Towards a Collaborative Learning Platform: The Use of GitHub in Computer Science and Software Engineering Courses

by

Joseph Feliciano
B.Sc., University of Victoria, 2015

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Computer Science

in the Department of Computer Science

Towards a Collaborative Learning Platform: The Use of GitHub in Computer Science and Software Engineering Courses

by

Joseph Feliciano
B.Sc., University of Victoria, 2015

Supervisory Committee

Dr. Margaret-Anne Storey, Supervisor
(Department of Computer Science)

Dr. Allyson Hadwin, Outside Member
(Department of Educational Psychology)

ABSTRACT

Technical fields such as computer science and software engineering have placed an emphasis on collaboration and teamwork, and training students entering these fields is a challenge that educators and researchers have attempted to tackle. To develop students' skills for these technical fields, some educators have integrated learning activities where students collaborate heavily and make contributions to each other's learning, emulating the type of work students will perform in industry. Consequently, the learning tools that instructors use for their courses need to support these collaborative and contributive activities.

GitHub is a social coding tool that has seen rapid adoption in the software development field because of the open, collaborative workflow it encourages. This thesis explores the use of GitHub as a collaborative platform for computer science and software engineering education. GitHub provides users with opportunities to contribute to each other's work through its transparency features, supports integrated discussions, and provides support for reusing and remixing work—opportunities which may be extended to education.

In this thesis, I investigate how GitHub's unique features, such as 'pull requests' and commit histories, can be used to support learning and teaching. This work also explores the benefits and challenges that emerge from using GitHub in this context from both the instructor's and the student's perspectives. We found that GitHub afforded instructors with opportunities to encourage student participation by contributing to the course materials through the use of 'pull requests' and provided instructors with ways to reuse and share their course materials. As well, students gained experience with a tool and a workflow they expected to encounter in industry, and were provided ways to further engage in their learning by giving feedback to or further developing other students' work. However, we found that instructors and students were challenged by GitHub's lack of educational focus, as well as the implications of using GitHub's open workflow on the public availability of student work.

Findings from this work determine the viability of GitHub as a tool for supporting computer science and software engineering education, and contribute to our understanding of what activities and benefits GitHub provides beyond traditional learning tools. The contributions of this work include a set of recommendations for instructors wishing to use GitHub to augment their courses, utilizing GitHub's features to support educational activities such as student contributions to course materials and providing continuous feedback to students.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Margaret-Anne Storey, for your guidance, your encouragement, and for giving me the opportunity to pursue something I never imagined I would be doing.
My family, for the love and support that helped keep me going throughout the years.
Sydney Barnes, for believing in me and for picking me up when I needed it. Thanks so much for motivating me and for being there for me.
Cassandra Petrachenko, for the editing of this thesis, for the help throughout the two years, and for your infectious sense of humor.
Alexey Zagalsky, for kicking this whole thing off! Thank you so much for guiding me and offering your insights on everything.
Chisel Lab and everybody in it during my time there, for being a great and supportive environment, making it easy to work and to laugh.
Dr. Daniela Damian and **Dr. Yvonne Coady** for being a big part of a wonderful university experience and for the inspirations throughout.

DEDICATION

To Lolo Joe.

Chapter 1

Introduction

As technology continues to play a vital role in the education of post-secondary students, Web-based tools have evolved to cater to educational needs. Instructors in both local and remote classrooms utilize a number of these technologies to disseminate material and engage their students, including tools such as Learning Management Systems (LMS) that are focused on education, or tools created for other purposes used in an educational context, such as social media platforms. These technologies can be defined as tools for ‘e-learning’, a term that can be defined as *“electronically mediated asynchronous and synchronous communication for the purpose of constructing and confirming knowledge”* [25].

Coinciding with the Web 2.0 movement [53], education began to emphasize social interaction through the use of social media or computer-mediated communication tools. Such tools have characteristics which afford users more freedom to create and publish content. This allows for an approach to learning characterized by a *demand-pull* model rather than a *supply-push* model, and focuses on participation and providing students access to rich learning communities [62]. These tools have introduced ways for students to be more than just passive observers who absorb content as given to them. Instead, they are able to participate in communities, and as a result, their learning can become more social and engaging.

Computer science and software engineering education utilizes these e-learning tools in similar ways to other disciplines. However, work in these fields often involve a large amount of collaboration with others, such as being able to examine, understand, and build upon another person’s work. As such, the tools used for educating students in these technical fields need to support such activities, providing students training and practice in the skills related to collaboration such as communication and teamwork skills.

In the software development field, 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 [15]. 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 workflow 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. I describe GitHub and its features in more detail in Chapter 3.

This thesis explores tools in computer science and software engineering education, specifically identifying GitHub's use in this context as a way to address the needs of students in these technical fields. For both educators and students, working using the GitHub way has both potential benefits and drawbacks, which this work aims to explore and identify—we try to determine whether or not GitHub's open collaborative workflow is a viable approach in computer science and software engineering education.

1.1 Problem Statement

An important goal of computer science and software engineering education is to prepare students for their future careers in industry. As many software projects today are increasingly collaborative, such as those in globally distributed or open-source projects, it becomes necessary for educators to prepare their students for a career in this environment. Yet, many researchers and educators assert that students are generally lacking in their group work skills [67].

While some are adjusting their curricula to account for these weaknesses [33] and giving students opportunities to develop skills in environments closer to real world experiences [10], another area to investigate is the use of tools in computer science and software engineering education. Extending to other disciplines, Web-based e-learning tools are regularly used in education as the main portal for students to engage

[1] `http://readwrite.com/2013/11/08/seven-ways-to-use-github-that-arent-coding`
[2] `http://www.wired.com/2013/09/github-for-anything/`

with a course and its participants. Therefore, educational tools such as Learning Management Systems becomes an important point of interaction between not just the students and educators, but between the students themselves. Yet many of these tools are poorly equipped to handle collaborative tasks and student participation because of their focus on administrative and instructor tasks rather than on student learning [47]. As such, finding tools that have the potential to support collaborative activities similar to real-world experiences remains a challenge for computer science and software engineering education.

This work studies instructors and students that attempted to use GitHub as an e-learning tool. While the instructors and students reaped many benefits from using GitHub in 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.

1.2 Motivation

This project began when our research group noted the popularity of GitHub and how it was seeing widespread adoption in other areas[3], transforming how people collaborate over a shared repository [2]. One of GitHub's main strengths is in the awareness and transparency features it provides to team, project and community members [15]. These features positively influence how people contribute to projects [66].

The use of GitHub in an educational context is certainly not novel. A few years after GitHub's release, Greg Wilson, a well-known computer science educator, suggested that GitHub could be used for learning materials despite its limitations, citing remixing as the primary benefit of using GitHub in this context[4].

> *Would it be possible to create a "GitHub for education?" Right now, I think the answer is "no", because today's learning content formats make merging hard. Whatever a "GitHub for education" would look like, it would not be yet another repository of open learning materials. There are lots*

[3]`http://www.wired.com/2013/09/github-for-anything/`
[4]`http://software-carpentry.org/blog/2011/12/fork-merge-and-share.html`

> *of those already, but almost all their content is write-once-and-upload, . . . rather than sharing course content in a reusable, remixable way.*

In 2012, he elaborated on this idea[5]:

> *"GitHub for Education" isn't necessarily, "Let's put educational materials in GitHub", but rather, "Let's facilitate a culture of spontaneous-but-structured collaboration and improvement."*

Wilson recognized the limitations of the majority of LMSes in the difficulty for instructors trying to reuse and share materials. This is the type of problem that the software development community had already solved using tools such as GitHub. This brings into question whether such tools originally built for software developers have their place in other fields that have similar needs such as education.

In an effort to promote GitHub to higher education, GitHub launched the GitHub Education Website—`https://education.github.com/`—in 2014. This promotion offers support to students and instructors using GitHub for educational purposes, and signals GitHub's intention to become a more useful educational tool.

Consequently, software developers are sometimes described as the *"prototype of future knowledge workers"*, as they are often the first to adopt new tools and techniques[6]. Because software developers and programmers are able to create and change products to meet their needs, they more often than not become the early adopters of tools that other fields would eventually use. This is evident in examples such as e-mail, the Web, and wikis: tools that were invented and used by programmers, which later became significant tools in other fields.

Moreover, programmers and software developers experience 'always-on' learning, where they constantly have access to learning materials and content relevant in their field. In software development, knowledge is constantly evolving and developers often have to stay current with languages, libraries, frameworks, and tools. Unfortunately, courses are often self-contained, with little input or output to and from outside software development communities. This could potentially be addressed using tools in which users can interface with other communities and groups beyond the class.

[5]`http://software-carpentry.org/blog/2012/04/github-for-education.html`
[6]`http://allankelly.blogspot.ca/2014/04/the-prototype-of-future-knowledge.html`

1.3 Research Questions

This thesis aims to explore tools in computer science and software engineering education, particularly the use of a tool such as GitHub and its effectiveness in the educational context. These research questions are exploratory in nature:

- RQ1: Is GitHub's open, collaborative workflow a *viable* approach for the education of computer science and software engineering students? What does a tool like GitHub provide instructors and students *beyond* traditional learning tools?
- RQ2: What are the benefits and weaknesses to this approach from an *instructor's perspective*?
- RQ3: What are the benefits and weaknesses to this approach from the *student's perspective*?

1.4 Research Approach

I developed my research questions using a constructivist approach, where I sought to construct knowledge and theories, relying mainly on participants' views [20]. According to Easterbrook, a constructivist approach does not focus on verifying theories, but instead seeks to understand how people make sense of the world, and might build theories based on the context being studied. In this work, my goal was to seek student and instructor perspectives on the use of GitHub in an educational context. These perspectives would help determine whether or not the GitHub way is a viable approach to education, and one that can influence the development of tools focused on computer science and software engineering education.

I address the first research question with a thorough literature review, where I discuss work surrounding computer science and software engineering education, tools typically used in education in general, and tools specifically for educational use in these technical fields. This review also includes literature surrounding GitHub and similar Distributed Version Control Systems, particularly exploring their collaborative features and describing other instances in which they have been used in an educational context. In searching for literature, I focused on educational tools, exploring what researchers believe typical LMSes lack and how those limitations can be addressed.

The second research question is addressed from work conducted in two different phases—the first involving instructors as participants, and the second involving two courses as a case study. In both phases, the research methods used were congruent with exploratory research as we sought to explore the perspectives of instructors and students on the use of GitHub as a platform for education. In this thesis, both phases are described in separate chapters, and the research methodology is explained in more detail within each chapter.

1.5 Thesis Organization

The rest of this thesis is organized as follows.

Chapter 2 summarizes the related work, including literature on Computer Science and Software Engineering education, learning tools, and student engagement. The aim of this chapter is to elaborate on where this research fits into the surrounding literature.

In Chapter 3, GitHub and its features are described in greater detail, including the possibilities for application to education. The aim of the chapter is to provide an overview of how GitHub might fit into the educational landscape.

In Chapter 4, I describe the first phase, in which a study where instructors who were early adopters of using GitHub for education were interviewed. The aim of the study was to explore how and why educators used GitHub for their courses, including the benefits and challenges they experienced from using the tool.

In Chapter 5, I describe the second phase, which involves a study where GitHub was used in two courses at the University of Victoria. The primary aim of the study was to explore the student perspective regarding what they believed to be the benefits and challenges of using such a platform for their courses. Moreover, we explored the teaching team's perspective to better understand these benefits and challenges, particularly in comparison to the LMSes traditionally used in the computer science department in the university. The study also aimed to identify student recommendations towards workflows they believed would best fit the use of a platform like GitHub.

In Chapter 6, I discuss the relevance of the work done and the implications on future learning tools. I also describe some recommendations for possible uses of GitHub in an educational context.

In Chapter 7, I outline possible future work and conclude the thesis by discussing how my research questions were addressed. Additional documents are found in the Appendices, including the interview questions for both phases.

Chapter 2

Background

As computer science and software engineering evolves, so do the methods of educating students in those fields. Collaboration and developer contribution is often studied in current software engineering research as the ‘social programmer’ becomes an increasingly common description of today’s developer [65]. Being a software developer in today’s landscape involves a social element, as developers often need to collaborate and coordinate with each other or contribute to the knowledge maintained by a community via social media. Recent research addresses collaboration as a common topic, evident in Whitehead’s roadmap of collaboration in Software Engineering [69].

An important goal of software engineering education is to provide students with the skills needed to integrate theory and practice, preparing them for their professional careers. Students in this discipline need to be able to solve different problems both on their own and as members of a team. For these students, problem solving is best taught through examples and learned through practice [33]. Working in collaboration with other students on projects can, to an extent, simulate a real-world environment of working in a development team, and many university courses provide this by assigning group assignments.

As such, researchers have posited that collaboration is an important facet of software engineering that needs to be integrated into education. Included in Shaw’s roadmap for software engineering education is one such example of collaboration, where students need to study good examples of code and iterate on other people’s code [63]. Although this involves indirect collaboration, Shaw felt it was important to educate software engineers in such a way that they are able to read, understand, and build on other code. Jazayeri [33] developed a curriculum for his university, identifying a need to teach non-technical ‘soft’ skills (such as communication and teamwork) alongside technical ones. In discussing the future trends of software engineering education, both Lethbridge *et al.* [44] and Mead [48] highlight that teams are becoming increasingly distributed, and the social contact between group members can be very high. This suggests the importance of going beyond teaching just the technical skills, but to also account for the social nature of software engineering, where developers need to work together regularly in various ways. By creating opportunities for students to also work together, they gain experience in these real-world scenarios of collaborating in a team.

Participatory Culture in Education

Computer science and software engineering education has shifted away from the traditional method of learning in which material is transmitted unidirectionally by instructors and books to be absorbed by students. In 1998, Ben-Ari [3] discussed the concept of Constructivism, the theory that knowledge is actively constructed by students rather than passively absorbed through reading and lectures. Through Ben-Ari's survey of constructivism in science and mathematics education, he argued that there needs to be more consideration for constructivist education in computer science.

Research then shifted towards the social construction model, which places a larger emphasis on the social and cultural context surrounding students. In this model, individual knowledge is created through interactions with others as well as interactions with the environment [37]. This social emphasis is rooted in much of the research and theories in computer science and software engineering education. As an example, some researchers have explored communities of practice, which Wenger classifies as peers that share and develop knowledge in a common context [68], and whether or not these communities fit into an educational context [4].

Jenkins [34] discusses a similar concept—a participatory culture—as ideal in education, where students regularly contribute to each other's learning through the use of social tools. A participatory culture is characterized by the following attributes:

- Relatively low barriers to artistic expression, where participants can create and remix content at will.
- Strong support for creating and sharing one's creations so they will be easily available to others.
- Some type of informal membership, where more novice members can learn from the more experienced.

- Members believe their contributions matter.
- Members feel some degree of social connection with each other.

The concept of the participatory culture strongly reflects a social constructivist approach, as students contribute to each others' learning by interacting with the material and with others. This culture encourages students to remix and share content, give feedback, and help each other.

Specific to computer science education, Machanick [45] advocates a similar social constructivist model by making the learner more involved in knowledge creation and bringing them closer to experts. He proposes an action learning model where students formulate a model (the planning stage), attempt to carry it out (the action stage), and then reflect on what happened (the reflection stage) until a specific problem is solved. Social elements, such as reviewing other work or gathering advice from experts, would be included in each stage, and the learning would occur in the context of the group or with an 'expert' in the field. This model is one such example of researchers and educators attempting to integrate these social elements and a participatory culture into student coursework, exemplifying how the social construction model has permeated education. The next section discusses an approach that extends these social and participatory models based on student contributions.

The Contributing Student in Computer Science

Student engagement was a construct proposed by Astin [1] as a developmental theory for college students surrounding the concept of involvement. This referred to *"the amount of physical and psychological energy that the student devotes to the academic experience"*. According to Kuh [40], this is an important concept due to the effects student engagement can have on student grades and student retention between first and second year. Moreover, some literature suggests that engagement can be achieved through group work and group learning [5].

Collis and Moonen [11] proposed a more social approach to education, the contributing student', where students contribute materials for other students to learn from. In this concept, the tool being utilized in the classroom plays an important role, as they note that the tool or site being contributed to should be largely empty before the learners and instructor fill it through course activities.

As this concept evolved, the key idea remained the same: learners should create or find learning materials and share them with others as a way to engage in their learning

[12]. By contributing to the course material with their findings and experiences, students can affect each others' learning. This means a student adopts several roles in a learning community, including being a co-creator of learning materials, being someone who extends the work of others (rather than just reading them), and being someone involved in self and peer evaluation.

In the computer science discipline, Hamer's research group has conducted much of the research surrounding what he calls the "Contributing Student Pedagogy" (CSP), an extension of Collis' approach. Hamer first reports his experiences [29] with this pedagogy in a number of courses, concluding that although not all students liked the approach at first, it helped students gain new perspectives and develop skills such as communication and teamwork. His group later formally defined CSP [30] 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. This pedagogy places an emphasis on the social interactions between both students and instructors, which is a key concept in much of the computer science and software engineering education literature described in this chapter. Falkner and Falkner [22] 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.

In the literature surrounding the idea of "the contributing student", researchers emphasized the importance of the tools used in a course. Without the appropriate tools, according to Collis and Moonen [12], this approach to student engagement may not even be feasible in practice. In the next section, we explore the literature surrounding the tools often used in education, and how they fit the aforementioned social approaches to education.

The Supporting Technology in Education

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 [41]. The use of an LMS provides students and educators with a set of tools for typical classroom processes, such as managing a student roster, forum discussions, or making announcements to the class.

With the rise of the social web and 'Web 2.0' technologies and services, as well as the increasingly social approach to education, the tools used for teaching and learning changed in a number of ways. Many have advocated leveraging these Web 2.0 technologies to support learning and teaching to create 'e-learning 2.0' [19], where learning tools have transitioned into more social software, such as wikis and blogs. Social software can be defined as *"applications and services that facilitate collective action and social interaction online with rich exchange of multimedia information and evolution of aggregate knowledge"* [54]. These allow users to create content that is dynamic, remixable, and open to feedback. This transition to the use of social software, according to Downes [19], involves a different type of distribution than traditional learning management systems where materials are not just disseminated, but also remixed and repurposed to involve more student participation, increasing engagement in their learning.

Researchers and educators in various fields have conducted a variety of studies using technology to increase student engagement and performance. In utilizing Twitter, Junco *et al.* [35] increased student engagement and improved student grades by simply teaching students how they could utilize Twitter for their courses, such as by asking questions, continuing class discussions, and being given academic and personal support. Chen *et al.* [8] used data from the National Survey of Student Engagement (NSSE) to provide evidence for a positive relationship between using technology in learning and student engagement and desirable learning outcomes. Minocha [50] highlighted a number of case studies surrounding the use of social software to support student learning and engagement, seeing success in the use of virtual environment tools such as 'Second Life', wikis, blogs, and Twitter.

