Making *Git* Closer to Human Being

[A Problem Statement of Git for Beginners]

Yihuan Dong
North Carolina State University
ydong2@ncsu.edu

Niki Gitinabard
North Carolina State University
ngitina@ncsu.edu

Linting Xue
North Carolina State University
lxue3@ncsu.edu

Rui Zhi
North Carolina State University
rzhi@ncsu.edu

ABSTRACT
Git is one of the most popular source code management systems for software development. Besides its usage for keeping track of changes, it is used in many companies and courses to manage the group work because of its support for distributed, non-linear workflow. Git has many useful features like branches that can help make different people working on the same code easier. But as recent research and our survey show, it is difficult for people to learn. Thus, many users keep their usage of Git limited to some simple tasks. There are a few Graphical User Interfaces (GUI) for it that are easier to use, but as the results of our survey show, they are not really popular even among novice users. Also performing more complicated tasks using the GUI isn't easy. Another problem with GUI is that beginners won't end up learning Git commands after using them for a while.

Keywords
Git, Novice User, Usability, Version Control System

1. INTRODUCTION
Version control systems are widely used to keep track of changes in software projects. Using their change-logs, users always have a backup of everything they have done. If there is something wrong at a stage, they can always roll back to any previous stable version. Or if they want to check something in previous versions, they can just go back and take a look.

Git is one of the most popular version control systems. GitHub, a web-based service that hosts software development projects that use Git for version control, has over 3 million users and over 7 million repositories. Git has a good support for distributed work and non-linear workflows. As a result, it is also helpful for people who are working in teams. The non-linear workflow support and branches help several people changing the same lines of code simultaneously without losing any of their work. After they are done, they can decide how to merge different features together. They can always check and see who is responsible for each line of the code, so if something isn't working or someone doesn't know how it works, they can know whom to ask.

However, Git is not novice-friendly. Beginners need to spend a lot of time learning it before benefit from it. The steep learning curve for Git scares away many novice users. For many beginners, they only use the simplest commands in Git and haven't experienced other features of this powerful version control system. Given this situation, many tools were built with the aim of helping novices get familiar with Git features and encourage them to use it. We summarized these tools into three categories and will talk about these tools in detail in Literature Review section. All these tools have drawbacks in some ways. Some of them in fact, on the contrary of helping users to learn Git knowledge, keep users away from using Git commands by offering them with an "easier" and "friendlier" interface. This could be detrimental to users because they will never know the concepts of Git and may never be able to solve complex problems in large group projects.

We propose to build a tool that helps novice user to learn Git commands and understand Git mechanism in an easier and faster way. For example, imagine a person who just get into a software company, the person is required to learn Git as fast as possible to be able to work in a team project. Based on this scenario, we define our target user ("U") as the novice who is willing to learn Git; the problem ("P") is that the person needs to learn how to use git commands, more specifically, how to resolve a conflict using git commands in a short time; the goal ("G") is to help novices learn Git (resolve conflict with Git in this case) eventually.

In this paper, we will first present some literature review on Git usability and learning curve issues. We will also talk about some of the existing tools that attempt to address these issues and the weakness of the tools. Next, we'll present our survey on 51 Git users to show that Git indeed is hard for the novice user to learn and, more specifically, which commands they have problems with. The last part is a summary of our findings from literature reviews and user survey.

2. LITERATURE REVIEW

The literature review section is divided into two subsections. The first subsection is to present the problems that novice users face when learning Git from past research. The second section exhibits some existing tools to mitigate the difficulties of learning Git.

2.1 Problem Statements from Past Research

Users can benefit a lot from Git once they get familiar with it. However, the steep learning curve and complex functions may impede novice user from using Git. We did some literature review to confirm that Git is not friendly to beginners.

