



Collaboration in Computer Science Education
Investigating the Correlation Between Prior Programming Experience and Task Division

Jocelyn Woods¹

Supervisor(s): Dr.ir. E. Fenia Aivologlou¹, Merel Steenbergen¹ °

¹EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Jocelyn Woods

Final project course: CSE3000 Research Project

Thesis committee: Dr.ir E. Fenia Aivologlou, Merel Steenbergen ,Dr.ir. Mitchell Olsthoorn

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Abstract

In this project, the topic of collaboration in computer science education was researched, specifically regarding the correlation between prior programming experience and task distribution in student software development projects. Students self-reported prior programming experience, including their years of study, years of coding experience, their experience rank from 1-10, and their experience compared to their classmates via a survey. A thematic analysis of transcriptions of weekly sprint meetings was then conducted to capture students' reasoning behind task allocation. Additionally, a quantitative analysis correlated self-reported experience levels with Git metrics including task types (taken from tags), task difficulty (weight), and task reassignment frequency (assignees). While the quantitative analyses found no significant correlation between prior experience and task distribution, the thematic analysis found that students often cited prior experience, in addition to personal interest and time constraints as common reasons for task distribution decisions. These findings suggest that task allocation is affected by multiple factors. While prior experience may have an influence, other factors are likely more impactful.

1 Introduction

Collaborative software development is a crucial part of both computer science education and students' eventual careers. Most professional projects use scrum or agile methodologies in order to combine efforts effectively and produce the best resulting work. It is important to understand how these team members collaborate as a means to identify possible areas of improvement regarding collaboration and to better prepare students for their future software engineering roles. While many studies address collaboration and role division, few examine the specific role of prior programming experience in shaping task distribution. Understanding this role may offer insights into related factors such as gender, motivation, or tendency to under-contribute.

It is possible that one of the factors leading to the longstanding problem of under-contributing teammates, also known as social loafers, in student software development projects [2] could be an unequal distribution of prior knowledge. Students do not all arrive at university with the same sets of prior skills and experience, and the problem is not completely solved as they proceed through their education [12]. Differences in learning speeds, interests, or outside opportunities could all contribute to a widening of this gap.

This paper investigates whether prior programming experience affects division of work in the second year software project. Metrics from various previous studies have been combined to create a well-rounded measurement of task division. Additionally, this research combines quantitative and qualitative methods, attempting to identify reasoning during student task allocation.

This paper aims to answer three questions exploring the correlation between prior programming experience and task distribution in student software engineering projects.

- **RQ1:** What is the correlation between prior programming experience, and what types of tasks/roles team members take on?
- **RQ2:** How does prior experience correlate with students' likelihood to complete more difficult or time-consuming tasks?
- **RQ3:** What difference does prior programming experience make between tasks assigned to each team member and tasks done by each team member?

In order to answer the aforementioned research questions, both a thematic and quantitative analysis will be conducted. For the thematic analysis, transcriptions of sprint meetings from three volunteer groups, each with five students, from the TU Delft software project (CSE2000) have been made. The transcripts were analyzed to determine which students took on a more managerial or technical role and to identify reasons students commonly gave for task allocations. Next, an analysis was run on data collected via a survey completed by the students mentioned above with the addition of one more five-student group regarding their prior programming experience and data collected from the student repositories regarding task allocation. For the purpose of this study, an issue is considered a task and all students assigned to an issue are considered contributors and share credit equally.

Overall, the thematic analysis part of this study found that prior experience is one of the factors that dictates task distribution; however, it is not the only one. Towards the later part of the project (weeks five and six), students tend to include more considerations during task allocation. The quantitative section of this study showed mixed results. It found that there is limited correlation between the metrics chosen to represent previous experience (years of study, years of programming experience, self-rating, and rating relative to classmates) and task division.

The remainder of this paper will address the aforementioned research questions. First, in section 2, the methodology of the study will be explained, including all data sources. Next, specific contributions will be discussed in section 3, and the results of the study will be explained in section 4. In section 5, responsible research considerations will be examined. Then, in section 6, broader discussions and limitations will be addressed. Finally, the conclusion and future work recommendations will appear in section 7.

2 Related Work

Task division in student software engineering teams has been a recurring topic in computer science education research, especially relating to contribution imbalance and role differentiation [6] [9] [11]. Researchers have proposed various metrics to measure individual contributions to team programming projects, including lines of code, commit frequency, quantity of merges, and difficulty of tasks completed [3][6].

Kopek-Harding et al. found that comparing percentages of lines of code from each student often reveals stark unequal-

ities in contributions. They further discovered that nearly no student groups consist entirely of adequately contributing members, defined as those who complete between 90 and 110 percent of their expected workload [6]. However, this study fails to capture students' non-technical contributions, which will be addressed using the metric-based approach in this research. This approach was influenced by the work of Ramin et al. in another study that researched the roles that members of software engineering groups take on, and argued that not all contributions are technical, and some members gravitate toward managerial roles [9].

Prior research on unequal team participation has often focused on strategies to limit social loafing: the tendency of some students to contribute less in group settings. Fronza and Wang found that structured expectation setting and accountability provide one possible prevention technique against social loafing. They aimed to raise awareness of the issue among group mates to measure if it altered student behavior. However, they do not cite prior experience as a causal factor, and their focus lies mainly on prevention on a project by project basis [2]. Wolfe and Powell have also contributed to this area with their studies, comparing student and professional responses to given scenarios. This was done to gather information on key team working strategies used by professionals whose effects they suggested be later tested on students [12]. Aivaloglou and van der Meulen thematically analyzed interviews with students to better understand their personal motivations when assuming a task [11]. These researchers found that, when explaining their task distributions, students often preferred to work on familiar issues. This was concluded as they often cited previous experience in a related area as a driving factor.

Many papers cite gender differences [4], personal motivations [10], or group composition [6] as influences for this discrepancy in contributed work. Strijbos and De Laat argue that a student's personal motivations, for example, being team or individually motivated or aiming for a higher or passing grade, affect a student's participation and task selection [10]. Others have researched the role of gender in task division [4]. They found that gender affects the types of tasks a student is likely to assume. Specifically, they found that women often take on less technical roles than their male counterparts. However, differences in the prior programming experience of these students were not considered as a possible factor in this study. Kopec-Harding et al. further observed that group composition influences work imbalance, noting that students often shift roles throughout multiple projects. This suggests that the context of a specific project may play a larger role than a student's individual tenancies [6].

While prior research has investigated multiple aspects of task and role allocation in computer science education, fewer have specifically studied their correlation with prior programming experience, particularly in student software development projects. This research will attempt to further investigate this gap through both thematic and quantitative analysis.

3 Methodology

To better understand the correlation between a student's prior programming experience and the distribution of tasks in student software development projects, three self created student teams from the Delft University of Technology second year software project course were observed. These teams consisted of students who volunteered to participate in this research. In order to gather relevant information, the students were first asked to complete a survey. Next, between one and three sprint meetings per team which took place in the first six weeks of the project were transcribed and analyzed. Communications outside these one hour weekly meetings such as texts, client meetings, or general meetups were not considered in this study. and finally, access to their repositories was granted. Data from these three sources have been used to answer the previously mentioned research questions.