Importantly, these tools provide benefits beyond engaging students. In a study, Minocha and Thomas [51] introduced wikis as an environment for their students in a software requirements course, where students would collaborate to create and discuss requirements together on a wiki. The instructors described the benefits of using such a system, where students enjoyed the feedback they received from other students and the ability to assess each others' work. Moreover, the instructors felt that wiki use was commonplace in industry, and therefore, it was beneficial for the students to develop their communication and teamwork skills, which are transferable to industry.

In many cases, '2.0 technologies' were simply used in conjunction with an LMS with great success. For example, Conde *et al.* [13] utilized Twitter in a course by allowing students to tweet about questions and their opinions to issues, and they believed this resulted in an increase in students' grades. Students have come to value the way such tools provide convenience, and even expect these tools to be available for communication purposes [7]. As a result, developers began incorporating these technologies into Learning Management Systems to account for more social approaches. Edrees, for example, [21], 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 [47] 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 [16] 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. He does, however, acknowledge that effort needs to be made so that social tools like blogs and wikis can support educational activities, because they are otherwise not educationally-focused tools.

Further weaknesses of LMSes are outlined by Mott [52], who believes that LMSes impede teaching and learning innovation because courses often expire after some time, making the course activities, materials, and student contributions inaccessible for future reference. As well, he argues that they offer few opportunities for student-initiated learning and that courses are 'walled gardens'—closed off to outsiders—that limit the potential for collaboration and the remixing of work. Garcia-Penalvo [24] believes that students need to be placed at the centre of the e-learning process, but that the current generation of LMSes are insufficient for this due to their lack of openness, resistance to change, lack of integration with informal context, and so on. These criticisms of traditional LMSes suggest a need for change, and the next section describes the steps being made to make tools in computer science and software engineering education that are more focused on student participation and contribution.

Evolving Learning Tools for Computer Science

Researchers have attempted to conceptualize or build enhancements for LMSes to address weaknesses and further focus them towards computer science education. Rossling *et al.* [58] introduced the concept of CALMS, a Computer Augmented Learning Management System, wherein the typical LMS would be extended to support activities such as allowing students to assess each other, automatically grading programming submissions, and actually programming through a connected IDE. The majority of the features described in their idea of CALMS provide students with various ways to contribute to each others' learning. The same working group investigated the open source LMS Moodle [59], identifying plug-ins that can augment the system to support features like inclusion of source code, shared calendars for groups, automated assessment of programming assignments, and other features more focused for computer science and programming work rather than student participation.

Other attempts have been made to leverage existing tools to benefit computer science and software engineering students. Team projects have implemented collaboration tools such as the Jazz plug-in for the Eclipse IDE, with its version control, wiki, and instant messaging features [49], giving students more ways to interact with the material and with each other. Reid & Wilson [57] created the DrProject portal, focusing on developing social media-type features (wiki, mailing list, tags) around a coding repository (Subversion). Educators are seeing the advantages of using tools that software engineers and developers use in the real world as students are familiarized with the tools prior to starting their careers. Moreover, in many of these cases, the concepts or developments made to these tools attempt to further develop the social element of the tools used in the classroom, lending credence to the growing need for tools to support these social activities.

In 2011, Hamer [31] reviewed tools that support CSP in computer science education, noting seven different CSP activities that tools can support:

- Peer review—students can see and analyze each other's work and provide feedback.

- Dialogue and discussion—the student contributions occur in the communication between the students.
- Annotation—students can comment on existing (not student-created) materials and share their comments with other students.
- Content construction—students can create new learning material for other students to consume and learn from.
- Solution sharing—students can share their solutions with other students.
- Activity creation—students can create learning activities for other students to engage in.
- Making links—students can search for external resources that relate to the content.

While Hamer *et al.*'s literature search provided a number of tools that meet many of these characteristics, they were surprised that there weren't more examples of tools that support student-contributed learning activities. As well, they reported that many of the tools seemed to only be used within the institution where they were developed, not supporting cross-institutional use. This suggests that tools in the computer science and software engineering disciplines need improvements to further support student participation and collaboration.

As distributed version control systems play a crucial role in many software projects, including their support for developer contribution and collaboration, researchers have attempted to see how these systems can benefit education. Reid & Wilson [56], introduced Concurrent Version Systems (CVS) for their classes, making it easier for students to work in groups as well as providing a history of student work. Beyond those obvious advantages, instructors and teaching assistants were also able to assist students better as they could easily retrieve an up-to-date copy of student work. Similar advantages are found when other version control tools such as when Subversion [9] and Git [27] are used in education, using features such as branching and merging to organize assignments and assignment submission.

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 [36]. Haaranen & Lehtinen [28] 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.

In summary, the literature surrounding computer science and software engineering education suggests the importance of student participation, contributions, and collaboration. These activities allow students to gain valuable soft skills such as communication and analysis, as well as experience in tools and processes that can play an important role in their careers. In the next chapter, we describe GitHub as a tool that enables these activities for software development and highlight the motivations behind using it in educational contexts.

Chapter 3

The GitHub Way

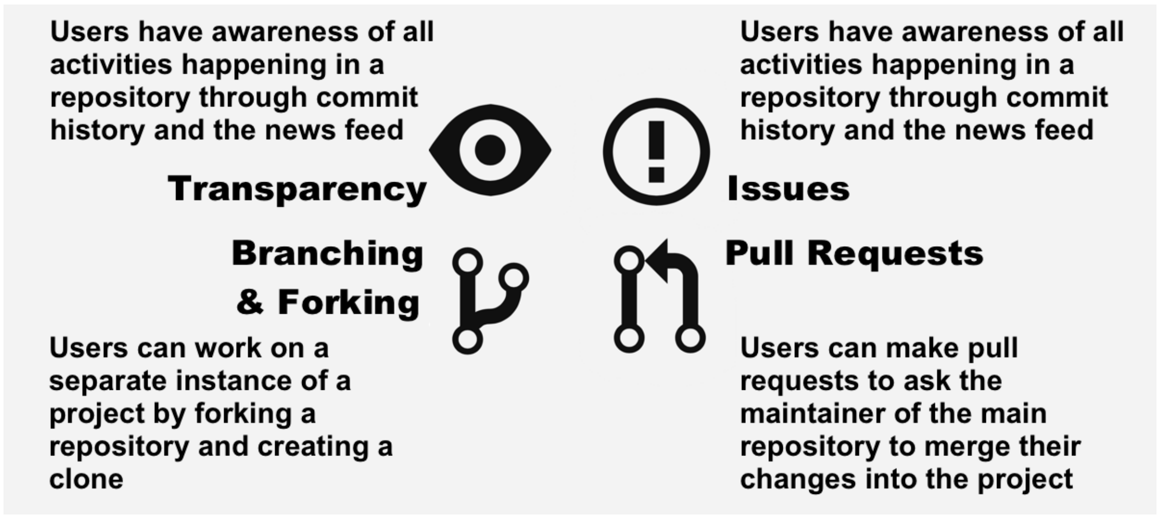
This chapter describes GitHub in detail, outlining what it is and some of its defining features, how it is used, and why it might be useful in education. GitHub exemplifies 'The GitHub Way': an open, collaborative workflow in which users of GitHub and similar platforms work and collaborate, and where one's work is open and available for multiple contributors to edit, add to, and discuss. This is an important distinction as the aim of this work is to investigate not just how GitHub, but also how the underlying activities that GitHub supports can impact education.

3.1 What is GitHub?

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. Some of its primary collaborative features are summarized in figure 3.1, and described in further detail in this chapter. The tool is organized so that developers can create repositories containing their work, which they develop on their own or share with other developers who can, in turn, contribute to the code. Repositories can be public, which means that anybody can see them and pull the code into their own repositories, though the owner can decide who can and cannot make changes. Alternatively, they can be private, whereby the repository is viewable and editable only by those given permission by the owner.

GitHub recommends repositories to be smaller than 1 GB, and for files to be smaller than 100 MB[1]. GitHub also recommends hosting file types that contain only plain text, such as code and Markdown files. Markdown is a markup language with plain text formatting designed to be easily converted to HTML. The language supports text formatting such as headings, lists, and bolded text, and is often the format of 'readme' files on GitHub, files which typically contain information about the other files in a directory. For example, a project repository's 'readme' might contain information such as a description of the project, the programming languages involved, and links to websites relevant to the project.

Figure 3.1: The primary features of GitHub that support collaboration between users.


Users have awareness of all activities happening in a repository through commit history and the news feed
Transparency
Users have awareness of all activities happening in a repository through commit history and the news feed
Issues
Branching & Forking
Users can work on a separate instance of a project by forking a repository and creating a clone
Pull Requests
Users can make pull requests to ask the maintainer of the main repository to merge their changes into the project

3.2 Git: Distributed Version Control

Git is the underlying version control system that GitHub utilizes. There are two very important aspects to Git: that work is distributed, and that work is handled by version control. Being distributed refers to the possibility of work being decentralized: instead of being forced to work in a repository where there is a central hub that everyone pushes code to, individual developers can create public 'clones' of that repository and 'push' to their respective clones before the original repository's maintainer or owner pulls in the work. 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.

[1]https://help.github.com/articles/what-is-my-disk-quota/

Version control means that developers can easily track changes to their code and that multiple developers can work on the same file, as combining changes requires a simple ‘merge’ process that Git handles. In this system, when a user makes changes to the project, they ‘commit’ their changes, effectively saving a snapshot of the project as it is at that point in time. These commits, or snapshots, are saved in the project’s history, allowing developers to revert changes as needed. The user can then push all of their changes to the server, meaning other collaborators can see the changes made. If a collaborator has made changes as well, these can be merged together to combine the different changes into the main code base.

3.3 Branching and Forking

‘Branching’ and ‘forking’ provide two ways of diverging from the main code base. A user can make changes in a repository ‘branch’, which is a deviation of the code from the code base (the ‘trunk’), but the changes remain a part of the main repository[2]. When a user branches from the trunk, they can still monitor changes to the trunk despite working in a different branch.

‘Forking’, meanwhile, achieves a similar function of deviating from the code base. The main difference is that a fork is independent of the main code base, meaning a user who forked a repository won’t be aware of the changes happening in the main code base unless they are explicitly watching the original repository[3]. When a user forks a repository, the repository, including all of its branches, are copied. On the other hand, if a repository is deleted, the fork still exists, a consequence of decentralizing these repositories.

Generally, branching provides a good workflow for development teams whose members need to be aware of all the changes made to each branch and the main code base. Meanwhile, forking tends to work for open-source projects where the repository owner does not want to manage user access to the repository and wants to keep collaborator changes independent until they are ready to be merged.

[2]`https://guides.github.com/introduction/flow/`
[3]`https://help.github.com/articles/fork-a-repo/`

3.4 Merging and Pull Requests

'Merging' is the mechanism for combining changes or pulling changes from a branch or fork into the main code base. If a user wants to make changes to a repository, they can branch or fork from the main code base and make changes to the code as they see fit, and it would remain in their branch or repository. In order to get changes into the main code base, the person managing the repository (maintainer) would have to merge the changes into it, combining whatever changes were made in the branch or fork with the code base.

'Pull requests' (PRs) are a way of handling these merges. In a PR, the person making the changes in a branch or a fork will request that the code base maintainer merge their code into the main trunk, otherwise known as the master branch. These PRs will be listed on GitHub in a separate tab where collaborators can see each of the commits made, what files were changed, and the conversation surrounding the code. The user making the PR can also include comments such as descriptions of their changes or explanations of how their changes affect the project or code. Collaborators can also make comments on PRs, typically when changes have to be made before a PR can be accepted, or a '+1' to signal agreement when they think a PR is ready to be merged into the main code base.

In some cases, such as when two or more users change the same section of code at the same time, 'a merge conflict' can occur. This requires the user to manually pick and choose which of the changes or pieces of code they want to keep. However, there can be issues with file types that are not versioned by Git, such as PDF documents or PowerPoint presentations, as changes anywhere in a file within two different branches can create unknown merge conflicts. Moreover, when someone changes a file that is an unsupported file type, collaborators cannot see the changes (the 'diff') easily.

3.5 Issues and Comments

The 'issues' feature on GitHub provides a mechanism for collaborators to engage in discussion. Issues can be tagged with any label the issue creator or editor wants, such as 'bug' or 'feature', and the list of issues can be filtered to only show issues that contain a certain tag, or only closed or open issues. Issues can be assigned to a user, letting others know who is in charge of that issue, or they can be assigned to a 'milestone', a due date set by a collaborator. PRs are also automatically posted as an issue, which is closed when the associated PR is accepted or closed. Users can comment on issues, making issues a hub for discussion. Users can also refer to issues in commit messages, which links the commit to the issue: for example, an issue fixed by a commit might say *"collaborator closed this in d2ad525 on Oct. 25, 2014."*

One of GitHub's more important features is the flexibility afforded to users making comments. For example, users can mention others by referring to their username preceded by an '@' character, thereby sending them a notification and, in most cases, an email. This allows collaborators to directly refer to another when, for example, they would like to ask someone to make a specific change to an issue or PR. Users can also comment on more specific artifacts, such as pull requests, commits, or individual lines of code. This gives users the ability to discuss various aspects of a project with other collaborators. For example, when a specific line of code is difficult to understand or causes a bug, that line of code can be highlighted in an issue.

3.6 Openness & Transparency

GitHub's openness and transparency features allow groups to facilitate both direct and indirect collaboration, allowing users to have a full view of activities occurring in a project. For example, users can look at the commit history and see specific changes on specific commits. This allows them to see exactly what people are working on. This history is useful, and when used with the 'blame' feature[4], shows who changed a specific line of code.

Upon logging into GitHub, the first screen the user will see is their News Feed[5]. This displays the activity on all the repositories that the user is involved in or is 'watching', a feature described below. Any comments, pushes to code, pull requests, or new issues appear in the News Feed, allowing the user to be aware of the events happening in their repositories.

GitHub also includes the 'Watch' feature, which has three options. If a user chooses to 'watch' a repository, they will get notifications from any activities in that repository, including changes to the code base, new pull requests, new issues, and comments on PRs, issues, commits, or code. These notifications appear in three potential places: the user's News Feed, as a notification in their Notifications list, and as an email. Alternatively, a user can choose to not watch a repository, meaning

[4]`https://help.github.com/articles/using-git-blame-to-trace-changes-in-a-file/`
[5]`https://help.github.com/articles/news-feed/`

they will only be notified if they are specifically mentioned in a comment or commit, or in comment threads that they have participated in. Finally, they can choose to ignore a repository, which will prevent them receiving any notifications for that repository, in any capacity.

The last important feature that promotes openness and transparency is the activity monitor on a repository. On the activity monitor, a user can look at who has contributed to the repository, for example, graphs of number of commits from each person, or lines of code added by each person. As well, numerous features allow a user to get a more wholistic picture of the activity on the repository, such as what time commits are normally made on each day, how each branch is handled in comparison to the master branch, or how many times the repository has been visited[6]. Overall, this feature allows a user to see and compare the activity among all collaborators.

3.7 Why GitHub for Education?

Although GitHub has focused on code and project management for software development, other domains that involve collaborative work, such as education [27], have recently begun to take advantage of its features. However, the use of GitHub a different context like education might require instructors to explore alternative teaching and learning activities, or require GitHub to be repurposed to better accommodate teaching activities. Jim Baker, a senior developer and University of Colorado Computer Science lecturer, shared his experiences with GitHub:

> *"We had a great experience using GitHub to support a collaborative workflow for the 70+ students in each of the 2 semesters of my CS course."*

He elaborated:

> *"Pull requests (PR) are the heart of the GitHub workflow, and we took advantage of PRs, including task lists so that students could report on their work in progress and get over initial humps. Any merged PR got extra credit(!). Because the course had been improved in some way—this seemed like an interesting standard for giving out extra credit. Consequently, we mostly didn't merge PRs for labs, except for bug fixes, but we were always on the lookout for* ***better solutions than ours****. PRs were also merged*

[6] `https://github.com/blog/1093-introducing-the-new-github-graphs`

for extra credit, such as ***corrections of my course notes****. Next fall we expect to have* ***autograding*** *implemented as a form of continuous integration, by running against the PRs through postcommit hooks."*

Other educators have since introduced GitHub into their classrooms and shared their experiences. In 2010, Luis Felipe Borjas[7] posted about using GitHub organizations[8]—a way to simplify the management of group-owned repositories—to manage class projects. He also suggested that teachers can use private repositories for exams or homework assignments that students could push to.

In 2011, David Humphrey blogged[9] that he asked his students to use Git/GitHub and highly recommended that other instructors do the same. He claimed that although it was a little painful to learn Git/GitHub at the beginning, the payoff would be huge. *"One of the great things about Git in an educational setting is that you don't need to rely on institutional IT, which, in my experience, is never agile enough to help you with revision control. You can put repos on laptops, use USB keys, use DropBox, use GitHub, etc. You don't have to wait for someone to set up a server and make you accounts, don't have to deal with permissions, or any other nonsense that comes with centralized revision control systems."*

Apart from these testimonies of using GitHub, as well as other similar, developer-focused tools highlighted in chapter 2, there are now platforms based on the GitHub model that are targeted for education. Created in 2013, Coursefork[10] is described as *"GitHub for course creation."*[11] It is a platform for open-sourcing and collaborating on educational material, where educators can upload course materials and allow others to create copies of courses and modify or share them.

Moreover, GitHub has recently launched a portal specifically for using GitHub in education[12], which aims to help both students and educators. Students can apply for an account with free private repositories, while teachers can also apply for an organizational account for their classes, granting them a number of free repositories for students. They also have a student developer pack, which includes free services or discounts from various companies, such as domain hosting or live programming help. Finally, they include a classroom guide where instructors are given guidelines

[7]`http://lfborjas.com/2010/10/30/git-classroom-exams.html`
[8]`https://github.com/blog/674-introducing-organizations`
[9]`http://vocamus.net/dave/?p=1358`
[10]`http://coursefork.org/`
[11]`http://opensource.com/education/13/9/coursefork-education-tool`
[12]`https://education.github.com`

and suggestions for using GitHub to manage their courses, including an example course repository. Given that they released this guide, along with the research where educators are choosing to use GitHub and similar tools for educational purposes, suggests that GitHub may provide numerous benefits to education.

Chapter 4

The Instructor Perspective

As part of this work, we conducted two studies (in two phases) to explore the use of GitHub in post-secondary courses. This chapter highlights the first of those two phases, where we interviewed instructors who were early adopters of using GitHub for educational purposes. The goals of the study carried out in this phase were to learn how educators were using the tool and for what reasons, as well as to understand what they believed to be the benefits and the challenges are of using GitHub in this context.

