student who makes more chat contributions will be more positively regarded by others on the team, provided those chat contributions are meaningful to team goals. A positive relationship between chat contributions and code contributions can be justified by the observation that a student who is writing code is engaged in the project and therefore is more inclined to chat about the project. Likewise, a student who makes posts to the chat is engaged in the project and therefore is more likely to be contributing code.

With respect to the equality of team members' chat and code contributions, we propose the following two hypotheses based on the Gini Index [11, 40]—a value between 0 and 1, where 0 represents perfect equality (everyone contributes equally) and 1 represents perfect inequality (only one team member contributes):

H2: Teams' Gini Index values for code and chat channel contributions are negatively correlated with their average peer evaluation ratings.

H3: Teams' Gini Index values for code and chat channel contributions are positively correlated with the variance in their peer evaluation ratings.

H2 states that teams with higher inequality in their individual members' contributions (i.e., Gini Index values approaching 1) tend to rate their teammates lower in peer evaluations. This hypothesis follows from the intuition that team members will tend to be less satisfied with their teammates' contributions when they perceive inequality in their teammates' contributions.

H3 states that members of teams with higher inequality in individual contributions tend to rate team members' contributions more variably. To the extent that team members are aware of the equality of their contributions and rate their teammates' contributions honestly, this hypothesis follows logically: higher inequality of contributions will lead to higher variably in peer evaluation ratings. We acknowledge, however, that peer ratings are always subject to various forms of bias and thus may not perfectly mirror teammates' actual contributions.

1.3 Contributions

In presenting an analysis of relationships between individual code and chat contributions, team contribution equality, and peer evaluations, this article makes three key contributions to computing education research:

(1) It expands on previous research that identified a positive relationship between team members' code contributions and peer evaluation ratings by furnishing evidence that software team members' chat contributions are also positively correlated with their peer evaluation ratings.
(2) It furnishes evidence that software team members' chat contributions are positively correlated with their GitHub contributions.
(3) It furnishes evidence that the equality of a team's GitHub and chat contributions (as measured by the Gini Index) is positively correlated with its average peer evaluation rating and negatively correlated with the variance in team members' peer evaluation ratings.

Taken together, these results lend support to the idea that a mix of objective and subjective metrics can be fruitfully combined to make the assessment of team software projects in computing courses more comprehensive and rigorous.

2 RELATED WORK

Studies of professional software developers indicate that, in the real world, developers most often work on team-based legacy code projects in which software is written over many years by multiple developers, some of whom may no longer be part of the team. In such projects, so-called soft skills in information literacy, metacognition, collaboration, coordination, and communication prove just as valuable as technical programming skills (e.g., see [4, 13, 18, 25, 53]). By participating in team software development projects that align with projects they will encounter in the professional world, students can obtain valuable learning opportunities that help prepare them for careers in the profession.

2.1 The Skills Gap

In their seminal study, Begel and Simon [4, 5] followed eight new hires at Microsoft, concluding that although their university educations had prepared them with adequate technical skills, their "communication, collaboration, and orientation skills are not as well addressed." In a similar vein, a systematic literature review by Radermacher and Walia [48] found that novice developers were lacking in both technical areas, and the so-called "soft skills" areas: communication and teamwork. Craig et al. [13] performed semi-structured interviews with 20 early career developers; they found that the gap persisted, and characterized it as a mismatch between students experiences in school vs. industry along six axes. This study also found significant gaps in the teamwork and communications aspects of students' preparation. In these and other studies (e.g., [9, 18, 44]), a common theme is that when it comes to promoting relevant authentic skills in general, and communication and teamwork skills in particular, undergraduate software engineering education has considerable room for improvement.

2.2 Implementing Team Software Development Projects

One way computing educators have attempted to address the skills gap is by engaging students in authentic software development projects that mirror the projects that students will encounter on the job. Although such projects are most often featured in capstone design courses [24, 41], they also appear in a variety of upper- and lower-division computing courses that involve software development [50]. To promote a greater degree of authenticity, computing instructors have engaged students in team projects with real clients [6, 29, 59], and with free and open source development communities [30, 36, 38, 42, 54, 55]. Our work also aims to make team software development projects more authentic by engaging teams with legacy code bases; however, rather than coming from development activities outside of the courses studied, the legacy code bases are developed by students and teaching personnel in prior offerings of the course.

2.3 Assessing Team Software Development Projects

Because they are undertaken in teams and encompass a broad range of activities and artifacts, team software development projects prove notoriously difficult to assess. There are at least two dimensions to the assessment problem: the difficulty of evaluating individual contributions to team projects fairly, and the difficulty of evaluating one team's work against another team's work when the projects they are working on may be entirely different [1, 12, 16, 19, 57, 61]. Our work addresses the first dimension of this problem by exploring objective metrics that can make assessments more rigorous and reliable.

In computing education, instructors have taken a variety of approaches to assigning individual grades to team members in a team-based software development project. One approach is to require individuals to submit their own work, which is assessed by instructors on its own merits. A second approach is to assign the same grade to all members of a team. A third approach is to assign the team a grade as a whole, but to compute individual grades by applying individual multipliers to the team grade. Such individual multipliers are typically derived from a peer evaluation survey (e.g., CATME [37]) in which team members assess their own and their teammates' contributions to the project.

A disadvantage of the preceding approaches is that they rely exclusively on the subjective judgments of instructors and students. In an attempt to make assessment more objective, computing educators have proposed approaches rooted in data harvested from shared code repositories [7, 22, 43, 47]. In an empirical study on which this work directly builds, Buffardi [7] proposed a series of metrics, derived from GitHub commit and issue logs, that account for the relative share of commits, line changes, and story points contributed by each team member. To determine whether these objective measures squared with the subjective metrics traditionally used to assess team projects (peer evaluations and instructor grades), Buffardi performed a correlational analysis of the objective and subjective metrics in his own software engineering course (n = 41 students and 10 teams), finding few correlations between the two. A larger-scale replication study [28] identified a larger set of statistically significant correlations between Buffardi's GitHub metrics and subjective metrics derived from CATME peer evaluations, thus providing more robust evidence that objective metrics can be used to reinforce traditional subjective measures of individual contributions to team software projects. The study presented in this article builds directly on this line of work by expanding the correlational analysis to include metrics of teams' chat activities.

2.4 Using Software Repository Data to Investigate Software Development Processes

A related line of research in empirical software engineering has used data from software repositories (e.g., GitHub) to study team software development. For example, software repository data has been used to

- understand factors that influence collaboration between geographically distributed software developers [23];
- investigate factors that affect pull request submission time [62];
- compute the relative importance of software development cycle events, including issues, pull requests, and code reviews [34];
- investigate the impact of project board interactions on collaborative software development [46]; and
- determine how open source software development activities (code contributions, mailing list posts, and bug report submissions) are distributed across contributors to the project [21].

Our work is similar to this line of work in its use of GitHub log data on commits, line changes, and issues to better understand collaborative software development.

2.5 Using Team Chat Data to Understand Team Software Development

A large body of research on software team communication has attempted to identify communication practices that improve team efficiency and productivity [14]. To that end, online communication tools have been found to help support team productivity by keeping team communications precise and goal oriented [56].

To investigate teams' online chat communication and how it supports software development, researchers have applied a variety of analysis approaches, including

- *exploratory analysis* of teams' use of online chat through surveys and observations of chat discussions [35, 56];
- *content analysis* to identify themes and topics of messages [3, 10]; and
- *quantitative analysis* to understand who is communicating, how much communication is happening, and how communication relates to other software development activities [32].

The study presented here builds on this line of work by investigating relationships between team members' chat activities and software development activities, along with how those activities influence team members' perceptions of each other.

