A Case Study of Using GitHub in Software Engineering Courses

Ben Trovato*
Institute for Clarity in Documentation
1932 Wallamaloo Lane
Wallamaloo, New Zealand
trovato@corporation.com

G.K.M. Tobin†
Institute for Clarity in Documentation
P.O. Box 1212
Dublin, Ohio 43017-6221
webmaster@marysville-ohio.com

Lars Thørväld‡
The Thørväld Group
1 Thørväld Circle
Hekla, Iceland
larst@affiliation.org

Lawrence P. Leipuner
Brookhaven Laboratories
Brookhaven National Lab
P.O. Box 5000
lleipuner@researchlabs.org

Sean Fogarty
NASA Ames Research Center
Moffett Field
California 94035
fogartys@amesres.org

Charles Palmer
Palmer Research Laboratories
8600 Datapoint Drive
San Antonio, Texas 78229
cpalmer@prl.com

ABSTRACT

This paper provides a sample of a LaTeX document which conforms to the formatting guidelines for ACM SIG Proceedings. It complements the document *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

*Dr. Trovato insisted his name be first.
†The secretary disavows any knowledge of this author's actions.
‡This author is the one who did all the really hard work.

Keywords

ACM proceedings, LaTeX, text tagging

1. INTRODUCTION

The use of version control systems has become increasingly commonplace in software development. Distributed version control systems (DVCS) in particular are typically used in collaboration and to interact with the global software development community. As such, using DVCS has become an essential skill for software engineers. However, the integration of DVCS into computing education as a way of teaching students these skills has so far gained little attention. In this study, we discover the implications of using a distributed version control system as a course platform.

Computer science and software engineering education has increasingly focused on the development of not just technical skills, but also soft skills such as communication and teamwork [**?**]. One such way to develop these skills is to allow students to contribute to each other's learning experiences and to course materials [**?**]. This concept, called 'Contributing Student Pedagogy' [**?**], relies highly on the technology surrounding the learning experience: the e-learning tool utilized to facilitate interactions between course participants and the learning materials.

GitHub is a social code sharing service and version control system. It is a popular tool for many groups and projects that require collaboration, and has even seen utilization in areas outside software development, such as technical writing [1]. The advantages of GitHub and similar tools include their awareness and transparency features, where collaborators can easily stay informed of others' work [**?**]. As well, collaborators in a GitHub repository can be involved in a project in a number of ways, such as contributing to discussions regarding bugs and features, or making changes to a project itself and allowing other collaborators to review and accept their changes. This open, collaborative work-

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

flow is called 'The GitHub Way'[2] as these features are not necessarily exclusive to GitHub and can be found in other Distributed Version Control Systems (DVCSes) such as BitBucket. In a previous study [?], we interviewed instructors who were early adopters of using GitHub as a learning platform and extracted their motivations and what benefits and challenges they encountered. It is also important, however, to explore the student perspective and to discern how the use of GitHub as an educational tool might affect students.

This work presents a case study whereupon GitHub was used as an e-learning tool for project-based, university courses, particularly focusing on the collaborative and contributive activities GitHub could enable students to partake in. The work is largely exploratory in order to discern how GitHub can impact learning, including the potential benefits and possible challenges students meet or expect to meet with GitHub's use in this context. While the instructors and students reaped many benefits from using GitHub in these courses, there were drawbacks and challenges with using a tool not built for education. Conclusions from this work can determine the viability of GitHub as an educational tool, or can help shape the development of future educationally-focused tools which, similar to GitHub, gives students the opportunity to contribute to the learning experience in multiple ways. To do so, we interviewed the course participants (the students and the teaching team) regarding their experiences with using GitHub in their course to gather their perspectives on whether or not the GitHub workflow can support computer science courses.

Our main contributions include the evaluation of using a DVCS such as GitHub for computing courses from the point of view of both students and the teaching team. As well, we extract suggestions for ways to use GitHub as a learning platform to maximize the benefits of its use in an educational context.

We conducted interviews with the students and the teaching team involved in the courses selected for these cases and followed up with a survey to validate our findings. Our findings indicate that students find that GitHub allows them to partake in activities not typically available in their typical courses, such as being able to easily contribute to each other's work in various ways. However, students expressed concerns regarding the public nature in which their work was hosted on GitHub for this case study.

2. BACKGROUND

Regardless of the field, the use of software tools to support learning, teaching, material dissemination, and course management is an important aspect of education. Traditionally, university educators employ the use of learning management systems (LMSes) to manage the courses they teach. LMSes, such as Blackboard, Moodle, and Sakai, give instructors a variety of features for managing courses, such as file management, grade tracking, assignment hosting, and chat [?]. The use of an LMS provides students and educators with a set of tools for typical classroom processes. [?] developed a model that dissects the quality of LMS tools 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 study shows that the most prominent use of an LMS is to transmit information to students, whereas the categories of creating class discussions and evaluating students receive moderate and low-to-moderate use, respectively.

With the rise of the 'Web 2.0' and the social web, developers began incorporating these technologies into Learning Management Systems to account for more social approaches. Edrees, for example, [?], compares the '2.0' tools and features of Moodle and Blackboard, two of the more popular LMSes, identifying that they both added features to become more social such as wikis, blogs, RSS, podcasts, bookmarking, and virtual environments. However, despite the increase of social features in LMSes, many researchers and educators have expressed concerns regarding their readiness to incorporate student participation. McLoughlin [?] 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 notes that LMSes tend to be more administration-focused, and that there were signs that Web 2.0 tools could make learning environments more personal, participatory, and collaborative. Similarly, Dalsgaard [?] believes that LMSes should only hold a minor role compared to separate, more social tools. He argues that students should be provided with a myriad of tools for independent work, reflection, construction, and collaboration.

