Bachelor Thesis
Computer Science

08 2015

Git - CLI or GUI

Which is most widely used and why?

Robin Olofsson
Sebastian Hultstrand

Department of Software Engineering
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

This thesis is submitted to the Department of Computer Science & Engineering at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Bachelors of Science in Computer Science. The thesis is equivalent to 20 weeks of part-time studies.

Contact Information:
Author(s):
Robin Olofsson
E-mail: Olofsson_@live.se
Sebastian Hultstrand
E-mail: Sebastian@storm-net.se

University advisor:
Assistant Prof. Conny Johansson
Department of Software Engineering

Department of Software Engineering
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se/didd
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Many of us have encountered the discussion about which interface is better for working with Git, command-line or graphical. This thesis is an attempt to find out which user interface new Git users prefer for Git and what experienced Git users prefer.
We aimed to find out if there's anything significant which can be gained from using either of the interfaces in comparison to each other. Lastly we looked at what factors influences git users choice of user interface and how?.
We have collected data through three interviews and a survey, which yielded approximately 370 responses.
Based on our results we've found that the command-line interface is the more popular user interface, in general, amongst Git users.
We've also found that most users stop using graphical user interfaces, as their primary user interface, as they get more experience with Git. They usually change their primary user interface to a command-line interface or start using both the graphical user interface and the command-line interface together. The results from our study regarding why, is presented in this thesis.

Keywords: Git, Interfaces, VCS, New users, Experienced users

Contents

List of Figures

Chapter 1

Introduction

Git is a very popular version control system, along with version control systems like SVN and Mercurial. The discussions between using CLIs' (Command-line interfaces) or GUIs' (Graphical user interfaces), in general, have been going on since the late 70's and early 80's when operating systems like the Lisa and Macintosh among others came out [16] and the GUI revolution was kicked into high gear. To date people seem to prefer the CLI over GUI when working with Git based on our own experiences and the results from the GitSurvey in 2012 [1], however it is not clear as to why. Theories do exist which state that the CLI provides a faster and more controlled way of dealing with Git. [1] [4]
The reason this topic was thought of in the first place was due to an observation made in one of our courses. As of the day of writing there is no real formal introduction course to version control at Blekinge Institute of Technology, much less to Git. Instead there are a small number of courses where students are introduced to Git alongside the actual content of the course and most of the learning is left to what the student chooses to find out.
During one of these introductory talks about Git the students were encouraged to start out by using the command-line interface in order to get a feel for how Git works. This was met by a lot of complaining from the students as they would rather use a graphical user interface. The subsequent discussion that ensued was the spark that lit the idea to do this research.
In this paper we've taken a closer look at what people use today, in terms of interfaces to work with Git, and if there's any correlation to be found between how people use Git and which interface they use. The aim was to find out which interface is more preferred, which is more suitable to novice users and if developers can benefit more from using either of the two interfaces when reaching a certain knowledge-level of Git. Developers using or wishing to use git for this therefore the group which this paper is focused on.
To find these answers we have conducted a survey, through an online service (Google forms), that was sent out to developers within the industry of software manufacturing and students at Blekinge Institute of Technology. The distribution to the industry was done through, in part, people we know whom agreed to spread the link to co-workers and so on, and in part through online forums.

A literature study has also been carried out, mainly to help gather as much knowledge about current opinions within the scientific community about the CLI and GUI interfaces. Thanks to the clues found in the literature review we could develop questions that would provide us with relevant data.
Three interviews were conducted to get a deeper understanding of the answers that were gathered through the survey. We felt that a survey could possibly provide enough information to carry out the work but by conducting interviews there was a greater chance of understanding why the participants of the survey, answered the way that they did.

Chapter 2

Background

2.1 Git

In 2005 the Linux development community started the development of Git. Up until that point they had been using BitKeeper as their version control system. Back then BitKeeper was free but when the company behind the product decided to price it the Linux development community quickly abandoned the solution to develop their own open-source version control system. [10] To date, Github alone has over 21.7 million repositories. [9] All of these repositories are different little packages of code or text that people have created and are working on, with varying levels of frequency. Git is, today, one of the most common version control systems [11] [12] [13]. The reason for Git being a powerful tool is its flexibility in work flow choice, the storage is well designed and consistent, [26] and the surrounding functionality is adaptive and highly customizable. This leads to the learning curve being quite steep and a challenge to get over. But once you get over the curve you can very easily oversee and understand the functionality and structure of a Git repository and have little difficulty learning new methods and work flows. This at least seems to be the consensus on forums and message boards around the web, and it coincides quite accurately with our experiences as well.
The basic structure of a Git repository is; 1) a .git directory that houses the meta data and object database for your repository and all the committed changes. 2) A staging area where changes that are to be committed are kept, in waiting for a commit to take place. 3) A working directory where your plain, working copy, of the source code is held. This is where development is done. [19] So depending on what version of the code you checkout from the .git directory, it will appear in the working directory. Then you code your changes and stage them, thus placing the changes securely in the staging area. The final step is to commit what is in the staging area and it will be saved into the .git directory. Some repositories have a remote host that the changes can be pushed to for ease of sharing the code. These are the very basics of how Git works [19]

2.2 Command-line interface and the Graphical user interface

There are two main ways of interfacing with Git, one is through a command-line interface and the other is through one of many graphical user interfaces that are available. [20]

When using the command-line interface you type commands manually to preform the desired actions whilst in a graphical user interface you will have something visual to interact with, such as buttons, input fields and so on. There are pros and cons with using either of the two interfaces not only for Git but in general. For example, through a command-line you can rename 100 files with a lot less effort than with a graphical user interface. Though it would require you to know the commands needed to rename the files. One command can perform a lot of different tasks, whilst in a graphical user interface you would normally have to rename each file one by one which would require a lot more time but less knowledge. As mentioned in the introduction the debate of GUI v. CLI has been going on since the 80's and not a great deal has changed in the arguments, since they are so deeply ingrained in either interface. A good analogy is the car dealership comparison made in the article (Later a book) *In the beginning was the command-line* (Stephens, 1999) [21]. The summary of the analogy would be that there are a bunch of competing automotive dealerships on the same street where some are expensive, but easy to use and fairly cheap to repair however they often break down. Another sells sleek euro-style cars that no one really knows the inner workings of, they are expensive but beautiful. Then there is the super advanced tanks that are being given away for free. Yet still all the customers go for the cheap car that is sort of bad or the expensive euro-car. No one wants a tank even though they are free, because no one can be bothered to learn how to drive it. This analogy helps to illustrate the situation between the GUI and CLI very simply but quite accurately.

"A command line interface (CLI) enables users to type commands in a terminal or console window to interact with an operating system. Users respond to a visual prompt by typing a command on a specified line, and receive a response back from the system. Users type a command or series of commands for each task they want to perform." [24].
An example of such an interface for Git can be seen in Figure 2.1 below.

