The Journal of Systems & Software 206 (2023) 111839

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

The impact of unequal contributions in student software engineering team projects[☆]

Kamilla Kopec-Harding[a], Sukru Eraslan[b,*], Bowen Cai[c], Suzanne M. Embury[c], Caroline Jay[c]

[a] Research IT, University of Manchester, Oxford Road, Manchester, M13 9PL, United Kingdom
[b] Middle East Technical University, Northern Cyprus Campus, Kalkanlı, Güzelyurt, 99738, Mersin 10, Turkey
[c] Department of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, United Kingdom

ARTICLE INFO

Article history:
Received 10 March 2023
Received in revised form 15 July 2023
Accepted 4 September 2023
Available online 6 September 2023

Keywords:
Student team projects
Team work
Work distribution
Cooperative/collaborative learning
Post-secondary education

ABSTRACT

Unequal distribution of work is a common problem in student team projects, undermining learning objectives and reducing student satisfaction with teamwork. More is needed to be known about the impact of unequal distribution of work on the performance of student software engineering teams and their members and the relationship between objective measures of unequal contribution and team-perceived unequal contribution. A greater understanding of these issues allows for targeted and personalised responses to these behaviours. We investigated several aspects of unequal contribution in student software engineering teams. We measured inequality of contribution using Git data from student software engineering teams, source code quality using code analyser SonarQube, and the team performance using grades. According to our results, most students under-contributed to their teams, and at least half the teams in each assignment had low equality of contribution or extreme inequality of contribution. Individual contribution styles did not strongly persist between modules (one-semester-long software engineering courses). There were no consistent associations between inequality of contribution in the student teams and performance or code quality. When the contribution of the least active team member was less than 14% of their fair share, teams were likely to perceive an unequal distribution of contributions.

1. Introduction

Many undergraduate software engineering courses include team projects which require students to build or modify a piece of software together (Vanhanen et al., 2012; Mahnic, 2012; van der Duim et al., 2007). These courses provide experience of industrial software engineering in a controlled environment, introduce best practices and expose students to industry standard software development tools such as version control systems, repository managers and continuous integration tools (Raibulet and Arcelli Fontana, 2018; Embury and Page, 2019; Bai et al., 2018). They also offer a valuable opportunity for students to develop non-technical skills through team work such as communication, conflict management and collaboration (Khakurel and Porras, 2020; Bastarrica et al., 2017), while allowing students the chance to explore the subject material together through cooperative learning (Soundarajan et al., 2015).

A common source of conflict and dissatisfaction in student team projects is the unequal distribution of work between team members (Williams et al., 1991; Tucker and Abbasi, 2015; LaBeouf et al., 2016). Within the education literature, the term "social loafing" or "free-riding" is used to refer to the tendency of some individuals to under-contribute within a team, relying on teammates to complete the assignment (Aggarwal and O'Brien, 2008). This behaviour may have a negative impact on the motivation and contributions of other team members (Karau and Williams, 1995; Comer, 1995). A related phenomenon, the one-person team, occurs when one team member completes most of the work (Iacob and Faily, 2019). Pieterse and Thompson (2010) refer to a related behaviour of "diligent isolation", where one person over-contributes because they do not trust others to contribute or wish to avoid the overheads of team-work such as communication and coordination of tasks.

Social loafing is an active area of research that has been reviewed elsewhere (Simms and Nichols, 2014). Lines of research focused on student teams include antecedents of social loafing in student teams (Jassawalla et al., 2009), student attitudes to social loafing (Jassawalla et al., 2009; Boren and Morales, 2018), impact on student team performance (Schippers, 2014) and strategies for

[☆] Editor: Dr. Kelly Blincoe.
[*] Corresponding author.
E-mail address: seraslan@metu.edu.tr (S. Eraslan).

https://doi.org/10.1016/j.jss.2023.111839
0164-1212/© 2023 Elsevier Inc. All rights reserved.

reducing social loafing in student projects (Aggarwal and O'Brien, 2008; Synnott, 2016). An unequal distribution of work in student software engineering teams has been noted by a wide range of studies including Stein (2003), Mittal and Sureka (2014), and Liu et al. (2004). Both under-contribution and over-contribution by team members have been recorded.

As professional software engineering is fundamentally a team-based activity, there is the potential for negative experiences of formative team-work to lead to negative attitudes towards future team projects (Adams, 2003). A diary-based case-study by Yasar et al. (2007) found that the self-efficacy of a female engineering student was diminished when "she was not given the opportunity to participate" in elements of a robotics team project. A survey study by Kapoor and Gardner-McCune (2018) found that female students who considered switching out of a computer science major reported negative team experiences such as working with team members who did not contribute sufficiently. As a result, software engineering educationalists are interested in reducing the unequal distribution of work in student team projects and helping teams to manage this problem (Fronza and Wang, 2017).

Previous efforts to reduce social-loafing in student software engineering projects have included structural approaches, such as optimising team formation methods (Pieterse and Thompson, 2010), monitoring team progress (Marques, 2016) and introducing mandatory lab sessions (Borg, 2020), and interpersonal interventions, such as raising students' awareness of how to deal with social loafing (Fronza and Wang, 2017), encouraging reflection on team projects (Leung, 2017) and deliberate exposure to challenging team-work scenarios ("Rocking the Boat", Pieterse et al., 2011). However, few of these approaches have been formally evaluated and effective interventions for student software engineering teams remain of interest to the software engineering education community.

The use of industry standard software development tools such as repository management systems e.g. GitLab (Bai et al., 2018; Embury and Page, 2019), GitHub (Feliciano et al., 2016), continuous integration tools e.g. Jenkins (Embury and Page, 2019), project management platforms e.g. Microsoft Teams (Raibulet and Arcelli Fontana, 2018) and code-quality tools such as SonarQube (Bai et al., 2018; Raibulet and Arcelli Fontana, 2018) in undergraduate software engineering courses facilitates the real-time acquisition of student team activity data. This presents an opportunity to intervene in dysfunctional team behaviours using novel approaches. A number of authors have suggested ways in which team activity data can be utilised to promote a more equal distribution of work in student software engineering teams.

Information-based intervention is a simple approach used in many domains and involves providing participants with factual information to encourage behaviour change (Blickem et al., 2011; Lautenbach et al., 2020; Lu, 2020). Several research groups have developed dashboards intended to increase the visibility of contributions in student team projects (Mahnic, 2012; Kay et al., 2007; Eraslan et al., 2020). Bai et al. (2018) developed a DevOps system for monitoring and providing automatic feedback to agile student teams and described how metrics from GitLab can be used to identify problems and trigger targeted offline interactions between Teaching Assistants (TAs) and students; they reported using the standard deviation of the number of commits within a team to monitor work distribution. Hamer et al. (2020, 2021b) calculated contribution inequality metrics for student team projects and suggested that these be used to monitor teams.

Studying unequal distribution of work in student software engineering teams is important because it can undermine the learning objectives of a team project. For example, Chen et al. (2014) suggest that under-contributors may miss out on opportunities to develop technical skills while over-contributors may become over-burdened and lose out on opportunities to develop teamwork skills such as coordination and communication. Sankaranarayanan et al. (2019) also note that under-contributors may lose practice opportunities while the team as a whole loses opportunities to discuss the project together. Although the potential negative impact of unequal work distribution has been acknowledged, few studies have attempted to quantify it and formally evaluate the impact on student software engineering teams.

In this paper, we assess the impact of unequal contributions in student software engineering teams using Git data from three cohorts of second year software engineering students at the University of Manchester, UK. Specifically, we study the impact of unequal contribution on four objectively-assessed outcomes: grades, code quality, future contributions to team work and team-perceived unequal contribution. We investigate the following research questions:

- RQ1: How uniform is the distribution of coding contributions in student software engineering teams?
- RQ2: Does over-contribution or under-contribution made by an individual affect their contribution in a subsequent software engineering team project?
- RQ3: Does the degree of inequality of contribution experienced by a student in their team in a software engineering team project affect their contribution in a subsequent software engineering team project?
- RQ4: Is there a relationship between inequality of contribution within student teams and team performance?
- RQ5: Is there a relationship between objective repository-based metrics of contribution inequality and team-perceived contribution inequality (indicated by the team accepting or successfully requesting a redistribution of marks from the course team)?

The main contribution of this paper is the quantitative assessment of the relationship between unequal contribution in software engineering student teams and these outcomes using objective Git metrics in multiple software engineering team assignments in a large sample of students across multiple cohorts.

Based on our findings, we recommend considering the students' behaviour in the current project rather than relying on previous projects, as the contribution style varies according to the situation. We also recommend considering equal work distribution explicitly in project design and providing alternatives to reporting under-contributors.

Our paper is structured as follows: first we examine related work relevant to each of these questions and describe our data sources and choice of inequality metrics. We then present our findings in relation to each of our research questions. Finally, we discuss our findings and the limitations of the study, and present recommendations for addressing unequal distribution of work in student teams.

2. Related work

Prior work has demonstrated the prevalence of unequal distribution of work in student software engineering teams using various data sources, including software repositories, team retrospectives, self-assessments and peer feedback. The majority of this work uses data from a small number of student teams. In an analysis of retrospectives from 35 student teams, Iacob and Faily (2019) found that 35% of the teams reported problematic lack of commitment by one or more team members, and an analysis of the Git repositories of 10 student teams by Buffardi (2020) found 50% of team members made less than 50% of their "expected share" of line changes where "expected share" was defined as total line changes divided by team size. A study of student self-assessments and peer feedback from six student teams by Pieterse and Thompson (2010) identified instances of free-riding and diligent isolation behaviours, and an analysis of the logbooks kept by nine student teams by Stein (2003) identified different team types based on the distribution of work, including "heroic teams" where one person has done the majority of the work and "slacker" teams where one person had done much less work than others.

Existing work has also investigated whether student contribution behaviour changes from one project to another. Specifically, Pieterse et al. (2012) looked at changes in student contribution across a number of rounds in a software engineering project based on researcher-assessed contribution categories (diligent isolate, insightful shaper, compliant worker, free-rider) using self-assessed and peer-assessed measures. They found that the prevalence of these categories changed over the rounds, for example, the prevalence of diligent isolate was 17.5%, 6.4%, 4.3%, 11.4% and 16.3% in the five rounds respectively indicating that these contribution styles may be context dependent. In our study, we complement this approach by examining changes in student participation across consecutive project assignments using objective data from student Git repositories

Instead of looking at the changes of student contribution behaviours in a particular project, Marshall et al. (2016) clustered students by participation trajectory across different projects. Although the participation behaviours of the students in different situations were observed, to the best of our knowledge no studies have reported on the impact of inequality of contribution within student software engineering teams and individuals' later contributions. As part of our study, we also investigate whether the degree of inequality of contribution experienced by students in a software engineering team project has effects on their contribution in a subsequent software engineering project.

It is generally assumed that social loafing and free-riding have a negative impact on team performance, but this has not been investigated in detail. Whilst Buffardi (2020) reported on the relationship between students' relative contributions (line changes) and individualised instructor-assigned grade ($n = 10$ teams), they did not look at team-level performance. Sankaranarayanan et al. (2019) investigated the relationship between unequal distribution of work in a computer science team project (not an extended software engineering project) and grade, but they did not find a relationship. However, He (2012b) provides evidence for the negative impact of unequal work distribution with a longitudinal study using two surveys at the beginning and the end of the software development process with 69 teams in a course where students were expected to work in a team to develop a relational-database system. The study shows that the presence of free-riding at a given stage may affect the team performance in the next stage by negatively affecting team morale. Pieterse et al. (2012) also provided evidence that the success of the team may be negatively affected when the proportion of social loafers increases, but the relationship was not formally assessed quantitatively. In our study, we extend this work by assessing the relationship between inequality of contribution in student software engineering teams using two measures of team-level performance: team grade and source code quality.

In our study, we also investigate the relationship between Git metrics and team-perceived unequal contribution, and to the best of our knowledge, there is no such study in the literature.

3. Methods

3.1. Study population

In this paper, we conduct an exploratory, multi-cohort study of students taking second-year software engineering project courses at the University of Manchester in academic years 2017/2018, 2018/2019 and 2019/2020 (see Tables A.1 and A.2). The study was approved by the University of Manchester Proportionate University Research ethics committee (No. 2018-5108-7285). Since some students opted out of the study and they became ineligible for this study, we could not use their teams in our analysis.

3.2. Course details

The second-year undergraduate software engineering offering at the University of Manchester consists of two one semester-long modules, each with a substantial project component (60%–70% module grade).

The practical element of Software Engineering 1 (SE1) delivered in Semester 1, comprises three consecutive three-week team exercises each with progressively less scaffolding and a greater requirement for collaboration. Students are asked to modify a large existing open-source codebase (Stendhal Arianne Project, 2017, version 1's release 26 in 2017, release 28 in 2018, and release 31 in 2019, respectively) by fixing bugs (Exercise 1, EX1), adding new functionality (Exercise 2, EX2) and refactoring parts of the system (Exercise 3, EX3). Students remain in the same teams throughout the module, are required to collaborate using the repository management platform GitLab and are introduced to the continuous integration tool Jenkins. Each iteration uses a recent release of the Stendhal codebase and the assignments are updated every year.

Software Engineering 2 (SE2), delivered in Semester 2, includes a 10-week team project (P1) component that requires a greater development effort than SE1 and builds on a skeleton codebase provided by the course team. Students are reassigned to new teams when they begin SE2 and are tasked with developing an MVC Java Spring event management application "Eventlite". Requirements are released on a weekly basis and teams are required to self-organise to meet the assessment objectives.

