Using *git* metrics to measure students' and teams' code contributions in software development projects

Sivana Hamer, Christian Quesada-López, Alexandra Martínez, Marcelo Jenkins

Universidad de Costa Rica, Escuela de Ciencias de la Computación e Informática,
San Jose, Costa Rica,

{*sivana.hamer, cristian.quesadalopez, alexandra.martinez, marcelo.jenkins*}@ucr.ac.cr

Abstract

Many software engineering courses are centered around team-based project development. Analyzing the source code contributions during the projects' development could provide both instructors and students with constant feedback to identify common trends and behaviors that can be improved during the courses. Evaluating course projects is a challenge due to the difficulty of measuring individual student contributions versus team contributions during the development. The adoption of distributed version control systems like *git* enables the measurement of students' and teams' contributions to the project. In this work, we analyze the contributions within eight software development projects, with 150 students in total, from undergraduate courses that used project-based learning. We generate visualizations of aggregated *git* metrics using inequality measures and the contribution per module, offering insights into the practices and processes followed by students and teams throughout the project development. This approach allowed us to identify inequality among students' contributions, the modules where students contributed, development processes with a non-steady pace, and integration practices rendering a useful feedback tool for instructors and students during the project's development. Further studies can be conducted to assess the quality, complexity, and ownership of the contributions by analyzing software artifacts.

Keywords: git contribution metrics, mining software repositories, software engineering education, continuous assessment, project-based learning

1 Introduction

Software engineering is a technical activity that also requires soft skills such as teamwork, initiative, commitment, communication, and time management [1]. Although these skills are essential for software engineering professionals, there is a gap between what is taught in academia and what is needed in the industry [2]. Hence a paradigm shift has emerged in education to help students acquire the knowledge and skills needed in the industry through project-based learning [3]. Nonetheless, evaluating such projects is a challenge for instructors due to the difficulty of objectively measuring individual students' contributions against team contributions because the projects are a product of collaborative effort [4–6]. To evaluate students fairly, instructors require metrics to assess the contributions of students.

Given the relevance of teamwork as a key skill in the industry, the adoption of collaboration tools has become essential [7]. One such tool is *git*, a distributed version control system that facilitates collaborative code writing [8]. *Git* is open-source and widely used in industry as well as educational settings [9].

The adoption of *git* in education [10–14] offers the opportunity to gain insights into the amount of work contributed to the project by each team member (e.g., source code added to the repository), as well as behavior patterns (e.g., software engineering practices and the team dynamics), through metrics visualizations. We believe that an assessment based on quantitative *git* metrics could serve as a helpful feedback tool (and fairer grading scheme) for both instructors and students, providing objective real-time measures of contributions to a development project. Continuous assessment of students' and teams' performance in development projects could ensure active participation from the students and teams during the project course [9, 15].

In previous studies, researchers have assessed the distribution of contributions in open-source software and private projects, focusing on analyzing the Pareto principle [16–18]. Studies in academic contexts have

focused on extracting metrics such as commits, merges, lines of code, number of modified files, types of modified files, time spent, and component and developer entropy [9, 12–14].

Our goal is to propose an approach for measuring and visualizing students' contributions in software development projects. The metrics extracted from *git* are commits, merges, and churns, whereas the aggregate measures are inequality indexes and inter-decile ratios. We also extract the daily commits and merges to analyze the contribution frequency. This approach was used to analyze eight undergraduate software development projects with 150 students and 31 collaborative teams. In this work, we extend our previous research [19] enhancing the detail of our approach's analysis, including in the evaluation of three new projects, inspecting the contributions by module, and examining integration practices. We defined the following research questions:

RQ1. How does contribution vary across different software development projects and students?

This question allows us to determine how the contributions are distributed for students and their teams, indicating the degree of inequality of these contributions.

RQ2. How does contribution within software development projects evolve?

This question permits us to identify trends in the evolution of the contribution concerning the frequency over time for students and teams.

The remainder of the paper is organized as follows: Section 2 summarizes relevant related work, Section 3 describes the methodology, Section 4 presents the results, Section 5 details the implications of our results. Section 6 outlines our conclusions and future work.

2 Related work

Here we present previous works that investigate *git* repositories for contributions in educational contexts with team-based projects.

Cochez et al. [12] analyzed *git* usage in several educational projects examining the commit log data, with teams ranging from two to five students. They tracked the commit activity in each class session, and during the entire course the quality of commit messages, the difference between student's participation and files committed. They found that most students have regular commits in sessions with more infrequent commits outside of sessions, a high amount of gibberish in the commit messages, a considerable amount of inequality in the teams, and the amount of garbage is related to students' awareness of good development practices. Their findings suggest that without proper adoption, the use of versioning systems is limited to a submission assignment tool while leaving out the benefits of collaborative and distributed work.

Mittal and Ashish [13] mined wiki-based document management systems, version control systems, and issue tracking systems to gain insights into the practices and procedures, in an educational setting, with five-member teams. From the Wiki, the authors extracted the quality of the messages, consistency of activity, and uniformity of contributions within the team. Using the version control systems, they analyzed the component and developer entropy and the commits' behavior near deadlines and from the issue tracking systems, they examined trends for bug creation and fixing, mean time to repair, defective components, and process compliance. They found varying degrees of equality among team members, consistency in the activity, quality, and process compliance.

Kertész [14] adopted Github as a teaching tool and collaborative platform for course assignments to enhance critical thinking and collaborative learning. Students used the repositories to check out materials and solve their assignments with examples. The code frequency and quality of assignments were investigated. They found that collaborative learning helped improve the course as it enhanced students' skills and provided an inherent advantage in group working skills.

Parizi et al. [9] proposed an architectural model to evaluate contributions in students' projects in an objective manner. Their model comprises the number of commits, number of merge pull requests, number of files, total lines of code, and time spent per day. Additionally, they proposed to take into account the difficulty of the problem assigned to a developer for a more accurate assessment of effort. They believe that the conducted assessment can help more productive learning preparing graduates better for the industry.

Raibulet and Arcelli [20] described their experience of enhancing collaboration and teamwork activities with different software tools in a university course. The tools used were Github for collaborative software development, Sonarqube for software quality analysis, and Microsoft project for project management. Feedback was recollected from students with questionnaires of the advantages and disadvantages of using such tools and found that students considered the tools useful. They recommend using such tools, as students were enthusiastic about learning new approaches and tools, stimulating the learning process.

Gustavsson and Brohede [15] created a tool that continuously assesses the source code and interactions from Github repositories. They collected the files, commits, students, and issues (merges) and extracted the

number of commits, merges, lines of code, events, and comments, using the metrics to rank students. They found that using the tool was useful to assess the individual contribution and the course outcome.

Tushev et al. [21] investigated how Github as a platform affects students' team dynamics and performance in large software engineering classrooms, including the benefits, challenges, and drawbacks of its use. They conducted entry and exit surveys, as well as an exit interview. They analyzed students' behavior based on the number of commits and quality of the commit messages. They found that the use of Github does not affect the quality of the work, but the metrics analyzed could not be used as a reliable proxy for team performance assessment. Furthermore, the social features of Github had several limitations.

Buffardi [6] evaluated the relationships between qualitative and quantitative assessments. For the qualitative assessment, they analyzed peer-reviewed evaluations, project scores, and quiz scores. Whereas, for the quantitative evaluation, they measured the number of commits, lines of code changes, user stories completed, and user points completed are used for the quantitative assessment. They found a positive correlation between the peer-reviewed evaluation of interactions and contributions. Furthermore, there was a lack of strong association between the qualitative and quantitative assessments. The use of quantitative metrics as a complement to traditional assessment is recommended.

Regarding the inequality measures, they have been previously used and analyzed in open source software projects. Goeminne and Mens [16] have used the inequality indexes (Hoover, Gini, and Theil index) to analyze the distribution of activity in open source projects. They analyzed the distribution of contributions for the number of commits, mails, new bug report submissions, comments added to existing bug reports, and changes to existing bug reports mining the data from version repositories, mailing lists, and bug trackers. They found a highly imbalanced distribution of activity across all contributions. Meanwhile, in the field of econometrics, other inequality metrics have been proposed, such as the inter-decile ratios [22].

The main contribution of our work is a measuring and visualization approach of students' contributions from *git* metrics, using aggregated inequality indexes and inter-decile ratios adopted from the field of econometrics. Our analysis shows contributions of students, contributions of teams, contributions of students within their teams, contributions to modules, and contributions through time.