Several studies show the problems novice users may face and the steep learning curve of Git. In Feliciano's thesis[6], when mentioned the barriers to entry GitHub, he pointed out that people feel challenging to solve merges and conflicts. Novice users and those who have limited technical background have difficulties in using Git. From Feliciano's research, one of his interviewees mentioned that the biggest challenge for Git is to lower its learning curve and make it more friendly to novice users. Similarly, Xu et al.[14] explores the effect of Git being used in an undergraduate Computer Science capstone class. The author points out that Git's learning curve is steep and students may need time to get familiar with a VCS. Students can benefit a lot once they get familiar with Git. Jansson et al.[8] did a study on investigating the effect of Git in an XP project. The author points out that beginners need much time to learn Git before they feel comfortable with using it.

For the challenges and problems students face when learning Git[7], Isomöttönen et al. showed an important work. They took surveys which are related challenges when learning Git and got answers from 21 students. The result shows that many students have difficulties dealing with conflicts when they work on the same part of a software. Besides, Git commands and Bash commands are easily confused for those who has little command line experience. Another problem is that some of them may treat branches as folders for files. Learners may think 'git add file' is creating a new file as well. What's more, sometimes VIM editor can impede students when they commit a message being interrupted in VIM. Besides, Cochez et al.[5] analyzed student's Git commit log data in several computing courses. The result shows that students often write trivial or nonsense commit messages.

The main reason for the steep learning curve of Git is its complex conceptual model, De Rosso et al.[10] pointed out that Git is not easy to use even for experienced developers. Users face a lot of difficulties because of Git's complicated conceptual model. In current Git's conceptual model, users need to understand different states of a file including working, staged, committed and stashed. For most users, the staging concept is unimportant and complicated. Another study also mentioned the complex conceptual model of Git indirectly. Jansson et al.[8] found that Git's staging functionality is quite confusing for beginners. Besides, branching is also a problem for the novice users. They are not sure when to merge, what branch they were working on, how to solve a merge conflict, etc. Because of those problems, they were discouraged from using branch feature. The author also points out that students have confusions on how to use correct command line parameters when pushing changes to a remote branch on Github. De Rosso et al.[10] proposed a new simpler conceptual model of Git and built Gitless based on the model. They tried to simplify the Git, as Bruce Tognazzini's said[11], "The great efficiency breakthroughs in software are to be found in the fundamental architecture of the system, not in the surface design of the interface."

Git has many features and it's difficult for beginners to remember related commands. Lawrance et al[9] presented in their work about how to adopt version control for non-CS engineering majors in a CS1 course. From the authors' experience, almost all the non-CS engineering students can learn enough Git commands to submit their code on time, but they usually need reminders about how to use Git. The paper also demonstrated that novice users can easily adopt Git's workflow, although they usually get confused.

Based on the problems presented in the papers mentioned above, we put the common problems novice user might encounter in table 1.

From table 1 we can see that the main problem Git has for novice user is its steep learning curve. According to the study of De Rosso et al.[10], the conceptual model of Git increases the difficulties of learning Git.

Table 1: Git Usability Problems for Novice User

Problems Novice Users Face	Study
Learning Curve is Steep	[5], [6], [8], [14]
Conceptual Model is Complex	[8], [10]
Has Difficulties in Solving Conflicts	[5], [6]
Need Remainders about How to Use Git	[9]

We also conducted a survey about Git usage in this study, we will talk about the survey and present the results in Data Collection section.

2.2 Existing Tools

Given the situation that Git commands are not easy to learn and might frustrate novice at the beginning, researchers have come up with several solutions try to improve the situation. From the literature review, we found that alternative solutions can be narrowed down into the following three respects. First is to teach Git features in class as a course platform. Second is to build a Git learning game to provide a mimic Git environment to allow users to practice Git commands, language embedded in the game are the same with Git commands. Third, a wrapper around Git command or a GUI to provide an easy interface for users. We will discuss the potential drawbacks of these solutions as below.