In computer science, Hamer's research group has conducted much of the research surrounding what he calls the 'Contributing Student Pedagogy' (CSP). Hamer first reports his experiences [?] with this pedagogy in a number of courses, concluding that it helped students gain new perspectives and develop skills such as communication and teamwork. His group later formally defined CSP [?] as: "A pedagogy that encourages students to contribute to the learning of others and to value the contributions of others."

They observed 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 [?] report on the effectiveness of CSP when they adopted it into their computer science curriculum, observing benefits such as increased engagement and participation, and the development of critical analysis, collaboration, and problem solving skills—important skills for a computer scientist.

Version control systems have been used in the classroom as a way of managing students and their work. Reid & Wilson [?] introduced the Concurrent Versions System (CVS) to a second-year computer science course. This provided the instructors with a simple way to manage student assignments, made it easier for students to work in pairs or groups, and gave the instructors a history of student work. Clifton, Kaczmarczyk & Mrozek [?] used Subversion, another version control system, to collaboratively develop and run introductory computer science courses. The ease of managing

[2] http://www.wired.com/2013/09/github-for-anything/

courses using Subversion allowed the instructors to free up time from administrative demands, allowing them to spend more time focusing on pedagogical issues. In 2013, Griffin & Seals used GitHub in the classroom as a version control tool, leveraging the *Branch* and *Merge* features [**?**]. When students worked on programming assignments, it was easy to *merge* back into the original project if their version worked, or abandon a branch without destroying the original project.

Git is one of the more popular distributed version control systems used for today's software projects, and a particular way of interfacing with Git is through the use of the GitHub or Bitbucket Web platforms that include collaborative features such as wikis, issue trackers, and tagging. Student projects could leverage the issue tracker on each tool so that issues, comments, and responses can be seen by all [**?**]. Haaranen & Lehtinen [**?**] provide an example of Git being utilized in a large-scale (200 student) computer science classroom through the GitLab Web portal, citing benefits such as the ability to correct course material and giving students experience using a tool relevant to the industry as a whole.

GitHub is a Web-based social code sharing service released in 2008 that utilizes the Git distributed version control system. It is a tool utilized by millions of developers all over the world to facilitate collaboration via the use of its awareness and transparency components, collaborative features such as pull requests, and version control. The tool is organized so that developers can create repositories with their work that can be public, available for other developers to contribute to. This provides many opportunities for remixing and reusing content, as well as supporting a workflow where multiple parties can do separate work at their own pace.

In a previous study [**?**], we conducted interviews with a number of educators who used GitHub for their courses from a number of disciplines. These educators spoke of benefits such as the ability to monitor student work continuously and the ease with which they can reuse and remix course materials from other instructors. We also noted benefits which impact their students, such as learning how to use a tool relevant to their field, and the ability to make changes to course materials.

3. GITHUB USE

This section describes the courses used in the case study and details GitHub and how it was used as a platform to support learning and teaching.

3.1 Courses

The two courses were a computer science (CS) course aimed at both undergraduate and graduate students, and a software engineering course (SE) that was only taken by undergraduate students. Both classes were similar in size (30-40 students) and in learning activities (weekly labs and two course projects). The course projects involved programming individually or collaboratively working with others (in groups of 2-4 students) to produce a project relating to the course topics - in one case, projects relating to the use of existing software systems to create new ones and in another, projects relating to building systems that involve multiple computational devices.

Projects were open-ended with regard to what the students created, what topic they address, and what technologies and languages they utilize for building. However, student work was required to be publicly available so that other people, both inside and outside of the course, could view their projects. The overwhelming majority of students opted to use GitHub to host their projects. However, there was no formal introduction of GitHub to the students. The course instructor for both courses informed them that course materials were to be hosted on GitHub, and during the first laboratory session, students had to create GitHub accounts if they did not already have one. Otherwise, students were left to teach themselves and each other, or ask questions to their lab instructors should they have any.

3.2 GitHub Use in the Course

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 various repositories. The advanced use cases we discussed in Chapter 4, such as utilizing pull requests and 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.

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 would update this repository weekly, adding lab assignments, links to readings, and the student homework for the week, as seen in figure **??**. All of the content was organized into a calendar table made from Markdown, and it was visible on the home page of the course repository as a 'readme' file. Students could 'fork' the repository to request changes to be made if they wished, though this possibility was not advertised to the students explicitly.

The other main use case was in the repository's 'issues' page, where all labs (2-3 hour long sessions once a week in addition to the course lectures) were hosted. These labs would often involve researching a topic and reporting results, or giving other groups feedback on their projects. A dedicated issue would be created for each lab, similar to a forum post, and students would then make comments on these issues based on their lab work. Students were free to work in groups, and when commenting on an issue, would '@mention' their group members to indicate who the respondents were.

GitHub was also used for project hosting. Although students were not mandated to use GitHub for their course projects, most projects were hosted on GitHub in individual repositories. These repositories were public so others in the course could view the work and give feedback.