A key aspect of both courses is that every team starts from the same initial codebase and completes the same assignment. All of the teams develop their code using the Git version control system with GitLab. A version control system is a key tool that allows developers to track changes to their code and to maintain parallel versions of their project (branches) in a single location (repository). Students "commit" changes to their repository after completing an atomic piece of work and share their changes with the rest of their team by "pushing" them to a remote repository hosted on GitLab, to which the whole team has access.

Both courses include measures intended to deter social loafing in student teams. In SE1, TAs (Teaching Assistants) assign a team grade at a marking interview at the end of each exercise with all team members present. An individual student's marks may be reduced from the team grade if one or more of the following conditions are met: (i) they do not make a meaningful commit during the exercise (0 marks awarded), (ii) they do not attend the marking session (50% team marks awarded) or (iii) the team highlights an individual's reduced/problematic contribution to the course leader and the complaint is upheld on further investigation (discretionary, proportionate reduction applied by the course leader). In SE2, TAs assign a team-level grade at a showcase at the end of the course with all team members present. Individual student marks may be reduced from the team grade if one or more of the following conditions are met: (i) they cannot demonstrate their contribution at the show-case (0 marks awarded) or (ii) the team highlights their small contribution to the course leader and the complaint is upheld on further investigation (discretionary, proportionate reduction to the individual's mark applied by the course leader in consultation with the team). In these cases, the student did not get the same marks as other team members. Additional checkpoint sessions were introduced into week 5 and week 10 of SE2 from academic year 2018/2019 (Eraslan et al., 2020). From this cohort onward, students who cannot demonstrate their contribution at a checkpoint lose 20% of the project grade. A difference between the approach of SE1 and SE2 is that when an individual's contribution is reduced in SE2, the lost marks are redistributed equally among the rest of the team.

3.3. Data sources

To address our research questions, we extracted, combined and anonymised data from the following sources using custom python scripts.

3.3.1. Student team git repositories

Team project repositories were hosted on GitLab. To generate an analysis-ready data-set of individual-level contributions, information about team membership and the commits contributed by each student was extracted from remote team projects and combined using custom python scripts. PyDriller v1.15.5 (Spadini et al., 2018), a python framework for mining software repositories was used to extract basic commit-level data from the git log, while python-gitlab library v2.5.0 was used to retrieve team membership information via the GitLab API (Version 12.10). The GitPython library v3.1.10 was used to calculate lines added and lines removed counts for each commit which ignored blank lines, white-space changes and differences in carriage returns.

For the purposes of marking, students were instructed to set their Git user and email address to the name and email matching their university IT account. A number of students deviated from this guidance and used multiple Git configurations. In most cases, it was possible to unambiguously identify the committer and we manually constructed a key mapping multiple Git configurations to a single GitLab ID. Only a small number of commits could not be associated with a known student user. These were excluded from our analysis.

3.3.2. Student grades

Team-level grades were obtained from the course team. In line with the terms of our ethical approval, grades were approximate — binned at 2.5% intervals for statistical disclosure control e.g. grades of 95.5% and 97.5% were both approximated to 97.5%, while grades of 98.0% and 100% were both approximated to 100.0%. The course team also provided an indicator (yes–no) variable to identify teams where one or more members received a reduced grade as a result of reduced/non-contribution.

3.3.3. Source code analysis results

To assess the quality of the source code generated by the students we conducted a static analysis of the final project generated by each team using SonarQube (Community Edition, Version 8.4.1 (build 35646)).

Each team project was cloned, built and analysed using a custom Jenkins job (Jenkins Version 2.268) which invoked the SonarQube Scanner plugin (v.2.11). SE1 projects were built under Java v1.7 (v1.8, 2019) using Ant (Ant Plugin v1.11), while SE2 projects were built under Java v1.7 (v11, 2019) using Maven (Maven Integration Plugin 3.6).

All analyses were conducted using the default Sonar Way ruleset without modification. It was only possible to generate source code metrics with SonarQube if the project was compiling successfully. Only teams with a passing end of project build (final commit to main or up to 2 commits earlier) were included in this analysis. Data was extracted and anonymised via the SonarQube API using a custom python script.

3.4. Definition of variables

3.4.1. Student contributions

Students did not record their contributions using function points or link their code contributions to specific issues in a consistent way during their projects. As a result, like Nguyen et al. (2020) and Ntirandekura and Eude (2018) we use lines of code added or changed as a measure of contribution, while acknowledging the limitations of this metric as a measure of developer productivity (Jaspan and Sadowski, 2019).

We measured each student's absolute contribution to a given assignment by counting the number of lines of code they added to the team's Git repository during the assignment period. Total lines of code added were summed over all commits authored by an individual except merge commits and revert commits. Additions of large third party libraries such as bootstrap and jquery were ignored. Commented lines were included as a line addition but blank lines, white-space changes and differences in carriage returns were not.

Following a similar approach to Buffardi (2020) we defined each student's relative contribution as the proportion of their fair share of LOC contributed to the repository. We express this proportion as a percentage. In the context of this relative contribution metric, 0% and 100% are natural thresholds for defining non-contribution and adequate contribution. Applying a tolerance of 10% to these thresholds, we used the following cut-points to categorise students as "under-" ($<$ 90% fair share), "adequate" (90%−110% fair share), and "over-" contributors ($>$ 110% fair share) and to identify non-contributors (0% fair share) and free-riders ($<$ 10% fair share).

3.4.2. Inequality metrics

A small number of prior studies have used economic inequality metrics to quantify inequality of contribution in student software engineering teams. Hamer et al. (2020, 2021b) calculated a set of inequality metrics (Gini coefficient, Theil Index, Hoover Index, 20/20 ratio, Palma ratio, 50/50 ratio) from the commits, merges and code churn of five small student cohorts (N Teams 3–5, Team size: 3–6), while Chounta et al. (2017) calculated Gini coefficients for 6 student projects at a 5-day intensive software development camp. We use the same approach, but draw on applied social research to select metrics for our study. We explain our selected metrics later in the section.

"Inequality of coding contribution" is a team-level characteristic. To generate a team-level inequality metric, we must combine information about individual-level contributions in an appropriate way. Any relationships between inequality of contribution and team/individual outcomes are likely to be sensitive to this team-level operationalisation of "inequality" (Arthur et al., 2007 as cited in Andrés et al., 2011). As a result, we quantified team inequality in a number of different ways, using a panel of metrics.

A simple approach used by many applied researchers is to consider summary statistics such as the minimum, maximum, mean or variance of team members' individual attributes as a team-level metric and to select the most appropriate summary statistic based on domain knowledge about the type of task the team is involved in Andrés et al. (2011).

Another approach from group diversity research is to conceptualise differences within groups as separation, variety or disparity (Harrison and Klein, 2007). A different set of diversity metrics is recommended for each conceptualisation. Separation refers to the distance between team members on a continuum, e.g. size of coding contribution, and is often measured using the standard deviation (SD) or mean Euclidean distance (MED). Variety refers to diversity within a team with respect to a categorical variable, e.g. gender, and is not applicable to our case study. Disparity refers to the unequal distribution of a resource among team members, e.g. income or coding contribution, and is typically measured using economic inequality indices such as the Coefficient of Variation (CV), Gini Coefficient (GC), Theil *T* Index (TI).

Drawing on these two approaches, we included the following candidate metrics in our panel:

- *Summary statistics*: To look at the impact of under-contribution in teams, we have included the minimum relative contribution (by LOC) of team members in our panel. To look at the impact of over-contribution, we have included the maximum relative contribution (by LOC) of team members and the ratio of 1st/2nd largest relative contribution (by LOC). To look at the effect of inequality of contribution within the team, we have included the median relative contribution (by LOC).
- *Indices of separation:* Standard deviation (SD) of LOC added by individual team members, mean euclidean distance (MED) of LOC added by individual team members.
- *Indices of disparity:* Coefficient of variation (CV), Theil Index (TI), Gini coefficient (GC). In each case, these metrics were calculated using number of lines of code added by individual team members as the data source.
  - *Coefficient of Variation (CV):* CV is defined as the standard deviation divided by the mean (Harrison and Klein, 2007). It reflects the fact that diversity is less meaningful when the resource is generally plentiful (Sorenson, 2002, cited in Harrison and Klein, 2007). For example, a standard deviation of 100 LOC in contribution within a software development team is indicative of greater disparity in a team where average contributions are small compared with a team where average contributions are large.
  - *Theil T Index (TI):* TI is an entropy based inequality metric derived using information theory; it is the difference between the maximum theoretical entropy (ln(N)) and the observed entropy of the data where *N* represents the sample size (Cowell, 2011). Values of TI vary from 0 (complete equality) to ln(N) (maximum inequality).
  - *Gini Coefficient (G):* G captures the degree to which a resource is distributed equally among members of a population. Ranging in value from 0 to 1, a Gini coefficient of 0 indicates perfect equality while a value of 1 indicates perfect inequality with a single individual possessing all of the resource e.g. income, code contribution etc. Cowell (2011). In our context, the Gini Coefficient is the average difference between the contributions of all possible pairs of students as a proportion of total contributed LOC (Cowell, 2011). As the Gini coefficient is a relative indicator, there is no single agreed interpretation applicable to all contexts. As a guide, we adopt the following categorisation used by several researchers in health services research: < 0.2 (high equality), $0.2-0.3$ (moderate inequality), $0.3-0.4$ (low equality) and > 0.4 (extreme inequality) (Cao et al., 2020; Erdenee et al., 2017; Meskarpour-Amiri et al., 2014).

All of the candidate diversity metrics take a minimum value of 0 when all members of a team make equal contributions (complete equality). The metrics of separation SD and MED reach a maximum when the contributions of team members are polarised, e.g. a 6-member team where three team members make no contribution and three team members make equal contributions of 1000 LOC (Harrison and Klein, 2007). In contrast, the metrics of disparity all take their maximum value when all of the code is contributed by one team member and the rest of the team contribute nothing (Wei et al., 2016).

These upper bounds depend on team size and are affected by small group bias, meaning that diversity is systematically underestimated in smaller teams; Inappropriate conclusions can be drawn if they are applied to samples of mixed team size like ours (Biemann and Kearney, 2010). As a result, we use bias-corrected definitions of these metrics as recommended by Biemann and Kearney (2010).

To limit the number of metrics included in our study, we calculated correlation coefficients between candidate metrics with the same conceptualisation and removed variables to eliminate correlations > 0.90 to generate a final panel of inequality metrics for analysis. Our final panel of inequality metrics is as follows: Minimum relative contribution; Median relative contribution; Ratio of 1st/2nd largest relative contribution (LOC) by individual team members; Standard deviation (SD); Gini Coefficient (G).

3.5. Software quality metrics

The quality of the source code generated by the student teams was measured using a standard panel of metrics generated using the code quality analysis platform SonarQube. We selected SonarQube as it is a popular source code analyser in industry (Raibulet and Arcelli Fontana, 2018), has been utilised in software engineering courses (Raibulet and Arcelli Fontana, 2018; Bai et al., 2018) and has also been used in a number of software engineering education research studies (Nguyen et al., 2020; Hamer et al., 2021a). SonarQube 8.4.1 provides 73 software quality metrics across nine different domains: complexity, duplications, issues, maintainability, quality gates, reliability, security, size and tests.

Quality gates were not utilised in the student projects and elements of build health and testability formed part of the project mark schemes. As a result, metrics from the quality gate and test domains were not used in our study. In SE1, teams worked with a large existing codebase and in SE2, teams built on skeleton code provided by the course team. As a result, calculation and analysis of project level metrics such as cognitive complexity was restricted to SE2. In addition, raw rule violation metrics provided by SonarQube were converted to delta values capturing the change in the metric between the start and the end of the assignment and where applicable, were also normalised by the number of lines of code added by the team during the assignment.

3.6. Team-perceived unequal contribution

In both modules, grade reductions were made if a student made no contribution to an assignment or if the team escalated a low or disruptive contribution to the course leader successfully. As the course team confirmed that the majority of escalations concerned low contributions, we used a reduction in one or more team member's individualised grade as an indicator of perceived inequality within the team.

In SE2, check-point sessions were introduced from 2018 onward, meaning that grade reductions could be applied before the end of the project. These check-point sessions aim to ensure that all the team members are actively working on a given project. A student who does not attend the check-point session and cannot

Table 1
Descriptive statistics of individual-level inequality metrics. The total numbers of students in SE1 (EX1, EX2, EX3) are 231, 225 and 253 in 2017, 2018, and 2019 respectively where the total numbers of students in SE2 (P1) are 227, 219 and 245 in 2017, 2018 and 2019 respectively. The number of students is provided for each specific metric along with its ratio to the total number of students in parentheses — Abbreviations: FS — fair share, PCT — percentage of contributed LOC, PCT1 — percentage of LOC contributed by highest contributor, PCT2 — percentage of LOC contributed by second highest contributor.