The findings from this study contributes to answering the overarching research question regarding GitHub in education: is this tool viable for education and how does it compare to traditional learning tools? By exploring the instructor's perspective, we discover what GitHub offers to instructors beyond what traditional learning tools can provide. Moreover, exploring these questions allowed us to discover which processes and activities educators support using GitHub, as well as how GitHub supports them. Given that educators select and administer the learning tools to use in their courses, it was important to gain insights from their perspective.

This study was conducted in collaboration with Alexey Zagalsky, Dr. Margaret-Anne Storey, Yiyun Zhao, and Weiliang Wang. My personal contributions include conducting a significant number of the interviews during data collection, being involved in the coding process during data analysis, and summarizing the findings and the discussion in collaboration with Alexey Zagalsky and Dr. Margaret-Anne Storey.

4.1 Research Questions

We devised a number of research questions geared towards exploring the use of GitHub in education from the perspective of early adopters. The research questions addressed in this study include:

RQ1: How does GitHub support learning and teaching? We investigate how GitHub is used within the education domain and for what purposes. Learning how educators take advantage of GitHub's unique features such as pull requests and its transparency features contributes to answering our overarching question regarding the viability and effectiveness of GitHub as an educational tool.

RQ2: What are the motivations and benefits of using GitHub for education? Based on testimonials from our study participants, we explore the motivations for GitHub use in education and the possible benefits it might bring to support learning. We also look at specific GitHub features that are being used in this context.

RQ3: What challenges are related to the use of GitHub for education? We examine the challenges educators and their students face when using GitHub to support learning and teaching. We provide specific examples based on interviews with educators. Exploring these challenges help shape the recommendations for educators wanting to use GitHub or similar tools for educational purposes, addressed in Chapter 6.

4.2 Research Design

In this study, we interviewed 15 instructors. Here we were able to thoroughly investigate the usefulness and potential of GitHub in education. Through iterative analysis of the data collected, several themes emerged around the motivations for and challenges of using GitHub to support learning. These themes informed the next phase of our research: a follow-up survey sent to interviewees from the second phase and to other educators using GitHub for education. The goal of this survey was to receive interviewee feedback on our interpretations, but also to gain additional perspectives from other educators that use GitHub.

4.2.1 Participants

Our study targeted lecturers and professors in higher education who use or have used GitHub to support teaching and learning. As our study aimed to investigate diverse populations as well as GitHub's usefulness in non-technical courses, we wanted to hear from lecturers and professors across domains.

Interview Participants

Table 4.1 shows some details about the courses each of the interviewees taught using GitHub. For each course taught, we list a summary of the students' GitHub knowledge, the type of course, and the number of students enrolled.

Table 4.1: Information on courses taught by interview participants while using GitHub.

P#	Course Type	Knew GitHub	Course Size(s)
1	CS	Yes	55
2	CS	Yes	40
3	CS	Yes	85, 130
4	Humanities	No	11, 40
5	Humanities	No	17
6	CS	Yes	60, 100
7	CS	Yes	50-237
8	CS	Varies	60
9	Sciences	No	235
10	CS	Varies	450
11	CS	Varies	20, 40
12	Statistics	No	40
13	CS	No	20
14	CS	Yes	8
15	CS	No	30-32

The participants represent a broad list of universities: University of Victoria, University of British Colombia, McGill University, University of California at Berkeley, University of California at Davis, Columbia University, The City University of New York, Harvard University, The University of Texas at Austin, Federal University of Pernambuco, and Delft University of Technology. Furthermore, one of the participants was from a company that teaches code to entrepreneurs and CEOs in Paris.

Follow-up Survey Respondents

We sent a follow-up survey to the interviewees to get their feedback on our interpretation of the interview findings. We also publicly broadcasted the survey using social media[1] and used snowball sampling to garner responses from additional educators. Eight of the interview participants and an additional seven new respondents completed the survey, for a total of fifteen responses.

4.2.2 Data Collection

Data collection began as part of a software engineering course in the University of Victoria. We emailed invitations to lecturers and professors that use or have used GitHub to support teaching or learning to invite them to interview. Potential participants were recruited in several ways: by contacting blog authors who shared their experiences of using GitHub in the classroom; by posting an invitation on Twitter[2]; and through snowball sampling, where interviewees could suggest other colleagues.

Upon agreeing to participate, instructors were sent participant consent forms to sign before interviews took place. The interviews lasted 20-60 minutes and were conducted face to face or with Skype. Audio was recorded and the interviewer took notes. The interviews were semi-structured based on 18 guiding questions (our interview form is available in Appendix A), and the interviewer *dug deeper* with additional questions as deemed appropriate. This supported the exploratory nature of our study and allowed us to examine interesting and unexpected insights.

We also interviewed John Britton, an education liaison from GitHub (this interview form is also available in Appendix A). This interview provided us with GitHub's corporate take on the use of its tool in education.

4.2.3 Data Analysis

Our analysis of the interview data followed qualitative data analysis guidelines [42, 61] and included the following stages: (1) transcription of the data recorded; (2) organization of the data into easily retrievable sections; (3) familiarization by reading and re-reading the data, making memos and summaries; (4) reading the data and labeling segments, i.e., coding; and (5) identifying themes or emergent concepts through dis-

[1]`https://twitter.com/alexeyzagalsky/status/471053256718692352`
[2]`https://twitter.com/alexeyzagalsky/status/465914075348619264`

cussions among the researchers and engaging in re-coding to develop more well-defined categories. After refining and merging some of the themes, a list was compiled. A theme was added to this list only when a concept was discussed by multiple interviewees. This process was performed iteratively for each category.

4.3 Findings

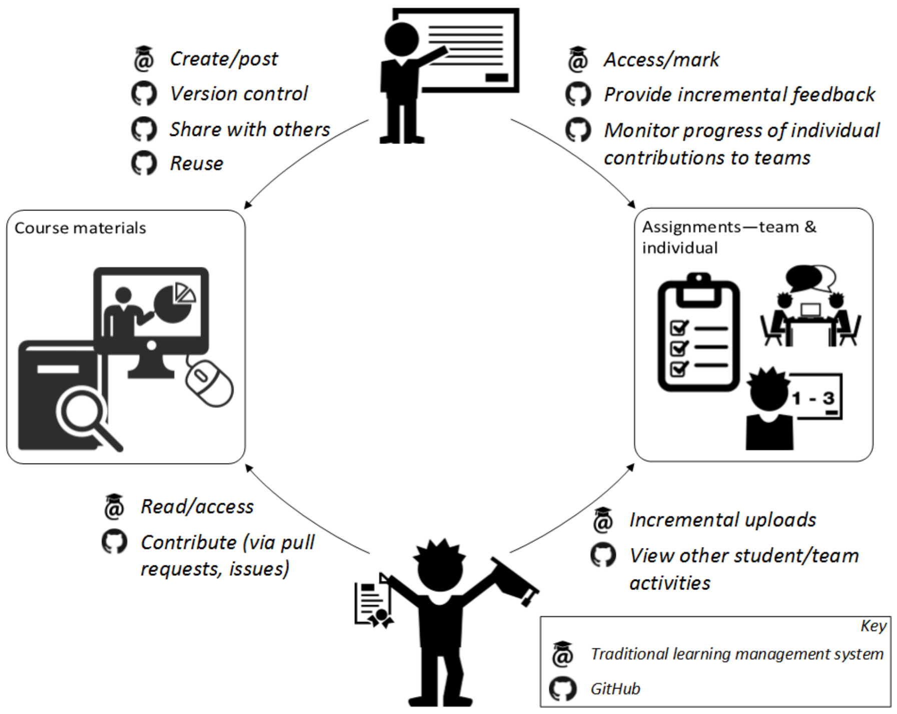
In this section, we address our research questions and present the themes that emerged from the interviews. To illustrate the different aspects of each theme, we provide selected quotes from the interviews, where each participant is identified by an anonymized identifier (P#).

4.3.1 RQ1: How does GitHub support learning and teaching?

Given that education is an emerging use case for GitHub, we first sought to understand how educators use the tool to support the learning and teaching activities in their courses. Until these interviews, our knowledge of GitHub use in education was limited to the blog posts highlighted in Chapter 3. The educators interviewed, however, are early adopters of GitHub as an educational tool, and as such, these findings help shape our vision of how GitHub acts as a learning tool, ultimately contributing to our understanding of the viability of the tool in an educational context.

There are two ways that our interviewees utilized GitHub in the classroom: as a submission platform, and as a way to host course content. These two basic uses mirror how teachers use typical Learning Management Systems (e.g., Moodle and Sakai) [46]. However, the implications of how they use these features differ substantially. Figure 4.1 gives a high-level overview of the interactions that are possible between teachers, students and content, and shows the additional interactions that the use of GitHub natively supports over traditional learning environments. For example, as seen in the bottom-left quadrant of the figure, while traditional LMSes support students reading and accessing course material, GitHub also allows them to easily contribute to the course material.

Figure 4.1: The additional features GitHub provides in addition to the features provided by traditional LMSes.


Create/post
Version control
Share with others
Reuse
Access/mark
Provide incremental feedback
Monitor progress of individual contributions to teams
Course materials
Assignments—team & individual
1 - 3
Read/access
Contribute (via pull requests, issues)
Incremental uploads
View other student/team activities
Key
Traditional learning management system
GitHub

GitHub as a Submission Platform

Many of the interviewees used GitHub as a place for students to host their work and submit their course assignments and projects, [P1, P2, P3, P4, P5, P6, P7, P11, P12, P13, P14, P15]. The benefits of using it as a submission platform are further discussed in the findings of RQ2.

Using GitHub as a submission platform was accomplished in one of two ways[3]. Some interviewees set up a base repository for the class and had each student fork it—everyone who forked the base repository could see all other forked repositories as well. This allowed students to cross-reference different solutions and encouraged student collaboration (e.g., by using PRs) and peer learning. *"When you do a pull, you can see what the others in the seminar are doing. For example, a student wrote a python script, and others want to use it, so they can just grab it and use it."*[P4]

Other interviewees set up private repositories for each student, and individual students could only see other repositories when explicitly given access. For P5's course, *"Students have separate repositories, and it's private so that people cannot see what*

[3]https://education.github.com/guide#4-set-up-the-repositories

they are working on." With this option, repositories and permissions are manually configured, which may be difficult in courses with a large number of students.

Hosting Course Material

Many interviewees used GitHub to host and deliver course materials [P4, P5, P8, P9, P10, P11, P12, P13, P15], including syllabi, slides, notes, reading materials, exercises and homework. It was even used to host the actual course Website, making GitHub *"a bulletin board."*[P10] The basic mechanism to host course Websites is by using GitHub markdown files, or for more traditional Websites, by using GitHub Pages[4] to host the material directly from one's repository.

With course materials hosted on GitHub, students and other educators were able to suggest course improvements [P1, P10, P11, P12] by submitting an issue or a PR. *"Students would actually find corrections and things and they would send me pull requests... But there was some sense of a community that was at least, certainly invited to collaborate on the course material itself."*[P12] As such, interviewees were able to facilitate student contribution to the course material itself using GitHub, a use case and benefit that GitHub offers beyond many traditional systems.

4.3.2 RQ2: What are the motivations and benefits of using GitHub for education?

We explored why educators chose to use GitHub and how they (and their students) benefited from its use. While GitHub is still an emerging tool for education and many benefits have not yet been fully realized, we extracted several themes from our interviews. Recognizing the benefits of GitHub use can justify its use in education by learning what GitHub provides beyond traditional learning tools.

Transparency of Activity

Using GitHub as a submission platform made it easy for our interviewees to monitor student progress, activity and participation. GitHub has numerous features that support transparency. For example, GitHub allows users to see the history of activities. As P1 mentioned, *"You really see the full history of how the document comes into being, including all the discussions, the former versions. And the groups can look at*

[4]`https://pages.github.com/`

Table 4.2: Summary of participants' motivations and benefits of using GitHub for teaching purposes.

Benefit	*Description*
Transparency of Activity	Instructors can see a history of student activities, use the graph view, and the News Feed.
Encourage Participation	Instructors can encourage students to submit issues or pull requests connected with the material.
Reuse and Sharing of Course Materials and Knowledge	Instructors can version their materials and share it with other educators. Students can also share work with each other.
Industry Relevance	Instructors and their students have had prior experience with using the tool, and students can share their class work with prospective employers.
Ease of Use	Instructors found it easy to set up a course on GitHub and update the repositories as needed.
Free Academic Licenses	Instructors can create free public repositories and can apply for an educational account that provides them an organization, which provides free private repositories.
Shared Space and Course Versioning	Students can share course notes and educators can fork a repository of a previous instance of a course.

each others' documents and see them emerge. I can see how they are creating it, so I can monitor who's active, working in certain teams, which is also handy, practical."

GitHub also provides a graph view that visualizes a summary of project activities. This allowed teachers to easily gain a high-level view of student activity. *"You can see who is doing what, you can see how people are sharing work because you have the commit history."*[P4] This same feature supported teachers in identifying students that were not participating as expected. *"I would go into their team repository, and...they have some graphs in there that show you what's the activity level and who's most active. I would use that to get a sense for what was happening because inevitably, you would have one of the students...that's not doing anything and you want to get some hard data on exactly what he isn't doing."*[P6]

Furthermore, GitHub has a News Feed that teachers used to keep up to date with activity. By watching the News Feed, they could catch problems early and saw how frequently students were participating in the course. *"Personally, what I did* *is that I subscribed to all [the students'] repositories and followed the feed of those repositories."*[P2]

However, GitHub allowed people to do more than just passively observe progress. Students opened issues and tagged them with the teacher's name to get their attention. *"You could watch the commits go by. They can ask for help by opening issues and tagging us, so that worked quite beautifully for monitoring and in some cases actively providing advice on group project work."*[P12] This kind of support encouraged a style of communication between the teacher and student that is not available in traditional Learning Management Systems.

Encourage Participation

The transparency features discussed above allowed interviewees to encourage student course participation and contribution to the hosted course materials. With traditional environments, students would need to contact the instructor directly to suggest changes to the materials. Using GitHub, a student could suggest changes by submitting an issue connected with the material. Or, they could make changes themselves and then submit a pull request which the teacher, if they agreed with the changes, could simply accept. The history of these actions helped the teacher keep track of who participated, and by its visibility, also encouraged students as their actions were logged and possibly used to improve their grades.

P1 spoke about how these features helped to persist the effort that students put forth: *"Yes, [using GitHub encouraged student participation] because everything you do is also recorded. It stays there forever, it's persistent. So your comments are persistent, your activities are persistent. So you work, you get rewarded, and if you don't work it's visible... in GitHub it's a little more transparent."*

Some interviewees felt that using GitHub helped them encourage participation, even in indirect ways: *"I will use the logs [that students put on GitHub] as materials for discussion in class. That can encourage participation because you know you are committing something that can be discussion material."*[P5]

Although few mandated the use of GitHub's issue tracker for their classes and group projects, some found novel uses for this feature. P12 and P14 had students use it when they needed help, and otherwise used it as a simple communication method between students and markers. P1's students participated by discussing an issue they had with a deadline: *"the students didn't like [a deadline], so they opened an issue* *on GitHub: 'Can we move it?'... So everyone responded to that issue on GitHub, and then at the end I said, 'based on the discussion, it is going to be on Sunday."'* These features supported student/teacher discussions—communication was recorded and tied directly to the relevant course content.

Reuse and Sharing of Course Materials and Knowledge

Another important advantage GitHub provided our interviewees was the ease with which materials and knowledge could be shared and reused, by both students and course instructors.

Some of our interviewees' students shared work and materials with each other. In P4's course, a student's python script was visible to others, and therefore, easily shared. The ability to make content either public, private amongst collaborators, or completely private, gave our interviewees' students an easy way to share with others or contribute to their work. For example, P1 noted that this visibility had the useful side effect that his students gave each other feedback on their pull requests and reports.

Moreover, our interviewees used GitHub to share courses and their associated materials with other educators. Although this is also a feature in many learning environments, they lack a way to visualize how a reused course has changed from the original content. Seeing when courses are forked and how they differ from one another is a key feature in GitHub. *"We were developing all the content from scratch... the various teaching staff had to collaborate and develop the material, and so GitHub was a nice fit for that... We had two repositories: we had a private one and a public one, and so we could sort of have private conversations among the staff to refine the content and then it was pretty easy to just migrate it over once it was ready."*[P10]

GitHub's access control features allowed the instructors to version their material and choose whom to share it with. *"Some of [the material] is shared with the world, and some of it is shared with other instructors of other universities. So for example, when we create new exam questions and stuff like that, we try to keep those [repositories] private for a little while so other instructors can use them. But we're versioning all of it."*[P7]

Interestingly, unregistered students and other GitHub users would also visit these public repositories: *"We kept everything public from the beginning, so it ended up getting a lot of attention outside of the course. I think the repository has something* *like 200 stars on GitHub right now and as far as I can tell most of those stars are from people who didn't take the course."*[P10]

Industry Relevance

As working with Git and GitHub are relevant skills for industry, this theme emerged as both a motivation to use GitHub in courses and a benefit of using the tool. This relevance to industry meant that some of our interviewees and their students had prior GitHub experience, and as a result, were motivated to use it in class. For example, P12 had already been using GitHub regularly and explained why they chose to use GitHub in their courses: *"... partially to unify what I'm doing in the research side of my life with what I do with the teaching side. Like once you've taken the trouble to learn [GitHub] and get it all set up, you start to see lots of areas of your life where this workflow would actually be very useful."*

Meanwhile, we saw some instances where students who had prior experience using GitHub encouraged their course instructors to use it for class. As P1 told us: *"So actually, some students came to me, and they said: you should watch this video on how GitHub uses GitHub to build GitHub."*

Knowing how to use Git and GitHub is a significant benefit in certain fields, particularly in computer science and software engineering[5]. In fact, P12 felt that learning to use GitHub was required to progress within their field: *"To prepare this cohort of graduate students for computational work, they should know how to use these tools. This is very much what people do these days in [statistics], and so I actually consider it a completely valid pedagogical goal in it of itself."*

By using GitHub for course work, students further benefited by having an end product that could easily be shared with prospective employers. *"you can use it sort of as an online resume... GitHub allowed you the opportunity to convert [your class work] to a public project so that employers for example could see what code you've been writing."*[P6]

Ease of Use

GitHub's administrative functions are relatively simple and easy to work with: getting students set up for the class, setting privacy options, and creating repositories. *"I don't think it saved time at the beginning... because I was starting an entirely new*

[5]`http://techcrunch.com/2014/07/23/modernizing-computer-science-education`

approach to doing this. But once it was up and running, the next semester would have been quite simple to administer because I knew the process."[P6] Although GitHub has a definite learning curve, it can be mitigated by sharing practices among practitioners. *"I worked with [the next instructor of the course] a bit on explaining how it worked and what we did with it, so she found it pretty easy to get started too."*[P6]

Despite the learning curve and some technical difficulties, several interviewees mentioned that GitHub was relatively easy to use and administer, both on its own and compared to existing university systems. *"For me it was the ease of updating the class schedule and course notes. The class schedule is something that is extremely painful to update on the university Website... Each course is a little window and you have to click arrows to move them up and down to reorder them, or something really horrible like that. And [with GitHub] it was very easy to just go into a markdown, add a link, and hit push."*[P9]

Free Academic Licenses

Generally, GitHub allows all users to create free public repositories. However, an additional benefit mentioned in the interviews [P3, P6, P7] is that GitHub provides free academic licenses. Students and educators can apply for a free micro plan that allows private repositories. Educators can also apply for a free organization account[6] that facilitates team management and administration. The implications of these free public accounts go beyond benefiting face-to-face instruction, allowing teachers to *scale-up* traditional education and create online courses that will benefit part-time and remote students.