2.6 Studying Equality in Team Software Development

Our study builds on a line of research focused on the inequality of individual contributions to both professional [21, 52, 60] and academic [7, 22, 43] software projects. To gauge the inequality of individual contributions to large open source software projects, software engineering researchers [21, 52, 60] have applied at least three metrics originally proposed by the field of econometrics to measure income inequality: the *Gini Index* [40], the *Theil Index* [31], and the *Hoover Index* [26]. The Gini and Hoover Indices yield values between 0 (perfect equality) and 1 (perfect inequality), whereas the Theil Index can be made to produce values in this range through a transformation function. In an empirical study of the equality of individual code, mailing list, and bug report contributions to large open source software repositories, Geoeminne and Mens [21] found that the three metrics yielded consistently high values indicating that a small number of people contributed most to these repositories.

In a similar vein, computing education researchers have applied metrics to measure contribution inequality in undergraduate software development projects. Hamer et al. [22] applied the Gini, Hoover, and Theil Indices, as well as *inter-decile* ratios involving the biggest and smallest project contributors, to explore contribution inequality in undergraduate mobile app development and software engineering courses. Nguyen et al. [43] measured contribution equality based on the percentage of a team's total code contributions made by each team member. Buffardi [7] used a *relative*

Table 1. Summary of Courses, Students, and Teams Involved in the Study

Course	Teams Enrolled	Students Enrolled	Teams Consenting	Students Consenting					Tool
				n	M	F	O	Mean Age	
A1	12	66	7	37	29	8	0	19.9	Slack
A2	10	54	2	9	9	0	0	20.7	Slack
A3	12	64	2	11	8	3	0	19.7	Slack
B	12	57	12	57	48	4	5	23.3	MS Teams
Totals	**46**	**243**	**23**	**117**	**97**	**15**	**5**	**20.1**	

n, number of students; *F*, female; *M*, male; *O*, other or prefer not to say.

share metric that provides an indication of an individual's contribution relative to their *expected* contribution, defined as the total team contributions divided by the number of team members.

As in the study of open source software development by Geoeminne and Men [21], a common theme in all these studies of academic software teams is that individual contributions were highly variable and unequal. Drawing on these past studies, our study leverages the relative share metric of Buffardi [7] to gauge individual contributions and the Gini Index [40] to gauge the inequality of individual contributions.

3 METHOD

3.1 Courses and Participants

This study focused on team software development projects in undergraduate computing courses at two universities:

- *Course A* ("Advanced Application Development") was taught by the second author at UC Santa Barbara, a large North American research university. It is a 10-week course taken primarily by second- and third-year undergraduate computer science majors. It focuses on the development of full-stack web applications through a team project lasting the duration of the term. The course emphasizes both technical skills and soft skills related to Agile practices (standups, sprints, Kanban, user stories, acceptance criteria) and GitHub workflows (pull requests, code reviews).
- *Course B* ("Web Development") was taught by the first author at Washington State University. It is a 15-week advanced undergraduate course in full-stack web development. During the first 10 weeks, students learn web programming through lectures, live coding demos, and a series of individual assignments that build upon each other to produce a full-stack web app. In the final 5 weeks, students form teams and apply what they have learned to build a full stack web app of their choice. As part of the team project curriculum, students learn about the same Agile practices and GitHub workflows emphasized in Course A.

Both courses took place during the pandemic and were conducted *completely online*. Class meetings were held synchronously through Zoom. Outside of class meetings, students engaged with online learning materials and communicated via online communication tools both synchronously and asynchronously.

Table 1 presents demographic data on study participants. The study involved three separate offerings of Course A and one offering of Course B. The study was approved by the Institutional Review Boards of each university; students could opt in to the study by signing an informed consent form. For a team's data to be included in the study, all team members had to consent. As shown in Table 1, in the four courses involved in this study, all members of 23 of the 46 teams (n = 117 students) consented to participate.

Table 2. Key Dimensions of Projects by Course

Project Dimension	Course A	Course B
Project Focus	Existing legacy code projects	Self-defined; option to select from recommended projects
Team Formation	Randomly assigned based on CATME survey	Self-selected; unassigned students randomly assigned
Team Size	5–6	3–5
Starting Code	Code base from existing project	Code base from app developed in first part of course
Project Length	3–4 weeks	5 weeks
Number of Sprints	3	4
Collaboration Tools	GitHub, Slack	GitHub, Microsoft Teams

3.2 Materials: Team Project

Table 2 presents information about the team software development projects in each course. In Course A, teams of up to six members were formed through a CATME team-building survey [33] and then randomly assigned to a legacy code project that had been developed by students in previous course offerings. In contrast, students in Course B were given the option of proposing their own web development project or choosing from a list of pre-approved projects. Students were asked to form their own teams of three to five students; students who did not sign up for a team were randomly assigned to teams that had openings available in both courses, and student teams were given starter code—either the code base from a legacy code project (Course A) or a code base from a demo project developed in the first part of the course (Course B).

Teams in both courses undertook projects of 3 to 5 weeks, split up into three to four sprint cycles. Students used GitHub as their source code repository, along with GitHub issues and project boards to monitor and track their development activities. In addition, teams used channels in either Slack (Course A) or Microsoft Teams (Course B) for online chat communications.

Each course utilized these channels slightly differently. In Course B, student teams used their Microsoft Teams channels exclusively for collaboration on the team project; the instructor had access to the channels but posted to them only to answer questions when explicitly tagged by a team member. In contrast, student teams in Course A used their Slack channels both for team collaboration and to respond to discussion prompts in some course assignments.

Student teams in both courses were expected to employ sound agile software engineering practices emphasized in each course, including the use of issues, Kanban (project) boards, feature branches, pull requests, code reviews, automated tests, and retrospectives. However, grading of students and teams differed significantly between the two courses. In Course A, teams were awarded a grade based on the number of story points they completed during the term, with teaching personnel determining the story point values of each team's completed issues. In contrast, teams in Course B were graded based on a structured rubric, which the instructor used to perform a detailed evaluation of teams' GitHub repositories, chat channels, and deployed software. Individual grades were computed from team grades by applying individual multipliers derived from peer evaluation surveys [33] administered after each sprint.

3.3 Data Collection

We collected data on teams' software development activities and individual team member achievement from four different sources:

- Teams' GitHub code repositories
- Teams' online chat channels
- Teams' CATME peer evaluation surveys administered at the end of each sprint cycle
- Team members' grades on individual assignments that were not related to the team project.

From these data sources, we derived the metrics summarized in Table 3. We describe these variables in greater detail next.

Table 3. Data Variables Collected for Each Individual Student in the Study

Source	Variable	Description	Possible Values
GitHub	*#C*	Number of commits	$\mathbb{Z}^+$
GitHub	*#LC*	Number of lines changed	$\mathbb{Z}^+$
GitHub	*#S*	Number of stories	$\mathbb{Z}^+$
Slack, MS Teams	*#M*	Number of messages	$\mathbb{Z}^+$
Slack, MS Teams	*#W*	Number of words	$\mathbb{Z}^+$
Slack, MS Teams	*#Ch*	Number of characters	$\mathbb{Z}^+$
CATME	*Contr*	Average rating of student contribution	$0 \leq \mathbb{R} \leq 5$
CATME	*Inter*	Average rating of student interaction	$0 \leq \mathbb{R} \leq 5$
Course	*Grade*	Individual assignment grade average	$0 \leq \mathbb{R} \leq 100$

3.3.1 GitHub Data. We used the GitHub GraphQL API to extract three types of data from teams' GitHub repositories: *number of commits (#C)*, *number of lines changed (#LC)*, and *number of stories completed (#S)*. A *commit* is an atomic operation that embodies a set of changes made to a set of file(s). The quantity *lines changed* captures the number of line additions and/or deletions made through all of an individual's commits. For this study, we counted only commits and line changes that applied to the `main` (default) branch of a repository. Thus, we included only code changes that were actually incorporated into a team's final software product; all other code changes were ignored. In addition, we excluded initial commits and line changes associated with setting up a repository—that is, commits that added external libraries and frameworks to a repository. We excluded such commits because they added coded not written by students and hence did not reflect true contributions to the code base.

The quantity *number of stories (#S)* denotes the number of GitHub *issues* completed by the student, where an issue represents a task to be completed, such as adding a new feature, testing code, or fixing a bug. We considered an issue to be complete if it was linked to a *pull request* that merged code into a repository's main branch to close the issue.

