2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion

Student Experiences Using GitHub in Software Engineering Courses: A Case Study

Joseph Feliciano, Margaret-Anne Storey, Alexey Zagalsky
University of Victoria
Victoria, BC, Canada
{noelf, mstorey, alexeyza}@uvic.ca

ABSTRACT

GitHub has been embraced by the software development community as an important social platform for managing software projects and to support collaborative development. More recently, educators have begun to adopt it for hosting course content and student assignments. From our previous research, we found that educators leverage GitHub's collaboration and transparency features to create, reuse and remix course materials, and to encourage student contributions and monitor student activity on assignments and projects. However, our previous research did not consider the student perspective.

In this paper, we present a case study where GitHub is used as a learning platform for two software engineering courses. We gathered student perspectives on how the use of GitHub in their courses might benefit them and to identify the challenges they may face. The findings from our case study indicate that software engineering students do benefit from GitHub's transparent and open workflow. However, students were concerned that since GitHub is not inherently an educational tool, it lacks key features important for education and poses learning and privacy concerns. Our findings provide recommendations for designers on how tools such as GitHub can be used to improve software engineering education, and also point to recommendations for instructors on how to use it more effectively in their courses.

Categories and Subject Descriptors

H.5.3. [**Group and Organization Interfaces**]: Computer-supported cooperative work

Keywords

GitHub, education, learning, software engineering, collaboration

ICSE '16 Companion, May 14-22, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2889160.2889195

1. INTRODUCTION

Learning about computer science and software engineering is challenging for many reasons. While students must learn the theory, they also need to apply what they have learned to practical settings. They learn best by doing, by making mistakes, and by iterating on solutions. And because modern complex systems are not written by developers working in isolation, students need to spend a significant amount of time learning how to collaborate and coordinate with others.

Perhaps in an effort to address some of these challenges, many software engineering and computer science instructors have recently adopted GitHub for hosting their course content or handling assignment submissions. In some cases, they are using it in place of a more traditional learning management environment, such as Moodle. GitHub is a popular social code sharing platform that leverages the Git distributed version control system (DVCS). It encourages an open workflow where collaborators can participate in a number of ways, such as by contributing to discussions regarding bugs and features, or making changes to a project and allowing other collaborators to review and accept the work. Millions of people use GitHub for collaboration, and while it was originally designed for software development, there has been recent uptake in a variety of domains[1].

In a previous study [33], we investigated how and why *educators* in disciplines such as computer science and statistics use GitHub. We found that they use GitHub as a learning platform because of its open workflow and transparency features, and because it offers educators the ability to reuse and remix course materials (through distributed version control), while offering their students the chance to also contribute to course materials (e.g., through *pull requests*).

The educators in our study mentioned several challenges when using GitHub, notably a steep learning curve and the lack of a knowledge base of best practices around how to use GitHub in teaching. We also found that they felt it was important that their students learn how to use GitHub due to its wide use in industry, and that the open and transparent features GitHub provides would support experiential and collaborative learning in a software engineering context. However, in our first study, we neither validated these expectations nor did we gain insights into the student experience.

We now turn our attention to understanding how *students* may benefit or face challenges when using GitHub in a pedagogical software engineering context. We wished to investigate whether GitHub helps students learn more effectively

[1]http://readwrite.com/2013/11/08/seven-ways-to-use-github-that-arent-coding

and if it opens up learning opportunities that the educators from our first study anticipated (e.g., peer review and easier integration of external learning resources). Such insights can help instructors that are struggling to decide if they should use GitHub in their courses, as well as guide instructors on the best ways to incorporate GitHub in an educational context. Furthermore, these insights can be beneficial to GitHub's design team (or to the designers of similar tools, such as GitLab or BitBucket) so that they can improve their tools' suitability as a learning management environment.

In this paper, we present a longitudinal case study whereby GitHub is used as a learning platform for two distinct project-based software engineering courses. We set out to investigate how GitHub impacted student learning in these courses, as well as to determine the benefits and challenges students experienced while using GitHub as part of their studies. We also gained some insights from the instructor that taught both of the courses in our investigation.

The research questions that shaped our study are:

RQ1 How do students **benefit** from using GitHub in their courses?

RQ2 What **challenges** do students face when GitHub is used by software engineering course instructors?

RQ3 What **recommendations** can we give to software engineering instructors who wish to use GitHub for teaching?

2. BACKGROUND

The use of software tools to support learning, teaching, material dissemination, and course management is an important aspect of education, regardless of the domain. In the following, we first provide some background on Learning Management Systems (LMSes) and research on how students learn through contributions. This is followed by a description of GitHub and an overview of research related to how GitHub-like tools have been used in education.

2.1 The Evolution of Learning Management Systems

Traditionally, university educators employ the use of LMSes to manage the courses they teach. An LMS provides students and educators with a set of tools for typical classroom processes. LMSes such as Blackboard, Moodle, and Sakai provide instructors with a variety of features for organizing and administering courses, including file management, grade tracking, assignment hosting, and discussion rooms [22].

In addressing common features in LMSes, Malikowski *et al.* [23] developed a model that distinguishes LMS tool features into five categories: (1) transmitting course content, (2) evaluating students, (3) evaluating courses and instructors, (4) creating class discussions, and (5) creating computer-based instruction. Their research showed that the most prominent use of an LMS is to transmit information to students, whereas features for creating class discussions and evaluating students saw moderate to low-to-moderate usage, respectively.

With the rise of Web 2.0 and 'the social Web', LMSes have become more social and collaborative. For example, Edrees [7] compared the '2.0' tools and features of Moodle and Blackboard, two of the more popular LMSes, noting that they both now include social features such as wikis, blogs, RSS, podcasts, bookmarking, and virtual environments. Despite the popularity of these features, many researchers and educators have expressed concerns about using traditional LMSes to incorporate and emphasize student participation. McLoughlin [25] believes that participatory learning lends itself well to education as students are provided with more learning opportunities where they can connect and learn from each other. However, he noted that while there were signs that Web 2.0 tools could make learning environments more personal, participatory, and collaborative, LMSes tend to be more focused on administration tasks than on the learner. Dalsgaard [6] also pointed out the weaknesses of LMSes for supporting learner-centered activities such as independent work, reflection, construction, and collaboration, arguing that students should be provided with a myriad of other tools to support such activities.