In addition to GitHub, the course instructor opted to use CourseSpaces[3], a version of the Moodle LMS[4]. CourseSpaces generally allows instructors to make their course con-

[3] `https://www.uvic.ca/til/services/services_cs/index.php`
[4] `https://moodle.org/`

tent available for students to access and interact with, to enable communication between the instructors and students through forums, to post quizzes, create wikis for a class to edit, and to track student progress and performance. For these courses, CourseSpaces was used for work that the course instructor felt should not be publicly available. For example, student grades were hosted on CourseSpaces, as well as student responses to the course readings as the course instructor felt that their potential criticisms needed to be private.

4. METHODOLOGY

Our study used exploratory research methods to gain insights on a relatively new phenomenon [**?**]. We strove to explore the student perspective to gain insights on how the use of GitHub can benefit their learning, as well as how GitHub compares to the traditional learning tools students are exposed to and what GitHub provides beyond those tools. The research questions addressed during this phase include:

RQ1: What are computer science and software engineering student perceptions on the benefits of using GitHub for their courses? We've seen evidence that GitHub can benefit educators in a number of ways. The natural progression was to explore student perceptions on how this tool and this way of working might also benefit them.

RQ2: Will students face challenges related to the use of GitHub in their courses? If so, what are these challenges? When adopting a new tool for a course, particularly a tool not tailored towards education, there may be some friction involved due to a lack of educationally-focused features. We aimed to identify these challenges so as to make recommendations towards designing a system more suitable for educational purposes.

RQ3: What are student recommendations for instructors wishing to use GitHub in a course? Just as there are multiple ways to use GitHub for development purposes, an educator has multiple options regarding how they can utilize GitHub as a tool for their courses. We aimed to learn what students consider to be important for instructors when creating a GitHub workflow that students would deem appropriate for their courses.

4.1 Research Design

We used a qualitative approach to study the student perspective of GitHub use in education. As Creswell [**?**] suggests, a qualitative and exploratory approach best suits research when a concept or phenomenon requires more understanding because there is little pre-existing research. Yin [**?**] introduces case studies as *"an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident."* Case study design, according to Yin, should be used when (a) the study seeks to answer 'how' or 'why' things happen; (b) the study is focused on the natural behavior of participants; (c) the context is important for the study; or (d) there are no clear descriptions of what is happening between the phenomenon and context. Because these conditions apply to the nature of the research questions asked in this study, I chose the case study design for this work. Specifically, the study was exploratory, serving as an early investigation on the phenomenon of GitHub in the classroom and to potentially build new theories or derive new hypotheses [**?**].

Specific to software engineering, Runeson [**?**] defines case studies as *"an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified."* In this work, we aimed to draw from multiple sources of evidence—multiple students and instructors—to investigate the potential of using GitHub for post-secondary computer science and software engineering courses. It's important to learn student perspectives in this context and to explore the suitability of GitHub for supporting education.

4.1.1 Recruitment

For this study, we were opportunistic in finding cases and sought instructors who could and were willing to try using GitHub. We were able to recruit a professor who wanted to try using the tool in two different courses. Having multiple cases would allow us to explore some possible differences between the two scenarios [**?**]. The two courses were a computer science (CS) course aimed at both undergraduate and graduate students, and a software engineering course (SE) that was only taken by undergraduate students. Both courses were similar in size (30-40 students) and in learning activities (weekly labs and two course projects). Student participants were recruited via a sign-up sheet during the first lectures of the course which gave us permission to contact them. Participation was voluntary and students who signed up were not required to participate in the study at all in any given phase, meaning those who were interviewed did not necessarily respond to the validation survey.

4.1.2 Research Methods

Students who signed up during the recruitment process were emailed to be invited to interview. Most interviews with the students were one-on-one; however, due to scheduling reasons, some students requested to be interviewed as a group of 2 or 3. Interviews with the students lasted 20-30 minutes and were all conducted face-to-face in a meeting room. Audio from every interview was recorded with participant consent and notes were taken for reference. The interviews were semi-structured based on 12 guiding questions and we probed further with additional questions as deemed appropriate.

We also interviewed the course instructors: the main course instructor for both courses was interviewed in the middle of the semester and again after the semester concluded, and the lab instructors (one for each course) were interviewed towards the end of the course in order to find out how they utilized GitHub in their labs and to uncover their opinions on the tool's effectiveness towards the learning activities they engaged in with the students. Interviews with the instructors had a similar format to those with the students: semi-structured, 20 minutes long, with approximately 7 guiding questions.

Finally, we conducted a survey to validate the findings and confirm or contradict the themes that emerged from analy-

sis. The survey was distributed during the final lab session of each course, where students were asked to anonymously fill in an online survey about their experiences. Respondents did not neccesarily participate in the interview phase. 18 students responded from the CS course (4 of which were interviewed), while the SE survey received 15 responses (9 of which were interviewed).

4.1.3 Interview Participants

We conducted interviews with 13 students from SE, 1 of which was in both courses, alongside 6 others from CS. These interviews began about 7 weeks after the semester began and concluded at the end of the semester. The main distinction between the two courses was that SE was an undergraduate Software Engineering (SENG) course whereas CS was a Computer Science course with a mix of undergraduate and graduate students. Otherwise, the courses were laid out in a similar manner (as outlined below in Section 5.5). Table 1 summarizes the students who participated in interviews.

Table 1: Participants and their prior experience with GitHub.

