A Comparative Analysis of GitHub Contributions Before and After An OSS Based Software Engineering Class

Jialin Cui
jcui9@ncsu.edu
North Carolina State University
Raleigh, USA

Runqiu Zhang
rz2cv@virginia.edu
University of Virginia
Charlottesville, USA

Ruochi Li
rli14@ncsu.edu
North Carolina State University
Raleigh, USA

Fangtong Zhou
fzhou@ncsu.edu
North Carolina State University
Raleigh, USA

Yang Song
songy@uncw.edu
University of North Carolina Wilmington
Wilmington, USA

Edward Gehringer
efg@ncsu.edu
North Carolina State University
Raleigh, USA

ABSTRACT

This study presents a comparative analysis of contributions to GitHub by students before and after participating in a Software Engineering class based on Open Source Software (OSS). The primary objective is to understand the influence of formal software engineering education on students' engagement in OSS projects, as reflected in their GitHub activities. The research addresses two key questions. Firstly, it examines how GitHub contributions change before and after the class. The corresponding hypothesis posits that students' average GitHub contributions will exhibit a distinct pattern post-class compared to pre-class. Additionally, the study explores the potential association between students' academic performance in the class and their level of GitHub contributions after the class. The strength and direction of the potential association are quantified using the Spearman correlation coefficient, considering the potential non-linear nature of the data. This analysis uses data from over 1000 students across more than 10 years, encompassing their GitHub contribution data over multiple timeframes and their grades in the class. The study employs a combination of statistical methods, including paired tests and correlation analysis, to explore these dynamics. While causality cannot be established due to the absence of a control group, the findings offer valuable insights into the correlation between academic engagement and practical contributions in the realm of OSS development. This research contributes to the understanding of how theoretical software engineering education might relate to practical application and engagement in real-world projects.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; **Student assessment**; • **Software and its engineering** → **Open source model**.

KEYWORDS

Software Engineering Education, Qualitative Study, Statistical Study, GitHub

ACM Reference Format:
Jialin Cui, Runqiu Zhang, Ruochi Li, Fangtong Zhou, Yang Song, and Edward Gehringer. 2024. A Comparative Analysis of GitHub Contributions Before and After An OSS Based Software Engineering Class. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653535

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ITiCSE 2024, July 8–10, 2024, Milan, Italy
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0600-4/24/07
https://doi.org/10.1145/3649217.3653535

1 INTRODUCTION

In the evolving landscape of software development, Open Source Software (OSS) has emerged as a pivotal force driving innovation and collaboration. With the burgeoning growth of OSS projects, there emerges an unparalleled opportunity for learning and contributing, especially for students in Software Engineering disciplines. This study delves into the nuanced intersection of formal software engineering education and active participation in OSS projects, as reflected through contributions on GitHub.

GitHub, a platform at the forefront of version control and collaborative software development, serves as a fertile ground for exploring individual contributions to various projects. These contributions, which include a range of activities such as commits, pull requests, and repository management, offer tangible metrics to gauge a developer's engagement in OSS. Our research aims to dissect these metrics to understand the nuanced impact of academic instruction on students' engagement with OSS.

Central to our investigation are two pivotal research questions:
RQ1. What are the patterns and magnitude of change in students' GitHub contributions before and after participating in a Software Engineering class focused on Open Source Software (OSS)?

This inquiry seeks to unravel the patterns and trends in student engagement with OSS projects, examining how formal education might influence these dynamics.
RQ2. To what extent does academic performance in the Software Engineering class correlate with the level of students' GitHub contributions in the subsequent period?

Here, we aim to explore the potential link between classroom success and active, meaningful participation in OSS development post-class.

The significance of this study lies in its potential to bridge the often-discussed gap between theoretical learning and practical application within the realm of software development. By analyzing the GitHub activities of over 700 students across multiple timeframes, coupled with their academic performance in a Software Engineering class, our research offers a unique lens through which to view how formal education may influence students' involvement in real-world software projects. While the absence of a control group limits our ability to make definitive statements about causality, our findings promise to shed light on the intricate relationship between academic engagement in software engineering and practical contributions to OSS.