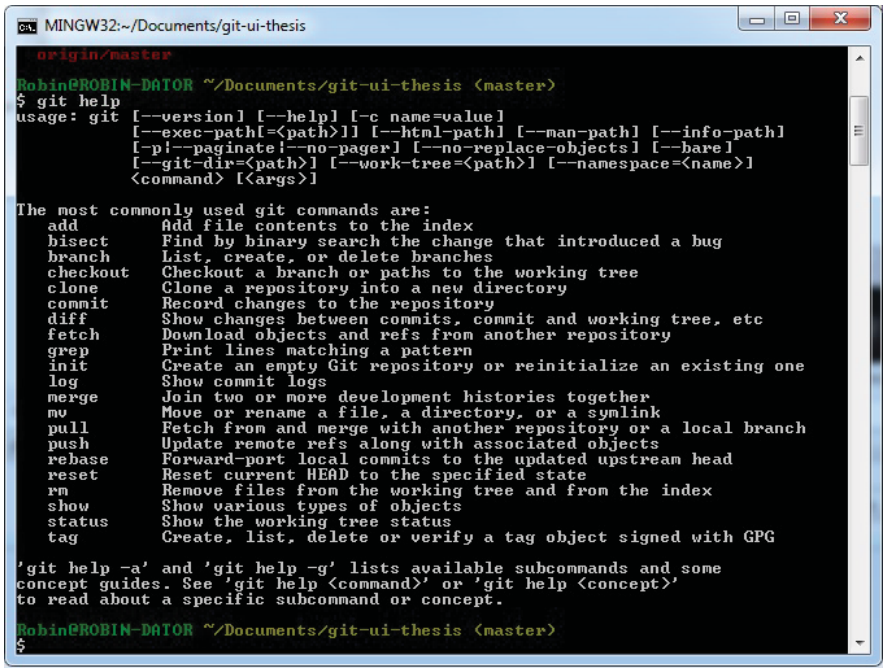


Figure 2.1: Git command-line interface, in Windows

Whilst - "A graphical user interface (GUI) uses graphics, along with a keyboard and a mouse, to provide an easy-to-use interface to a program. A GUI provides windows, pull-down menus, buttons, scrollbars, iconic images, wizards, other icons, and the mouse to enable users to interact with the operating system or application." [24].
An example of such an interface for Git can be seen in figure 2.2 below.

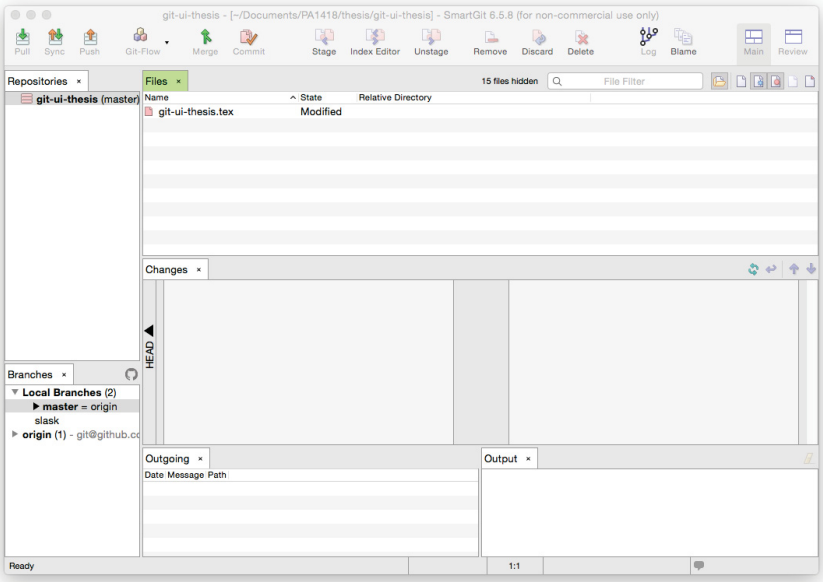


Figure 2.2: Git graphical user interface, Smart Git Hg on OSX

2.3 Terminology used

- Demographic - The word demographic is used in this thesis and refers to a certain group that answered our survey. The group's members are defined by their indicated level of proficiency with Git, and the amount of experience they have with Git. The determination of which demographic a respondent falls in is made up of a point system. The alternatives are given an arbitrary point value to signify rank and when the points for the chosen alternatives are added up we get the sum for the respondent. Low is defined as 2-4, mid 5-7 and high 8-9. We did this in order to make it possible to split the respondents into demographics.

- VCS (Version control system) - A VCS is a system that records changes to a file or a set of files over time so that you can recall specific versions later. There are three different types of version control systems; Local Version Control Systems, Centralized Version Control Systems and Distributed Version Control Systems.
  Since Git is a Distributed Version Control System and this thesis concerns Git - you can safely assume that we discuss about the Distributed Version Control System when we say Version Control System. [23]
  Version control systems usually consists of two applications, one on a server-side and one on a client-side. The basic functionality of a VCS is to maintain source code, other types of software files and their history. [15]

Initially, you either create a new project or import a project into the VCS. Once that's done you check out a version of the project into your private working directory and after that it's time to get to work and start coding. Every time you feel like you are content with a change you've made, such as a bug fix or the addition of a new feature, you commit your changes to the repository alongside with a explanatory message to let everyone know what's new/changed. Then, every once in a while, you update/synchronize your private version of the project with the changes your colleagues have committed. This action will allow all of your colleagues to have roughly the same source code base. From there on out you and your colleagues can battle against the mistakes that has been produced and once you feel confident enough on the software you can roll out a release. [22]

- UI - The word is an abbreviation of the term user interface and refers to either a graphical user interface, the command-line interface or both.
- CLI - Command-line interface.
- GUI - Graphical user interface.

Chapter 3

Research methodology

3.1 Research questions

1. What is most commonly used when getting started with Git, CLI or GUI?
2. What is most commonly used by experienced Git users, CLI or GUI?
3. What can be gained from using the GUI in comparison to the CLI, in the aspects of functionality and convenience?
4. What can be gained from using the CLI in comparison to the GUI, in the aspects of functionality and convenience?
5. What influences a git users choice of UI and how?

3.1.1 Reasoning for questions

We initially identified and used the above questions as the basis for our work because they work well as guidelines towards what we wanted to find out. Firstly we had to find out which UI new Git users use and how they got started with Git, in order to see what the reality is today. Secondly we had to look at what experienced users employ currently in terms of UIs, in order to see how the trend of swapping looks. This is something we could do through looking at the results from our survey and the Git survey from 2012 [1]. Thirdly we wanted to examine a few examples of the UIs available in order to see what could be gained, in terms of functionality and convenience. But since we lacked time and the knowledge, we couldn't do a clear objective analysis of the UIs. Instead we used some of the opinions and reflections about the UIs gained from the survey and the interviews to get an idea of what users have gain through using the UIs. Fourthly we wanted to examine all the results collected while researching the other questions in order to build an answer to the fifth question.
The reason that we thought this would be important is that there are so many different approaches to learning git and so many opinions that you have to get through before you can decide how you want to learn. These results may help shortening that period.

3.2 Approach for answering research questions

To answer the research questions a literature study was conducted to find out how Git works, workflows and discussions about the command-line interface and the graphical user interface in general. Based on what was found, the design of the survey and a number of questions for the interviews was created.
The intention of the survey was to collect information about current users preferences and perceptions. We wanted to know what they were currently using (either CLI or GUI tool), how long they have been using it. What their perception of the two different types of interface was. How they work with Git. How they began using it and how they learnt to use it. Using this information we attempted to identify correlations between their experience with Git, how they use Git and their preferences and reasoning behind their choice of UI.
The interviews were based on more open questions where the aim was to allow the subject to elaborate around why they have chosen to work like they do, and discuss the perks of the two different types of interfaces that we put forward. The aim of the exercise was to get a deeper understanding of the reasoning that goes into their choices. Which is an understanding that the survey would not be able to supply.
Studies have been conducted to try to find out what people use when interfacing with Git, but the most recent study to cover the area, that we could find was conducted in 2012 [1]. By conducting a survey, where the aim was to find out what people use today and why. There was the possibility of gathering fresh data as the data collected back in 2012 might not accurately represent how people interface with Git today.

