GitKit: Learning Free and Open Source Collaboration in Context

Grant Braught
Dickinson College
Carlisle, PA, USA
braught@dickinson.edu

Stoney Jackson
Western New England University
Springfield, MA, USA
hjackson@wne.edu

Cam Macdonell
MacEwan University
Edmonton, AB, Canada
macdonellc4@macewan.ca

Lori Postner
Nassau Community College
Garden City, NY, USA
lori.postner@ncc.edu

Wesley Shumar
Drexel University
Philadelphia, PA, USA
shumarw@drexel.edu

Karl R. Wurst
Worcester State University
Worcester, MA, USA
kwurst@worcester.edu

ABSTRACT

Modern version control tools and workflow practices are required skills for nearly all production software development, making them essential for students and in high demand among employers. Since these tools and processes were created for distributed, asynchronous collaboration on large scale projects, teaching them in an authentic context that makes clear their utility and design presents myriad challenges for both faculty and students. The GitKit is a snapshot of the FarmData2 Humanitarian Free and Open Source (HFOSS) project's artifacts (code, issues, documentation, etc.) frozen at a particular point in time and packaged with learning activities, an instructor guide, and a choice of containerized development environments. The GitKit thus provides students with the authentic context of a real-world project in which to learn and practice key Git and GitHub skills and workflows, while mitigating many of the challenges of doing so in an educational setting. The GitKit, including its learning activities and development environments are described in sufficient detail to encourage instructor adoption and feedback. A pilot study of student experiences with the GitKit is promising, suggesting that students gained an understanding of FOSS concepts and key skills, noticed automated guidance and feedback built into the development environment, and found it helpful in their learning. Future plans for the GitKit based on these surveys and instructor experiences with pilot uses are described along with plans for the development of *HFOSS Kits* for teaching and learning of other software development and aligned skills in authentic contexts.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; • **Software and its engineering** → *Open source model*.

KEYWORDS

Git, GitHub, Workflow, Curricula, Open Source, Humanitarian

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0423-9/24/03…$15.00
https://doi.org/10.1145/3626252.3630864

ACM Reference Format:
Grant Braught, Stoney Jackson, Cam Macdonell, Lori Postner, Wesley Shumar, and Karl R. Wurst. 2024. GitKit: Learning Free and Open Source Collaboration in Context. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630864

1 INTRODUCTION

Modern software development requires developers to know and understand version control tools and workflow practices. Many of these tools (e.g. Git/Mercurial, GitHub/GitLab) and workflows (forking/branching) were developed to facilitate collaboration in free and open source software (FOSS) communities [1, 15, 23, 28]. The usefulness of these tools and processes has now made them required skills for nearly all software development whether it be proprietary, FOSS, or research code. As a result these skills are in high demand [11] and are seen as essential for students [24].

However, because these tools and processes were created to facilitate large scale, asynchronous, distributed collaboration, understanding their utility and rationale can be challenging for students learning from small individual assignments. The alternative of learning these topics in a more authentic context, such as that of a live FOSS project, presents a wide range of challenges. For students, the complexity of the project can be overwhelming, obscuring key skills and concepts. Requiring students to interact with an unknown FOSS community can add unnecessary stress and anxiety to assignments, result in delays in receiving feedback, and can be frustrating or discouraging when communications are not at an appropriate level for students. Students may get "scooped" by another contributor who completes the task they are working on more quickly. While getting "scooped" does not negate the learning that happens, it can be quite demotivating. Similarly students wanting to repeat an activity for more practice may find it difficult to find opportunities at an appropriate level in a live project. For faculty, the constant evolution of a FOSS project quickly renders assignments obsolete, making it difficult to amortize the costs of creating high quality assignments across semesters. Additionally, having a class of students suddenly and/or repeatedly use a particular FOSS project for an assignment may place an unreasonable burden on its community.

The GitKit is an instance of a more general idea that we call *FOSS Kits* [18], which provide a general approach to teaching software development skills and concepts in more authentic contexts while

mitigating many of the challenges just described. A FOSS kit is a snapshot of an existing FOSS project's artifacts (code, issues, documentation, etc.) "frozen" at a particular point in time and packaged with learning activities, an instructor guide, and a containerized development environment. For example, the GitKit freezes the code, documentation and parts of the issue tracker from the FarmData2 project [9].