In 2015, Haaranen[12] proposed a novel way to teach students Git in class by incorporating the Git features into the course work flow. Students are required to cooperate with classmates by specific operations such as merge and fetch which are predefined by instructors. These operations are the same concept with Git features. Thus by the students following the project work flow to process the course projects using namely "Git" operations. Students will eventually get the idea of how Git works. However, it has just been proposed and haven't yet been used widely. What's more, this teaching method is solely focusing on introducing the feature of Git rather than Git commands. In reality, the greatest obstacle for novices to learn Git is that they may encounter numeric unpredicted problems including the coding problem and they don't know how to solve it.

The second solution is to learn Git by interacting with a learning game. Githug[13] is such a programming game tool. It helps users to improve their Git skills gradually by interacting with designed games with different level in a mimic Git environment. There are more than 50 levels in Githug, that start off with an easy task and go deeper every level. The disadvantage about this tutoring game is that it doesn't provide feedback to the users as tutors, which means when users encounter a technical problem, it doesn't offer any hint or suggestion to users to solve problem. Instead, users have to ask for help on the website and search the solution online. In this case, Githug seems to be a redundant work. Because users could just build a example project on Github and interact with online repository using real Git commands to practice and learn.

In order to make Git easily to use, more and more tools have been developed to provide a friendly interface with Github without typing Git commands. After we did surveys, we find two kind of easy interface around Git. First one is GUI app to access GitHub. GitHub Desktop[1] is a representative tool that has been widely used by most users in recent years. Two different desktop apps for GitHub are invented respectively for Windows and OSX operating system with very friendly interface. After users setup an account within the app, it will connect repositories on GitHub with locally projects automatically. Users can interact with GitHub by simply click the button to accomplish tasks instead of typing Git commands. However, the weakness of this tool with similar features is that it doesn't help users who want to learn Git commands and prefer to using command lines ,specifically Linux users. Most important thing is that this tool will keep users away from learning git commands, but this is an ultimate goal of our research.

The second easy interface tool we discovered is a wrapper for Git[2][3]. [3] is a PIP wrapper that allows PHP code to interact with online Git repositories from within an application. It's a PHP library contains function of a Git repository abstraction that can be used to access Git repositories and of a stream wrapper that can be hooked into PHP stream infrastructure to allow the users to access file and directory in a Git repository. It offers many other functions to access the status information on a Git repository, such as log, status and etc. Given the special design, it's apparent that this wrapper is for the group of people who are familiar with PHP coding schema, so they can interact with GitHub using this tool. Thus, this wrapper is not built for broad users. Another PHP wrapper around the Git commands line tool[2], which is very similar with[3]. Both of these tools are developed for PHP users and not for novices to learn Git commands and to help them solve Git conflict problem.

3. DATA COLLECTION

3.1 Survey Design

Besides literature review, we also conducted a user survey to identify user problems with Git. The survey[4] contains a total number of 14 questions, with 2 questions asking about user proficiency with Git, 1 question asking about user's purpose to learn Git, 11 questions to identify the problems users were facing with Git.

The first two questions asked about how long the users have been using Git and how proficient the users consider themselves are with Git. These questions were designed to help us distinguish novice users from expert users. At first, we only asked about how long have the user used Git. However, we realized that it is possible for one to master Git in a short period under certain pressure (company requirement), or still being inexperienced after a long time. Thus, we decided to add another question that asks user themselves to rate their proficiency with Git to get a more reliable grouping of novice users. The users can choose from 5 options where the proficiency level increases from "Know nothing about Git" to "I'm an expert." We finally decided to identify users who rate their proficiency level on or below the midpoint, "Know simple commands, but still sometimes need help", as our target novice users.