2.2 The Contributing Student

Computer science and software engineering education has started embracing a pedagogy that not only focuses on technical skills, but also on soft skills such as communication and teamwork [18]. One way to develop these skills is to allow students to contribute to each other's learning experiences [13]. This concept, which Hamer calls a Contributing Student Pedagogy (CSP) [14], is formally defined as *"A pedagogy that encourages students to contribute to the learning of others and to value the contributions of others."* It relies on technology to facilitate the learning experience, where the learning tools typically support activities such as peer review, content construction, and solution sharing, amongst others.

There are various characteristics of CSP in practice: (a) the people involved (students and instructors) switch roles from passive to active, (b) there is a focus on student contribution, (c) the quality of contributions is assessed, (d) learning communities develop, and (e) student contributions are facilitated by technology. Falkner and Falkner [8] observe the benefits of incorporating student contributions into their curriculum, such as increased engagement and participation, and the development of critical analysis, collaboration, and problem solving skills—important skills for a computer scientist or engineer.

2.3 GitHub-style Systems in Education

GitHub is one such technology that shows the ability to support certain CSP activities. In particular, it provides several features that aid collaboration and support user contributions. Users can make changes to other people's work in separate repositories or branches, and they can make a *pull request* to invite the original repository owner to review and *merge* their changes into the base version of the software repository. Issue tracking allows contributors to discuss any aspect of a project, including bugs, feature requests, and documentation [1]. Moreover, GitHub's openness and transparency features, which allow users to easily see all activities inside a repository or from a user they're following, fosters both direct and indirect collaboration [5].

Haaranen & Lehtinen [12] conducted a case study where Git and GitLab (an open source platform similar to GitHub) were used in a large computer science course. Students could contribute to the course material by making corrections via *pull requests*. The authors discussed that the ability to perform pull requests is an essential industry skill.

The educators we probed in our previous study [33] described how using GitHub's transparency features in their courses allowed them to encourage student participation. Moreover, *diffs*, issue tracking, and merge requests in GitHub provide support for code reviews [20], a peer-review process that promotes a positive student attitude towards work, as well as training in critical reviewing and communication skills [16].

Kelleher [21] documented his process of using GitHub in the classroom, describing how the transparency of activities alerted him to possible acts of plagiarism and how the integrated issue tracking could be used for annotating code. Griffin & Seals [11] leveraged Git's *branch* and *merge* features to simplify assignments and submissions, however, they felt that GitHub's 'social coding' platform might not suit standard programming assignments that need to remain private. These studies follow early examples of an educational use of other version control tools, such as Concurrent Version Control (CVS) [27] and Subversion [3]. In most of these cases, the tools were used for assignment submission, to simplify the management of courses, and to allow students to collaborate with less effort.

3. METHODOLOGY

Since we previously studied the use of GitHub by educators from various disciplines [33], we turned to investigate student perspectives on the suitability of GitHub for supporting their education. We focused on how students feel their learning benefits from GitHub and the challenges they meet in using such a tool as part of their education. To gain insights on our research questions, we conducted a case study [32, 28] where we drew from multiple sources of evidence—interviews and a survey—to investigate the potential of using GitHub for post-secondary computer science and software engineering courses. The research questions addressed in this work include:

RQ1: How do students benefit from using GitHub in their courses? We've seen evidence that GitHub can benefit educators in a number of ways [33] and that they believe GitHub helps their students. We wished to validate educators' impressions, but also reveal students' perceptions of how GitHub and its workflow may benefit them.

RQ2: What challenges do students face when GitHub is used by software engineering course instructors? When adopting a new tool for a course, particularly a tool not tailored towards education, users may experience friction or a variety of other challenges. We aimed to identify these issues in order to alert educators using GitHub in their courses.

RQ3: What recommendations can we provide software engineering instructors who wish to use GitHub for teaching? There are a variety of ways to use GitHub for development purposes—educators also have many options for using GitHub in their courses. Based on insights obtained from students as well as relevant literature, we provide recommendations to future software engineering instructors wanting to use GitHub to support their courses.

3.1 The Case Study

For this study, we opportunistically sought instructors who could and were willing to try using GitHub. We were fortunate to recruit a university instructor who wanted to try using the tool in two different software engineering courses offered in the same semester: a Distributed Systems (DS) course aimed at both undergraduate and graduate students, and a Software Evolution (SE) course for senior undergraduate students. The DS course covered topics surrounding distributed systems and included concepts such as design considerations, fault tolerance, and cloud computing. The SE course covered the development of large-scale systems and how software evolves due to the many individuals that play a part in developing it over its lifetime.

Both classes were similar in size (29 in SE, 34 in DS) and learning activities (weekly labs, two course projects). The course projects were done either individually or collaboratively with others (in groups of 2-4 students): in the DS course, students built systems that involved multiple computational devices; in the SE course, students evolved existing systems or utilized them to create new ones.

Projects were open-ended and students could choose what they created, what topics they addressed, and what technologies and languages they utilized. Students were required to make their work publicly available so that other people, both inside and outside the course, could view their projects. While not mandated, the overwhelming majority of students opted to use GitHub to host their projects. The instructor did not formally introduce GitHub to the students, so those unfamiliar with the tool had to learn from others or teach themselves.

3.2 How GitHub Was Used During the Case Study

Despite being relatively unfamiliar with GitHub and its features, the course instructor opted to utilize GitHub in the same way for both courses, using its features in three pivotal ways: material dissemination through the course repository, lab work through the 'Issues' feature, and project hosting through student repositories. The advanced use cases other instructors described in our previous study [33], such as utilizing pull requests for assignment submissions, were not used for these courses. The main course instructor was aware of some of these features but was not comfortable using GitHub beyond their knowledge of the tool.