By freezing project artifacts a Kit can be repeatedly deployed in the same initial state. This insulates the Kit from the evolution of the live project, creating a stable instance of the project and allowing learning activities to be revised and reused over many semesters. Because the Kit is deployed separate from the live project, actions by the students do not disrupt or distract the live project. Similarly, instructors and students are able to take on project roles that may not be available to them in the live project (e.g. maintainers). Instructors are able to provide timely responses to the questions, issues, and pull requests at a level appropriate to the students' backgrounds. In the GitKit, for example, the instructor acts as a maintainer demonstrating to students how changes are merged upstream, ensuring that students are not "scooped", and introducing merge conflicts for students to resolve. The containerized development environments provided, called *KitClient*s, include a virtual assistant, called the *Kit-tty*, that gives students immediate context-sensitive guidance and feedback in response to common missteps. Other automations built into the kit simulate community interaction as students complete hands-on activities, providing more immediate responses than may be possible by the instructor.

Also, we have made the philosophical decision to base our FOSS Kits on *humanitarian FOSS (or HFOSS)* projects that have at their core a mission that broadly aims at the betterment of the human condition (e.g. medical care, disaster relief, etc) [7, 20, 25]. We refer to this subset of Kits as *HFOSS Kits*. These projects provide the same opportunities for technical learning as FOSS projects, but also bring the potential to broaden participation in computing. These projects provide an opportunity to motivate students by presenting computing, not just as technology, but as a powerful means for students to positively impact causes and communities that they care about [2, 8] and can help counter the "computing-is-coding myth" [19, 30]. Many of these projects connect to on-campus community engagement initiatives [13, 22, 29, 31] and to more general efforts on computing for social good [14]. There is growing empirical evidence that given the choice, most students prefer to work on projects with a human focus, and this preference is stronger among women and possibly also among underrepresented groups in the discipline [4, 16, 27].

The remainder of this article describes the GitKit in detail (Section 2), presents a pilot study of students' experiences with the GitKit (Section 3) and outlines our future plans (Section 4).

2 THE GITKIT

The GitKit is an HFOSS Kit that introduces students to fundamental skills and concepts used in collaborative software development. Its learning materials are organized into a four-topic sequence that guides students through a complete FOSS development workflow [1]. Each topic is described in Section 2.1.1. It has been used by students with varying levels of experience which we will elaborate on in Section 2.2.1 Students complete hands-on activities in one of the *KitClient* containerized development environments provided with the GitKit. To support students, these environments are pre-installed and configured with all required software and include the *Kit-tty* virtual assistant. In addition to hands-on activities for students, the GitKit supports faculty with slides and discussion activities for each topic, an instructor guide, and instructions for deploying the kit for use.

The authentic context for GitKit is based on a snapshot of the FarmData2 project. FarmData2 supports farmers in day-to-day operation and record keeping needs of their small organic diversified vegetable farms. However, no farming knowledge is necessary for completing GitKit. FarmData2 simply provides the authentic context of a full FOSS project while subtly highlighting a humanitarian application of software development. The snapshot of FarmData2 includes the full code base and documentation, as well as an issue tracker seeded with tickets captured from the project and tickets specifically created for GitKit. Issues described by the tickets that students address in GitKit require only small edits to Markdown documentation located in the root of the project. This allows students to work authentically in the project, but also to focus on the concepts, tools and processes, without being overwhelmed by technical details. Relatedly, FarmData2 was created as an *Education-Oriented HFOSS project* [3]. These are real-world FOSS projects that are intentionally designed to engage undergraduate students in their development.

2.1 Student Experience

This section gives an overview of the GitKit from the students' perspective. It describes each of the four GitKit topics, the available KitClients, illustrates how the Kit-tty assists students, and provides examples of the way GitKit simulates project community.

2.1.1 The GitKit Topics. The GitKit covers a sequence of four topics that begin by introducing students to FOSS communities and then guides them through the essential concepts, tools, and skills necessary to contribute to an existing FOSS project using a forking workflow. Each topic includes classroom materials and a hands-on student learning activity. The classroom materials introduce the important terminology, ideas, and concepts and provide opportunities for the instructor to round out the experience by illustrating some of the actions performed by project maintainers. The classroom materials focus on concepts and are largely tool and platform neutral. However, the hands-on activities are currently based on Git and GitHub and use VS Code/Codium. We are currently working to allow GitKit to also work on GitLab. Nominal use of the GitKit requires four 75-minute class periods, one for each topic, with students spending 2-3 hours between each period completing hands-on activities. However, as described in Section 2.2.1, instructors have found the materials adaptable to other deliveries.

