Gitsubmit and VeCVL: Integrating Version Control in Introductory Computer Science Education

Nathan W. Eloe
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468, USA
nathane@nwmissouri.edu

Abstract

Version control systems (VCS), such as Subversion and Git, are pervasive in industry; they are invaluable tools for collaborative development that allow software engineers to track changes, monitor issues, merge work from multiple people, and manage releases. These tools are most effective when they are a part of a developer's habitual workflow. Unfortunately, the use of these powerful tools is often taught much later in a developer's educational career than other tools like programming languages or databases. Even an experienced student's first experience with version control can be unpleasant. In this work, an assignment submission system built around the Git version control system is introduced and analyzed for usability and suitability for use in entry level computer science classes.

Keywords- *computer science education, education technology, pedagogy, version control, visual language*

1 Introducion

Version control tools and methodologies are essential tools in the increasingly collaborative environment. The size and complexity of many modern software development projects require the talent and time of multiple developers working together. While group projects and collaboration are a mainstay of computer science classes, the tools that are used in industry to promote teamwork are often not introduced until later in a computer science curriculum (such as in a Software Engineering course).

Version Control is most effective when it is interacted with on a regular basis. It should ideally become part of the developer's regular workflow. Introducing such an invaluable tool so late in the curriculum forces students to recreate their workflow by breaking bad habits that have been reinforced through early computer science courses and integrating industry best practices. Additionally, the use of these best practices may or may not be reinforced in future classes, requiring the student to self motivate in maintaining the use of those skills.

This paper examines an implementation of a submission system built around the Git version control system that simplifies the process of interacting with version control to only the important steps in a simple workflow (clone, add, commit, push). The interface developed for students implements the Version Control Visual Language (VeCVL) [4]. Additionally, there are tools for teachers and graders to manage assignments and grade submissions. The process aims to simplify the assignment distribution process and reinforce version control workflow in pedagogy, not just as a topic in a course. The student interface and visual language is analyzed from a usability perspective through an evaluation using the Cognitive Dimensions of Notations [10].

2 Background and Related Work

2.1 Git Workflow

Git [2] is a Distributed Version Control System that exhibits a great amount of flexibility to allow powerful and varied workflows [3, 8] to be designed around it. These workflows primarily differ in their approaches and timings for branching and merging. Beyond the branching, the workflows use the same basic cycle of operations for an individual developer: make changes to the local repository, add the changes to the index, and commit the changes (marking them with a commit message). Once a task has been completed, the developer can sync their changes to a remote server (if using a remote for collaboration or backup). This cycle repeats itself, with the developer pulling changes from a remote repository, making local changes, and pushing the changes to the remote repository.

DOI reference number: 10.18293/DMSVLSS2017-005

2.2 VeCVL and Scaffolding

Gitsubmit was developed in concert with VeCVL [4] as a method of introducing version control in education through *scaffolding* [1], and can be seen as the initial implementation of the visual language, expanded to be a full GUI instead of an icon set. VeCVL is a visual representation of the general steps present in version control system in a way that conveys direction of the changes' movement. Section 3 contains an examination of Gitsubmit and its close ties to VeCVL, as well as a discussion of the deviations from the icon set introduced in [4] to adapt the visual language to a full GUI.

2.3 Git in Education

Version control is a topic that is increasingly being introduced in computer science curricula. Significant amounts of research focus on going beyond introducing version control as a topic in a course to embedding the use of version control into pedagogy [11, 5, 1]. As version control becomes more prevalent in industry, computer science education should move to embrace these technologies and introduce them to students.

Code hosting services such as GitHub are joining in the efforts to educate computer science students about version control by offering programs like GitHub Classroom [6] and offering students free benefits when using their services [7].

2.4 Cognitive Dimensions of Notations

The original 14 Cognitive Dimensions of Notations, as introduced by Green [10] are used to evaluate the usability of an existing interface or appropriateness of the method of information delivery. As the student interface to Gitsubmit is aimed at exposing only the needed functionality for a simple git-based workflow, evaluation of the interface will be done by students using the interface in classes (who may have varying levels of experience with git). Certain dimensions, such as Abstraction Gradient, can only be evaluated by developers proficient with the workflow, who are familiar enough with the steps to know whether more details can be encapsulated.