The main use of GitHub was for material dissemination: the instructor hosted a public repository which all students could access to find the work they had to do for any given week. The instructor updated this repository weekly, adding lab assignments, links to readings, and the student homework for the week. All of the content was organized into a calendar-style table made with Markdown and was posted on the home page of the course repository. If students wanted to make changes to the content, they could 'fork' the repository and use a pull request to alert the instructors, although this possibility was not emphasized during the courses.

The other main use of GitHub was for hosting lab content and related discussions. The courses contained labs—a two- to three-hour session once a week—in addition to the regular lectures, which often involved researching a topic and reporting results, or giving other groups feedback on their projects. Using each course repository's 'Issues' page, a dedicated issue was created for each lab, similar to a forum post, and students made comments on the appropriate issue based on their lab work. Students were free to work in groups, and when commenting on an issue, they would *'@mention'* their group members to indicate who they were interacting with.

GitHub was also used for students to host their individual or group project work. Although students were not mandated to use GitHub for their projects (as mentioned above), most work was hosted on GitHub in individual repositories. These repositories were publicly available so others in the course could view the work and give feedback.

In addition to GitHub, the course instructor opted to use a version of the Moodle LMS. Moodle generally allows instructors to make their course content available for students to access and interact with, enable communication between the instructors and students through forums, post quizzes, create wikis for a class to edit, and track student progress and performance. Moodle is a closed environment that is invite-only and, unlike GitHub, has more sophisticated security settings where artifacts can be kept private from individual participants, not just the public. For the courses in our study, Moodle was used for artifacts that the course instructor felt should not be publicly available, including student grades and student responses to the course readings.

3.3 Research Methods

We recruited student participants using a sign-up sheet during the first week of each course. Participation was voluntary and students who signed up were not required to participate in all phases of the study—those who were interviewed did not necessarily respond to the subsequent validation survey (discussed below).

The first phase of the study consisted of interviews with the students. Most interviews were one-on-one, however, due to scheduling reasons, some students requested to be interviewed in groups of two or three. Interviews lasted 20-30 minutes and were conducted face to face in a meeting room. Audio from each interview was recorded with participant consent and notes were taken for reference. The interviews were semi-structured based on 12 guiding questions[2] and we probed further with additional questions (as needed) to gain more insights. We also interviewed the course instructor at the end of the semester to further understand how GitHub supported their teaching. This fit the exploratory nature of our research questions and helped us discover interesting insights.

In a second phase, we conducted a survey with the students to validate our findings and to confirm or contradict the themes that emerged from our analysis of the interview data in phase one.

3.4 Data Collection and Analysis

We conducted interviews with 12 students from the SE course, 6 students from the DS course, and 1 student who was taking both courses, for a total of 19 interviews. To give students sufficient experience with GitHub, these interviews occurred no earlier than 7 weeks after the start of the semester and all were concluded within 5 weeks.

The main distinction between the two courses was that SE was an undergraduate course, whereas DS had a mix of undergraduate and graduate students. Otherwise, the courses were laid out in a similar manner (as outlined in Section 3.2). Table 1 summarizes the previous experience the interviewed students had with GitHub.

We transcribed every interview and coded the responses using a content analysis methodology [2]. We labeled segments according to the research questions of the study, and from these codes, we iteratively identified themes and concepts that surfaced multiple times. After grouping the themes into well-defined categories, we compiled a final list of themes. To check for biases, the coding of the interviews was reviewed by another researcher. To reduce possible biases, as themes emerged, we also searched for and reported counter examples to the findings (some of these counter examples are discussed in a thesis due to space constraints in this paper [9]). Furthermore, the themes that emerged from the interviews with students were validated through the survey and the interviews with the instructor.

[2]https://goo.gl/oszl17

Table 1: Participants and their prior experience with GitHub.

ID	Prior GitHub Experience	Degree Type
DS1	Inexperienced	Graduate
DS2	Used Academically, Professionally	Graduate
DS3	Used Academically, Professionally	Graduate
DS4	Inexperienced	Graduate
DS5	Used Academically	Graduate
DS6	Used Academically	Graduate
SE1	Used Academically, Professionally	Undergraduate
SE2	Inexperienced	Undergraduate
SE3	Used Professionally	Undergraduate
SE4	Inexperienced	Undergraduate
SE5	Used Personally	Undergraduate
SE6	Used Academically	Undergraduate
SE7	Used Professionally	Undergraduate
SE8	Inexperienced	Undergraduate
SE9	Used Professionally	Undergraduate
SE10	Used Casually	Undergraduate
SE11	Used Professionally	Undergraduate
SE12	Used Academically	Undergraduate
SE13	Used Academically	Undergraduate

The validation survey in the second phase of our research was distributed during the final lab session of each course. Students were asked to anonymously fill out an online form with details about their experiences. As mentioned above, survey respondents did not necessarily participate in the interview phase. We received 18 student responses from the DS course (4 of which were interviewed) and 15 responses from the SE course (9 of which were interviewed), for a total of 33 responses.

3.5 Limitations

There are a number of limitations with our research design, which we describe here.

Our decision to recruit an instructor that had not used GitHub before (either for other work or for teaching) may have influenced (negatively) the experience of the students we wished to study. We made this decision as our previous research on this topic had gathered perspectives from instructors that generally knew GitHub quite well.

The recruitment methods we used to engage students may have resulted in biased opinions: the students that were willing to be interviewed or surveyed may have been students who felt strongly about GitHub (positively or negatively), whereas those without a strong opinion may have chosen not to participate. By interviewing many students, we were able to uncover contradictions or discrepancies between the opinions we gathered. We also tried to offset this limitation

by including a validation survey in our design, which had a reasonable response rate of 53%.