Community and Collaboration. The first topic begins by introducing students to what Open Source communities are and how they collaborate to build software. Students begin by watching "Open Source Basics" (a.k.a. the "Cookie" video) [21]. This video introduces key vocabulary (e.g. *upstreaming, branch, fork*), roles (e.g. *contributor*, *maintainer*), and the overall collaborative process

of Open Source in a very approachable way. Additional in-class activities aim to further develop students' understanding of FOSS communities via exploration of the principles on which they operate (e.g. *transparency*, *inclusivity*, *meritocracy*) [17], and the additional roles that individuals may take on (e.g. *user*, *requester*, *leader*). The classroom materials then map the concepts of the Open Source contribution process to the technical terms that are used in the forking workflow (*upstream*, *fork*, *origin*, *clone*, *edit*, *push*, *pull request*). Diagrams like the example shown in Figure 1 are used throughout the materials to connect terminology and concepts to the tools and practices.

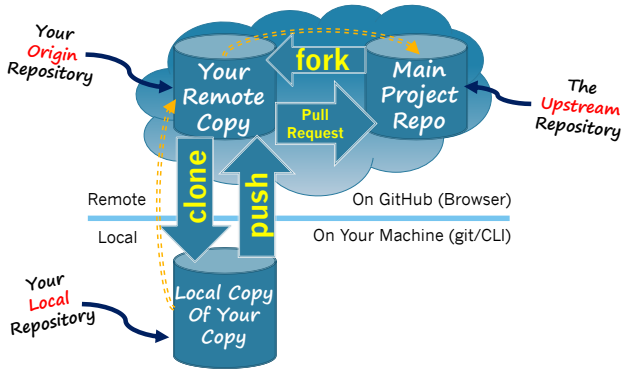

Figure 1: Vocabulary and steps in the basic forking workflow

In the hands-on learning activity students take their conceptual understanding and turn it into practice. They learn to use GitHub and Git to fork and clone the FarmData2 GitKit repository that has been deployed for their use (See 2.2.2). This first topic concludes with each student visiting the issue tracker and claiming an issue that is labeled with "Round 1". Each "Round 1" issue requires a small edit to one of the Markdown files in the root of the repository and have been engineered not to create merge conflicts.

Working Local and Upstreaming. The second topic focuses on making changes to a local copy of the project and offering them back to the *upstream* project. The classroom materials focus on concepts such as creating *feature branches*, and *staging* and *committing* changes to them. In the hands-on activities, students learn the `git` commands that realize these concepts and practice them to fix their "Round 1" issue. Lastly they carry out the *push* and *pull request* steps of the workflow. After completing the second topic, each student has made a pull request that fixes their "Round 1" issue.

Staying Synchronized. The third topic begins with the instructor demonstrating how a project maintainer reviews and merges pull requests into the upstream. As students' pull requests are merged it becomes evident that students' local and origin repositories are now "out of synch" with the upstream. The process of pulling the *main* branch from upstream and pushing it to origin is introduced as the means of "synchronizing" with upstream. In the hands-on activities students learn and use the `git` commands for synchronizing their repositories with upstream. The activity concludes with students practicing the skills and concepts from the first two topics by claiming a "Round 2" issue and creating a pull request to fix it. Contrary to the "Round 1" issues, the "Round 2" issues are engineered so that merge conflicts can be introduced.

Merge Conflicts. The final topic focuses on the causes and resolutions of merge conflicts. Prior to this activity, the instructor creates a pull request for a feature branch included in the GitKit that contains changes specifically designed to cause a merge conflict with each of the "Round 2" issues. Reviewing this pull request with students illustrates how the changes in their pull request might conflict with a pull request from another contributor. After this pull request is merged, GitHub indicates that the students' Round 2 pull requests can no longer be merged automatically. The classroom materials show how conflicting changes are detected and differentiated from non-conflicting changes and how a merge tool can be used to resolve the conflicts. In the hands-on activities, students use what they learned in the previous topic to synchronize their main branch with the upstream repository. They learn the `git` commands to merge those changes into their feature branch, creating the merge conflict. They then use a merge tool to resolve the conflict, stage and commit the changes, and push them to update their pull request. Students complete this topic by observing that they have resolved the merge conflict as evidenced by GitHub indicating that their pull request can again be merged automatically.