3.1 Data Collection

For the thematic analysis of sprint meetings from three of the four volunteer five-student teams were considered. One member of the research team attended each meeting and recorded the conversation from beginning to end without interrupting or directing the discussion in any way. Once this was done, transcriptions were made and shared with the other members of the group. For each meeting, the first step was to extract and code phrases relevant to the research question. After coding the entire text, the coded sentences were examined and grouped thematically to look for any common reasons behind task distribution. Due to time and resource constraints coding was not validated by a second researcher. This process was first conducted for sub-question one, and then for sub-question two. For the quantitative analysis, data was extracted from the students' GitLab. Each issue was individually examined for its weight, creator, assignees, re-assignees, and tags. If no weight was provided by the student, the issue descriptions were given to Gemma 3, a local large language model (LLM) created by Google, to evaluate. These evaluations were validated for consistency by manually comparing weights to similar issues in other projects where students had assigned a weight. Once all the data was collected, each data point was plotted on a scatterplot comparing prior experience with specific aspects of task division. Additionally, an analysis was performed to determine the strength of the correlations.

3.2 Survey Design

To assess prior programming experience, a survey was used to gather the students' self-reported experience. It was created by combining previously researched and validated surveys. One such survey, created by Feigenspan et al., was the result of research that compared the responses of over 100 students to find a way to define student programming experience [1]. This was the inspiration for the survey questions relevant to this research topic. Relevant questions have been incorporated into a survey designed by the research group responsible for this and related research. As the survey was also used for parallel research, additional questions for related research were included in the final resulting survey.

3.3 Thematic Analysis of Sprint Meetings

A thematic analysis was performed on the transcribed conversations. This research is based on the method used by van der Meulen and Aivaloglou to collect student reasoning behind task allocation decisions through interviews [11] was changed to fit the environment present in this research. Instead of the interview format, similar themes to ones identified in previous research were pulled from the sprint meeting transcripts. There was a particular focus on students' reasoning for taking on specific types of tasks, roles of students, and difficulty of tasks. Additionally, correlation between prior experience, and the prominence and type of role students take on in a meeting will be studied by combining insights from [10] and [9]. Themes included common motivations for task allocation (difficulty, interest, necessity, taking/being assigned a task) and conversations relating to assuming managerial or technical roles (taking a task, structuring the meeting, making a programming plan, asking or answering organizational questions).

3.4 Repository Analysis

Finally, students' repositories were analyzed. Information about issue distribution, difficulty of issues, and task reassignment were aggregated from all teams. The measurement of team members' contributions is inspired by a paper written by Kopeck-Harding et al. [6]. However, it was adapted and broadened for the purpose of this research to use the number and difficulty of issues in place of lines of code to measure contribution. A Pearson correlation analysis was first added to offer further insight into the correlation with prior programming experience. Later, a Spearman correlation analysis was additionally included to account for the possibility of a non-linear relationship. Difficulty of issues has been chosen as a metric based on the research of Parizi et al. regarding using git metrics, especially time spent on each issue, as an indicator of contributions in software development projects [3]. Students have assigned each issue a weight based on how much effort they believed the issue to require groups that did not assign issue weights have had the values approximated. This value was calculated by comparing issues to similar issues in other groups and was aided by the use of Gemma 3, a local large language model (LLM) developed by Google. Gemma 3 was given the issue descriptions and given the following instructions: "I will give you the issue description of a task completed by a student following the 10-week software project course at TU Delft. I need you to give me a difficulty ranking from 1-20 based on this information. Be sure to keep consistency in mind while creating these values". Then, once all issues were assigned a value, Gemma was asked to "Reevaluate the scores you have assigned to each issue with the knowledge you now have of all the issues in the project." If there are any weights you would like to change, please give me the issue number, old and new weight, and reasoning", to verify the accuracy of the given weights. A local LLM was used to maintain the privacy of students. These metrics were then used to assess the difficulty of a task. Furthermore, the research done by Mittal and Sureka [3][8] was adapted to compare original task assignments to final distributions. Since not all groups have completed their wiki and retrospec-

tives issue reassignments on Gitlab have instead been used to measure changes.

To answer the research questions mentioned above, a correlation analysis was performed on between each measure of prior experience and the area addressed by the question (managerial/technical role assumption, distribution of difficult/easy tasks, and task reassignment). Both the p-values and R values were considered as measures of correlation in this research. Based on the work of Komaroff, as there were fewer than 1000 data points, a p-value lower than .05 has been chosen to indicate statistical significance. Additionally a r value of less than .1 is considered uncorrelated, a value between .1 and .3 is mildly correlated, a value of .3 to .5 is considered moderate correlation, and a r value of over .5 is considered strong correlation. [5]

4 Results

For the purposes of this research, both thematic and quantitative analyses have been conducted. Results will be compared and used to further strengthen possible conclusions or contrast differing findings.

4.1 RQ1

A combination of quantitative and qualitative methods has been used to measure the correlation between prior programming experience and the type of tasks students assume in software development projects.

Quantitative Analysis

To measure the correlation between prior programming experience and the roles that students take on in software engineering projects, four measures of prior programming experience have been compared to the number of managerial, documentation, or technical tasks students have performed.

The strongest correlation for this measure of managerial contribution, issues created by or assigned by a student, was with the metric years of programming experience as shown below in figure 1. The resulting value, $r(18) = -.21$, does indicate a mild correlation. However, this is not statistically significant as $p = .378$ [5]. The correlations with other metrics are listed below, and additional graphs can be found in the appendix.

- Experience 1-10: $r(18) = -.01$, $p = .957$
- Experience compared to classmates: $r(18) = -.10$, $p = .690$
- Years of Study: $r(18) = -.16$, $p = .493$

The correlation between the number of documentation-related tasks and years of programming experience is higher, as seen in figure 2. This analysis has $r(18) = .35$, and $p = .125$. However, as $p > .05$. This correlation is not considered statistically significant [5]. The calculated r and p values of the other analyses can be seen below, and the corresponding graphs can be found in the appendix.

- Experience 1-10: $r(18) = .12$, $p = .604$
- Experience compared to classmates: $r(18) = .09$, $p = .694$

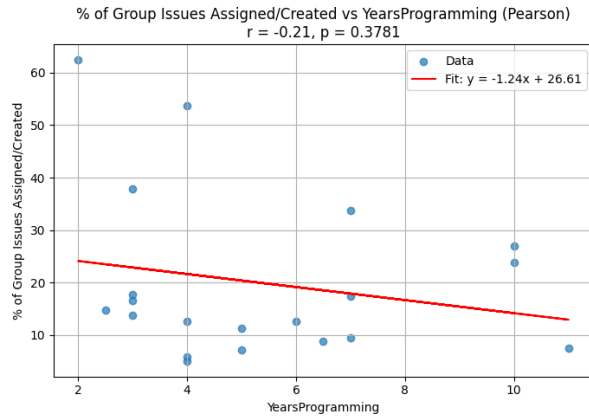


Figure 1: Percentage of issues created or assigned by each student compared to years of programming experience