3 Methodology

This section describes our approach and the process followed to calculate contributions, the metric extraction process, as well as the *git* repositories and course descriptions.

3.1 Measuring Approach and Process

Our approach was designed from the point of view of the instructors. Fig. 1 shows the objective, questions, and metrics used in our analysis [23]. The goal of the instructors was to measure student's contributions. Hence, the following questions were asked: How can we measure contributions? How can we measure contribution inequality? How can we measure contributions over time? The first question helps us identify metrics that represent contribution. The second question allows us to determine whether a project's contribution is equal or unequal, providing information about high and low contributors and team dynamics. The third question enables us to discover patterns of contributions over time. Each of these metrics will be analyzed for the student within the project, the team within the project, and the student within their team.

Figure 1: Objective, questions, and metrics used in our analysis.

The approach used to extract the metrics is shown in Fig. 2. There are three main steps: parsing the students' identification, extracting the *git* metrics, and calculating the metrics from econometrics. First and foremost, the *git* repositories must be selected. In our case, we selected eight undergraduate software development repositories. Then we proceeded with parsing, which begins with grouping student's identifications and removing outliers. Account identifications were grouped up to avoid fragmenting the results

using *mailmaps*¹. Also, students who abandoned the course were excluded. Then, *git* metrics (including the metrics per day) for each student were extracted. The modules that each student contributed to were mined using the file path of the commits. The results per student were grouped up in their corresponding team to obtain the team's contribution and were divided for each team resulting in the data for the students within their teams. Finally, the inequality indexes and inter-decile ratios were calculated for each data set. With the different metrics results, several visualizations, such as heat maps and Pareto charts, were plotted to analyze the contributions throughout the development process.

Figure 2: Measurement approach for *git* metrics.

3.2 Metrics

The following metrics were extracted from *git* repositories:

- **Commits:** The number of commits a student pushed to the repository. To obtain this, the *git log*² command was used with the options *--use-mailmap*, *--author*, *--all* and *--no-merges*. Then, each line indicates a commit that is counted per developer. Commits do not include merges.
- **Merges:** The number of merges a student has integrated into the repository. This is obtained using the *git log* command with the options *--use-mailmap*, *--author*, *--all* and *--merges*.
- **Churn:** The lines of code added and deleted by a student, including comments. To obtain this, the *git log* command was used with the options *--use-mailmap*, *--author*, *--numstat*, *--all*, and *--pretty=tformat*. Then, the added and removed lines of code were added.
- **Daily commits or merges:** The amount of commits or merges a student contributes per day. It was obtained by the *git log* command with the options *--use-mailmap*, *--author*, *--date=iso8601*, *--pretty=format:%cd* and *--all*. We added the option *--merges* for merges and *--no-merges* for commits. The number of times a date appears is counted and used as the metric.

We only considered date intervals according to projects' sprints and the courses' total duration, using the *git* options *--since* and *--until*.

Then, we proceeded to calculate the aggregate inequality metrics. These inequality metrics come from the econometrics field, where they are used to measure income distribution. However, they have been applied in other settings to measure distributions [24]. In our study, we used them to analyze the distribution of student contributions in projects. The following inequality metrics were calculated:

- **Inequality indexes:** We used the Hoover index, Gini index, and Theil index, as defined by Goeminne and Mens [16]. The value of these measures ranges from 0 to 1, with 0 indicating perfect equality and 1 perfect inequality. They were calculated with R³ and the packages REAT⁴ for Hoover index, and ineq⁵ for Gini and Theil indexes.
- **Inter-decile ratios:** We used the ratio of the population richest over the poorest to analyze the inequality in the tails of the population, as defined by Summer et al. [22], indicating how many times the richest are wealthier than the poorest. In our case, *wealth* is measured in terms of contributions.

¹https://git-scm.com/docs/git-check-mailmap#_mapping_authors

²<https://git-scm.com/docs/git-log>

³<https://www.r-project.org/>

⁴<https://www.rdocumentation.org/packages/REAT/versions/3.0.2>

⁵<https://www.rdocumentation.org/packages/ineq/versions/0.2-13>

We used the 20/20 ratio (contributions of the 20% top contributors versus those of the 20% bottom contributors) to give insight into the disparity between the top and bottom contributors. Additionally, we used the Palma ratio (10/40) [22] because inequality changes in income are affected mostly by these populations; hence, we are interested in analyzing this metric in software development. Finally, the 50/50 ratio was chosen to analyze the top half vs. the bottom half contributors. This ratio differs from previous ones as it includes the entire population and not just the tails.

3.3 Courses and *git* repositories

Our analysis includes eight software engineering projects, with a total of 150 students and 31 collaborative teams, developed in undergraduate courses at the Universidad de Costa Rica. The courses (and their projects) lasted one semester each (equivalent to 4 months of software development). The *git* repositories used in this study were from the following courses:

- **Mobile application development (M):** This is a four-credit course taken by fourth-year students, where they learn to develop apps for mobile platforms such as Android and iOS. The main course technologies are Android Studio and Java, and Bitbucket. Projects M1, M2, and M3 correspond to this course. Examples of what students have developed include virtual campus services such as the location of buildings, cafes, libraries, social and academic events, transport, and others.
- **Software engineering (SE):** This is a four-credit course taken by third-year students, where they learn to apply agile methodologies for the development of web-based applications. The main course technologies are Visual Studio, ASP.NET MVC, C#, SQL Server, JavaScript, and Bitbucket. This course is taught as part of the *integrative project in software engineering and databases* block, where there is one merged project for both the software engineering and the database courses. Projects SE1, SE2, SE3, SE4, and SE5 correspond to this course. Examples of what students have developed include reservation systems and teacher assessment systems.

In these courses, all students worked on a single shared project in collaborative teams (*scrums of scrums*⁶). Each team was in charge of a different module or theme with shared features, all of which had to be integrated into the repository. Projects were developed using agile methodologies: mostly Scrum with some added practices from Extreme Programming (XP). The main methodological difference between the two courses was the *sprint 0* in SE, where students focused on the conception and general design rather than functional software (non-existent in M courses). As not all students had the knowledge required to use Git-based repositories, they were taught how to use this technology. Details of student projects are shown in Table 1. In total, there were 31 teams, with most projects having four teams. There were in total 9221 commits, 4490 merges and 10,062,406 churn. For mobile projects, the averages are 412.67 commits, 123.33 merges, and 102,677.67 churn. In the case of software engineering projects, the averages were 1596.6 commits, 824 merges, and 1,950,874.6 churn.

4 Results

In this section, we present the results. Using the proposed approach, we answer each research question, along the following dimensions: student contribution to the project, team contribution to the project, and student contribution to the team.

4.1 RQ1: Contribution variation across projects and students

4.1.1 Student contribution to the project

Tables 2, 3, and 4 show the statistics of the students contributions to the project. Fig. 3 shows the students' contributions to projects, where each subfigure represents a project and contains three histograms (for commits, merges, and churn). In each histogram, the horizontal axis contains 30 intervals of contributions, and the vertical axis displays the number of students whose amount of contributions fall within each horizontal interval. These results show that students contributed unequally to projects.

We will analyze the projects per course as direct comparisons between courses cannot be made due to their technological differences. Results for the projects M1, M2, and M3 show significant differences in the number of contributions for the projects. The difference in the total quantity of contributions between M2, and M1 and M3 was because project M2 had fewer students. Project M1 had half the commits and merges of project M3, while project M3 had half the churn of project M1. This may indicate that project

⁶<https://www.agilealliance.org/glossary/scrums-of-scrums>

Table 1: Information of student projects.