2.1.2 The KitClients. A KitClient is a containerized development environment for FOSS Kits, including the GitKit, in which students complete the learning activities. There are currently two KitClients that can be used with GitKit. They are functionally equivalent. An instructor can choose which to use based on their class and students. One KitClient is a Linux system with its own GUI desktop that runs in a container and that students interact with in a window on their machine. The other KitClient runs within Visual Studio Code as a *Dev Container* and students interact with it through VS Code on their machine. Whichever KitClient is chosen, students complete all of the GitKit activities within the KitClient, ensuring that they have the necessary tools and configuration. The use of a KitClient is also what allows the GitKit to install the Kit-tty to assist students in completing their assignments (see Section 2.1.3). To run the KitClient, students themselves need only install Docker and either a VNC Client (recommended for the Linux KitClient), or VSCode and Git (required for the VSCode KitClient). Instructions for these installations and starting the KitClients are relatively short and are provided with GitKit.

2.1.3 The Kit-tty. The Kit-tty[1] is a virtual assistant that a FOSS Kit, like the GitKit, customizes and adds into the KitClient when a student begins using it to work on the Kit. Each kit customizes the Kit-tty to catch common student errors and provide hints on how to perform activity steps correctly. For example, in early uses of the GitKit (pre Kit-tty) we observed students frequently committing changes to the main branch rather than to their feature branch, an action that should not happen in the forking workflow. This error was typically not discovered until students had progressed several steps further into the activity, and requires more advanced Git skills to undo. Now when a student attempts to commit to the main branch the Kit-tty prevents the commit and responds with a helpful message as shown in Figure 2. The GitKit adds other Kit-tty interventions that occur when students attempt to: merge a feature

branch into main (instead of vice versa); set the upstream remote to the origin (instead of the upstream); clone the upstream (rather than their fork); and clone one repository inside of another.

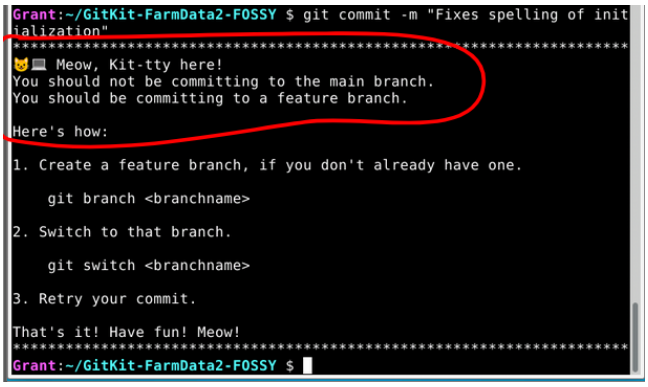

Figure 2: The Kit-tty preventing a commit to the main branch.

2.1.4 Community Automations. When working on a FOSS project, contributors regularly interact with maintainers and other community members through comments on tickets in the issue tracker and pull requests. The GitKit, and FOSS kits more generally, can include automations (e.g. GitHub Actions [12]) that simulate this type of interaction for some common scenarios encountered in learning activities. For example, when completing the first GitKit activity students request to be assigned an issue by adding a comment to its ticket in the issue tracker. An automation notices this comment, assigns the issue to the student (if it hasn't already been assigned to someone else) and responds personally as a maintainer might: "Great! I assigned you (@TheirUsername) to the issue. Have fun working on it!" Several additional automations are planned for future development (See Section 4).

2.2 Faculty Experience

This section provides additional information relevant to an instructor considering using the GitKit.

2.2.1 Delivering the GitKit. The instructor guide for the GitKit outlines a typical use case for the GitKit using the classroom materials and the hands-on activity for each of the four topics. This use case assumes that the GitKit is the students' first formal exposure to FOSS and the use of Git/GitHub. It requires four 75-minute class periods and 2-3 hours outside of each class for students to complete the hands-on activities. However, instructors have found the GitKit to be adaptable to different educational settings, student experience levels and learning objectives. For example, an instructor with students who have had prior exposure to Git fundamentals (but not GitHub or the forking workflow) have skipped most of the class materials and used the hands-on activities as in-class lab activities rather than homework. Another organization delivered the GitKit as a one-day workshop for students from low-income, first-generation, underrepresented minority backgrounds. Another instructor is currently developing a set of Process Oriented Guided Inquiry (POGIL) [26] activities to be used in place of GitKit's more traditional classroom materials. In lower level courses, or with less experienced students, the first two topics could be spread over a longer time and be used as a cohesive unit without continuing onto the final two topics. It is worth mentioning that all of the work on GitKit, and FOSS kits more broadly, is being conducted under open licenses and welcomes participation, contribution and derivative work. See Section 2.2.3 for more information and pointers to the project resources.