This paper focuses on the following dimensions:

- Diffuseness/Terseness: how many symbols are needed to express a solution?
- Error-proneness: how well does the interface protect the user from mistakes?
- Hard Mental Operations: How much additional "processing" must a user of the system do to complete a solution? How much additional information is needed that is not tracked by the UI?
- Premature Commitment: Is there a firm ordering of steps needed to express a solution? Can a user go back and correct mistakes?
- Progressive Evaluation: Does the UI provide enough feedback as to the current state of the solution?
- Role-expressiveness: How clear is the meaning of each symbol, and the role it plays in the solution?

Some of the dimensions require analysis from people who are skilled in the domain solution. These include:

- Abstraction Gradient: How much can be abstracted by the notation?
- Closeness of Mapping: How well does the notation correspond to the problem?

3 Gitsubmit

Gitsubmit is a submission system for programming assignments with three main parts: a student interface, an instructor interface, and a Git hosting system. What follows is a brief discussion of the hosting system and instructor interface, and an in-depth analysis of the student UI.

3.1 Hosting System

Currently, Gitsubmit uses a self-hosted instance of Gitlab [9] as the hosting backend. This hosting solution was chosen to allow full control of user creation, authentication, and project visibility. The system itself is not tied to Gitlab, and could be modified easily to use other well known hosting solutions such as GitHub or BitBucket. This would require cooperation from these respective companies to set up this functionality, but is an attractive option, as this would move the host system administration away from the instructor.

3.2 Instructor Interface

The instructor interface is a simple interface to automate the creation of a class, the assignment of a project to a class, and the fetching of the student submissions for a specific assignment. The interface is not designed to be a replacement for a full-fledged git client (and indeed abstracts all of the git operations away from the instructor). When discussing the different functionalities of the instructor UI, Gitlab terminology is used; where possible the corresponding concepts from GitHub and BitBucket have been provided.

3.2.1 Creating a Class

A class in Gitsubmit maps to the concept of a *group* in Gitlab (similar to Organizations in GitHub or Projects in BitBucket). The instructor provides the semester, course number, and course name, and a new group is created with the instructor added as a group administrator. The group name is formatted to contain this information (for example, the group W2017.44242.Data_Structures is the Data Structures class (course number 44-242 at Northwest Missouri State University) that is running during the Spring/Winter 2017 semester). Graders and TAs can be added to this group as group administrators as well (allowing them automatic access to student submissions for grading and assistance).

The instructor also provides a CSV file containing student emails and student names. The UI creates a roster in a git repository that contains a mapping of all students in the class to their Gitlab user. If any student does not exist in Gitlab, a user is automatically created. This is one of the differences between GitHub Classroom and Gitsubmit; Gitsubmit removes the need for the student to create their own user on the system. One of the stated goals of Gitsubmit is to remove the parts of the process that are not directly tied to integrating the workflow into the students' everyday development cycle. It also allows a consistent naming convention for student user IDs.

3.2.2 Creating an Assignment

To create an assignment in the instructor interface, the teacher selects a class, names the assignment, and provides either a skeleton directory or a Markdown formatted project description. Optionally, the instructor may also specify a CSV of groups (based on student ID) if the project being assigned is a group project.

The interface fetches the roster from the Gitlab, and creates a repository in the Gitlab group for each individual or student team. The skeleton or description is then pushed to each repository, and the appropriate students are added to the repositories with the Developer role; this allows the pupil to push and pull code from the repository, but not change access permissions (and allow other students to see their submission or working progress). Finally, the interface creates a single repository that contains every student repo as a git submodule. This repository enables efficient fetching of the student submissions with only a few git commands.

This assignment structure is one way Gitsubmit differentiates itself from GitHub Classroom; the assignment structure is designed to not give students the ability to modify the permissions on the assignments. In this way, assignment confidentiality is preserved.