We also recognize that the questions we asked in the interviews and in the surveys may have been biased. In an attempt to limit this as much as possible, we piloted the questions and had members outside our team review them for biases.

When we coded the interviews, only a single coder was involved, but we had an expert reviewer question the themes that emerged and review the coding for potential biases and inconsistencies. We further validated the emergent themes by interviewing the instructor and by hosting a validation survey. Finally, we reported discrepant information—in the interviews and between the interview and survey data—to illustrate that not all students shared the same opinions.

Finally, we recognize that the findings from this single case study cannot be generalized to other settings [28] where GitHub may be used in software engineering courses. We tried to offset this by asking the instructor to use the tool in two different courses. One particular issue that we alluded to above was the instructor's lack of experience with GitHub. This may have influenced our results, but in a negative way as the instructor was not able to give more guidance to the students using the tool. However, many of the findings we report are reflected in other studies that use similar tools for classes, such as Kelleher's study on Git and GitHub [21], and Haaranen and Lehtinen's study on Git and GitLab [12].

4. FINDINGS

We present the findings according to our research questions and highlight the main themes that emerged alongside representative quotes from the interviews.

4.1 Student Benefits of Using GitHub for Software Engineering Courses

Previous work [33] has shown that GitHub introduces numerous and sometimes surprising advantages in the context of learning and teaching, however, it is important to consider how this is perceived by the students involved. Our study of the student perspective revealed several key benefits that students experience when they use GitHub as a tool for managing education.

Benefit: Gaining and Demonstrating Industry-Relevant Skills and Practices

To succeed in modern software development, students need to be familiar with best practices (e.g., peer review, cross-team collaboration) and commonly used tools (e.g., continuous integration tools, distributed version control systems). Many of the interviewees [SE2, SE3, SE4, SE5, SE6, SE7, SE8, SE11, SE13, DS4] mentioned that using GitHub in their courses provided a good introduction to the tool and to relevant practices: *"I think when you go and work in software development too, you should get used to [having] lots of eyes being all over your work; that's just the way it's gonna be, so it's practice before real life."* [SE8]

Despite previous experience with the tool, using GitHub in their courses introduced some students to specific features that they were not necessarily aware of but felt were important to learn: *"This is the first time I've actually used the Issues portion of GitHub."* [SE13] Even with the most basic use of GitHub in a course (material dissemination), students were able to reap the benefit of being exposed to a tool widely used in industry.

Another important aspect of the use of GitHub is the current concept of mutual assessment [30] and the ability to use the tool as part of one's portfolio. Interviewees [SE5, SE6, SE7, SE8, SE11, SE13, DS3, DS4] mentioned the importance of publicly presenting their work on GitHub. It is not uncommon for employers nowadays to refer to GitHub for hiring purposes[3]. In fact, some of the students we interviewed had already been contacted by potential employers who wanted to view their GitHub accounts: *"I think all three companies that I applied to this semester wanted me to link to my GitHub. So I was really lucky that I had [a class] project on there. And I think when this [course's] project is done too, it'll also be really nice to have up there, after we clean it up."* [SE6]

Benefit: Enabling Cross-Team Collaboration and Contributions

Requiring students to host their projects publicly and the heavy reliance on GitHub in the course resulted in students looking at and contributing to the work of others. Students provided feedback and received suggestions from others [SE2, SE3, SE5, SE7, SE10, SE11, SE12, SE13]. And while some lab assignments required students to look at the work of other students, many students reported that they would often peruse other projects outside of the requirements. In a few cases, students utilized code that other groups built, which resulted in them discovering and fixing issues in the original code: *"I believe that one other group decided for project 2 to use [our project 1] and they made a couple of pull requests I think."* [SE10]

By facilitating student contributions and cross-team collaboration, the use of GitHub enabled a participatory culture [19] where students openly created and remixed content and felt their contributions mattered. As a result, GitHub encouraged peer review practices among students: *"I thought [peer reviews] was the best way to learn actually . . . It forced you to put yourself in a position where you have to defend what you did, which I think is good for quality because you have to actually care."* [SE11] The use of GitHub also provided collaborators in group projects an easy way to track and keep up with each other's work: *"You can see exactly what the other person has contributed, and you can look it up again a month later . . . then it's a good way to keep accountable. And it's good for yourself too, because you know they can see your work, so you wanna make sure that it's top notch and easily readable."* [SE5]

Benefit: Encouraging Student Contributions to Course Content

One of the benefits that GitHub offers over traditional Learning Management Systems is the ability for students to suggest corrections and make changes to course materials via pull requests (PRs). This is a simple change mechanism that provides students with a degree of autonomy, but that instructors can easily accept, reject, or request resubmission.

Throughout the two courses in this case study, three PRs were submitted to make corrections to the course materials or to add links to new materials. These PRs were submitted during the first month of each course and by a single student (SE1) that had previous experience with GitHub (the student was registered in both courses). SE1 commented : *"I like being able to fix the mistakes that [the course instructor] might make, like with a bad link or something, by making a*

[3]http://www.cnet.com/news/forget-linkedin-companies-turn-to-github-to-find-tech-talent

PR... because it makes me feel a little more involved."

It is interesting to note that this style of student contribution didn't continue. A few of the interviewees [SE1, DS2, course instructor] reported that the PRs were not merged *"quickly enough"*—the students expected an immediate response (due to a deadline), but the instructor did not merge them for a day or two. Nonetheless, students [SE1, SE3, SE5, SE6, SE10, SE13, DS2, DS4] felt that contributing to the class materials could have been a useful exercise had they taken advantage of it or had the possibility to do so been more advertised.