2.2.2 Deploying GitKit. A faculty member wanting to use GitKit in a class must deploy one or more instances of the Kit. Each deployed instance contains a repository based on the "frozen" FarmData2 and a populated issue tracker that will support up to 32 students (limited by the number of "Round 1" issues). To deploy an instance of the GitKit, the faculty member runs a single docker command providing it with the name of a GitHub organization where the kit should be deployed and a GitHub personal access token (PAT). That command creates the repository, populates the issue tracker, and provides the branch used to introduce the merge conflicts at the start of the fourth topic. The students in the class are then given the URL of the deployed GitKit to use as their upstream repository for the hands-on activities. To support more than 32 students, an instructor can deploy multiple instances of the GitKit as shown in Figure 3. Complete instructions for deploying the GitKit are given in the instructor guide.

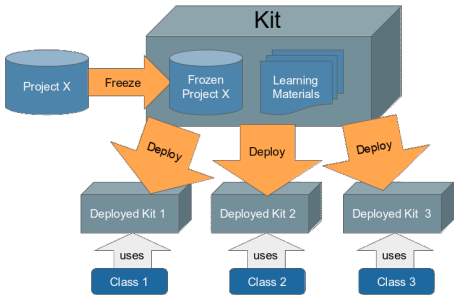

Figure 3: A FOSS Kit with multiple Deployments

2.2.3 Licensing. All of the classroom materials, hands-on student activities and instructor materials for the GitKit are provided for use and adaptation under a Creative Commons CC-BY-NC-SA license [5] at: https://gitlab.com/hfossedu/kits/GitKit. The documentation in the "frozen" FarmData2 repository is used under a CC-BY-SA license [6] and its code is used under the GNU Public Licence v3 (GPL3) [10]. In addition, the KitClients and the tools used for creating and deploying FOSS kits (including the GitKit) are also openly available under the GPL3 license at https://gitlab.com/hfossedu/kits.

3 METHODS & RESULTS

To evaluate the GitKit as an intervention, a pre-post survey was designed to measure two things: the change in students' understanding of and confidence in using FOSS tools due to the GitKit intervention and students' experiences using the GitKit as a tool.

[1]Kit-tty is pronounced "kitty" and is a play on kit and tty, as the kit-tty output appears in the tty ;) .

Table 1: Pre-Post Survey Averages [2]

Survey Questions	Pre-Avg All	Post-Avg All	Pre-Avg Women	Post-Avg Women	Pre-Avg Men	Post-Avg Men
I can explain what it means for software to be Free and Open Source (FOSS).	3.96	4.34	3.88	4.20	4.00	4.38
I want to contribute to FOSS in the future.	3.90	4.09	4.00	4.30	3.82	3.94
Software developers can use their skills with FOSS to make society better.	4.39	4.46	4.55	4.40	4.30	4.44
I can use version control software (e.g. git).	3.81	4.34	3.88	4.30	3.73	4.44
I can use version control software to resolve code conflicts (e.g. merge changes).	3.45	4.15	3.55	4.30	3.34	4.16
I can explain the purpose of pull/merge requests.	3.63	4.43	3.44	4.20	3.82	4.55
Software development is collaborative.	4.54	4.65	4.77	4.70	4.43	4.66

Table 2: Post Survey Averages [2]

Survey Questions	Avg All	Avg Women	Avg Men
the Kit was easy to set up on my computer	3.84	4.20	3.77
the Kit set up instructions were easy to follow	4.21	4.30	4.38
the Kit activities helped me understand the concepts of version control	4.18	4.20	4.33
the Kit activities helped me learn the mechanics of version control	4.15	4.20	4.27
I completed the Kit activities	4.25	4.60	4.05
the Kit activities were difficult	2.78	3.10	2.72
the Kit helped me understand how software development is collaborative	4.12	4.30	4.27
the Kit activities required a reasonable amount of in-class time	3.65	3.90	3.66
the Kit activities required a reasonable amount of out-of-class time	3.5	3.60	3.27
I was motivated by using a project with real world clients	3.28	3.60	3.05
I was motivated by using a project that will help other people	3.34	3.70	3.11
I was motivated by using a project/tools I will see in the future	3.78	3.80	3.77

A pilot survey was conducted after students used the GitKit during the fall 2022 semester. Upon review of the data it appeared that students misunderstood several of the questions. The survey was revised into a pre-post survey, carefully rewording questions to be both clearer and more specific. The revised pre-post survey was IRB approved and administered to students at two institutions in spring 2023. These students used the GitKit in 2nd year courses required for Computer Science majors. A total of 33 pre-surveys and 32 post-surveys were analyzed.