Project	Team	Commits		Merges		Churn	
		All	Team	All	Team	All	Team
M1	1		48		9		27,394
	2		47		8		51,038
	3	378	150	83	14	163,545	57,421
	4		133		52		27,692
M2	1		30		6		11,383
	2	153	42	29	7	73,775	45,985
	3		81		16		16,407
M3	1		184		59		29,246
	2		198		61		13,503
	3	707	154	258	52	70,713	12,731
	4		171		86		15,233
SE1	1		688		286		720,227
	2		371		207		144,578
	3	2,371	396	1,106	202	2,132,111	126,044
	4		437		190		930,438
	5		479		221		210,824
SE2	1		395		238		182,134
	2		204		101		54,482
	3	1,079	167	577	84	516,519	98,674
	4		313		154		181,229
SE3	1		300		161		168,457
	2		292		150		1,011,330
	3	1,431	374	822	201	2,320,148	800,828
	4		465		310		339,533
SE4	1		402		172		171,085
	2		234		92		115,479
	3	1,692	574	772	291	4,120,574	267,093
	4		482		217		3,566,917
SE5	1		229		183		102,633
	2	1,410	352	843	197	665,021	32,420
	3		829		463		529,968

Table 2: Statistics of commits per project

Project	Min	25Q	Median	Mean	75Q	Max	St. Dev.
M1	1.00	8.50	19.00	19.89	22.50	61.00	16.63
M2	1.00	6.00	19.00	17.00	28.00	35.00	13.44
M3	13.00	24.00	37.00	37.21	45.00	90.00	17.62
SE1	32.00	59.00	79.00	94.84	99.00	375.00	70.17
SE2	16.00	32.00	45.00	51.38	58.00	161.00	31.99
SE3	24.00	59.25	74.50	71.55	89.75	117.00	25.83
SE4	26.00	59.00	66.00	80.57	109.00	168.00	36.25
SE5	27.00	32.00	60.00	88.20	130.25	268.00	68.75

Table 3: Statistics of merges per project

Project	Min	25Q	Median	Mean	75Q	Max	St. Dev.
M1	0.00	0.00	2.00	4.37	6.00	18.00	5.20
M2	0.00	1.00	1.00	3.22	5.00	8.00	3.03
M3	3.00	10.00	13.00	13.58	18.00	27.00	6.45
SE1	20.00	31.00	44.00	44.24	53.00	120.00	20.02
SE2	7.00	19.00	21.00	27.48	33.00	90.00	17.58
SE3	9.00	31.00	42.50	41.10	51.00	82.00	18.38
SE4	17.00	24.00	35.00	36.76	44.00	71.00	14.85
SE5	21.00	26.00	42.00	52.69	69.50	153.00	35.86

Table 4: Statistics of churn per project

Project	Min	25Q	Median	Mean	75Q	Max	St. Dev.
M1	73.00	2,551.00	7,583.00	8,608.00	11,080.00	32,074.00	8,770.05
M2	219.00	384.00	2,614.00	8,197.00	9,031.00	45,382.00	14,429.00
M3	1,890.00	2,111.00	3,226.00	3,721.74.00	4,192.00	9,010.00	2,110.16
SE1	2,744.00	10,122.00	50,532.00	85,284.44	73,007.00	542,473.00	135,632.02
SE2	3,519.00	8,169.00	14,753.00	24,596.00	35,873.00	87,595.00	24,112.04
SE3	1,394.00	18,085.00	35,121.00	116,007.40	144,694.00	742,620.00	173,484.79
SE4	2,421.00	7,814.00	14,081.00	196,217.80	45,924.00	3,494,945.00	756,719.00
SE5	1,717.00	5,104.25	10,001.00	41,563.81	21,021.30	391,053.00	96,828.21

Figure 3: Histograms of student contributions (commits, merges, churn) per project.

M1 delivered more functionality while project M3 had a more stable process. The projects show a right-skewed distribution of contributions, except for the commits of M2. Inequality of contributions was more evident in merges: 7 students out of 28 for projects M1 and M2 made zero merges, meaning that 25% of students did not participate in integration efforts. Another key inequality indicator was the difference between the highest and lowest contributors. M1, M2, and M3 ratio of contribution between the top and bottom contributors for commits were 61, 35 and 6.92 respectively, for merges were *NA* (no contribution from the bottom contributor), *NA* and 9 respectively, and for churn was 439.37, 207.224 and 4.77 respectively. Considerable variations in contribution were found across all projects and contribution types being at least 13.44 for commits, 3.03 for merges and 2,110.16 for churn. Only in SE2 and SE5, there was a student who was the top contributor across all the contributions, while only in SE2 there was a student who was the bottom contributor for all the contributions. Differences in contribution do not necessarily imply differences in effort, as there might be students with fewer contributions but implement code with greater difficulty in each contribution. Therefore, it is essential to take into consideration other aspects of the contributions.

On the other hand, results for SE1, SE2, SE3, SE4, and SE5 projects had at least twice more contributions than M1, M2, and M3. Higher churn contributions in SE projects can be attributed to development frameworks generating code automatically. While for commits and merges, the higher contribution might be because SE is a required course and M is an elective course, hence students put more effort into the course. Moreover, most distributions of contributions were right-skewed, except for commits and merges in SE3, which were close to a uniform distribution. SE1, SE2, SE3, SE4, and SE5 ratio of contribution between the top and bottom contributors for commits were 11.72, 10.06, 4.88, 6.46 and 9.93 respectively, for merges were 6, 12.86, 9.11, 4.18 and 7.29 respectively, and for churn were 197.694, 24.89, 532.73, 1,443.6 and 227.754 respectively. Variations in contributions were at least 25.83 commits, 17.58 merges and 24,112.04 churn. The maximum variation was 70.17 commits, 35.86 merges and 756,719 churn.

Fig. 4 shows the Venn diagram for commit and merges contributions per module. In mobile projects, the modules were the code, the user interface (UI), and the tests. For the SE projects, the modules were the models, controllers, views, unit, integration and user interface tests (tests), and database (DB). In mobile projects, M2 and M3 had different behavior than M1. In M2 and M3, students mostly contributed to code, followed by the UI and then tests. For both projects, most contributions involved both the code and the UI. Meanwhile, M1 still had code as the biggest module, followed by UI and in last place tests. However, tests were only performed in changes that also modify the code, which may be because code was not testable and required changes. For SE projects, the most contributed modules across all projects were the views followed by the controllers. After these modules, models were more contributed than the DB in projects SE2, SE4 and SE5, while for SE1 and SE3 they contribute more to the DB than models. The least contributed module was the tests for all the projects. In all these projects, changes to only the views module were the most frequent changes. The number of cross-module changes was high, indicating that the commits and merges tend to involve multiple modules. For changes to multiple modules, the most frequent ones for SE1 and SE3 were changes to all modules, controllers and views, and DB and models respectively, indicating a high prevalence of changes modifying multiple modules with an emphasis on similar modules. In SE2, the most frequently changed modules were the DB and models, all modules, and the controllers, views and models. Finally, for SE4 and SE5 the order was controllers and views, DB and models, and all models indicating fewer cross-module changes in general than other projects.

Inequality indexes for commits, merges, and churn is shown in Table 5. The indexes are classified as low inequality if it is less than 0.34, moderate if it is more than or equal to 0.34 and less than 0.67, and high if it is more than or equal to 0.67. A value of 0 indicates perfect equality. Overall, results show that generally there was low to moderate inequality for commits and merges, and moderate to high inequality for churn (values close to 1 mean higher inequality). Mainly, SE projects show low inequality for commits and merges, indicating that students had a similar total number of commits and merges by the end of the course. This could be partially explained by the fact that in these projects development practices such as continuous integration and steady pace were emphasized. Yet, for churn, inequality was moderate to high. For mobile projects, commits and merges had a low to moderate level of inequality. Churn had a moderate to high inequality for M1 and M2, while M3 had low inequality. This low inequality might be due to the development practices emphasized and having the least amount of churn contributed leading to less variation between students. The high churn inequality exhibited across courses demonstrates that inequality was not dependent on development tools.

The inter-decile ratios for commits, merges and churn by the project are shown in Fig. 5. Bars represent the ratio scaled to a percentage, with green and blue representing bottom and top contributors, respectively. The number at the right of each bar corresponds to the ratio. Results show that the tails of contribution distributions were considerably unequal. The inequality for the 20/20 and 50/50 ratio is classified as low if it is less than or equal to 2, moderate if it is more than 2 and less than or equal to 3, and high if it is more

(a) Project M1

(b) Project M2

(c) Project M3

(d) Project SE1

(e) Project SE2

(f) Project SE3

(g) Project SE4

(h) Project SE5

Figure 4: Venn diagrams of the modules contributed to by students in commits and merges.

Table 5: Inequality indexes of student contributions (commits, merges, and churn) per project.