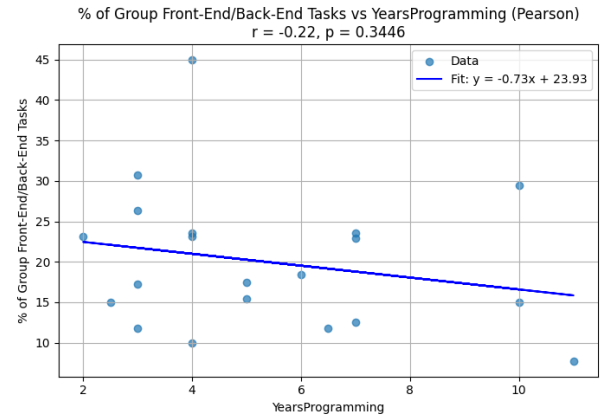


Figure 3: percentage of technical issues completed by each student compared to years of programming experience

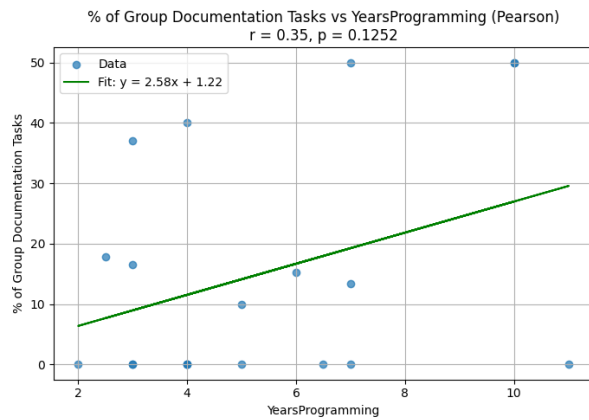


Figure 2: Percentage of documentation issues done by each student compared to years of programming experience

- Years of Study: $r(18) = .08$, $p = .737$

As with the above, Pearson analyses years of programming experience has the strongest correlation. It can be seen that $r(18) = -.22$, which indicates a weak correlation. However, as the p value is higher than .05, this result cannot be considered statistically significant [5]. The r and p values for the other experience metrics can be seen below, and additional graphs appear in the appendix.

- Experience compared to classmates: $r = -.12$, $p = .628$
- Years of Study: $r = .20$, $p = .402$
- Experience self (1-10): $r = -.14$, $p = .555$

While the initial data observations appeared to be linear, as more data was introduced, the consideration that the correlation may be non-linear was examined. In order to investigate this, a Spearman correlation analysis was included as seen in table 1. This found no additional statistically significant results.

Experience Metric	CreateAssign (r/p)	Documentation (r/p)	Technical (r/p)
Years Programming	-0.20 / 0.391	0.22 / 0.362	-0.22 / 0.358
Experience Self	0.15 / 0.524	0.11 / 0.649	-0.14 / 0.562
Experience Relative	-0.05 / 0.832	0.06 / 0.786	-0.11 / 0.631
Years Study	0.07 / 0.767	0.14 / 0.550	0.39 / 0.089

Table 1: Spearman correlation (r) and p -values between experience metrics and contribution types.

Thematic Analysis

Conversations regarding tasks in two of the three groups, a clear "manager" was found. This person took on the responsibility of structuring meetings, planning and allocating tasks, and guiding the workflow. Additionally, in the same two groups there was also a second member that often supported or contradicted the "manager". This means they would speak up if they thought the "manager" was incorrect or to bring up tasks they believed should also be addressed. They spoke slightly less often to organize the meeting, but were still active participants. The other three in both these groups were slightly more passive. They were willing to take on tasks suggested by others but tended to plan less.

The people who took on these two roles were not the students who rated themselves with the highest experience level in any of the two self-reported levels considered. Two of the four actually rated themselves on the lower end of programming experience. However, all had already completed two years of university before beginning the Software Project course.

In the remaining group, there was more active participation from all members. While the two members spoke slightly more in an organizational sense, all team members actively answered questions and participated in the planning of the next sprint.

In this team, the two students who took on a slightly more managerial role were again not the students with the highest self-reported experience level. However, since the difference was not very large and only one meeting was recorded, it is possible that this pattern did not continue throughout their project. In addition, this group consisted of the largest per-

centage of students who were only at the end of their second year of university, so that may have some influence.

4.2 RQ2

For the purposes of answering RQ2, a combination of quantitative and qualitative measures has been used. A thematic analysis of transcribed sprint meetings was done to find student reasoning behind task allocation. An analysis was done to find the correlation between the four prior experience metrics and the difficulty of the task done by each student.

Quantitative Analysis

For the quantitative analysis, the average weight normalized per group of all tasks was compared to the four previously mentioned metrics used to measure prior experience (years of study, years of programming, skill rating 1-10, skill relative to classmates).

There was the greatest correlation when average weight was compared to the number of years of programming experience a student has (See Figure 4). This analysis resulted in $r(18) = -.44$, which, according to Komaroff, indicates a medium correlation. However, as the sample size is small, it is recommended to have a p-value of less than or equal to .05, which is not the case in this analysis [5]. Therefore, the results of this research are not statistically significant. The resulting p and r values for the other three analyses can be seen below, and the corresponding graphs can be found in the appendix.

- Experience 1-10: $r(18) = -.43$, $p = .249$
- Experience compared to classmates: $r(18) = -.17$, $p = .571$
- Years of Study: $r(18) = -.13$, $p = .680$

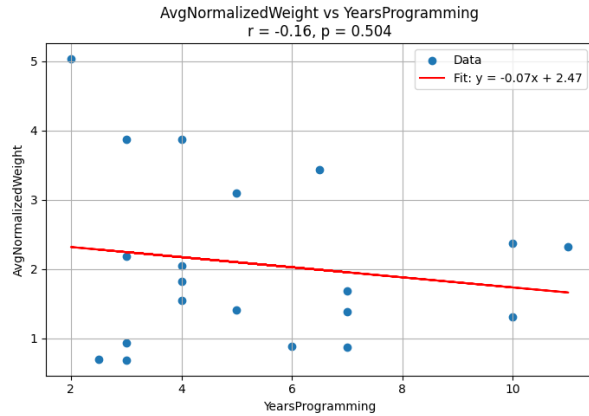


Figure 4: Average weight of issues completed compared to total number of years of programming experience

In the case of the analysis for RQ2, the initial data observations again appeared to be linear. As more data was collected, the possibility that the correlation may be non-linear was considered. A Spearman correlation analysis was included as seen in table 2 to account for this. However, this found no additional statistically significant results.

Experience Metric	AvgNormalizedWeight (r / p)
YearsStudy	-0.09 / 0.7105
ExperienceSelf	0.08 / 0.7312
ExperienceRelativeNum	0.05 / 0.8346
YearsProgramming	-0.04 / 0.8562

Table 2: Spearman Correlation with AvgNormalizedWeight.

