An Interactive Approach to Teaching Git Version Control System

Elgun Jabrayilzade
Bilkent University
elgun@bilkent.edu.tr

Fatih Sevban Uyanık
Bilkent University
fatihsevban15@gmail.com

Emre Sülün
Bilkent University
emre.sulun@bilkent.edu.tr

Eray Tüzün
Bilkent University
eraytuzun@cs.bilkent.edu.tr

Abstract

Although the Git version control system is widely used in software engineering, it has been observed that most Computer Science and Software Engineering students do not have the necessary knowledge and practices to use Git. To address this issue, we have prepared a Git and GitHub training program consisting of four sessions as a part of the Object-Oriented Software Engineering course where junior students utilized these tools for their term projects. The program was conducted in three academic terms for a total of 258 students.

To evaluate the effectiveness of the training sessions, we have conducted two surveys, before (224 respondents) and after (200 respondents) the program. According to the survey results, the number of students considering themselves insufficient to use the tools for their projects decreased from 67% to 9% after the training program. Additionally, the majority of the students found the lectures and laboratory assignments beneficial.

1. Introduction

Version control systems allow software developers to track changes to source code over time and recall a specific version later if needed. These systems are generally divided into two categories: centralized and distributed. In centralized version control systems (e.g., Subversion[1]), there is a single server that contains all the project files, and users can only work on the latest snapshot of the project. One downside of such systems is that there is a single point of failure, meaning that the project could be lost if something goes wrong with the server (e.g., database corruption). In the last decade, many developers moved their projects to distributed version control systems such as Git[2] where each user stores the complete history of a project, making it immune to server related issues. Git is currently one of the most popular version control systems, and it was used by 87.2% of responding developers in the annual developer survey carried out by Stack Overflow in 2018 [1].

However, despite the trend, it has been noticed that most students lack the necessary knowledge and practice to use Git. This is because version control systems are not adequately addressed in typical Computer Science or Software Engineering curricula [2, 3]. To this end, we prepared an educational training program for junior Computer Science students as a part of the Object-Oriented Software Engineering course in three academic terms (which are referred to as the first, second and third iterations of the program). In this course, students are divided into 5-member teams to develop a software project using object-oriented principles collaboratively. Moreover, we made it mandatory for students to use Git and GitHub[3] throughout their term projects. It should also be noted that this is the first course in the university where students are introduced to Git and GitHub.

The training program aimed to introduce frequently used Git and GitHub concepts to students and prepare them for their software engineering careers. We followed a similar structure for the training programs in all iterations, starting with a pre-training survey to see the students' level of experience in Git and GitHub. Then, we delivered a lecture introducing the general concepts of version control systems and GitHub. Next, we conducted a live session to show the Git instructions in more detail. Before executing the last session, we allowed the students to learn more about and practice the tools remotely for a week through online materials as well as the lecture slides and tutorial videos we have

[1] https://subversion.apache.org/

[2] https://git-scm.com/

[3] https://github.com/

Table 1: Comparison of our program and other teaching methodologies.

	Traditional training	Reid et al. [4]	Isomöttönen et al. [5]	Bonakdarian et al. [6]	Beckman et al. [7]	Online tutorials	Our program
Introducing the concepts	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Live code examples	No	No	No	No	No	Yes	Yes
Hands-on exercises	No	Yes	Yes	Yes	Yes	Yes	Yes
Interactive lab	No	No	Yes	No	No	No	Yes
Online delivery	Yes	No	No	No	No	Yes	Yes
Survey about the effectiveness	N/A	No	Yes	Yes	No	N/A	Yes
Large survey sample (>100)	N/A	No	No	No	No	N/A	Yes

prepared. Afterward, as the last session of the program, the students were given a graded (5% of the total course grade) laboratory assignment, where they had to execute a specific scenario using Git/GitHub operations. At the end of the program, we conducted a post-training survey to assess the efficacy of the training sessions.

The rest of the paper is organized as follows. Section 2 reviews the literature regarding the use of Git and GitHub in education and the teaching of these tools. Section 3 describes the training process in detail, while Section 4 presents the results and observations. Section 5 and 6 discuss the potential threats to the validity of the study and lessons learned, respectively. Finally, Section 7 concludes the study.

2. Related work

2.1. VCS as a course management platform

There are various studies in the literature that report experiences of using version control systems for managing course materials and assignment submissions. Before the release of Git, instructors mostly employed Subversion and Concurrent Versions System (CVS)[4] version control systems. Linder et al. [8] had students use CVS for the weekly assignments and term project of the software design course. Reid and Wilson [4] used CVS as a platform for managing the course where assignments were shared through CVS, and students were expected to submit their solutions by committing to the respective CVS repository. Furthermore, teaching assistants graded students' submissions through CVS. Liu et al. [9] used CVS to monitor and analyze each project repository of students. The analysis aimed at understanding how students collaborate and determine if there is a relationship between their grades and collaboration patterns. Clifton et al. [10] used Subversion as a course management tool to enhance the teaching activities by allowing the instructors and teaching assistants to collaborate remotely in two CS1 courses.