Metric	Cohort	EX1	EX2	EX3	P1
Non-contributor (FS = 0)	2017	1 (0.4)	3 (1.3)	10 (4.3)	4 (1.8)
	2018	8 (3.6)	12 (5.3)	15 (6.7)	3 (1.4)
	2019	4 (1.6)	15 (5.9)	5 (2.0)	3 (1.2)
Undercontributor (FS < 90)	2017	122 (53)	138 (60)	136 (59)	121 (53)
	2018	125 (56)	117 (52)	133 (59)	115 (53)
	2019	146 (58)	138 (55)	130 (51)	128 (52)
Adequate contributor (90 ≥ FS < 110)	2017	26 (11)	16 (6.9)	21 (9.1)	31 (14)
	2018	26 (12)	24 (11)	17 (7.6)	28 (13)
	2019	14 (5.5)	24 (9.5)	21 (8.3)	32 (13)
Over contributor (FS ≥ 110)	2017	83 (36)	77 (33)	74 (32)	75 (33)
	2018	74 (33)	84 (37)	75 (33)	76 (35)
	2019	93 (37)	91 (36)	102 (40)	85 (35)
Severe undercontributor (FS < 25)	2017	49 (21)	39 (17)	44 (19)	23 (10)
	2018	31 (14)	32 (14)	46 (20)	18 (8.2)
	2019	67 (26)	27 (11)	38 (15)	20 (8.2)
Severe over contributor (FS ≥ 200)	2017	25 (11)	27 (12)	32 (14)	24 (11)
	2018	21 (9.3)	24 (11)	33 (15)	22 (10)
	2019	38 (15)	27 (11)	26 (10)	26 (11)
Free-rider (FS < 5)	2017	9 (3.9)	6 (2.6)	11 (4.8)	7 (3.1)
	2018	10 (4.4)	13 (5.8)	16 (7.1)	5 (2.3)
	2019	17 (6.7)	16 (6.3)	10 (4.0)	6 (2.4)
One person team (PCT1/PCT2 ≥ 200)	2017	10 (4.3)	13 (5.6)	13 (5.6)	9 (4.0)
	2018	13 (5.8)	9 (4.0)	15 (6.7)	8 (3.7)
	2019	13 (5.1)	8 (3.2)	6 (2.4)	8 (3.3)
One person team (PCT ≥ 50)	2017	3 (1.3)	6 (2.6)	8 (3.5)	5 (2.2)
	2018	10 (4.4)	4 (1.8)	11 (4.9)	3 (1.4)
	2019	7 (2.8)	3 (1.2)	3 (1.2)	2 (0.8)

demonstrate their individual contribution loses 20% of the project grade. Whilst there is a possibility that the checkpoints might have reduced inequality, we do not formally analyse this here, focusing instead on the team-perceived unequal contribution, as reported by the team members.

3.7. Statistical analysis

All statistical analysis was carried out using the R statistical software (R Core Team, 2020). As this is an exploratory study, we do not correct p-values for multiple testing (Bender and Lange, 2001).

4. Results

4.1. Contribution inequality in student software engineering teams

RQ1: How uniform is the distribution of coding contributions in student software engineering teams?

To address this question, we calculated individual-level contribution patterns (see Table 1) and a panel of team-level inequality metrics (see Tables 2 and 3) to capture the distribution of coding effort within our student teams.

4.1.1. Individual-level contribution style

Individual relative contributions were strongly right-skewed with the majority of individuals contributing less than their fair share and a small minority contributing much more.

The median relative contribution ranged from Mdn 66.5%, (P5-P95 = 3.9%–300%, EX1 2019) to Mdn 87.5%, (P5-P95 = 6.58%–244%, EX1 2017) indicating that for each assignment, across all cohorts, 50% of students contributed less than or equal to 87.5% of their fair share.

Table 2
The number of different team types in different cohorts for SE1 (EX1, EX2, EX3) and SE2 (P1). Descriptive statistics of team types in the cohorts. The total numbers of teams in SE1 are 34, 37 and 37 in 2017, 2018, and 2019 respectively where the total teams of students in SE2 are 37, 36 and 40 in 2017, 2018 and 2019 respectively. The number of teams is provided for each specific metric along with its ratio to the total number of teams in parentheses. – Abbreviations: FS — fair share, OPT (Abs50) — one person team based on absolute contributions, OPT (Rel100) — one person team based on relative contribution.

Metric	Cohort	EX1	EX2	EX3	P1
Non-contributor	2017	1 (2.9)	3 (8.8)	9 (26)	3 (8.1)
	2018	8 (22)	10 (27)	11 (30)	3(8.3)
	2019	4 (11)	9 (24)	5 (14)	3(7.5)
Freerider (FS<5)	2017	7 (21)	5 (15)	10 (29)	6 (16)
	2018	10 (27)	11 (30)	12 (32)	5(14)
	2019	16 (43)	10 (27)	9 (24)	5(12)
Equal Team (All FS ≥ 90)	2017	0 (0)	0 (0)	0 (0)	0 (0)
	2018	0 (0)	0 (0)	0 (0)	0(0)
	2019	0 (0)	0 (0)	0 (0)	0(0)
One Person Team (Rel100)	2017	10 (29)	13 (38)	13 (38)	9 (24)
	2018	13 (35)	9 (24)	15 (41)	8(22)
	2019	13 (35)	8 (22)	6 (16)	8(20)
One Person Team (Abs50)	2017	3 (8.8)	6 (18)	8 (24)	5 (14)
	2018	10 (27)	4 (11)	11 (30)	3(8.3)
	2019	7 (19)	3 (8.1)	3 (8.1)	2(5.0)

When we classified students as "under-" (<90% fair share), "adequate" (90%–110% fair share), and "over-" contributors (>110% fair share) using ad-hoc cut-points, as shown in Table 1, we found that the majority of students were under-contributors (ranging from 51.4% (130/253, EX3 2019) to 59.7% (138/231, EX2 2017) across all assignments and cohorts, while a minority were adequate contributors (ranging from 5.53% (14/253, EX1 2019) to 13.7% (31/227, P1 2017) across all assignments and cohorts) or

Table 3
The median of each team-level inequality metrics along with the 5th and 95th percentiles in parentheses — Abbreviations: FS — fair share.

Metric	Cohort	EX1	EX2	EX3	P1
Ratio of 1st/2nd Contributions	2017	1.50 (1.04, 10.57)	1.77 (1.11, 3.93)	1.58 (1.10, 6.36)	1.55 (1.06, 3.61)
	2018	1.39 (1.05, 7.08)	1.32 (1.03, 3.03)	1.77 (1.04, 6.24)	1.58 (1.04, 3.91)
	2019	1.71 (1.03, 2.97)	1.49 (1.04, 3.65)	1.34 (1.04, 2.99)	1.46 (1.07, 2.74)
Smallest Contribution (%FS)	2017	13 (1, 53)	19 (0, 41)	17 (0, 48)	29 (0, 57)
	2018	23 (0, 68)	16 (0, 44)	15 (0, 47)	28 (0, 63)
	2019	6 (0, 19)	27 (0, 52)	20 (0, 51)	30 (0, 56)
Median Contribution (%FS)	2017	90 (22, 111)	76 (41, 110)	71 (32, 112)	86 (29, 107)
	2018	79 (35, 113)	84 (48, 120)	74 (26, 100)	85 (64, 109)
	2019	71 (45, 108)	85 (61, 116)	90 (47, 118)	84 (58, 108)
Gini Coefficient	2017	0.44 (0.29, 0.83)	0.47 (0.31, 0.69)	0.54 (0.28, 0.74)	0.37 (0.23, 0.72)
	2018	0.45 (0.16, 0.75)	0.45 (0.25, 0.61)	0.54 (0.30, 0.79)	0.41 (0.16, 0.59)
	2019	0.56 (0.44, 0.71)	0.39 (0.26, 0.67)	0.42 (0.24, 0.64)	0.40 (0.19, 0.59)
SD Contribution (Abs)	2017	46 (30, 580)	339 (161, 637)	369 (132, 1,243)	527 (183, 1,715)
	2018	97 (30, 444)	147 (68, 454)	402 (169, 1,122)	570 (188, 1,361)
	2019	100 (56, 210)	191 (80, 393)	315 (120, 842)	600 (236, 1,242)

over-contributors (ranging from 32% (74/231, EX3 2017) to 40.3% (102/253, EX3 2019) across all assignments and cohorts).

Looking at more extreme forms of under and over contribution in Table 1, we observed that the proportion of students contributing <25% of their fair share ranged from 8.16% (20/245, P1 2019) to 26.5% (67/253, EX1 2019), while the proportion of students contributing more than double their fair share ranged from 9.33% (21/225, EX1 2018) to 15% (38/253, EX1 2019).

As can be seen in Table 1, a very small minority of students made no contribution (0.433% (1/231, EX1 2017) to 6.67% (15/225, EX3 2018), or were free-riders 2.28% (5/219, P1 2018) to 7.11% (16/225, EX3 2018).

A small minority of students made a very large contribution to their project (see Table 1). The prevalence of team members who generated more than 50% of their project's code ranged from 0.816% (2/245, P1 2019) to 4.89% (11/225, EX3 2018), while the prevalence of highest contributors who added at least twice the code of the second most prolific contributor, ranged from 2.37% (6/253, EX3 2019) to 6.67% (15/225, EX3 2018).

4.1.2. Team-level inequality

Equal teams where every team member contributed at least 90% of their fair share were not found in our cohorts. Fig. 1 visualises the percentages of different team types with unequal work distribution in different cohorts in SE1 (EX1, EX2, and EX3) and SE2 (P1).

As can be seen in Table 2, the number of teams affected by non-contribution, free-riding or both varied by assignment and cohort, but was generally modest ranging from 14.7% (5/34, EX2 2017) to 43.2% (16/37, EX1 2019) in SE1 and 12.5% (5/40, P1 2019) to 16.2% (6/37, P1 2017) in SE2.

"Absolute one-person teams" where one team member contributed the majority of the code were observed in all assignments and cohorts. As shown in Table 2, in SE1, the prevalence of one-person teams varied by assignment and cohort and ranged from 8.11% (EX2 2019; EX3 2019) to 29.7% (EX3 2018). In SE2, the prevalence of one-person teams fell each year from 14% in 2017 to 5% in 2019. The proportion of one-person teams was lowest in 2019 for all assignments except EX1.

"Relative one-person teams" where the largest contributor added at least twice the code of the second largest contributor, were more common. This affected 16.2% (EX3 2019) to 40.5% (EX3 2018) in SE1 and 20% (P1 2019) to 24.3% (P1 2017) in SE2 (see Table 2). The ratio of the 1st/2nd largest contribution ranged from Mdn 1.32, (P5-P95 = 1.03–3.03) in EX2 2018 to Mdn 1.77, (P5-P95 = 1.11–3.93; P5-P95 = 1.04–6.24) in EX2 2017; EX3 2018 in SE1 and from Mdn 1.46, (P5-P95 = 1.07–2.74) in P1 2019 to Mdn 1.58, (P5-P95 = 1.04–3.91) in P1 2018 in SE2 (see Table 3).

Table 3 shows that the Gini Coefficient ranged from Mdn 0.392, (P5-P95 = 0.261–0.674) in EX2 2019 to Mdn 0.56, (P5-P95 = 0.441–0.712) in EX1 2019 in SE1 and from Mdn 0.375, (P5-P95 = 0.234–0.721) in P1 2017 to Mdn 0.407, (P5-P95 = 0.161–0.594) in P1 2018 in SE2. In all years, the Gini coefficient increased from SE1 EX2 to SE1 EX3 and decreased from SE1 EX3 to SE2. That is, inequality of contribution increased as the exercises in SE1 got more challenging in terms of team work, but recovered when teams started from scratch on a new, longer and more substantial project.

4.1.3. Key findings

Different types of teams with unequal distribution were observed in all cohorts, including the teams with a non-contributing member, the teams with a free rider, the teams with an over-contributor who contributed the majority of the code and the teams with an over-contributor who added at least twice the code of the second-largest contributor. Most individuals in our student software engineering teams contributed less than their fair share and in all assignments, the majority of teams had "low equality of contribution" (Gini 0.3—0.4) or "extreme inequality of contribution" (Gini > 0.4) suggesting that unequal contribution is a material issue for student software engineering teams.

4.2. Persistence of individual contribution style between team projects

RQ2: Does over-contribution or under-contribution made by an individual affect their contribution in a subsequent software engineering team project?

For the purposes of this research question, we restricted our analysis to data from exercise three (EX3) in SE1 as this was the final team exercise of the module, with the greatest scope for the students to organise their own work and therefore most likely to impact on behaviour in SE2.

To address this question, we calculated Spearman rho correlations to assess the strength and direction of the association between students' relative coding contribution in SE1 and their relative contribution in SE2. T-tests were carried out to test the hypothesis that each correlation coefficient was significantly different from zero.

We found a significant, but extremely weak correlation between individuals' relative contribution in SE1 and SE2 in 2017 and 2018, but not 2019. The correlation coefficient for this association ranged from $r_s = 0.091$, $p = 0.16$, N = 241 (2019) to $r_s = 0.22$, $p = 0.0017$, N = 203 (2018).