3.1 Instruments

The goal of the pre-survey was to gather information about students' perceptions of their knowledge of skills and concepts that would be covered in the GitKit prior to the intervention. The goal of the post-survey was to measure any change in students' perceptions of their knowledge of the skills and concepts and also to learn about students' experiences using the GitKit. The seven questions shown in Table 1 were included in both the pre- and post-surveys, allowing an analysis of how their perceptions of their knowledge was changed by the intervention. The twelve statements shown in Table 2 were added to the post-survey to gain insights into students' experiences using the GitKit. On both the pre- and post-survey, students were asked to rate their agreement with each of the statements using a 5-point Likert scale: Strongly Disagree (1), Disagree (2), Neutral (3), Agree (4), Strongly Agree (5). The demographic questions about age, gender identity and race/ethnic origin were included at the end of each survey.

[2] Some Avg All columns are less than averages for men and for women because a small number of students were excluded from the gender analysis, see section 3.2

3.2 Data Analysis

The pre-survey respondents were 9 women, 23 men and 1 non-binary. The post-survey respondents were 10 women, 18 men, 1 non-binary and 3 students who chose not to specify a gender. A gender analysis for women and men was completed to see if there were any gender differences. The non-binary and the students who did not answer the gender identity question were eliminated from this analysis to reduce the risk of deanonymization due to small sub-group samples. Race-ethnicity data was collected but not used in the analysis as the sample groups were not large enough. Since the number of surveys is small, the analysis is descriptive and does not focus on statistical significance.

3.2.1 Pre-Post Survey Results. There were seven questions asked in both the pre and post surveys about students' understanding of FOSS and FOSS tools. Table 1 demonstrates that the average student rating on all the questions were higher on the post-survey than

on the pre-survey. The two questions regarding merge conflicts and pull requests showed the largest gains. Table 1 also contains the results for the same questions disaggregated by gender identity and shows similar gains for both women and men in the questions about merge conflicts and pull requests. However, men also had a large gain in reported agreement with version control in general. There are two questions where the averages for women went down between the pre and post surveys. Each of these were heavily influenced by a single student selecting Disagree.

3.2.2 Post Survey results. The post-survey contained questions about the GitKit use. Table 2 shows the average rating for each question for all students as well as the breakdown by gender identity. The set up questions about the GitKit received high ratings with a mean of over 4.2. Students rated the questions about learning version control and understanding the collaborative nature of software development with high ratings of over 4.0 on average. Students did not indicate that the GitKit activities were difficult and most students reported completing them. Students seemed to find the amount of time dedicated to the activities both in and out of class to be reasonable. Men and women rated most of the statements similarly except for two of the motivation questions. Women found the real-world context and the ability to help others as more motivating than their male counterparts.

Of the 32 students who completed the post-survey, 18 reported seeing at least one Kit-tty message. Of those, 13 stated that they found the Kit-tty message helpful. Sample comments are,

- *Yes because it explains exactly what you were doing wrong so you can correct it in the future.*
- *Yes, Kit-tty cleared some confusion for me when I was having a git issue.*
- *Yes the Kit-tty messages were helpful because they prevented you from moving forward with any potentially dangerous commands. They also provided you with hints or tips on what [t]o do before you commit a command.*

When asked if there was anything else the student would want to provide feedback about one student stated,

- *First thing is Kit-tty is very helpful in preventing many mistakes. Secondly, KitClient is useful when I don't want those file to stay or stuck in my local machine. Also KitClient has many configuration set up already, so it's convenient. One [of] the bad but not worst thing about KitClient is about its setting up, which many people encountered some kinds of errors during the set up.*

3.3 Discussion

The data collected indicates that students gained an understanding of FOSS concepts and tools through their use of the GitKit. This documents that the GitKit does cover the intended concepts in a way that students can comprehend. The built-in support of Kit-tty was noticed by students and many found it helpful in their learning. This data is being used to inform improvements to the GitKit and students will be asked to complete the pre-post survey in future.

4 FUTURE WORK

Encouraged by the results of the pilot study, we have many plans for improving GitKit and developing other Kits.

Student-side improvements planned for GitKit include: adding additional simulations of community interaction, such as checking issue linking in student pull requests; adding the ability for students to check their work at predefined points; adding the ability to reset the GitKit back to a previous point; translating activities into a free interactive e-book platform allowing asynchronous learning.

Instructor-side improvements planned for GitKit include: allowing deployment to GitLab (not just GitHub); adding tools to make it easier to capture project artifacts and create Kits from those artifacts; instrumenting GitKit to capture student data for future research into student learning and refinement of the GitKit and activities; and translating activities into team-based, POGIL-style in-class activities.