Benefit: Breaking Down the Walled Garden
Work hosted on GitHub is often publicly available for others to see and contribute to, allowing interactions with external entities (i.e., practitioners and experts). Our study revealed a case where a student [SE1] invited community engagement: the student was an active member in the Rust programming language community and advertised their course work to that community. As a result, members of the community tried to help with the project in multiple ways: *"So here I have people involved in the discussion. These are just people in the community I've been talking to about how to do different things, and they've been giving me suggestions. And that's really cool because I actually have some community involvement in my course project."* [SE1] And while considered an outlier, this example illustrates the ease of supporting external interactions and the potential of tapping into new sources of knowledge, effectively breaking the *"walled garden"* imposed by traditional LMSes [26].

Benefit: Version Controlled Assignments
Some students [SE8, DS3] noted the possibility of using GitHub's history and version control mechanism for providing continuous and constructive feedback: *"You'd see all the mistakes [the student] made getting there, too, which is just as important to learning as the finished product."* [DS3] While this feature was not utilized in the courses we studied, students saw how it provides the ability to examine the final product and the process used to get there.

4.2 The Challenges Students Face When Using GitHub for Software Engineering Courses

Our case study provided a glimpse into the challenges students may face when using GitHub for software engineering courses.

Challenge: The Perils of Public Projects
Having the course materials and student work open to the outside world can be a double-edged sword. As previously mentioned, it allows students to benefit from interactions with people external to the project. However, students had concerns about this that should not be ignored.

The students we interviewed didn't mind that the class repository and their project work was publicly shared on GitHub. However, several students [SE1, SE4, SE5, SE6, SE7, SE10, SE13, DS3, DS6] recognized the potential problems that might surface from publishing their work publicly. They mentioned two possible issues: 1) their school work may not be of interest to the public, and 2) their work may not be at a level they feel comfortable sharing, especially when colleagues and potential employers may see it. *"It would actually be nice if [our projects] were separate or private somehow so I wouldn't have to go through everything and sanitize all the stuff I've submitted, because for as much as you'd want to think you're putting 100% into it, you're not really."* [SE6]

Not surprisingly, students found a workaround to some of these privacy issues: students do not have to attach their full names to their work on GitHub, or they can use a separate GitHub account. *"You can decide that on your own, depending on if you use your main git account or just make a separate git account for your class."* [SE3] Our case study revealed a student had created a new GitHub account solely for their contributions in class. Unfortunately, they refused to be interviewed. It is important to note that not all the interviewees shared this concern, and a few [SE5, DS2, DS5] didn't have any issues sharing their work publicly.

Challenge: Unfamiliarity with Git and GitHub
When asked about challenges, students mentioned how an unfamiliarity with Git and GitHub caused difficulties and missed opportunities in taking advantage of existing features (e.g., pull requests, GitHub issues) [SE1, SE2, SE3, SE4, SE5, SE6, SE9, SE11, SE12, SE13, DS1, DS3, DS5].

Furthermore, the course instructor was inexperienced with using GitHub, which made it difficult to educate the students on its features and caused frustration for some of the interviewees. Students [SE6, DS1] suggested that an introduction lecture dedicated to learning how to use GitHub at the beginning of the course would have been helpful. Others [SE1, SE2, SE3] proposed that this challenge could have been alleviated by a greater focus on version control and version control tools earlier in the curriculum.

Challenge: Notification Overload
Another challenge lies in how GitHub handles notifications and one's level of familiarity with the possible options. GitHub's mechanism for managing notifications is the 'Watch' functionality, where for every repository there are three options:

- **Not watching**: The user receives notifications only when participating or when their username is mentioned.
- **Watching**: The user receives notifications on all conversations and activities (e.g., commits, PRs, comments on issues).
- **Ignoring**: The user does not receive any notifications.

Each user can also control whether the notifications are shown in the tool itself, sent by email, or both.

Unless students were 'watching' the repository, they did not receive email notifications for any activities except when they were specifically mentioned. However, when the students did 'watch' the repository, they received an influx of notifications for every user comment, which became overwhelming. SE10 shared that they were engaged less in the activities of others because of the noise from notifications: *"It sent me a million emails, both of [the tools] actually. I should have just turned that off, but I was worried about missing something. Because every time someone would post, you would get another email ... I actually did not read anyone else's feedback because it was just so many emails, to be totally honest."* [SE10]

Challenge: GitHub is Not Designed for Education
Students [SE2, SE5, SE6, SE7, SE9, SE11, SE12, SE13, DS4] described one drawback from using GitHub for education: GitHub is simply not built for it. Those that mentioned this particular issue acknowledged that although there may

be workarounds for many of the tasks required in education, GitHub certainly struggles to meet some basic needs, such as gradebooks and a formal assignment submission feature. This was one reason why the course instructor for this study decided to use a customized version of Moodle in conjunction with GitHub—to ensure privacy for matters such as grades, or to make announcements, something that would be too cumbersome to do in GitHub.

This issue potentially hinders some of the benefits listed above. For example, it is more difficult to contribute to course content because GitHub's *diffs* feature does not support file types commonly used in education (such as PDFs and PowerPoint presentations): *"I think one drawback of GitHub is that you cannot actually see the diff [of] commonly used files such as PPTs or PDFs, so you can't really use it for correcting professor's slides, or PDFs."* [SE12] The pull request process then requires an extra step, which some students felt would discourage them from using the feature: *"I think for readme files, it's a lot easier to edit, cause you can edit directly in GitHub. But for other files, you'll probably have to change and make a branch and then commit it and then send a PR, it might actually be more work."* [SE13]

4.3 Recommendations for Software Engineering Instructors

A main challenge educators face is the lack of a shared knowledge base [33] on how to use GitHub to support learning and teaching. Some educators have shared their experiences and recommendations with others, either as personal blog posts or as part of a discussion[4]. Additionally, GitHub provides basic guidelines[5] for setting up an organization for a class, and has developed a command-line tool[6] to help set up a class repository. Contributing to these resources helps build a common knowledge base for instructors to share with and learn from. As a main contribution of this study, we illustrate the recommendations to instructors and educational tool designers below.