Based on their relative contribution in SE1 we classified students as "under-" (< 90% fair share), "adequate" (90%–110% fair

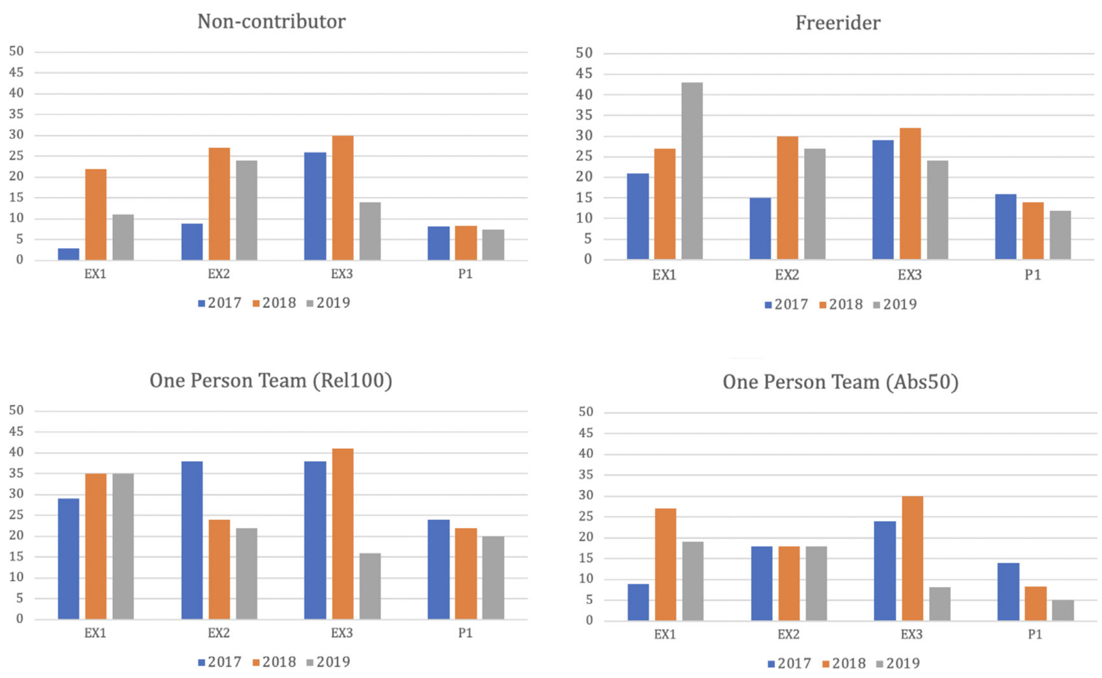

Fig. 1. The percentage of different types of teams with unequal work distribution in different cohorts in SE1 (EX1, EX2, and EX3) and SE2 (P1).

share), or "over-" contributors (> 110% fair share) using ad-hoc cut-points. We then calculated the prevalence of these categories in each module and transition probabilities for change in category membership between SE1 and SE2. These results are shown as an alluvial plot in Fig. 2 where black vertical bars are divided based on the percentage of the over-contributors, adequate contributors and under-contributors in SE1-EX3 and SE2.. We found that in each case the probability of change was high; 41–44.9%, 52.9–65.7% and 85.7–94.7% of under-, adequate and over-contributors changed contribution category from SE1 to SE2. For example, 51.7% of students changed their contribution style from SE1-EX3 to SE2 in 2017. Specifically, 41% of the under-contributors, 95% of the adequate contributors, and 57% of the over-contributors in SE1-EX3 changed their contribution style in SE2.

Finally, we calculated the prevalence and transition probabilities for contribution styles of particular interest: non-contributors, free-riders and absolute over-contributors. A small number of students were non-contributors, free-riders or absolute "one-person teams" in both modules (SE1-EX3 & SE2 P1). For example, the number of students who were free-riders in EX3 and who went on to make a similar contribution in P1 was 1/8 (12.5%) in 2017, 1/7 (14.3%) in 2018 and 1/5 (20.0%) in 2019. The proportion of students who were noncontributors in EX3 and P1 was 1/7 (14.3%) in 2017, 0/6 (0.0%) in 2018 and 0/2 (0.0%) in 2019. Finally, the proportion of students contributing ≥50% of contributed LOC (one-person teams) in EX3 and P1 was 0/8 (0.0%) in 2017, 0/10 (0.0%) in 2018 and 0/3 (0.0%) in 2019 suggesting that substantial over-contribution in both modules was uncommon. The denominators differ from Section 4.1, as this analysis only considers students who have data for both modules, as not everyone who had data on both modules consented to take part.

4.2.1. Key findings

Individual-level contribution styles were not strongly persistent between modules. When extreme contribution styles such as free-riding and being a one-person team were observed in SE1 (EX3), they were rarely repeated by the same students in SE2. This finding means that when a student under contributes to one project, it does not necessarily mean that the student will under contribute to another project. The contribution style of a student depends on the situation and may differ from one situation to another.

4.3. Relationship between team inequality of contribution and individuals' future contribution

RQ3: Does the degree of inequality of contribution experienced by a student in their team in a software engineering team project affect their contribution in a subsequent software engineering team project?

To address this question, we conducted a number of different analyses. First, we calculated Spearman correlations to assess the strength and direction of the association between team-level inequality experienced by students in SE1 EX3, measured using team-level core inequality metrics and their relative contribution in SE2. T-tests were carried out to test the hypothesis that each correlation coefficient was significantly different from zero.

As shown in Fig. 3, we observed a single weak statistically significant ($p < 0.05$) correlation between individuals' relative contribution in SE2 and the median relative contribution of their team in SE1 (2018). However, this association was not observed in more than one cohort.

We then repeated this analysis, using change in relative contribution from SE1 to SE2 as the outcome. A small number of students who contributed in SE2, but not SE1 EX3 were arbitrarily assigned a change in relative contribution of +100%. As shown in Fig. 3, we observed a single weak statistically significant correlation ($p < 0.05$) between individual-level change in relative contribution between SE1 and SE2 and the 1st/2nd PFS ratio of their team in SE1 (2017). However, this association was not observed in more than one cohort.

Finally, we divided students into three groups based on their relative contribution in SE1: "Under contributors" (< 90%), "Adequate Contributors" (90%−110%) and "Over Contributors" (> 110%). We then fitted a series of logistic regression models to look at the association between inequality experienced in SE1 and the odds of belonging to a different contribution category in SE2. Despite observing a small number of statistically significant associations ($p < 0.05$) between inequality in SE1 and contribution

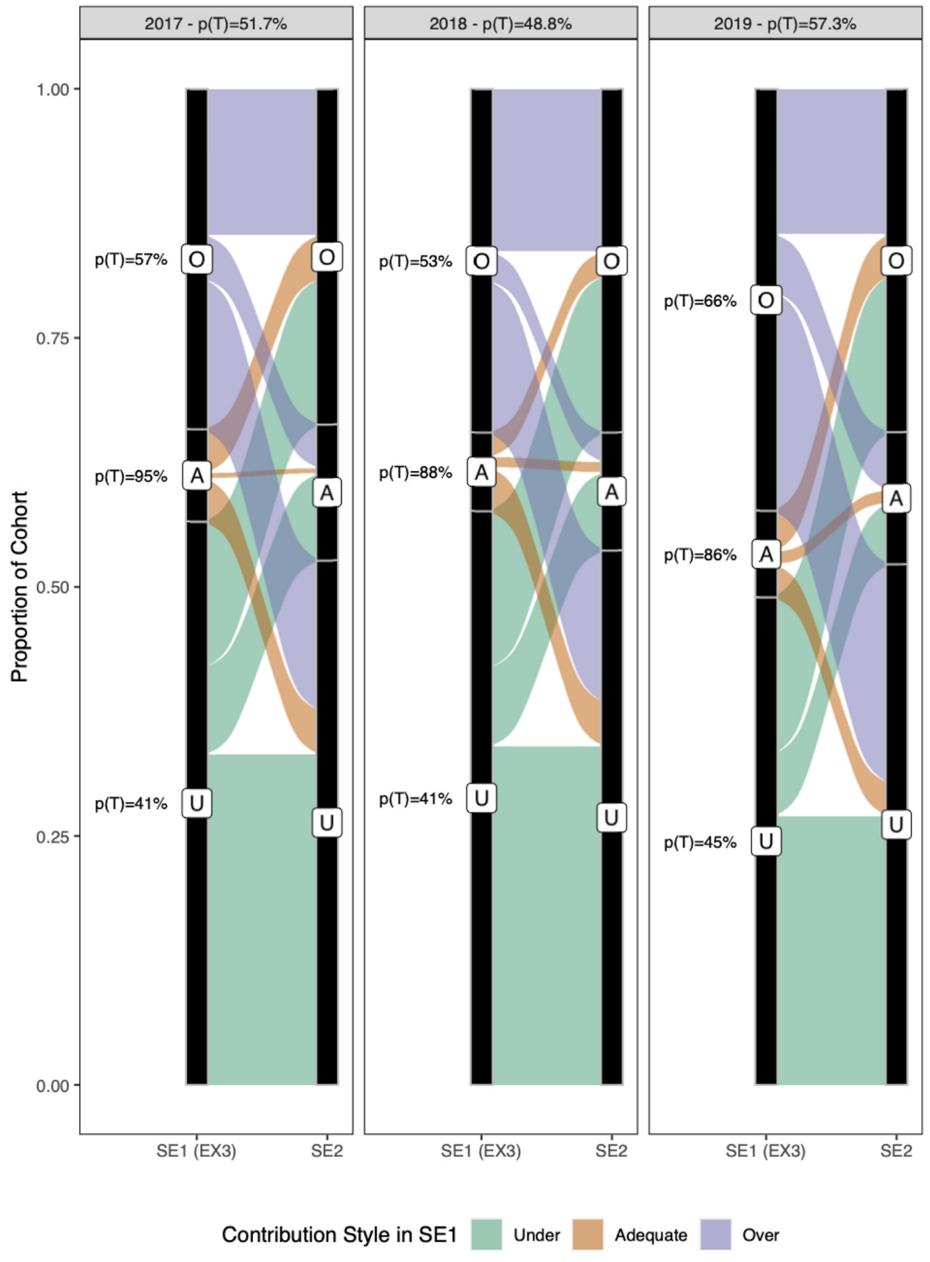

Fig. 2. Alluvial plot to show changes in individual-level contribution style between SE1-EX1 and SE2-P1 for each cohort. Percentages printed to the left of each stratum indicate the probability of individuals in that stratum changing contribution style between SE1 and SE2. The overall probability (%) of any individual within the cohort changing contribution style between SE1 and SE2 is indicated next to the cohort year. — Abbreviations: O — Over-contributors, A — Adequate contributors, and U — Under-contributors.

category in SE2, none of these were observed in more than one cohort.

4.3.1. *Key findings*

In general, individual-level contributions in SE2 were not associated with team-level inequality in SE1. Changes in individual level contributions between SE1 and SE2 were not associated with team-level inequality in SE1. This finding suggests that when a student experiences unequal contribution in their team in one project, it does not affect their individual contribution in a subsequent project.

4.4. *Relationship between team inequality of contribution and team performance*

RQ4: Is there a relationship between inequality of contribution within student teams and team performance?

4.4.1. *Inequality and grade*

To identify relationships between the unequal distribution of contribution in student teams and team grade which was given based on the product they developed, we looked for associations between our set of core inequality metrics and (i) continuous team grade and (ii) achieving a high pass ($> 80\%$) vs. an ordinary pass (40–80%), failing teams excluded).

To assess the strength and direction of the association between continuous inequality metrics and team grade, we calculated Spearman correlations between each pair of variables and conducted t-tests to test the hypothesis that each correlation coefficient was significantly different from zero. These results are summarised in Fig. 4.

We found a small number of significant associations between continuous inequality variables and grade in both SE1 and SE2. However, none of these were consistent across all three cohorts. For example, we observed a weak positive association between

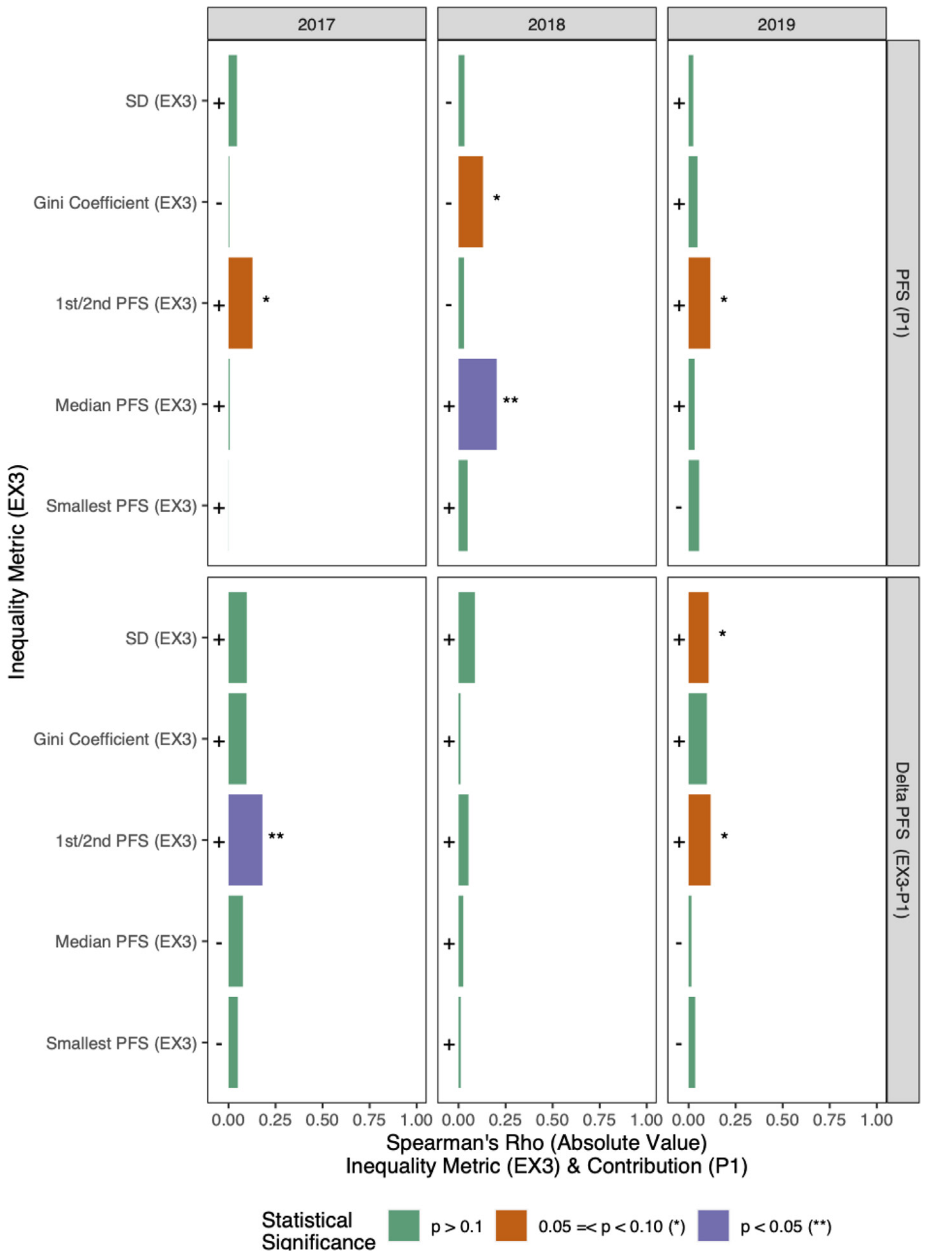

Fig. 3. Correlations between inequality metrics in SE1-EX3 and (i) individual-level relative contribution in SE2-P1 (upper part of the figure) (ii) changes in individual-level relative contribution between SE1 and SE2 (lower part of the figure). Absolute values of the Spearman's rho correlation coefficients are reported and the sign of the correlation coefficient is indicated on the y axis.