The third question focused on students' purpose to learn Git. People are introduced to Git for different reasons. Some users use Git to submit their homework assignments, some use it due to company requirement, others learn it just as a way to manage their personal work. We conduct this question in order to collect the information that why people would like to use Git.
Question 4 to 14 are designed to find the problems that user have when learning and using Git. For question 4 and 14, we put them in the survey because we'd like to investigate their overall feelings and whether they mess up with their repository while using Git. The answer to these two questions will provide us with evidence if Git has usability problem and steep learning curve among novice users. Question 5 asks if users prefer command line or GUI user interface when using Git. This question helps us identify which kind of interface that we should focus more on. In question 7, we ask users to check the commands that they feel they can use appropriately from a list of most common Git commands, which includes pull, push, commit, add, status, etc. Question 6, 8-13 all ask about students opinion and previous experience with Git. The data from the answer to this question will give us the general information about how well they use Git and which feature they don't use very often.

3.2 Survey Process

We made both paper-based survey and online survey forms. The participants of our survey are all from Computer Science department. Paper-based surveys were handed out in graduate level Software Engineering class and online surveys were presented to Computer Science major friends and colleagues. It took around 3 minutes for participants to fill out the survey. The survey was conducted anonymously.

3.3 Survey Result and Analysis

We collected 51 responses in total from both online and paper-based questionnaires. After analysing the survey result, we decided to use the response for question "How proficient do you think you are with Git" to identify different kinds of users, with users rated their proficiency above "Know simple commands, but still sometimes need help" as expert user, users rated their proficiency as "know simple commands, but still sometimes need help" and below as novice user. We did not use the time user has been using Git to classify expert and novice user. Because the data suggest that there is no strict relationship between being an expert user and the amount of time using Git. As a matter of fact, many users become experts due to company requirements (36.7% difference). We removed 2 responses from our data set because one of them claimed himself as expert user but only knew 4 commands and the other claimed himself only know simple commands but also claimed that he can use all the commands we listed(16) aptly, thus the responses were considered unreliable.

Of the remaining 49 responses, 15 are labeled as expert user responses and 34 are labeled as novice user responses. We drew pie chart to see how many people checked each option for each question and compared the pie chart between expert user group and novice user group to see the difference. Here are our findings.

In response to the reason why the users learn Git, 73.3% of expert users learnt Git to either manage their own work or because of company requirement, while 68% of novice users use Git to submit course projects. This is most likely because people gain more experience while dealing with complex situations in company and managing their own projects than merely completing the simple task to submit course projects.

33.3% of expert users agree or strongly agree on that Git was difficult for them to learn while 63.7% of novice users have the same opinion. There are chances that experts forget how hard it was for them to learn Git at first since 40% of expert users answered not agree or disagree with this question. But the 63.7% of novice users still means that Git is hard for more than half of the users to learn. Among the expert users, 80% admit that they have messed up stuff trying to learn new command and 69.3% of the novice users also suffered from the same issue. This further confirmed that Git has usability problems. Specifically, it has steep learning curve and is probably difficult for user to recover from mistakes.

One interesting result is that all the expert users and almost half (45.5%) of novice users like Git command line better than GUI. Even within the novice user who admitted Git is hard for them to learn, there is still half of them prefer Git command line more. This result contradicts with our common sense that most of the novice computer users like GUI more than command line. We hypothesize that this is because all of the participants in the survey are programmers who are familiar with command line interface and know its strengths. Another reason might be the fact that current Git GUI is hard to use or to perform complex tasks.

After analyzing the data we get, from question "which Git commands do you feel you can use appropriately", we found that novice user typically knows a lot less commands than expert users. From the bar chart (see Figure 1), we can see that besides push and commit commands, which are well known by both expert and novice user, many commonly known commands (know by 80% or more) among expert users like pull, add, status, checkout, branch, merge and init, are known by fewer than 50% of novice user. This result will help us identify the commands novice user needs most help with if we are to help them with learning Git command line commands.