3.3.2 Chat Data. We exported teams' chat channel messages from Slack (Course A teams) and Microsoft Teams (Course B teams). Using a custom script, we screened and pre-processed the chat messages. Each message was checked for validity (non-empty and unicode characters). URLs were padded with escaping characters to allow future contextual coding of the attachments. In total, the original corpus included 6,811 messages. Only student messages written in the context of the team project were considered. We excluded messages that

- were sent before or after the time window of the team project;
- were sent to or from the teaching staff, or that were prompted by course assignments unrelated to the team project (in Course A) and had negligible impact on the team activity; and
- were sent by students who were incorrectly assigned to a team, or who were sent by someone who dropped the course.

After we applied the preceding exclusion criteria, the corpus was reduced to 4,905 messages.

In our message corpus, we calculated three different quantities:

- The number of chat *messages (#M)* posted by each student
- The number of *words (#W)* in the chat messages posted by each student
- The number of *characters (#Ch)* in the chat messages posted by each student.

We decided to capture these different quantities based on the observation that some students may write shorter, less substantial, and more frequent messages, whereas other students may write

Table 4. Descriptions of CATME Contribution Scale Ratings [58]

Rating	Description
5	• Does more or higher-quality work than expected • Makes important contributions that improve the team's work • Helps teammates who are having difficulty completing their work
4	Demonstrates behaviors described immediately above and below
3	• Completes a fair share of the team's work with acceptable quality • Keeps commitments and completes assignments on time • Helps teammates who are having difficulty when it is easy or important
2	Demonstrates behaviors described immediately above and below
1	• Does not do a fair share of the team's work • Delivers sloppy or incomplete work

longer, more substantial, and less frequent messages. Thus, we reasoned that also capturing the word and character counts of students' messages could provide a fuller picture of students' contributions to their teams' chat channels.

3.3.3 Peer and Instructor Evaluation Data. CATME [45] is a normed and validated survey that is widely used for peer evaluation of teamwork in higher education. Although the CATME survey includes five scales, in this study we used the two scales most relevant to our research questions:

- *Contribution*: The extent to which an individual helps achieve team goals by completing assigned tasks [58] (Table 4)
- *Interaction*: The extent to which an individual interacts with the team in a supportive way [58] (Table 5).

Team members rated each other on the CATME scales at the end of each sprint. The ratings received by each student over *all* sprints were averaged together to compute the CATME ratings analyzed in this study.

3.4 Data Analysis

3.4.1 Measuring a Team Member's Contribution. Adopting the approach of Buffardi [7] to measuring individual contributions relative to team contributions, we derived *relative share* metrics from the GitHub and chat variables shown in Table 3. Table 6 describes these metrics, which have values ranging from 0 (i.e., the team member did not contribute at all) to n_{team} (i.e., the team member was the only person to contribute). A value of 1.0 indicates that a team member contributed their fair (expected) share.

3.4.2 Measuring the Equality of Team Members' Contributions. Based on prior studies of the inequality of individual contributions to team software projects [21, 22], we used the Gini Index [40] to measure the inequality of individual contributions to the team. Theoretically, the Gini Index ranges from 0 (*complete equality*) to 1 (*complete inequality*) based on the number of contributions made by each team member [17, 39]. However, for the team sizes considered in this study (three to

Table 5. Descriptions of CATME Interaction Scale Ratings [58]

Rating	Description
5	• Asks for and shows an interest in teammates' ideas and contributions • Makes sure teammates stay informed and understand each other • Provides encouragement or enthusiasm to the team • Asks teammates for feedback and uses their suggestions to improve
4	Demonstrates behaviors described immediately above and below
3	• Listens to teammates and respects their contributions • Communicates clearly; shares information with teammates • Participates fully in team activities • Respects and responds to feedback from teammates
2	Demonstrates behaviors described immediately above and below
1	• Interrupts, ignores, bosses, or makes fun of teammates • Takes actions that affect teammates without their input • Does not share information; complains, makes excuses, or does not interact with teammates • Is defensive; will not accept help or advice from teammates

Table 6. Relative Share Metrics Derived from Variables Shown in Table 3

Metric	Description	Derivation	Possible Values
RCS	Relative commit share	$\frac{\#C_{individual}}{\#C_{team} \div n_{team}}$	$0 \leq \mathbb{R} \leq n_{team}$
RLCS	Relative line change share	$\frac{\#LC_{individual}}{\#LC_{team} \div n_{team}}$	$0 \leq \mathbb{R} \leq n_{team}$
RSS	Relative story share	$\frac{\#S_{individual}}{\#S_{team} \div n_{team}}$	$0 \leq \mathbb{R} \leq n_{team}$
RMS	Relative message share	$\frac{\#M_{individual}}{\#M_{team} \div n_{team}}$	$0 \leq \mathbb{R} \leq n_{team}$
RWS	Relative word share	$\frac{\#W_{individual}}{\#W_{team} \div n_{team}}$	$0 \leq \mathbb{R} \leq n_{team}$
RChS	Relative character share	$\frac{\#Ch_{individual}}{\#Ch_{team} \div n_{team}}$	$0 \leq \mathbb{R} \leq n_{team}$

n_{team}, number of team members.

six) members, the highest possible Gini Index value (denoting high inequality) ranges from 0.667 for a team of size 3 to 0.833 for a team of size 6.

3.4.3 Correlational Analysis. Our first research question and hypothesis are concerned with relationships between objective metrics of individual performance derived from GitHub and chat data, and subjective assessments of individual performance made by students and instructors. We explore these relationships using the Pearson correlation coefficient (R) [20].

Our second research question and final two hypotheses are concerned with relationships between the equality of team members' contributions and team members' perceptions of their teammates. We apply the Gini Index, which was introduced in Section 2.6, to determine the equality of team members' commits, line changes, stories, chat messages, chat words, and chat characters

Table 7. Summary of Individual Student Activity Counts and Assessment Results by Course

			GitHub Activities						Chat Activities						Assessment Results					
			Commits		Line Changes		Stories		Messages		Words		Chars		Contrib		Inter		Grade	
Course	*n*	*t*	*M*	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*	*M′*	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
A1	37	7	15.7	14.8	2293.4	3172.3	1.4	1.0	31.5	35.0	330.5	350.6	1346.1	1419.3	4.1	0.4	4.2	0.3	0.9	0.1
A2	11	2	10.9	8.2	1090.8	1199.0	5.1	2.9	18.4	12.2	297.2	143.5	1221.6	574.1	3.8	0.4	3.7	0.3	1.0	0.0
A3	9	2	17.4	11.5	3787.3	2731.7	4.3	2.6	14.9	12.7	96.9	94.3	370.0	364.2	4.2	0.4	4.0	0.4	0.9	0.0
B	57	12	15.2	18.3	599.3	861.6	4.7	3.9	52.4	57.3	615.1	659.4	2499.8	2681.0	4.1	0.8	4.3	0.7	0.9	0.2
Totals	**114**	**23**	15.1	15.9	1448.3	2296.7	3.6	3.4	39.4	47.3	451.1	536.4	1833.9	2179.7	4.1	0.6	4.1	0.6	0.9	0.2

n, number of students; *t*, number of teams; *M*, mean; *SD*, standard deviation.

contributed by members of each team. We then use the Pearson correlation coefficient to test for the hypothesized correlations.

4 RESULTS

4.1 Individual Activities and Assessments

We begin by presenting findings relevant to RQ1, which focuses on identifying possible correlations between individuals' chat activities and the objective (GitHub contributions) and subjective metrics (peer evaluations, instructor grades) explored in prior research.

4.1.1 Summary. Table 7 presents the individual metrics considered in this study. The table breaks the data down by course studied and includes overall averages across all courses. As the table indicates, and as would be expected, there was wide variance in individual students' GitHub and chat activities. Further inspection of Table 7 yields the following observations:

- Students in the three offerings of Course A changed two to seven times more lines of code than students in Course B; however, the average number of commits per student was fairly consistent across all courses, ranging from 11 to 17.
- Students in Course B posted roughly 1.5 to 3 times as many chat messages as students in the three offerings of Course A. These differences are also present when considering the average words and characters posted.
- Students' average CATME contribution and interaction ratings were consistently around 4.1/5 for Courses A1, A3, and B; however, average ratings for Course A2 were notably lower—around 3.7/5.
- Although students' average individual grades were consistently above 90% in the three offerings of Course A, they were somewhat lower (88%) in Course B.