More recent studies employed Git as a platform to disseminate course materials and facilitate assignment submissions [5, 7, 11–15]. Two studies integrated Git into non-CS degrees. Beckman et al. [7] employed the tool in four statistics courses, one of them being a graduate-level course, whereas Lawrance et al. [15] utilized it in a CS1 course for mechanical and electrical engineering majors.

Two studies conducted a survey for measuring the effectiveness of the platform. Conner et al. [12] carried out a single-question survey for 107 students where 57% of them agreed that the weekly submissions helped their understanding of Git. Harannen and Lehtinen [13] also surveyed students at the end of the course to determine their previous experience, perceptions towards using Git, and their general feedback about the course procedures. The majority of students who answered the survey (141 students) agreed that using Git as a course management tool was effective.

2.2. Teaching VCS

There are several experience reports about teaching version control systems. Alongside using VCS as a course management platform, in three studies, instructors also taught the tool to students. Reid et al. [4] conducted a one-hour lecture and one-hour tutorial sessions for teaching the basics of CVS. Similarly, Isomöttönen and Cochez [5] demonstrated Git concepts, after which students practiced the tool in a single session. Beckman et al. [7] prepared a 15-minute tutorial and 30-minute practice sessions for teaching Git to students.

Unlike the previous studies, Bonakdarian [6] adopted a four-stage tutorial for teaching Git and GitHub as a part of the "Introduction to Unix" course. The stages were composed of a tutorial on the command line interface followed by GitHub and two Git tutorials. Students (17 in total) were surveyed before and after the tutorials answering two questions: (1) *How confident are you using Git/GitHub?* and (2) *Do you think*

[4] https://www.gnu.org/software/trans-coord/manual/cvs/cvs.html

knowing the basics of Git/GitHub will be helpful for your future work (in school/career)?. The majority of students felt confident about their skills in Git/GitHub after the tutorial sessions.

There are also various kinds of tutorials or courses online for people to benefit from. These include but not limited to, YouTube videos[5], online courses[6] and games for learning Git[7]. However, such online tutorials lack interactivity and teaching assistants where students can get feedback whenever they are stuck.

Table 1 compares current teaching methodologies and prior approaches with our method in teaching VCSs. The traditional training includes teaching via presentation slides for introducing the VCS concepts. Our work differs from the aforementioned studies and online tutorials in several ways. First, we carried out the laboratory session in an interactive manner where teaching assistants helped the students throughout the session. Second, the tutorial sessions were carried out on-campus in iteration 1 and online through Zoom[8] for iteration 2 & 3, indicating that the program is suitable for both environments. Moreover, we prepared an extensive survey to assess the effectiveness of the program in detail, where we got responses from 200 students (78% response rate).

3. Program setup

The process we followed to execute the training program is shown in Figure 1. We followed the same program structure in all iterations. It consists of four main sessions (lecture, live session, offline learning, and laboratory assignment) and additional pre and post-training surveys. The lecture slides used for teaching and the laboratory assignments are available online[9]. In the following subsections, we state the learning objectives of the tutorials and elaborate on the parts of the program.

3.1. Learning objectives

Participants of the program were junior students majoring in Computer Science. Thus, considering their potential future careers, we have defined the following three learning objectives for the Git and GitHub training program:

- Understand the benefits of using version control systems.
- Grasp the main concepts of Git workflow.
- Develop the required skills to use Git and GitHub for future projects.

3.2. Pre-training survey

Before starting the training program, we conducted a preliminary survey to determine the knowledge and experience level of students in Git and GitHub. The survey included the following five questions:

1. Have you ever used Git? (Yes/No)
2. If you have used Git, have you used GUI apps for Git? (Yes/No)
3. If you have used Git/GitHub, which operations and concepts have you performed? (Multiple selections from {commit, clone, fork, push, pull, branch, fetch, merge, checkout})
4. In how many projects did you use GitHub? (0, 1-5, 6-10, 10+)
5. Do you see yourself sufficient to use Git and GitHub during your term project? (Yes/No)

3.3. Lectures and live coding session

The training program started with a two-hour lecture on Git and GitHub. The lecture was conducted in person for the first iteration and online for the second and third iterations due to the pandemic. Initially, we showed what version control systems are and why they are essential for software development projects. Then, we explained the basic Git operations such as *init, add, commit, push, pull, clone* as well as some advanced ones such as *rebase, squash, amend,* etc. There are various Git GUI applications that allow performing the operations much easier and provide a better visual representation of the commit history. Thus, we also demonstrated how to apply Git operations using Sourcetree[10] in the first iteration. However, because such GUI applications have limited operating system support and they may prevent students from learning Git in detail, we decided to utilize a command-line interface most of the time in iteration 2 & 3. Furthermore, we went over an example project on GitHub explaining *issues, pull requests,* and other useful collaboration features. We made the lectures interactive by asking questions to students and running polls. Directly after the lectures, we conducted a live coding session for an hour, showing how to install and setup the Git as well as apply the operations on a sample project.