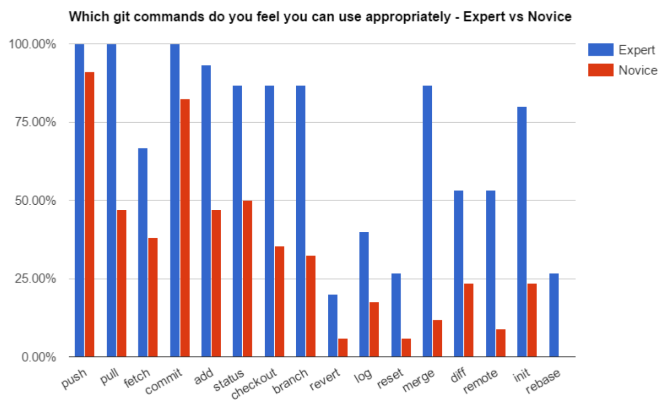

Figure 1: Git Commands Density Distribution between Expert User and Novice User

There is also another interesting result in our data which was pointed in previous works, too. Isomöttönen, Ville and Cochez and Michael[7] stated that many of their observed students used to clone the repository again instead of merging a code which has conflicts. Our data show the same thing. Among all users, 42% of them have given up on resolving conflicts and re-cloned the repository at least once. This is interesting for us, since as stated in the introduction, one of the most useful features of git is that it helps people work in groups and edit the same sources and they don't need to give up their work.

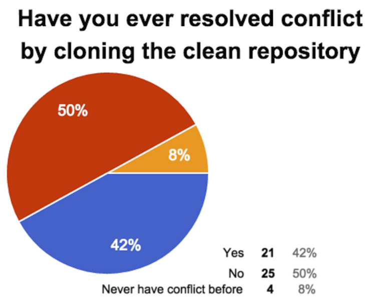

Figure 2: Resolve Conflict by Cloning the Clean Repository Distribution

4. CONCLUSION

Literature review and the survey lead us to conclude that Git is not easy to use for novice users. First most import reason is that learning curve of Git is steep according to our literature review. Second, novice users are not familiar with Git mechanism. Without knowing the Git features, it's relatively difficult for users to use it in an appropriate way, especially performing some complicated tasks such as using branches and solving conflicts. It might be the case that when they face conflicts, they don't even understand the reason for this problem, which leaves the user in a bad situation. Our survey on the hand confirmed this fact. Thus the goal of our work is to develop a tool to help the user to understand the Git mechanism by offering help solving conflicts.

Believe it or not, the fact that git is hard to learn is a issue that needs to be addressed.

5. REFERENCES

[1] `https://desktop.github.com/`.

[2] `https://github.com/cpliakas/git-wrapper`.

[3] `https://github.com/teqneers/PHP-Stream-Wrapper-for-Git`.

[4] `https://goo.gl/w6OHIO`.

[5] M. Cochez, V. Isomöttönen, V. Tirronen, and J. Itkonen. How do computer science students use distributed version control systems? In *Information and Communication Technologies in Education, Research, and Industrial Applications*, pages 210–228. Springer, 2013.

[6] J. Feliciano. *Towards a Collaborative Learning Platform: The Use of GitHub in Computer Science and Software Engineering Courses*. PhD thesis, University of Victoria, 2015.

[7] V. Isomöttönen and M. Cochez. Challenges and confusions in learning version control with git. In *Information and Communication Technologies in Education, Research, and Industrial Applications*, pages 178–193. Springer, 2014.

[8] E. Jansson. An investigation of git in an xp project. 2013.

[9] 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.

[10] S. Perez De Rosso and D. 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.

[11] B. Tognazzini. First principles of interaction design. *Interaction design solutions for the real world, AskTog*, 2003.

[12] B. Tognazzini. Teaching git on the side: Version control system as a course platform. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, 2015.

[13] M. Wnelson. `http://www.michaelwnelson.com/2013/12/15/githug/`.

[14] Z. Xu. Using git to manage capstone software projects. In *ICCGI 2012, The Seventh International Multi-Conference on Computing in the Global Information Technology*, pages 159–164, 2012.