Additionally, this study contributes to the broader discourse on the role of OSS in educational settings. By providing a detailed examination of how students transition from theoretical understanding to practical application, the research underscores the value of OSS projects as a pedagogical tool in software engineering education. Furthermore, our analysis seeks to inform educators and curriculum designers about the potential long-term benefits of incorporating OSS projects into software engineering courses, thereby shaping future educational strategies in this field.

2 RELATED WORK

The integration of GitHub and Open Source Software (OSS) in software engineering education has been a focal point in recent academic discourse. The research of Courtney and Vanessa [15] is pivotal in this regard, demonstrating that the use of GitHub in educational settings not only enhances students' collaborative and project management skills but also prepares them for professional challenges. Their findings suggest that students who actively engage with GitHub in their coursework feel more connected to their field and achieve greater academic success.

Building upon this, Feliciano et al. [12] explored the specific benefits of GitHub's collaborative features. They found that these features facilitate peer feedback and foster the development of soft skills, contributing to a deeper understanding of course content. This study also suggests a positive correlation between active GitHub engagement and academic success, particularly through enhanced critical analysis and peer reviews.

Fang et al. [11] provided a more nuanced view, acknowledging the stress and challenges associated with OSS contributions. Despite these challenges, the majority of students reported a positive overall experience, citing valuable insights into real-world software practices and the application of classroom-learned skills in practical settings. This balance between qualitative and quantitative data underscores the complexity of GitHub's impact on student learning.

Pinto et al. [19] and Spinellis [22] contributed significantly to this area by emphasizing the importance of selecting the right OSS projects and tasks to align academic objectives with practical software development practices. They argue that this alignment not only enhances the flow of the course but also significantly improves students' learning experiences.

The insights of Pinto et al. [20] are especially valuable, revealing through interviews with software engineering professors the tangible benefits of integrating OSS projects into courses. These benefits include enhanced technical skills, development of collaboration skills, and improvement in students' resumes. This finding is echoed by Trishala et al. [2], who observed that engagement with GitHub and Stack Overflow platforms enhances both soft and technical skills, crucial for collaborative environments.

However, despite the recognized benefits, certain studies like those by Brannock and Napie [3] and He et al. [14] have only provided anecdotal or limited evidence of the impacts of OSS in education. These studies, focusing on pre-class and post-class surveys, offer insights into students' interest levels and perceptions but lack a comprehensive, quantitative analysis.

In contrast, Andrews and Lutfiyya [1], Carrington and Kim [5], and Buchta et al. [4] employed more structured approaches to gauge students' experiences with OSS projects. Their studies, involving surveys and questionnaires, indicated a high level of student satisfaction and perceived real-world software experience, highlighting the practical benefits of OSS in education.

The work of Schneider et al. [21] and Marmorstein [17] further supports the positive impact of OSS contributions in software engineering courses. Their studies revealed improvements in students' IT skills, project enjoyment, and perceptions of employment readiness.

While these studies collectively underline the benefits of OSS in software engineering education, they predominantly rely on qualitative data, anecdotal evidence, and short-term assessments. Our research seeks to fill this gap by providing a quantitative and longitudinal analysis of students' GitHub contributions, offering a more detailed and long-term perspective on the impact of OSS project integration in software engineering courses. This approach not only contributes to a deeper understanding of the immediate benefits but also explores the sustained influence of such educational strategies over time.

3 BACKGROUND

3.1 Class Structure

Our study was conducted within the framework of a software engineering course, uniquely structured to blend object-oriented design and development with a hands-on approach using Ruby on Rails. This course, consistently taught by the same instructor over 15 years, immerses students in the Ruby programming language and the Ruby on Rails framework. Its curriculum is meticulously divided into two segments: The initial half introduces fundamental concepts including writing use cases, refactoring, Test-Driven Design (TDD), Behavior-Driven Design (BDD), Agile methodologies, and Scrum. The latter half is dedicated to advanced topics like SOLID principles [18] and the 23 GoF design patterns [13].