ID	Prior GitHub Experience	Degree Type
CS1	Inexperienced	Graduate
CS2	Used Academically, Professionally	Graduate
CS3	Used Academically, Professionally	Graduate
CS4	Inexperienced	Graduate
CS5	Used Academically	Graduate
CS6	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

4.1.4 Data Analysis

Following the interviews, we carefully transcribed every interview, then read and re-read the content for familiarity, noting important sections or responses. Next, we coded the data by labeling various segments based on the research questions of the study. Afterwards, we identified themes and concepts that surfaced multiple times. After separating the themes into well-defined categories, we compiled a final list of themes.

5. FINDINGS

We present the findings according to each of the research questions posed for this study. For each question, we discuss the main themes that emerged from my analysis, providing relevant participant quotes from the interviews. In cases where student responses were discrepant, we note them in the description of the theme.

5.1 RQ1: What are computer science and software engineering student perceptions on the benefits of using GitHub for their courses?

In this section, I discuss the benefits that emerged from the students' perspectives from the three main uses of GitHub in their courses: for schedule and material dissemination, for discussions, and for hosting their project work.

Benefit: Gaining Experience with an Industrially Relevant Tool

In the current software development landscape, GitHub is a very popular tool for working collaboratively. As such, it is essential that developers are familiar with either GitHub or other distributed version control systems, particularly when working on collaborative, multi-person projects.

Many of the interviewees mentioned that the use of GitHub in class provided a good introduction to the tool for them: *"I think it's pretty good. I mean one thing is that because I'm using it in class, it's made me learn the tool ... and that's where the big takeaway is: that I've been able to transfer those skills, I've done some other projects just on my own time using GitHub."* [SE2]

Importantly, however, putting their projects on GitHub provides practice for real-life scenarios. SE8 describes why it was beneficial to have their work publicly available for both classmates and outsiders to see: *"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]

Beyond the benefit of using GitHub in programming projects, which is what it was designed for, the basic use of GitHub to manage course activities such as material dissemination and discussion was also beneficial to students as an introduction to the tool, with some caveats. *"It's a good introduction to GitHub as a platform; it might not be a good introduction to Git as a tool. Because there's a lot of wizardry that you can do with Git that you'd never learn just doing what we did here ... but definitely a good start to get people using Git."* [SE11]

Some students were introduced to specific GitHub features that they were not necessarily aware of. *"This is the first time I've actually used the issues portion of GitHub ... So it showed me that portion of the capabilities of GitHub."* [SE13]

Benefit: Supporting Student Contributions to Course Content

According to various instructors [**?**], one of the benefits that GitHub offered over traditional Learning Management Systems is the ability for students to make changes, fixes, or suggestions to the course materials via Pull Requests (PRs) that instructors can easily reject or accept.

Throughout the two courses in this case study, three PRs were submitted to make fixes to the materials or to add links to new materials. These pull requests were submitted

in the first month of the courses and by only one student who was well-versed in GitHub (and who was registered in both courses). SE1 explains their reasoning: *"I like being able to fix the mistakes that [the course instructor] might make, like with a bad link or something, by making a PR . . . I really like being able to do that because it makes me feel a little more involved."*

This style of contribution didn't continue, however, perhaps because these initial pull requests to fix material or add links to other content were not merged quickly enough by the instructor and no one else was able to help. Another student described how this hindered more participation of this kind: *"Because we did not have the access. If we had the access, then I think people would have collaborated"* [CS2]

Although SE6 did not contribute in this manner, they saw the potential advantages of using pull requests as follows: *"I think everybody's had experience with mistakes in the course material. . . . The alternative is just emailing the prof and asking them to change something . . . this is always there, and they can always check it to see if there's something. This way someone can actually make the change, all they'd have to do is accept it."*

Discrepancy: This method of contributing to the course is limited because of the types of files GitHub supports. I elaborate on these limitations in section 5.6.4. Moreover, SE11 felt that the PR system might be problematic: *"I think it's kinda weird to be able to fix your professor's mistakes... [because] it's calling the professor out on their mistake, which some people might think is rude... I think I would do it in private or send them an email before I put in that PR."*

Benefit: Support for Students to Contribute to Each Other's Work

In these courses, projects were open and visible to other students, which allowed more opportunities for student contribution. This was demonstrated by a student's group working with others: *"For instance, one [issue] was our script wasn't taking in command line arguments if there were spaces in them properly. And then someone was like, 'you can just put in quotes'. And we were like 'oh, that's a lot better than what we were doing.' And then to be able to see what other people are having problems with and give suggestions. Even at one point, they were trying to find refactorings, and we said 'hey you can use our tool, it'll help.' "* [SE3]

Making students host their course projects on GitHub and relying heavily on GitHub in the courses resulted in students looking at each other's work (mandatory or voluntary) and making contributions in the way of advice or suggestions. As well, students would actually utilize code from other groups and help fix issues in the code when necessary. *"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]

Two specific lab assignments asked students to look at the repositories of other groups and comment on them, an exercise that students found useful, or at the very least, interesting. *"Yeah, I liked getting the comments, I liked knowing that people were kind of checking it out, and I assume they would let me know if I was doing anything horribly wrong, and I didn't get any of those comments, I'm assuming that everything was going alright."* [SE5]

SE11 extended the concept of providing feedback to the idea of peer reviewing, where students would judge the work of others and make comments on their work. SE11 explained the benefit of having others looking at and judging their work: *"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."*