Project	Commits			Merges			Churn		
	<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>	<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>	<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>
M1	0.2985	0.4238	0.3023	0.4629	0.5961	0.2744	0.3593	0.4951	0.4237
M2	0.3333	0.4212	0.3303	0.4176	0.4751	0.3105	0.5385	0.6935	0.9422
M3	0.1713	0.2419	0.0994	0.1891	0.2615	0.113	0.2024	0.2709	0.1279
SE1	0.2266	0.3179	0.1918	0.1513	0.2147	0.0832	0.5041	0.6612	0.8257
SE2	0.2061	0.2888	0.1484	0.2092	0.2959	0.1569	0.3791	0.4914	0.3994
SE3	0.1435	0.1991	0.0680	0.1629	0.2404	0.1038	0.4890	0.6337	0.7413
SE4	0.1855	0.2429	0.0934	0.1611	0.2198	0.0755	0.8006	0.8809	2.2453
SE5	0.3129	0.3937	0.2556	0.2529	0.3373	0.1882	0.6271	0.7522	1.2483

than 3. For the Palma ratio, it is classified as low if it is less than or equal to 1, moderate if it is more than 1 and less than or equal to 2, and high if it is more than 2.

The 20/20 ratio commits ranged between 3.16 and 34.5, merges between 2.84 and *NA* (bottom contributors did not contribute at all), and churn between 3.48 and 151.87. The projects exhibit high inequality across all contribution types, except for merges in SE1 and SE4 that had moderate inequality. Churn had a higher inequality in the tails than commits and merges as all projects except M3 had a churn inter-decile ratio higher than 14.96, indicating high inequality. The Palma ratio commits ranged between 0.6 and 2.13, merges between 0.77 and 7.75, and churn between 1 and 55.69. M3, SE3 and SE4 for both commits and merges, and SE1 for merges had values under 1, indicating low inequality between the top 10% and the bottom 40%. Merges for M1 were highly unequal as the ratio was 7.75. For churn, M3 had low inequality and the rest of the projects had high inequality ranging from 3.56 to 55.69. Lastly, for the 50/50 ratio, commits ranged between 1.79 and 3.93, merges between 1.82 and 9.62, and churn between 2.07 and 42.56. Commits and merges had low to moderate inequality for M3, SE1, SE2, SE3, and SE4, with values ranging from 1.79 to 2.28. While M1, M2, and SE5 had high inequality in commits and merges ranging from 3.03 to 9.62. The churn ratio for all projects except M3 was over 4.96 indicating high inequality. The churn inequality for M3 was moderate. The results, therefore, suggest that the inequality between the top and bottom contributors showed significant differences between the top and bottom contributors. Only M3 had atypical low inequality across all ratios. This low inequality may be due to the development practices emphasized in the course.

Figure 5: Inter-decile ratios of student contributions (commits, merges, churn) per project.

4.1.2 Team contribution to the project

The contributions of teams to projects are shown in Fig. 6 for each project and contain three Pareto charts (for commits, merges, and churn). In each chart, the horizontal axis displays the teams in decreasing order of contributions, the left vertical axis shows the frequency of contributions, and the right vertical axis represents the cumulative percentage of contributions. Overall, these results show high inequality in churn contributions, with moderate inequality in commits and merges. Top contributing teams vary across contribution types.

Figure 6: Pareto charts of team contributions (commits, merges, churn) per project.

M1 and M2 had high inequality for churn, and moderate to high inequality for commits and merges. Meanwhile, M3 had low inequality across all contributions. M1's highest and lowest contributing teams yield 150 and 47 commits, 52 and 8 merges, and 57,421 and 27,394 churn, respectively. M2's highest and lowest contributing teams yield 81 and 30 commits, 16 and 6 merges, and 45,985 and 11,383 churn, respectively. M3's highest and lowest contributing teams yield 198 and 154 commits, 86 and 52 merges, and 29,246 and 15,233 churn, respectively. The cumulative percentage of contributions also evidences high inequality. Two of the four teams for M1 contributed with 75% of commits, 80% of merges, and 66% of churn. In M2, one of three teams contributed 53% of commits, 55% of merges, and 62% of churn. Lastly, for M3 two of four teams contributed 54% of commits, 57% of merges, and 63% of churn.

For SE projects, the inequality for commits and merges was moderate, but the churn inequality was high. The highest and lowest contributing teams for SE1 yield 688 and 371 commits, 286 and 190 merges, and 930,438 and 126,044 churn, respectively. SE2's highest and lowest contributing teams yield 395 and 167 commits, 238 and 84 merges, and 182,134 and 54,482 churn, respectively. SE3's highest and lowest contributing teams yield 465 and 292 commits, 310 and 150 merges, and 1,011,330 and 168,457 churn,

respectively. The highest and lowest contributing teams for SE4 yield 574 and 234 commits, 291 and 92 merges, and 3,566,917 and 115,479 churn, respectively. Lastly, the highest and lowest contributing teams for SE5 yield 829 and 229 commits, 463 and 183 merges, and 529,968 and 32,420 churn, respectively. SE1 project's cumulative percentage shows that two of the five teams produced 49% of commits, 46% of merges, and 77% of churn. In SE2, two of the four teams contributed 66% of commits, 68% of merges, and 70% of churn. For SE3, two of the four teams generate 59% of commits, 62% of merges, and 78% of churn. Two of the four teams for SE4 contributed with 62% of commits, 66% of merges, and 93% of churn. Lastly, one of the three teams for SE5 contributed with 59% of commits, 55% of merges, and 80% of churn. Only SE5 had the same team as the highest contributor across all contribution types.

Table 6 contains the inequality indexes of team contributions to projects. These results show that inequality among teams was low to moderate. For SE projects, inequality indexes indicate low inequality for commits and merges. The churn inequality for SE2 and SE3 was low, while for SE1, SE4, and SE5 was low to moderate. Hence, there was low to moderate inequality among teams' contributions. All mobile projects had low inequality across all contributions; particularly, M3 had a very low inequality. M2 had higher inequality indexes than M1 for commits and merges, but lower inequality indexes for churn. The indexes of team contributions to projects were less than the indexes of student contributions to the projects. This may indicate that top and bottom contributors were distributed across teams.

Table 6: Inequality indexes of team contributions (commits, merges, churn) per project.

Project	Commits			Merges			Churn		
	<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>	<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>	<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>
M1	0.2487	0.2606	0.1307	0.3765	0.4127	0.3268	0.1632	0.1734	0.0554
M2	0.1961	0.2222	0.0876	0.2184	0.2299	0.1014	0.2900	0.3127	0.1813
M3	0.0403	0.0513	0.0042	0.0833	0.1008	0.0189	0.1636	0.1813	0.0656
SE1	0.0922	0.1210	0.0265	0.0586	0.0763	0.0111	0.3742	0.4098	0.3025
SE2	0.1562	0.1837	0.0557	0.1794	0.2231	0.0829	0.2035	0.2253	0.0978
SE3	0.0863	0.1036	0.0185	0.1271	0.1582	0.0444	0.2811	0.3222	0.1855
SE4	0.1241	0.1625	0.0468	0.1580	0.2079	0.0738	0.6156	0.634	0.8518
SE5	0.2546	0.2837	0.1447	0.2159	0.2214	0.0982	0.4636	0.4988	0.4820

The inter-decile ratios of team contributions per project are shown in Fig. 7. The inequality in team contributions was lower than in student contributions, yet there was considerable inequality among teams. The 20/20 ratio commits ranged between 1.29 and 3.62, merges between 1.51 and 6.5, and churn between 2.1 and 30.89. Commits and merges for M1, commits for SE5 and merges for SE4 indicated high inequality, while the other projects' commits and merges indicated low to moderate inequality. The churn inequality for M1 and M3 and SE2 was low to moderate ranging from 2.1 to 2.3, while for M2, SE1, SE2, SE3, SE4, and SE5 it was high, being at least 3.34. The Palma ratio commits ranged between 0.61 and 1.58, merges between 0.77 and 3.06, and churn between 1.04 and 12.45. Projects M3, SE1, SE3, and SE4 had low inequality for commits. The merges inequality for ME3 and SE1 was low, for SE1 was high, and for the rest of the projects was moderate. Churn's inequality was high for SE1, SE4 and SE5, and low for the other projects. As the Palma ratio measures populations that were almost 0 and requires a complete team to calculate, the measure was analyzing the top 20 – 33% with the bottom 40 – 67% of the projects. The 50/50 ratio commits ranged between 1.18 and 2.98, merges between 1.19 and 3.88, and churn between 1.7 and 13.38. M1 and SE5 had a moderate inequality for commits, while the other projects had a low commit inequality. The merge inequality was high for M1, moderate for SE2, and low for the other projects. Churn had low inequality for M1 and M3, moderate inequality for M2 and SE2, and high inequality for SE1, SE3, SE4 and SE5. As most team contribution ratios were lower than the corresponding student contribution ratios, this may corroborate that top and bottom performers were distributed across teams.