In summary questions 1 and 2 were answered using the survey where questions pertaining to what UI the subject was currently using and what UI the subject began using for Git interfacing. Correlations was searched for in the 2012 GitSurvey [1] to see how the results compare to other studies. Questions 3 and 4 were answered in part through some of the perception related questions in the survey coupled with the interviews. Finally question 5 was somewhat answered by, in part, statistics from the survey by looking at questions pertaining to perception of the UIs, time consumption associated with the subjects use of Git, both general and problem based. And the subjects varying use of, and proficiency with Git. Also partly through the interviews where we were able to hear what drew the subjects to the UI that they use and what they believe to be important in the choice of a UI. Since we were able to determine the demographics based on the level of proficiency, from the survey we could use those to find and interview subjects which represented a certain demographic and discuss their needs and wants from the user interfaces. Our belief was that, by answering questions one to five, developers would get a general overview about how the present day Git users interface with Git and why. The "why" didn't have to be detailed for the

information to be valuable because it might be enough to know that, hypothetically, making a commit and pushing code through a graphical user interface is slower than doing the same in the command-line interface.

3.3 Literature review design

In our literature study we used BTH Summon, Engineering Village, IEEE Xplore and Google scholar as our primary search databases.

Keywords used: Git, configuration control, version control, public domain software, software engineering, linux, UI, User interface, Graphical user interface, GUI, CLI, Command line interface, Case study, User experience, Usability, UX, Interface.

To sort out what literature that was relevant we had to read the abstracts. Whenever we found literature that had a relevant abstract we would proceed to add it to a stack of literature that we believed to be relevant to our work. After this we would gradually go through these and determine if they held any value for our paper. This was determined by looking at if it had information about the following things: general GUI v. CLI discussion going on, Git in general, Git UI analysis work, ways of measuring usability in interfaces, statistics about Git or GUI v. CLI, and so on. After getting roughly 10 papers we started snowballing in the hope of finding more relevant papers. This was done until we hit a wall where we did not find any more interesting abstracts which in our case was roughly two iterations in.

3.4 Empirical study design

3.4.1 Survey

Goal with the survey

The survey consisted of 13 multiple choice questions, the questions were aimed to be as easy and quick to answer as possible in order to persuade people to answer the survey. What the survey was aimed at finding out was the following:

- What level of experience the subject has with Git.
- What UI they are currently using.
- How the subject learnt Git.
- What the subject primarily uses Git for.

- What operating system the subject uses.
- How much time the subject spend on average interacting with Git.
- How often the subject needs to consult another person, resource online or handbook to solve a issue related to Git.
- How the subject perceives the GUI and the CLI, by word association.

The questions we chose to include are the bare minimum we needed to try to achieve our goal with the survey. We tried to design them in a simplistic manner that should generate data concerning how the participants worked with Git. Just finding out how the respondents work with Git makes a huge impact as an interface might be more suitable for certain types tasks due to it's set of functionality and the way it operates. Experience was another factor we wanted to include as it could possibly influence the choice of interface.

Questions and alternatives

1. How long have you been working with Git?
   a) <6 months *b*) 6 months - 1 year *c*) 1-3 years *d*) 3-5 years *e*) 5-10 years
2. How would you rate your knowledge of Git?
   a) Novice *b*) Average *c*) Professional *d*) Master
3. What type of user interface are you primarily using for Git?
   a) Command-line interface *b*) Graphical user interface *c*) Both equally
4. How did you learn to use Git?
   a) Colleague or friend *b*) Open source community *c*) Online tutorials *d*) Book *e*) Course *f*) Self taught
5. What interface did you use when starting out with Git?
   a) Command-line interface *b*) Graphical user interface
6. Why do you use the interface you use today?
   a) Personal preference *b*) Standards for work environment *c*) Outside influences *d*) It is what i know best
7. What operating system do you primarily use?
   a) Windows *b*) Mac *c*) Linux
8. What do you mainly use Git for?
   a) Basic change tracking *b*) Collaborative work *c*) Large scale version control with branch management and code reviewing through Git

9. How many minutes per hour of development time, on average, do you spend on interfacing with Git?
   a) <5 min *b*) 5-10 min *c*) 10-15 min *d*) 15-20 min *e*) 20-25 min *f*) 25-30 min *g*) >30 min

10. How many times during a work day do you need to seek help for an issue with Git?
    a) 0 *b*) 1-2 *c*) 2-4 *d*) 4-6 *e*) 6-10 *f*) 10-15 *g*) 15-20 *h*) >20

11. What words do you associate with a Git GUI (Graphical user interface)?
    a) Aesthetic *b*) Easy *c*) Bulgy *d*) Auto-magic *e*) Difficult *f*) Helpful *g*) Simple *h*) Intuitive *i*) Insightful *j*) Hard *k*) Time consuming *l*) Cool *m*) Professional looking *n*) Control *o*) Ugly *p*) Old fashioned

12. What words do you associate with the Git CLI (Command-line interface)?
    a) Aesthetic *b*) Easy *c*) Bulgy *d*) Auto-magic *e*) Difficult *f*) Helpful *g*) Simple *h*) Intuitive *i*) Insightful *j*) Hard *k*) Time consuming *l*) Cool *m*) Professional looking *n*) Control *o*) Ugly *p*) Old fashioned

13. Which type of Git interface would you recommend others to use, no matter skill level?
    a) Command-line interface *b*) Graphical user interface *c*) Both

Crafting and distribution

The survey was made as a web form that we can easily collect the data from and most of all distribute easily. Distribution of the survey was done through a number of different channels such as BTH:s student mailing list, friends working with software development in Sweden whom spread the survey as well and finally 8 different sub-reddits (Forum boards at the popular web forum Reddit) associated with software development (Look in the acknowledgements for the exact names of the sub-reddits). These gave a very large portion of the replies to the survey as a few of the sub-reddits that we managed to find were specifically for Git users. This generated some very fortunate feedback and interest from that community. To get as many responses as possible the survey was designed to be easy and quick to fill out and to motivate participants even further we had a prize drawn amongst the participants for three, one month gift subscriptions to the curio service Loot Crate. [17]

Analysis of survey

The survey was crafted so that it can be evaluated and analysed in conjunction with the interviews. First a demographic for the replying party will be determined based on their answers to questions one and two. This demographic was used when looking at the questions about what user interface they use and have

used, their reasoning for using it and their perceptions of the user interfaces. The demographic was necessary to determine since the fifth research question is based on both experienced and inexperienced users.
We made cross tabulations between what OS the respondents use/how they learnt Git/what they mainly use Git for and what interface they are using now. A conscious decision was made to make correlation analyses between experience with Git and what interface the respondents use. This is something we had to do because the cross tabulations alone wasn't enough.
The cross tabulations and the correlation analyses can be found in Appendix B. Questions three and five, in the survey (check Appendix A Figure A.2a and A.3a), was used to determine the answer to the first research question and in conjunction with the determined demographic to answer the second research question.
The rest of the questions on the survey (Questions 4, 6 - 13) was used as factors for answering the third, fourth and fifth research questions. As they pertain to difficulties, perceptions and opinions about the user interfaces. When coupled with the results from the interviews. They would place weight on elements that are found, thus allowing the determination of what parts of a user interface for Git, that the different demographics value or are influenced by. Graphs were crafted in order to easily convey the messages that the collected data holds. For example bubble graphs to look at how the different demographics are responding to the word association questions to find relationships between ability and perception of the interface type.
The data that was collected through the survey was stored in a Google spreadsheet [18] and through that spreadsheet we parsed the resulting XML (Extensible Markup Language) into a MySQL database where querying the data to build datasets for graphs is a lot easier. Through the diagrams we could easily find out percentages of how many answered a question in a certain way and through that we could draw conclusions to help us in answering the research questions.