4.1.2 Correlational Analyses. To explore correlations among the individual metrics, Table 8 presents the Pearson correlation values. In this analysis, we set the alpha value for significance to $p < 0.05$, and, per the advice of Akoglu [2], we interpret r_p values of below 0.3 as *weak* correlations, r_p values between 0.3 and 0.6 as *moderate* correlations, and r_p values above 0.6 as *strong* correlations.

H1 posits positive correlations between students' GitHub and chat metrics, and between objective (GitHub and chat) and subjective (peer assessment and grade) metrics. To shed light on this hypothesis, Table 8 highlights the 36 correlations between GitHub and chat metrics in dark gray, and the 36 correlations between subjective and objective metrics in light gray. As these highlighted areas indicate, 33 our of a possible 36 correlations between GitHub and chat metrics (92%) were statistically significant, with 29 of the 36 correlations (81%) having moderate strength. Likewise, 30 of a possible 36 correlations between the objective and subjective metrics (83%) were statistically significant, with 23 of the 36 correlations (64%) having moderate strength. Given these results, we can conclude that H1 is partially confirmed.

Table 8. Pearson Correlations Among *Individual* GitHub Metrics, Chat Metrics, and Peer/Instructor Evaluations

				GitHub Metrics						Chat Metrics						P/I Evaluation		
	Variable	M	SD	#C	RCS	#LC	RLCS	#S	RSS	#M	RMS	#W	RWS	#Ch	RChS	Contr	Inter	Grade
GitHub Metrics	#C	15.1	15.9	—														
	RCS	0.9	0.8	.73**	—													
	#LC	599.3	861.6	.49**	.34**	—												
	RLCS	0.9	1.0	.50**	.72**	.51**	—											
	#S	3.6	3.4	.42**	.38**	.06	.38**	—										
	RSS	0.9	0.6	.44*	.53**	.22*	.57**	.62**	—									
Chat Metrics	#M	39.4	47.3	.41**	.26**	.18	.35**	.28**	.37**	—								
	RMS	1.0	0.7	.33**	.44**	.40**	.58**	.36**	.48**	.55**	—							
	#W	451.1	536.4	.42**	.25**	.12	.32**	.28**	.29**	.89**	.46**	—						
	RWS	1.0	0.7	.29**	.41**	.36**	.55**	.33**	.44**	.45**	.89**	.46**	—					
	#Ch	1833.8	2179.7	.42**	.25**	.12	.32**	.28**	.29**	.88**	.46**	.99**	.46**	—				
	RChS	1.0	0.7	.30**	.42**	.36**	.56**	.33**	.48**	.45**	.88**	.46**	.99*	.46**	—			
P/I Evals	Contr	4.0	0.6	.45**	.50**	.27**	.47**	.45**	.51**	.41**	.44*	.36**	.44**	.36**	.45**	—		
	Inter	4.1	0.5	.37**	.40**	.17	.35**	.39**	.41**	.38**	.37**	.35**	.34**	.35**	.34**	.89**	—	
	Grade	0.9	0.1	.20**	.28**	.16	.26**	.14	.29**	.13	.30**	.03	.28**	.03	.28**	.37**	.26**	—

** $p < 0.001$; * $p < 0.05$; *M*, mean; *SD*, standard deviation; #C, commit count; RCS, relative commit share; #LC, line change count; RCLS, relative line change share; #S, story count; RSS, relative story share; #M, message count; RMS, relative message share; #W, message word count; RWS, relative message word share; #Ch, message character count; RChS, relative message character share; Contr, CATME contribution rating; Inter, CATME interaction rating; Grade, individual assignment grade.

Table 9. Teams' Mean Gini Index Values

Code or Communication Equality Metric	*Min*	*Max*	*M*	*SD*
Relative Commit Share Gini Index	0.192	0.632	0.375	0.123
Relative Line Share Gini Index	0.238	0.748	0.503	0.136
Relative Story Share Gini Index	0.113	0.750	0.295	0.149
Relative Message Share Gini Index	0.126	0.637	0.351	0.135
Relative Word Share Gini Index	0.049	0.628	0.345	0.155
Relative Character Share Gini Index	0.053	0.627	0.359	0.158

Min, minimum; *Max*, maximum; *M*, mean; *SD*, standard deviation.

Table 10. Teams' Min, Max, and Mean Peer Evaluation Ratings

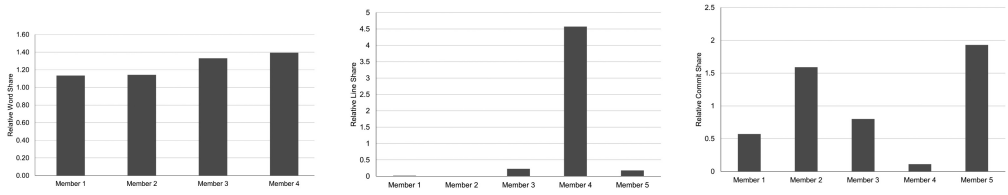
Peer Evaluation Metric	*Min*	*Max*	*M*	*SD*
CATME Rating: Contribution	2.90	4.90	3.99	0.47
CATME Rating: Interaction	2.82	4.80	4.08	0.50

Min, minimum; *Max*, maximum; *M*, mean; *SD*, standard deviation.

4.2 Relation Between Equality of Team Members' Contributions and Peer Evaluation Ratings

We now present findings relevant to RQ2, which is concerned with how the equality of team members' chat and code contributions relates to students' perceptions of their team members. Table 9 presents a summary of the Gini Index values across all teams for each code and chat metric, whereas Table 10 presents a summary of teams' CATME peer evaluation ratings. After providing a visual explanation of Gini Index values relative to our data, the remainder of this section visually explores these results and presents a correlational analysis.

4.2.1 Visualization of Gini Index Values. To provide some intuition for a range of Gini Index values, Figure 1 presents bar graphs of individual team members' contributions for a high-equality,



(a) Example of low Gini Index (0.05)

(b) Example of high Gini Index (0.75)

(c) Example of average Gini Index (0.37)

Fig. 1. Illustration of low, high, and average Gini Indices relative to individual team member contributions.

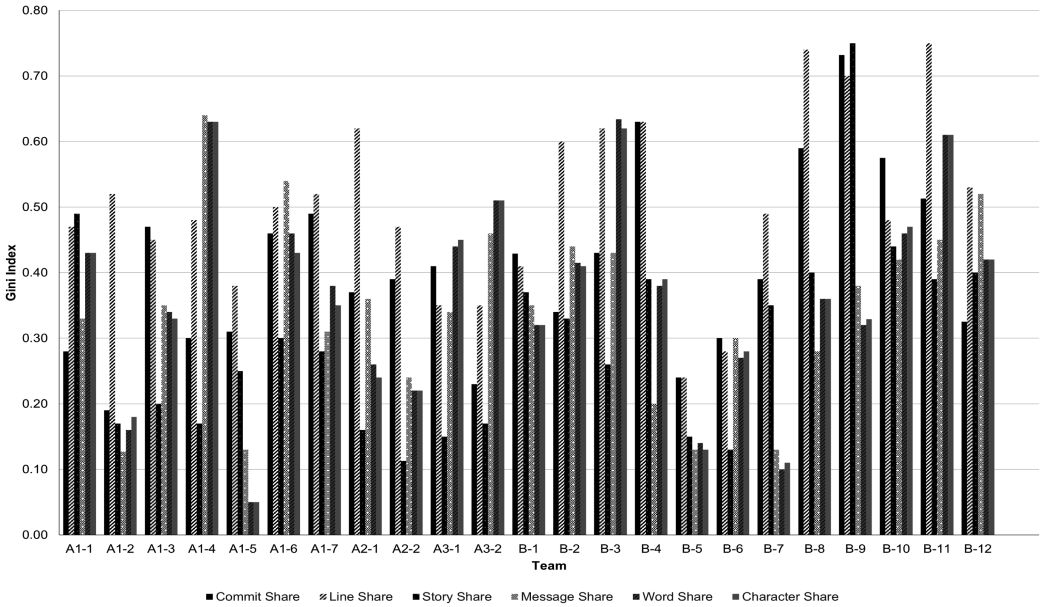


Fig. 2. Gini Index values for code and chat metrics by team.