SD and grade in SE2 in 2017 ($r_s = 0.489$, $p = 0.002$, N = 37) and 2019 ($r_s = 0.412$, $p = 0.008$, N = 40), but not 2018, as shown in Fig. 4.

To find out whether grades achieved differed between teams w/without specific contribution styles e.g. free-riders and over-contributors, we used the Wilcoxon rank-sum test. We identified categorical inequality metrics with a frequency of at least 5 teams across all years for a given assignment and in each case tested the hypothesis that the median grade was equal in teams w/without the characteristic of interest. We found that in SE1-EX3 (2017) team grade was lower among teams affected by non-contribution and free-riding ($p < 0.05$), but this finding was not replicated in 2018 or 2019. In SE1, we found that the median grade was consistently lower among teams affected by free-riding in all cohorts, but this difference did not reach statistical significance in any cohort.

To assess the association between continuous inequality metrics and achieving a high pass vs an ordinary pass, we conducted Wilcoxon rank-sum tests to test the hypothesis that the median inequality metric was equal in teams w/without a high pass. We found a small number of significant associations between achieving a high pass and continuous inequality variables and in both SE1 and SE2. None of these were consistent across cohorts apart from a significant association between grade (high pass vs ordinary pass) and the SD of the team's contributions in SE1; The SD of the team's contributions was higher in teams achieving a high pass. SD is also positively correlated with the lines of code in SE1, suggesting that this may reflect underlying differences in project size rather than the inequality of contributions of these teams.

To find out whether the proportion of high passes differed between teams w/without specific contribution styles e.g. free-riders and over-contributors, we conducted chi-square tests of

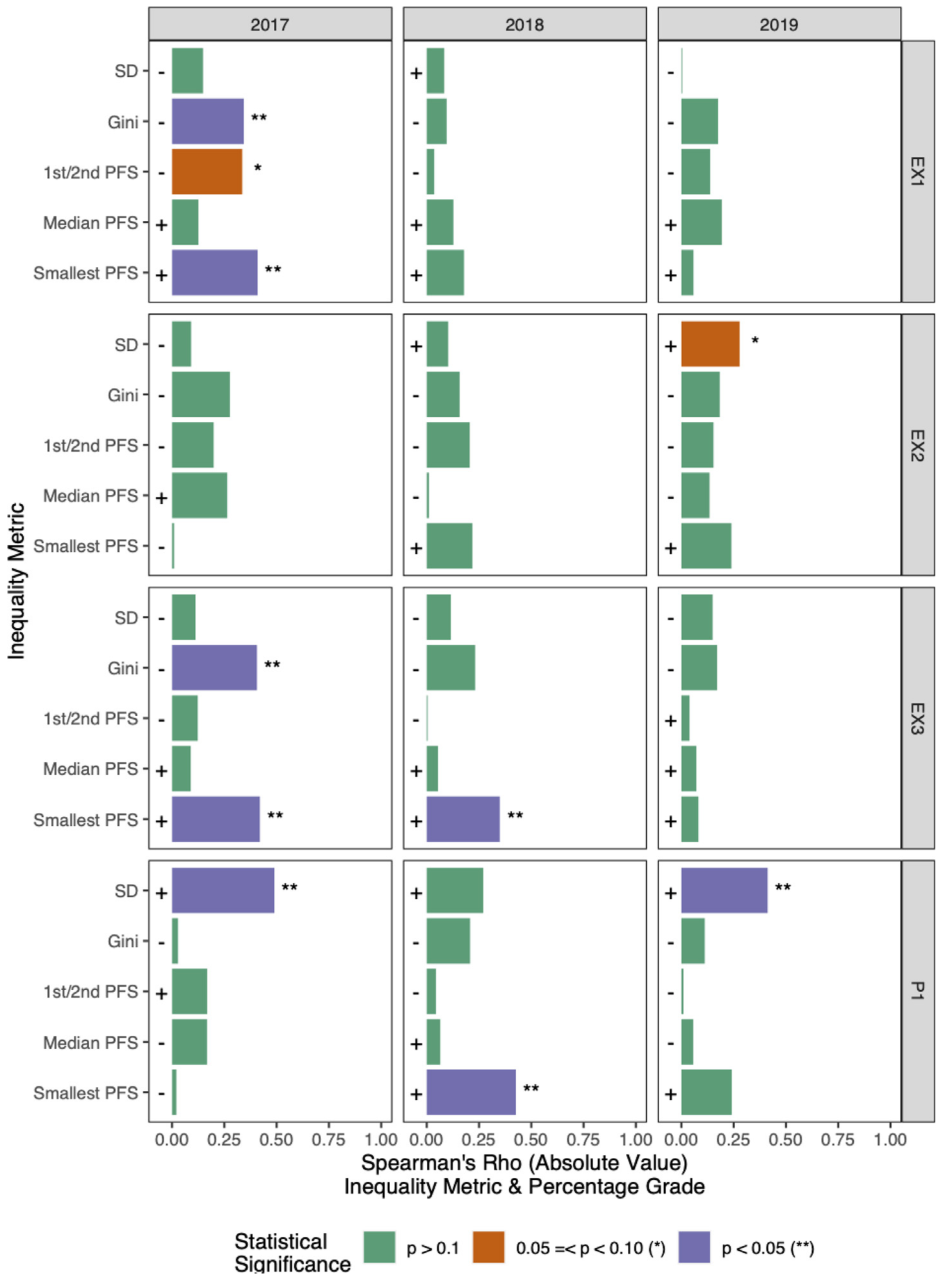

Fig. 4. Summary of correlations between continuous inequality metrics and team-level percentage grade. Absolute values of the Spearman's rho correlation coefficients are reported and the sign of the correlation coefficient is indicated on the y axis.

independence for pairs of categorical variables with all expected cell counts >= 5 and Fisher exact tests for those with any expected cell count < 5. We found few statistically significant associations between team type and achieving a high pass and none were consistent across assignment or cohort. For example, the proportion of high passes was significantly lower among teams affected by free-riding in SE1-EX3, but this finding was not replicated in 2018 or 2019.

4.4.2. Inequality and software quality

To assess the strength and direction of the association between continuous inequality metrics and continuous software quality metrics, we calculated Spearman correlations between each pair of variables and conducted t-tests to test the hypothesis that each correlation coefficient was significantly different from zero.

To find out whether software quality metrics differed between teams w/without specific contribution styles e.g. free-riders and over-contributors, we identified categorical inequality metrics with a frequency of at least 5 teams and used the Wilcoxon rank-sum test to test the hypothesis that the median software quality metric was equal in teams w/without the contribution pattern of interest. We found a number of statistically significant associations between continuous inequality metrics and software quality metrics. However, many of these were only observed in a single cohort for a given assignment.

Of all the inequality metrics examined, the SD of contributions within a team had the largest number of significant associations with software quality outcomes (Fig. 5). Two of these associations were statistically significant or marginally significant in all three years studied. In two-week assignment SE1-EX2, we found that teams with a higher contribution SD had a higher density of code-smell rule violation events in all three years. In contrast, in longer 12-week SE2, we found that teams with a higher contribution SD had a lower density of unique bug rule violations in their code.

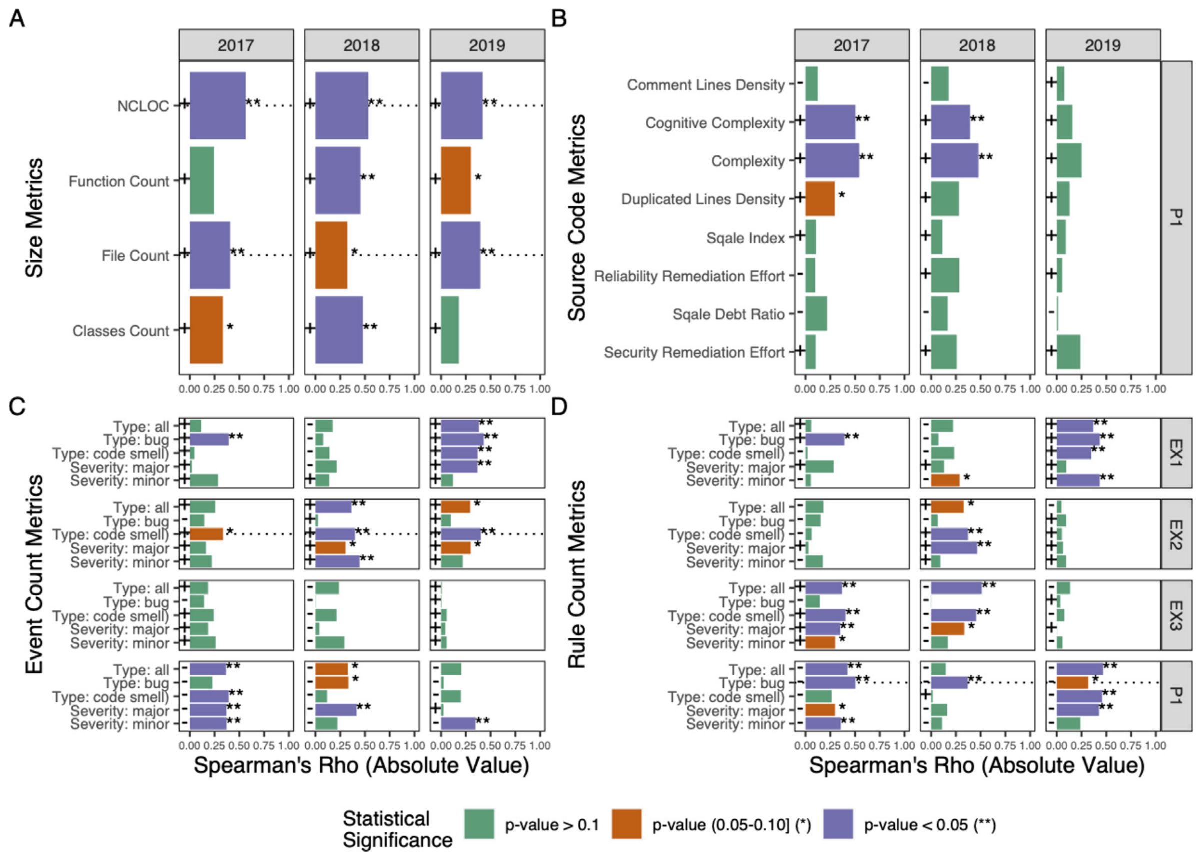

Fig. 5. Summary of Spearman's rho correlations between team-level contribution SD and source code metrics. Absolute values of correlation coefficients are reported and the sign of the correlation coefficient is annotated on the y axis. (A) Size metrics (B) Complexity, Duplications, Maintainability, Reliability (C) Issues (Number rule violation events) (D) Issues (Number of unique rules violated).

Teams with a higher contribution SD also generated larger code-bases by the end of each assignment, so these observations may reflect differing underlying relationships between software size and quality within assignments of differing type and duration.

4.4.3. *Key findings*

Four out of five of the inequality metrics analysed were not consistently associated with a lower grade or poorer code quality. Higher SD of team member contributions was associated with a greater probability of a higher pass in all three cohorts in SE2 (especially 2017 and 2019). This may occur because more knowledgeable students might be spending more time on the practical work and completing a greater proportion of it, thus leading to higher standard work on average.. Higher SD of team member contributions was also associated with a lower density of unique bugs in SE2 and a higher density of code smells in SE1 – EX2.

4.5. *Relationship between repository based metrics of unequal contribution and team perceived inequality*

RQ5: Is there a relationship between objective repository based metrics of contribution inequality and team-perceived contribution inequality (indicated by the team accepting or successfully requesting a redistribution of marks from the course team)?

To address this question, we fitted a series of univariate logistic regression models to identify which core inequality metrics were most strongly associated with team-perceived inequality of contribution. We used reduction of one or more individual's grades within a team due to poor contribution as a surrogate measure of team-perceived inequality of contribution. We restricted our analysis to SE2 (2017) as (i) the number of teams with a relevant grade reduction in SE1 was too low for statistical analysis (ii) grade reduction data was unavailable for SE1 EX3 (2017) (iii) check-point sessions were introduced to SE2 from 2018 onward, modifying the processes for grade reduction making it a poor surrogate for team-perceived inequality from this point onward.

Table 4
Logistic regression analysis of the association between objective inequality metrics calculated from git repositories and team-perceived inequality of contribution.

Inequality Metric	N	Event N	OR	95% CI	p-value
Gini	37	11	2.96	1.59, 7.14	<.001
SD	37	11	1.03	0.86, 1.21	0.74
Smallest PFS	37	11	0.92	0.86, 0.97	<.001
Median PFS	37	11	0.95	0.91, 0.99	0.006
1st/2nd PFS	37	11	1.69	0.78, 4.09	0.18

As shown in Table 4, we found that several core inequality metrics were associated with grade reduction including the Gini coefficient, the median relative contribution and the smallest relative contribution. The ratio of the 1st/2nd relative contribution and SD of contribution were not associated with grade reduction.