3.4.2 Interviews

The interviews were conducted with a small group of subjects that were representative of the different demographics that answered the survey. The way that we determined a demographic in this context was the level of proficiency with Git that the subject indicates they had and how long the subject had been using Git. These two aspects allowed analysis on a deeper level, of the needs and wants of different user groups. The minimum demographics to cover are new users of Git whom were learning to use Git at the time and experienced users whom had been using Git often and for a long time. The identifiers for these two groups are;

Group 1 - New users. These users have less than one year of experience and identify themselves as Novice or Amateur users.

Group 2 - Experienced users. These users have more than two years of experience

and identify themselves as Pro or Master users.

There are no clear guidelines that we've followed in the creation of these groups, it was based on deduction done by us. It seemed like a logical approach to assume that a user either has experience with Git or a user doesn't, which is how we categorized them into Group 1 and 2. A third group was also used that was a representative of the subjects falling between the criteria for group 1 and group 2.

Interview format

The format of the interviews was a discussion on the topic of this paper where the aim was to get the subjects to explain their standing on the issue of CLI v. GUI in Git interaction. Most of all we wanted them to elaborate around perceptions and experiences that they have. We did supply some information about the UI:s to allow the sessions to go into as much detail as possible. We conducted three interviews with subjects from both of the above mentioned groups and one with a user with an indicated level of knowledge to be in between the two groups. The sessions were recorded and transcribed later to minimize the loss of information that would occur if we only took notes during the interviews.
At the end we carried out a small quality analysis of the results.

Questions and topics to discuss

- What functionalities do you most often use?
- How do you perceive working with a Git GUI?
- How do you perceive working with Git CLI?
- Which, if any functionality do you miss in the interface you use today? In terms of ease of use, efficiency and comfort.

These questions were aimed at finding out more about what the subject needs and wants from the interface he or she works with. However the factors that needed to be taken into account when looking at this are how much the subjects know about the interfaces, what type of work they actually do in Git and if the work is collaborative or not. These factors were covered in either the responses to the survey or the interview itself and was taken into account in the discussion.

Chapter 4

Literature review

4.1 Command-line versus Graphical user interface

Based on what we found, people with experience regarding Git will in most, if not all cases prefer the command-line interface over a graphical user interface.
The reason behind this is however, not crystal clear. In the 2012 edition of GitSurvey [1], about 97% of the people who answered the question "What Git implementations do you use?", indicated the Git command-line whilst only 8% use some graphical user interface.

If we go back to a time, before Git. A time when the concept of graphical user interfaces were quite new, there was a study conducted on students to see whether they would prefer the command-line interface over a graphical user interface. The study was done at Waikato University, New Zealand, back in 1992 and involved a mix of Macintoshes and IBM PC compatibles [5].
A group of students were placed in front of these machines so that the staff were able to observe how they responded to interacting towards Macintoshes graphical interface and IBM PC compatibles command-line.
The theory was that the students, whom had no prior computer experience, would prefer the Macintosh because of the fact that a graphical user interface has a less harsh learning curve than a command-line.This theory was backed up by Morgan, et al [2] and Kirkpatrick et al [3].
Based on this study and another more Git-focused study that was conducted in 2014 [4], the theory seems to be that graphical user interfaces would appeal to novice users more.
However theoretically, as users get more and more experienced they will gradually start to prefer the command-line interface over a graphical user interface because they are able to complete their tasks faster and to maintain more control over what they are doing and what's going on, in general.

Generally speaking it would seem that a user tends to use what they find to be easier [5]. Because graphical user interfaces do have a more lenient learning curve it would therefore appear logical that a user would prefer that over a

command-line interface. In terms of Git, graphical user interfaces are likely more valuable when the knowledge of the system matures for the user [4]. Git is complex, if you check the answers for the question "What do you hate about Git?" in the 2012 edition of GitSurvey you will notice that this is one of the answers. Another answer to this question is "requires steep learning curve for newbies". This is what spurred the development of graphical user interfaces for Git, the goal was to hide some of Git:s complexity. [6]

According to Eric S. Raymond [7], the disadvantage of the command-line interface is that it almost always has high mnemonic load (low ease), and usually has low transparency. Most people (especially non-technical end users) find such interfaces relatively cryptic and difficult to learn. [7] Juergen Haas believes that the CLI is generally easier to use if you can remember the commands and options, which you probably would if you use them frequently. [8]

4.2 Summary of the literature study results

The graphical user interfaces that are available for Git are a lot less used than the Git command-line. The results from our literature study shows that people with low experience seem to prefer a graphical user interfaces because they appear to be easier to use. The people that answered the GitSurvey in 2012 was a group of mostly everyday users or - more experienced than average users - of Git [1], so it is hard to judge whether the statement in the line above is valid or not but if you are to believe the previously introduced studies then it would make sense. After looking at the previous studies, we would have to say that command-line interfaces seem to be something that attracts more experienced users whilst graphical user interfaces attracts novice users.

Chapter 5

Results

5.1 Survey

The survey got 367 responses from people around the world whom use Git in one way or another. The sizes of each demographic can be seen below in figure 5.1.

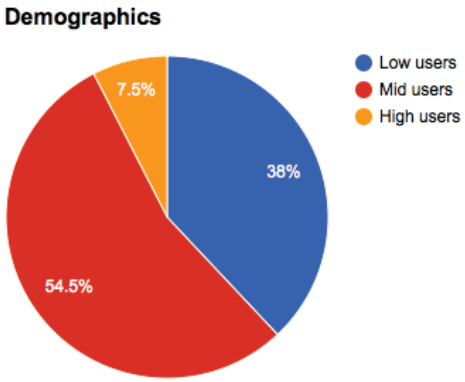
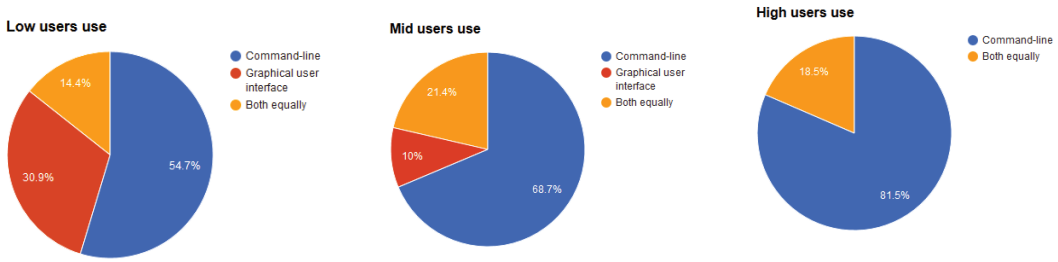