Benefit: Keeping Each Other Accountable

One benefit that stemmed from GitHub's transparency features was the ability to see a history of commits to a project. This was cited by some students who used GitHub to manage their group projects—they could easily see if and when their partners submitted work. Their repositories kept an account of when each change was made, which provided collaborators an easy way to track the work being done on the project. This helped the students to 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 . . . So then if they're trying to say that 'I did this huge massive thing', and you look and it's only teeny-tiny, 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: Version Controlled Assignments

Although the instructors for these courses did not use their repositories for marking, some students believed the system could allow instructors to give constructive feedback as they built their projects and assignments. One student believed that the ability to see the student's process could be important: *"You'd see all the mistakes they made getting there, too, which is just as important to learning as the finished product."* [CS3]

SE8 described a hypothetical situation where instructors could use the student's repositories as submissions as opposed to the traditional way of submitting through an LMS: sending the code only when finished. SE8 said that this way of submission would be *"so much more useful . . . You could see everybody's contributions, you could comment on them too . . . Unless you're doing a live code demo with a TA or any instructor, you're not getting any real feedback [with the traditional submission system] . . . You have no idea where you lost the marks or where you went wrong."*

Benefit: Connecting with the Outside

The final benefit that emerged from the interviews relates to the way in which work hosted on GitHub is often publicly available for others to contribute to. For example, SE1 was highly active in the community of a certain programming language, and for their first project, they were building something related to the language. They advertised their work to the community, and members from the community then tried to help with their 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." They also noted how this was helpful: *"But for me, I find it really validating when someone else is like 'that is really cool, have you considered doing this?' "* [SE1]

5.2 RQ2: Will students face challenges related to the use of GitHub in their courses? If so, what are these challenges?

Challenge: Privacy is All-or-Nothing

While publicly sharing student projects on GitHub publicly provided several benefits, others acknowledged that it may not be appropriate for a class environment. SE4 describes this dilemma: *"So [using GitHub for your work has] got benefits and drawbacks: benefits being that other people can access your data, drawbacks being that other people can access your data."*

Most interviewees didn't mind that the class repository and their project work was public. However, many students could see the potential problems that might surface from their work being publicly available. Students mentioned that although they would ideally put 100% effort into all their submissions, this is not always realistic due to the time constraints students face. For example, SE6 acknowledged that sometimes, students rush through their work, and therefore, they might not want that work to be publicly available: *"You know it would actually be nice if they were separate or private somehow so I wouldn't have to go through everything and sanitize all the stuff I've submitted, because you know, for as much as you'd want to think you're putting 100% into it, you're not really, you know, writing some great work of art or careful analysis, so private would be nicer. For things like that."*

As well, SE1 felt that some of the work in the course repository wouldn't even be of interest to the public or to potential employers, and as such, they saw no need for the repository to be public: *"I'd rather have [our comments] be private. But only because there's not a whole lot of participation, so I don't feel they're of interest to someone publicly."*

Discrepancy: However, others saw no issue and even preferred all their work be in the public space. *"Personally I don't have a problem with it being public. I would like to have a good online activity of myself on GitHub, so that's not really an issue. I'm not really concerned if someone is going to read my blog or not."* [CS2]

One student acknowledged that there are workarounds to some of these privacy issues—where students do not have to attach their names to the work they contribute to the GitHub repositories used for the courses. *"I think part of that would be . . . 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] Indeed, one student created a new GitHub account solely for their contributions to the class. Unfortunately, this student did not want to be interviewed, and group members were uncertain as to what the motivation behind creating a new user was; presumably, they were motivated by some of the issues discussed above.

Challenge: Lack of Training on Git and GitHub

Another issue that many students described related to education and training is that there were varying degrees of experience with and knowledge of GitHub and its features, which presented difficulties with the use of GitHub in these courses. For example, SE9 believed that students who were less experienced with the tool could not take advantage of its benefits, such as the ability to make pull requests on the course materials. SE9 believed that if the instructor did not set a precedent for that behavior, it may not be used: *"I think you just have to advertise it so that the students know [to] use this as a communication tool. And then layout or give some examples on how it could be used."*

This lends itself to a bigger issue—educating students on Git and GitHub's features and on the instructor's intended workflow for using the tool in the classroom. The main course instructor for these two cases inexperienced with using GitHub, which made it difficult to educate the students on its features and caused some frustration for some of the interviewees. In fact, most students who were asked mentioned that the course could have benefited from more education on Git, GitHub, and what they can do with it. Students said that they could have hosted a lecture or a lab dedicated to learning the tool, perhaps at the beginning of the course or as an extra session. *"I think it would've been good to do some demo . . . cause I think [the instructor] talked too much about theory in class and there's no actual coding or no actual demoing."* [CS1]

This is an issue of tool literacy and an instructor who was experienced in using GitHub might have been able to better educate their students on GitHub and the features they intended to use. Beyond the instructor, this issue could have been alleviated by a greater focus on version control and DVCSes and what students can do with these tools earlier in a curriculum. As it stands, students were not able to properly utilize some of the benefits of using GitHub due to inexperience and unfamiliarity.

Challenge: Notification Overload

Although few students brought up this issue, how GitHub handles notifications from the repository nevertheless emerged as a challenge. The only way to get notifications from a repository is to 'watch' the repository. 'Watching' provides two different options: 1) to get a notification and an email only when the user is mentioned in issues or commits, and in discussions the user has commented on; or 2) to get a notification and an email when anything at all happens in the main branch (master), when someone comments on issues, commits, or pull requests, and when someone makes or accepts a pull request.