[5] https://www.youtube.com/watch?v=RGOj5yH7evk
[6] https://www.udemy.com/course/git-and-github-bootcamp/
[7] https://ohmygit.org/
[8] https://zoom.us/
[9] https://figshare.com/s/3426342d57238aeaad57
[10] https://www.sourcetreeapp.com/

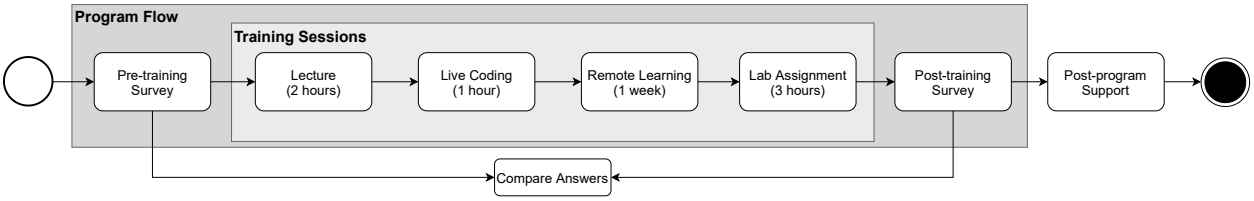


Figure 1: Activity diagram of the training program.

3.4. Remote learning

In addition to the live lectures, the same concepts were also taught on YouTube. That way, the students had the opportunity to revise the concepts in-depth in order to get ready for the graded laboratory session, which was conducted a week after the lectures. The video lectures for the Git and GitHub tutorials can be accessed online[11].

3.5. Laboratory assignment

In the laboratory assignment, the students were asked to implement a mini-project using Git. Students had the opportunity to practice various Git concepts that were taught in the tutorials. The prepared assignments for the iterations of the training program were different in terms of scenario but were similar in structure. A history graph of the assignment from the second iteration is shown in Figure 2. It can be seen that the students performed operations such as *commit*, *branch*, *merge*, *fetch*, etc. An example scenario of a commit operation is shown in Figure 3. The laboratory session was conducted virtually through Zoom in all iterations. Throughout the laboratory session, the teaching assistants were available to help the students with their assignments. The students having problems with their assignments joined the teaching assistants' Zoom meeting rooms and waited in the waiting rooms. The teaching assistants accepted the waiting students to the meeting room one by one and tried to resolve their problems. The students were also able to share their screens during the meeting sessions, enabling the teaching assistants to tackle the issues much more easily.

3.6. Assessment of the laboratory assignment

The assignment was divided into multiple parts where each part had a predetermined score according to its difficulty. The teaching assistants were responsible for grading the students' submissions in all iterations. Firstly, they cloned each student's repository and

[11] https://www.youtube.com/watch?v=n2raDhAP_50&list=PLz0GfNCvpTM7KZekxRPdHBFdReGeg3qza

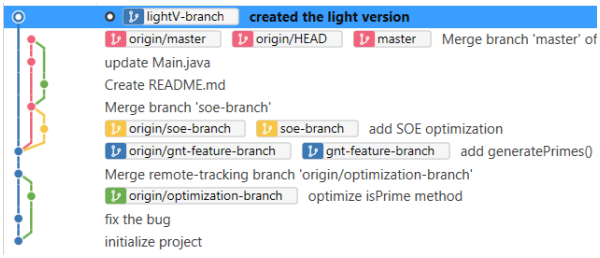
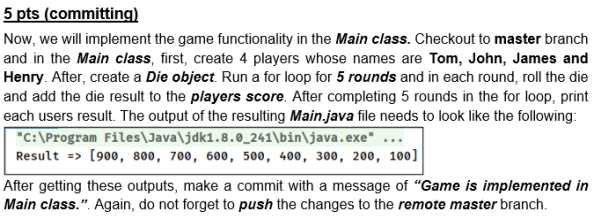


Figure 2: Git history graph of the assignment from the second iteration (taken from Sourcetree).