In addition to GitKit, the following FOSS Kits are currently under development: Microservices; Issue Tracker/Ticket Writing; Education-oriented HFOSS project-specific Kits for on-boarding students; Sprint Planning; and Scrum Team Development Workflow.

We are also interested in developing the following FOSS Kits: a CI/CD Kit; a Testing Kit (e.g. unit or e2e testing); Kits for assignments to be paired with specific textbooks (e.g. refactoring, design patterns, data structures, algorithms, etc.).

5 CONCLUSION

In this paper, we introduced GitKit which contains learning activities, an instructor guide, and a snapshot of the FarmData2 project. When deployed, GitKit provides a repeatable, authentic, and isolated development environment with simulated community responses, under the control of instructors and their students, for learning modern, distributed version control systems with a minimum of software dependencies. It includes a four-topic lesson that is typically delivered over four 75-minute sessions and four homework assignments. To interact with the learning environment, instructors and students choose between one of two KitClients: a complete Linux development environment in a container including a GUI, or a VS Code Dev Container. These clients provide a consistent interface for lessons and enable Kit-tty which acts as an automated personal assistant, providing students with instant, useful, context sensitive feedback as students work through the lessons.

This paper also described a pilot study that administered a pre-post survey, before and after using GitKit, to 2nd year computer science majors at two institutions. The data collected indicates that students gained an understanding of FOSS concepts and tools through their use of GitKit, and that the built-in support of Kit-tty was noticed and many found it helpful in their learning.

If you would like to use GitKit, find other Kits, or help improve or develop them, please join us at https://gitlab.com/hfossedu/kits.

ACKNOWLEDGMENTS

This work was supported under National Science Foundation Grants DUE- 1225738, 1225688, 1225708, 2012966, 2013069, 2012979, 2012999, and 2012990. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

REFERENCES

[1] Atlassian. Unspecified. *Forking Workflow.* Atlassian. https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow

[2] S. Beyer. 2008. Predictors of Female and Male Computer Science Students' Grades. *Journal of Women and Minorities in Science and Engineering* 14 (2008), 377–409. https://doi.org/10.1615/JWomenMinorScienEng.v14.i4.30

[3] Grant Braught, Steven Huss-Lederman, Stoney Jackson, Wes Turner, and Karl R. Wurst. 2023. Engagement Models in Education-Oriented H/FOSS Projects. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 409–415. https://doi.org/10.1145/3545945.3569835

[4] Grant Braught and Farhan Siddiqui. 2022. Factors Affecting Project Selection in an Open Source Capstone. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (Dublin, Ireland) *(ITiCSE '22)*. ACM, New York, NY, USA, 358–364. https://doi.org/10.1145/3502718.3524760

[5] Creative Commons. Unspecified. *Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)*. Creative Commons. https://creativecommons.org/licenses/by-nc-sa/4.0/

[6] Creative Commons. Unspecified. *Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-SA 4.0)*. Creative Commons. https://creativecommons.org/licenses/by-sa/4.0/

[7] Heidi J. C. Ellis. 2022. *Humanitarian FOSS*. TeachingOpenSource. http://teachingopensource.org/hfoss/

[8] Heidi J. C. Ellis, Ralph A. Morelli, Trishan R. de Lanerolle, Jonathan Damon, and Jonathan Raye. 2007. Can Humanitarian Open-source Software Development Draw New Students to CS?. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Covington, Kentucky, USA) *(SIGCSE '07)*. ACM, New York, NY, USA, 551–555. https://doi.org/10.1145/1227310.1227495

[9] FarmData2. 2022. *The FarmData2 Project.* FarmData2. https://github.com/DickinsonCollege/FarmData2

[10] Free Software Foundation. 2007. *GNU GENERAL PUBLIC LICENSE*. Free Software Foundation. https://www.gnu.org/licenses/gpl-3.0.txt

[11] The Linux Foundation. 2022. The 10th Annual Open Source Jobs Report: Critical Skills, Hiring Trends, and Education. https://linuxfoundation.org/tools/the-10th-annual-open-source-jobs-report/

[12] GitHub. 2023. *GitHub Actions documentation.* GitHub. https://docs.github.com/en/actions