Thematic Analysis

In all three teams, prior programming experience was not given as a reason a teammate should take on a more difficult task. One team would only even discuss the weight of an issue once it was assigned. This is shown when one student has already taken responsibility for a task by saying, “*And then you run the tests and then, if those tests pass, you push the whole thing into production, yeah*”, and then only after this does their teammate ask about difficulty, “*Do you think this is five hours 8 hours?*” However, it was considered if a teammate was struggling with an issue. This is shown when one student asks “*But I’m going to need someone to explain to me how to use things like piling and rough like?*”. Here it is seen that one student will take on a task they do not personally have experience with, but know one of their teammates could offer support. Students are also willing to use the experience of another group mate to help them better compete a task, as shown when one student says, “*I mean I would like you maybe to also join in with the general UI stuff, because you already like working a bit on that.*”

In the beginning, students took on tasks they found interesting or believed they could do well. The student with the highest self-reported experience took on a task deemed easier, saying “*I would take this part because I can just plug it*”. They believe that since they can do it quickly and easily, they should take it. On the other hand, another student with less experience took on a harder task because they found it interesting, saying, “*I feel like I would want to work on getting, like, the back end working. Also, getting simple system information on everything going on, like if you can extract information from the device.*” This shows that personal interests also play a role in task allocation.

In the next weeks, students tended to stick to the type of task they had started with, whether the issue was easy or difficult. One student said, “*But I feel like we don’t have that much to discuss because most people already have kind of an idea of what they’ll be doing.*” They had built some experience in the area, and found it logical to continue with what they knew. This is further shown when a student says, “*Since you’re already familiar with the network stuff, maybe it’s better if you take on them.*” This did not relate to their experience prior to the project, but to what they had familiarized themselves with in the first weeks.

Later in the project, a common factor that came up with all groups regarding task distribution was necessity. One student said, “*That is, once we have the structure, with that mindset, we can start working on the test because we’re really neglecting that part of the goal.*” Once this was addressed, other teammates quickly volunteered to take on this issue. In another team, one student says “*I would love to take a gander at this, but not high priority so...*”, which implies that the im-

portance of an issue overrules a student's personal interest in a topic. The students took on additional tasks that needed extra work to be done whether they had worked on them before or not.

The correlation between the difficulty of tasks, measured as the average weight of the tasks each student completes, and the four measurements of prior experience was also graphed.

4.3 RQ3

To see the correlation between prior experience and task reassignment, an analysis was again performed. On the X-axis, the same metrics for prior experience were used as above: years of programming experience, years of university, experience ranked from 1-10, and experience compared to classmates. These were then compared to each student's number of tasks given away and taken on.

The strongest correlation in both cases was with regard to the metric: self-reported experience on a scale of 1-10 (See figures 5 and 6). For tasks given away, $r(18) = -.31$, which indicates moderate correlation. However, the correlations are again not statistically significant, as the sample size is under 1000 and $p = .166$ [5]. In the case of additional tasks reassigned to a person, $r(18) = -.17$ and $p = 0.453$. This indicates a low correlation was found between the variables. However, as p is again greater than .05, this is not a statistically significant finding.

And has an R^2 value of .09. The corresponding graph is shown in figure 1 below (the other graphs are available in the appendix). Here you see that students who report lower programming experience are more likely to have their tasks taken over by others. However, since the r^2 value is not very high, it is likely that there are many other factors that also influence a student's likelihood of giving a task away. The r and p values of the other analyses can be seen below, and the corresponding graphs can be found in the appendix.



Figure 5: Number of issues given away by a student compared to self-reported programming experience on a scale of 1-10

Additional analyses were performed to find a correlation between prior experience and the likelihood of being assigned to additional tasks.

- Experience compared to classmates: $r(18) = -.12$, $p =$

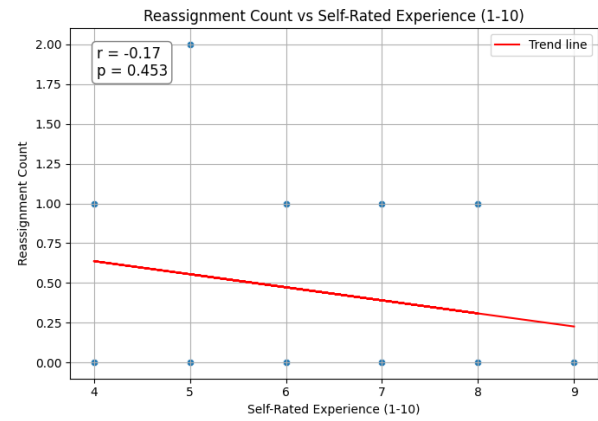


Figure 6: Number of issues reassigned to a student compared to self-reported programming experience on a scale of 1-10

.594

- Years of Study: $r(18) = -.10$, $p = .651$
- Years of programming experience: $r(18) = -.04$, $p = .873$

The correlation between prior experience and the likelihood of giving away tasks previously assigned was also analyzed.

- Experience compared to classmates: $r(18) = -.24$, $p = .277$
- Years of Study: $r(18) = -.12$, $p = .596$
- Years of programming experience: $r(18) = -.23$, $p = .294$

For RQ3 the possibility of a non-linear relationship between the variables was also accounted for through a Spearman analysis as seen in table 3. These results again did not produce any additional statistically significant correlation between the variables.

Experience Metric	ReassignmentCount	GaveAwayCount
YearsStudy	-0.08 / 0.7391	-0.17 / 0.4434
ExperienceSelf	-0.13 / 0.5783	-0.33 / 0.1336
ExperienceRelativeNum	0.01 / 0.9793	-0.25 / 0.2691
YearsProgramming	0.03 / 0.8959	-0.11 / 0.6310

Table 3: Spearman correlation (r / p) between experience metrics and two target variables.

Overall, it has been found that in some cases there is a moderate correlation between some measures of prior programming experience and the distribution of tasks in student software development projects. Across all research questions, years of programming experience had the strongest possible correlation. However, none of these findings have proven statistically significant, this means that it is possible that all found correlations appeared purely by chance. It is possible that if other metrics were chosen or a larger number of subjects were used, there may be stronger evidence supporting a correlation between these variables. However, for the scope of this research, the results were not statistically significant.

5 Responsible Research

This research follows the core principles of responsible research, and uses The Netherlands Code of Conduct for Research Integrity [7] as a guideline.

- **Honesty** - All findings, whether positive or negative, have been reported in the results section, no data has been fabricated, and margins of uncertainty have been described.
- **Scrupulousness** - When designing this study, all methods are based on previous research done in the field.
- **Transparency** - All steps taken to acquire the data used in this study have been explained and no results or data have been withheld.
- **Independence** - Only scholarly sources have been used to design the study and all papers came from accredited journals.
- **Responsibility** - The interests of the students participating in this study have been accounted for. The identities of all participants have been made anonymous and no personal data has been shared outside the research group. Additionally, all participants consented to the use of their data for this research and when consent was revoked all data was quickly removed. These principles have been considered throughout the entirety of the project.

Additionally, before beginning, an HREC was filed to ensure the research complied with TU Delft standards. Beyond these principles, the importance of reproducibility was also considered. In order to maintain this, the methodology and results of this study have been clearly and accurately documented. All steps of both analyses have been detailed so that future researchers will be able to recreate these results.

6 Discussion and Limitations

The findings of this study revealed insights into the possible factors influencing task distribution in student software development projects. Contrary to initial expectations, prior programming experience was not identified as one of the most prominent factors influencing student task distribution.