5 pts (committing)

Now, we will implement the game functionality in the ***Main class***. Checkout to **master** branch and in the ***Main class***, first, create 4 players whose names are **Tom, John, James and Henry**. After, create a ***Die object***. Run a for loop for ***5 rounds*** and in each round, roll the die and add the die result to the ***players score***. After completing 5 rounds in the for loop, print each users result. The output of the resulting ***Main.java*** file needs to look like the following:

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Result => [900, 800, 700, 600, 500, 400, 300, 200, 100]
```

After getting these outputs, make a commit with a message of ***"Game is implemented in Main class."***. Again, do not forget to ***push*** the changes to the ***remote master*** branch.

Figure 3: Example scenario of a commit operation.

extracted the corresponding Git history graph. By looking into the graphs, the teaching assistants were able to see individual commits in each branch created by students and grade the assignment parts accordingly.

3.7. Post-training survey

After completing the laboratory session, a post-training survey was conducted to assess how well the students have learned Git and GitHub. Textual feedback was received from the students about the lecture tutorial and laboratory assignment. The questions that were asked to the students are as follows:

1. How were the Git and GitHub tutorials? (Likert scale)
2. Which Git/GitHub operations have you performed after the tutorials? (Multiple selections from {commit, clone, fork, push, pull, branch, fetch, merge, checkout})
3. How beneficial was the laboratory assignment?

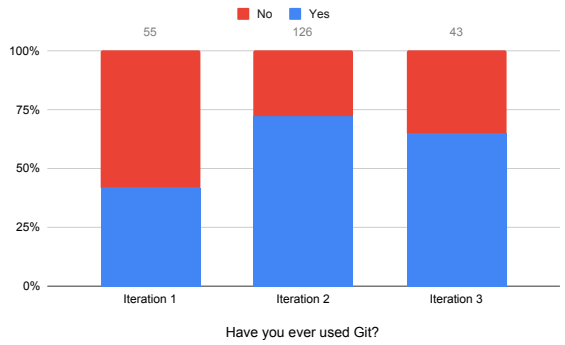


Figure 4: Results of the pre-survey question 1.

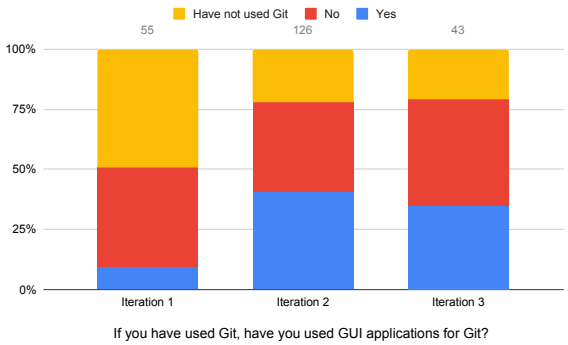


Figure 5: Results of the pre-survey question 2.

(Likert scale)

4. How realistic was the assignment scenario? (Likert scale, iteration 2 & 3 only)

5. How difficult was the laboratory assignment? (Likert scale)

6. After the tutorials, how well did you learn Git and GitHub? (Likert scale)

7. How sufficient do you see yourself using Git and GitHub during your projects? (Likert scale)

8. Did you like the Sourcetree software? (Likert scale, iteration 1 only)

9. What are your opinions about the training program? (Text)

Question 8 was only asked in iteration 1 due to the limitations of GUI applications for Git discussed in Section 6.

3.8. Post-program support

After the lecturing and laboratory assignment, further assistance was given to the students who contacted the teaching assistants. Short Zoom meetings were organized in order to resolve the encountered issues in their projects regarding Git and GitHub throughout the semesters.

4. Results

In this section, we present the results of surveys and the grades of laboratory assignments we conducted in each iteration. The answers to each survey question are shown in Figure 4 to 15, while the distribution of grades can be seen in Figure 16.

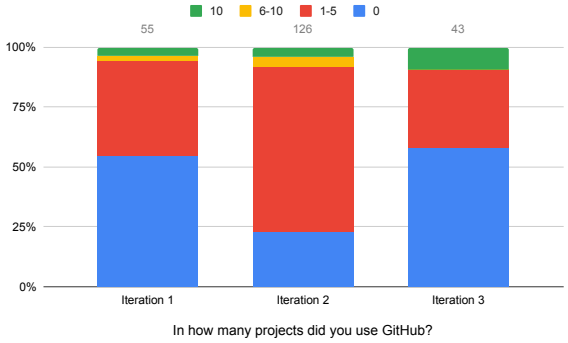


Figure 6: Results of the pre-survey question 4.

4.1. Pre-training Survey

A preliminary survey was conducted to assess the knowledge of the students before the tutorials. The overall response of the survey suggests that 67% of the students did not think that their Git and GitHub knowledge is sufficient (see Figure 7) and even, in the first iteration, around 60% of the class have not utilized Git before (see Figure 4). More importantly, it can be inferred from Figures 6 and 15 that the majority of the students who have used Git and GitHub before experimented with the tools in a few projects and performed basic operations such as *commit*, *push* and *pull* more frequently in comparison to complex operations such as *branch* and *merge*.