4.1.3 Student contribution to the team

The contributions of students within their teams for the projects are shown in Fig. 8. Overall, we found varying degrees of inequality within teams.

M1 minimum contributions ranged from 1 to 20 commits, 0 to 9 merges, and 73 to 3,928 churn, while the maximum contributions ranged from 19 to 61 commits, 3 to 18 merges, and 13,278 to 32,074 churn. Team 4 minimum contributor contributed more in commits than the maximum contributor for teams 1 and 2, and contributed more in merges than the maximum contributor of all the other teams. Hence, a student

Figure 7: Inter-decile ratios of team contributions (commits, merges, churn) per project.

being a bottom contributor in their team does not necessarily indicate them being a bottom contributor in the project. In the teams with five students, the two students who contributed the most in each contribution had a cumulative percentage ranging from 63% to 68% of the commits, 56% to 100% of the merges, and 65% to 88% of the churn. This high cumulative percentage for merges in teams 2 and 3 was due to three and two students, respectively, not merging in their team. In the team with four students, two students contributed 68% of the commits, 60% of the merges, and 79% of the churn. The top contributor was the same across all contributions for two teams and was the same for two types of contributions for two teams. The bottom contributor was the same for all contributions in two teams and was the same for two contributions in two teams. Furthermore, in team 3 the student with the highest commits and churn was also one of the lowest contributors in merges indicating that being a top contributor in a metric does not necessarily indicate being a high performer across all contributions.

For M2, minimum contributions ranged from 1 to 19 commits, 0 to 1 merges, and 219 to 2,614 churn, while the maximum contributions ranged from 20 to 35 commits, 5 to 8 merges, and 9,031 to 45,382 churn. The cumulative percentage of contributions had one student per team contributing from 42% to 83% of the commits, 50% to 83% of the merges, and 55% to 99% of the churn. Though teams 1 and 2 had high inequality in the contributions within the team (with contributions for the top developer being at least 67%), team 3 had more equal contributions ranging from 42% to 55%. Therefore, the inequality in contributions was different for each team. The top contributor was the same across all contributions for two teams and was the same for two types of contributions in one team. The bottom contributor was the same student across all contributions for all of the teams.

M3 minimum contributions ranged from 13 to 21 commits, 3 to 10 merges, and 1,890 to 3,226 churn, while the maximum contributions range from 46 to 90 commits, 15 to 27 merges, and 3,006 to 9,010 churn. Teams with five students cumulative percentage had two students contributing between 51% to 53% for commits, 52% to 61% for merges, and 46% to 61% for churn. The team with four had two students' cumulative percentage as 71% for commits, 72% for merges, and 70% for churn. The top contributor for all contributions was the same in two teams, was the same for two contributions in one team, and was different for all contributions in one team. The bottom contributor for all contributions was the same in two teams, was the same for two contributions in one team, and was different for all contributions in one team. The team with different students as the bottom contributor across the contribution types was not the same team with different students as top contributors across the contribution types; hence, having distributed bottom contributors does not necessarily indicate having distributed top contributors.

Regarding SE projects, SE1 minimum contributions ranged from 32 to 70 commits, 20 to 33 merges, and 2,744 to 10,227 churn. The difference in minimum commit contributions for team 1 from the other teams was substantial as it was 70 for team 1, but ranged from 32 to 41 in other teams. Furthermore, a similar pattern was found in the minimum churn contributions for teams 1 and 5, ranging from 10,122 to 10,227, with the other teams, ranging from 2,744 to 3,912. Therefore, the contributions of bottom contributors in teams may not always be similar. The maximum contributions range from 135 to 375 commits, 50 to 120 merges, and 65,195 to 542,473 churn. The cumulative percentage of contributions had two students out of five contributing 55% to 67% of the commits, 50% to 60% of the merges, and 60% to 88% of the churn. The churn contributions for team 5 were substantially lower for team 5, being 60%, with the other teams ranging from 84% to 88%, hence churn was much more similar in team 5 than other teams. The top contributor for all contributions was the same in one team, for two contributions was the same in two teams, and was

Figure 8: Pareto charts of student contributions within teams.

Figure 8: Pareto charts of student contributions within teams (cont.).

Figure 8: Pareto charts of student contributions within teams (cont.).

different for all contributions in two teams. The bottom contributor was the same across all contribution types for one team, the bottom contributor was the same for two types of contributions for two teams, and the bottom contributor was different for each contribution type for two teams. The team that only had one student as the top contributor across all contributions was also one of the teams with distributed bottom contributors. On the other hand, one of the teams with distributed top contributors was also the only team with the same student as the bottom contributor for all contribution types. Therefore, distributed top or bottom contributors does not indicate that the other top or bottom contributor may also be distributed.

In SE2, minimum contributions ranged from 16 to 32 commits, 7 to 17 merges, and 3,519 to 5,324 churn, while the maximum contributions ranged from 58 to 161 commits, 28 to 90 merges, and 18,807 to 87,595 churn. In the team with seven students, two students contributed 54% of the commits, 52% of the merges, and 63% of the churn. In the teams with five students, two students' contributions ranged from 50% to 57% of the commits, 52% to 56% of the merges, and 53% to 62% of the churn. Finally, in the team with four students, two students contributed 75% of the commits, 69% of the merges, and 93% of the churn. The top contributor was the same across all contribution types for two teams and was the same for two types of contributions for two teams. The bottom contributor was the same across all contributions for one team and was the same for two types of contributions in three teams. We also found that the teams with only one student as the top contributor of all the contributions were not the same teams with only one student as the bottom contributor.

SE3 minimum contributions ranged from 24 to 48 commits, 9 to 31 merges, and 1,394 to 18,899 churn, while the maximum contributions ranged from 94 to 117 commits, 51 to 82 merges, and 123,887 to 742,620 churn. The cumulative percentage of contributions for two students in teams of four ranges from 62% to 64% of the commits, 61% to 73% of the merges, and 87% to 93% of the churn. In teams of six, two students' contributions ranged from 41% to 52% of the commits, 46% to 47% of the merges, and 67% to 75% of the churn. The churn seems to be substantially lower in teams with six team members than teams with four team members. The top contributor in three teams was the same student for two types of contributions, and was different for the contribution types for one team. The bottom contributor was the same for all contributions in three teams and was the same for two contributions in one team.

For SE4, minimum contributions ranged from 26 to 46 commits, 17 to 35 merges, and 2,421 to 17,018 churn, while the maximum contributions ranged from 89 to 168 commits, 32 to 71 merges, and 46,022 to 3,494,945 churn. The minimum contributor for team 3 outperforms in merges the top contributor of team 2. In teams of five, two students' contributions ranged from 57% to 64% for the commits, 48% to 54% for the merges, and 54% to 89% for the churn. In teams of six, two students contributed from 44% to 52% of the commits, 42% to 53% of the merges, and 88% to 99% of the churn. The top contributor was the same across all contributions for one team, was the same for two contributions for two teams, and was different for each contribution for one team. The bottom contributor was the same for all contributions in two teams, was the same for two contributions in one team, and was different for all contributions in one team. The team with the distributed top contributors was the same team with the distributed bottom contributors, while the team with only one student as the top contributor was one of the teams with only one student being the bottom contributor.

Lastly, in SE5 minimum contributions ranged from 27 to 111 commits, 21 to 54 merges, and 1,717 to 7,307 churn, while the maximum contributions ranged from 66 to 268 commits, 53 to 153 merges, and 17,511 to 391,053 churn. Team 3's minimum contributor for commits and merges outperforms the maximum contributor of team 1. In the six-member team, two students contributed to 47% of the commits, 46% of the merges, and 63% of the churn. The five-member teams had two students' contributions ranging from 54% to 65% of the commits, 53% to 60% of the merges, and 72% to 94% of the churn. The top contributor was the same across all contributions for one team and was the same for two contributions for two teams. The bottom contributor was the same for two contributions in two teams and was different for all contributions in one team.