Unless students were 'watching' the repository, they would not receive email notifications for any activities unless they were directly mentioned. However, if the students did 'watch' the repository, they would receive an influx of notifications for every user comment on the discussions, which can become 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]

5.3 RQ3: What are student recommendations for instructors wishing to use GitHub in a course?

Given that the use of GitHub in these courses was relatively basic, many students, particularly those who were experienced with using GitHub for collaboration purposes, had ideas on how GitHub could be further utilized to be more beneficial for both themselves and for their instructors. Many students discussed recommendations such as which classes GitHub could best serve and the need to utilize additional GitHub features. This section outlines those responses, highlighting the suggestions students gave about the workflow for using GitHub in a course.

Recommendation: Use GitHub in More Open-Ended Courses

As discussed earlier, students had concerns regarding the public nature of the work they host on GitHub. While most students interviewed did not mind their work and their comments being in the public space, there were concerns regarding how this way of working could apply to different types of courses, particularly courses in which students are afforded less freedom in the nature of their work.

As such, some students [SE1, SE5, SE6] suggested that for courses where the discussions are self-contained, the repository does not need to be public. This would avoid some challenges, such as students submitting incomplete or messy work, but would also conflict with some of the benefits extracted from these interviews, where students can build their online presence with public work and instructors can make their courses open for outsiders to contribute to. A suggestion for avoiding these issues came from one interviewee: *"I think that as long as we have the option to make [our discussion comments] private, maybe after the course ends. So keep it intact while the course is ongoing and then we have the option [to change the privacy], everything will be okay."* [SE12] Currently, GitHub does not support doing such tasks, unless the course instructor decides to privatize the course repository as a whole after a course concludes or an individual student deletes their comments and posts.

However, students had opinions regarding what type of course would best suit GitHub. Interviewees suggested that a course similar to the two cases studied, where the work is very open-ended and could therefore exist in a public space, is where using a tool like GitHub would benefit students the most. When asked about their experiences with viewing others' projects in this course versus in other courses, SE7 said: *"I would say this class is specifically different because we had so much flexibility over what we were doing. It's not like in our Operating Systems class, [where] we make a shell that does this, this, and this. Where this was way more open ended, everyone's doing something different, so even if you could see what everyone else is doing, no one could've helped us."* [SE7] Other students acknowledged that the use of GitHub may not work in less-open ended courses because of plagiarism concerns.

It should be noted that there are ways to use GitHub privately within a course, where even student assignments are private. This type of workflow involves the instructor creating an organization and having each student create private repositories for their work to keep them private from each other. However, this workflow would then minimize many of the benefits listed in RQ1.

Recommendation: Mandate the Use of GitHub's Collaborative Features

The students who were more experienced with GitHub mentioned that GitHub's more collaborative features should have been further utilized to take advantage of the uniqueness of GitHub over traditional LMSes. One issue that some students discussed was that they saw little reason to use GitHub for courses if it was used only for material dissemination. For example, as mentioned in RQ1 above, only three pull requests were made throughout the semester.

CS3 was very outspoken on why using GitHub for this course was somewhat unnecessary: *"I don't see any benefit that GitHub has offered that we wouldn't have had in CourseSpaces. All it appears to me is it's a place where it's a file repo . . . and we already have that."* They also noted that while there's potential, the unidirectional nature of the work being done meant that the 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's 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."* [CS3]

As such, many students believed that GitHub was not being used to its full potential in their courses. The underlying suggestion was to consider which features of GitHub the instructor would like to use, such as pull requests or grading via commits, and use those features thoroughly and consistently. As it stands, some of the benefits they described to using such a system were only possibilities. An example, which will be highlighted later in Section 5.8, was reported from a student in the CS course, where even the issues were not used for discussion during labs: *"So basically we had to show it to our TA that we have done [the lab], and [the TA] used to mark it in a piece of paper. So putting [our responses in the issues] was not really necessary?"* [CS2]

An important lesson to learn is that GitHub only equips instructors and students with the possibility to take advantage of the benefits on offer. It is then up to the instructor to realize those benefits by using the features involved.

Recommendation: Define and Advertise a Workflow

Students acknowledged that GitHub was not being used to its full potential and that there was confusion surrounding the use of two tools (GitHub and CourseSpaces). CourseSpaces was used to fill some of the gaps in education support offered by GitHub, such as private forums and a gradebook. However, students tended to be displeased with this deci-

sion. *"One thing I really don't like is that we have both systems set up, and so sometimes the announcements are in GitHub, and some of the times, they're in CourseSpaces, and that can get kind of confusing, like did [the instructor] post an assignment here or there?"* [SE2]

This was an almost unanimous issue between the students interviewed, with only a few stating that they did not mind either way. Most mentioned that they would have preferred the use of just one tool, even if everything was public in GitHub. As a result, many students suggested that it would have been important to define a workflow for using this tool in a course in order to gain the benefits described earlier in the chapter. This workflow could include aforementioned activities such as utilizing pull requests or using just one tool instead of two. In the case of pull requests, for example, students advocated that the instructor should be advertising their use, thereby defining to the students that contributing to the material would be part of the course workflow: *"I think [the idea is] good, but 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]

While most students did not have suggestions as to what workflow to use, they acknowledged the importance of defining it and teaching it to the students early on in the course. SE6 wanted to *"enforce more actual Git and GitHub features in the way that we interact with the course material, and enforce GitHub use for actual projects. In a way that everybody had sort of a base level of understanding. So maybe at the beginning of the course ... there should definitely be a time when you learn Git."*

5.4 Validation Survey

In both courses, students indicated that their level of familiarity with GitHub had improved from when the course began. 30 students agreed that they would continue using GitHub for group work and for 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 in some way.