Shared Space and Course Versioning

By hosting course materials on GitHub, students could easily share course notes, references or other material for the same class, and educators could quickly share [P13] or duplicate course resources. *"The simplest thing I've been using [GitHub for], especially from an instructor's view, is to duplicate the course Website for different semesters."*[P4] Therefore, when teaching the same course again, preparing the course materials was simple: participants used the same repository or forked it. *"If I improve the material in some way, I'd just keep it there...And if someone wants an older version of the repository, they just need to get it. However, if I did want to create a*

[6]`https://help.github.com/articles/what-s-the-difference-between-user-and-organization-account`

separate course with just part of the material of the original course, I would [fork the repository]."[P11]

4.3.3 RQ3: What challenges are related to the use of GitHub for education?

By exploring what challenges educators experience when using GitHub, we learned the weaknesses of GitHub as a learning platform. These results inform our recommendations for instructors who want to use GitHub for these purposes, listed in Chapter 6. We have grouped the challenges our analysis revealed into five themes: shared knowledge base of suggested practices, barriers to entry, support for additional formats, external restrictions, and large-scale management.

Table 4.3: Summary of participants' challenges with using GitHub for teaching purposes.

Challenge	*Description*
Shared Knowledge Base of Suggested Practices	As of the time of this study, there was no shared knowledge base where instructors wishing to use GitHub for a course could refer to for tutorials and suggestions.
Barriers to Entry	There is a learning curve associated with using Git features such as merging and merge conflicts.
Support for Additional Formats	GitHub does not support viewing or rendering certain formats such as PDF documents and Powerpoint Presentations.
External Restrictions	Instructors experienced restrictions such as university policies on where student work and data are stored.
Large-Scale Management	Some educators had difficulties when setting up courses with large numbers of students.

Shared Knowledge Base of Suggested Practices

Interviewees showed a need for *a shared knowledge base* of suggested practices on how to use GitHub for educational purposes. P6 went into great detail about their struggles in this area: *"It's a different use in a team at some startup or what have you. And it's not clear how exactly you should, for example, enforce a methodology for the students for working. Should you use pull requests? When should you file issues?*

Divide up the repository, should you each have an individual repository as well and then fork the main project? There's a lot of good evidence and good data for how you should work in Git for your software development project, for commercial projects. For educational projects, they're quite different because they're short in duration, there's 4 or 5 people working on it that aren't very experienced with software development."

Educators wished for easily accessible *how-to guides* or shared experiences by others who have used GitHub. *"Maybe some documentation on best practices or some kind of shared knowledge base that would say here's what so-and-so at UC Irvine is doing with GitHub."*[P6] Interestingly, the GitHub education Website contains a guide that could meet some basic needs, but our interviewees did not know of this relatively new feature.

Barriers to Entry

At its core, GitHub is a Git-based system, and while it provides a simple Web interface, tasks that involve collaboration require an understanding of Git and its command-line arguments. In particular, dealing with merges and conflicts can be challenging: *"To get the most out of GitHub, you need to understand Git. Even if you just use it to edit documents together, you will get conflicts once in a while. And if you want to use the review mechanism, then you want/should use the pull requests. That requires some pretty deep knowledge of Git still. If you use it the right way it is simple, but somehow with Git you end up with conflicts, and if you don't understand it, it's magic."*[P1]

Difficulties using Git are experienced by technical novices and software developers alike [55]. Our interviewees reinforced the need to improve accessibility for novice users or users with a limited technical background: *"Largely, the biggest challenge is to lower the learning curve, not on the high end, but the very low end for people who are new to it to have much gentler learning curve."*[P5]

Support for Additional Formats

GitHub's file and format support is another main challenge mentioned by the interviewees, specifically the lack of support for formats widely used in education, such as PDF and LaTeX. The ability to view or render these formats directly on GitHub, similar to Markdown rendering, would be very helpful. But more importantly, there is a need to support the powerful features Git has for text files: *diff* and review functionality, and the mechanism to add inline comments to a file without altering the original file. For example, when an instructor wishes to mark the students' assignments or reports, they are forced to download and comment on each file separately (e.g., using the comment feature in a PDF file), which also alters the original file. Another option is to use the *issues* mechanism of GitHub to provide comments, however, this mechanism can't reference a specific place in the file and may be tedious if the instructor wants to add many small comments to a submission.

Interviewees also mentioned GitHub's poor support for slides: *"You can not easily view slides in GitHub, because the PDF is too large and you have to download it, it's a bit cumbersome... in a course you always have presentation material, so you want some sort of integration with slides here. They [GitHub] have Speaker Deck, so I want Speaker Deck integrated with GitHub."*[P1] Being able to *diff* slides would be highly beneficial.

External Restrictions

Interviewees also mentioned external restrictions that limit or prevent them from using GitHub for education. External restrictions can be local restrictions (e.g., university policy) or global restrictions (e.g., regional publishing licenses). *"We have LMS systems that we use, and in a way it's very important to use them because they're authenticated by the university. So things like student's grades and stuff like that you don't (store on GitHub), because of the rules in the US. You cannot store it anywhere else."*[P3] Another teacher also shared this concern: *"Ethically, I wish I could know where the server is, where I am pushing all my material."*[P4]

Knowing the server's location plays a significant role when educators decide whether to use GitHub. Not only from the ethical aspects, but also for copyright reasons. As P5 mentioned: *"Copyright is a big issue. For instance, we are working with a novel. In Canada, that novel is in public domain so it can be accessed online, but not in the United States."*

Large-scale Management

Educators are challenged when managing courses with large numbers of students, teams, projects or issues. *"One thing I'm not so happy with, is the way group management [works]. So managing teams, different sizes, I have an organization for the students and the organization has 50 members and I need to create different groups,* *and different roles, and different access... I think that could be done... [in a] more convenient way."*[P1]

P6 further discusses the challenges with managing teams and repositories: *"The thing that I think was missing... was more management for the administrator: assigning people to teams, assigning teams to repositories, finer grained permission control, I think [that] is something that could have been useful."* However, the cause of these difficulties might not be an issue with GitHub, but perhaps the lack of *how-to guides* and suggested practices by others.

4.3.4 GitHub's Perspective

To gain insights into GitHub's perspective on the topic, we interviewed John Britton, an education liaison at GitHub and one of two engineers working full time to support the use of GitHub in education. GitHub's main goal in this regard is to make the tool easier to use, and to make sure users are aware of what resources are available to them to meet their needs and solve their challenges.

Regarding GitHub's benefits for education, John described the importance of providing students experience with a tool relevant in industry, a benefit supported by our findings and by literature [28]:

> *"We're working with students and teachers on using GitHub. It's essentially leveraging those tools in the classroom, so you can get a classroom experience that's more similar to what people in the industry are using, as software developers."*

However, in the scope of education in general, the learning curve is a challenge for educators outside highly technical fields such as computer science. This challenge suggests that GitHub as an educational tool may not be ready for use in non-technical fields:

> *"When I first started, I definitely was targeting all forms of education. But I think that it's most applicable in computer science, software engineering, and technical fields. Maybe someday when the tools are easier to use and... require less technical knowledge upfront, it will be much more useful to non-technical fields, or to other types of educational stuff. But I think that Git as it stands requires a certain amount of basic knowledge of computing."*

He described how educators used the pull request method of submitting work and assignments, comparing GitHub's submission method from how submission is handled traditionally:

> *"There's different groups. People who know Git and GitHub already and just want to use it in the classroom—they're totally on board with using pull requests right from the beginning. The group of people who's like, 'What's Git? What's GitHub?', they have a little bit harder time getting into the mindset of students submitting their code as a pull request. Rather than submitting an assignment with a zip file, you create a branch and make a pull request, and then grade the student on the pull request."*

John concluded by describing how GitHub plans to assist educators. He emphasizes the flexibility in using GitHub for education and describes the creation of materials that since became available in the GitHub for Education website[7]:

> *"I think we're going to do more of those [stories of existing use cases], along with technical documentation, or technical information on 'Here's what they're doing. Here's how you can do it too'. But as far as...we don't have a set, like, this is the way, right? There are multiple ways to use GitHub in the classroom, and it depends on what goal you're trying to achieve."*

The education use case is important for GitHub. While their focus remains on its use in software development, they provide dedicated support and resources for educators to take advantage of the tool.

4.3.5 Follow-up Survey

After conducting the interviews, we sent a follow-up survey to both interviewees and the public. We wanted to see if respondents agreed with our interpretation of the interview findings. However, the number of respondents (15) was too low to confidently validate our study. Regardless, the survey supports many of our findings, with respondents mostly agreeing or strongly agreeing to many of the uses (as a submission platform and course material host) and benefits (simpler than current university systems, reuse and sharing of material, and facilitating collaboration). Interestingly,

[7]`https://education.github.com`

many respondents from this group of educators stated that they didn't struggle with the challenges that emerged in our interviews (e.g., technical difficulties limiting educator GitHub use, external restrictions, and managing large groups). However, while strongly disagreeing that technical difficulties limited their own use of GitHub, most respondents agreed that technical difficulties limited their students' use of GitHub.

4.4 Discussion

Our study uncovered how some educators use GitHub to support learning and teaching, while extending or even replacing traditional LMSes. However, the implications of our findings go beyond GitHub itself. The emergence of *"the GitHub way"*—an open collaborative workflow—within education is transforming the traditional e-learning model, as seen in figure 4.1, and will better support socio-collaborative learning environments [38] of the future.

4.4.1 Comparing GitHub to Traditional LMSes

GitHub was not designed as an LMS, but our study shows it can be used as such. Judging the use of Git and GitHub based on LMS features [41], GitHub supports many of the important learning components (e.g., assignment delivery and submission), but it also lacks certain features (e.g., grading management tools). In examining the model published by Malikowski *et al.* [46], we come to similar conclusions, where platforms like GitHub have the capability to support all five categories of the model. However, student and course evaluation may require additional work, as well as developing programs to detect and automatically grade student pushes.

Lane [43] discusses two sides of the Course Management System (CMS). From one side, CMS are a toolbox for educators, providing a default course structure with managerial and administrative features. From the other side, integrated commercial CMS are a trap that limit faculty creativity, are difficult to customize (sometimes this involves additional cost), and they make it hard to accommodate individual teaching styles. As GitHub is adopted for more and more courses, it remains to be seen if it suffers from similar or different issues to mainstream CMSs.

4.4.2 Going Beyond Traditional LMSes

Beyond the features discussed above, GitHub provides a number of other tools and features that allow educators more novel possibilities.

Version Control

According to the interviewees that teach technical courses (such as computer science), students destined for careers in software and information technology benefit from learning to use version control systems [6]. This reflects the benefits mentioned in previous work on these types of systems [56]. At the same time, version control allows educators to see the final results as well as the processes students used to produce the results [26]. However, this is a double-edged sword as the technical aspects of version control, specifically Git, is one of the main challenges mentioned by interviewees.

Transparency & Awareness

Kreijns *et al.* [39] discuss the shortcomings of contemporary Computer-Supported Collaborative Learning, particularly in group learning, social construction of knowledge, and the learning process itself. They link these problems to the impeded social interaction of CSCL environments caused by two major pitfalls: taking social interaction in groups for granted, and the lack of attention paid to the social psychological dimension of social interaction outside of the task context. Transparency of activities allows GitHub to address this by embedding social affordances and by supporting group awareness, as proposed by the suggested theoretical framework [38].

However, only a few of our participants mentioned using the transparency and awareness features to promote collaboration among students. If taken advantage of, GitHub's transparency could be beneficial for collaborating students [17], creating awareness of each others' activities that could, in turn, support collaborator creativity [23]. As such, GitHub is a good example of a transition from a space to a place, as it transitioned from a hosting service that provided space for projects, to a place where work, activities, ideas and discussion can be shared [18].

Student Engagement & Participation

GitHub enables students to contribute to course material or work done by other students with the use of the pull request mechanism—a novel way to participate in courses. GitHub provides more in-depth ways to communicate and collaborate, and provides teachers and students with a way to gain social group awareness and get information about what group members are doing, who they are communicating with, and how they are contributing [32]. Thus, GitHub enables a participatory culture [34] where students can create, edit, and share in such a way that they feel their contributions matter.

Indeed, one of our interview questions asked whether the use of GitHub encouraged student participation. Interview participants mentioned the impact on student engagement and participation several times, however, except for some specific examples, the interviewees didn't have any metrics to measure this. In that sense, our study is limited and leads to future work to investigate if GitHub encourages student participation.

Reusing and Remixing

Some interviewees described the ability to pull from other sources or old content as useful for creating and organizing course materials. The main way to reuse materials as seen from our findings was to fork an old instance of a course. However, there were also instances of educators collaborating with each other to create course materials, or sharing material with instructors in other universities so that content can be remixed as an instructor sees fit. With current learning management tools, as educators ourselves, we have noted it is extremely tedious to version and share course materials. We end up with each new course as a new entity, losing the history of where we got the materials and who contributed to them. GitHub mitigates this through the use of the *Blame* feature, which visualizes the providence of file changes. In time, we expect to see large networks of educators contributing to the same course, and know who contributed to the material and how.

4.4.3 Not Just GitHub

We note that our findings extend beyond GitHub and apply to related environments, such as BitBucket[8], that provide similar social and collaborative features. In fact, a few of our interviewees also mentioned using BitBucket and described similar experiences with that tool. The lessons learned may be applicable to other social hosting tools as well.

4.4.4 Limitations

We collected data from the educator's point of view only. However, the student's point of view is also crucial and may provide additional support to our findings or reveal new insights. Additionally, the participants of our study were recruited under the condition that they already use GitHub to support learning, where it may imply that their experience had to be (somewhat) successful. This means that our study may have missed the challenges faced by potential participants who failed to use GitHub to support teaching. Additionally, our participants used GitHub at different periods of time, thus their experiences may differ and this may affect our findings.

This chapter presented the first phase of this work involving a study in which we investigated why and how GitHub is used for education, identifying the potential benefits and challenges that follow. We note, however, that we have discussed this from the perspective of instructors, where the findings apply, for the most part, to the instructor and not the students. In the next chapter, I highlight a study which focuses on the student perspectives of using GitHub in education.

[8] `https://bitbucket.org/`

Chapter 5

The Student Perspective

In the previous chapter, we interviewed educators to discover why and how they use GitHub to augment their classes, and gained a variety of insights on the benefits and drawbacks of using such a system. 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. They 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. With that, I aimed to uncover more about the student perspective, as well as to discover what challenges they may experience or concerns they may have with using GitHub for their courses.

This phase explores how students perceive GitHub as a learning tool. To do so, I conducted a study where GitHub was used to support the teaching and learning activities in two technical courses. The instructors would utilize GitHub in ways similar to those described by the instructors we interviewed in the previous chapter—to host course materials, projects, and assignments, and to facilitate discussion. In this phase, I used exploratory questions to interview students and learn how they perceive GitHub's effectiveness as a platform for education and for their coursework.

As learning tools become more focused on student participation and contribution, it was important to place a focus on the students. By exploring the student perspective, we can determine how GitHub as a tool might help enable a participatory culture [34] or serve as a tool to meet the needs of 'the contributing student' model [30]. This may impact the development of future tools, particularly if the benefits of using GitHub are large enough to consider implementing features to support similar activities in educational tools.

5.1 Research Questions

This phase explores 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.

RQ4: From the student perspective, how does GitHub compare to traditional Learning Management Systems? Specifically, we aimed to discover student perceptions on currently used Learning Management Systems (LMS) like CourseSpaces (Moodle) and Connex, specifically pertaining to GitHub's potential as such a portal for student interactions in their courses.

5.2 Research Design

We used a qualitative approach to study the student perspective of GitHub use in education. As Creswell [14] suggests, a qualitative and exploratory approach best suits research when a concept or phenomenon requires more understanding because there is little pre-existing research. Moreover, a qualitative approach is consistent with the previous study discussed in Chapter 4, and this is important because the aim was to investigate similar outcomes from the same phenomenon, but from other perspectives.

Yin [70] 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 [20].

Specific to software engineering, Runeson [60] 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, I 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.

5.2.1 Recruitment

The participants in this phase included the main stakeholders in technical courses: the professor (the main instructor), lab instructors, and students. As the learning tools used in courses directly involve and impact these stakeholders, I wished to obtain their perspectives and opinions to answer our research questions. The aim was to explore their perspectives while the course was ongoing so that they could recall recent experiences and provide their opinions on GitHub as a learning tool.

For this study, we were opportunistic in finding cases and sought instructors who could and were willing to try using GitHub. I was 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 [70]. 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).

When the term began, I attended one of the first lectures to describe my goals and to recruit students to participate in the study—interested students signed up by providing their names and email addresses. Students who signed up gave me permission to contact them to participate in various phases of the study. However, participation was voluntary and students who signed up were not required to participate in the study at all in any given phase. This method of recruitment meant that different students participated in the different phases. For example, those who responded to the preliminary survey were not necessarily interviewed.

5.2.2 Research Methods

To begin the study, I sent a preliminary survey to the students, asking them to discuss their opinions on learning tools in general and on GitHub as a specific learning tool. This was distributed to all students who signed up to participate and would guide the questions asked in the next phase. Not all students who were emailed responded to the survey.