3.2.3 Fetching Student Submissions

Fetching student submissions can be done easily from a command line with four simple commands:

```
git clone <url_of_grading_repo>
cd <grading_repo>
git submodule update --init --recursive
git submodule foreach \
    git pull origin master
```

In order to remove the need for the instructor to drop to the command line, the instructor interface allows the teacher or grader to select an assignment for a class and download all student repositories for that assignment.

3.3 Student Interface

The student interface (Figure 1) was designed to be a very simplified Git client that supports the basic operations needed to use Git as a submission system: clone, push, pull, add, and commit. The icons and UI were designed in tandem with VeCVL [4].

To begin an assignment, the student selects the semester, course, and the assignment (Figure 2). The list of commits in the repository is shown in the right-most pane. To begin the selection, the student clicks the Clone/Pull button (circled in Figure 2). Colors as well as icons are used to indicate the status of the commits. Commits that reside on the server but not locally are indicated with an orange (or red if the commit is the HEAD of origin/master) ID and the Clone/Pull icon. Commits that reside only on the local machine are blue and indicated with the Push icon. If a commit is indicated with a green ID and a check icon, the commit is the HEAD commit on the remote repository and also exists on the local repository (and is the commit that will be graded). Gray commit IDs are common to both local and remote.

After an assignment is started, the student can complete the work normally; using whatever IDEs or other tools are used in the course. As files are modified or added, they appear in the "Unstaged Changes" pane. Students can select which changes should be submitted. The changes are chosen using the add button (circled in Figure 3). Once the student has selected the changes to submit, a commit message can be specified describing what the changes were, and the commit finalized by hitting the Commit button (as circled in Figure 4).

When the student stops working on a n assignment, they can push their work to the server with the Push button (circled in Figure 5). Note the colors indicating the states of the commits in the repositories. The student finalizes their submission by clicking the push button. A successful submission is indicated with a green check mark next to the

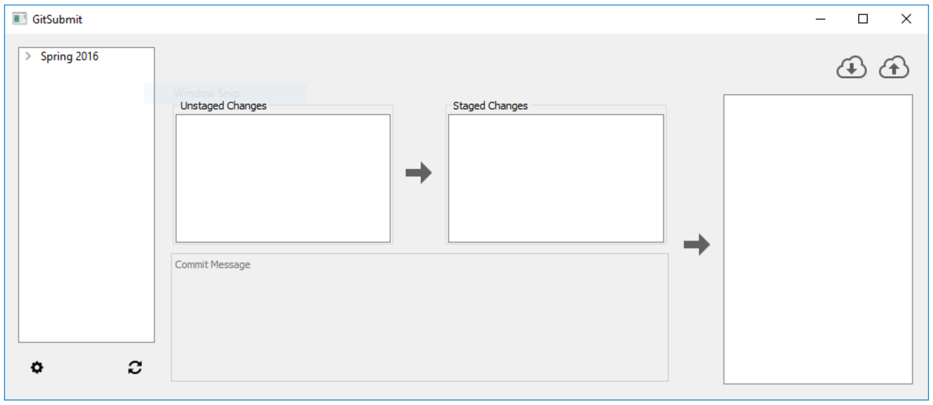
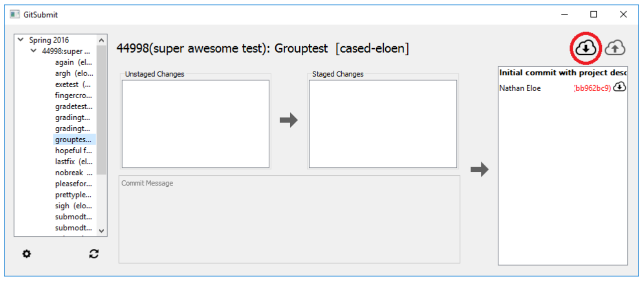