4.5.1. *Thresholds for team dissatisfaction*

Once we had identified a subset of the core continuous inequality variables that were associated with grade reduction, we examined each of these relationships further to identify a

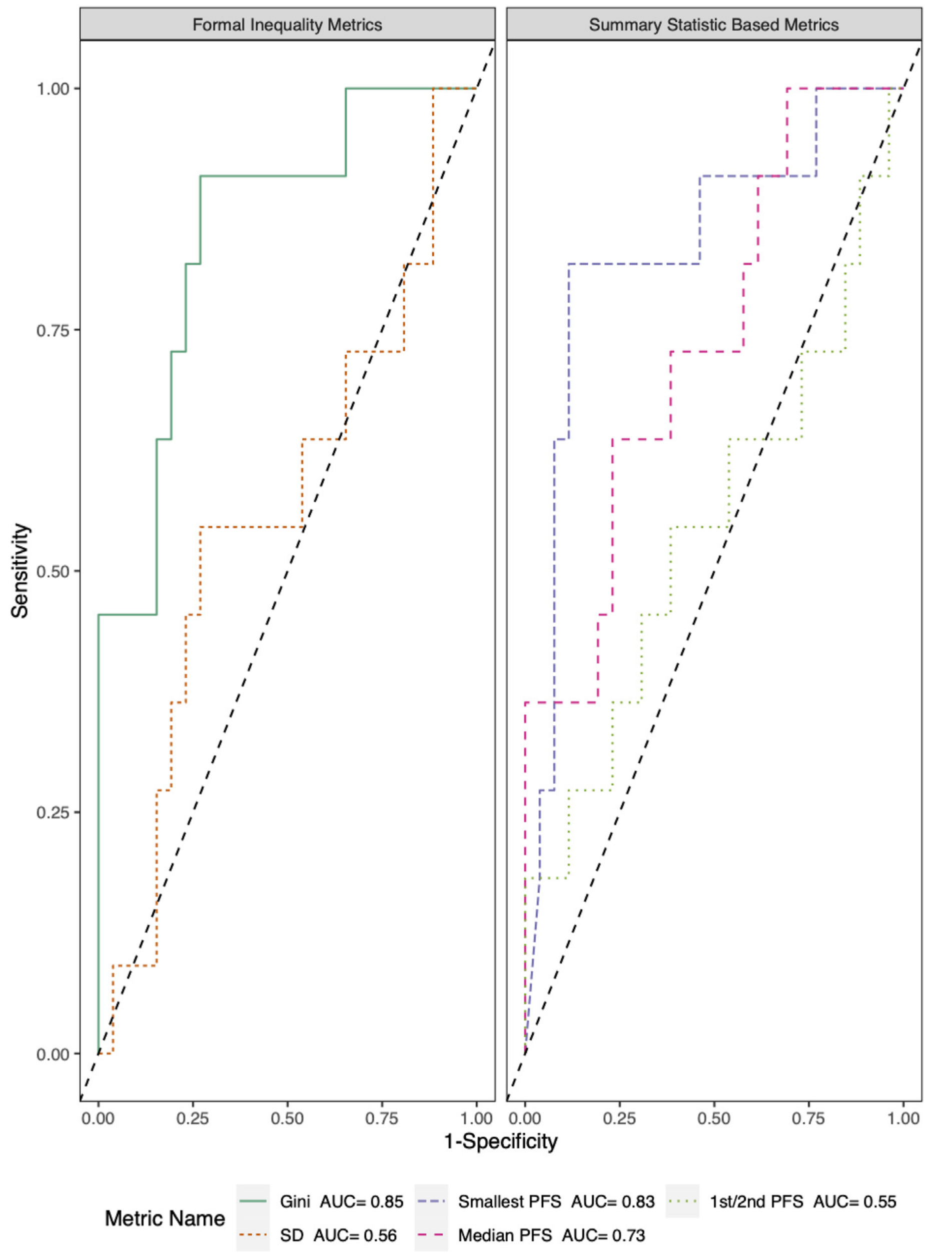

Fig. 6. ROC curves to show the association between inequality metrics and team-perceived inequality of contribution. In a ROC curve, the x-axis shows 1 - specificity (false positive fraction), and the y-axis shows sensitivity (true positive fraction). We use the area under the RUC curve (AUC) to compare the predictive ability of each inequality metrics where an AUC value closer to 1 indicates greater predictive ability.

practical threshold value indicative of an increased risk of grade adjustment and hence team perceived contribution inequality.

A variety of statistical approaches are available for estimating the optimal cut-point for a continuous variable with respect to a binary outcome. We identified optimal cut-points using Receiver Operating Characteristics (ROC) analysis (see Fig. 6). In a ROC analysis, the true positive rate is plotted against the false positive rate for predicting the binary outcome at each possible cut-off value of the predictor of interest. An optimal cut-off can then be identified by locating the point on the ROC curve which optimises a chosen index of diagnostic accuracy. A variety of indices of diagnostic accuracy are available. Some, such as the Youden Index and distance to corner (d), give equal weight to sensitivity and specificity, while others incorporate information about prevalence and the cost of each of the four possible diagnostic outcomes: true positive, true negative, false positive and false negative. For interpretability, we have chosen the Youden Index (Sensitivity + Specificity −1) which results in a cutoff that maximises Sensitivity + Specificity.

Based on AUC (higher is better), the ROC analysis showed that the Gini coefficient, closely followed by the smallest relative contribution were the strongest predictors of team-perceived unequal contribution The optimal cut points identified by ROC analysis were as follows: Gini Coefficient 0.40 (F1 0.71); smallest relative contribution 14.0% fair share (F1 0.78).

4.5.2. *Key findings*

Team-perceived inequality was more likely (i) when the contribution of the least active team-member was less than 14% of their fair-share, or (ii) when the Gini coefficient approached a value indicating extreme inequality (> 0.4). These results show the relationship between the objective repository based metrics of contribution inequality and team-perceived contribution inequality.

5. Discussion and conclusions

In this paper, we presented an exploratory analysis of the impact of unequal contribution in student software engineering teams, to inform future interventions and find out if there is any basis for warning students about the negative impact of this behaviour on themselves or their team.

We investigated various inequality metrics in our study and selected a specific set of metrics by eliminating the ones with correlations > 0.90. These metrics represent different types of unequal work distributions; for example, the ratio of 1st/2nd largest relative contribution by individual members in terms of LOC may allow us to detect possible over-contributors, whereas the minimum relative contribution may enable us to identify under-contributors. In addition, the Gini co-efficient allows us to compute the degree of the work distribution among the team members. The selected metrics allowed us to consider different unequal contribution styles. However, if we wanted to consider a specific case, we would select the most relevant metric; for example, if we wanted to check whether one person is mainly doing all the work, we would consider the ratio of 1st/2nd largest relative contribution by individual members.

We found that inequality of coding contribution did not have a clear negative impact on the collaborative performance of our student teams. The findings (i) confirm that the prevalence of free-riding and one-person teams is highly variable and that the contribution style of individual student software developers is situational e.g. depending on the assignment, circumstances and team, (ii) highlight student software teams' tolerance of and/or ability to compensate for low contributions from some team members and (iii) demonstrate that team inequality does not have an impact in subsequent student team projects.

We started our analysis by looking at the prevalence and severity of inequality of contribution in student software engineering teams on two course units taught at our university. Equal teams where everyone contributed their fair share were not detected in any assignment. We found that many individuals (23%–41%) contributed less than 50% of their fair share (LOC contributed/team size) depending on assignment and cohort (N(teams) = 34–40). The prevalence of both free-riding and one-person teams varied by assignment and cohort. These findings are consistent with those of Buffardi (2020) who found that 50% of students contributed $<$ 50% of their fair share of line changes to a student software engineering project (N (teams) = 10) and with Pieterse et al. (2012) who found that the prevalence of free-riders and diligent isolates varied by project phase and team allocation mechanism (N (teams) $\approx$ 10). Hamer et al. (2021b) also found considerable inequality in the contributions of the students.

We showed that contribution style did not strongly persist from one module to another. Although some researchers have suggested that student group work could be improved by screening student team members for "lone wolf" and "social loafing" tendencies respectively (Barr et al., 2005; Synnott, 2016), we found that a high proportion of students (49%–57%, depending on year) changed their contribution style from SE1 to SE2 (see Fig. 2). This supports the findings of Pieterse et al. (2012) who observed that the contribution style of individual students "tended to change from round to round" of their "Rocking the Boat" teamwork experience and concluded that the "level of participation is situational" in their student software engineering courses.

Students who witness or perceive social loafing among team members may respond by increasing their own efforts to compensate ("social compensation") (Jassawalla et al., 2009) or decrease their efforts to avoid being taken advantage of ("sucker effect") (Mulvey and Klein, 1998). Additionally, a poor team work experience can have a negative impact on future attitudes to team work (Pfaff and Huddleston, 2003; Lewis et al., 1998). We examined whether students' experiences of inequality in SE1 had a material effect on their contribution in SE2. We found that they were unrelated; one possibility is that given the high visibility of team members' real-time contributions in software engineering team projects through platforms like GitLab, recent team experiences are less relevant and have less influence on a student's behaviour in a current team project.

When we looked at the relationship between inequality of contribution and team performance, in general we found that neither team grade nor team source code quality were associated with inequality of coding contributions within student teams. Teams affected by free-riding and over-contribution did not achieve lower grades or produce lower quality source code. We saw a number of weak negative associations between inequality metrics and grade, but these were not consistent across assignments or cohorts. While limited studies on lone-wolf behaviour have demonstrated a negative relationship with team performance (Barr et al., 2005), results from studies on social loafing have produced inconsistent results, indicating that it may have a positive or negative effect on team performance. Individuals in a team may compensate for the low effort of social loafers, particularly when the task is perceived to be of high importance.

One interpretation of our finding is that in most cases, students compensated for the smaller contributions of some team members and were able to do so without a negative impact on quality due to the manageable size and complexity of our assignments. A related, but different interpretation is that, as suggested by Sankaranarayanan et al. (2019), product-focussed mark schemes (like SE2 and to a lesser extent SE1) foster a "performance orientation" which incentivises students to assign more work to more capable students within a team in order to achieve a good grade. The underlying drivers of unequal distribution of work in student software engineering teams warrants further research.

Our results match those of Sankaranarayanan et al. (2019) who found no relationship between evenness of contribution and team grade based on percentage code contribution in a cloud computing course, but differ from those of He (2012a) who reported a negative correlation between self-assessed free-riding and team performance in 69 student software engineering teams, those of Pieterse et al. (2012) who commented that teams that were least successful in a student software engineering project round had the highest number of free-riders, and those of Nguyen et al. (2023) who observed that the average effort of all team members (determined based on the number of submissions and the awareness in addressing coding failures) is strongly correlated with the average grades of the team members. These differences may be attributable to differences in course design between our course and these studies or the way that free-riding and/or team performance was measured.

In our final analysis, we found that the Gini coefficient and coding contribution of the team member who contributed the least were the strongest predictors of team-perceived unequal contribution in SE2 (2017). The threshold above which teams were unlikely to report an issue was very low: 14% of fair share by the smallest team contributor. This high tolerance for low contribution is consistent with prior work which found that students are reluctant to report social loafers to the course team apart from serious cases (Jassawalla et al., 2009, cited in Boren and Morales, 2018). The 1st/2nd contribution ratio was not associated with team-perceived unequal contribution in our study. This suggests that over-contributors either did not feel that their contribution was unjust or felt unable to escalate the issue to the course team.

An important issue we did not address in this study is impact of inequality of contribution on the cooperative learning of the teams and their acquisition of core software engineering concepts. Drawing on the learning theory of Piaget, Soundarajan et al. (2015) emphasises that software engineering team projects are valuable cooperative learning experiences (as well as collaborative ones) that help students to learn key software engineering concepts by bringing their ideas and understanding into conflict with those of other team-members. Future work might look at the relationship between inequality of contribution and student's acquisition of conceptual knowledge during the course.

In this study, we focused on Git-related data of the students, but in the future, it would be interesting to understand more about the students' perceptions, feedback and thoughts on the impact of unequal contributions in team projects by conducting qualitative studies.

5.1. Recommendations

Based on our findings, we make the following recommendations:

- Recommendation 1: Direct interventions should target behaviour in a current project, and should not be determined by behaviour in previous projects,as individual contribution may be different in one project to a subsequent one.
  We found that individual contribution style was weakly correlated between modules and that inequality in one module did not strongly influence contribution in another. This supports the view that, for practical purposes, contribution to software engineering projects should be regarded as situational. As a result, interventions might focus on current projects and need not be informed by students' prior team experiences or behaviours.
- Recommendation 2: Consider incentivising a more equal distribution of work through project design.
  Students were able to successfully complete their projects and achieve high grades despite an unequal distribution of coding effort among developers. higher inequality measures were correlated with higher standard deviation in the project grade assigned to individual students.
  As suggested by Chen et al. (2014), courses should be designed to discourage this. Course leaders could consider modifying the size or complexity of the software development task to incentivise a more equal distribution of contributions or consider a less product focused assessment scheme.
- Recommendation 3: Consider providing alternatives to "reporting" team members.
  We found that teams will accommodate high levels of inequality without raising issues. This may be because teams are reluctant to "report" under-contributing team members. We suggest that course leaders support students by providing guidance on interpersonal strategies for handling both under-contribution and over-contribution behaviour. An intervention of this type for student software engineering teams has been described by Fronza and Wang (2017), but not evaluated.
- Recommendation 4: Leverage the situational nature of under and over-contribution to encourage teams to take responsibility for and address these behaviours.
  Our results suggest that anyone may be an under-contributor or an over-contributor depending on circumstances. We speculate that being made aware of this may help students to take responsibility for and address these behaviours, rather than assuming that they are fixed characteristics or entrenched behaviours of their team-mates.