[13] Mark H. Goadrich, Michael Goldweber, Matthew C. Jadud, Sarah Monisha Pulimood, and Samuel A. Rebelsky. 2019. Civic Engagement Across the Computing Curriculum. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE 2019, Minneapolis, MN, USA, February 27 - March 02, 2019*, Elizabeth K. Hawthorne, Manuel A. Pérez-Quiñones, Sarah Heckman, and Jian Zhang (Eds.). ACM, New York, NY, USA, 649–650. https://doi.org/10.1145/3287324.3287335

[14] Mikey Goldweber, Lisa Kaczmarczyk, and Richard Blumenthal. 2019. Computing for the Social Good in Education. *ACM Inroads* 10, 4 (nov 2019), 24–29. https://doi.org/10.1145/3368206

[15] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-Based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 345–355. https://doi.org/10.1145/2568225.2568260

[16] Pawel Grabarczyk, Alma Freiesleben, Amanda Bastrup, and Claus Brabrand. 2022. Computing Educational Programmes with More Women Are More about People & Less about Things. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (Dublin, Ireland) *(ITiCSE '22)*. ACM, New York, NY, USA, 172–178. https://doi.org/10.1145/3502718.3524784

[17] Red Hat. 2023. *The Open Source Way.* Red Hat. https://opensource.com/open-source-way

[18] Stoney Jackson, Karl R. Wurst, Grant Braught, and Cam Macdonell. 2023. Kits: Creating Repeatable Learning Experiences Using Real HFOSS Projects. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3545945.3569835

[19] Ralph Morelli, Allen Tucker, Norman Danner, Trishan R. De Lanerolle, Heidi J. C. Ellis, Ozgur Izmirli, Danny Krizanc, and Gary Parker. 2009. Revitalizing Computing Education Through Free and Open Source Software for Humanity. *Commun. ACM* 52, 8 (Aug. 2009), 67–75. https://doi.org/10.1145/1536616.1536635

[20] R. A. Morelli, Heidi J. C. Ellis, Trishan de Lanerolle, Jonathan Damon, and Christopher Walti. 2007. Can student-written software help sustain humanitarian FOSS?. In *The 4th International Conference on ISCRAM*. 41–44.

[21] Sarah Moyle. 2014. *Open Source Basics.* Sarah Moyle. https://www.youtube.com/watch?v=upxUAI-fAtE

[22] Christian Murphy, Kevin Buffardi, Josh Dehlinger, Lynn Lambert, and Nanette Veilleux. 2017. Community Engagement with Free and Open Source Software. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. ACM, New York, NY, USA, 669–670. https://doi.org/10.1145/3017680.3017682

[23] Dirkjan Ochtman. 2011. Mercurial. In *The Architecture of Open Source Applications*, Greg Wilson Amy Brown (Ed.). Vol. 1. lulu, Chapter 12.

[24] The Joint Task Force on Computing Curricula. 2023. Software Engineering (SE). In *Computer Science Curricula 2023*. 314–336. https://csed.acm.org/wp-content/uploads/2023/03/Version-Beta-v2.pdf

[25] Esteban Parra, Sonia Haiduc, and Rebecca James. 2016. Making a Difference: An Overview of Humanitarian Free Open Source Systems. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 731–733.

[26] POGIL. 2023. *What is POGIL?* POGIL. https://pogil.org/what-is-pogil

[27] Lori Postner, Gregory W Hislop, and Heidi J.C. Ellis. 2023. Humanitarian Applications Increase Interest and Motivation of Women in Computing. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 416–422. https://doi.org/10.1145/3545945.3569832

[28] Susan Potter. 2011. Git. In *The Architecture of Open Source Applications*, Greg Wilson Amy Brown (Ed.). Vol. 2. lulu, Chapter 6.

[29] Sarah Monisha Pulimood, Kim Pearson, and Diane C. Bates. 2020. Encouraging CS students to compute for social good through collaborative, community-engaged projects. *SIGCAS Comput. Soc.* 49, 1 (2020), 21–22. https://doi.org/10.1145/3447892.3447900

[30] Allen Tucker, Ralph Morelli, and Trishan de Lanerolle. 2011. The humanitarian FOSS project: Goals, activities, and outcomes. In *Global Humanitarian Technology Conference (GHTC), 2011 IEEE*. IEEE, IEEE, Champaign, IL, USA, 98–101.

[31] Karl R. Wurst, Christopher Radkowski, Stoney Jackson, Heidi J. C. Ellis, Darci Burdge, and Lori Postner. 2020. LibreFoodPantry: Developing a Multi-Institutional, Faculty-Led, Humanitarian Free and Open Source Software Community. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 441–447. https://doi.org/10.1145/3328778.3366929