Figure 5.1: Graph showing the split among the demographics covered in the survey

In Appendix A you can find graphical representations of how the respondents to the survey answered each question, these diagrams show the full spectrum of respondents as a whole.
In Appendix B you can find correlation analysis and cross tabulations that we've put together using IBM SPSS Statistics [25].
We've chosen to include a Sig (2-Tailed) value in our correlation analysis to check if there's a statistically significant correlation between the variables we're measuring. Basically if the value is greater than 0.05 there's no statistically significant correlation else if the value is less or equal to 0.05 there is a statistically significant correlation.
A statistically significant correlation means that an increase or a decrease in one of our variables significantly relate to an increase or a decrease in our other variable. For example, Appendix B Figure B.1 which has a Sig.(2-tailed) value of

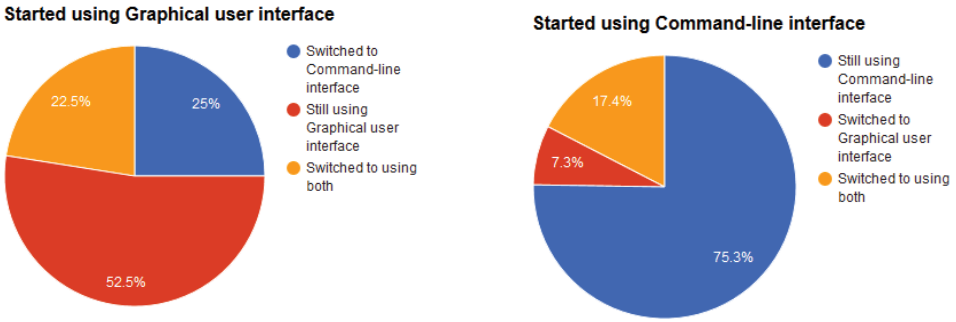
0,016 and therefore shows that there's a statistically significant correlation.

Below you'll see what kind of interfaces the different demographic groups use and below that are graphs displaying what interface our respondents started with and what interface they use today.



(a) Low experienced Git users (b) Mid experienced Git users (c) Highly experienced Git users

Figure 5.2: Representation of which interface users indicate that they use depending on demographic



(a) Started using GUI - shows what interface respondents use today. (b) Started using CLI - shows what interface respondents use today.

Figure 5.3: Started using - now using.

5.2 Interviews

The interviews were conducted with three subjects, each interview lasted approximately 5-15 minutes. The more experienced subjects were a little bit more talkative and had more to say on the subject since they have had the time to test other solutions and options, whilst the less experienced subject didn't quite understand all of the questions so we had to try to explain what we meant in an

objective way - as best as we could. By "in an objective way" - we mean that we avoided positive or negative words that could have influenced the subjects responses. A quality analysis has been carried out and the results can be found in Appendix C. The full transcripts, in Swedish, of the interviews can be found in Appendix D. The reason for the transcripts being in Swedish is that the interviews were carried out in Swedish, we felt that a translation could potentially damage the content and therefore we decided not to do it.

Chapter 6

Analysis

As shown in Appendix A, in Figure A.3(a) the most common interface to get started with - amongst the responders to the survey - is the CLI. The majority, 78.2% percent of the 367 respondents, stated that they started out with Git using the CLI and 21.8% of the respondents stated that they got started with the GUI. So to answer RQ1 - CLI is the most common interface used when getting started with Git. Diagrams in Figures 5.2b and 5.2c, shows that the command-line dominates amongst the mid to highly experienced users as well. The more experienced the less often they seem to entirely rely on a graphical user interface.
In comparison to 2012's edition of the GitSurvey, 4498 respondents to the question "What Git implementation do you use?" 95% answered "Git (core) command-line" [19].
The data makes the command-line interface appear to be the most used interface for Git amongst experienced Git users. However, it important to note that experienced users sometimes use both a command-line interface and a graphical user interface. With this in mind, it could be a hint that the users whom use both of the interfaces are missing a feature in the command-line interface that is available in graphical user interfaces or vice versa. Never the less, the answer to RQ2 is the same as for RQ1 - the CLI is most commonly used amongst experienced users as well.

Based on the answers we received in our interviews it could very well be that the command-line seems to lack a good visualization of the repository log, something that most graphical user interfaces seem to handle pretty decently. A good visualization of the log in the command-line interface is, and we quote, "difficult to do without breaking your fingers" (Bagge, personal communication, May 11, 2015).

"What can be gained from using the GUI" (RQ3) is a question that is hard to answer, simply because there are a number of different GUI's available and their functionality therefore naturally differs. One thing that does appear to be a common phenomenon in graphical user interfaces is the visualisation of the log. Another thing that is worth noting is that by using a graphical user interface you

do not have to remember all of the commands necessary to commit and push code.

Even though the CLI appears to be difficult for new users, see Appendix A Figure A.10, our results claim that it offers more control. The "control"-part seems to be something that all user groups agree on, see Appendix A Figure A.10, A.12 and A.14. However, as explained by one of the people we interviewed, a graphical user interface might perform multiple commands when you press one button. This would naturally make it harder to actually understand which commands are executed when a button is pressed (Bjorner, personal communication, May 11, 2015). On the other hand, in the CLI you have to, unless you've created your own aliases, type commands manually. This leads to us believing that the CLI would provide a better understanding of how Git works in general than a GUI, even though it would mean a steeper learning curve. The same goes for RQ3 and RQ4 it is tough to answer without a proper UI investigation. But in terms of convenience and when coupled with our data, it would seem that you would gain more control over your workflow in the CLI than in a GUI.

Looking at what interface our respondents started out using and what they use today, we can see that out of all the respondents whom began using the CLI to interface with Git roughly 76% of them still use the CLI and only 7.2% swapped to a graphical user interface, see Figure 5.3(b). What is interesting though is that 16.9% swapped over to use both the CLI and a GUI. Meanwhile, if we take a look at the users who began using a graphical user interface about 52% still use a graphical user interface while 25% swapped over to the CLI and 22.5% swapped to both CLI and GUI, see Figure 5.3(a).

There is a strong positive correlation between how experienced the respondents are and how many of them use the CLI, as seen in Appendix B Figure B.1. The same does not apply to the usage of the GUI, because as we can see in Appendix B Figure B.2 there's a strong negative correlation which can be interpreted as the more experience the respondents have the less likely they are to use solely a GUI when interfacing with Git. In fact, none of the highly experienced respondents use a GUI solely (Figure 5.2c).

Looking at the cross tabulations we can see that a graphical user interface is more used amongst respondents whom learned Git through a friend/colleague or in a course rather than through an open source community, book, online tutorials or by themselves (Appendix B, Figure B.4).

In collaborative work it's more common to use a graphical user interface than when working with large scale version control with branch management and code review through Git. About 30% use a GUI for basic change tracking and collaborative work, whilst only 10% use it for large scale version control with branch management and code review.