11 SE students agreed with feeling more involved in the class from viewing and commenting on other projects compared to 8 CS students who agreed. As well, most students in SE (9) felt that there was enough collaboration or student contribution to justify using GitHub in the course, whereas only half of the participants in CS felt that GitHub use was justified.

Surprisingly, most students in both courses disagreed or were neutral with the suggestion that their school work should not be publicly available. 10 CS students disliked using the discussion system on GitHub over forums with threaded discussions, while only 5 SE students disliked using GitHub 'issues' for discussion. 10 students from each course disagreed that the classes needed a tutorial in the beginning of the semester, and both strongly agreed that Git, GitHub, and other DVCSes should play a bigger role in UVic education.

6. DISCUSSION

The motivation behind this study was to uncover student perceptions on using GitHub as an educational tool by asking them to describe their thoughts and opinions during the experience. GitHub was used in three main ways: (a) as a place to disseminate material and host the class schedule, (b) as a place for students to submit their lab assignments and discuss these assignments, and (c) as a place where most students interviewed hosted their course projects, either collaboratively or alone.

A Student-Oriented Learning Tool

At a basic level, using GitHub for education can provide similar functions to those of traditional LMSes. As discussed in the last chapter, GitHub has the capabilities of providing many of the common activities found in Malikowski *et al.*'s model of features found in LMSes [**?**]. However, accomplishing tasks related to some of the finer-grain features of traditional LMSes, such as a formal assignment submission, requires workarounds. Even though GitHub can serve a similar purpose to formal educational tools, it was simply not built for education and is therefore lacking some educational features.

Where GitHub has the potential to excel, however, is in addressing some of the concerns regarding traditional LMSes outlined by various authors. Mott [**?**] discusses the 'walled garden' approach of LMSes, lamenting that the content is limited to those officially enrolled in the course, and that the LMSes support administrative functions much more than actual teaching and learning activities. Garcia-Penalvo [**?**] echoes these concerns, asserting that students need to be placed at the centre of the e-learning process. This could be addressed by giving students opportunities to participate in the course and connect with and learn from each other. GitHub can support these opportunities for students to become a part of each others' learning, creating a culture of participation [**?**].

The Contributing Student

GitHub provides opportunities for students to participate in their learning. Students are able to openly contribute to the course materials by making changes or additions directly to a course repository. Traditionally, students needed to talk to the instructor or send an email to make corrections or additions. GitHub provides a much more open and direct way for students to contribute to the course materials. This plays a key role in Collis and Moonen's concept of a 'Contributing Student' [**?**], where GitHub provides students the ability to drive their coursework. Moreover, GitHub provides students opportunities to partake in many of the 'Contributing Student Pedagogy' activities Hamer *et al.* described [**?**], including peer reviews, discussion, content construction, solution sharing, and making links.

When student assignments and projects are public, GitHub can provide students the opportunity to contribute to other students' learning by easily providing direct feedback to each other's assignments or project work. A number of groups in one of the cases in this study used this ability by leaving feedback for other groups when they noted bugs or issues in the code, and students seemed to appreciate this ability to

see others' work and provide feedback as they saw fit. Contributing to other students' work may provide benefits in developing soft skills such as communication and teamwork skills [**?**]. An instructor may also utilize GitHub to provide opportunities for students to peer review or grade each other's work. This could provide potential benefits such as more reflection for students while working, and the development of analysis and evaluation skills [**?**].

Transparency of Activities
In describing the benefits of using GitHub to support their group projects, some students described the transparency of activities as helpful for collaborating with each other. Few of the transparency features of GitHub were mentioned by the students—for example, the News Feed or the graphs were not discussed in the context of group projects. However, some students acknowledged the importance of seeing a history of work from other group members, describing the feature as a way to hold accountability and to keep up-to-date with the work. This is in line with the benefits related to GitHub use in industry [**?**].

Moreover, some students described the potential for better grading methods as a benefit of the transparency of activities on GitHub, despite these courses not utilizing the tool for grading. Compared to the traditional way of assignment submission 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.

Beyond the Course
Supporting the findings from interviews with early adopters of GitHub in education [**?**], most of the students interviewed described being exposed to GitHub and its features as a benefit to using the tool in a course. As such, the exposure to GitHub its associated open, collaborative workflow may result in some transferable skills towards their careers. Moreover, the popularity of GitHub means that student GitHub accounts become part of their online presence [**?**], which may serve an important role with potential employers who use GitHub for hiring purposes.

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

6.1 Limitations

One limitation in this study was that the semi-structured nature of the interviews meant that the interviewer would often go off-script to probe further, potentially resulting in leading questions. Moreover, the recruitment methods may have biased the population: by searching for instructors teaching appropriate courses, we first approached instructors we knew to invite them to participate in my study, which may have introduced a bias in comparison to finding a class that was already intending to use GitHub as a learning tool. As well, opportunistic recruitment may have resulted in a situation where the students willing to be interviewed were students who felt strongly about GitHub in either direction—those who may have had insights but had no strong opinions may have chosen not to participate.

6.2 Recommendations for Educators

This section provides recommendations for educators who want to use GitHub to support their courses. These recommendations are based on the findings from the two phases of this work, as well as from the review of literature surrounding tools in computer science and software engineering education.