6.1 Discussion

Overall, prior programming experience was not found to be one of the most prominent factors influencing student task distribution. However, these results can be used to build on the discoveries of previous researchers, and hopefully indicate useful future work.

The thematic analysis of transcribed sprint meetings of students following the second year TU Delft course software project revealed possible insights into the reasoning behind task distribution in student software development projects.

The analysis indicated that the distribution of different types of tasks during the sprint meetings is not completely equal. The analysis revealed that some students assume a more managerial role during the meetings, and that they work to guide the conversation and their fellow students throughout the project. This conclusion supports the work of Ramin et al.

that not all contributions to software engineering are technical [9]. Additionally, the roles taken on by the two most organizationally oriented members do correspond to the work of Stribos and De Laat, as they identify common roles students take on in small (6 or fewer students) software development projects. The main “manager” could possibly correspond to the team captain, and the second student who often contradicts the “manager” to point out oversights could correspond to the over-rider mentioned in their paper. This would leave the quieter teammates to fall under the ghost and free rider labels. However, this is not certain as their technical contributions were not taken into consideration in this section of the thematic analysis. This indicates that perhaps the work done by Stribos and De Laat may not be comprehensive in the case of different types of tasks done by students,

The results supported the findings of Meulen and Aivaloglou, that task division is affected by the students’ prior experience [11]. Especially at the beginning of the project, when students have less pressure and more freedom to select the tasks they personally would like to complete. In the early middle stages, the analysis showed that students often continued to work on issues related to their initial issues. This does still support Meulen and Aivaloglou as these are now the areas of the project that each student has the most experience with. However, as time passes, the analysis found that students’ reasoning begins to change. In the later stages of the project, the research found that the main factor influencing task distribution is the completion of essential issues (must-haves). At this point, the pressure to complete the project on time and with a good grade forces students to take on new tasks that they may not yet have experience with.

As shown in the results section, there has not been a strong correlation found between the chosen metrics for prior experience and the distribution of tasks in student software development projects. Since it has been shown in previous research that there is often an imbalance in task distribution [2] [6], it is likely that there is another factor besides prior experience that causes this divide. This conclusion does somewhat align with the findings of Kopec-Harding et al. [6] as they found that students often change their contribution style from project to project. Because these projects were directly after each other and the students’ lines of code contributed varied greatly, both upward and downward, this would also imply that prior experience was likely not the most influencing factor for the size of their contribution. This contradicts the original idea that it is essential for lecturers and TA’s to account for prior skill during group formation and grading in order to allow students equal chances to develop.

6.2 Limitations

This study has several limitations that could affect the validity of the results. The limited period of this research, the difficulty of recruiting willing participants, the small sample from which we could recruit participants, and the self-reported nature of the data are the most significant limitations.

Because the research time frame overlapped with the software project, it was not possible to follow the students to the completion of their work. Only the first six weeks of repository data have been included in the analysis. It is possible

that students may have changed their behavior during the final four weeks of the project or that some patterns were missed due to lack of data.

Additionally, fewer than the ideal number of participants were recruited for this study. This limits the statistical significance of the results. As there were only 20 participants in total, it cannot be definitively concluded that the correlations found were caused by the investigated variables rather than occurring by chance.

Furthermore, since all subjects were volunteers from TU Delft, the findings cannot be confidently generalized to all student software development projects. It is also possible that biases have been introduced in the recruitment process. This may be because students with more self-confidence and knowledge are more likely to agree to participate in a study about their work and project contributions.

Finally, a significant amount of the data collected relied on information provided by students. In the case of collecting prior programming experience, the survey relied on the students' own opinions and could be affected by their skills. This could introduce a bias, because a student's own self-confidence more than actual skill could affect the results. Additionally, all data collected on the issues completed by students relied on their own documentation. This caused problems with data collection, because many issues created were resolved but not assigned to any member of the team, so it was not possible to use all issues in the analysis. Furthermore, it is possible that there were no issues created for all the tasks done. This would also exclude some tasks done by students from this research.

7 Conclusions and Future Work

This study examined the correlation between prior programming experience and task distribution in student software development courses. The types of tasks, whether managerial or technical, that students completed were investigated. Additionally, the average difficulty of the tasks each student takes on has been compared to his or her experience level. Finally, the relationship between task reassignment and prior experience was analyzed.

7.1 Conclusions

Overall, some insights were found into the reasoning behind task distribution in student software development projects.

The thematic analysis revealed that students tend to take on specific roles in the project, with some becoming a sort of scrum master for the project and others preferring to contribute in a more technical manner. Additionally, the analysis supported the idea that prior programming experience does have some impact. However, it also found that student's priorities shift as the project progresses. This shows that it is not the only factor that students consider when dividing tasks among themselves.

The quantitative analysis did not reveal as many insights into the correlation between prior programming experience and issue allocation. Most of the results found were statistically insignificant, so this suggests that either the pool of participants was either too small to draw definitive conclusions

or that the chosen measures for prior programming experience do not correlate with task distribution.

7.2 Future Work

As a result of the smaller scale of this study, a possible avenue for future work relating to this research would be a larger-scale replication of the work done in this paper. This larger study could ideally follow students throughout the entire ten weeks of their project instead of the six used here, and could even include students from other universities and courses. This could build on the insights given in the thematic analysis relating to the changes in student behaviors throughout the project, as it would be viable to also analyze conversations in the last weeks of the project. Additionally, it would possibly provide more data for the quantitative analysis and improve the likelihood that a relevant conclusion could be drawn. Furthermore, if the scale of the study was increased to include more students, especially those from other universities, the results would be more broadly generalizable.

Another potential option for future research would be to investigate alternative reasons for student task division decisions. The quantitative analysis found that there was little correlation between prior experience and task allocation, and that may not only be due to the relatively small sample size. Alternative reasons such as gender, personality type, self-confidence, or other factors could also potentially have a significant influence on student participation in software projects.

Finally, another consideration that could be addressed in future works would be investigating a multivariate correlation between the variables. While significant correlations were not found between task distribution and the individual chosen metrics, investigating subsets or all metrics combined could possibly provide further insight. This should be investigated before definitively concluding that there is no correlation between prior programming experience and task distribution in student software development teams.