average-equality, and low-equality team. Figure 1(a) shows the team with the lowest Gini Index value (0.05) on any measure considered in the study. As this figure shows, all four team members contributed nearly equally to the word count of the teams' messages. Figure 1(b) shows the team with the highest Gini Index value (0.75) on any measure. As this figure illustrates, one team member contributed nearly all of the lines of code to this team's code base. Finally, Figure 1(c) shows a team that is close to the mean Gini Index value with respect to the relative commit share. The commit shares of two team members were markedly higher than those of the three other team members. However, two of the three remaining team members also contributed significantly, leading to a Gini Index value of 0.37.

4.2.2 Visual Exploration of Gini Index Values by Team and Metric. Figure 2 presents a side-by-side comparison of the 23 study teams with respect to their Gini Index values for each code and chat metric. In this figure, we have labeled each team sequentially according to the course in which they were enrolled, using the same course abbreviations introduced in Table 1 (A[1-3] represents UC Santa Barbara, and B represents Washington State University).

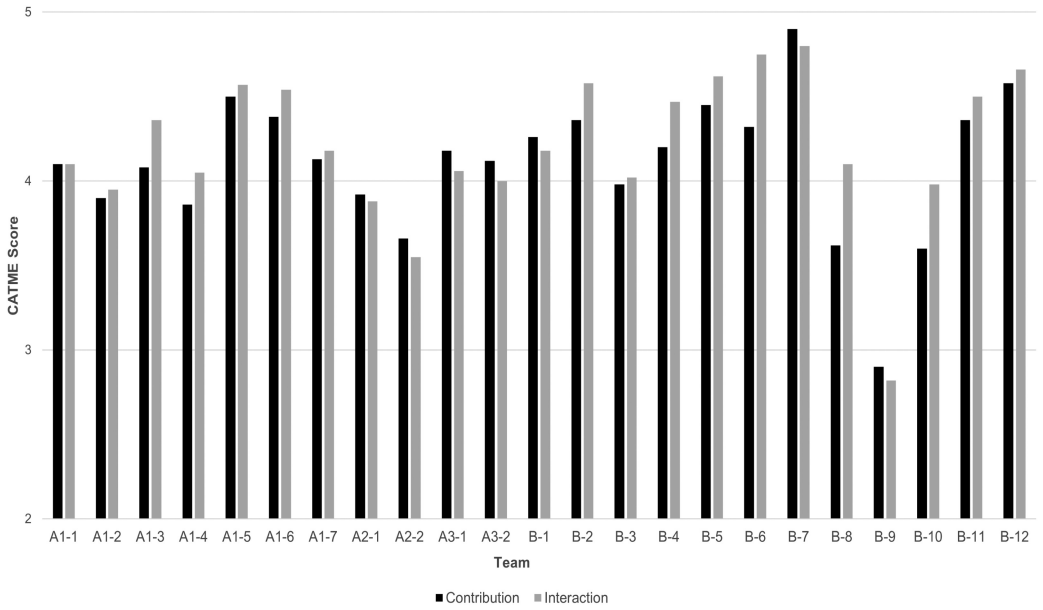


Fig. 3. Average CATME peer evaluation ratings (1–5 scale) by team.

Inspection of Figure 2 yields several notable observations:

- One team (A1-2) stands out for having equal commit contributions (Gini Index value = 0.19), whereas five teams (B-4, B-8, B-9, and B-11) stand out for having unequal commit contributions, with Gini Index values all above 0.5.
- With respect to line changes, teams with unequal team member contributions were more common, and inequality was more pronounced: seven teams had a member who contributed more than two times the expected number of line changes. An additional 8 teams had a team member who contributed more than three times as many line changes as expected. In all, 10 of these teams had Gini Index values above 0.5.
- Imbalances with respect to the number of completed stories (issues) closed were less common than imbalances in commits and lines changed. Team B-9 appears to be an outlier, with one team member contributing four times the expected number of stories, yielding a Gini Index of 0.73. No other team had a Gini Index value above 0.5 for this metric.
- Teams appeared to exhibit greater overall equality in the chat metrics than in the GitHub metrics. Indeed, for the relative message share metric, just 3 of the 23 teams had a Gini Index value above 0.5. Notably, this pattern held for the relative word and character share metrics as well.
- Team B-5 stands out as having consistent equality across all metrics, with all but two Gini Index values below 0.2. In contrast, Team B-10 stands out as being consistently unequal, with all Gini Index values exceeding 0.4.

4.2.3 Visual Exploration of CATME Peer Evaluation Ratings by Team. Figure 3 compares the 23 teams relative to their average CATME interaction and contribution ratings. This figure labels teams as in Figure 2.

Inspection of Figure 3 leads to two key observations:

- Most teams' mean peer evaluation ratings were above 4 on the CATME survey's 5-point scale. Just 7 of the 23 teams had an average peer evaluation rating that fell below this level.

Table 11. Pearson Correlations Between Teams' GitHub and Chat Contribution Equality and Their CATME Peer Evaluation Ratings

	CATME Peer Evaluation Ratings			
Contribution Equality	Contr	Contr Var	Inter	Inter Var
Commit Gini	−.45*	.58**	−.32	.53*
Line Change Gini	−.36	.39	−.27	.34
Story Gini	−.34	.66**	−.31	.67**
Message Gini	−.17	.06	−.13	.09
Word Gini	−.20	.04	−.13	.04
Char Gini	−.20	.05	−.13	.05

*$p < 0.05$; **$p < 0.01$.

- Each teams' average CATME contribution and interaction ratings appear similar, suggesting that these measures are highly correlated with each other.

4.2.4 Correlational Analysis. H2 and H3 posit relationships between the equality of teams' GitHub and chat contributions, as measured by the Gini Index, and their peer evaluation ratings. Whereas H2 predicts a negative relation between contribution equality and average peer evaluation rating, H3 predicts a positive relation between contribution equality and the variance in peer evaluation ratings.

Table 11 presents Pearson correlations relevant to these hypotheses. As the table shows, H2 and H3 held for two of the three GitHub metrics, correctly predicting the relationships between teams' commit and story equality and and their peer evaluation ratings. These relationships were found to be moderate in strength. Contrary to H2 and H3, however, no statistically significant correlations were detected between teams' chat activities and their peer evaluations.

5 DISCUSSION

Our results provide insight into the two research questions we posed for the study.

5.1 RQ1

With respect to *RQ1* (*How do students' individual contributions to their teams' online chats correlate with the subjective and objective metrics considered in prior work?*), we discovered that students' chat contributions were significantly correlated not only with their GitHub contributions (commits, line changes, stories) but also with their peer evaluation scores. Thus, students who were more involved with the coding aspects of a team software development project also tended to be more active communicators on the team. Moreover, their increased involvement in both realms was positively associated with higher peer evaluation ratings and higher individual course grades.

This finding aligns with a recent analysis of the chat and coding activities of a professional software team [32], which found that chat activity is strongly associated with both commits and line changes, with R values (0.27 to 0.33) similar to those in our study. Moreover, this lends support to H1, which posits positive correlations between students' chat contributions, code contributions, and peer evaluation ratings.