The course assessment includes a variety of graded components: two midterms, a final exam, four programming projects, two design documents, and three peer-assessed assignments. The project sequence is carefully designed to deepen the students' coding expertise and collaborative skills progressively. The first project, centered around coding interview-style questions, serves as an introductory exercise in Ruby. The subsequent project challenges students to form small teams and create a basic full-stack web application using Ruby on Rails, laying the groundwork for collaborative development. The highlight of the course is the third and fourth projects, spanning over half the semester, where students engage directly with real-world Open Source Software projects on GitHub. This immersive experience is not just about coding; it's about contributing to a live OSS project that has a significant footprint in the real world,

with over a thousand forks and more than five hundred contributors. These small projects, built around an NSF-funded OSS project led by the instructor, offer students an unparalleled opportunity to apply their skills in a dynamic, real-world context. Teams are tasked with a range of responsibilities, from refactoring and writing tests to fixing bugs and adding new features to the existing codebase. Teams are required to submit a pull request to the original repository as the project submission. Additionally, this course integrates a peer review component, wherein alongside their project work, students critically evaluate the design documents and code of two other teams. This not only fosters a culture of collaborative learning and feedback but also sharpens their analytical skills, crucial for software development. Through this innovative approach, the course bridges the gap between theoretical knowledge and practical application, highlighting the profound impact of integrating OSS projects into software engineering education. It provides students with a robust platform to experience real-world software development, preparing them for the challenges and dynamics of the industry.

3.2 Data Collection

Our data collection spanned from Fall 2011 to Spring 2023, encompassing 24 semesters of project submissions. A unique aspect of these projects was their requirement to be submitted as pull requests on the respective Open Source Software (OSS) projects. This approach enabled us to accurately identify the GitHub users involved in each project. We meticulously matched the participants' GitHub accounts with the students listed on the class roster, cross-referencing GitHub profiles for verification. However, it is important to note that the GitHub accounts of some students were not retrievable. This was due to various reasons, such as the absence of individual commits (owing to practices like pair programming where only one partner committed the code, or instances of non-participation). Over the past 12 years, we successfully retrieved the GitHub accounts of 1,444 students.

To systematically gather and analyze the GitHub contribution data, we developed specialized scripts using the GitHub API. These scripts were designed to collect publicly available user GitHub contribution statistics, recorded from the inception of the account until December 2023. The comprehensive metrics we gathered are outlined in Table 1. We classified the repositories into two distinct categories for a nuanced analysis: Type A repositories, which are either originally created or forked by the user, and Type B repositories, where the user contributes as a collaborator on a project initiated by another user. Moreover, to refine our analysis, we focused on the 10 most popular languages on GitHub. These languages are Python, JavaScript, Java, TypeScript, Go, C++, Ruby, PHP, C#, and C. We calculated the size of code each user contributed in these languages. This approach was adopted to mitigate the potential skewing of data by languages like HTML, which, although voluminous in size, might not accurately reflect a user's coding contribution due to its nature as a markup rather than a general-purpose programming language. Our data collection approach has been proven by previous studies [6–10] to be valid.

Our dataset was further processed and segmented based on five distinct timeframes: six months, one year, two years, three years, and total length, both before and after the class. This segmentation allowed for a more granular and temporal analysis of the GitHub contributions in relation to the software engineering class. We will explain later.

For each contribution type listed in Table 1, we tracked the creation time using the GitHub API. Concurrently, we consulted the school's academic calendars from 2011 to 2023 to pinpoint the start and end dates of each semester. For every student in our study, we aggregated the contributions, as defined in Table 1, both before and after the semester dates. This process was replicated across five different time spans: six months, one year, two years, three years, and total number of days both before the class start and after the class end.

Table 1: GitHub Metrics Collected from GitHub

Metric	Explanation
Private	The number of private contributions made by this user.
Commits	The number of commit contributions made by this user.
Issues	The number of issue contributions made by this user.
PRs	The number of pull requests created by this user.
PR Reviews	The number of pull request reviews created by this user.
Repos	The total number of type A, and type B repositories.
Comments	The number of comments made by this user in commits, issues, gists, and pull request discussions.
A lang cnts	Number of different languages used in Type A repositories.
B lang cnts	Number of different languages used in Type B repositories
A code size	The size of code written in bytes in GitHub popular languages in user's type A repositories.
B code size	The size of code written in bytes in GitHub popular languages in user's type B repositories.
A Ruby size	The total Ruby code size in type A repositories.
B Ruby size	The total Ruby code size in type B repositories.