Next, I emailed all students who signed up and requested a meeting time for a face-to-face interview. This portion of the study began midway through the semester and continued until the end of the course. As the course was concluding, I attended one more lecture to recruit more participants in a manner similar to the methods highlighted in 5.2.1. Here, I interviewed the participants who were willing to discuss their experiences and opinions. 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 I took notes for reference. The interviews were semi-structured based on 12 guiding questions (listed in Appendix B) and I probed further with additional questions as deemed appropriate. This supported the exploratory nature of the work and allowed for the discovery of interesting insights.

I also interviewed the course instructors: the main course instructor was interviewed at the start of the semester and after both courses 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 (listed in Appendix B).

Finally, I conducted a survey to validate the findings and confirm or contradict the themes that emerged from analysis. The survey was distributed during the final lab session of each course, where students were asked to anonymously fill in a 5-10 minute online survey about their experiences. The questions posed in this survey are listed in Appendix C. Respondents did not neccesarily participate in either the preliminary survey or 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).

In summary, there were three main sources of data separated into three parts of the study: the preliminary survey distributed to students, the interviews with the students and the teaching team, and a validation survey distributed to the students. Students who participated in each phase did not necessarily participate in the other phases. The preliminary survey informed many of the questions to be asked in the interviews, and the findings that emerged from the interviews were confirmed or contradicted by the responses in the validation survey.

5.3 Preliminary Questionnaire

At the start of each course, a questionnaire was sent out to all students who indicated their willingness to participate during the recruitment stage. The purpose of this questionnaire was to determine the general level of experience and familiarity with GitHub, as well as to collect early impressions on GitHub as a learning tool. The questionnaire received 9 responses from the students in the SE course and 6 responses from the students in the CS course. The questions are listed in Appendix C.

Students who answered the questionnaire were generally familiar with GitHub, with only one student in both courses indicating that they were unfamiliar with the tool. The majority of respondents had not experienced courses where their instructors would use GitHub to manage course activities. When asked about how its use might benefit them, students in the CS course mainly discussed the benefits of using it for collaborating in group projects, while some raised the concern that it may not have much value for the course overall when used like an LMS. In the SE course, respondents discussed the real-world experiences using the tool would provide alongside the project collaboration advantages, as well as the public nature of projects where their work could be seen and reviewed regularly by other students and the instructors.

When asked about potential challenges, students in both courses had similar concerns, such as the implications of having publicly viewable work on cheating and plagiarism, and the lack of threaded discussions. Overall, many of the responses gathered from this questionnaire were reflective of the responses students gave during the interviews.

5.4 Interview Participants

I conducted interviews with 13 students from SE, 1 of which was in the CS course, alongside 6 others from CS. These interviews began at the midway point of the semester and concluded at the end of the semester, and students interviewed did not necessarily participate in the preliminary survey. 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 5.1 summarizes the students who participated in interviews.

5.4.1 Data Analysis

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

Table 5.1: Participants and their prior experience with GitHub.

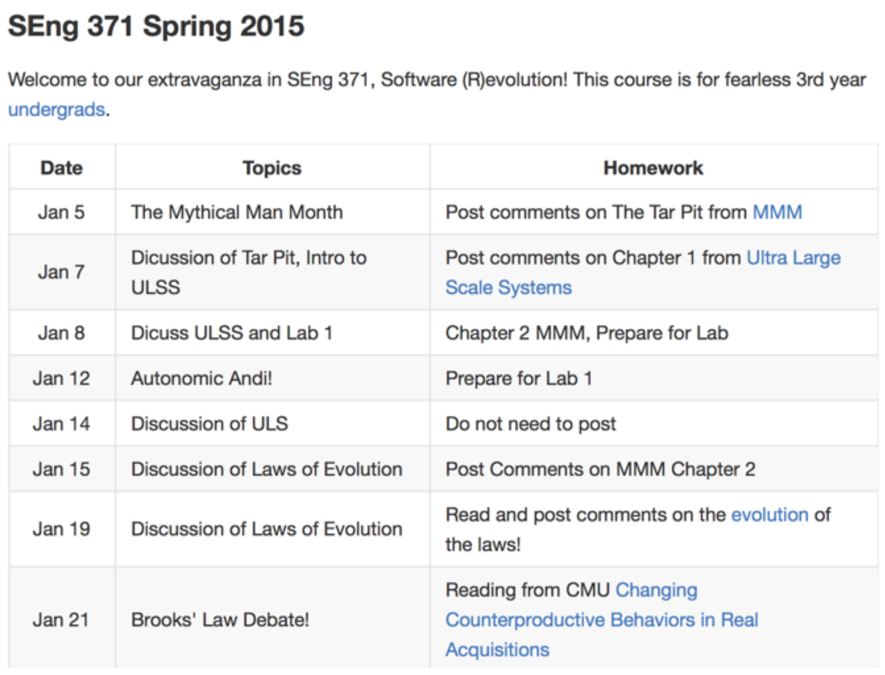
ID	Prior GitHub Experience	Degree Type
CS1	Inexperienced	Graduate
CS2	Used Academically, Professionally	Graduate
CS3	Used Academically, Professionally	Graduate
CS4	Inexperienced	Graduate
CS5	Used Academically	Graduate
CS6	Used Academically	Graduate
SE1	Used Academically, Professionally	Undergraduate
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

5.5 How GitHub was used

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 5.1. 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'

Figure 5.1: The front page of the SE Course Repository—the course schedule



SEng 371 Spring 2015

Welcome to our extravaganza in SEng 371, Software (R)evolution! This course is for fearless 3rd year undergrads.

Date	Topics	Homework
Jan 5	The Mythical Man Month	Post comments on The Tar Pit from MMM
Jan 7	Dicussion of Tar Pit, Intro to ULSS	Post comments on Chapter 1 from Ultra Large Scale Systems
Jan 8	Dicuss ULSS and Lab 1	Chapter 2 MMM, Prepare for Lab
Jan 12	Autonomic Andi!	Prepare for Lab 1
Jan 14	Discussion of ULS	Do not need to post
Jan 15	Discussion of Laws of Evolution	Post Comments on MMM Chapter 2
Jan 19	Discussion of Laws of Evolution	Read and post comments on the evolution of the laws!
Jan 21	Brooks' Law Debate!	Reading from CMU Changing Counterproductive Behaviors in Real Acquisitions

file.

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.

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[1], a version of the Moodle LMS[2]. CourseSpaces generally allows UVic instructors to make their course content 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

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

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.

5.6 Findings

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

5.6.1 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.

Students came into the course with varying degrees of experience with GitHub, as shown on table 5.1. Five interviewees had minimal or no experience using it, while others were very knowledgeable about the tool, either through their own uses, through group projects for other classes, or through co-op jobs. Many, at least those who attended the University of Victoria for the majority of their undergraduate studies, had some experience with Subversion, a different version control tool that is taught in a second-year course.

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*

Table 5.2: Summary of benefits of using GitHub for education from the student perspective.

Benefits	*Description*
Gaining Experience with an Industrially Relevant Tool	Students found it useful to practice using GitHub, a tool they believed they would often encounter in industry.
GitHub as a Portfolio	Course projects and assignments students stored on GitHub could later be used for seeking employment.
Supporting Student Contributions to Course Content	Students could make pull requests to change or add to the course material.
Support for Students to Contribute to Each Other's Work	Students could make comments, fixes, and pull requests on each other's projects and add the potential for peer reviewing each other's work.
Keeping Each Other Accountable	The commit history and other transparency features allowed students to be aware of each other's activity within groups.
Version Controlled Assignments	Students could revert to previous versions and believed instructors could use commit histories to provide continuous feedback.
Connecting with the Outside	Students could invite people from outside of the course to work with them on their projects. As well, they can easily find public projects to reuse for their work or for inspiration.

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]

For the most part, students who supported this theme believed that the use of GitHub for their courses and projects helped them experience a style of collaboration that they will encounter often in their careers. In comparison to the use of more traditional LMSes, one student noted why using GitHub might be advantageous for them: *"Well, I like how it's the bonus of more practice of something you're gonna use in industry, whereas none of us are gonna use CourseSpaces or Connex when we're out on a co-op or out on a job."* [SE3]

As well, 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. *"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]

Out of all the benefits described by students, the benefit of getting an introduction to GitHub and its features was talked about the most, as [SE2, SE3, SE4, SE5, SE6, SE7, SE8, SE11, SE13, CS4] specifically mentioned this benefit. The importance of this benefit is further emphasized by these students asserting their intention to continue using GitHub or to use GitHub even more after the course ends. This benefit was shared by students irrespective of their prior experience with GitHub.

Benefit: GitHub as a Portfolio

Many students believed that using GitHub to host their course projects will be beneficial to them in the future. These students described that hosting their code from other courses or from personal projects on their GitHub accounts benefited them in various ways. For example, SE5 organized their code on GitHub for easy access when helping friends: *"I know that when you're trying to help somebody out, you can always just say 'Check out my GitHub', I know I've done that with a few of my buddies . . . and I don't have to search through my files, it's just on GitHub, and you look on there. It's a good organization tool."*

Many interviewees shared that GitHub could serve as a type of portfolio where their publicly-hosted projects and code could be used to present to potential employers when job hunting. Many employers nowadays refer to GitHub for hiring purposes[3]. In fact, some of the students already experienced job interviewers asking them to show 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]

CS2 also shared that interviewers inspected their code during an interview, highlighting the importance of having functional code in one's GitHub account: *"These days I see that employers also want to see your GitHub page. While I was giving an interview for my coop, he did actually go into my GitHub profile and try to compile some of my code, so they do want you to have some online presence on GitHub."*

The benefit of using GitHub as a portfolio was shared by [SE5, SE6, SE7, SE8, SE11, SE13, CS3, CS4].

Benefit: Supporting Student Contributions to Course Content

In the previous chapter, 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. Traditionally, this could be done by speaking to the instructor, either in person or by email, and then the instructor could make the changes as necessary. With GitHub, however, the students are able to make pull requests (PRs), changing the material themselves, notifying the instructor and prompting them to 'accept' or 'close' (reject) their PR as deemed appropriate.

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*

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

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 . . . I feel that either we should have had the access to merge it, or at least someone else would have had [access] who would have merged it quite quickly, like someone handling the pull request . . . [Otherwise] that just defeats the purpose."* [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."* SE6 discussed the convenience this feature offers to instructors, as changes are listed on a separate page in the GitHub repository and can be accepted with one click. Of the three PRs submitted to the courses, two other students participated by either trying to accept the PR (and failing), or adding a '+1' to the PR's comments, supporting its acceptance.

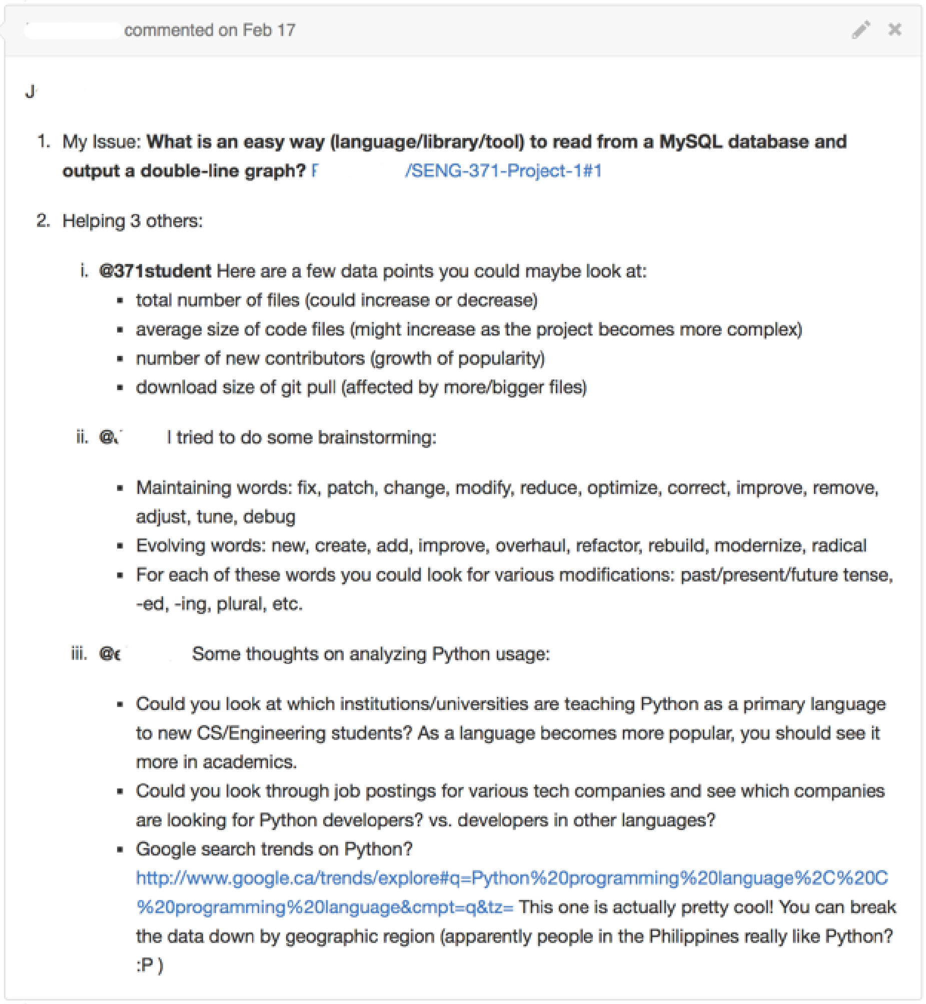
Discrepancy: This method of contributing to the course is limited because of the types of files GitHub supports. I elaborate on these limitations in section 5.6.4. Moreover, SE11 felt that the PR system might be problematic: *"I think it's kinda weird to be able to fix your professor's mistakes... [because] 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."* Nevertheless, the following students agreed that being able to contribute to course materials via GitHub is a benefit: [SE1, SE3, SE5, SE6, SE10, SE13, CS2, CS4].

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]

Figure 5.2: A student providing feedback for other students' projects in an 'issue'.


commented on Feb 17

J

1. My Issue: **What is an easy way (language/library/tool) to read from a MySQL database and output a double-line graph?** F /SENG-371-Project-1#1
2. Helping 3 others:
   i. **@371student** Here are a few data points you could maybe look at:
      - total number of files (could increase or decrease)
      - average size of code files (might increase as the project becomes more complex)
      - number of new contributors (growth of popularity)
      - download size of git pull (affected by more/bigger files)

   ii. @ I tried to do some brainstorming:
      - Maintaining words: fix, patch, change, modify, reduce, optimize, correct, improve, remove, adjust, tune, debug
      - Evolving words: new, create, add, improve, overhaul, refactor, rebuild, modernize, radical
      - For each of these words you could look for various modifications: past/present/future tense, -ed, -ing, plural, etc.

   iii. @ Some thoughts on analyzing Python usage:
      - Could you look at which institutions/universities are teaching Python as a primary language to new CS/Engineering students? As a language becomes more popular, you should see it more in academics.
      - Could you look through job postings for various tech companies and see which companies are looking for Python developers? vs. developers in other languages?
      - Google search trends on Python? http://www.google.ca/trends/explore#q=Python%20programming%20language%2C%20C%20programming%20language&cmpt=q&tz= This one is actually pretty cool! You can break the data down by geographic region (apparently people in the Philippines really like Python? :P)

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. For students who spoke in detail about this benefit, they enjoyed receiving comments from the others: *"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] An example of a student providing feedback can be seen in figure 5.2, where a student attempted to give three other students helpful advice on their projects.

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."*

Helping other projects, either through discussion or through collaborating on the code, offered students new ways to participate that are unique to GitHub and similar types of systems. Students effectively collaborated with each other with the aim of producing better work, as was noted by [SE2, SE3, SE5, SE7, SE10, SE11, SE12, SE13].

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]

Moreover, the students knew exactly how much work each member of their group contributed to the project and this helped the student keep themselves and each other accountable: *"We decided to switch to pull requests instead of just committing straight to master, because . . . for a couple of reasons, first of all, if there's something majorly* *wrong with it, everyone can see it, right? And the second thing is, everyone sees it, so if people have to work on [the same code], in the future, which we all did, then they know exactly what just went in, so that next time they come to the code and pull it, they're not like 'where did this all come from?'"* [SE9]

This is a useful feature to have when working in group work as it allows for awareness between group members. By using GitHub for their group projects, students were able to take advantage of the collaborative features that GitHub offers to improve their processes or their product. This benefit was described by [SE5, SE9, SE11].

Benefit: Version Controlled Assignments

Using version control for their assignments and projects benefited the students in multiple ways. CS1 worked alone on a project and shared that using GitHub *"makes it more traceable . . . you can see when and where [your work has been done]."* This student utilized the history of their commits as a reminder of where to continue from when they returned to work on their project. For others, the ability to revert to previous states of the code was useful: *"You're working on a project, and you make a change that breaks everything. Well you can just go back to a different commit, one that works. Boom, fixed, try again."* [SE11]

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."*

Students felt that hosting work on GitHub and using Git to manage their work was beneficial because of the ability to pull from anywhere, and to see and be able to revert to previous versions. As well, they believed that having instructors use version control to mark their work could be beneficial (if implemented) because of the new ways in which they could provide feedback. [SE1, SE2, SE8, SE11, CS3] shared this benefit.

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]

SE1 was the only student interviewed who used the public nature of the course projects to solicit outside contributions. However, the exposure to GitHub gave students opportunities to discover work outside of the course and to use other repositories to aid their projects. When prompted, most interviewees mentioned that they sought out public repositories either to pull their code and use it, or to find inspiration for their own projects. One student recalled an experience where their group looked at an open-source library: *"We just looked at how Gitstats, [an open-source library] did it, and then wrote our own thing into our project . . . I think that more than anything is the biggest reason why Git should be used for education, because it takes, I think, until you start being forced to do it . . . to actually go and look at other people's code, and I think looking at other people's code is the most important thing."* [SE6]

Speculatively, however, this likely would have happened regardless of whether or not GitHub was pushed by the instructor as students tended to seek out other code and libraries for their projects: *"And in industry, the first thing you do is check Stack Overflow, look for someone else who has done the same thing and jack their code."* [SE7] [SE1, SE2, SE3, SE4, SE6, SE7, SE10, SE12, SE13, CS2, CS5] mentioned looking at outside work and public repositories for their projects.

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

This section outlines the challenges the students described relating to GitHub use in courses. Some of these challenges were related to tool literacy, where more knowledge of the tool and more experience using it in an educational context could have mitigated the challenges. Yet they are worth mentioning as potential challenges that students might encounter.

Table 5.3: Summary of challenges with using GitHub for education from the student perspective.

Challenges	*Description*
Privacy is All-or-Nothing	When student work is publicly available, students saw potential issues such as incomplete or rushed work being judged by others. However, making their projects private might limit some of the benefits from the previous section.
Lack of Training on Git and GitHub	Students felt it would have benefitted them to have a tutorial session or even further integration of the tool in previous courses so that they may learn how to use GitHub and what benefits it provides.
Notification Overload	If students opted to receive notifications, they would receive too many notifications and emails, which they perceived as noise.