References

- [1] J. Feigenspan, C. Kästner, J. Liebig, S. Apel, and S. Hanenberg. Measuring programming experience. In *2012 20th IEEE International Conference on Program Comprehension (ICPC)*, pages 73–82, 2012.
- [2] Ilenia Fronza and Xiaofeng Wang. Towards an approach to prevent social loafing in software development teams. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 241–246, 2017.
- [3] 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)*, pages 531–540, 2020.
- [4] Laura Heels and Marie Devlin. Investigating the role choice of female students in a software engineering team project. In *Proceedings of the 3rd Conference on Computing Education Practice (CEP ’19)*. Association for Computing Machinery, 2019.
- [5] Eugene Komaroff. Relationships between p-values and pearson correlation coefficients, type 1 errors and effect size errors, under a true null hypothesis. *Journal of Statistical Theory and Practice*, 14, 2020.
- [6] Kamilla Kopec-Harding, Sukru Eraslan, Bowen Cai, Suzanne M. Embury, and Caroline Jay. The impact of unequal contributions in student software engineering team projects. *Journal of Systems and Software*, 206:111839, 2023.
- [7] Netherlands Code of Conduct for Research Integrity. Netherlands code of conduct for research integrity. <https://www.nwo.nl/en/netherlands-code-conduct-research-integrity>, n.d. Accessed: 2025-06-02.
- [8] 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)*, pages 1–5, 2018.
- [9] Frederike Ramin, Christoph Matthies, and Ralf Teusner. More than code: Contributions in scrum software engineering teams. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering*, 2020.
- [10] J.-W. Strijbos and M. F. De Laat. Developing the role concept for computer-supported collaborative learning: An explorative synthesis. *Computers in Human Behavior*, 26(4):495–505, 2010.
- [11] Anna van der Meulen and Efthimia Aivaloglou. Who does what? work division and allocation strategies of computer science student teams. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 273–282, 2021.
- [12] Joanna Wolfe and Elizabeth Powell. Strategies for dealing with slacker and underperforming teammates in class projects. In *Proceedings of the IEEE International Professional Communication Conference*, 2015.

A Appendix

A.1 AI Usage

During the creation of this research paper AI and grammar checking tools were used to improve the overall structure and flow of this paper. Prompts include:

- Can you help me find strengths and weaknesses in the flow of this section of my paper?
- What would you suggest as areas of improvement for this research paper?

A.2 Questionnaire

The questionnaire used to measure prior-programming experience consisted of the sections consent and demographics.

Consent

You are being invited to participate in research about Student Software Development Project Groups. This study is being done as part of the CS3000 Research Project at the TU Delft. The data will be used for analysis by up to five BSc students working on the Research Project and their supervisor. The survey will take approximately 20 minutes to complete. We ask that you fill out this survey honestly, and to the best of your ability. One question asks for the link to your repository, by filling this in, you consent to the researchers having read-only access to your repository for the duration of the Research Project.

As with any online activity the risk of a breach is always possible. To the best of our ability your answers in this study will remain confidential. Participating in this research does not impact your course performance, and the course instructors do not have access to the data. We will minimize any risks by storing the data on TU Delft servers only. We will be collecting your email address on this consent form for consent purposes and for linking your survey answers to the observation data, but this will not be used in data analysis and can only be accessed by the responsible researcher. This form will not be stored together with the survey answers, but in a separately encrypted space to minimise the risk of a data breach. After this research, the data will be stored for at least ten years in a restricted repository for reproduction purposes, in line with the TU Delft Research Data Framework Policy.

Your participation in this study is entirely voluntary and you can withdraw at any time and for any reason. You are free to omit any questions. You can revoke consent at any time and your data will be deleted, you do not have to provide a reason. Please note that we cannot remove results that have been published already. If you have any questions, you can reach the responsible researcher

that you have read this opening statement and explicitly consent to participate in this research and the processing of your data.

Answer: _____

Demographic Information

2. What is your age?

Answer: _____

3. How many years of studying CSE at TU Delft have you completed?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5+

4. What is your gender?

Answer: _____

5. Please enter the link to your GitLab repository, so we can request (read-)access

Answer: _____

6. What is your Gitlab name or email address?

Answer: _____

A.3 Programming Experience

7. How do you estimate your programming experience on a scale from 1 to 10, where 1 is very inexperienced and 10 is very experienced?

Answer: _____

8. Please answer the following statements.

8.1 How do you estimate your programming experience compared to your classmates?

Very inexperienced **Neutral** **Very Experienced**
☐ ☐ ☐ ☐ ☐

8.2 How experienced are you with logical programming? (This refers to programming based on formal logic, where code expresses facts and rules — examples include Prolog, Datalog, or logic-based exercises in courses like AI or Declarative Programming.)

Very inexperienced **Neutral** **Very Experienced**
☐ ☐ ☐ ☐ ☐

8.3 How experienced are you with object-oriented programming?

Very inexperienced **Neutral** **Very Experienced**
☐ ☐ ☐ ☐ ☐

8.4 How complex do you perceive your code to usually be? (1 = very low complexity, 5 = very complex)

Very low complexity **Neutral** **Very Complex**
☐ ☐ ☐ ☐ ☐

8.5 How experienced are you with front-end development?

Very inexperienced Neutral Very Experienced

☐ ☐ ☐ ☐ ☐

9. For how many years have you been programming?

Answer: _____

10. How many programming languages do you know at a medium level or higher?

Answer: _____

11. How many courses have you taken that required you to implement source code?

Answer: _____

12. For how many years have you been programming for larger software projects, e.g., in a company?

Answer: _____

13. For how many years have you been programming outside of academic context (as a job)?

Answer: _____

14. How long were the professional projects typically?

- ☐ Not applicable
- ☐ <900 Lines of Code
- ☐ 900-40000 Lines of Code
- ☐ >40000 Lines of Code

A.4 Additional Graphs

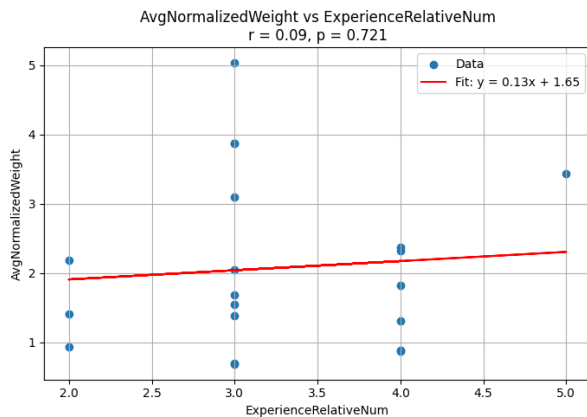


Figure 7: Average weight of issues done by each student compared to relative experience

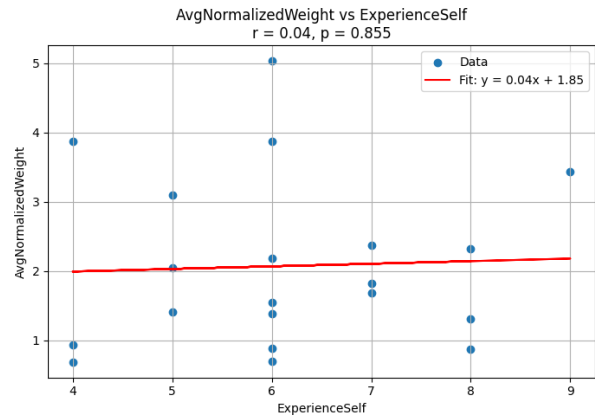


Figure 8: Average weight of issues done by each student compared to rating of programming skills 1-10

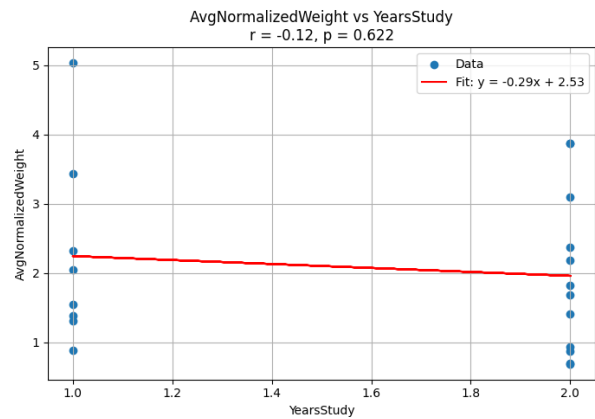


Figure 9: Average weight of issues done by each student compared to years of study