To illustrate our data collection process, let's consider a real example with 'Student A'. This student created their GitHub account on 2016-06-11 and enrolled in our class from 2017-08-16 to 2017-12-14 (Fall 2017 semester). We tracked all of Student A's contributions from the account creation date up to the class start date, a period of 430 days. Within this timeframe, we specifically extracted contributions made during the 180 days preceding the class (from 2017-02-17 to 2017-08-16) to compile the 'six-month pre-class' dataset. Similarly, we isolated contributions made in the one-year period before the class start date (from 2016-08-16 to 2017-08-16) for the 'one-year pre-class' dataset. Given that Student A's GitHub account was active for only 430 days before the class, it was not feasible to calculate contributions for two years and three years prior to the class for this student. Additionally, we calculated the average frequency of each type of contribution made by students both before and after the class. For example, we divided the total number of commits before the class by 430 (the number of days before the class) to determine the average daily commit rate.

Similarly, for post-class, we logged all of Student A's contributions from the end of the class (2017-12-14) until 2023-12-31, totaling 2,208 days. We then applied the same filtering process for post-class contributions, creating records for the six-month, one-year, two-year, and three-year post-class datasets, as well as for the average post-class contribution rates.

For each student, we only recorded data in specific time frame datasets when the time span was valid for both before and after the

class. In Student A's case, their data is included in the six-month and one-year datasets for both before and after the class, but not in the two-year and three-year datasets, as these spans were not applicable before the class.

Our final datasets varied in size: 765 students in the six-month dataset, 573 in the one-year, 344 in the two years, 165 in the three years, and 921 in the average contribution dataset. Though we collected 1,444 GitHub accounts over 12 years, only 921 were used for the average dataset. This is because 523 students created their GitHub accounts only after joining the class. Our data shows a significant increase in students having GitHub accounts before class in recent years – over 90%. This number is around 50% before 2016. This growth is likely due to GitHub's growing popularity. However, since these 523 students had no pre-class records, including them would skew the comparison of contribution changes before and after the class. It's also noteworthy that our institution provides enterprise GitHub accounts for internal collaboration, which are revoked after graduation and cannot be used for contributions to publicly hosted repos on GitHub.com. Thus, the observed post-class increase in contributions is attributed to personal GitHub accounts, not influenced by other classes within the program.

4 METHODOLOGY

This section delineates the methodology employed to conduct our analysis, tailored to address the specific research questions posed in the study.

4.1 Paired Samples *t*-Test

To address RQ1, we utilized the paired samples t-test, a statistical tool ideal for comparing the means of two related measurements. This test is particularly suitable when the measurements are taken from the same subject at different times, such as a pre-test and post-test score with an intervention occurring in between. The primary aim of this test is to ascertain if the mean difference between these paired observations significantly deviates from zero [16].

In our context, we applied this test to evaluate the differences in specific types of GitHub contributions by the same student, before and after enrolling in our class. This involved conducting paired t-tests on each contribution metric, as outlined in Table 1. Our null hypothesis posits that there is no increase in a student's contributions post-class, suggesting that the contribution level before the class is either greater than or equal to that after the class. The alternative hypothesis, conversely, proposes that the contribution level after the class is higher. We established an alpha level ($\alpha = 0.05$) of 0.05 for our significance threshold. A p-value less than 0.05 would lead us to reject the null hypothesis in favor of the alternative, indicating a significant increase in contributions post-class.

4.2 Spearman's Rank Correlation

For RQ2, we employed Spearman's Rank Correlation, an effective non-parametric measure to assess the strength and direction of association between two ranked variables. Spearman's rank correlation coefficient ρ is particularly suited for analyzing monotonic relationships, which may not necessarily be linear, using ranked data. A Spearman ρ value of +1 indicates a perfect positive correlation, -1 signifies a perfect negative correlation, and a value near 0 implies no significant correlation.

In our analysis, we ranked each student based on their post-class GitHub metrics as well as their academic performance across various criteria (exams, projects, review writing, and documentation writing). A Spearman rank correlation was then computed to determine the relationship between these two sets of ranks. Consistent with the paired t-test, we set our significance level at $\alpha = 0.05$. A p-value below this threshold would indicate a significant correlation between students' GitHub contributions and their academic performance.

5 RESULTS

5.1 Contributions Before and After Class

Our study employed paired t-tests to investigate the variations in several GitHub contribution metrics before and after students completed the class. The results, detailed in table 2, provide a comparative analysis of the mean contributions over various time frames: 6 months, 1 year, 2 years, 3 years, and a cumulative average period.