Inequality indexes of student contributions within teams for the projects are shown in Table 7. Results show that inequality among team members' contributions was low to moderate, suggesting that top and bottom performers were distributed across teams.

Regarding the mobile projects, the behavior of projects M1 and M2 were somewhat similar, while project M3 was considerably different. For projects M1 and M2, commits had low to moderate inequality; merges had low, moderate, and high inequality; and churn had low, moderate, and high inequality. Commits and merges tend to had low inequality for M1 and moderate inequality for M2, while churn tends to had moderate inequality for both projects. The only team for these projects that had low inequality across all metrics was team 3 of M2. M3 had, for all of the metrics, low inequality in the teams. Hence, inequality among teams varies greatly: some teams had similar contributions per team member, while others vastly differ in their team members' contributions. Regarding SE projects, commits and merges had low inequality for all

Table 7: Inequality indexes of student contributions within teams.

Project	Team	Commits			Merges			Churn		
		<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>	<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>	<i>Hoover</i>	<i>Gini</i>	<i>Theil</i>
M1	1	0.233	0.333	0.217	0.200	0.267	0.017	0.420	0.488	0.448
	2	0.281	0.340	0.190	0.600	0.700	0.131	0.483	0.573	0.601
	3	0.233	0.288	0.137	0.400	0.486	0.038	0.294	0.347	0.235
	4	0.209	0.246	0.110	0.096	0.135	0.031	0.288	0.358	0.217
M2	1	0.333	0.422	0.354	0.500	0.556	0.243	0.537	0.559	0.640
	2	0.500	0.540	0.580	0.381	0.381	0.302	0.654	0.655	1.020
	3	0.099	0.124	0.027	0.271	0.292	0.217	0.217	0.261	0.118
M3	1	0.129	0.183	0.074	0.210	0.271	0.136	0.213	0.233	0.097
	2	0.217	0.283	0.133	0.221	0.242	0.114	0.205	0.243	0.100
	3	0.113	0.158	0.042	0.169	0.200	0.071	0.071	0.081	0.011
	4	0.178	0.194	0.077	0.147	0.186	0.056	0.127	0.162	0.045
SE1	1	0.345	0.358	0.291	0.220	0.284	0.147	0.474	0.574	0.598
	2	0.180	0.245	0.102	0.122	0.131	0.031	0.483	0.522	0.525
	3	0.236	0.282	0.131	0.167	0.214	0.079	0.522	0.591	0.678
	4	0.147	0.208	0.076	0.153	0.168	0.055	0.439	0.536	0.528
	5	0.216	0.287	0.145	0.098	0.116	0.022	0.238	0.279	0.152
SE2	1	0.265	0.331	0.226	0.235	0.329	0.202	0.345	0.464	0.394
	2	0.105	0.126	0.027	0.133	0.162	0.044	0.216	0.272	0.125
	3	0.243	0.274	0.134	0.214	0.286	0.151	0.533	0.586	0.672
	4	0.169	0.228	0.083	0.165	0.210	0.070	0.173	0.218	0.138
SE3	1	0.120	0.148	0.044	0.138	0.157	0.062	0.485	0.539	0.554
	2	0.140	0.187	0.067	0.233	0.317	0.180	0.484	0.571	0.609
	3	0.189	0.270	0.124	0.184	0.235	0.104	0.341	0.463	0.407
	4	0.094	0.126	0.026	0.128	0.172	0.048	0.473	0.537	0.585
SE4	1	0.167	0.192	0.062	0.077	0.088	0.013	0.172	0.192	0.070
	2	0.139	0.203	0.078	0.098	0.125	0.028	0.582	0.596	0.753
	3	0.154	0.192	0.067	0.089	0.124	0.026	0.546	0.600	0.684
	4	0.187	0.251	0.114	0.210	0.259	0.108	0.813	0.818	1.663
SE5	1	0.134	0.164	0.053	0.131	0.176	0.056	0.322	0.419	0.302
	2	0.251	0.296	0.144	0.199	0.232	0.088	0.340	0.417	0.315
	3	0.144	0.177	0.054	0.131	0.181	0.059	0.544	0.655	0.838

teams except SE1's commits for team 1. Churn had low, moderate, and high inequality, with a tendency for most teams of moderate inequality. Still, team 5 of SE1, teams 2 and 4 of SE2, and team 1 of SE4 had low inequality in churn for all metrics. In SE4 and SE5, four out of 7 teams had a churn metric that indicates high inequality. Hence, SE projects had low inequality in general, while the churn shows no clear tendency for the teams with a large disparity in churn, and teams with more uniform churn.

The inter-decile ratios of student contributions within teams are shown in Fig. 9. We found inequality between the top and bottom contributors within teams.

The 20/20 ratio for M1 had commits between 3.05 and 19, merges between 2 and *NA*, and churn between 5.93 and 181.89. High inequality was exhibited across all contribution types except the merges for team 4, with a moderate inequality. Team 4 had the lowest inequality for all of the contributions. The Palma ratio had commits ranging between 1.45 and 2.38, merges between 0.86 and *NA*, and churn between 2.40 and 10.43. Teams 1 and 2 had high inequality for commits, while teams 3 and 4 had moderate inequality. The merges were high for team 2 and 3, moderate for team 1, and low for team 4. The churn had high inequality for all the teams. The 50/50 ratio ranged between 2.15 and 2.6 commits, 1.48 and *NA* merges, and 2.26 and 8.01 churn. The commits had moderate inequality for all the teams. The merges had low inequality, for teams 1 and 4, and high inequality, for teams 2 and 3. The churn had high inequality, except for team 3 with moderate inequality.

For M2, the 20/20 ratio's commits ranged between 1.79 and 35, merges between 5 and *NA*, and churn between 3.45 and 207.22. All teams had high inequality, except team 3 in commits with low inequality. The Palma ratio ranges between 0.72 and 5 for commits, 1 and 5 for merges, and 1.22 and 75.26 for churn. Team 3 had a low inequality for commits and merges with a moderate inequality for churn. Team 1's commit inequality was moderate. The rest of the ratios exhibit high inequality. The 50/50 ratio ranges between 1.32 and 5.86 commits, 1.88 and 6 merges, and 1.87 and 75.9 churn. Team 3's contributions had low inequality, the commits for team 1 and the merges for team 2 had moderate inequality, and the rest of the contributions had high inequality.

In M3, the 20/20 ratio's commits ranged between 2.35 and 4.29, merges between 2.7 and 6.67, and churn between 1.46 and 3.01. The commits had high inequality except for team 3, the merges had high inequality except for team 4, and the churn had high inequality for team 2, moderate inequality for team 1 and 4, and low inequality for team 3. The Palma ratio had commits ranging between 1.04 and 1.61, merges between 1.12 and 1.54, and churn between 0.72 and 1.43. The inequality of the contributions for the teams of M3 was moderate, except for the churn of team 3 that had low inequality. The 50/50 ratio's commits ranged between 1.45 and 2.54, merges ranged between 1.59 and 2.59, and churn ranged between 1.23 and 2.38. Team 2 of M3 had moderate inequality for the ratio, while all the other teams had low inequality across all contributions.

The 20/20 ratio for SE1 had commits ranged between 3.78 and 5.36, merges ranged between 1.73 and 4, and churn ranged between 6.44 and 197.69. The inequality was high for most contributions except in merges, where teams 3 and 4 had moderate inequality, and teams 2 and 5 had low inequality. The Palma ratio for SE1 had commits ranging between 1.22 and 2.52, merges between 0.86 and 2, and churn between 1.91 and 18.31. For commits, the teams had moderate inequality except for team 1. The merges had moderate inequality except for teams 2 and 5 with low inequality. The churn had high inequality except for team 5 with moderate inequality. The 50/50 ratio had commits ranging between 1.63 and 2.35, merges between 1.36 and 2, and churn between 2.08 and 8.06 for SE1. The commits had moderate inequality in teams 1, 3 and 5, and low inequality in teams 2 and 4. Merges had low inequality. The churn had high inequality except for team 5 with moderate inequality.