As mentioned earlier, the GUI is used quite often in combination with the CLI,

at least when working with large scale version control with branch management and code review.
Another interesting thing, shown in Figure B.6 in Appendix B, is that the type of OS does seem to matter in terms of how many use the CLI/GUI or both.
In Windows roughly 32% use a GUI whilst in Mac the number is down to 15,5% and in Linux 1,5%. From our point of view we believe that those numbers show a correlation between how GUI dependant an OS is and what type of UI the users use for Git. In that case, it would mean that there's a quite strong negative correlation. The same thing could be applied for the CLI where roughly 49% of the Windows users, 61,9% of the Mac users and 82% of the Linux users use a CLI which would suggest a strong positive correlation instead.

The correlation analysis put together with the results from the interviews - the fact that the CLI does lack visualization of logs but provide a more controlled environment - shows that graphical user interfaces are more attractive amongst novice users than experienced users. But as the users gain experience they seem to swap their GUI for a CLI or to use them both together.

Our findings summarized in a list:

- The command-line interface is the most widely used interface, amongst all Git users.
- The graphical user interface is used less and less by it's own as the user gains experience.
- The graphical user interface is still used by highly experienced users but only used together with the CLI and not on its own.
- How you learnt Git, what OS you use and what you use Git for - are factors that influence what interface you use. For example, a graphical user interface is much more common amongst Windows users than amongst Mac or Linux users. Also a graphical user interface is much more common amongst those who learnt Git through a friend/colleague or in a course than amongst those who learnt Git through open source communities or online tutorials.
- The general image, provided by the respondents and based on the analysed data, is that the CLI has a steeper learning curve but offers more control, than the GUI, and the GUI is helpful and simple but contains a lot of auto-magic.
- Both interfaces have their pros and cons but it's impossible to say which is the better one. It's all a matter of what you use Git for and your preferred way of working.

- According to the interviews we've conducted, graphical user interfaces have detailed logs that are easy to browse through - something that the CLI lacks by default.
- The GUI and the CLI has a lot of similarities in terms of functionality, most of the functionality found in the CLI can be found in the GUI such as alias, action customizing and freedom of workflow construction. However, it's hard to say what the exact differences are in terms of convenience and functionality without a proper usability investigation, which is something we didn't have time to do.

Chapter 7

Conclusion

In order to conclude that we have satisfied our own curiosity, and answered the research questions, we needed to have collected quite a bit of information. We needed to know how the ratio of GUI to CLI users of git is weighted. We needed to know what git users of a more experienced standing use in their version control work. We needed to identify some possible factors that may sway a users judgement of an interface. Finally we needed to know a bit about how statistics about swapping looks, and along with that some correlations with possible factors that play into the choice.
Bellow we will sum up our findings and draw what conclusions we can around what we have found.

- The command-line interface is the most widely used interface, amongst all Git users no matter previous experience.
- We have found three major factors that influence what UI is typically used - How you learnt Git, what OS you're using and what you are using Git for in your work.
- GUIs are more common amongst novice users and less common amongst experienced users.
- It is not unusual that experienced users use a GUI together with a CLI.
- It is hard to determine which UI is the better one in terms of functionality and convenience. A proper UI investigation might help in such a regard.

These are the major findings in our work. Our guess as to why the CLI is more used than a GUI is that users want control and it's hard to offer the same type of control in a GUI as the CLI does. The entire point of having a GUI is to simplify things, so when you press a button in a GUI you might actually trigger ten different actions in the background - actions that you normally would have to perform one by one in a CLI.
Another thing, that has been mentioned before in the thesis, is that the CLI lacks good visualizations of logs which GUIs seem to do a great job at offering. We

believe - based on the data we've collected - that this is why a lot of users use both interfaces, a GUI to get good visualizations of logs and the CLI for control.

Three major factors influence what type of interface the users are most likely to use. The first one is how they learnt Git, if they learnt it through a colleague/friend or through a book seems to make a difference. The second factor is what OS they are using, Windows is a more GUI-based OS than Linux. In correlation to that we can see that there is a larger percentage of the Git users that use Windows and a GUI for Git than there is Git users that use Linux and a GUI for Git. The last factor is what they are using Git for, in collaborative work it is more common to use a GUI than it is when working with large scale version control with branch management and code review through Git.

It is not uncommon that the GUI is used in combination with the CLI, so it is important to remember that the GUI does play a role for some of the more experienced users.

The use of graphical user interfaces alone declines as knowledge of Git increases, as shown in the correlation analysis in Appendix B. One theory, of ours, of why this is happening is that as a user gains more knowledge he/she will want to optimize his or her work-flow to the fullest extent. To save time and to have maximum control. Since roughly 70% of the respondents associated the CLI with control in comparison to the 11% that associate the GUI with control it would make sense that the CLI would be significantly favoured amongst the experienced Git users.

When speaking of learning, a graphical user interface could be considered a less painful way of learning Git. However you probably don't achieve the same type of control and understanding of Git through a GUI compared to the command-line interface.

The command-line interface is associated with the words "control" and "difficult" by the majority of our respondents whilst the GUI is associated with words such as "Simple", "Easy", "Helpful", "Automagic" and "Time consuming".

It is still hard to argue which of the UIs that is the better one, it would probably depend entirely on how you use Git.

Chapter 8

Risks

There are obvious risks such as people might have filled in the survey more than once. We tried to minimize the risk of that by deleting obviously duplicated entries once all the data was placed in a database based on the respondents email addresses, but since there is an option to not submit the survey with an email address there is a risk that someone might have messed with the results.
Speaking of the interviews on the other hand, the fact that one of our subject didn't have a lot of experience with Git could potentially be a risk as well seeing as we had to explain our questions further at some points which might influence his answer in a way that it shouldn't. We also lack some key components in our research that we had originally planned to include, such as the UI analysis.

Chapter 9

Future work

It would be interesting to see a complete user interface investigation to see what the actual differences are between the CLI and a GUI. Maybe put focus on either time consumption when performing certain actions or a focus on customization of actions.
Another thing that would be interesting to find out is if there's any type of milestone, speaking of experience of Git, at which most users swap their interface for another. Like a straw that broke the camels back kind of a moment.

Acknowledgement

We would like to thank our advisor Conny Johansson for his continued support and feedback. The feedback we've received throughout working with this thesis has been very valuable to us.
We would also like to thank all of the people who took the time to answer our survey, a special thanks goes to the members of the sub-reddits; /r/git, /r/SampleSize, /r/GradSchool, /r/AskReddit, /r/github, /r/software, /r/learnprogramming, /r/asktechnology. The feedback and excitement we received from the reddit community really cheered us up and gave us confidence in our topic.
Furthermore we would like to extend our gratitude to the three subjects that allowed us to interview them, Tord Eliasson, Niclas Björner and Martin Bagge. They were very insightful and contributed to figuring out how to research a lot of the content in this thesis.
There are others whom have helped us as well but we can not list everyone, but a collective thank you to all of the participants of this thesis, large and small contributions included!
Finally we would like to thank our program manager Mikael Roos, without you allowing the class to bicker about what UI to use we would not have thought of this topic.

References

[1] GitSurvey2012 - Git SCM Wiki. (2012). GitSurvey2012 - Git SCM Wiki. Available: `http://git.wiki.kernel.org/index.php/GitSurvey2012`. Last accessed 4th April 2015..

[2] Morgan, K., Morris, R. & Gibbs, S. (1991). When does a Mouse become a Rat? or ... Comparing Performance and Preferences in Direct Manipulation and Command Line Environment. The Computer Journal. 34 (3), p265-271.