The table employs color coding to highlight statistically significant changes in contribution metrics. Cells marked in red indicate a statistically significant increase in contributions post-class compared to the pre-class period, while blue cells denote a statistically significant decrease. For example, in the 'Commits' category, a notable increase in the average number of commits post-class was observed. During the 6-month post-class period, the mean number of commits increased from 17.83 (pre-class) to 26.29 (post-class), a trend underscored by the red cell color, indicating statistical significance. This pattern of increase continues in the 1-year and 2-year post-class periods, with mean numbers rising to 54.84 and 71.65, respectively. However, in the 3-year post-class period, the increase to 68.58 from 58.14 pre-class does not reach statistical significance, as reflected by the absence of color coding in these cells. A similar interpretation can be generated for other rows in the table.

The analysis extends across various metrics, including issues, pull requests, PR reviews, repositories created, comments made, number of languages used in both type A and type B repositories, and Ruby code size in these repositories. In all these categories, except for the size of type B repositories, there is a statistically significant growth in post-class contributions compared to their pre-class counterparts within the same time span.

However, it's worth noting that the total average column does not exhibit substantial differences. This is attributed to the longer post-class time span, which tends to average out many of the contributions, resulting in values closer to zero. This observation suggests that using annual averages might be a more effective measure than daily averages for capturing the true impact of the class on GitHub contributions.

5.2 Contributions and Academic Performance

In our study, we investigated the correlations between students' GitHub contributions after completing their software engineering class and their academic performance, focusing on project and exam averages as well as review scores. We excluded documentation averages from this analysis, as our findings showed no significant correlation between documentation averages and GitHub contributions. The resulting Spearman correlation coefficients, presented in table 3, are color-coded to highlight statistically significant correlations.

Table 2: Paired t-test Results and Mean Contribution of Before and After Class Contribution

	6 Month		1 Year		2 Year		3 Year		Total Avg	
	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post
Private	8.07	12.03	15.03	24.89	19.56	56.41	29.67	77.67	0.02	0.06
Commits	17.83	26.29	26.54	54.84	42.61	71.65	58.14	68.58	0.06	0.07
Issues	1.31	0.96	1.81	3.19	2.42	4.88	3.68	5.45	0	0
PRs	1.63	1.43	1.68	4.41	2.4	9.76	2.17	12.06	0.01	0.01
PR Reviews	0.59	0.33	0.33	1.08	0.16	2.96	0.14	6.18	0	0.01
Repos	1.46	2.46	2.48	4.80	4.43	6.95	6.42	7.57	0.01	0.01
Comments	1.75	1.43	2.88	4.97	4.49	11.53	6.6	9.91	0.01	0.01
A lang cnts	2.28	4.14	3.50	7.33	5.69	10.41	7.54	10.69	0.01	0.01
B lang cnts	1.12	0.87	1.23	2.11	1.31	2.82	1.77	2.94	0	0
A size	1.38E+06	5.06E+06	2.59E+06	8.26E+06	7.67E+06	1.57E+07	1.32E+07	1.45E+07	9.21E+03	1.29E+04
B size	4.00E+05	2.39E+05	4.92E+05	6.15E+05	4.21E+06	1.02E+06	8.62E+06	7.21E+05	3.25E+03	7.95E+02
A Ruby size	1.83E+04	1.30E+05	2.28E+04	1.98E+05	1.05E+05	3.38E+05	9.95E+04	3.03E+05	5.01E+01	2.66E+02
B Ruby size	2.21E+03	4.49E+04	1.37E+04	1.22E+05	1.32E+04	2.20E+05	1.22E+04	2.06E+05	1.19E+01	1.02E+02

The color coding of the cells indicates the statistical significance of the difference between pre-class and post-class contributions for each metric. Blue cells indicate metrics where the mean contribution before the class is statistically significantly lower than after the class. Red cells represent metrics where the mean contribution after the class is statistically significantly greater than before the class.