Since these results are correlational and not causal, they do not speak to the directionality of the relationships. However, based on an understanding of the collaborative software development process, we can speculate on why the relationships might exist. With respect to the correlation between students' code contributions and chat contributions, research has shown that software developers who contribute code to a shared code base need to coordinate their coding efforts with others on the team [15]. Coordination involves communication to delegate work, share progress,

and plan future activities. Teams in this study appear to have used their chat channels largely for these purposes. Indeed, a follow-up content analysis of teams' chat channels [27] revealed that 86% of team's chat channel messages focused on topics related to contribution, coordination, and planning. Given this, we suspect that those team members who contributed more actively to the team's code base, and who completed more user stories in the process, tended to have a higher need to communicate with others and therefore tended to post more messages to the team's chat channel.

With respect to the correlations between students' code/chat contributions and their average peer evaluation scores, we speculate that the observed relationships exist not because students who are viewed more positively by their peers tend to be more productive and communicative, but because students who contribute more code and communicate more extensively with their team are perceived more positively by their teammates. In work that inspired this study, Buffardi [7] posited that the *halo effect* [51] (i.e., the idea that one's overall impression of a person creates an assessment bias) may account for positive peer evaluation scores. Resonant with our previous study [28], the results of this study provide strong evidence that a student's *performance* on a software team, and not their reputation, has a powerful influence on how they are rated by teammates in peer evaluations.

Finally, we consider the significant correlations between students' code and chat contributions and their average course assignment grade. We observe that these were generally weaker ($0.20 \leq R \leq 0.29$) than the significant correlations identified between code metrics, chat metrics, and peer evaluation ratings. One might account for these correlations by observing that students who perform better in course assignments are also likely to perform better in the course's team software development project, which presumably requires them to apply the same skills and effort they applied to the course assignments.

5.2 RQ2

With regard to *RQ2* (*How does the equality of individual contributions within a team correlate with peer and instructor evaluations?*), the story appears to be more nuanced. First, we found that as the equality of team commits increases, (a) average peer evaluation ratings relative to the CATME *contribution* scale increase ($R = 0.45$) and (b) the variance in those ratings decreases ($R = 0.58$). Similarly, significant and strong negative relationships were found between the equality of teams' completion of stories and (c) the variance of their CATME contribution ratings ($R = 0.66$) and (d) the variance of their CATME interaction ratings ($R = 0.67$). These results provide support for H2 and H3, with result (a) aligning with H2, and results (b), (c), and (d) aligning with H3. However, the remaining 19 of 24 possible relationships between Gini Index values of team contribution equality and CATME peer evaluation ratings shown in Table 11 did not materialize.

What stands out in these results is that we failed to identify any significant relationships between the equality of teams' *chat contributions* and their peer evaluation ratings. In other words, equal contributions to the chat channels was *not* associated with higher average peer evaluation ratings and lower variance in the ratings. One explanation of this result is that we failed to capture all team communications. Even though the study took place during the pandemic and students were required to interact online, we collected data on just one online venue for online communication: the official team chat channel (in Slack or MS Teams) sanctioned by the course instructor. Members of student teams likely communicated through other online technologies, including unofficial chat tools (e.g., WhatsApp, Discord), text messaging, and video conferencing (e.g., Zoom, MS Teams). Thus, inequalities in contributions to team communication could have occurred without our being able to measure them. In turn, this could explain our inability to measure a hypothesized correlation between chat equality and peer evaluation ratings.

5.3 Threats to Validity

5.3.1 Internal Validity. Internal validity concerns the extent to which our research methods uncover the truth regarding our research questions in our specific population. One such threat was just mentioned: we considered only team communications that students posted in official team chat channels (in Slack or MS Teams) sanctioned by the course instructor. Students certainly used other means to communicate, including Discord, Zoom, and direct messages within our official chat apps; however, we did not have access to those communications. Other places where students may have communicated, but that our analysis missed, include commit messages and comments in GitHub issues and pull requests; analyzing these could provide a fertile area for future work.

Another threat to internal validity relates to the way in which we measured student contributions. Using counts of commits, line changes, completed stories, and chat messages as the sole gauges of students' contributions to a team software project neglects the myriad other ways in which students can contribute. For instance, students could have played a leadership role that was unregistered in GitHub and chat metrics. Likewise, they may have provided solutions whose value was simply not reflected in counts of commits, line changes, or chat messages. Thus, a challenge for future research is not only to consider a broader range of potential contributions but also to incorporate some measure of the value or impact of those contributions.

5.3.2 External Validity. External validity considers the extent to which our results can be generalized to software engineering education or other team software development contexts. A threat to external validity is that all of the course offerings in the study occurred during academic year 2020–2021, the first full academic year of the global Covid-19 pandemic. During this period, instruction was fully online in the courses studied. The additional stress of this change, along with the external stressors of the pandemic itself, may have influenced the outcomes in ways that will not be apparent without further follow-up study under non-pandemic conditions.

Another threat to external validity is that we considered the work of small student software teams working over relatively short periods of time (5–10 weeks). In professional settings, teams are frequently larger and project durations are significantly longer. Although large inequalities in code and communication contributions have been observed in large-scale open source projects [21], one must clearly be careful in any attempt to generalize our results beyond the limited academic settings in which we performed the study.

A third threat to validity relates to a potential sampling bias. Although all 12 teams in the course at Washington State University consented to participate, just 11 of 34 teams at UC Santa Barbara consented. The main reason for this low consent rate is that we excluded an entire team from the study if even one of the five to six team members did not consent. If the data on the non-consenting teams were included in our analysis, the results could change.

6 CONCLUSION, IMPLICATIONS, AND FUTURE WORK

To address the problem of assessing individual contributions to team software development projects in undergraduate computing education, we have presented an empirical study of the relationships between (a) subjective and objective measures of individual contributions, and (b) team contribution equality and student perceptions of team functioning. With respect to (a), we have identified statistically significant, moderately strong relationships between subjective assessments (peer and instructor evaluations) and objective assessments rooted in empirical data on students' code and chat contributions. With respect to (b), we have found statistically significant, strong correlations between the equality of individual commit and story contributions and the positivity of teams' peer evaluation ratings. By extending prior correlational analyses that focused only on teams' code contributions, these results contribute new evidence that

objective metrics of individual team members' code and chat contributions can be combined with subjective metrics to provide assessments of individual and team performance that are not only more comprehensive but also consistent with each other.

6.1 Implications

Our results have at least three implications for computing courses that include team software development projects.

6.1.1 Combine Objective and Subjective Assessment. To make the assessment of individuals in team projects more objective and consistent, instructors should integrate objective performance metrics derived from data on students' software development activities, including their activities in coding repositories and chat channels. By developing structured rubrics that align with these metrics, instructors can communicate performance requirements more clearly and evaluate performance more objectively. In so doing, however, instructors need to be mindful of Campbell's law [8]: the possibility that students, or entire teams, could modify their behavior to meet performance requirements, even if such modifications are contrary to course learning goals.

6.1.2 Use Objective Metrics for Formative Assessment. Our results suggest that metrics on individual and team performance based on objective data well complement traditional peer and instructor assessments, providing more comprehensive feedback than subjective assessments alone can provide. However, because the objective metrics explored in this article can be computed *automatically* from log data, they offer a potential advantage: they can be computed quickly throughout the software development process, thus offering more timely feedback. Instructors should therefore consider the possibility of using them as a basis for more frequent formative assessments of team and individual progress.

6.1.3 Equality Metrics as Early Warning Signs of Trouble. Our results, like those of previous research into collaborative software development, indicate that individuals rarely contribute equally to teams. This is not necessarily a problem: team members bring many talents to the table to which equality metrics are simply insensitive. Nonetheless, our results suggest that instructors should consider using equality metrics such as the Gini Index as an early warning sign of possible team dysfunction or discord. If inequalities are brought to instructors' and students' attention early on, there is a better chance that teams can make changes that will positively influence their performance.

6.2 Future Work

Based on the results of this study, three avenues for future work stand out to us. First, a significant threat to the validity of our results stems from the fact that we considered data on only one component of team communications: posts to official course chat channels. Moreover, we considered only one aspect of those communications: *counts* of messages, words, and characters. Although this limited set of data provided insight into relationships between team members' chat contributions, code contributions, and peer assessments, it can paint only a small part of the overall picture. In future work, we would like to analyze a broader set of data on team communications, including their communications within the GitHub platform (commit messages, issue comments, code reviews, pull requests), meetings (e.g., agile standups and retrospectives) via video conferencing, and informal communications via unofficial chat platforms (e.g., Discord). By combining and triangulating data from multiple sources, we can deepen our understanding of team communication and increase the internal validity of our results.