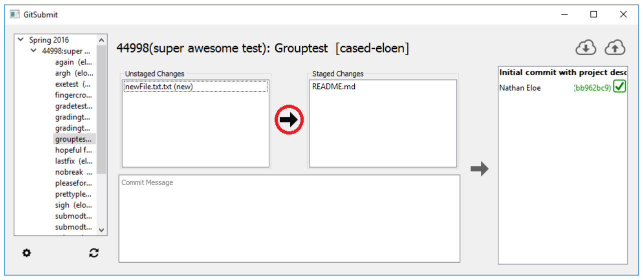
Figure 1. The Gitsubmit Main Window



44998(super awesome test): Grouptest [cased-eloen]

Initial commit with project desc
Nathan Eloe (bb962bc9)

Figure 2. Assignment Selection in Gitsubmit



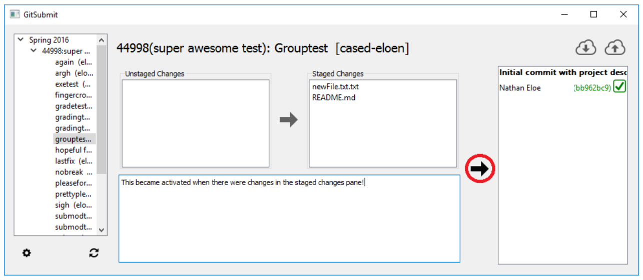
44998(super awesome test): Grouptest [cased-eloen]

newFile.txt.txt (new)

README.md

Initial commit with project desc
Nathan Eloe (bb962bc9)

Figure 3. Adding files to the Submission in Gitsubmit



44998(super awesome test): Grouptest [cased-eloen]

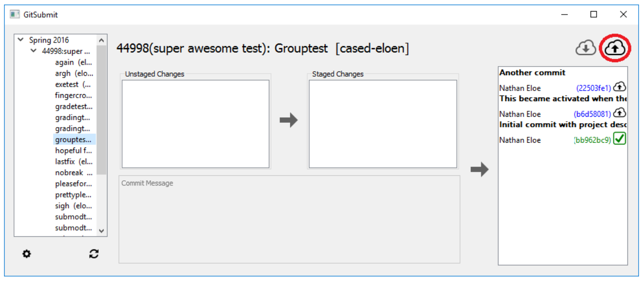
newFile.txt.txt
README.md

This became activated when there were changes in the staged changes pane!

Initial commit with project desc
Nathan Eloe (bb962bc9)

Figure 4. Making a Commit in Gitsubmit

commit the student wants to have submitted (as in Figure 6).



44998(super awesome test): Grouptest [cased-eloen]

Another commit
Nathan Eloe (22503fe1)
This became activated when the
Nathan Eloe (b6d58081)
Initial commit with project desc
Nathan Eloe (bb962bc9)

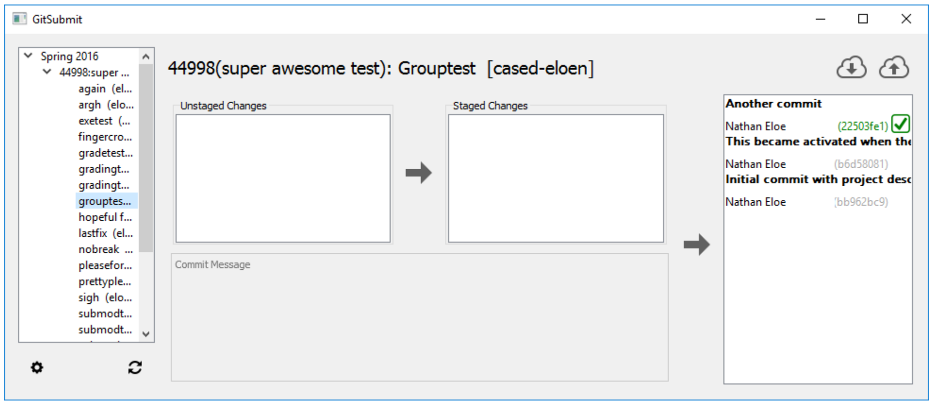
Figure 5. All Commits Ready to Push in Gitsubmit

3.4 Usability Features

Gitsubmit contains some embedded features to help guide the students through the submission process, as well as some visual cues that help form a connection between the submission process and the version control process.