5.2. Threats to validity

Threats to external validity relate to the generalisability of our results. In this study, we analysed data from two software engineering modules at the University of Manchester. SE1 included 3 × 3-week team-based brownfield development tasks of increasing complexity, while SE2 comprised an extended greenfield development project. Although we analysed data from a variety of development tasks, our results cannot be representative of all software engineering course designs which often differ with respect to many design elements such as grading, involvement of external clients, and prescribed software development practices e.g. Agile/Kanban. In our analysis of team-perceived unequal contribution we were only able to include data from one project, SE2 2017. Replication of our analyses in other courses with different design elements is required to confirm our findings and assess their wider applicability.

Threats to the internal validity of the relationships identified in this study are confounding by unmeasured variables. For example, it is likely that the lack of relationship between inequality and grade/source code quality might be confounded by differences in other characteristics such as skills, experience or personality traits such as conscientiousness, which might also affect team performance.

Threats to construct validity relate to how well the metrics used in our study reflect the underlying constructs of interest. For example, we assume that the distribution of the number of lines of code added between team members reflects the distribution of coding contributions between them. This distribution might also be influenced to some extent by inter-individual differences in coding style (e.g. someone who repeatedly re-works existing code without committing their re-work to Git vs. someone who does not), might fail to capture hidden contributions during pair programming or oral code reviews, and might ignore the differences in technical complexity of certain coding tasks; for example, students writing additional test cases may generate many new lines of code by copying and pasting a basic template and adjusting values which require much less effort than writing the first test case. When used as a metric of free-riding and one-person teams, this measure does not take account of non-coding contributions such as meeting attendance, participation in team communications, design, documentation, etc. These contributions may also be considered by keeping track of the attendance of the meetings, the people who actively participate in team communications, the people who work on design and documentation, and how long they work on these tasks. Chen et al. (2022) point out that unscrupulous students can manipulate Git-based metrics to show that they contribute to the teamwork, and present a tool that helps to identify free-riders with the help of code quality factors, as the contribution of the free-riders to addressing code quality issues is negligible. Even though we considered the Git-related data to identify unequal work distribution, code-quality metrics can also be taken into consideration for this purpose.

We restricted our analysis for RQ2 (the effect of over-contribution and under-contribution in one project compared with a subsequent project) to EX3 in SE1, as the students needed to organise their own work, and this could affect their behaviour in SE2. We also restricted our analysis for RQ5 (the relationship between the repository metrics and team-perceived contribution inequality) to SE2 (2017) due a low number of instances in or data unavailability in SE1, and the change in the grade-reduction process introduced to SE2 (2018).

Other limitations include the use of approximate grades and the fact that source code quality metrics were based on retrospective analysis of source code repositories, meaning that automatically detected rule violations that were false-positives could not be flagged by students during the project and discounted in our analysis.

Table A.1
Characteristics of each student cohort in SE1.

Assignment	Characteristic	2017	2018	2019
All	Participation (Students)	231/257 (90%)	225/244 (92%)	253/258 (98%)
	Participation (Teams)	34/38 (89%)	37/40 (92%)	37/38 (97%)
	7 Member Team	27/34 (79%)	5/37 (14%)	31/37 (84%)
	6 Member Team	7/34 (21%)	30/37 (81%)	6/37 (16%)
	5 Member Team	0/34 (0%)	2/37 (5.4%)	0/37 (0%)
EX1	LOC per student	58 (5, 170)	97 (13, 357)	72 (4, 322)
	Commits per student	3 (1, 8)	3 (1, 9)	3 (1, 8)
	LOC per team	462 (271, 1,793)	780 (414, 1,565)	708 (306, 1,211)
	Commits per team	23 (13, 32)	21 (11, 33)	22 (14, 34)
	Grade (%)	81 (71, 91)	81 (73, 89)	91 (76, 99)
	High Pass (Yes/No)	23/34 (68%)	26/37 (70%)	32/37 (86%)
	Team Perceived Unequal Contribution	1/34 (2.9%)	0/37 (0%)	3/37 (8.1%)
	Source Code Metrics (Students)	231/257 (90%)	225/244 (92%)	246/258 (95%)
	Source Code Metrics (Teams)	34/38 (89%)	37/40 (92%)	36/38 (95%)
EX2	LOC per student	266 (35, 1,184)	167 (1, 657)	224 (0, 669)
	Commits per student	5 (1, 14)	4 (1, 12)	4 (0, 11)
	LOC per team	2,526 (1,396, 3,975)	1,218 (713, 2,358)	1,930 (755, 3,224)
	Commits per team	36 (23, 59)	26 (16, 45)	31 (14, 53)
	Grade (%)	81 (72, 89)	81 (73, 89)	84 (68, 94)
	High Pass (Yes/No)	23/34 (68%)	24/37 (65%)	27/37 (73%)
	Team Perceived Unequal Contribution	2/34 (5.9%)	2/37 (5.4%)	6/37 (16%)
	Source Code Metrics (Students)	231/257 (90%)	225/244 (92%)	253/258 (98%)
	Source Code Metrics (Teams)	34/38 (89%)	37/40 (92%)	37/38 (97%)
EX3	LOC per student	308 (15, 1,908)	247 (0, 1,225)	336 (30, 1,275)
	Commits per student	6 (1, 19)	4 (0, 14)	5 (1, 14)
	LOC per team	2,862 (1,317, 6,562)	2,409 (1,538, 4,506)	2,559 (1,517, 7,373)
	Commits per team	44 (25, 87)	30 (13, 54)	39 (20, 67)
	Grade (%)	80 (34, 89)	76 (51, 92)	84 (53, 94)
	High Pass (Yes/No)	20/34 (59%)	15/37 (41%)	24/37 (65%)
	Team Perceived Unequal Contribution	0/34 (0%)	5/37 (14%)	6/37 (16%)
	Source Code Metrics (Students)	231/257 (90%)	181/244 (74%)	253/258 (98%)
	Source Code Metrics (Teams)	34/38 (89%)	30/40 (75%)	37/38 (97%)

Table A.2
Characteristics of each student cohort in SE2.

Assignment	Characteristic	2017	2018	2019
All	Participation (Students)	227/250 (91%)	219/231 (95%)	245/251 (98%)
	Participation (Teams)	37/41 (90%)	36/38 (95%)	40/41 (98%)
	7 Member Team	8/37 (22%)	3/36 (8.3%)	6/40 (15%)
	6 Member Team	26/37 (70%)	33/36 (92%)	33/40 (82%)
	5 Member Team	3/37 (8.1%)	0/36 (0%)	1/40 (2.5%)
P1	LOC per student	647 (58, 2,098)	660 (140, 2,160)	670 (121, 2,292)
	Commits per student	17 (3, 46)	22 (6, 48)	18 (4, 41)
	LOC per team	4,967 (2,090, 8,080)	4,826 (2,684, 7,956)	5,026 (2,768, 8,052)
	Commits per team	121 (53, 211)	132 (69, 210)	120 (76, 179)
	Grade (%)	79 (25, 94)	84 (58, 94)	82 (53, 96)
	High Pass (Yes/No)	17/37 (46%)	24/36 (67%)	23/40 (57%)
	Team Perceived Unequal Contribution	11/37 (30%)	11/36 (31%)	17/40 (42%)
	Source Code Metrics (Students)	209/250 (84%)	195/231 (84%)	233/251 (93%)
	Source Code Metrics (Teams)	34/41 (83%)	32/38 (84%)	38/41 (93%)

CRediT authorship contribution statement

Kamilla Kopec-Harding: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Visualization, Writing – original draft. **Sukru Eraslan:** Conceptualization, Methodology, Visualization, Writing – original draft. **Bowen Cai:** Software. **Suzanne M. Embury:** Writing – review & editing. **Caroline Jay:** Conceptualization, Methodology, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: This work was supported by the Institute of Coding which received £20 m of funding from the Office for Students (OfS), as well as support from the Higher Education Funding Council for Wales (HEFCW). No competing interests exist.

Data availability

The data that has been used is confidential.

Acknowledgements

The work reported in this paper was supported by the Institute of Coding, which received funding from the Office for Students (OfS) and the University of Manchester, United Kingdom, and support from the Higher Education Funding Council for England (HEFCE), United Kingdom.

Appendix. Characteristics of each student cohort

Basic characteristics of each cohort were calculated. For continuous variables, the median, 5th and 95th percentiles are reported. For categorical variables, percentages are reported. For Participation (Students) and Participation (Teams), denominators correspond to the number of eligible students and teams to indicate participation rate. Source Code Metrics (Students) and Source Code Metrics (Teams) refer to the proportion of students and teams for whom source code metrics were available (i.e. the ones with a successful project build (final commit to main or up to 2 commits earlier)). Tables A.1 and A.2 show the characteristics of each student cohort in SE1 and SE2 respectively.

References

Adams, S.G., 2003. Building successful student teams in the engineering classroom. J. STEM Educ. Innov. Res. 4 (3).

Aggarwal, P., O'Brien, C.L., 2008. Social loafing on group projects. J. Mark. Educ. 30 (3), 255–264. http://dx.doi.org/10.1177/0273475308322283.

Andrés, A., Salafranca, L., Solanas, A., 2011. Predicting team output using indices at group level. Span. J. Psychol. 14 (2), 773–788. http://dx.doi.org/10.5209/rev_sjop.2011.v14.n2.25.

Arianne Project, 2017. Stendhal. URL: https://github.com/arianne/stendhal.

Arthur, Jr., W., Bell, S.T., Edwards, B.D., 2007. A longitudinal examination of the comparative criterion-related validity of additive and referent-shift consensus operationalizations of team efficacy. Organ. Res. Methods 10 (1), 35–58. http://dx.doi.org/10.1177/1094428106287574.

Bai, X., Li, M., Pei, D., Li, S., Ye, D., 2018. Continuous delivery of personalized assessment and feedback in agile software engineering projects. In: 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training. ICSE-SEET, pp. 58–67.

Barr, T.F., Dixon, A.L., Gassenheimer, J.B., 2005. Exploring the "lone wolf" phenomenon in student teams. J. Mark. Educ. 27 (1), 81–90. http://dx.doi.org/10.1177/0273475304273459.

Bastarrica, M.C., Perovich, D., Samary, M.M., 2017. What can students get from a software engineering capstone course? In: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track. ICSE-SEET, pp. 137–145. http://dx.doi.org/10.1109/ICSE-SEET.2017.15.

Bender, R., Lange, S., 2001. Adjusting for multiple testing—when and how? J. Clin. Epidemiol. 54 (4), 343–349.

Biemann, T., Kearney, E., 2010. Size does matter: How varying group sizes in a sample affect the most common measures of group diversity. Organ. Res. Methods 13 (3), 582–599. http://dx.doi.org/10.1177/1094428109338875.

Blickem, C., Bower, P., Protheroe, J., Kennedy, A., Vassilev, I., Sanders, C., Kirk, S., Chew-Graham, C., Rogers, A., 2011. The role of information in supporting self-care in vascular conditions: a conceptual and empirical review. Health Soc. Care Community 19 (5), 449–459.

Boren, A.E., Morales, S., 2018. Celebrities and slackers: A grounded theory of the dynamics of social loafing on student teams. J. Leadersh. Educ. 17 (2), 42–59.

Borg, M., 2020. Making lab sessions mandatory — on student work distribution in a gamified project course on market-driven software engineering. In: 2020 IEEE 32nd Conference on Software Engineering Education and Training. CSEE T, pp. 1–10. http://dx.doi.org/10.1109/CSEET49119.2020.9206218.

Buffardi, K., 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. SIGCSE '20, ACM, New York, NY, USA, pp. 650–656. http://dx.doi.org/10.1145/3328778.3366948.

Cao, X., Bai, G., Cao, C., Zhou, Y., Xiong, X., Huang, J., Luo, L., 2020. Comparing regional distribution equity among doctors in China before and after the 2009 medical reform policy: A data analysis from 2002 to 2017. Int. J. Environ. Res. Public Health 17 (5), 1520. http://dx.doi.org/10.3390/ijerph17051520.

Chen, C.Y., Hong, Y.C., Chen, P.C., 2014. Effects of the meetings-flow approach on quality teamwork in the training of software capstone projects. IEEE Trans. Educ. 57 (3), 201–208. http://dx.doi.org/10.1109/TE.2014.2305918.

Chen, H.-M., Nguyen, B.-A., Dow, C.-R., 2022. Code-quality evaluation scheme for assessment of student contributions to programming projects. J. Syst. Softw. 188, 111273. http://dx.doi.org/10.1016/j.jss.2022.111273.

Chounta, I.A., Manske, S., Hoppe, H.U., 2017. "From making to learning": introducing dev camps as an educational paradigm for re-inventing problem-based learning. Int. J. Educ. Technol. High. Educ. 14, 1–15, 21. http://dx.doi.org/10.1186/s41239-017-0061-2.

Comer, D.R., 1995. A model of social loafing in real work groups. Hum. Relat. 48 (6), 647–667.

Cowell, F.A., 2011. Measuring Inequality, third ed. In: LSE Perspectives in Economic Analysis, Oxford University Press, Oxford.

Embury, S.M., Page, C., 2019. Effect of Continuous Integration on Build Health in Undergraduate Team Projects. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11350 LNCS, Springer Verlag, pp. 169–183. http://dx.doi.org/10.1007/978-3-030-06019-0_13.