Table 3: Correlation Between GitHub Contributions after Class and Performance in Class

	Project Avg				Exam Avg				Review Avg			
	6 Month	1 Year	2 Year	Avg	6 Month	1 Year	2 Year	Avg	6 Month	1 Year	2 Year	Avg
Private	0.04	0.08	0.07	0.07	0.17	0.16	0.16	0.17	0.05	0.03	-0.05	0.01
Commits	0.10	0.09	0.16	0.06	0.10	0.08	0.11	0.07	0.04	0.06	0.02	0.01
Issues	0.07	0.01	0.07	0.06	0.07	0.11	0.13	0.10	0.03	0.07	0.05	0.00
PRs	0.10	0.03	0.09	0.07	0.08	0.08	0.13	0.09	0.06	0.08	0.03	-0.01
PR Reviews	0.05	0.01	0.03	0.06	0.11	0.12	0.12	0.15	0.07	0.03	-0.03	-0.04
Repos	0.01	0.01	0.04	-0.01	0.02	-0.02	0.00	0.00	0.03	0.08	0.04	0.00
Comments	0.10	0.06	0.14	0.11	0.08	0.15	0.18	0.13	-0.03	0.01	-0.01	-0.05
A langs cnts	-0.01	-0.02	-0.01	-0.04	0.03	0.00	0.03	0.03	0.02	0.03	0.05	-0.02
B langs cnts	0.08	0.06	0.05	0.02	0.05	0.05	0.08	0.08	0.00	0.09	0.08	0.05
A size	-0.03	-0.05	-0.03	-0.04	0.01	-0.03	0.02	-0.01	0.02	0.02	0.01	-0.04
B size	0.07	0.07	0.07	0.02	0.05	0.04	0.08	0.07	0.11	0.11	0.08	0.06
A Ruby size	0.03	0.01	0.01	0.00	-0.03	0.01	0.00	-0.03	0.03	0.02	0.02	0.02
B Ruby size	0.02	0.01	0.00	-0.02	0.03	0.01	0.03	0.01	0.07	0.03	0.03	0.04

This table presents the Spearman correlation coefficients between various GitHub contribution metrics after completing the class and different academic performance measures, including project averages, exam averages, and review averages. Green Cells Indicate statistically significant correlations. A green cell shows that there is a significant relationship between the post-class GitHub contribution metric and the corresponding academic performance measure.

The table is organized to display correlations over several post-class time frames: 6 months, 1 year, 2 years, and an aggregated overall average. 3 years was excluded also because no significant correlation was observed. Each row represents a different GitHub contribution metric, such as 'Private', 'Commits', and 'Issues'. For instance, the 'Private' category shows a statistically significant positive correlation with exam averages in the 6-month post-class period, indicated by a correlation coefficient of 0.17 and a green-colored cell. This suggests that higher private contributions on GitHub correlate with better exam performance soon after the class.

In the 'Commits' row, significant positive correlations are evident in the 6-month, 1-year, and 2-year periods with project averages, and in the 6-month, 1-year, and 2-year periods with exam averages. This trend implies that students who commit more frequently on GitHub tend to score higher in both projects and exams during these specific periods.

It's important to note that not all correlations reached statistical significance. Cells without color coding represent coefficients that are not statistically significant, indicating either a weak or no discernible relationship between those GitHub contributions and the academic metrics. This aspect of our analysis sheds light on the potential influence of active involvement in software development on GitHub on students' academic achievements in software engineering courses. The statistically significant correlations observed in several key metrics emphasize the relevance of practical software development skills to academic success.

6 DISCUSSION

6.1 RQ1: Changes in GitHub Contributions

Our analysis revealed significant changes in students' GitHub contributions post-class, with paired t-tests indicating a statistically significant increase in various metrics including commits, private contributions, comments, the number of repositories created, and the size of Ruby code contributions. Notably, these increases were most pronounced within shorter time frames post-class (6 months to 2 years). While the absence of a control group precludes definitive conclusions about causality, these findings suggest that participation in the Software Engineering class, with its focus on OSS projects, may have had a considerable influence on students' practical engagement in software development.

One plausible interpretation of the increased GitHub activity is that the class effectively connected theoretical knowledge with practical application. The course's emphasis on OSS likely motivated students to actively contribute to real-world projects, thereby enhancing their software development understanding and skills. This is evidenced by the significant rises in private contributions and commits, suggesting that students were not only more engaged in coding activities but also more involved in private project work, possibly reflecting a growth in their confidence and abilities in software development.