As a student progresses through an assignment submission, options that cannot be performed are disabled; for example, if no files have been added to the staging area or a commit message has not been provided, the button to make a commit is disabled. Actions are only made available to the student when all prerequisites for the step in the submission process have been satisfied. Additionally if there is an indication that the student has not completed the submission process when he or she tries to exit the program (files have not been added to the staging area, or changes have not been committed, or commits have not been pushed), the interface verifies this is intended before closing.

One major stated goal of Gitsubmit is to simplify the introduction to Git and remove steps from the process that are



44998(super awesome test): Grouptest [cased-eloen]

Another commit
Nathan Eloe (22503fe1) ✓
This became activated when the
Nathan Eloe (b6d58081)
Initial commit with project desc
Nathan Eloe (bb962bc9)

Figure 6. Submission Pushed to Gitlab in Gitsubmit

not related to the Version Control workflow. To this end, all interactions that are necessary but auxiliary to the Version Control process are abstracted away. This includes authentication with the central hosting service. When Gitsubmit is first run, the student provides their Gitlab username and password in a first run configuration window (not shown in this paper); this is the only time a student is required to interact with the authentication system. Gitsubmit uses the student provided username and password to obtain an API token from the Gitlab server and generate a SSH key that is used to authenticate and encrypt all Git traffic. This removes the burden of managing authentication methods from the students without requiring that they learn how to generate an SSH key, or provide a password on every push or pull. While authentication is an important part of securing the workflow, it is an external mechanism that is not necessary to understanding the basics of version control.

Tool tips provide both usability enhancement and subtle introduction to the Git verbs. In the assignment select panel, tool tips show additional information (the full name of the project, for example) to help the student determine which project should be chosen. This is particularly helpful when there are multiple similarly named projects that might only differ by the group members (but is not noticeable in the project name itself). The tool tips for the VCS action buttons are the git verbs; in this way, the student begins to make connections between the verbs and the actions those verbs represent in the Version Control process.

4 Usability Evaluation

The survey used to evaluate the usability of Gitsubmit was designed to target the six cognitive dimensions that could be analyzed by a novice in the practice of Version Control. Individuals using the system were asked to evaluate these cognitive dimensions in much the same way one would ask a novice user to perform a Cognitive Walkthrough. Two statements targeted each of the six identified dimensions; one statement approached the dimension from a positive perspective (the UI does X well), while the second looked at it in a negative way (the UI does not do X well). For example, for the dimension of Error-Proneness, the survey gives the following statements:

- The UI makes it easy to make a mistake in the submission process.
- The UI makes it difficult to make a mistake in the submission process.

The exception to this are the statements focusing on Difuseness/Terseness, which asks the user to evaluate two negative statements:

- The UI is not expressive enough to complete a submission.
- The UI is cluttered or complex.

All responses are in the form of a five point Likert Scale, with scores of 1, 2, 3, 4, and 5 corresponding to Strongly Disagree, Disagree, Neutral, Agree, and Strongly Agree, respectively. To determine the UI's overall score for a specific dimension, the scores for the negative and positive statements need to be comparable; as such, the scores for a negative statement are converted to a positive score by determining the distance of the average score from 1 (Strongly Disagree) and taking the score the same distance from 5 (Strongly Agree). This becomes a simple equation:

$$\begin{aligned} adjustedScore &= 5 - (negativeScore - 1) \\ &= 6 - negativeScore \end{aligned}$$

The survey was distributed to sections of classes that have used or are currently using Gitsubmit in their coursework. This includes two sections of a Sophomore level Data Structures class, a Senior level Operating Systems class, and a Junior/Senior level Algorithms class. There was some overlap in students between classes. Of the survey invitations sent out, 34 responses were elicited.