[3] Kirkpatrick, D., Sherman, S., & Deutschman, A. (1993). Mac vs Windows. Fortune Internc#ional. 128 (8), p57-64.

[4] Kelleher, J. (2014, 17-19 Jan). Employing git in the classroom. Paper presented at 2014 World Congress on Computer Applications and Information Systems (WCCAIS). Retrieved 4th April, 2015, at IEEE Xplore. DOI: 10.1109/WCCAIS.2014.6916568.

[5] Treweek, P. (1996). Comparing Interfaces: Should We Assume that Ease of Use Influences Users' Preference?. Presented at Sixth Australian Conference on Computer-Human Interaction. Retrieved 4th April, 2015, at IEEE Xplore. DOI: 10.1109/OZCHI.1996.560004.

[6] Perez De Rosso, S. & Jackson, D. (2013). What's wrong with git?: a conceptual design analysis. Presented at 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software. Retrieved 4th April, 2015, at ACM Digital Library. DOI: 10.1145/2509578.2509584.

[7] Eric S. Raymond. (2005). The Art of Unix Programming. Available: `http://www.catb.org/esr/writings/taoup/html/ch11s04.html`. Last accessed 6th April 2015.

[8] Juergen Haas. Linux: GUI vs. Command line. Available: `http://linux.about.com/cs/softofficeutility/a/gui_cli.htm`. Last accessed 6th April 2015.

[9] Github.com - Features. (2015). Available: `https://github.com/features/`. Last accessed 16th April 2015.

[10] S. Chacon, Pro Git, New York, NY: Apress, 2009.

[11] The Eclipse Foundation. "Eclipse Open Source Developer Report". The Eclipse Foundation. San Francisco, CA. 2012.

[12] ZeroTurnaround. "Developer Productivity Report 2012: Java Tools, Tech, Devs & Data". ZeroTurnaround. Boston, MA. 2012.

[13] Microsoft Corporation, "Survey Results: Open Source Developer Preferences (June 2011)," Microsoft Corporation, 11 July 2011. [Online]. Available: `http://blogs.msdn.com/b/codeplex/archive/2011/07/11/survey-results-open-source-developer-preferences-june-2011.aspx`. [Accessed 18th April 2015].

[14] Webopedia.com. (2015). Available: `http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html`. [Accessed 18th April 2015].

[15] C. Kemper and I. Oxley. Foundation Version Control for Web Developers. Berkeley, CA: Apress. 2012.

[16] Wikipedia. (Unknown). History of the graphical user interface. Available: `http://en.wikipedia.org/wiki/History_of_the_graphical_user_interface#Apple_Lisa_and_Macintosh_.28and_later.2C_the_Apple_IIgs.29`. [Accessed 20th April 2015].

[17] Loot Crate. Available: `www.lootcrate.com` [Accessed 20th April 2015]

[18] Google Spreadsheet. Available: `https://www.google.se/intx/sv/work/apps/business/products/sheets/?utm_source=google&utm_medium=cpc&utm_campaign=emea-se-sv-drive-bkws-all-trial-e&utm_term=google%2Bkalkylark` [Accessed 26th April 2015]

[19] Unlearn what you have learned. (2014). Git: How it works. Available: `http://bartoz.no-ip.org/archives/2981`. [Accessed 14th May 2015].

[20] Git SCM. GUI Clients. Available: `http://git-scm.com/downloads/guis`. [Accessed 14th May 2015].

[21] Neal Stephenson. (1999). In the Beginning was the Command Line. Available: `http://cristal.inria.fr/~weis/info/commandline.html` [Accessed: 14th May 2015].

[22] Spinellis, D. 2005. Version control systems. IEEE Software 22, (5): 108-109

[23] Git SCM. Getting Started - About Version Control. Available: `https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control`. [Accessed 3rd August 2015].

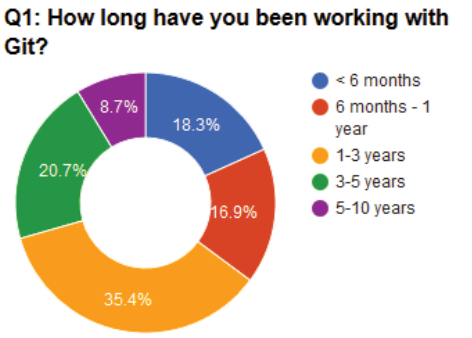
[24] Chapter 1 Differences Between Command Line Interface and Graphical User Interface (Solaris Advanced User's Guide). Available: `http://docs.oracle.com/cd/E19683-01/806-7612/startup-78447/index.html` [Accessed 10th August 2015].

[25] IBM SPSS Statistics. Available: `http://www-01.ibm.com/software/analytics/spss/products/statistics/` [Accessed 10th August 2015].

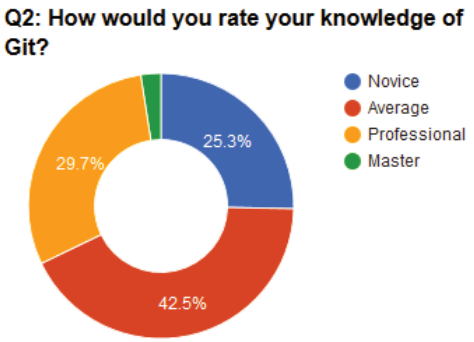
[26] Giancarlo Lionetti. (2012). Git vs Mercurial: Why Git?. Available: `http://blogs.atlassian.com/2012/03/git-vs-mercurial-why-git/`. Last accessed 6th September 2015.