The observed significant increase in the size of Ruby code contributions after the class is particularly noteworthy. It indicates that teaching a new programming language can motivate students to contribute more in that language. This suggests that the specific skills and languages taught in class have a lasting impact on students' contributions, a finding that underscores the importance of curriculum design in software engineering education.

However, our analysis also indicates that the impact of the class on OSS contributions may be more short-lived than permanent. While there was a marked increase in contributions shortly after the class, this effect tended to diminish over time. This is evidenced by the lack of significant differences in many contributions over a 3-year span and the relatively unchanged average contributions before and after the class. It appears that while students are initially motivated to contribute to OSS projects following the class, this enthusiasm and engagement may wane over longer periods. This observation suggests that while software engineering education can catalyze immediate increases in practical engagement, sustaining this momentum might require ongoing encouragement or engagement strategies beyond the classroom. In contrast to the general trend of diminishing engagement over time, the sustained increase in contributions in the specific language taught (Ruby) suggests a long-term adoption of skills acquired in class. This enduring impact on language proficiency highlights the lasting value of teaching specific programming languages and technologies in software engineering courses.

6.2 RQ2: Correlation with Performance

Our study utilized Spearman's rank correlation analysis to delve into the relationship between students' academic performance and their post-class contributions on GitHub. Notably, we found significant correlations between certain GitHub metrics and students' performance, especially in project and exam scores. For instance, an increased frequency of commits and private contributions was positively correlated with higher performance in both exams and projects.

These findings highlight a reciprocal relationship between practical engagement in OSS projects and academic success in software engineering. This suggests that students who excel academically are more likely to be effective and active contributors on GitHub. This could be attributed to a deeper comprehension of course material or a higher level of motivation. Conversely, active engagement in GitHub projects appears to bolster the understanding and application of classroom concepts, potentially enhancing students' academic performance.

It is important to note that while these correlations exist, they are not particularly strong, indicating a nuanced relationship between academic performance and OSS contributions. Our analysis revealed that exam performance most significantly translates to post-class OSS contributions. On the other hand, areas like documentation writing showed no significant correlation, and review writing displayed only a weak, albeit significant, relationship with post-class contributions to pull request reviews. This variation suggests that the direct translation of academic performance to practical contributions post-class is not straightforward. There are likely other factors at play that influence a student's ability to apply theoretical knowledge in practical scenarios. This observation raises intriguing questions for educators in software engineering: while imparting theoretical knowledge is crucial, understanding the elements that facilitate the transition of this knowledge into practical, real-world contributions is equally vital.

7 THREATS TO VALIDITY

Several factors pose potential threats to the validity of our study. Firstly, our results are derived from data collected at a single institution and from a specific software engineering course. This raises concerns about the generalizability of the findings to other educational settings or disciplines. Secondly, the lack of a control group in our study design limits our ability to conclusively establish causality between participation in the software engineering class and the observed changes in GitHub contributions. Lastly, our study relies on GitHub metrics to gauge practical engagement in software development. While these metrics provide valuable insights, they may not fully encompass the complex nuances of students' software development activities.

8 CONCLUSIONS AND FUTURE WORK

Our study explored the impact of a Software Engineering class on students' GitHub contributions and the correlation between these contributions and academic performance. The findings reveal a significant increase in GitHub activities post-class, suggesting that the course may promote practical software development skills. Additionally, we observed correlations between academic performance and specific GitHub metrics, although these were not uniformly strong across all measures. These results underscore the value of integrating practical OSS projects in software engineering education, highlighting the potential benefits for both theoretical understanding and practical application. Future research should aim to address the limitations of this study, potentially exploring diverse educational settings and longer-term impacts, to further understand the dynamics between academic instruction and practical software development skills.

REFERENCES

[1] James H Andrews and Hanan L Lutfiyya. 2000. Experiences with a software maintenance project course. *IEEE Transactions on Education* 43, 4 (2000), 383–388.

[2] Trishala Bhasin, Adam Murray, and Margaret-Anne Storey. 2021. Student Experiences with GitHub and Stack Overflow: An Exploratory Study. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 81–90. https://doi.org/10.1109/CHASE52884.2021.00017

[3] Evelyn Brannock and Nannette Napier. 2012. Real-world testing: using foss for software development courses. In *Proceedings of the 13th annual conference on Information technology education*. 87–88.