Before proceeding, we note that GitHub has their own set of recommendations for setting up an organization for a class[5]. Their classroom guide is useful for those looking for a step-by-step process, where they recommend applying for an organization for a course and assigning a private repository for each assignment for each student. Likewise, it can also be helpful to use the available resources: use GitHub support, look for other instructor experiences for guidance, or discuss experiences in a blog or in spaces dedicated to the topic[6]. Contributing to these resources can serve towards building a common knowledge base for instructors to share to and learn from. Moreover, GitHub recently released a tool that automates many of the tasks educators need to set up on GitHub [7].

Recommendation: Utilize GitHub's Features
Computer science and software engineering students benefit from early exposure to Git and GitHub. By utilizing these (or similar) tools in their courses, educators provide students a way to familiarize themselves and practice with these tools, which can benefit their careers. Beyond exposure, hosting assignments, projects, and code on student accounts could be valuable when seeking employment, as companies continue to investigate the online presence prospective employees have (e.g., their GitHub accounts) for hiring purposes.

While simply using GitHub as a system for material dissemination can be helpful, using more of GitHub's features, such as pull requests and issues, provides even more benefits for the students. For example, allowing students to contribute to the course and to each other's work can help develop skills such as teamwork and communication [**?**]. For example, educators can use GitHub's transparency features to provide feedback to students in unique ways, such as tracing the history of student projects and assignments hosted on GitHub, detailing where students made mistakes and intervening when a student seems to be struggling. Moreover, in group projects, instructors can note how much work each student has contributed, and can use this transparency for assigning grades.

Recommendation: Use Free Private Repositories for Single Solution Assignments
Many students believed that GitHub worked best when a

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

course has open-ended projects and assignments. This stems from plaigarism concerns that exist when students are putting their code up online where others can potentially see their solutions. Of course, students can host their code in private repositories controlled by the instructor; if the instructor creates a private repository for each student to submit their assignments and adds only the student as a collaborator, plaigarism would only be as much of a concern as it would be without using GitHub. Otherwise, an instructor could ask students to create a private repository for their assignments that only the instructors can view and contribute to.

However, single-solution assignments being hosted in private repositories limit one of the most important benefits of using a system like GitHub—the ability to view, comment on, and contribute to the work of other students. As such, although GitHub is usable and helpful in any type of course, courses with open-ended projects and courses with a culture of participation are where instructors and students will see the primary benefits of using GitHub as a learning tool. If an instructor chooses to pursue the open-ended style of work similar to the courses in this study, it is recommended that they list projects and assignments on the home page using the readme markdown file so students can easily access the other projects.

Recommendation: Encourage Contributions from the Students

Another way to utilize GitHub is to encourage contribution from the students in the ways that GitHub affords them. First, students can contribute to the course materials by making corrections, changes, and adding resources. Second, students can contribute to other students' work and projects (provided the work is open-ended). And third, students can contribute to projects outside the course by making changes and pull requests in open-source repositories. Encouraging this 'Contributing Student Pedagogy' can help students develop skills such as critical analysis and collaboration [**?**].

Moreover, all student contributions are available for the course instructor to see. As an example, an instructor can grade students based on their contributions, such as when they create an issue or a pull request on another project. However, one issue with student contributions that must be noted is that contributing to the course materials could present difficulties depending on the file types used, as binary files such as PDF documents and PowerPoint slides are not compatible with the GitHub web interface. Although GitHub has recently provided support for viewing PDF files on the platform[8], these files remain unsupported by GitHub's 'diff' feature, which means that changes to the file are difficult to discern and changes to the file by multiple people will always result in a 'merge conflict'. For this reason, I recommend hosting class material and slides in either markdown or HTML, file types that GitHub supports and can be easily altered using its Web platform.

7. CONCLUSION

Another 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 [**?**]. This work demonstrates how using GitHub can unlock activities where students can contribute to each other's learning, and as a result, I believe it can be beneficial to add support for GitHub's open, collaborative workflow to current and future tools focused on learning.

As such, one possible path is the 'GitHub for Education' Greg Wilson discussed[9], where a tool like GitHub can be altered or built to be more focused towards 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[10], that could be further developed into a tool that fulfils 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 issues within a repository private while others remain public. This is potentially an avenue for future work, where such a tool can be evaluated.

In summary, this work has shown the viability of using GitHub for education, and has demonstrated why the open, collaborative workflow associated with GitHub should be considered when deciding which tools to use to support a course. Based on the findings of this work, we included a set of recommendations for educators interested in using GitHub as a learning tool, and list the implications on tools that could provide the same benefits as GitHub while mitigating the limitations.

8. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

[8] `https://github.com/blog/1974-pdf-viewing`

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

[10] `https://about.gitlab.com`

A.2.2 Math Equations

Inline (In-text) Equations

Display Equations

A.2.3 Citations
A.2.4 Tables
A.2.5 Figures
A.2.6 Theorem-like Constructs
A Caveat for the TEX Expert
A.3 Conclusions
A.4 Acknowledgments
A.5 Additional Authors
This section is inserted by LATEX; you do not insert it. You just add the names and information in the `\additionalauthors` command at the start of the document.

A.6 References
Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.

B. MORE HELP FOR THE HARDY
The acm_proc_article-sp document class file itself is chockfull of succinct and helpful comments. If you consider yourself a moderately experienced to expert user of LATEX, you may find reading it useful but please remember not to change it.