There are many graphical user interface (GUI) applications for Git such as Sourcetree, GitKraken[12], and GitHub Desktop[13]. The students were asked whether they used such applications before. The results are shown in Figure 5. In the first iteration, only 9% of the students had used GUI applications for Git before. For the latter iterations, it was approximately 35%.

[12] https://www.gitkraken.com/
[13] https://desktop.github.com/

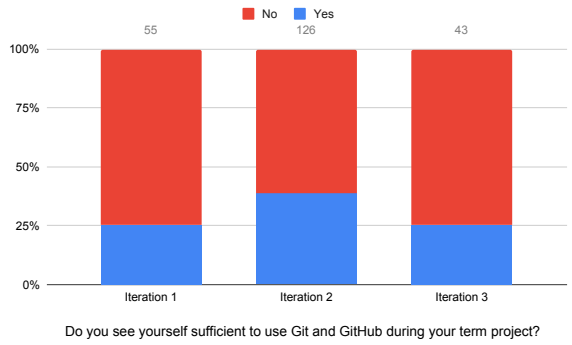


Figure 7: Results of the pre-survey question 5.

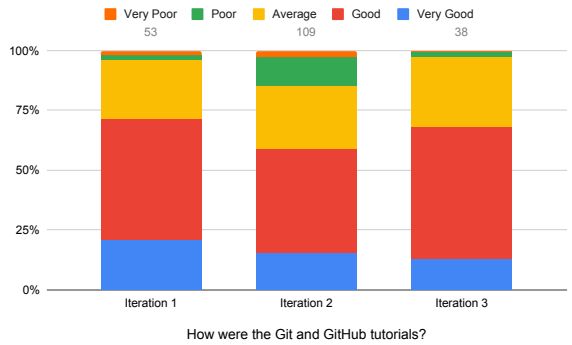


Figure 8: Results of the post-survey question 1.

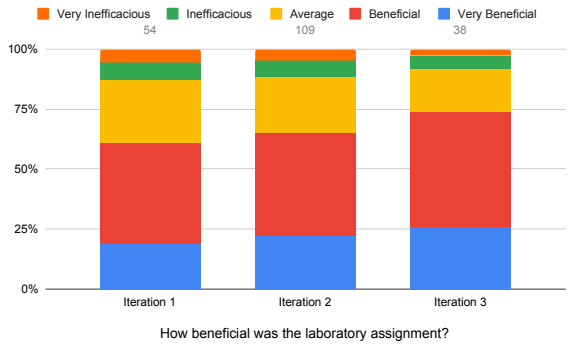


Figure 9: Results of the post-survey question 3.

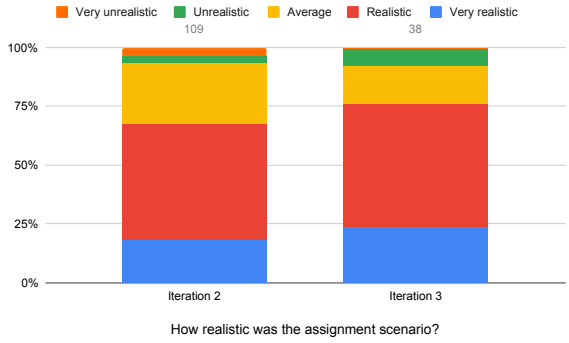


Figure 10: Results of the post-survey question 4 (Iteration 2 & 3 only).

Sourcetree was used in the tutorials of the first iteration. The reason for utilizing Sourcetree was that it has a good commit history visualization and teaching assistants had previous experience in it.

The initial survey results reflect the necessity for having Git and GitHub lectures in the Object-Oriented Software Engineering course, as the main aim of this course is to learn the processes during the development phase of a software product and how to contribute to a software project as a well-organized group. Hence, it can be concluded that the knowledge of Git becomes essential during the development of a software project due to collaboration.

4.2. Post-training Survey

After the laboratory assignment, another survey was conducted to assess whether the students learned the utilization of Git and GitHub to the degree that they can contribute to projects efficiently via these platforms. Furthermore, constructive feedback was collected from the students about the live tutorials, offline learning materials, and laboratory assignments.

From the post-survey results, it can be inferred from Figure 8 that the tutorials in the class and on YouTube were effective because 48% of the students rated the tutorials as good and 17% as very good (all iterations combined).

Approximately 66% (very beneficial, beneficial) of the students found the laboratory assignment beneficial (see Figure 9). Moreover, the results suggest that the difficulty of the assignments was perceived roughly as balanced by the majority of the students in the iterations (see Figure 11).

In the first iteration, students have rated the Sourcetree tool. According to the results, Sourcetree was considered a good tool by the students as roughly 60% of the students voted that the tool is qualified and 28% of the students voted for it as an average tool (see Figure 14).

To assess the effectiveness of the tutorials and the assignment, we compared the results of questions that were asked before the tutorials and after the tutorials.

Before the tutorials, 33% of the students stated that they were capable of using Git in their term projects. However, after the tutorials and the laboratory assignment, 72% of the students stated that

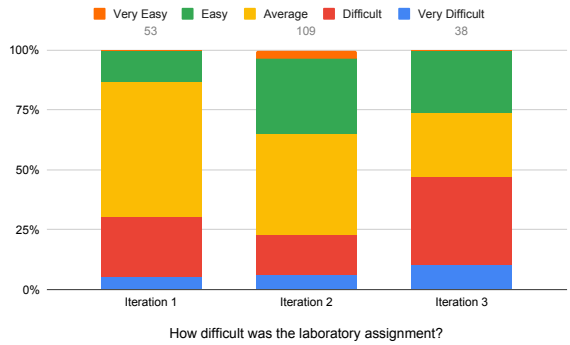


Figure 11: Results of the post-survey question 5.

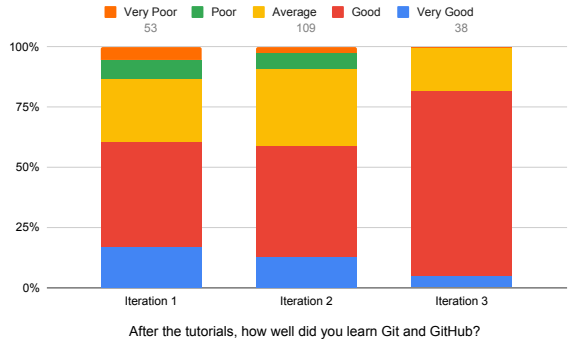


Figure 12: Results of the post-survey question 6.

they considered themselves skilled (very sufficient, sufficient) for using Git in their term projects (see Figure 7 and 13).

Figure 15 shows which Git operations students performed before and after the tutorials & assignment. It can be observed that more students were able to use complicated operations rather than basic operations such as *commit* and *push* after the tutorials and laboratory sessions. Hence, due to this progress, it would be safe to claim that the students gained hands-on experience alongside theoretical experience.