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, CS3 noted that although it can be advantageous to host code publicly so that employers are able to see their projects, the employer may not always agree: *"I think it comes back to what do you want to show your employers? When your employer looks at your work, will they understand that work I submitted in Git was when I didn't yet understand what I was doing, I was still learning? . . . If I could make the assumption that an employer would understand that, I would have no problem with it being public. That said, I can't make that assumption. I have to assume that everything they look at they're judging in the harshest light possible. So I try to show only things that are of quality that I'm proud of. And that's unfortunately not a lot of the classwork until I'm done with it . . . The final product I'm happy to show, but all those steps getting there, they're often filled with pitfalls and horrible programming and badly factored code."* As such, courses mandating the use of Git and GitHub to publicly host student work could be problematic for students if employers are looking at work-in-progress in a negative light.

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.

In summary, although students enjoyed the benefits that came with making their work publicly available, many students also acknowledged that these benefits are accompanied by a number of caveats. Importantly, some students described the lack of a middle ground as a limitation, where in the context of this course, students had to host their work publicly. This challenge may be mitigated by the instructor giving the students the option of creating a new GitHub user account for work done in their courses, as suggested by SE3 above. This challenge was shared by [SE1, SE4, SE5, SE6, SE7, SE10, SE13, CS3, CS6], while [SE3, SE5, CS2, CS5] disagreed on these issues.

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]

SE2 acknowledged the potential difficulties in hosting such a session: *"On the other hand, when someone teaches it to you, it often doesn't make sense until you actually do it yourself. Cause you'd actually have to go through the struggles of actually doing a commit and pressing all the buttons, so I don't really know how much could be done in that regard."*

Students also asserted that the University of Victoria needs to further emphasize teaching version control systems such as GitHub at the undergraduate level. As it stands, there is one required course that teaches version control systems and how to use them, utilizing Subversion and touching on Git. Some students, however, felt that one course was not enough, particularly when Subversion is not very popular anymore: *"I think in [SENG265], we did SVN, which is a good introduction to the idea. But I don't think it's widely used anymore."* [SE3]

Three of the interviewees [SE1, SE6, and CS3] believed that students should get an account quickly after their first introductory Computer Science courses. *"If I was teaching someone how to code, as soon as they start working on code that was bigger than 100 lines, I would teach them how to use version control."* [SE1]

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 the curriculum. As it stands, students were not able to properly utilize some of the benefits of using GitHub due to inexperience and unfamiliarity. This was discussed by [SE1, SE2, SE3, SE4, SE5, SE6, SE9, SE11, SE12, SE13, CS1, CS3, CS5].

Challenge: Notification Overload

Although few students brought up this issue, how GitHub handles notifications from the repository emerged as a challenge. The only way to get notifications from a course 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. The 'Watch' feature comes with some drawbacks, not the least of which was how a student's lack of familiarity with the feature prevented them from using it: *"I* *didn't like that [repository] at all, because I didn't get notified when [the instructor] adds stuff to there, so I don't really know what's going on without remembering to check it on GitHub. "* [SE9] This student did hear about the 'watch' solution, but thought that it *"would be a good solution, but it might be overkill. For like a spelling change."* [SE9]

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]

As such, the 'Watch' feature was problematic for courses like the ones studied, where every single comment would trigger a notification and an email, causing an overload of notifications. [SE7, SE9, SE10, SE11] shared this issue.

5.6.3 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.

Table 5.4: Summary of student recommendations for instructors who wish to use GitHub for their courses.

Recommendations	*Description*
Use GitHub in More Open-Ended Courses	Students recommended that GitHub should be used in courses were students are given more freedom for their projects and assignments rather than single-solution work.
Mandate the Use of GitHub's Unique Features	Students suggested that instructors use features such as pull requests and commit history extensively to take advantage of GitHub's benefits.
Define and Advertise a Workflow	Students believed the instructor should define a workflow for using GitHub and advertise the workflow to successfully use the tool in a course.

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]

Students acknowledged that the open-ended nature of these courses was what enabled the successful use of GitHub, but that it would not work in less open-ended courses because of plagiarism concerns. Regarding the potential use of GitHub in their future courses, SE5 discussed: *"Like I said, I like seeing other people's work and whatnot. Maybe not if everyone has the same assignment, because everyone's just gonna cheat off each other."*

These students related back to the privacy issue, where having completely public work might be a detriment to the work being done when there are concerns of plagiarism, such as when the assignments posted have a single solution. Some students, such as SE6, attempted to conceptualize a way to use private repositories, but ultimately felt it might be too cumbersome. As such, students believed that instructors would have to consider the nature of the work before deciding on the workflow they will use GitHub with, or indeed, whether they want to use GitHub at all. This consideration was suggested by [SE2, SE5, SE6, SE7, SE13].

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] It should be noted that this student's project group did not use GitHub to collaborate, but they used Docker Hub[4] instead.

SE7 echoed these sentiments: *"I think that you can accomplish the same thing with a simple HTML website, honestly . . . It's not using a lot of the features of Git, like looking at changes, commits, pull requests. The issues were kinda cool for the lab, and, again, you can accomplish that with any sort of forum, I would think . . . We're not actually delivering code to the professor, so maybe it doesn't make a ton of sense [to be using GitHub]."*

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. [SE3, SE5, SE6, SE7, SE11, CS2, CS3, CS4, CS6] touched on this suggestion.

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 decision. *"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]

[4]`https://hub.docker.com/account/signup/`

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]

One student mentioned that although GitHub does not do everything needed in a course, defining a workflow will cover up many of those weaknesses: *"Even if there are no enhancements on GitHub, but if you define a proper workflow for using it, then it can be quite successful, because even the present Learning Management Systems are not perfect right?"* [CS2]

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."*

In summary, many of the students interviewed were frustrated by the lack of a clearly defined workflow, and believed that the course would have been improved greatly if a workflow had been created and advertised in the beginning. This recommendation emerged from interviews with [SE1, SE2, SE3, SE4, SE5, SE7, SE9, SE11, SE12, SE13, CS2, CS3, CS4, CS5].

5.6.4 RQ4: From the student perspective, how does GitHub compare to traditional Learning Management Systems?

One of the goals of this research study was to discover the effectiveness of GitHub as a tool when used in ways similar to Learning Management Systems. Several instructors that we spoke to in Chapter 4 described using GitHub for purposes that LMSes typically serve, such as material dissemination, evaluating students, and discussion among the students and instructors. As such, the use of GitHub as an LMS is a use case worthy of investigation.

The primary comparisons made during the interviews were between GitHub and CourseSpaces, which was described in section 5.5. However, students also described their experiences with Connex[5], an LMS developed using Sakai[6]. Connex provides instructors with basic course management tools such as material dissemination and assignment submission, as well as options for plugins to enable the use of other tools. For example, instructors can enable the use of Drop Box[7] to share files, the creation of polls or podcasts, and the use of a chat room to support class communication. Below, I discuss the comparisons students made between GitHub and LMSes they've used in UVic.

Table 5.5: Summary of comparisons between GitHub as a learning tool and LMSes students have used at UVic.

Comparisons	*Description*
GitHub Lacks Threaded Discussions	In comparison to the LMSes students are accustomed to in UVic, GitHub lacks the ability to reply to comments in discussions. This means discussions in GitHub are linear and chronologically ordered, which some felt was difficult to follow.
GitHub is Not Built for Education	GitHub does not allow for educationally-focused activities such as formal assignment submission and gradebooks. Its lack of support for file types often used in education such as PDF documents and Powerpoint Presentations could make it difficult for students to alter class material.
GitHub has a more Intuitive User Interface	Some students felt that in comparison to the LMSes used in UVic, GitHub was much easier to navigate.

Comparison: GitHub Lacks Threaded Discussions

When GitHub was compared to traditional LMSes students have used at the University of Victoria (Connex/Sakai, CourseSpaces/Moodle), many of our interviewees

[5]`https://connex.csc.uvic.ca/`
[6]`https://www.sakaiproject.org/`
[7]`https://www.dropbox.com/`

mentioned discussion and forum features. For both courses in this study, there were two main platforms where students could discuss and comment: on the course repository's GitHub 'issues' page, and on CourseSpaces' forums. The course instructor separated these by assigning all lab work and discussion to GitHub and the assignment readings and comments to CourseSpaces.

Students were generally receptive to the use of issues as posts, particularly as it offered flexibility not seen on regular forums. For example, students liked the ability to 'mention' others and to receive notifications when they are 'mentioned', as they felt these notifications were convenient and efficient. *"I really like how once somebody's commented on the thread, you can just use the little @ symbol and send them a notification and vice versa. When someone mentions you in a comment, you'll see it on your email. Which is good."* [SE2] The 'mention' feature is furthermore generally unavailable in the discussion features of Learning Management Systems used in the University of Victoria.

However, not all the students appreciated using GitHub Issues for discussions. The main difference between the 'issues' page on GitHub and the forums in traditional LMSes is the lack of 'tree-style' discussion in GitHub where all comments on an issue are arranged by time in a linear fashion. In the CourseSpaces forums, in contrast, there are top-level comments in a thread which users can reply to individually, and these are arranged in a manner which makes conversations easy to follow. *"GitHub unfortunately doesn't have any sort of like tree style view of conversation, it's just linear, so it's really hard to actually have a conversation between multiple people ... As opposed to CourseSpaces, which is kind of more tree-styled."* [SE1]

The main issue, according to SE1, is following these conversations: *"And you literally have to go and search for all of the conversations from foo and bar and try to piece them together. It would be nice if it had like a reply that would put it right underneath."* [SE1]

Discrepancy: One student did, however, believe the linear style of GitHub comments was useful in comparison to the tree-style discussion on CourseSpaces. When asked if they read through the posts on CourseSpaces, SE7 responded that they did not as much as they read the GitHub comments, because *"you'd have to scroll through all the responses before getting down to the bottom to write your own."* Scrolling down through the other comments, they said, encouraged them to read, or at the very least skim the comments they passed.

Ultimately, however, most students that compared between the two types of discussion wanted at least the option to have tree-styled comments on GitHub. This was echoed by [SE1, SE3, CS2, CS3, CS4].

Comparison: GitHub is Not Built for Education

A few students described one drawback from using GitHub for education—GitHub is simply not built for it. Those that mentioned this particular drawback acknowledged that although there may be workarounds for many of the tasks needed, GitHub certainly struggles to meet some basic educational needs such as gradebooks and a formal assignment submission feature. In traditional LMSes, for example, assignment submission tends to be a private upload of relevant files visible only to the instructor. This simplifies the process and ensures that there is only one submission.

The courses in our study did not utilize GitHub for any form of submission of the students' code for projects. However, when asked about GitHub serving as an LMS, some students highlighted the lack of formal submission features in GitHub as a weakness. For example, SE6 stated that *"If there was gonna be assignment submissions, you'd have to figure out a way to do that properly"*, and suggested potential workarounds by using pull requests. When listing features that would hinder GitHub from being a learning tool, others [CS3, CS4, SE7] also highlighted the lack of formal submission as a potential hindrance.

The benefits of using GitHub were also minimized because it is not a tool designed for educational purposes. As discussed in the last chapter, GitHub does not handle file types common to class material well, such as PDF documents and PowerPoint presentations. This creates difficulty for students who want to make changes to course materials that involve these file formats. *"I think one disadvantage of.. or 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]

This may hinder the potential benefit of using pull requests to contribute to the material as it complicates the process. SE13 was dissuaded from the idea of using PRs as a result of this complication: *"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."*

Students also discussed the inability to do tasks commonly performed in most LMSes. For example, for the University of Victoria and many other universities, including a gradebook is impossible to do without a tool in which the data is controlled by the university. *"Because at least with CourseSpaces, you can access your grades and access your schedule that's coming up, and you can access your entire transcript. But that needs to stay in the school domain, not in the public GitHub."* [SE5] As such, there's difficulty managing more administrative-type tasks because GitHub data is stored on GitHub's servers rather than those belonging to the university.

SE9 acknowledges the difficulties in using GitHub in a context it wasn't designed for: *"Right now it definitely feels like we're using a tool that's geared for something else and trying to throw education on top of it . . . So if you could integrate them better into another tool or maybe a plug-in for GitHub if they ever did something like that."* Unfortunately, the two potential solutions they described are not realistic options. When asked exactly what GitHub would need to provide to make it more suitable for education, they noted some features typically found in LMSes: *"Like deliverables, like grades, announcements that you'd actually get by email, not just code, commit changes."* [SE9]

Because GitHub is a tool designed for developers, it cannot be used for some educational activities because the features do not exist. While there are workarounds to some of these missing features, they often require extra effort on the part of the instructors and the students. This complaint about GitHub surfaced from comments from [SE2, SE5, SE6, SE7, SE9, SE11, SE12, SE13, CS4].

Comparison: GitHub has a more Intuitive User Interface

Many of the students described a preference towards the GitHub interface over the CourseSpaces interface. Some students, such as [SE8, SE12], complained that the User Interface of the Learning Management Systems used at UVic often did not meet their expectations. *"I think Connex and CourseSpaces were made specifically for course management. So I think they're more capable [for that]; just functionality wise, the capability's there, but maybe the UI-wise, is not as friendly."* [SE12]

Consequently, [CS1, CS4, CS5, SE13] described GitHub's User Interface as cleaner or friendlier, particularly in the way the course schedule was presented. [SE1, SE13] said they were more comfortable with the GitHub interface because of their familiarity with the tool, as opposed to a tool like CourseSpaces: *"Also, because it's how big GitHub is and how familiar I am with it, I can navigate it a lot easier than say, CourseSpaces."* [SE13]

5.7 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.

5.8 Instructors' Perspectives

We interviewed all three instructors (one course instructor, two lab instructors) after the course concluded so they were able to give their perspectives on the themes extracted from the student interviews. The interview questions asked are listed in Appendix B. Overall, the instructors were satisfied with the use of the tool and optimistic about its potential for future use. There was, however, a key difference between both cases. In the computer science course, lab assignments were done on their own and then shown to the lab instructor. This contrasts to how lab assignments were handled in the SE course, where students posted comments to the corresponding lab 'issue' with their submissions and who they worked with. As such, one course used the tool much more than the other, and this is explained by the lab instructors below.

5.8.1 Insights from the Course Instructor

The course instructor felt that their relative inexperience with the tool limited its effectiveness, an issue a few students such as SE1 and CS3 identified. The course instructor did, however, speak of the importance of working with a tool that is relevant in industry, as they believe it drove the successful use of GitHub for course projects. *"For students, my sense is, for those who are ready to get in the game, it really amplified their commitment and really was in keeping with the real world . . . So I believe that the benefit was that kind of participation in something that's bigger than just what we can normally give them."*

Indeed, the course instructor believed that this ability to work in an ecosystem that students would need to work in for their future careers helped drive the enthusiasm towards the course. In fact, the instructor mentioned that even if the repository was not publicly visible, the tool's relevancy to student careers would have offered enough of a reason for students to participate. *"I actually think it could've been private too and still there would've been some more excitement with just sharing with each other. But I also feel like being a part of something that's going on around them, you know, contributing to a space that they know is a part of their ecosystem, I think that's it. So even if it was a private GitHub, I think it would've upped the game."*

The course instructor also saw advantages with the transparency features GitHub offered when students collaborate using GitHub for their projects. For example, having publicly available work would allow the teaching team to track student contributions to projects and raise concerns based on that activity. *"Certainly, when it came to looking at the projects and people that participated in projects, it was very interesting. And there was a difference between what we saw in terms of the code repo and how students were participating in their own repos and some of the things that they were handing in. . . . I think for us, these subtleties are important."*

As described earlier, the teaching team experienced an issue when there was a gap between the course instructor's criteria for marking and how one lab instructor decided to mark. The course instructor believed that this problem might have been avoided had the teaching team agreed upon and defined a workflow for using GitHub at the beginning of the semester. However, the course instructor felt it important to gather ideas from students regarding the definition of this workflow: *"Because although I didn't get direct pushback, they were saying, 'well it would be really nice if the teaching team knew how to use it', and I said right away, 'yup I'm gonna* *be learning'. But at the same time, I think the challenge is to tease out from [the students] what they see as being the right way to use that."*

Regarding the criticism that an open system like GitHub may not work for courses where there are assignments with a single solution, the course instructor disagreed. However, the instructor mentioned how having publicly available assignments displeased some students in a different course they previously taught: *"I did something before with another platform where it was public and students were doing labs, and of course... the first lab that ran through certainly provided things that other students used. Which I didn't think was a bad thing, students didn't complain to me directly; indirectly, I heard that students in the first lab thought that it wasn't fair ... I think it was interesting because it created a hierarchy within the class."*

The course instructor explained that more thought would be required for using a system like GitHub for a course with single solution assignments, but the benefits would still exist: *"I just feel like the more we can help them help each other, the better off we'd be. So I'd have to think about how to do [a course like that] more, but I would not be against it."*

Finally, the course instructor discussed how GitHub compares to more traditional LMSes, explaining how important it is to introduce a system like GitHub to the education of computer scientists and software engineers. They discussed the concept of the 'Ivory Tower' as something that traditional LMSes fall victim to, where a course exists in a vacuum: *"I just feel like [GitHub has] got that real world edge, and that is everything to destroy this kind of ivory tower, 'I'm just doing what I have to do to get by', horrible thing that sometimes happens to our students, [where] they feel like there's nothing they could do within this system that would really make a difference anyway. Because from an instructor's perspective, I need to get information out to them, and I could do that in an email, in a listserve, I don't know. I need to get them to engage with the material and with each other. And I need to get them to engage with the community in a broader sense. And I think that's critical, and that's what GitHub has."*

5.8.2 Insights from Lab Instructors

The lab instructors generally agreed with the feedback given by the students and the course instructor. However, speaking to them highlighted the need to encourage the specific use of certain features. The difference between the two course labs was that one group had much higher levels of participation than the other, and this disparity was perhaps a result of one lab having a defined workflow. In the course with higher levels of participation, the lab instructor often relied on using GitHub and its issues feature, while the other lab instructor did not. *"For me, [the use of issues] was not applicable, cause ... everything was done in the lab, right? So demos were all done in the lab... But if there were more hours allocated for it, I guess students would have posted more."* [TA CS]

The SE lab instructor utilized the 'issues' page to benefit his teaching: *"Basically I told them to post their responses to [the lab] as a response to that issue. For me it was nice because one, I could see they were doing stuff as they posted things, so occasionally I would get up and I would like to see what they were doing, so I would tell them to post a response even in the middle of the lab just so I could see what they were working on. Just to see in real time, and then I could analyze it, and if I had feedback, go around to them and give them feedback on what they've done so far."* [TA SE]

It seemed the lab instructor for the SE course wanted to take advantage of the tool's available features and defined a specific workflow to take advantage of them. This may have resulted in the course instructor perceiving more enthusiasm from that group of students within the course. According to both of the lab instructors, the course instructor gave no specific instructions regarding how to use GitHub throughout the course, which meant that the lab instructors were left to define their own workflow for using GitHub in their labs. The CS lab instructor stressed this need in future iterations of using GitHub in courses: *"You just come up with a system that works and just stick with it for the rest of the class."* [TA CS]

As well, both lab instructors seemed to agree on the potential of GitHub to benefit the education of students. As well, they believed in the advantages GitHub might offer instructors for marking. They appreciated the openness of GitHub for being able to see the projects as they develop as well as for marking those projects. *"Towards the end of the [each project] run, I would start looking at the projects through GitHub, and then try to provide some feedback to the students before they were due, to help them figure out what they could improve on ... the labs were not interactive for me, so it was a good way to provide feedback because nobody was really asking me questions."* [TA SE]

They also acknowledged the potential benefits of the pull request system for course material. *"It's a collaborative environment, so I don't see why students would not be* *allowed to participate in it. Especially in a university environment, people like to restrict things and like put boundaries. But that usually ends up restricting people's creativity. So the more open, for my perspective, the better . . . I think that if people are able to modify material that will just enhance the class experience for everyone."* [TA CS]

Regarding the issue of publicly viewable assignments, the SE lab instructor described his experiences in a course where there was one-solution assignments. Although they acknowledged that it would be more difficult to use a tool like GitHub in less open-ended courses, they felt it could have helped their teaching experience. *"This is nice because I can see the code quite easily. I guess for this class, I wasn't judging them based on their code, I wasn't really looking at the code much. But for [a different course], something like this would be really nice, because I would have access to their code, it'd be a little bit easier . . . I never had to look at their code anyway other than going over . . . and checking when they had questions. But in that case, I may have looked at it more had I had GitHub rather than trying to figure out how to download the submissions through Connex."* [TA SE]

Overall, both felt that with some changes to workflow and perhaps to the system itself, GitHub has the potential to serve as a powerful learning platform, particularly for classes that involve a heavy focus on collaboration.