Second, our results demonstrate the potential for objective data, collected automatically through software teams' code repositories and communication tools, to provide useful insights into individual and team performance. In future work, we will share other computing educators' interest (e.g., [22]) in developing *analytics dashboards* to provide instructors and teams with *formative* feedback on their development processes. Along the same lines, metrics like the ones explored in this work can provide a basis for automated prompts or interventions to guide individuals and teams toward best practices. For example, a chat bot could prompt individuals who are not involved in team discussions to engage in the chat. Likewise, teams could be prompted to communicate after key software development events occur. The overall goal would be to leverage formative assessments based on automatically collected data to help students and teams to reflect on, and ultimately to improve, their software development processes.

Finally, we recognize that the conclusions we can draw from this study are limited due to its correlational nature. As this line of work matures and we obtain greater insight into relationships between software development and communication activities, we would like to design and implement more controlled studies to test evidence- and theory-based hypotheses regarding the impact of specific pedagogical and software interventions on students' communication, software development, and learning behaviors in team software projects.

ACKNOWLEDGMENTS

The authors would also like to thank Kevin Buffardi for his assistance in interpreting his original paper [7], upon which both our previous paper [28] and this article builds.

REFERENCES

[1] Efthimia Aivaloglou and Anna van der Meulen. 2021. An empirical study of students' perceptions on the setup and grading of group programming assignments. *ACM Transactions on Computing Education* 21, 3 (2021), Article 17, 22 pages. https://doi.org/10.1145/3440994

[2] Haldun Akoglu. 2018. User's guide to correlation coefficients. *Turkish Journal of Emergency Medicine* 18, 3 (2018), 91–93. https://doi.org/10.1016/j.tjem.2018.08.001

[3] Rana Alkadhi, Teodora Lata, Emitza Guzmany, and Bernd Bruegge. 2017. Rationale in development chat messages: An exploratory study. In *Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR'17)*. IEEE, Los Alamitos, CA, 436–446. https://doi.org/10.1109/MSR.2017.43

[4] Andrew Begel and Beth Simon. 2008. Novice software developers, all over again. In *Proceedings of the 4th International Workshop on Computing Education Research (ICER'08)*. ACM, New York, NY, 3–14. https://doi.org/10.1145/1404520.1404522

[5] Andrew Begel and Beth Simon. 2008. Struggles of new college graduates in their first software development job. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'08)*. ACM, New York, NY, 226–230. https://doi.org/10.1145/1352135.1352218

[6] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. 2015. Software engineering project courses with industrial clients. *ACM Transactions on Computing Education* 15, 4 (Dec. 2015), Article 17, 31 pages. https://doi.org/10.1145/2732155

[7] Kevin Buffardi. 2020. Assessing individual contributions to software engineering projects with Git logs and user stories. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, 650–656.

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

[9] Lori Carter. 2011. Ideas for adding soft skills education to service learning and capstone courses for computer science students. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE'11)*. ACM, New York, NY, 517–522. https://doi.org/10.1145/1953163.1953312

[10] Preetha Chatterjee, Kostadin Damevski, Lori Pollock, Vinay Augustine, and Nicholas A. Kraft. 2019. Exploratory study of slack Q&A chats as a mining source for software engineering tools. In *Proceedings of the 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR'19)*. IEEE, Los Alamitos, CA, 490–501. https://doi.org/10.1109/MSR.2019.00075

[11] Chau-Nan Chen, Tien-Wang Tsaur, and Tong-Shieng Rhai. 1982. The Gini coefficient and negative income. *Oxford Economic Papers* 34, 3 (1982), 473–478.

[12] Tony Clear. 2009. Thinking ISsues: The three p's of capstone project performance. *ACM SIGCSE Bulletin* 41, 2 (June 2009), 69–70. https://doi.org/10.1145/1595453.1595468

[13] Michelle Craig, Phill Conrad, Dylan Lynch, Natasha Lee, and Laura Anthony. 2018. Listening to early career software developers. *Journal of Computing Sciences in Colleges* 33, 4 (April 2018), 138–149.

[14] Joanna F. DeFranco and Philip A. Laplante. 2017. Review and analysis of software development team communication research. *IEEE Transactions on Professional Communication* 60, 2 (2017), 165–182. https://doi.org/10.1109/TPC.2017.2656626

[15] Torgeir Dingsøyr and Tore Dybå. 2012. Team effectiveness in software development: Human and cooperative aspects in team effectiveness models and priorities for future studies. In *Proceedings of the 2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering (CHASE'12)*. IEEE, Los Alamitos, CA, 27–29. https://doi.org/10.1109/CHASE.2012.6223016

[16] César Domínguez, Arturo Jaime, Francisco J. García-Izquierdo, and Juan J. Olarte. 2020. Factors considered in the assessment of computer science engineering capstone projects and their influence on discrepancies between assessors. *ACM Transactions on Computing Education* 20, 2 (March 2020), Article 14, 23 pages. https://doi.org/10.1145/3381836

[17] Robert Dorfman. 1979. A formula for the Gini coefficient. *Review of Economics and Statistics* 61, 1 (1979), 146–149. https://EconPapers.repec.org/RePEc:tpr:restat:v:61:y:1979:i:1:p:146-49.

[18] Marisa Exter. 2014. Comparing educational experiences and on-the-job needs of educational software designers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE'14)*. ACM, New York, NY, 355–360. https://doi.org/10.1145/2538862.2538970

[19] Vivienne Farrell, Gilbert Ravalli, Graham Farrell, Paul Kindler, and David Hall. 2012. Capstone project: Fair, just and accountable assessment. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'12)*. ACM, New York, NY, 168–173. https://doi.org/10.1145/2325296.2325339

[20] Stephanie Glen. 2022. Correlation Coefficient: Simple Definition, Formula, Easy Steps. Retrieved May 11, 2023 from https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/.

[21] Mathieu Goeminne and Tom Mens. 2011. Evidence for the pareto principle in open source software activity. In *Proceedings of the 2011 Workshop on Software Quality and Maintainability (SQM'11)*. 74–82.

[22] Sivana Hamer, Christian Quesada-López, Alexandra Martínez, and Marcelo Jenkins. 2020. Measuring students' contributions in software development projects using Git metrics. In *Proceedings of the 2020 XLVI Latin American Computing Conference (CLEI'20)*. IEEE, Los Alamitos, CA, 531–540. https://doi.org/10.1109/CLEI52000.2020.00068

[23] Brandon Heller, Eli Marschner, Evan Rosenfeld, and Jeffrey Heer. 2011. Visualizing collaboration and influence in the open-source software community. In *Proceedings of the 8th Working Conference on Mining Software Repositories*. IEEE, Los Alamitos, CA, 223–226.

[24] Nicole Herbert. 2018. Reflections on 17 years of ICT capstone project coordination: Effective strategies for managing clients, teams and assessment. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE'18)*. ACM, New York, NY, 215–220. https://doi.org/10.1145/3159450.3159584

[25] Michael Hewner and Mark Guzdial. 2010. What game developers look for in a new graduate: Interviews and surveys at one game company. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10)*. ACM, New York, NY, 275–279. https://doi.org/10.1145/1734263.1734359

[26] E. M. Hoover. 1936. The measurement of industrial localization. *Review of Economics and Statistics* 18 (1936), 162–171.