Some of the respondents have experience with Git in other courses (such as a Software Engineering Course) or in industry (through an internship or other professional experience). A portion of the survey asks these students to evaluate Gitsubmit (and VeCVL) on both the Abstract Gradient and the Closeness of Mapping. The statements posed to these more experienced respondents include:

- The UI exposes too many git operations to complete a submission.
- The UI doesn't expose enough git operations to complete a submission.
- The UI abstracts away too many of the git operations.
- The UI should combine more operations into abstractions.
- The UI uses too many symbols to indicate a git operation.
- The UI doesn't use enough symbols to indicate a git operation.

For all of these statements, an average score of less than 3 is a positive indicator.

The full survey can be accessed at `https://www.surveymonkey.com/r/922NT9X`.

5 Results

Figure 7 shows the average score for each statement in the survey aimed at all respondents, as well as the aggregate overall score for each Dimension. For statements posed as a negative, a score below 3.0 indicates that on average students disagree that with the negative statement, and is a desirable score. For positively posed statements, a score above 3.0 shows that Gitsubmit is on average doing well in that category. The reported overall score is an average of the scores of the statements for the given Cognitive Dimension (adjusted in the case of negative statements).

Table 1 shows the average results for the questions directed at students with Git experience. In all cases, the statements were negative, so an average less than 3 reflects well on Gitsubmit and VeCVL. The number of responses (that were not N/A or prefer not to answer) varies from question to question based on student understanding of version control and Git.

Table 1. Average Results for Non-novice Statements. All statements reflected negatively; average scores less than 3 is desirable.

Metric	Avg. Score	Responses
Too Few Operations	2.19	26
Too Many Operations	2.04	26
Too Much Abstraction	2.24	25
Too Little Abstraction	2.71	24
Too Many Symbols	2.08	26
Too Few Symbols	2.31	26

6 Conclusions

The results in Figure 7 show that Gitsubmit as an implementation of VeCVL performs well when analyzed by these eight Cognitive Dimensions of Notations. Overall, students of varying experience levels indicated that Gitsubmit's strongest areas were Diffuseness/Terseness and Progressive Evaluation. This suggests that students find Gitsubmit's interface to be simple enough to use, but provide sufficient functionality to submit the assignment. Students also like that it is able to show them the status of their submission.

The weakest area of those examined is Error-Proneness. Responses were on average slightly positive; this indicates that this is an area where Gitsubmit can improve. Further exploration of the kinds of errors that students are encountering will be needed to determine whether the failings are in the UI, in VeCVL, or both. From an instructor and grader perspective, there have been fewer instances of students submitting the wrong file (or corrupted files) since moving to using Gitsubmit in these classes.

The results in Table 1 are overwhelmingly positive towards both Gitsubmit and VeCVL. The survey indicates that the "Goldilocks Zone" has been reached in terms of number of operations and symbols needed to complete the task. The weakest area for GitSubmit is the amount of abstraction; while students on average agree that there is neither too much or too little abstraction, the results for too little abstraction are closer to neutral than outright disagreement. This indicates an area where Gitsubmit could improve in its usability.

7 Future Work

Gitsubmit and VeCVL are continually evolving works; every course they are used in give feedback and a chance for refinement and improvement. These results show that one area that Gitsubmit could improve in is how well it prevents users from making mistakes. Of particular interest