For SE2, the 20/20 ratio commits ranged between 1.81 and 4.19, merges between 2.15 and 5.57, and churn between 5.21 and 21.95. The commits' inequality was low for team 2 and high for the rest. The merges' inequality was moderate for teams 2 and 4 and high for teams 1 and 3. The churn's inequality was high for all teams. The Palma ratio ranges between 0.89 and 1.79 commits, 1.04 and 1.7 merges, and 1.21 and 10.53 churn. The commits had moderate inequality except for team 2, with low inequality. The merges had moderate inequality for all the teams. The churn was moderate for teams 2 and 4, and high for teams 1 and 3. The 50/50 ratio ranges between 1.38 and 2.88 commits, 1.54 and 2.28 merges, and 1.64 and 12.45 churn. The commits and merges for teams 1 and 3 were moderate, while for teams 2 and 4 it was low. The churn inequality for teams 1 and 3 was high, team 2 was moderate and team 4 was low.

In SE3, the 20/20 ratio's commits ranged between 1.69 and 3.59, merges between 1.96 and 7.67, and churn between 8.15 and 39.29. The inequality for commits and merges for team 4 was low, for team 1 was moderate, and was high for teams 2 and 3. The churn had high inequality for all the teams. The commits ranged between 0.54 and 1, merges between 0.71 and 1.73, and churn between 2.44 and 10.04 for the Palma ratio. The commits had low inequality for all teams. The merges had low inequality for all teams except team 2 with moderate inequality. Churn's inequality was high across all teams. The 50/50 ratio ranges between 1.46 and 2.2 commits, 1.56 and 2.75 merges, and 5.03 and 12.67 churn. Teams 1 and 4 had low

Figure 9: Inter-decile ratios of students contributions within teams.

Figure 9: Inter-decile ratios of students contributions within teams (cont.).

inequality for commits and merges. Team 2 had low inequality for commits, and moderate inequality for merges. Team 3 had moderate inequality for commits and merges. The churn inequality was high across all teams.

The 20/20 ratio for SE4 ranges between 2.48 and 3.42 commits, 1.5 and 2.95 merges, and 2.7 and 213.36 churn. Teams 1, 3 and 4 had moderate inequality for commits, while team 2 had high commit inequality. The merges had low inequality except for team 4 with moderate inequality. The churn had high inequality except for team 1 with moderate inequality. The Palma's ratio commits ranged between 0.67 and 1.1, merges ranged between 0.59 and 1.03, and churn ranged between 1.18 and 119.43. The inequality for commits was moderate, except for team 3 with low inequality, for merges was low except for team 4 with moderate inequality, and for churn was high except for team 1 with moderate inequality. The commits ranged between 1.69 and 2.07, merges between 1.27 and 2.44, and churn between 1.67 and 120.89 for the 50/50 ratio. Teams 1, 2, and 3 had low inequality for commits and merges, while team 4 had moderate inequality. Churn's inequality was high, except for team 1 with low inequality.

In SE5, the 20/20 ratio ranges between 1.84 and 4.23 commits, 1.98 and 3 merges, and 9.01 and 53.52 churn. The inequality for commits and churn for team 1 was low, for team 3 was moderate, and for team 2 it was high and moderate, respectively. The churn had high inequality for all the teams. The commits ranged between 0.73 and 1.9, merges between 0.77 and 1.38, and churn between 2.22 and 24.2 for the Palma ratio. Team 1 had low inequality for commits and merges, while teams 2 and 3 had moderate inequality. The churn exhibits high inequality across all teams. The 50/50 ratio ranges between 1.54 and 2.3 for commits, 1.51 and 1.89 for merges, and 2.97 and 17.22 for churn. The commits had low inequality for all teams except team 2 with moderate inequality. The merges exhibit low inequality for all teams. The churn had high inequality for the teams, except for team 2 with moderate inequality.

4.2 RQ2: Contribution evolution over time

To answer this research question, we analyzed the frequencies of commits and merges over time.

4.2.1 Student contribution over time

A heat map of the contribution frequency of students to projects over time is shown in Fig. 10. The horizontal axis depicts time (tags on sprint milestones), and the vertical axis shows contributions (commits and merges) per project. According to the legend on the right, frequency is represented by a color: high frequencies are dark blue and low frequencies are light green. Data shows that students were contributing at a non-steady pace.

Mobile projects show a slow start, but frequency improves slightly over time. Still, we can see that the pace was not constant, as there were peaks around due dates. This indicates that the core agile principle of constant work was not being followed. Furthermore, contributions increase over time, as evident from hot zones toward the right end of the map. This might be because students get more familiar with the technologies, reducing learning effort, and increasing productivity. It may also be the result of increased feature expectations from instructors, and students investing more time in the project to meet those expectations. M3 had the most contributions per day, followed by M1 with M2 with the least amount of contributions.

SE projects behave similarly to mobile projects, as they start with a low contribution frequency but this frequency intensifies over time. Most contributions can be found around deadlines, though for SE2, SE3, SE4, and SE5 the peaks were reduced in the last sprints. SE projects had higher contributions per day than mobile projects. In the first sprint, merges happened very close to the deadline leading to difficult and time-consuming merges. However, after the first sprint, students improved their *continuous integration* skills to ease these problems and reduce integration defects.

4.2.2 Team contribution over time

A heat map of the contribution frequency of teams to projects over time is shown in Fig. 11. We found that the frequency pace varies between teams.

For SE projects, the frequency of contributions was not the same for all teams. Some teams commit and merge more frequently, while others contribute mostly around deadlines. The commits' highest frequency was concentrated in a few teams. Nonetheless, teams do improve their contribution frequency throughout the project.

Regarding mobile projects, we observed that most of the M1 and M2 teams do not contribute steadily. There were teams with significant gaps in contributions, like team 1 in M2. Yet some teams contribute more evenly, like team 3 in M2. This could indicate that not all teams demonstrate agile principles of constant work. This behavior could be explained by the fact that after each sprint, students devote their time to complete assignments for other courses.

Figure 10: Heat map of contribution frequency of students to projects over time.

Figure 11: Heat map of contribution frequency of teams to projects over time.

4.2.3 Student contribution within a team over time

Heat maps of the contribution frequency of students within teams over time are shown in Fig. 12. Data shows that there were variations in contribution frequency among students within a team.

In M1, it is noticeable that students in teams 3 and 4, in general, had more contributions. Team 3's students had the best frequency of commits, not showing wide gaps. Most merges effectuated by students were sporadic and mostly concentrated around deadlines. M2 also had very few and infrequent contributions. Merges, in general, were non-existent and concentrated around deadlines. Students of teams 1 and 2 committed almost exclusively around deadlines, while students of team 3 show a much steadier pace. M3 contrasts with the other mobile projects as there were more contributions and a steadier pace. Still, not all students had a steady pace, with some only contributing near deadlines. This phenomenon also occurs in teams where one student has a more frequent pace while another contributes near deadlines, like in team 4.

For SE projects, in SE1 we can notice that there were students that had a very steady pace almost always committing, while most students' frequency was not as steady. Merges seem to be performed around the same dates for all students in a team. SE2 had a steadier commit pace for most students in teams 1, 2, and 4. Still, within these teams, some team members do not follow this pattern like in team 4. Merges for students in team 1 seem to be more frequent than in the other teams. For SE3, teams 1 and 2 had fewer commit peaks than teams 3 and 4. Team 4 had a steadier merge frequency for all of the students than other teams. In SE4, team 3 had for most students a steady frequency, while in teams 1 and 4 the frequency between students varied more. Team 2 had a more sporadic frequency than the other teams. Finally, for SE5 the frequency and amount of contributions varied greatly between teams, with a similar pattern in the teams. Team 3 had the steadiest pace, followed by team 2, and lastly team 3. Still, for merges in team 1 some team members also do not seem to contribute at such a steady pace.

5 Discussion

The analysis of code contributions using *git* metrics allowed us to identify contributions from different perspectives. The churn metric was useful to determine the quantity of the contributions, while the commits and merges were helpful to evaluate the contribution process. Further research should be conducted to explore other attributes of contributions such as quality, complexity, difficulty, and value, providing more information about the contribution to a project.

The inequality in the contributions helps to determine how contributions are distributed among developers and teams. We found that all our projects had inequalities, hence, inequality metrics were useful to determine how unequal students contributions were within a project, team contributions were within a project, and students contributions were within their team. Additionally, it is important to recognize high achievers to reward them and low achievers to help them improve.

Analyzing the contribution to modules allows to identify cross-functional contributions by students. It is important for teachers to find out whether students are participating in all of the project functionality and gaining cross-functional skills needed in their careers. Tracking this might motivate students to make contributions to all of the modules. Further research of what students are contributing to and the ownership of these artifacts can be conducted.