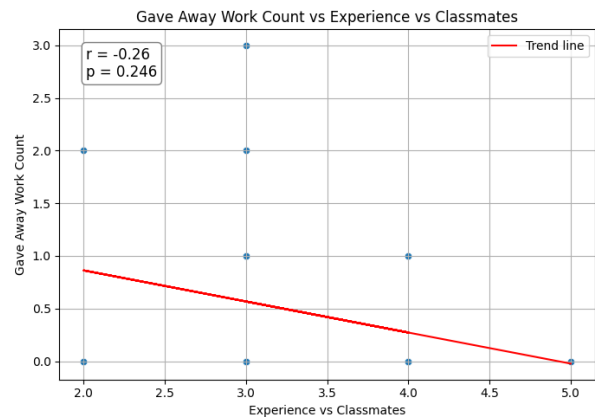


Figure 10: Percentage of issues given away by each student compared to relative experience



Figure 11: Percentage of issues created or assigned by each student away compared to years of programming experience

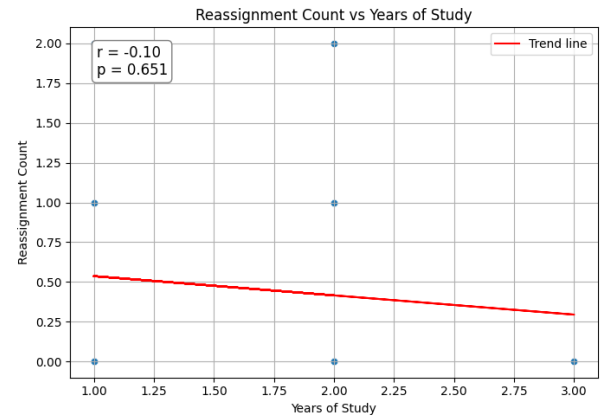


Figure 14: Percentage of issues created or assigned by each student away compared to years of programming experience

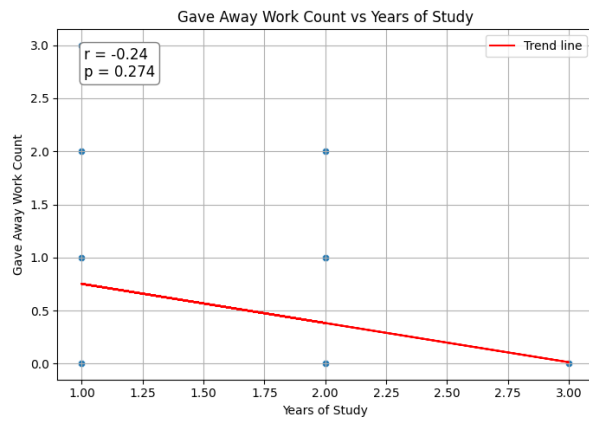


Figure 12: Percentage of issues given away by each student compared to compared to years of study

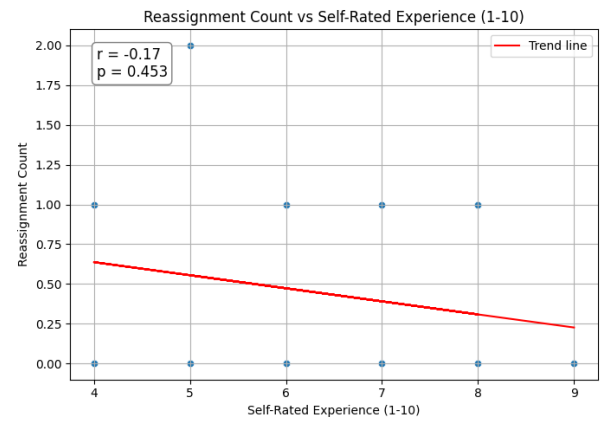


Figure 15: Percentage of issues reassigned to each student compared to self reported experience (1-10)

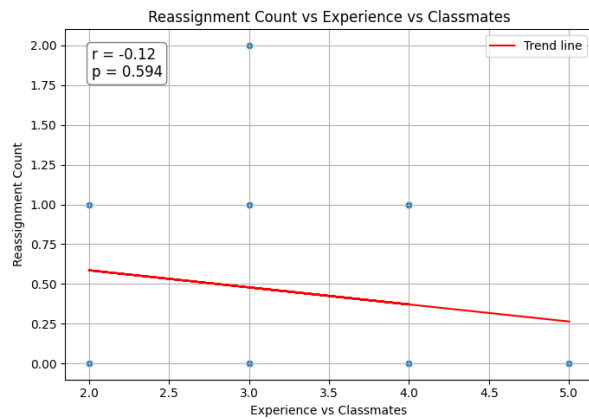


Figure 13: Percentage of issues reassigned to each student compared to relative experience

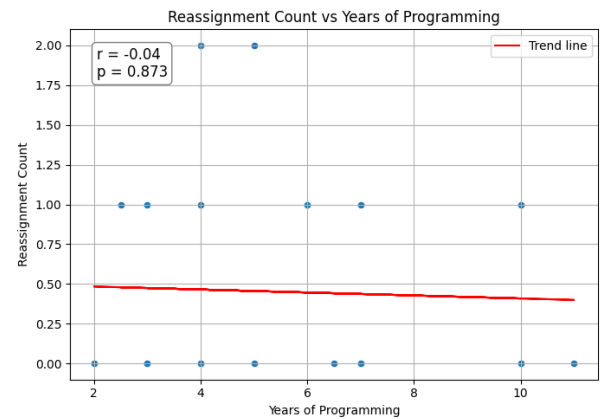


Figure 16: Percentage of issues reassigned to each student compared to years of programming experience

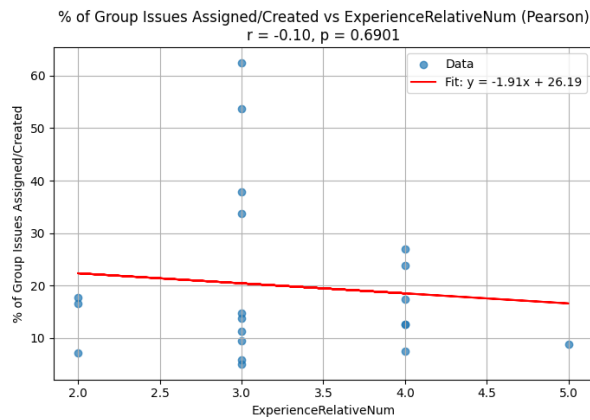


Figure 17: Percentage of issues created or assigned by each student compared to relative experience