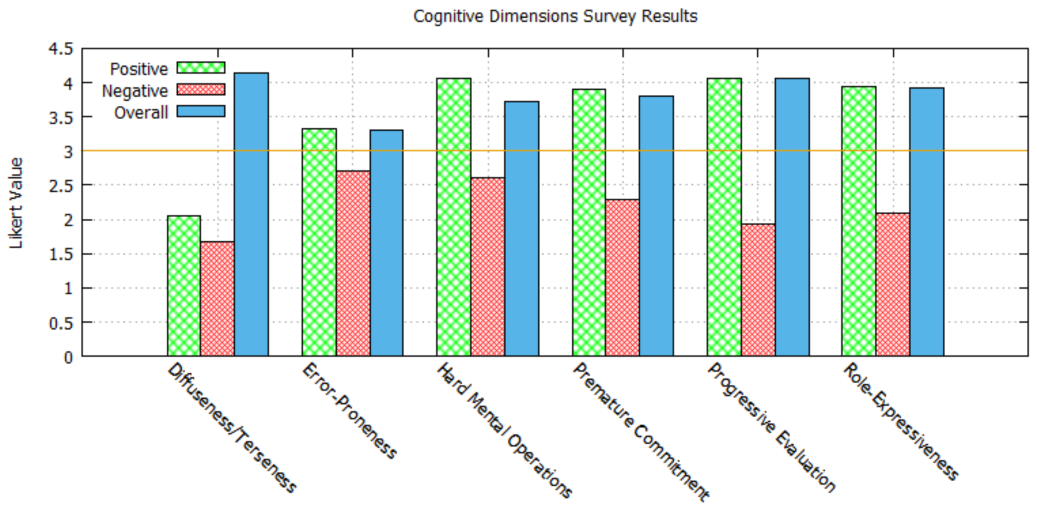


Figure 7. Average Novice Statement Results from Survey. Note that both statements for Diffuseness/Terseness were negatively framed questions; an average score of less than 3 is desirable for both, and both scores were adjusted to determine the Overall score.

is the kind of errors users are making; the notation needs to prevent users from making errors related to submitting the correct files. Most errors that instructors and graders encounter students making relate to the student closing the UI in the middle of a git operation (such as clone or pull), which puts the UI in a state it cannot easily recover from. If students are making other kinds of errors, it needs to be determined whether the notation (VeCVL) or the UI (Gitsubmit) needs to be modified to solve the errors that students are encountering.

Further analysis of the UI is in progress, both through surveys of users and analyses with Human/Computer Interaction tools. A Cognitive Walkthrough analysis is ongoing, and further directed research will investigate the error-proneness of the UI.

Additionally, work is progressing to make the UI look more attractive to users and easier to deploy. Gitsubmit is currently implemented using Python and QT; while development is quick, deployment to multiple platforms (specifically OSX and Windows) is difficult. Updating the UI on student computers is also difficult. Additionally, the UI is not optimal when it comes to interface real estate or visual appeal.

References

[1] D. M. Case, N. W. Eloe, and J. L. Leopold. Scaffolding Version Control into the Computer Science Curriculum. In *Proceedings of the 2016 International Workshop on Distance Education Technology (in conjunction with the 22nd International Conference on Distributed Multimedia Systems (DMS'16))*, 2016.

[2] S. Chacon. *Pro Git*. Apress, Berkely, CA, USA, 2nd edition, 2014.

[3] V. Driessen. A successful Git branching model, 5 Jan. 2010. http://nvie.com/posts/a-successful-git-branching-model.

[4] N. W. Eloe, D. M. Case, and J. L. Leopold. VeCVL: A Visual Language for Version Control. In *Proceedings of the 2016 International Workshop on Visual Languages and Computing (in conjunction with the 22nd International Conference on Distributed Multimedia Systems (DMS'16))*, 2016.

[5] R. Francese, C. Gravino, M. Risi, G. Scanniello, and G. Tortora. On the Experience of Using Git-hub in the Context of an Academic Course for the Development of Apps for Smart Devices. In *Proceedings of the 21st International Conference on Distributed Multimedia Systems (DMS'15)*, pages 292–299, 2015.

[6] GitHub. GitHub Classroom. https://classroom.github.com.

[7] GitHub. GitHub Education. https://education.github.com/.

[8] GitHub. Understanding the GitHub Flow, 12 Dec. 2013. https://guides.github.com/introduction/flow/.

[9] GitLab. Code, test, and deploy together with GitLab open source git repo management software. https://about.gitlab.com.

[10] T. R. Green. Cognitive dimensions of notations. *People and Computers V*, pages 443–460, 1989.

[11] J. Lawrance, S. Jung, and C. Wiseman. Git on the cloud in the classroom. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 639–644. ACM, 2013.