The contribution frequency enables the discovery of patterns regarding the pace of contributions. This helped in identifying if student's pace was constant or determining which teams or students need to improve their pace. Also, the time of merge helped find out whether continuous integration practices were followed. Additional research could analyze the evolution and impact of other types of contribution aspects such as churn and quality.

The proposed approach can be used in any software development project that uses a Git-based repository, and is interested in gathering process and product data. Furthermore, metrics such as commits, merges, churn and inequality can be automatically calculated for any software development project with the goal of gaining an objective understanding of developers' contributions or determining how software engineering practices were followed. In the academic context instructors can benefit from a tool like this that aids in grading and provides data to identify improvement opportunities for courses. Likewise, students may benefit from getting objective, timely feedback on their performance, which helps them to improve their development skills. Moreover, this approach with the visualization of these metrics can be used in real-time sessions with students, in order to help them improve their software practices, such as in continuous integration and contribution pace.

With regards to how generalizable are the results, we found that students tend to work near deadlines and there is considerable inequality in the projects. This means that instructors must consider the fact that students do not work at a constant pace and consider grade assignment nuances in contribution dynamics where students do not participate in equal measure. Hence, information gathered from projects can be

Figure 12: Heat map of contribution frequency of students to teams over time.

Figure 12: Heat map of contribution frequency of students to teams over time (cont.).

used for improvement opportunities in courses. These results may be generalized to projects with similar characteristics; however, they would be influenced by aspects like students' motivation towards the project, group dynamic, year of study, previous knowledge, and specific characteristics of the developed project.

We acknowledge that our approach should be assessed in a complementary manner by gathering students' perceptions, including the benefits or limitations, of using these metrics. Their perceptions would be useful to determine if tools like ours really help improve the process, product, or skills. This could be part of a future analysis where other metrics such as quality of the contributions are investigated [25] and their impact in courses analyzed.

6 Conclusions

Measuring student and team contributions in project-based courses is a challenge. Therefore, instructors would greatly benefit from using metrics that help them objectively quantify students' and teams' contributions. In this work, we analyzed the contributions in eight projects with multiple collaborative teams, using two aggregate inequality metrics (inter-decile ratios and inequality indexes) and five *git* metrics (commits, merges, churn, commits per day and merges per day). Contributions were analyzed for each student and team in the project and students within their teams. We used aggregate measures such as inequality indexes and inter-decile ratios to determine the inequality of contributions in projects. We found considerable inequality in students' contributions with a focus on contributing to functionality. Furthermore, based on the daily commits and merges, we found that students do not have a steady pace, and improvements in contribution frequency are not seen across all students.

Our findings suggest that the distribution of contributions provides critical information for assessing the learning process. It helps in assessing student's contribution, and how well the methodologies are applied in the projects. Hence, the use of such metrics may benefit instructors and students alike. Furthermore, applying these metrics in a professional setting may be useful to detect risky projects that rely on few developers, reward developers for their contributions, or assign developers to tasks where they are more productive.

In future work, we plan to examine the complexity, quality, and value of the contributions. We believe such qualitative factors can complement an assessment based on quantitative metrics. Furthermore, we plan to analyze software ownership to determine trends in the components, modules, and files. Finally, we plan on gathering instructors' and students' perceptions to determine the benefits and limitations of using these metrics.

Acknowledgments

The work was partially supported by the Universidad de Costa Rica under projects No. 834-B8-A27 and 834-C1-011. We thank the Empirical Software Engineering Group at UCR.

References

- [1] G. Maturro, “Soft skills in software engineering: A study of its demand by software companies in uruguay,” in *2013 6th international workshop on cooperative and human aspects of software engineering (CHASE)*. IEEE, 2013, pp. 133–136.
- [2] V. Garousi, G. Giray, E. Tuzun, C. Catal, and M. Felderer, “Closing the gap between software engineering education and industrial needs,” *IEEE Software*, 2019.
- [3] C. Quesada-López and A. Martínez, “Implementation of project based learning: Lessons learned,” in *2019 XLV Latin American Computing Conference (CLEI)*, 2019, pp. 1–10.
- [4] J. E. Sims-Knight, R. L. Upchurch, T. Powers, S. Haden, and R. Topciu, “Teams in software engineering education,” in *32nd Annual Frontiers in Education*, vol. 3. IEEE, 2002, pp. S3G–S3G.
- [5] R. F. Gamble and M. L. Hale, “Assessing individual performance in agile undergraduate software engineering teams,” in *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2013, pp. 1678–1684.
- [6] K. Buffardi, “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*. Portland OR USA: ACM, Feb. 2020, pp. 650–656.
- [7] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno, “Collaboration tools for global software engineering,” *IEEE software*, vol. 27, no. 2, pp. 52–55, 2010.
- [8] S. Chacon and B. Straub, *Pro git*. Springer Nature, 2014.
- [9] R. M. Parizi, P. Spoletini, and A. Singh, “Measuring team members’ contributions in software engineering projects using git-driven technology,” in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–5.
- [10] J. Feliciano, M.-A. Storey, and A. Zagalsky, “Student experiences using github in software engineering courses: a case study,” in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2016, pp. 422–431.
- [11] J. Kelleher, “Employing git in the classroom,” in *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. IEEE, 2014, pp. 1–4.
- [12] M. Cochez, V. Isomöttönen, V. Tirronen, and J. Itkonen, “How do computer science students use distributed version control systems?” in *International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications*. Springer, 2013, pp. 210–228.
- [13] M. Mittal and A. Sureka, “Process mining software repositories from student projects in an undergraduate software engineering course,” in *Companion Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 344–353.
- [14] C.-Z. Kertész, “Using github in the classroom-a collaborative learning experience,” in *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)*. IEEE, 2015, pp. 381–386.
- [15] H. Gustavsson and M. Brohede, “Continuous assessment in software engineering project course using publicly available data from GitHub,” in *Proceedings of the 15th International Symposium on Open Collaboration*. Skövde Sweden: ACM, Aug. 2019, pp. 1–6.
- [16] M. Goeminne and T. Mens, “Evidence for the pareto principle in open source software activity,” in *the Joint Proceedings of the 1st International workshop on Model Driven Software Maintenance and 5th International Workshop on Software Quality and Maintainability*. Citeseer, 2011, pp. 74–82.
- [17] K. Yamashita, S. McIntosh, Y. Kamei, A. E. Hassan, and N. Ubayashi, “Revisiting the applicability of the pareto principle to core development teams in open source software projects,” in *Proceedings of the 14th International Workshop on Principles of Software Evolution*, 2015, pp. 46–55.
- [18] A. Agrawal, A. Rahman, R. Krishna, A. Sobran, and T. Menzies, “We don’t need another hero? the impact of “heroes” on software development,” in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, 2018, pp. 245–253.

- [19] S. Hamer, C. Quesada-López, A. Martínez, and M. Jenkins, “Measuring students’ contributions in software development projects using git metrics,” in *2020 XLVI Latin American Computing Conference (CLEI)*. IEEE, 2020.
- [20] C. Raibulet and F. Arcelli Fontana, “Collaborative and teamwork software development in an undergraduate software engineering course,” *Journal of Systems and Software*, vol. 144, pp. 409–422, Oct. 2018.
- [21] M. Tushev, G. Williams, and A. Mahmoud, “Using GitHub in Large Software Engineering Classes An Exploratory Case Study,” *Computer Science Education*, p. 28, 2020.
- [22] A. Cobham, L. Schlögl, and A. Sumner, “Inequality and the tails: the palma proposition and ratio,” *Global Policy*, vol. 7, no. 1, pp. 25–36, 2016.
- [23] V. Basili, J. Heidrich, M. Lindvall, J. Münch, M. Regardie, D. Rombach, C. Seaman, and A. Trendowicz, “Gqm+ strategies: A comprehensive methodology for aligning business strategies with software measurement,” *arXiv preprint arXiv:1402.0292*, 2014.
- [24] F. A. Cowell, “Measurement of inequality,” *Handbook of income distribution*, vol. 1, pp. 87–166, 2000.
- [25] P. Behnamghader, R. Alfayez, K. Srisopha, and B. Boehm, “Towards Better Understanding of Software Quality Evolution through Commit-Impact Analysis,” in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Jul. 2017, pp. 251–262.