[4] Joseph Buchta, Maksym Petrenko, Denys Poshyvanyk, and Vaclav Rajlich. 2006. Teaching Evolution of Open-Source Projects in Software Engineering Courses. In *2006 22nd IEEE International Conference on Software Maintenance*. 136–144. https://doi.org/10.1109/ICSM.2006.66

[5] David Carrington and S-K Kim. 2003. Teaching software design with open source software. In *33rd Annual Frontiers in Education, 2003. FIE 2003.*, Vol. 3. IEEE, S1C–9.

[6] Jialin Cui, Ruochi Li, Kaida Lou, Chengyuan Liu, Yunkai Xiao, Qinjin Jia, Edward Gehringer, and Runqiu Zhang. 2022. Can Pre-class GitHub Contributions Predict Success by Student Teams?. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 40–49.

[7] Jialin Cui, Runqiu Zhang, Ruochi Li, Yang Song, Fangtong Zhou, and Edward Gehringer. 2023. Correlating Students' Class Performance Based on GitHub Metrics: A Statistical Study. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 526–532.

[8] Jialin Cui, Runqiu Zhang, Ruochi Li, Fangtong Zhou, Yang Song, and Edward Gehringer. 2024. How Pre-class Programming Experience Influences Students' Contribution to Their Team Project: A Statistical Study. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (, Portland, OR, USA,) *(SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 255–261. https://doi.org/10.1145/3626252.3630870

[9] Jialin Cui, Fangtong Zhou, Chengyuan Liu, Qinjin Jia, Song Yang, and Edward Gehringer. 2024. Utilizing the Constrained K-Means Algorithm and Pre-Class GitHub Contribution Statistics for Forming Student Teams. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1*.

[10] Jialin Cui, Fangtong Zhou, Runqiu Zhang, Ruochi Li, Chengyuan Liu, and Ed Gehringer. 2023. Predicting Students' Software Engineering Class Performance with Machine Learning and Pre-Class GitHub Metrics. In *2023 IEEE Frontiers in Education Conference (FIE)*. 1–9. https://doi.org/10.1109/FIE58773.2023.10343357

[11] Zihan Fang, Madeline Endres, Thomas Zimmermann, Denae Ford, Westley Weimer, Kevin Leach, and Yu Huang. 2023. A Four-Year Study of Student Contributions to OSS vs. OSS4SG with a Lightweight Intervention. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (<conf-loc>, <city>San Francisco</city>, <state>CA</state>, <country>USA</country>, </conf-loc>) *(ESEC/FSE 2023)*. Association for Computing Machinery, New York, NY, USA, 3–15. https://doi.org/10.1145/3611643.3616250

[12] Joseph Feliciano, Margaret-Anne Storey, and Alexey Zagalsky. 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)*. 422–431.

[13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., USA.

[14] Hao He, Minghui Zhou, Qingye Wang, and Jingyue Li. 2023. Open Source Software Onboarding as a University Course: An Experience Report. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. 324–336. https://doi.org/10.1109/ICSE-SEET58685.2023.00037

[15] Courtney Hsing and Vanessa Gennarelli. 2019. Using GitHub in the Classroom Predicts Student Learning Outcomes and Classroom Experiences: Findings from a Survey of Students and Teachers. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 672–678. https://doi.org/10.1145/3287324.3287460

[16] Henry Hsu and Peter A Lachenbruch. 2014. Paired t test. *Wiley StatsRef: statistics reference online* (2014).

[17] Robert Marmorstein. 2011. Open source contribution as an effective software engineering class project. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. 268–272.

[18] Robert Cecil Martin. 2003. *Agile software development: principles, patterns, and practices*. Prentice Hall PTR.

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

[20] Gustavo Henrique Lima Pinto, Fernando Figueira Filho, Igor Steinmacher, and Marco Aurelio Gerosa. 2017. Training Software Engineers Using Open-Source Software: The Professors' Perspective. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. 117–121. https://doi.org/10.1109/CSEET.2017.27

[21] Jean-Guy Schneider, Peter W Eklund, Kevin Lee, Feifei Chen, Andrew Cain, and Mohamed Abdelrazek. 2020. Adopting industry agile practices in large-scale capstone education. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 119–129.

[22] Diomidis Spinellis. 2021. Why computing students should contribute to open source software projects. *Commun. ACM* 64, 7 (2021), 36–38.