Recommendation: Promote the Desired Workflow
As discussed earlier, students raised concerns regarding the public nature of the work they host on GitHub. As way to alleviate this concern, educators can promote the option of using pseudonyms and alternate accounts for the course. Students who were more experienced with GitHub also mentioned that some of GitHub's collaborative features should have been further utilized to provide additional benefits [SE3, SE5, SE6, SE7, SE11, DS2, DS3, DS4, DS6]. They felt there was little reason to use GitHub for a course if it was only used for material dissemination. They also noted that while there's potential, the unidirectional nature of the work being performed meant that potential benefits were not realized. *"If there was a way to collaborate on the material, that would be useful . . . But in this class, every one of our labs so far has been demo to the lab TA, so nothing is going back to GitHub . . . Maybe if we were submitting things to it, maybe that would be helpful. I can see how it could be useful, it's just that in our usage it's not really adding anything to the experience."* [DS3]

Consequently, students [SE3, SE5, SE6, SE7, SE11, DS2, DS3, DS4, DS6] stressed the importance of defining a workflow when using GitHub in a course and then advertising the desired workflow to the class. For example, promoting the use of pull requests as a way to suggest corrections to course materials, or encouraging student-to-student feedback and contributions: *"I think it would've needed to have been advertised more that [the instructor] was looking for input on things, and if [the instructor] said that, maybe more people would have [contributed] to maybe propose extensions for assignments or something."* [SE7] However, merely promoting the workflow is not enough. GitHub is a powerful framework, but it is up to the instructor to consider which features they would like to use and how (e.g., PRs for assignment submission), and use those features thoroughly and consistently (e.g., merging PRs more quickly).

Recommendation: Familiarize Yourself with GitHub
Several students questioned the possibility of using the same workflow for all courses, however, GitHub actually provides mechanisms to specify the level of privacy depending on the workflow used. For example, if educators wish to have the course materials and assignments viewable by only their students, they can create an organization on GitHub just for the course.

In courses where student plagiarism is a concern, educators can use private repositories while still taking advantage of the other benefits that GitHub offers. In order to set up an appropriate workflow, educators can either create private repositories for their students or ask the students to do so themselves. This workflow requires students to submit assignments through pull requests.

Recommendation: Use Supported Formats
To simplify the contribution process, we recommend the use of plain text file formats (e.g., Markdown, iPython documents) in order to take advantage of the *diff* and line commenting functionalities.

4.4 Validation Survey

To validate the themes that emerged from the interviews, we sent a survey to all of the students in both courses at the end of the term (during the last week of the course). 33 students (18/34 DS and 15/29 SE) responded to the survey, giving us a response rate of 53%. The survey consisted of a set of Likert scale-style questions that were designed and pilot tested to confirm or refute our findings concerning the benefits and challenges students experienced.

Figure 1 provides a summary of the main questions we asked in the survey, as well as the responses we received. As there were few differences between responses from the two different courses, we aggregate the responses shown in the figure (due to space limitations), but we discuss any notable differences between the two courses below.

In general, the survey validated the benefits and challenges that emerged from the interviews. Of note is that 30 students (of the 33 that responded across both courses) agreed that they would **continue using GitHub** for group and individual work after the course concluded. Given that 14 of these students were completely or somewhat unfamiliar with GitHub before the course began, students seemed to believe that using GitHub can be beneficial for them outside of courses. The majority of the students that responded also agreed that Git, GitHub, or other DVCSes should play a bigger role in their future courses and curriculum (20 agreed, 9 were neutral, and 4 disagreed).

[4]https://github.com/education/teachers/issues
[5]https://education.github.com/guide
[6]https://classroom.github.com

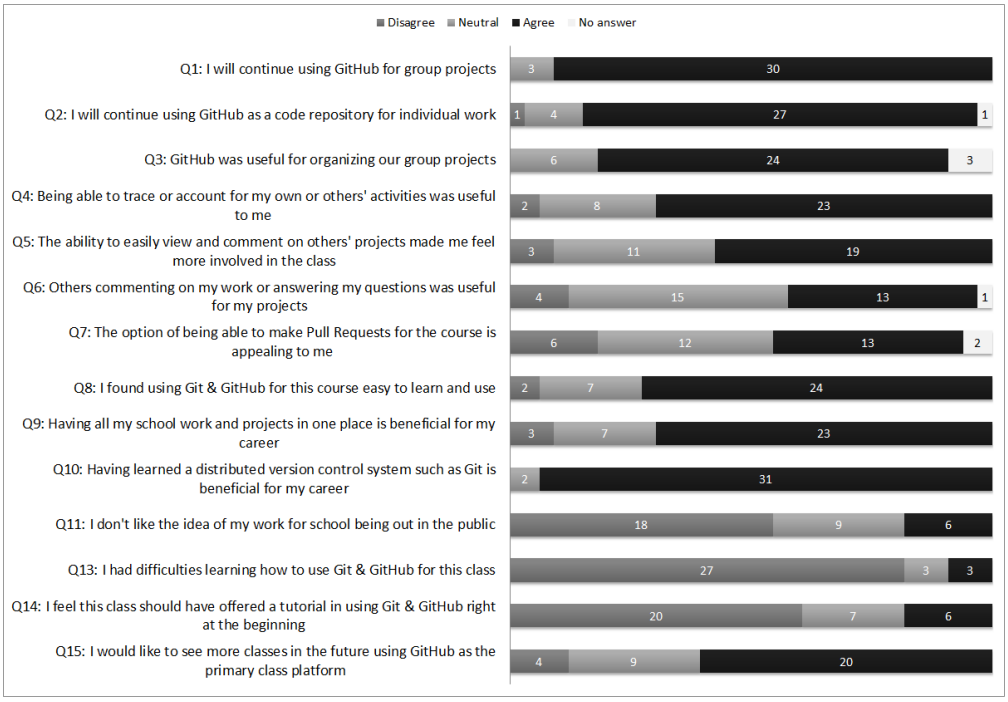

Figure 1: Aggregate of responses to validation survey questions

In total, 19/33 students across both courses agreed that they were **more involved in the class** from viewing and commenting on other projects hosted on GitHub. However, 11 of the 15 SE respondents agreed, compared to only 8 of the 18 DS students who agreed. This small difference may be due to the different levels of collaboration required across the two different courses, as the SE course required more collaboration in their lab assignments.