Figure 16 shows the histogram of the grades that the students obtained from the laboratory session conducted in each iteration. It can be seen that the majority of students got scores in the range of 76 to 100 (64%, 83%, 74% in iteration 1, 2, and 3, respectively) in all iterations.

Finally, the claims (see Figure 12) and scores of the students suggest that they learned Git and GitHub well, which indicates the effort in teaching the tools was successful.

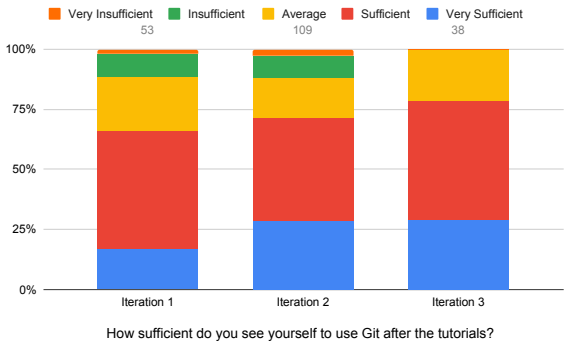


Figure 13: Results of the post-survey question 7.

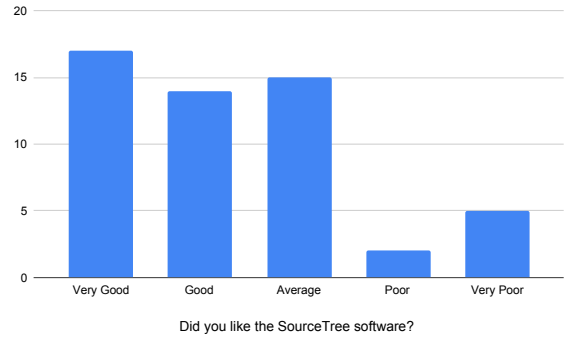
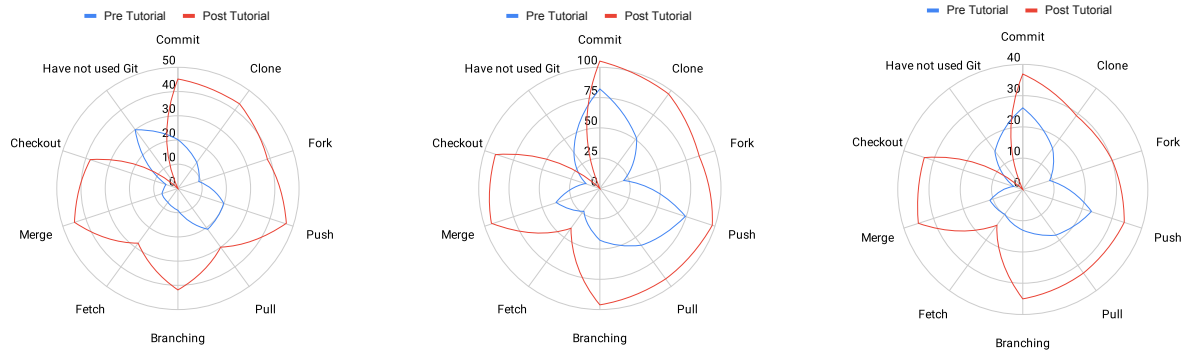


Figure 14: Results of the post-survey question 8 (iteration 1 only).

5. Threats to Validity

In this section, we discuss potential threats that might have an effect on the provided results. First of all, in the first iteration, students specified their identities while responding to their survey questions, whereas, in the second and third iteration, they responded anonymously. Thus, the results in the first training could be a bit biased because they may abstain from answering, knowing that their names would appear in their responses. In addition, it was not compulsory to participate in post-surveys in the second and third iterations. For this reason, 152 of the 169 students participating in the pre-training survey participated in the post-training survey, which may have caused the second program's responses to be biased because 17 students who did not participate could have given negative answers.

Moreover, Git and GitHub are known to provide version control and make it easier to write software as a team. However, laboratory assignments were individual tasks that may not have simulated how Git and GitHub are used in real life. Finally, since the assignment was



(a) Iteration 1 **(b) Iteration 2** **(c) Iteration 3**

Figure 15: Results of the pre-survey question 3 and post-survey question 2 (Which Git/GitHub operations have you performed before/after the tutorials?).

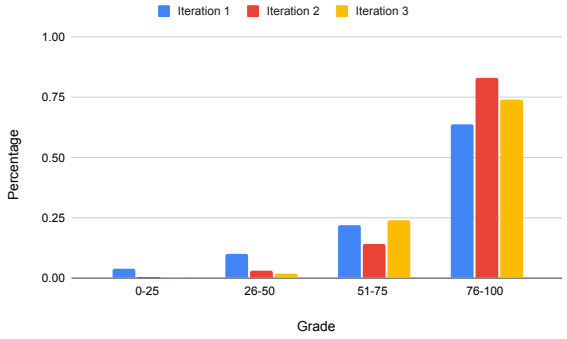


Figure 16: Distribution of student grades from the laboratory assignments.

conducted in a virtual environment, the students could not be inspected thoroughly, which may have caused them to resort to academic misconduct, even though it was mandatory for all students to keep their webcams on throughout the laboratory session.

6. Discussion

In the first iteration, the lecturing was done in a real classroom and at the end of the lecture, students interacted with the instructor and asked their clarifying questions. In the second and third iteration, because of the coronavirus pandemic, the lecturing has been conducted virtually, and likewise, students asked their questions at the end of the lecture. Importantly, the virtual lecture was recorded thanks to the feature provided by the video streaming application. Also, small modifications were done to the tutorial slides in the second and third iterations; however, the YouTube tutorials from the first iteration have been reused.

In addition, adopting an interactive teaching methodology through laboratory assignments and having teaching materials for remote learning such as YouTube lectures enhanced the quality of lecturing because students had the opportunity of revising the concepts afterward and practicing the Git concepts on their own. Moreover, having pre-training surveys and post-training surveys provided meaningful analytics that reflected how successful the training program was. Initially, 63% of the students knew Git, whereas around 92% (very good, good, average) of the students learned it after the training sessions (see Figure 12).