[27] Christopher Hundhausen, Phillip Conrad, Olusola Adesope, Ahsun Tariq, Samir Sbai, and Andrew Lu. 2023. Investigating reflection in undergraduate software development teams: An analysis of online chat transcripts. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V.1 (SIGCSE'23)*. ACM, New York, NY, 7. https://doi.org/10.1145/3545945.3569790

[28] C. D. Hundhausen, P. T. Conrad, A. S. Carter, and O. Adesope. 2022. Assessing individual contributions to software engineering projects: A replication study. *Computer Science Education* 32, 3 (2022), 335–354. https://doi.org/10.1080/08993408.2022.2071543

[29] Ville Isomöttönen and Tommi Kärkkäinen. 2008. The value of a real customer in a capstone project. In *Proceedings of the 2008 21st Conference on Software Engineering Education and Training*. IEEE, Los Alamitos, CA, 85–92. https://doi.org/10.1109/CSEET.2008.24

[30] Letizia Jaccheri and Thomas Osterlie. 2007. Open source software: A source of possibilities for software engineering education and empirical software engineering. In *Proceedings of the 1st International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07)*. IEEE, Los Alamitos, CA, 5. https://doi.org/10.1109/FLOSS.2007.12

[31] J. Johnston. 1969. H. Theil. *Economics and Information Theory. Economic Journal* 79, 315 (Sept. 1969), 601–602. https://doi.org/10.2307/2230396

[32] Miikka Kuutila, Mika V. Mäntyla, and Maëlick Claes. 2020. Chat activity is a better predictor than chat sentiment on software developers productivity. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20)*. ACM, New York, NY, 553–556. https://doi.org/10.1145/3387940.3392224

[33] Richard Layton, Matthew Ohland, and Hal R. Pomeranz. 2007. Software for student team formation and peer evaluation: CATME incorporates team-maker. In *Proceedings of the 2007 Annual Conference and Exposition*. 12.1286.1–12.1286.5. https://peer.asee.org/software-for-student-team-formation-and-peer-evaluation-catme-incorporates-team-maker.

[34] Zhifang Liao, Dayu He, Zhijie Chen, Xiaoping Fan, Yan Zhang, and Shengzong Liu. 2018. Exploring the characteristics of issue-related behaviors in GitHub using visualization techniques. *IEEE Access* 6 (2018), 24003–24015. https://doi.org/10.1109/ACCESS.2018.2810295

[35] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. Why developers are slacking off: Understanding how software teams use slack. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW'16 Companion)*. ACM, New York, NY, 333–336. https://doi.org/10.1145/2818052.2869117

[36] Chang Liu. 2005. Enriching software engineering courses with service-learning projects and the open-source approach. In *Proceedings of the 27th International Conference on Software Engineering (ICSE'05)*. ACM, New York, NY, 613–614. https://doi.org/10.1145/1062455.1062566

[37] Andrew C. Loignon, David J. Woehr, Jane S. Thomas, Misty L. Loughry, Matthew W. Ohland, and Daniel M. Ferguson. 2017. Facilitating peer evaluation in team contexts: The impact of frame-of-reference rater training. *Academy of Management Learning & Education* 16, 4 (Dec. 2017), 562–578. https://doi.org/10.5465/amle.2016.0163

[38] Robert Marmorstein. 2011. Open source contribution as an effective software engineering class project. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education (ITiCSE'11)*. ACM, New York, NY, 268–272. https://doi.org/10.1145/1999747.1999823

[39] Branko Milanovic. 1997. A simple way to calculate the Gini coefficient, and some implications. *Economics Letters* 56, 1 (1997), 45–49.

[40] James Morgan. 1962. The anatomy of income distribution. *Review of Economics and Statistics* 44, 3 (1962), 270–283. http://www.jstor.org/stable/1926398.

[41] 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, 669–670. https://doi.org/10.1145/3017680.3017682

[42] Debora M. C. Nascimento, Christina F. G. Chavez, and Roberto A. Bittencourt. 2018. The adoption of open source projects in engineering education: A real software development experience. In *Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE'18)*. IEEE, Los Alamitos, CA, 1–9. https://doi.org/10.1109/FIE.2018.8658908

[43] Bao-An Nguyen, Kuan-Yu Ho, and Hsi-Min Chen. 2020. Measure students' contribution in web programming projects by exploring source code repository. In *Proceedings of the 2020 International Computer Symposium (ICS'20)*. IEEE, Los Alamitos, CA, 473–478. https://doi.org/10.1109/ICS51289.2020.00099

[44] Tom Nurkkala and Stefan Brandle. 2011. Software studio: Teaching professional software engineering. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE'11)*. ACM, New York, NY, 153–158. https://doi.org/10.1145/1953163.1953209

[45] 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.

[46] Nilay Oza, Fabian Fagerholm, and Jürgen Münch. 2013. How does Kanban impact communication and collaboration in software engineering teams? In *Proceedings of the 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE'13)*. IEEE, Los Alamitos, CA, 125–128. https://doi.org/10.1109/CHASE.2013.6614747

[47] Reza M. Parizi, Paola Spoletini, and Amritraj Singh. 2018. Measuring team members' contributions in software engineering projects using Git-driven technology. In *Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE'18)*. IEEE, Los Alamitos, CA, 1–5. https://doi.org/10.1109/FIE.2018.8658983

[48] Alex Radermacher and Gursimran Walia. 2013. Gaps between industry expectations and the abilities of graduates. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE'13)*. ACM, New York, NY, 525–530. https://doi.org/10.1145/2445196.2445351

[49] Samaneh Saadat, Olivia B. Newton, Gita Sukthankar, and Stephen M. Fiore. 2020. Analyzing the productivity of GitHub teams based on formation phase activity. In *Proceedings of the 2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'20)*. IEEE, Los Alamitos, CA, 169–176.

[50] Jeffrey S. Saltz and Robert R. Heckman. 2018. A scalable methodology to guide student teams executing computing projects. *ACM Transactions on Computing Education* 18, 2 (July 2018), Article 9, 19 pages. https://doi.org/10.1145/3145477

[51] Frank W. Schneider, Jamie A. Gruman, and Larry M. Coutts. 2011. *Applied Social Psychology: Understanding and Addressing Social and Practical Problems*. SAGE, Thousand Oaks, CA.

[52] Alexander Serebrenik and Mark van den Brand. 2010. Theil index for aggregation of software metrics values. In *Proceedings of the 2010 IEEE International Conference on Software Maintenance*. IEEE, Los Alamitos, CA, 1–9. https://doi.org/10.1109/ICSM.2010.5609637

[53] Mark Sherriff and Sarah Heckman. 2018. Capstones and large projects in computing education. *ACM Transactions on Computing Education* 18, 2 (July 2018), Article 6, 4 pages. https://doi.org/10.1145/3229882

[54] 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, 397–402. https://doi.org/10.1145/2538862.2538932

[55] Ioannis Stamelos. 2009. Teaching software engineering with Free/Libre open source projects. *International Journal of Open Source Software and Processes* 1, 1 (2009), 72–90.

[56] Viktoria Stray and Nils Brede Moe. 2020. Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and slack. *Journal of Systems and Software* 170 (2020), 110717.

[57] 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, 494–499. https://doi.org/10.1145/2676723.2677223

[58] Purdue University. 2023. CATME Five Teamwork Dimensions. Retrieved May 11, 2023 from https://info.catme.org/features/catme-five-dimensions/.

[59] Jari Vanhanen, Timo O. A. Lehtinen, and Casper Lassenius. 2018. Software engineering problems and their relationship to perceived learning and customer satisfaction on a software capstone project. *Journal of Systems and Software* 137 (2018), 50–66.

[60] Rajesh Vasa, Markus Lumpe, Philip Branch, and Oscar Nierstrasz. 2009. Comparative analysis of evolving software systems using the Gini coefficient. In *Proceedings of the 2009 IEEE International Conference on Software Maintenance*. IEEE, Los Alamitos, CA, 179–188. https://doi.org/10.1109/ICSM.2009.5306322

[61] Brian R. von Konsky and Jim Ivins. 2008. Assessing the capability and maturity of capstone software engineering projects. In *Proceedings of the 10th Conference on Australasian Computing Education—Volume 78 (ACE'08)*. 171–180.

[62] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. 2015. GitHub. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR'15)*. IEEE, Los Alamitos, CA, 367–371.

Received 31 October 2022; revised 9 February 2023; accepted 3 March 2023