Eraslan, S., Kopec-Harding, K., Jay, C., Embury, S.M., Haines, R., Cortés Ríos, J.C., Crowther, P., 2020. Integrating GitLab metrics into coursework consultation sessions in a software engineering course. J. Syst. Softw. 167, 110613. http://dx.doi.org/10.1016/j.jss.2020.110613.

Erdenee, O., Paramita, S.A., Yamazaki, C., Koyama, H., 2017. Distribution of health care resources in Mongolia using the Gini coefficient. Hum. Resour. Health 15, 56. http://dx.doi.org/10.1186/s12960-017-0232-1.

Feliciano, J., Storey, M., Zagalsky, A., 2016. Student experiences using GitHub in software engineering courses: A case study. In: 2016 IEEE/ACM 38th International Conference on Software Engineering Companion. ICSE-C, pp. 422–431.

Fronza, I., Wang, X., 2017. Towards an approach to prevent social loafing in software development teams. In: 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM, pp. 241–246. http://dx.doi.org/10.1109/ESEM.2017.37.

Hamer, S., Quesada-López, C., Martínez, A., Jenkins, M., 2020. Measuring students' contributions in software development projects using git metrics. In: 2020 XLVI Latin American Computing Conference. CLEI, IEEE, pp. 531–540.

Hamer, S., Quesada-López, C., Martínez, A., Jenkins, M., 2021a. Measuring students' source code quality in software development projects through commit-impact analysis. In: International Conference on Information Technology & Systems. Springer, pp. 100–109.

Hamer, S., Quesada-López, C., Martínez, A., Jenkins, M., 2021b. Using git metrics to measure students' and teams' code contributions in software development projects. CLEI Electron. J. 24 (2), 8.

Harrison, D.A., Klein, K.J., 2007. What's the difference? Diversity constructs as separation, variety, or disparity in organizations. Acad. Manag. Rev. 32 (4), 1199–1228.

He, J., 2012a. Understanding the Effects of Freeriding in Team Dynamics. Technical Report 1, URL: http://aisel.aisnet.org/amcis2012/proceedings/SocialIssues/12.

He, J., 2012b. Understanding the effects of freeriding in team dynamics. In: Proceedings of the Eighteenth Americas Conference on Information Systems. AMCIS, Seattle, Washington.

Iacob, C., Faily, S., 2019. Exploring the gap between the student expectations and the reality of teamwork in undergraduate software engineering group projects. J. Syst. Softw. 157, 110393. http://dx.doi.org/10.1016/j.jss.2019.110393.

Jaspan, C., Sadowski, C., 2019. No single metric captures productivity. In: Rethinking Productivity in Software Engineering. Springer, pp. 13–20.

Jassawalla, A., Sashittal, H., Sashittal, A., 2009. Students' perceptions of social loafing: Its antecedents and consequences in undergraduate business classroom teams. Acad. Manag. Learn. Educ. 8 (1), 42–54.

Kapoor, A., Gardner-McCune, C., 2018. Considerations for switching: Exploring factors behind CS students' desire to leave a CS major. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. In: ITiCSE 2018, ACM, New York, NY, USA, pp. 290–295. http://dx.doi.org/10.1145/3197091.3197113.

Karau, S.J., Williams, K.D., 1995. Social loafing: Research findings, implications, and future directions. Curr. Dir. Psychol. Sci. 4 (5), 134–140.

Kay, J., Yacef, K., Reimann, P., 2007. Visualisations for team learning: Small teams working on long-term projects. In: Proceedings of the 8th International Conference on Computer Supported Collaborative Learning. CSCL '07, International Society of the Learning Sciences, pp. 354–356.

Khakurel, J., Porras, J., 2020. The effect of real-world capstone project in an acquisition of soft skills among software engineering students. In: 2020 IEEE 32nd Conference on Software Engineering Education and Training. CSEE and T 2020, Institute of Electrical and Electronics Engineers Inc., pp. 36–44. http://dx.doi.org/10.1109/CSEET49119.2020.9206201.

LaBeouf, J.P., Griffith, J.C., Roberts, D.L., 2016. Faculty and student issues with group work: What is problematic with college group assignments and why? J. Educ. Hum. Dev. 5 (1), 13.

Lautenbach, F., Korte, J., Möhwald, A., Heyder, A., Grimminger-Seidensticker, E., 2020. A 14-week intervention study on changing preservice teachers' psychological perspectives on inclusion: Explicit and implicit attitudes, self-efficacy, and stress perception toward inclusion. Front. Educ. 5, 7.

Leung, W.S., 2017. Bad blood: Managing toxic relationships through belbin roles for first year software engineering students. In: ACM International Conference Proceeding Series. ACM, New York, New York, USA, pp. 82–86. http://dx.doi.org/10.1145/3162957.3163010.

Lewis, P., Aldridge, D., Swamidass, P.M., 1998. Assessing teaming skills acquisition on undergraduate project teams. J. Eng. Educ. 87 (2), 149–155.

Liu, Y., Stroulia, E., Wong, K., Germán, D.M., 2004. Using CVS historical information to understand how students develop software. In: MSR. IET, pp. 32–36.

Lu, L., 2020. Information-based interventions for household water efficiency in England and Wales: evidence, barriers and learning opportunities. Int. J. Water Resour. Dev. 36 (6), 926–939. http://dx.doi.org/10.1080/07900627.2019.1684244.

Mahnic, V., 2012. A capstone course on agile software development using scrum. IEEE Trans. Educ. 55 (1), 99–106. http://dx.doi.org/10.1109/TE.2011.2142311.

Marques, M.R., 2016. Monitoring: An intervention to improve team results in software engineering education. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education. SIGCSE '16, ACM, New York, NY, USA, p. 724. http://dx.doi.org/10.1145/2839509.2851054.

Marshall, L., Pieterse, V., Thompson, L., Venter, D.M., Pieterse, V., 2016. Exploration of participation in student software engineering teams. ACM Trans. Comput. Educ. 16 (2), 5. http://dx.doi.org/10.1145/2791396.

Meskarpour-Amiri, M., Mehdizadeh, P., Barouni, M., Dopeykar, N., Ramezanian, M., 2014. Assessment the trend of inequality in the distribution of intensive care beds in Iran: using GINI index. Glob. J. Health Sci. 6 (6), 28–36. http://dx.doi.org/10.5539/gjhs.v6n6p28.

Mittal, M., Sureka, A., 2014. Process mining software repositories from student projects in an undergraduate software engineering course. In: Companion Proceedings of the 36th International Conference on Software Engineering. In: ICSE Companion 2014, ACM, New York, NY, USA, pp. 344–353. http://dx.doi.org/10.1145/2591062.2591152.

Mulvey, P.W., Klein, H.J., 1998. The impact of perceived loafing and collective efficacy on group goal processes and group performance. Organ. Behav. Hum. Decis. Process. 74 (1), 62–87. http://dx.doi.org/10.1006/obhd.1998.2753.

Nguyen, B.-A., Chen, H.-M., Dow, C.-R., 2023. Identifying nonconformities in contributions to programming projects: from an engagement perspective in improving code quality. Behav. Inf. Technol. 42 (1), 141–157. http://dx.doi.org/10.1080/0144929X.2021.2017483.

Nguyen, B.-A., Ho, K.-Y., Chen, H.-M., 2020. Measure students' contribution in web programming projects by exploring source code repository. In: 2020 International Computer Symposium. ICS, pp. 473–478. http://dx.doi.org/10.1109/ICS51289.2020.00099.

Ntirandekura, M.C., Eude, T., 2018. Git as a support to assess students' contribution in teamwork. In: Yang, J., Chang, M., Wong, L., Rodrigo, M. (Eds.), Proceedings of the 26th International Conference on Computers in Education. Philippines: Asia-Pacific Society for Computers in Education. pp. 340–342.

Pfaff, E., Huddleston, P., 2003. Does it matter if I hate teamwork? What impacts student attitudes toward teamwork. J. Mark. Educ. 25 (1), 37–45. http://dx.doi.org/10.1177/0273475302250571.

Pieterse, V., Thompson, L., 2010. Academic alignment to reduce the presence of 'social loafers' and 'diligent isolates' in student teams. Teach. High. Educ. 15 (4), 355–367. http://dx.doi.org/10.1080/13562517.2010.493346.

Pieterse, V., Thompson, L., Marshall, L., 2011. Rocking the boat: an approach to facilitate formation of effective student teams. In: Proceedings of the Southern African Computer Lecturers' Association. SACLA'11, Citeseer, pp. 115–123.

Pieterse, V., Thompson, L., Marshall, L., Venter, D.M., 2012. An intensive software engineering learning experience. In: Proceedings of Second Computer Science Education Research Conference. CSERC '12, ACM, New York, NY, USA, pp. 47–54. http://dx.doi.org/10.1145/2421277.2421283.

R Core Team, 2020. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL: https://www.R-project.org/.

Raibulet, C., Arcelli Fontana, F., 2018. Collaborative and teamwork software development in an undergraduate software engineering course. J. Syst. Softw. 144, 409–422. http://dx.doi.org/10.1016/j.jss.2018.07.010.

Sankaranarayanan, S., Wang, X., Dashti, C., An, M., Ngoh, C., Hilton, M., Sakr, M., Rosé, C., 2019. An Intelligent-Agent Facilitated Scaffold for Fostering Reflection in a Team-Based Project Course. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11626 LNAI, Springer Verlag, pp. 252–256. http://dx.doi.org/10.1007/978-3-030-23207-8_47, URL: http://teel.cs.cmu.edu/.

Schippers, M.C., 2014. Social loafing tendencies and team performance: The compensating effect of agreeableness and conscientiousness. Acad. Manag. Learn. Educ. 13 (1), 62–81.

Simms, A., Nichols, T., 2014. Social loafing: a review of the literature. J. Manag. Policy Pract. 15 (1), 58.

Sorenson, J.B., 2002. The use and misuse of the coefficient of variation in organizational demography research. Sociol. Methods Res. 30 (4), 475–491. http://dx.doi.org/10.1177/0049124102030004001.

Soundarajan, N., Joshi, S., Ramnath, R., 2015. Collaborative and cooperative-learning in software engineering courses. In: Proceedings - International Conference on Software Engineering, Vol. 2. IEEE Computer Society, pp. 319–322. http://dx.doi.org/10.1109/ICSE.2015.164.

Spadini, D., Aniche, M., Bacchelli, A., 2018. Pydriller: Python framework for mining software repositories. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2018, ACM Press, New York, New York, USA, pp. 908–911. http://dx.doi.org/10.1145/3236024.3264598.

Stein, M.V., 2003. Student effort in semester-long and condensed capstone project courses. J. Comput. Sci. Coll. 18 (4), 200–212.

Synnott, C.K., 2016. Guides to reducing social loafing in group projects: Faculty development. J. High. Educ. Manag. 31 (1), 211–221.

Tucker, R., Abbasi, N., 2015. The architecture of teamwork: examining relationships between teaching, assessment, student learning and satisfaction with creative design outcomes. Archit. Eng. Des. Manag. 11 (6), 405–422. http://dx.doi.org/10.1080/17452007.2014.927750.

van der Duim, L., Andersson, J., Sinnema, M., 2007. Good practices for educational software engineering projects. In: Proceedings of the 29th International Conference on Software Engineering. ICSE '07, IEEE Computer Society, USA, pp. 698–707. http://dx.doi.org/10.1109/ICSE.2007.40.

Vanhanen, J., Lehtinen, T.O.A., Lassenius, C., 2012. Teaching real-world software engineering through a capstone project course with industrial customers. In: 2012 First International Workshop on Software Engineering Education Based on Real-World Experiences. EduRex, pp. 29–32. http://dx.doi.org/10.1109/EduRex.2012.6225702.

Wei, X., Allen, N.J., Liu, Y., 2016. Disparity in organizational research: How should we measure it? Behav. Res. Methods 48 (1), 72–90. http://dx.doi.org/10.3758/s13428-015-0585-0.

Williams, D.L., Beard, J.D., Rymer, J., 1991. Team projects: Achieving their full potential. J. Mark. Educ. 13, 45–53.

Yasar, S., Baker, D., Krause, S., Roberts, C., 2007. In her shoes: How team interactions affect engineering self-efficacy. In: ASEE Annual Conference and Exposition, Conference Proceedings. pp. 12.866.1–12.866.15.

Kamilla Kopec-Harding holds a Ph.D. in Computational Chemistry and is a Research Software Engineer/Data Scientist at the University of Manchester. She has provided data management, statistical programming, and analytics support to projects in a range of disciplines, including epidemiology and computer science.

Sukru Eraslan is a lecturer in the Computer Engineering Program at Middle East Technical University Northern Cyprus Campus (METU NCC). He completed his Ph.D. in Computer Science at the University of Manchester. He then worked as a research associate at METU NCC and the University of Manchester. His primary research interests are centred around human–computer interaction. He also undertakes research in collaborative software development.

Bowen Cai completed his B.Sc. in Computer Science at the University of Manchester and his M.Sc. in Advanced Computer Science at Imperial College London. He is currently working as a Software Engineer at a Fintech startup.

Suzanne M. Embury is a Reader in Software Engineering at the University of Manchester and the Director of Imago, the University of Manchester's Student Software Company. Her research is in the area of software testing and software discovery. She is a co-founder of the Git Workflows Warehouse, a repository of tools and information for the collaborative use of Git. She has a Ph.D. from the University of Aberdeen.

Caroline Jay is the Research Director of the Software Sustainability Institute, a Professor of Computer Science at the University of Manchester and a Fellow of the Alan Turing Institute. She is a Chartered Psychologist and Computer Scientist and undertakes research exploring the relationship between human behaviour, data science, and software engineering.