Textual feedback and comments were received from the students at the end of the post-training survey. From the feedback, we observed that the students were satisfied with the training program. Two of the comments were as follows:

> *"I think the lab was perfect. I was starting to feel really bad about not knowing the intricacies of Git, but this helped me a lot."*

> *"Tutorials make me feel better while I need to learn something for courses. Sending videos before tutorials was also beneficial because it enables us to look for the same things and choose right things to study. Having opportunity to see examples and ask questions during lecture time made me to learn better."*

In the first iteration, some students complained about Sourcetree's lack of Linux support. That's why in the second and third iterations, IntelliJ IDEA and the command line interface were used for performing the Git operations, and the students that desired to benefit from Sourcetree utilized the YouTube lectures from the

first iteration. In the last iteration, we got negative comments about the duration of the laboratory session, one of them being:

> *"In my opinion, time shortage was not a meaningful scenario for Git & GitHub lab. It was really stressful to understand and complete all the parts in time. I learned many things about Git and GitHub, however, time shortage made it unnecessarily stressful."*

Our main goal for the laboratory sessions was to allow students practice and learn Git. Considering this, we plan to increase the time duration of the laboratory assignment in the future iterations.

In the first and third iterations, two teaching assistants were guiding the students during the assignment. In contrast, in the second iteration, three teaching assistants supported the students because there were more students attending the training program. Furthermore, in the first iteration, initially, 1.5 hours were given to the students for completing the assignment. However, the students could not finish the assignment in the expected time. Thus, the duration of the assignment was extended to three hours during the laboratory session of the first training program and likewise, in the second and third iterations, three hours of time was given for the assignment.

Due to the pandemic, the laboratory assignment was conducted virtually. The interactivity of the assignment could be enhanced if it could be performed in a real classroom environment. This is because communicating in a virtual environment requires a stable internet connection, students have to wait in virtual waiting rooms, and sharing the desktop with the teaching assistant is time-consuming, whereas, in a classroom, communication is a lot easier and students can state their problems faster. When the pandemic ends, this fact can be evaluated and reflected in the laboratory assignment.

7. Conclusion

In this study, we have demonstrated how we conducted a Git and GitHub training program for third-year undergraduate students in three academic terms as a part of the Object-Oriented Software Engineering course. The motivation for conducting the training program came from the lack of representation regarding the usage of version control systems in Computer Science and Software Engineering curricula. We designed the program to address our learning objectives directly. During the program, we delivered lectures and shared our pre-prepared YouTube videos with the students so that they could gain practical experience. To evaluate the students' knowledge, an interactive laboratory assignment was conducted in a virtual environment. We shared the lecture slides, YouTube tutorials, and assignment materials publicly, making them accessible for the Computer Science education community.