The survey also pointed to some divergences with the findings from the interviews. From the interview responses, we expected that **privacy** was going to be a major concern to many students. It was surprising to us that most students that responded to the survey from both courses disagreed or were neutral with the suggestion that their school work should not be publicly available. This may be because the survey was conducted at the end of the term when their work was more polished and perhaps ready to be viewed by others. Future work should consider this further.

Also, from the interviews, we expected that students would feel that a **tutorial** was necessary at the beginning of the course as they said they had trouble learning GitHub. But in the survey, 20/33 (10 students from each course) disagreed that the classes needed a tutorial in the beginning of the semester. This may have been because by the end of the term, students realized they could learn GitHub without such support.

5. DISCUSSION

The motivation behind this study was to uncover student perceptions based on their experiences using GitHub as an educational tool. We noted that GitHub was used in three main ways: (a) as a place to disseminate material and host class schedules, (b) as a place for students to submit and discuss their lab assignments, and (c) as a place where most of our interviewees hosted their course projects, either collaboratively or alone.

GitHub Supports the Contributing Student

At a basic level, GitHub can provide similar functions to those of traditional Learning Management Systems. It allows for many of the activities found in Malikowski *et al.*'s model of features found in a traditional LMSes [23]. However, in the courses we investigated, GitHub was only used to support two of the primary uses from this model: information transmission and class discussions. And even though GitHub can serve a purpose similar to formal educational tools, it is important to note that it was not designed for education: GitHub is a framework, and therefore, it lacks features tailored specifically for education (e.g., grading features).

GitHub does excel by creating a culture of participation [19] and providing opportunities for students to participate in their learning. Students are able to openly contribute to course materials by making changes or additions directly to the course repository. This type of action plays a key role in Collis and Moonen's concept of a 'Contributing Student' [4] as GitHub provides students the ability to drive their coursework. Moreover, GitHub's collaborative features simplify many of the 'Contributing Student Pedagogy' activities [15], including peer reviews, discussions, content construction, solution sharing, and making links. Although only a few students contributed to the course materials in our study, many felt that this activity would have seen more use had it been advertised more.

When assignments and projects are public, GitHub provides people the opportunity to contribute to other students' learning with tools that allow them to easily provide direct feedback on assignments or project work. A number of groups in our study left feedback for other groups when they noted bugs or issues in the code, and students appreciated the ability to see others' work and provide feedback as they saw fit. Contributing to other students' work also

helps refine soft skills such as communication and teamwork [13]. Additionally, re-purposing or remixing code requires students to show a deeper understanding of the material and the code involved [29]. An instructor may also utilize GitHub to provide opportunities for students to peer review or grade each other's work, which can help develop important analysis and evaluation skills [31].

Transparency of Activities

In describing the benefits of using GitHub to support their group projects, some students felt that the ability to see others' work encouraged collaboration. Students also acknowledged the importance of seeing the history of work from other group members, describing the feature as a way to hold people accountable and stay up to date with the work, gaining awareness that can be important in collaborative learning [17]. This is in line with the benefits related to GitHub use in industry [5].

Some students felt that GitHub's transparency could lead to better grading methods, despite these courses not utilizing the tool for grading. Compared to the traditional way of submitting assignments, where an assignment is handed in as a complete product when it is due, GitHub offers instructors the opportunity to monitor assignments and projects, giving feedback while they are in progress—a useful exercise for both parties [10].

Beyond the Course

Supporting our findings from interviews with early adopters of GitHub in education [33], most of the students interviewed described being exposed to GitHub and its features during a course as a benefit—the exposure to GitHub's open and collaborative workflow may result in transferable skills for their careers. Moreover, the popularity of the tool means that students' GitHub accounts become part of their online presence [24], which may serve an important role with future collaborators or potential employers who use GitHub for hiring purposes.

With GitHub's popularity, many developers are putting their code on the platform, both publicly and privately. When a course is publicly visible, the 'walled garden' that traditional LMSes tend to suffer from [26] can be overcome. For example, student projects can involve people from another community, or outsiders can contribute to course materials in some manner.

6. CONCLUSIONS AND FUTURE WORK

In this work, we conducted a case study to investigate student perceptions of using GitHub in an educational context. From our findings, we uncovered ways in which GitHub benefited students in the project-based courses we investigated, as well as discovered what students struggled with. More importantly, however, we extracted some ways in which GitHub has the potential to support their learning, even further than what they experienced in these two courses. Beyond the exposure to a popular industry tool, GitHub simplifies the process in which students can participate in a number of activities, such as reviewing each other's work, contributing work or helpful comments to each other, and making changes or suggestions to course materials.

An important consideration from this work relates to the future of tools for computer science and software engineering education: what's next? First, we consider the importance of participation, group work, and group learning for students in technical fields in order to develop non-technical 'soft' skills such as communication and teamwork [18]. This work demonstrates how using GitHub can unlock activities where students can contribute to each other's learning, and as a result, we believe it can be beneficial for current and future tools focused on learning to add support for the open, collaborative workflow GitHub encourages.

As such, one possible path is the 'GitHub for Education' Greg Wilson discussed[7], where a tool like GitHub can be altered or built to be more focused on education. The main weakness of GitHub when used in this context is in the lack of flexibility in its privacy and in the lack of administrative functions such as gradebooks and announcements. Meanwhile, there are open-source alternatives to GitHub, such as GitLab, that could be further developed into a tool that fulfills more educational needs. As an example, it could be valuable to implement a form of announcements, a notification feature that students have more control over, and a way to make some discussions or files within a repository private while others remain public. This a potential avenue for future work, where such a tool can be evaluated.