5.9 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 [46], which will be further discussed in the next chapter. 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 [52] 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 [24] 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 [34].

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’ [12], 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 [31], 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 [29]. 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 [64].

However, it is important to note that like any technology, accessing these benefits requires the stakeholders to ‘buy in’ and use the relevant features of the tool to support this pedagogy. It is possible, for example, that there were different levels of enthusiasm for the tool between the two courses because of the differences in how it was used in the lab sessions. The SE case required students to post often, which possibly encouraged them to look at others’ responses, while the CS did not utilize the tool as much, requiring only a demo of the weekly assignments to the lab instructor instead.

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 [15].

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 the instructor interviews in the previous chapter, 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 [65], 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 [52] can be overcome. Student projects, for example, could involve people from another community, or outsiders can contribute to the course materials in some manner.

Tool Literacy

An important note from some of the limitations that the students and the instructors described is the importance of understanding and being proficient with the tool. As an example, in discussing what considerations need to be made to design an effective workflow, students would discuss the difficulty of conducting courses with single-solution assignments rather than open-ended projects. This was due to the way in which GitHub repositories are required to be private or public, making it difficult to handle assignment submission.

However, some experience with the tool or some investigation of GitHub's recommended practices for using their tool in education would have revealed the possibility of using private repositories for each assignment. An instructor could introduce new assignments or make clarifications in a student's private repository if they were simply added as a collaborator. As such, it is important to consider that some of the limitations described by the students and by the instructor may be from unfamiliarity with using the tool, especially in a context it originally was not meant to serve.

5.10 Limitations

In this section, the limitations and the threats to validity of this study are outlined. Multiple limitations surfaced due to the study having only one researcher, which introduced many potential biases.

5.10.1 Internal Threats to Validity

Internal validity is concerned with biases within a study [14]. This is threatened when a researcher's actions and biases affect the work done in each process, such as during data collection or during data analysis.

In this study, I was the sole researcher, meaning my biases may have played a role in both the data collection and the data analysis. The semi-structured nature of the interviews meant that I would often go off-script to probe further, potentially resulting in leading questions. Moreover, the initial questions in the surveys and the interviews may have been biased as well.

Moreover, the recruitment methods listed earlier in the chapter may have biased the population: by searching for instructors teaching appropriate courses, I first approached instructors I 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. This resulted in a less than optimal use of GitHub (as described by many of the students interviewed) because the instructor had little prior experience with GitHub as a tool. As well, because of this inexperience, I would give the instructor advice or resources on possibilities of how they can use GitHub to meet a goal—I didn't, however, directly give them step-by-step directions to avoid influencing the direction of the class as much as possible.

In the data analysis, there was no inter-rater reliability because I was the sole coder. As such, biases may have been introduced in my selection of the themes. Having multiple raters analyze the data would have introduced more perspectives and interpretations, which would have reduced the potential biases in the analysis. Unfortunately, this study suffers from a single-rater limitation.

Finally, the opportunistic nature of recruitment may have resulted in possible biases in multiple ways. With only one instructor teaching two courses, this study is limited from having no other cases to compare with. 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. Therefore, it cannot be assumed that the views of the interviewees represent the rest of the class.

5.10.2 External Threats to Validity

External validity is concerned with the extent to which the findings from this work can be generalized and to what extent the findings are of interest to people outside the case [60]. As a case study, it cannot be assumed that these cases can be generalized to the use of GitHub in education. However, many of the findings are reflected in other studies that use similar tools for classes, such as Kelleher's study on Git and GitHub [36], and Haaranen and Lehtinen's study on Git and GitLab [28].

5.10.3 Standards of Rigor

I used a number of approaches[60] to establish rigor and to minimize the threats to validity described above. One such approach included triangulating data from multiple sources—the students as well as the teaching team. In doing so, I was able to identify and highlight contradictions between different sources and report discrepant information.

This study was conducted over the span of two courses which provided prolonged involvement with the population and allowed for a good understanding of the participants' perspectives. Moreover, I deployed a survey to students in both courses in order to validate the themes extracted from the interviews. This is a form of member checking and allows students to verify the analysis of the data. Finally, this study also involved a peer with whom I discussed the study with regularly. This peer, who is experienced with qualitative research methods, was consulted throughout the study, including the study design and the collection and analysis of data, and would lower the risk of any biases affecting the study.

In summary, this study has shown the effectiveness of using GitHub for educational purposes from the student perspective. This study describes the benefits of using GitHub for education, such as the possibilities for student contributions. However, these benefits are accompanied by limitations, such as the implications of having publicly available work on cheating and academic integrity. In the next chapter, I offer recommendations for instructors who want to attempt using GitHub in their courses in order to maximize the benefits of using the tool.

Chapter 6

Discussion

The studies in the two previous chapters explored the use of GitHub as an educational tool, first from the instructor's perspective and second from the student's. In this chapter, we discuss the viability of using GitHub and an open, collaborative workflow in courses, a number of recommendations for instructors who wish to use GitHub in their courses, and the implications of this research.

6.1 The Viability of GitHub for Education

In this thesis, we have discussed how GitHub can be used to support teaching and learning, as well as why instructors might consider using GitHub for their courses. Even though there are multiple Learning Management Systems available for higher education institutions to consider, it is compelling to ask: can GitHub on its own be one such tool, or does its use in education require another, administratively-focused tool to be used in conjunction? Coming back to the first research question posed in chapter 1 (RQ1), is GitHub and its open, collaborative workflow a viable approach for education?

Investigating Malikowski *et al.*'s [46] model, LMS activities are split into five categories:

- *Transmitting Course Content*, which instructors most often use LMSes for. GitHub not only supports activities related to this category, but it can also extend it by providing a two-way transmission wherein students can also easily contribute to the course content.

- *Creating Class Interactions*, which GitHub supports through the commenting system, either in-code or in the 'issues' pane. These interactions are mostly asynchronous, however, as GitHub provides no support for interactions similar to a chat environment, though external chat tools such as Slack[1] and Gitter[2] can integrate GitHub to display the activities in a repository. Student interactions on GitHub can extend to reviewing and contributing to each other's work.

- *Evaluating Students*, which GitHub enables by allowing instructors to comment on student work, which notifies students upon receiving comments. One important limitation in this category, however, is that in the typical case, data stored in GitHub's repositories will be stored on GitHub servers in the United States, rather than in something an institution can have control over.
- *Evaluating Courses and Instructors* is an activity GitHub does not natively support. An instructor would need to use external tools such as survey generators.
- *Creating Computer-Based Instruction* is an activity that instructors could use GitHub for with some work. A basic form of computer-based instruction is online quiz generation, a feature built into many modern LMSes, but a feature GitHub does not support without building an external tool. However, automatic grading tools can be built and utilized whenever a student makes a push, creating a form of computer-based instruction.

As such, GitHub enables many of the activities typically done on LMSes. However, with data typically being stored on GitHub's servers, many institutions require much of the administrative artifacts such as class rosters and grading to be under their control, which means that another tool will need to be used in conjunction with GitHub. Fortunately, there are solutions, as outlined below.

Some of the instructors we interviewed were able to set up servers to be used for GitHub in their own space, thereby having control of the data. This can be done through GitHub Enterprise[3]. Another solution would be to set up a tool that is similar to GitHub that can use servers that are independently set up, such as GitLab[4]. These solutions would eliminate the limitation of having external servers, therefore enabling the hosting of secure artifacts. The disadvantage with these approaches,

[1]`https://slack.com`
[2]`https://gitter.im`
[3]`https://enterprise.github.com/home`
[4]`https://about.gitlab.com`

however, is that they minimize the benefits of having a course visible and open for outside communities to participate in.

Overall, using GitHub for educational purposes can be an effective alternative for instructors who wish to avoid the limitations of traditional LMSes[24]. GitHub also offers advantages and opportunities not always present in LMSes. Therefore, it should be a tool that instructors, particularly those teaching computer science and software engineering, should consider using to support their courses.

Regarding the second research question (RQ2) posed in chapter 1—what are the benefits and weaknesses to GitHub's open, collaborative approach from an instructor's perspective?—GitHub offers instructors unique opportunities for grading and for easy ways for reusing and remixing course material from previous and for future iterations of a course. Table 6.1 summarizes our findings regarding the benefits and potential challenges when instructors use GitHub.

Addressing the third research question (RQ3)—what are the benefits and weaknesses to GitHub's open, collaborative approach from student's perspective?—GitHub offers unique ways of engaging students by encouraging them to contribute to courses and to each other's learning. These opportunities to engage students build a culture where students can perform better as a result [40]. Table 6.2 summarizes the findings related to the use of GitHub in education for students, gathered from both phases of this work.

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, I 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

[5]`https://education.github.com/guide`

Table 6.1: RQ2: Summary of potential benefits and challenges in using GitHub for education from an instructor's perspective.

Findings	*Description*
Benefit: Transparency of Activity	Instructors can see the full history of student work and the can grade work and provide feedback incrementally (Phase 1, 2).
Benefit: Reuse and Sharing of Materials	Instructors can reuse course materials and share with other instructors via the forking mechanism (Phase 1).
Benefit: Student Contributions to Material	Instructors can easily approve changes to materials made by pull requests (Phase 1, 2).
Benefit: Free Academic Licenses	Instructors can create free public repositories and can apply for an educational account that provides them free private repositories (Phase 1).
Challenge: Shared Knowledge Base	Instructors are lacking a shared knowledge base to learn best practices for using GitHub in education (Phase 1).
Challenge: Barriers to Entry	Learning Git can be difficult for novices (Phase 1).
Challenge: Support for Other Formats	GitHub does not support file types like PDFs or PowerPoints, potentially making the pull request system cumbersome for students to use for changes to course files (Phase 1, 2).
Challenge: External Restrictions	Hosting data such as grades on GitHub is not allowed in some universities (Phase 1, 2).

Table 6.2: RQ3: Summary of potential benefits and challenges with using GitHub for education from a student's perspective.

Findings	*Description*
Benefit: Student Contributions to Each Others Work	Students can suggest or make changes and fixes to other students projects (Phase 2).
Benefit: Industry Relevance	Students are exposed to a tool popular in industry, and they can use their profiles as their portfolio (Phase 1, 2)
Benefit: Accountability	For group projects, students know exactly what group members contributed and has a record of it (Phase 2).
Benefit: Version-Controlled Assignments	Students can revert to previous versions of their work (Phase 2).
Benefit: Bringing in Outsiders	People from outside the course can contribute to student work by making suggestions or changes (Phase 2).
Challenge: Privacy is All-or-Nothing	Concerns of plagiarism for public work, while making student work private eliminates some of the benefits (Phase 1, 2).
Challenge: Lack of Training for Git and GitHub	Not all students are experienced with GitHub, which makes training beforehand essential (Phase 2).
Challenge: Notification Overload	Students watching the repository are sent email notifications for every action, leading to noise (Phase 2).

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.

Recommendation: Use 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 [29]. As another example, exposure to GitHub's Issues feature, even for basic discussions, was helpful for one of the students interviewed during the second phase as the student learned how the feature works for use in future projects.

Educators can furthermore use GitHub's transparency features to provide feedback to students in unique ways. For example, instructors can trace the history of student projects and assignments hosted on GitHub, and instructors can detail where students made mistakes and can intervene 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.

One important lesson noted from the case study was to communicate the workflow the instructor decides clearly and properly to the teaching team and to the students. When deciding to use a feature like pull requests on course material, for example, the instructor must advertise this workflow properly, perhaps even offering bonus points for added material. To communicate a workflow to students and introduce GitHub and its features to novices, instructors should consider creating a guide or hosting a tutorial session.

Recommendation: Use Free Private Repositories for Single Solution Assignments

[6]`https://github.com/education/teachers/issues`

Many students and instructors believed that GitHub worked best when a course has open-ended projects and assignments. This belief is because of the 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.

This style of repository management (where a private repository is dedicated to each student) could work for assignment submission as well. The instructor could ask the students to create a branch, or ask the students to fork off the main repositories and make the forks private, and then mandate that the student must make a pull request before a deadline. Thanks to GitHub's transparency features, an instructor can continuously observe the work in each student's repository and can provide further assistance to students based on the work history.

However, the set up for this more private style of repository management requires some time and assistance from GitHub. There are two options: first, students can apply for student accounts which grants them 10 free private repositories, one of which can be used for the course in question. That is, however, a time consuming process as students must wait for GitHub to approve their request. The second option is to create an organization for the course, which is granted an amount of private repositories depending on how much the instructor pays. While GitHub has stated that they would give teachers a free organization for their courses[7], an organization must be set up well before the course begins in order to get the private repositories in time.

Moreover, if assignments are in private repositories and are single solution assignments, you 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.

[7] `https://github.com/blog/1775-github-goes-to-school`

That said, GitHub continues to offer its benefits when used to submit single solution assignments. It involves some preparation to get free private repositories for students, but at the same time, it allows instructors to provide better feedback through versioning, and it maintains the benefits for students of learning Git and GitHub and hosting their work for future portfolio use (if allowed to publicize their work after the course concludes).

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 [22].

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.

6.3 Implications for the Future

Another important consideration from this work relates to the future of tools for computer science and software engineering education—what's next? First, we con-

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

sider 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 [33]. The two phases of this work demonstrate 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 the GitHub Way—an open, collaborative workflow—to current and future learning tools.

The fact that GitHub easily supports participatory activities has multiple implications. Literature has shown that LMSes have been adding 'Web 2.0' features such as blogs and wikis to their feature set [19]—students are being offered more opportunities to participate by discussion or by contributing content in blogs or wikis. Where the GitHub Way excels in education, however, is in the opportunities for students to contribute to and change the materials, and to contribute to each other's learning by getting involved in and providing feedback to projects other than their own. This is potentially the next step for Learning Management Systems, where students are more easily able to make these contributions to the work of others. The concern, however, is that implementing features similar to GitHub in an LMS might seem forced and haphazardly planned, and tool builders would be better served building a tool that supports and encourages an open, collaborative workflow from the outset.

As such, another 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.

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, I 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. In the next chapter, I

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

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

provide concluding remarks as well as some possible directions for researchers to take this work in the future.

Chapter 7

Future Work and Conclusions

This thesis explored learning tools in computer science and software engineering education, proposing GitHub as a solution to the problems that traditional learning systems suffer from, such as a lack of focus on student contributions and a 'walled garden' approach to education where content is limited to the scope of a course. As contributing to code and to the community has become an important part of being a software developer, I proposed through this work that the learning tools in this field need to account for the need to develop related skills such as teamwork, critical analysis, and communication, skills that could be gained through collaboration with others via contributions on projects hosted on GitHub.

7.1 Future Work

From these studies, a number of questions and possibilities for future work emerged. This section outlines these possibilities.

How would the use of tools similar to GitHub affect student performance?
Because of the exploratory nature of the work, I sought to obtain teacher and student perspectives regarding just the viability of GitHub as a tool for education. However, other studies have investigated using tools such as wikis [51] and how they possibly affect or correlate with student performance. This is one natural extension of this work: running a field experiment to see whether or not using the tool simply engages the students more or if it can ultimately affect grades.

Building a tool that is more focused on education
Discussed in the last chapter, this is another area that can be explored in the future. One could use an open-source tool like GitLab, which shares similar features to GitHub, and transform it into a tool more suitable for education by adding some privacy options and administrative features. From there, one would test the tool in a similar study to identify its strengths and weaknesses, and whether a tool that is more focused on educational activities would be well-received.

How scalable is the use of this tool in a course? What considerations have to be made for larger class sizes?
Another question that remains unanswered from this work is how this use of GitHub for courses might scale. With class sizes of over 100 students, for example, would using GitHub work for earlier (e.g., first year) computer science courses? Haaranen and Lehtinen [28] used GitLab in a study with 200 students and found it effective. However, it would be interesting to see which findings and benefits remain, and what new challenges emerge from larger class sizes.

7.2 Concluding Remarks

In interviewing early adopters who used GitHub for educational purposes, we uncovered how GitHub could be used in this context and what benefits and challenges accompany those uses. For instructors, they are given novel ways to allow their students to participate in class, and they are provided a more transparent way to grade projects and assignments. However, with no shared knowledge base of suggested practices (when the study was conducted), it's difficult to determine exactly how GitHub can be most effectively used in a course.

I then conducted a case study to explore the student perspective and uncover what benefits they might see from using the tool or what challenges they may experience. For students, the ability to provide and receive feedback from each other was an important benefit, alongside the experience and training they received in using a tool relevant to their careers outside of the university. Privacy concerns, however, were an important issue for some students, where some were not comfortable having their work be publicly available.