In order to assess the effectiveness of the tutorial and laboratory assignment, pre and post-training surveys were carried out. Comparing the combined survey results in both iterations, in the beginning, 63% of the students used Git previously; however, after the tutorials and the laboratory assignment, 92% of the students stated that they learned Git prominently. The percentage of students who were confident in using Git and GitHub for their term projects rose from 33% to 91%, which indicates that the training program we designed for teaching Git and GitHub was successful.

In the future iterations of the program, we plan to make the laboratory session a collaborative activity to allow students to have an experience closer to real-life scenarios. Moreover, we are planning to expand the scope of the laboratory assignment to include more GitHub operations such as creating pull requests, opening issues, etc.

References

[1] "Stack overflow developer survey 2018," 2018. [Accessed Jan. 13, 2021].

[2] V. Garousi, G. Giray, E. Tüzün, C. Catal, and M. Felderer, "Aligning software engineering education with industrial needs: A meta-analysis," *Journal of Systems and Software*, vol. 156, pp. 65 – 83, 2019.

[3] V. Garousi, G. Giray, and E. Tuzun, "Understanding the knowledge gaps of software engineers: an empirical analysis based on swebok," *ACM Transactions on Computing Education (TOCE)*, vol. 20, no. 1, pp. 1–33, 2019.

[4] K. L. Reid and G. V. Wilson, "Learning by doing: Introducing version control as a way to manage student assignments," in *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '05, (New York, NY, USA), p. 272–276, Association for Computing Machinery, 2005.

[5] V. Isomöttönen and M. Cochez, "Challenges and confusions in learning version control with git," in *International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications*, pp. 178–193, Springer, 2014.

[6] E. Bonakdarian, "Pushing git & github in undergraduate computer science classes," *Journal of Computing Sciences in Colleges*, vol. 32, no. 3, pp. 119–125, 2017.

[7] M. D. Beckman, M. Çetinkaya Rundel, N. J. Horton, C. W. Rundel, A. J. Sullivan, and M. Tackett, "Implementing version control with git and github as a learning objective in statistics and data science courses," *Journal of Statistics Education*, p. 1–35, Nov 2020.

[8] S. P. Linder, D. Abbott, and M. J. Fromberger, "An instructional scaffolding approach to teaching software design," *Journal of Computing Sciences in Colleges*, vol. 21, no. 6, pp. 238–250, 2006.

[9] Y. Liu, E. Stroulia, K. Wong, and D. German, "Using cvs historical information to understand how students develop software," in *International Workshop on Mining Software Repositories (MSR 2004)*, pp. 32–36, 2004.

[10] C. Clifton, L. C. Kaczmarczyk, and M. Mrozek, "Subverting the fundamentals sequence: using version control to enhance course management," *ACM SIGCSE Bulletin*, vol. 39, no. 1, pp. 86–90, 2007.

[11] F. F. Blauw, "The use of git as version control in the south african software engineering classroom," in *2018 IST-Africa Week Conference (IST-Africa)*, pp. Page–1, IEEE, 2018.

[12] D. C. Conner, M. McCarthy, and L. Lambert, "Integrating git into cs1/2," *Journal of Computing Sciences in Colleges*, vol. 35, no. 3, pp. 112–121, 2019.

[13] L. Haaranen and T. Lehtinen, "Teaching git on the side: Version control system as a course platform," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '15, (New York, NY, USA), p. 87–92, Association for Computing Machinery, 2015.

[14] J. Kelleher, "Employing git in the classroom," in *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, pp. 1–4, 2014.

[15] J. Lawrance, S. Jung, and C. Wiseman, "Git on the cloud in the classroom," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, (New York, NY, USA), p. 639–644, Association for Computing Machinery, 2013.