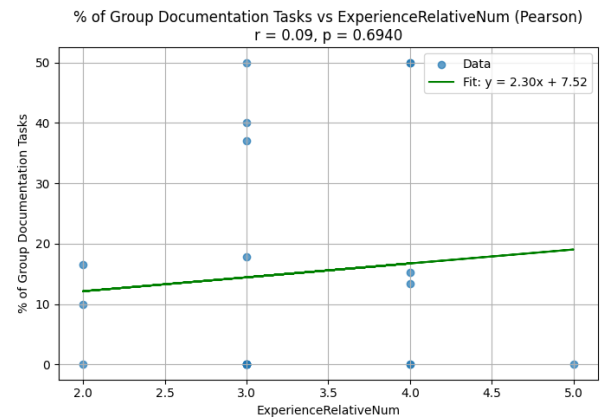


Figure 20: Percentage of documentation related issues done by each student compared to relative experience

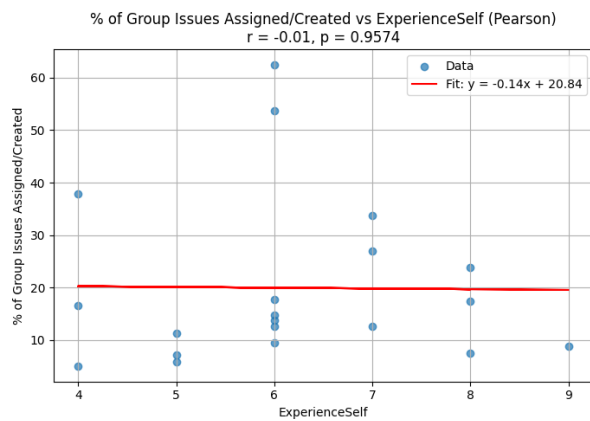


Figure 18: Percentage of issues created or assigned by each student compared to self reported experience (1-10)

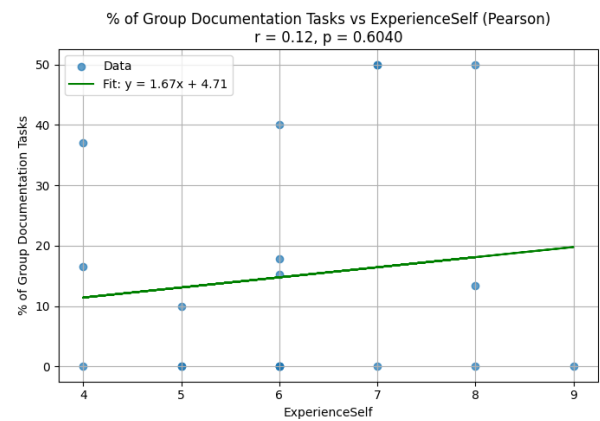


Figure 21: Percentage of documentation related issues done by each student compared to self reported experience (1-10)

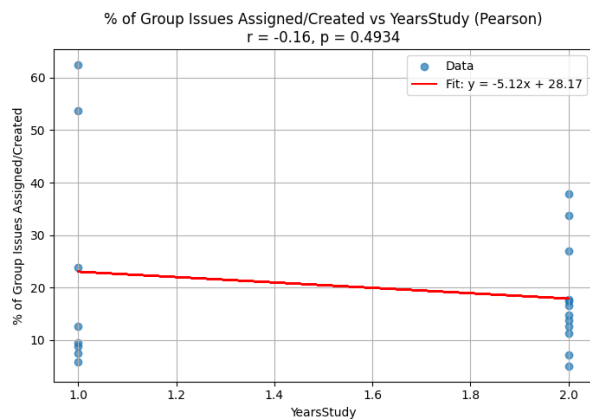


Figure 19: percentage of issues created or assigned by each student compared to years of study

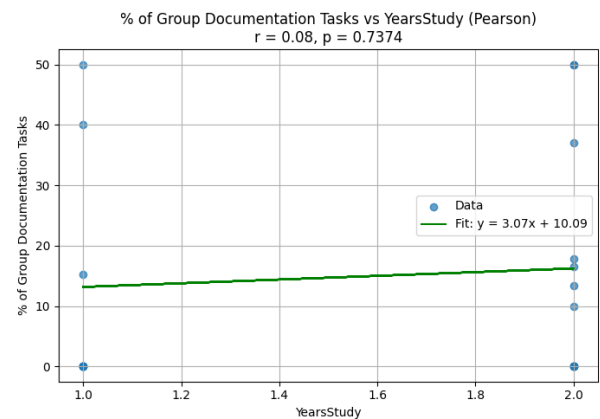


Figure 22: Percentage of documentation related issues done by each student compared to years of study

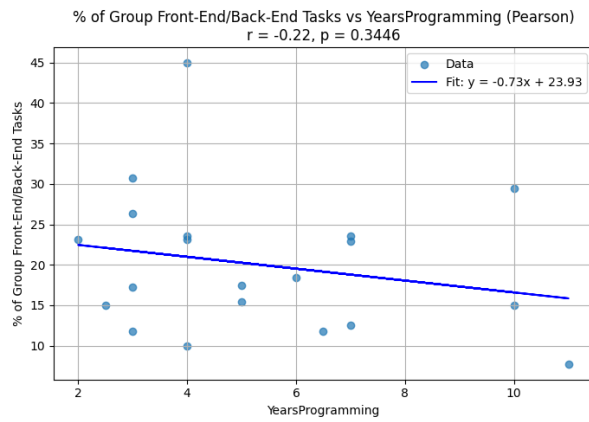


Figure 23: Percentage of issues technical issues done by each student away compared to years of programming experience

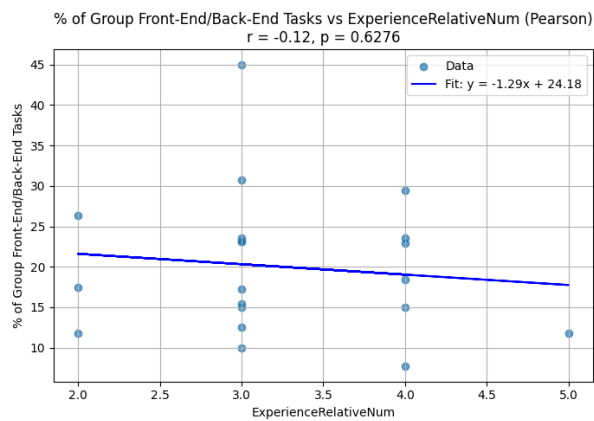


Figure 24: Percentage of issues technical issues done by each student away compared to relative experience

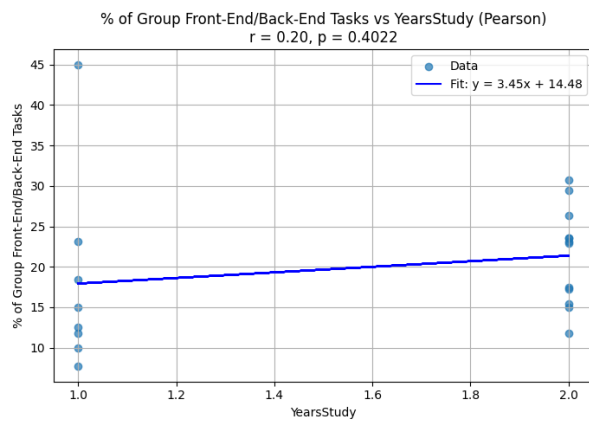


Figure 25: Percentage of issues technical issues done by each student away compared to years of study