This work is just the beginning of learning more about how GitHub can benefit students and educators, what challenges people face when GitHub is used for this purpose, and how these challenges might be addressed. Possible directions for future research include investigating the effectiveness of GitHub's open workflow in large-scale courses and a deeper investigation on whether GitHub can properly support education outside of the software engineering domain.

7. REFERENCES

[1] T. F. Bissyande, D. Lo, L. Jiang, L. Reveillere, J. Klein, and Y. Le Traon. Got issues? who cares about it? a large scale investigation of issue trackers from github. In *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, pages 188–197. IEEE, 2013.

[2] K. Charmaz. Constructing grounded theory: A practical guide through qualitative analysis (introducing qualitative methods series). 2006.

[3] C. Clifton, L. C. Kaczmarczyk, and M. Mrozek. Subverting the fundamentals sequence: using version control to enhance course management. In *ACM SIGCSE Bulletin*, volume 39, pages 86–90. ACM, 2007.

[4] B. Collis and J. Moonen. The contributing student: Learners as co-developers of learning resources for reuse in web environments. In *Engaged learning with emerging technologies*, pages 49–67. Springer, 2006.

[5] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

[6] C. Dalsgaard. Social software: E-learning beyond learning management systems. *European Journal of Open, Distance and E-Learning*, 2006(2), 2006.

[7] M. E. Edrees. elearning 2.0: Learning management systems readiness. In *e-Learning" Best Practices in Management, Design and Development of e-Courses:*

[7] http://software-carpentry.org/blog/2011/12/fork-merge-and-share.html

Standards of Excellence and Creativity", 2013 Fourth International Conference on, pages 90–96. IEEE, 2013.

[8] K. Falkner and N. J. Falkner. Supporting and structuring âĂIJcontributing student pedagogyâĂİ in computer science curricula. *Computer Science Education*, 22(4):413–443, 2012.

[9] J. Feliciano. Towards a collaborative learning platform: The use of github in computer science and software engineering courses. Master's thesis, University of Victoria, 2015.

[10] L. Glassy. Using version control to observe student software development processes. *Journal of Computing Sciences in Colleges*, 21(3):99–106, 2006.

[11] T. Griffin and S. Seals. Github in the classroom: Not just for group projects. *Journal of Computing Sciences in Colleges*, 28(4):74–74, 2013.

[12] 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*, pages 87–92. ACM, 2015.

[13] J. Hamer. Some experiences with the contributing student approach. *ACM SIGCSE Bulletin*, 38(3):68–72, 2006.

[14] J. Hamer, Q. Cutts, J. Jackova, A. Luxton-Reilly, R. McCartney, H. Purchase, C. Riedesel, M. Saeli, K. Sanders, and J. Sheard. Contributing student pedagogy. *ACM SIGCSE Bulletin*, 40(4):194–212, 2008.

[15] J. Hamer, A. Luxton-Reilly, H. C. Purchase, and J. Sheard. Tools for contributing student learning. *ACM Inroads*, 2(2):78–91, 2011.

[16] C. D. Hundhausen, A. Agrawal, and P. Agarwal. Talking about code: Integrating pedagogical code reviews into early computing courses. *ACM Transactions on Computing Education (TOCE)*, 13(3):14, 2013.

[17] J. Janssen and D. Bodemer. Coordinated computer-supported collaborative learning: Awareness and awareness tools. *Educational Psychologist*, 48(1):40–55, 2013.

[18] M. Jazayeri. The education of a software engineer. In *Proceedings of the 19th IEEE international conference on Automated software engineering*, pages 18–xxvii. IEEE Computer Society, 2004.

[19] H. Jenkins. *Confronting the challenges of participatory culture: Media education for the 21st century*. Mit Press, 2009.

[20] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian. The promises and perils of mining github. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 92–101. ACM, 2014.

[21] J. Kelleher. Employing git in the classroom. In *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, pages 1–4. IEEE, 2014.

[22] S. Kumar, A. K. Gankotiya, and K. Dutta. A comparative study of moodle with other e-learning systems. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 5, pages 414–418. IEEE, 2011.

[23] S. R. Malikowski, M. E. Thompson, and J. G. Theis. A model for research into course management systems: Bridging technology and learning theory. *Journal of educational computing research*, 36(2):149–173, 2007.

[24] J. Marlow, L. Dabbish, and J. Herbsleb. Impression formation in online peer production: activity traces and personal profiles in github. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 117–128. ACM, 2013.

[25] C. McLoughlin and M. J. Lee. Social software and participatory learning: Pedagogical choices with technology affordances in the web 2.0 era. In *ICT: Providing choices for learners and learning. Proc. ascilite Singapore 2007*, pages 664–675, 2007.

[26] J. Mott. Envisioning the post-lms era: The open learning network. *Educause Quarterly*, 33(1):1–9, 2010.

[27] K. L. Reid and G. V. Wilson. Learning by doing: introducing version control as a way to manage student assignments. *ACM SIGCSE Bulletin*, 37(1):272–276, 2005.

[28] P. Runeson, M. Host, A. Rainer, and B. Regnell. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[29] J. Sant. Code repurposing as an assessment tool. In *Proc. of the 37th International Conference on Software Engineering-Volume 2*, pages 295–298. IEEE Press, 2015.

[30] L. Singer, F. Figueira Filho, B. Cleary, C. Treude, M.-A. Storey, and K. Schneider. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 103–116, New York, NY, USA, 2013. ACM.

[31] H. Søndergaard and R. A. Mulder. Collaborative learning through formative peer review: pedagogy, programs and potential. *Computer Science Education*, 22(4):343–367, 2012.

[32] R. K. Yin. *Case study research: Design and methods*. Sage publications, 2013.

[33] A. Zagalsky, J. Feliciano, M.-A. Storey, Y. Zhao, and W. Wang. The emergence of github as a collaborative platform for education. In *Proc. of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 1906–1917. ACM, 2015.