Overall, this work suggests that using GitHub is viable in an educational context, despite the fact that GitHub is not designed for education. From the findings of this work, I believe that the GitHub Way of working is worthy of consideration in the development of future learning tools, particularly for tools designed for computer science and software engineering education. The ability to collaborate with others and contribute to the course and to other students' work provides valuable experience for students in those fields.

Appendix A

Interview Questions - Instructor Perspective

This appendix lists the interview questions we used in the study highlighted in Chapter 4. The first section lists the starting set of questions we asked the educators. The interviewer would probe and ask further questions when interesting topics emerged. The second section lists the interview questions used when we spoke to the representative for GitHub.

A.1 Instructor Interviews

- Do you have any experience with using Github in aspects other than education? If so, how long have you been using Github?
- Did you use Github in your class?
  - If yes ,What course(s) did you use Github for? How many semesters you used Github for teaching?
  - If no, why not?
- What was your motivation to use GitHub in your class?
- What was the class size (i.e. number of students)? Was it a frontal (face-to-face) or virtual (online) course? Did the students have any technical background in your class?
- Did you provide GitHub training to the students?

- What was GitHub used for in your class (such as announcement/assignments/platform for resource)? What features of Github did you use in your class (for example, push, merge, pull request)? How were these features used to achieve your purpose? Please elaborate and give examples.
- What was missing for you in GitHub that would have helped you in your teaching?
- What did you (or your class / students) benefit by using Github? What were the strengths of GitHub that made it useful to you?
- What were the difficulties/challenges for using GitHub in your teaching?
- Do you think Github saved you time, comparing to the traditional ways you used before? If so, in what ways?
- What was students' attitude towards the usage of GitHub in the course? Did any student complain about it?
- Are you satisfied with the user interface? (1-10 or doesn't matter) If not, please tell us which part and why.
- Did the usage of Github encourage students participation in your class? Was student performance improved when using Github comparing to when they don't use it?
- Are you satisfied with the experience with Github overall? (1-10) How did you like the experience of using GitHub in your teaching?
- Any other comments to help Github to improve its application to education?
- Do you have any further plans to use GitHub in the future for your teaching?
- Do you see it useful in non technical classes (such as programming classes)?
- Would you like to add or comment anything else?

A.2 Interview with GitHub Representative

- You mentioned (in the first email) that you work a lot with students and teachers on using GitHub in the education context. Can you tell more about this? How do you help them?
- In what scope do they use GitHub?
- What was their feedback? Most useful features? Biggest Hurdles?
- What do you think the motivation of educators to use GitHub?
- What other examples of usage of GitHub for the purposes of education have you seen (or heard of)?
- Any idea if teachers share materials/resources with each other?
- Do you see GitHub being useful to support learning and teaching? In non-technical aspects as well?
- What benefits does GitHub has, that might be useful for education?
- What challenges do you think people have in the context of education?
- How do you think these challenges can be mitigated?
- Do you think there is a way to resolve the 'privacy/patriot act/us located servers/university policy' problem? How?
- How does GitHub (the company) support educators at it's current state?
- Does GitHub plans to extend it's support? In what way?
- How important is it for GitHub (the company) to support GitHub in the context of learning, education and teaching? (since it's not its main core).
- Does GitHub directly trains educators/students in using GitHub for learning? What resources does it uses for that? (this might be covered in the first question)
- Would you like to add or comment anything else? Share any insights?

Appendix B

Interview Questions - Case Study

This appendix lists the interview questions I used in the study detailed in Chapter 5. The first section outlines the questions posed to each student, while the second and third sections list the questions asked of the teaching team.

B.1 Student Interviews

The first 7 items were asked of all students. As themes began emerging from the early interviews, I added the rest of the questions.

- Have you used GitHub before? In what capacity? If not, have you used any similar version control tools?
- How have you used GitHub in your class so far?
  - What do you think so far? Elaborate.
  - Have you contributed to the course repository in any way? How? Why or why not?
- Could you show me your project repository so far (if you've began your project)?
- Are you using GitHub in any other way than mandated in class? How?
- What are the challenges or drawbacks from using GitHub in your class so far? How do you think they can be mitigated?
- What are some features or things that GitHub could use to have assisted your class or group?

- What would you change, if anything, about how the course has used GitHub?
- What is your opinion on your course work being public in GitHub compared to being private in CourseSpaces?
- What is your opinion regarding the education of using tools like Git, GitHub, and other DVCS in UVic?
- Have you looked at other public Git repositories as you wrote your project? Why?
- What are your opinions on GitHub's use as an LMS as compared to other LMS like Connex or CourseSpaces?
- You had a lab or two where you were required to look at each other's work and make comments. What did you think of this exercise?

B.2 Course Instructor Interview

- Do you believe you benefited from the use of GitHub in your class? How?
- Do you believe the students benefited from the use of GitHub in your class? How?
- What were the main challenges you encountered? If a professor you know were trying it for their class, what would your advice be?
- Do you think you might use GitHub or similar tools in your classes in the future? Why or why not?
- What features of GitHub did you feel were the most useful for a classroom environment? Any features that you wish you utilized more of that the students would use more?
- What improvements would you suggest, if any, for GitHub to be useful as a learning tool?
- How do you think the students felt overall about using GitHub in this way?
- How do you feel GH as a learning tool compares to more traditional LMS or other tools you've used in your classes?

B.3 Lab Instructor Interviews

- What have you TA'd before this course? What tools were used then, if any?
- How much experience have you had with GH before the course?
- What was Yvonne's instructions in terms of using GitHub for the labs?
- What was Yvonne's instructions in terms of using GitHub for the labs?
- What are some of the difficulties you experienced with using GitHub, if any?
- Do you look at student projects? How? On their repositories?
- What do you think of GitHub as an LMS? What are the advantages and disadvantages compared to traditional LMS like Connex and CourseSpaces?
- Anything you think you would have wanted to do differently?

Appendix C

Questionnaire & Survey - Student Perspective

This appendix lists the questions used in the early questionnaire and in the validation survey for the study highlighted in Chapter 5. The first section lists the questions from the Questionnaire, given to the students soon after the course began. These questions were mostly open-ended questions. The second section lists the questions from the Survey, which served to validate or invalidate my findings. These questions were mostly Likert Scale type questions, where 1 indicated that they strongly disagreed with the question and 5 indicated that they strongly agreed.

C.1 Questionnaire

- How would you rate your familiarity with Distributed Version Control (such as Git, Mercurial, etc.)? [1-5]
- In which contexts have you used GitHub before? [Work, Class, Group Projects, Personal Projects]
- How would you rate your familiarity with GitHub? [1-5]
- From what you know of GitHub, what are your thoughts on how its use might benefit your education?
- From what you know of GitHub, what are your thoughts on the challenges of using it in your education?

- What is your favorite learning management tool? [Connex, CourseSpaces, Moodle, Other]
- What do you think are some of the most important features of that learning management tool? And why do you feel these features are important?
- If you have used GitHub in a class before, please discuss your experiences.
- Is there anything else you'd like to comment on?

C.2 Validation Survey

- What level of study are you in? [Second, Third, Fourth, Fifth+, Graduate Student]
- What program are you in? [Computer Science, Software Engineering, Other]
- Did we already interview you about this topic? Or did you sign up to be interviewed in the next two weeks? [Yes, No]
- If we haven't yet interviewed you and you are up for a 20-30 minute interview about this topic, please enter your email address so we can learn more about your responses!
- BEFORE this class, I have used Git & GitHub in the following contexts: [Class, Group Projects, Work, Projects]
- BEFORE this course, this was my level of familiarity with GitHub: [1-5]
- Currently, this is my level of familiarity with GitHub: [1-5]
- I enjoyed how GitHub overall how GitHub was used in our class, labs, and projects. [1-5]
- I will continue using GitHub for group projects. [1-5]
- I will continue using GitHub as a code repository for individual work. [1-5]
- GitHub was useful for organizing our group projects. [1-5]
- Being able to trace or account for my own or others' activities was useful to me. [1-5]

- The ability to easily view others' projects and comment on them made me feel more involved in the class [1-5]
- Others commenting on my work or answering my questions was useful for my projects. [1-5]
- The option of being able to make pull requests for the course is appealing to me. [1-5]
- I found using Git & GitHub for this course easy to learn and use. [1-5]
- Having all my school work and projects in one place is beneficial for my career. [1-5]
- Using GitHub in this class helped me learn a lot more about Git & GitHub. [1-5]
- Having learned a distributed version control system such as Git is beneficial for my career. [1-5]
- I don't like the idea of my work for school being out in the public. [1-5]
- I don't like the idea of my comments for my classes being out in the public. [1-5]
- I disliked that we had two systems set up for this course. [1-5]
- I felt there wasn't enough collaboration in this class or contribution from the students to justify using GitHub for the course. [1-5]
- I had difficulties learning how to use Git & GitHub for this class. [1-5]
- I feel this class should have offered a tutorial in using Git & GitHub right at the beginning. [1-5]
- I feel that Git, GitHub, and other Distributed Version Control Systems should play a bigger role in our education at UVic. [1-5]
- I feel that GitHub features are sufficient such that it can serve as a standalone Learning Management System [1-5]

- I think the GitHub file organization is much cleaner and easier to read than other LMS I've used. [1-5]
- These are some features that I think GitHub needs so that it can better support education.
- The workflow in using Git & GitHub in this class was sufficient. [1-5]
- I would like to see more classes in the future using GitHub as the primary class platform. [1-5]
- Do you have any further suggestions on improving the workflow of using GitHub to better support learning in your courses?
- Do you have any suggestions on how GitHub itself can be improved so it can better support learning in your courses?
- Is there anything else you'd like to add or comment on?
- If you would like to be notified of our results, please enter your email address below

Bibliography

[1] Alexander W Astin. Student involvement: A developmental theory for higher education. *Journal of college student personnel*, 25(4):297–308, 1984.

[2] Andrew Begel, Jan Bosch, and M-A Storey. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *Software, IEEE*, 30(1):52–66, 2013.

[3] Mordechai Ben-Ari. Constructivism in computer science education. In *Acm sigcse bulletin*, volume 30, pages 257–261. ACM, 1998.

[4] Mordechai Ben-Ari. Situated learning in computer science education. *Computer Science Education*, 14(2):85–100, 2004.

[5] Matt Bower. Groupwork activities in synchronous online classroom spaces. In *ACM SIGCSE Bulletin*, volume 39, pages 91–95. ACM, 2007.

[6] John Britton and Tim Berglund. Using version control in the classroom. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 753–753. ACM, 2013.

[7] Judith Borreson Caruso and Gail Salaway. The ecar study of undergraduate students and information technology, 2008. *Retrieved December*, 8:2007, 2007.

[8] Pu-Shih Daniel Chen, Amber D Lambert, and Kevin R Guidry. Engaging online learners: The impact of web-based learning technology on college student engagement. *Computers & Education*, 54(4):1222–1232, 2010.

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

[10] Ben Coleman and Matthew Lang. Collaboration across the curriculum: a disciplined approach todeveloping team skills. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 277–282. ACM, 2012.

[11] Betty Collis and Jef Moonen. *Flexible learning in a digital world: Experiences and expectations.* Psychology Press, 2001.

[12] Betty Collis and Jef 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.

[13] Miguel Á Conde, Francisco J García-Peñalvo, María J Rodríguez-Conde, Marc Alier, María J Casany, and Jordi Piguillem. An evolving learning management system for new educational environments using 2.0 tools. *Interactive Learning Environments*, 22(2):188–204, 2014.

[14] John W Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches.* Sage publications, 2013.

[15] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim 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.

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

[17] Christian Dalsgaard and Morten Flate Paulsen. Transparency in cooperative online education. *The International Review of Research in Open and Distance Learning*, 10(3), 2009.

[18] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 107–114. ACM, 1992.

[19] Stephen Downes. Feature: E-learning 2.0. *Elearn magazine*, 1, 2005.

[20] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer, 2008.

[21] Mohamed E Edrees. elearning 2.0: Learning management systems readiness. In *e-Learning" Best Practices in Management, Design and Development of e-Courses: Standards of Excellence and Creativity", 2013 Fourth International Conference on*, pages 90–96. IEEE, 2013.

[22] Katrina Falkner and Nickolas JG Falkner. Supporting and structuring contributing student pedagogy in computer science curricula. *Computer Science Education*, 22(4):413–443, 2012.

[23] Umer Farooq, John M Carroll, and Craig H Ganoe. Supporting creativity with awareness in distributed collaboration. In *Proceedings of the 2007 international ACM conference on Supporting group work*, pages 31–40. ACM, 2007.

[24] Francisco J García-Peñalvo, Miguel Á Conde, Marc Alier, and María J Casany. Opening learning management systems to personal learning environments. *Journal of Universal Computer Science*, 17(9):1222–1240, 2011.

[25] D Randy Garrison. *E-learning in the 21st century: A framework for research and practice*. Taylor & Francis, 2011.

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

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

[28] Lassi Haaranen and Teemu 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.

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

[30] John Hamer, Quintin Cutts, Jana Jackova, Andrew Luxton-Reilly, Robert McCartney, Helen Purchase, Charles Riedesel, Mara Saeli, Kate Sanders, and Judithe Sheard. Contributing student pedagogy. *ACM SIGCSE Bulletin*, 40(4):194–212, 2008.

[31] John Hamer, Andrew Luxton-Reilly, Helen C Purchase, and Judithe Sheard. Tools for contributing student learning. *ACM Inroads*, 2(2):78–91, 2011.

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

[33] Mehdi 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.

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

[35] Reynol Junco, Greg Heiberger, and Eric Loken. The effect of twitter on college student engagement and grades. *Journal of computer assisted learning*, 27(2):119–132, 2011.

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

[37] Beaumie Kim. Social constructivism. *Emerging perspectives on learning, teaching, and technology*, 1(1):16, 2001.

[38] Karel Kreijns, Paul A Kirschner, and Wim Jochems. The sociability of computer-supported collaborative learning environments. *Educational Technology & Society*, 5(1):8–22, 2002.

[39] Karel Kreijns, Paul A Kirschner, and Marjan Vermeulen. Social aspects of cscl environments: A research framework. *Educational Psychologist*, 48(4):229–242, 2013.

[40] George D Kuh. Assessing what really matters to student learning inside the national survey of student engagement. *Change: The Magazine of Higher Learning*, 33(3):10–17, 2001.

[41] Sheo Kumar, Anil Kumar Gankotiya, and Kamlesh 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.

[42] Anne Lacey and Donna Luff. *Qualitative data analysis.* Trent Focus Sheffield, 2001.

[43] Lisa M Lane. Toolbox or trap? course management systems and pedagogy. *Educause Quarterly*, 31(2):4, 2008.

[44] Timothy C Lethbridge, Jorge Diaz-Herrera, Richard J LeBlanc Jr, and J Barrie Thompson. Improving software practice through education: Challenges and future trends. In *Future of Software Engineering, 2007. FOSE'07*, pages 12–28. IEEE, 2007.

[45] Philip Machanick. A social construction approach to computer science education. *Computer Science Education*, 17(1):1–20, 2007.

[46] Steven R Malikowski, Merton E Thompson, and John 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.

[47] Catherine McLoughlin and Mark JW 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. Proceedings ascilite Singapore 2007*, pages 664–675, 2007.

[48] Nancy R Mead. Software engineering education: How far we've come and how far we have to go. *Journal of Systems and Software*, 82(4):571–575, 2009.

[49] Andrew Meneely and Laurie Williams. On preparing students for distributed software development with a synchronous, collaborative development platform. In *ACM SIGCSE Bulletin*, volume 41, pages 529–533. ACM, 2009.

[50] Shailey Minocha. A study of the effective use of social software to support student learning and engagement. 2009.

[51] Shailey Minocha and Peter G Thomas. Collaborative learning in a wiki environment: Experiences from a software engineering course. *New Review of Hypermedia and Multimedia*, 13(2):187–209, 2007.

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

[53] Tim O'Reilly. What is web 2.0? design patterns and business models for the next generation of software. 2005.

[54] Manoj Parameswaran and Andrew B Whinston. Social computing: An overview. *Communications of the Association for Information Systems*, 19(1):37, 2007.

[55] Santiago Perez De Rosso and Daniel Jackson. What's wrong with git?: a conceptual design analysis. In *Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software*, pages 37–52. ACM, 2013.

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

[57] Karen L Reid and Gregory V Wilson. Drproject: a software project management portal to meet educational needs. In *ACM SIGCSE Bulletin*, volume 39, pages 317–321. ACM, 2007.

[58] Guido Rößling, Mike Joy, Andrés Moreno, Atanas Radenski, Lauri Malmi, Andreas Kerren, Thomas Naps, Rockford J Ross, Michael Clancy, Ari Korhonen, et al. Enhancing learning management systems to better support computer science education. *ACM SIGCSE Bulletin*, 40(4):142–166, 2008.

[59] Guido Rößling, Myles McNally, Pierluigi Crescenzi, Atanas Radenski, Petri Ihantola, and M Gloria Sánchez-Torrubia. Adapting moodle to better support cs education. In *Proceedings of the 2010 ITiCSE working group reports*, pages 15–27. ACM, 2010.

[60] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. *Case study research in software engineering: Guidelines and examples.* John Wiley & Sons, 2012.

[61] Carolyn B. Seaman. Qualitative methods in empirical studies of software engineering. *Software Engineering, IEEE Transactions on*, 25(4):557–572, 1999.

[62] John Seely Brown. Open education, the long tail, and learning 2.0. *Educause review*, 43(1):16–20, 2008.

[63] Mary Shaw. Software engineering education: a roadmap. In *Proceedings of the conference on The future of Software Engineering*, pages 371–380. ACM, 2000.

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

[65] Christoph Treude, Fernando Figueira Filho, Brendan Cleary, and Margaret-Anne Storey. Programming in a socially networked world: the evolution of the social programmer. *The Future of Collaborative Software Development*, pages 1–3, 2012.

[66] Jason T. Tsay, Laura Dabbish, and James Herbsleb. Social media and success in open source projects. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion*, CSCW '12, pages 223–226, New York, NY, USA, 2012. ACM.

[67] William M Waite, Michele H Jackson, Amer Diwan, and Paul M Leonardi. Student culture vs group work in computer science. *ACM SIGCSE Bulletin*, 36(1):12–16, 2004.

[68] Etienne Wenger. Communities of practice: Learning as a social system. *Systems thinker*, 9(5):2–3, 1998.

[69] Jim Whitehead. Collaboration in software engineering: A roadmap. *FOSE*, 7(2007):214–225, 2007.

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