Appendix A

Data from survey

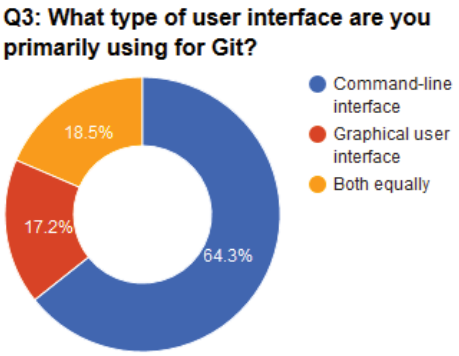


(a) Representation of results of question 1

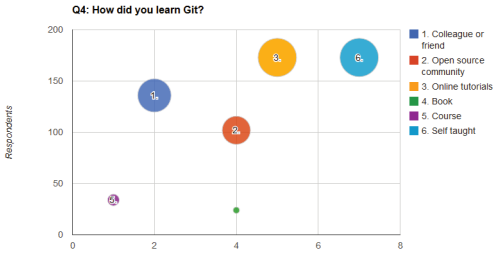


(b) Representation of results of question 2

Figure A.1: Questions 1 and 2

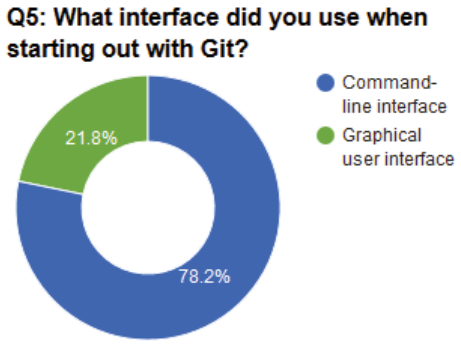


(a) Representation of results of question 3

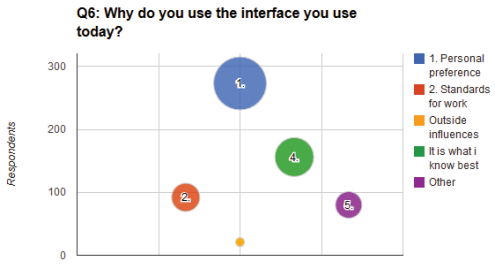


(b) Representation of results of question 4

Figure A.2: Questions 3 and 4

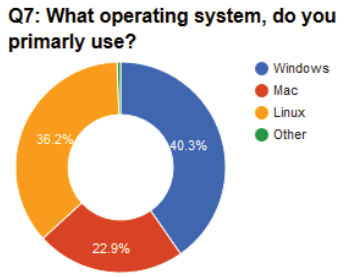


(a) Representation of results of question 5

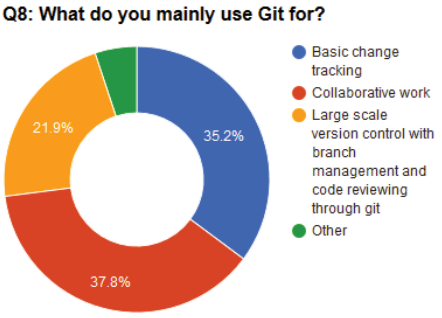


(b) Representation of results of question 6

Figure A.3: Questions 5 and 6

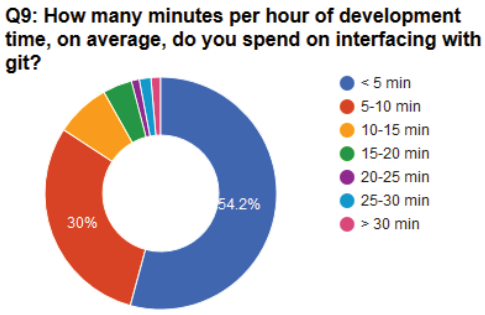


(a) Representation of results of question 7

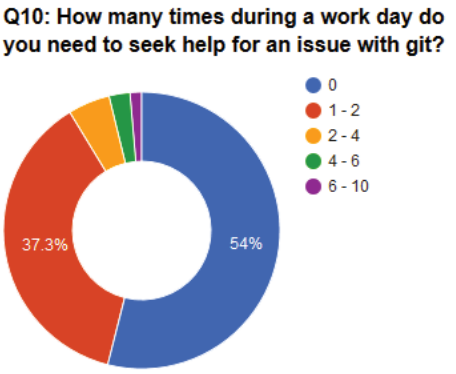


(b) Representation of results of question 8

Figure A.4: Questions 7 and 8



(a) Representation of results of question 9



(b) Representation of results of question 10

Figure A.5: Questions 9 and 10

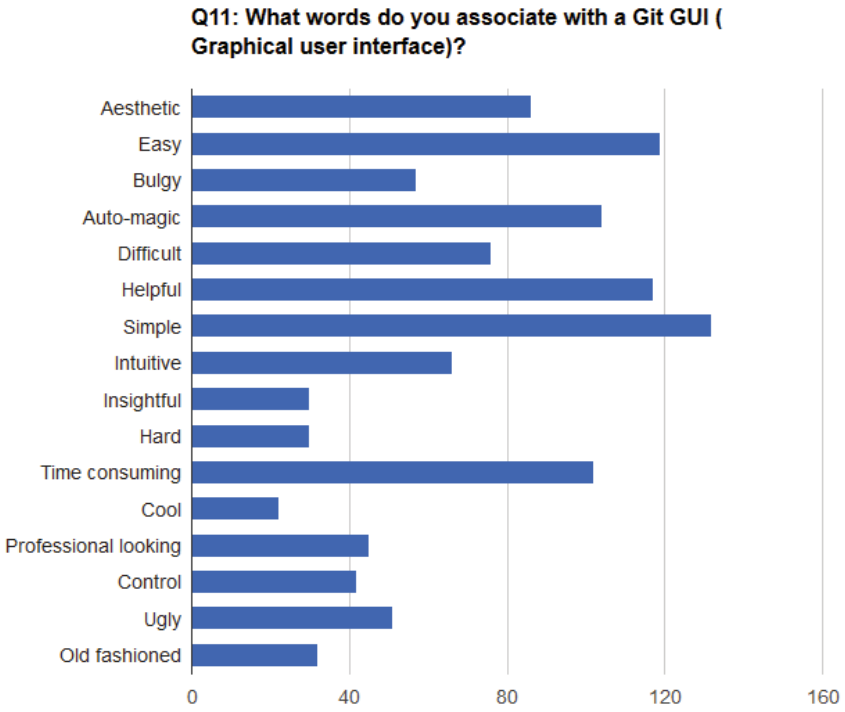


Figure A.6: Representation of results of question 11

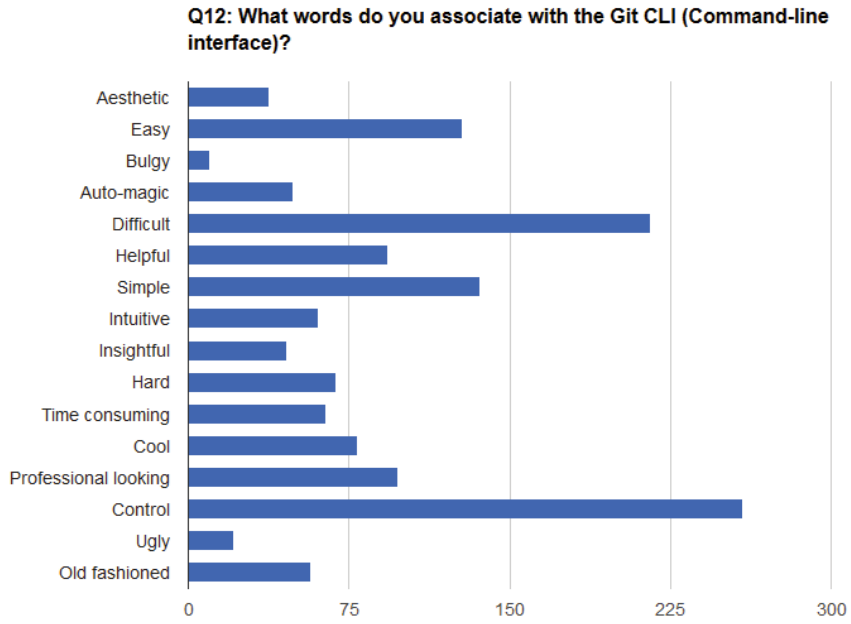


Figure A.7: Representation of results of question 12

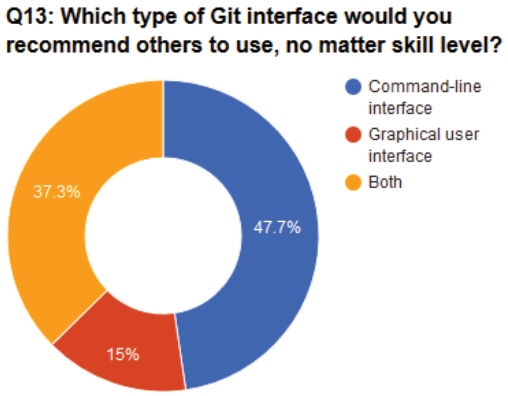


Figure A.8: Representation of results of question 13

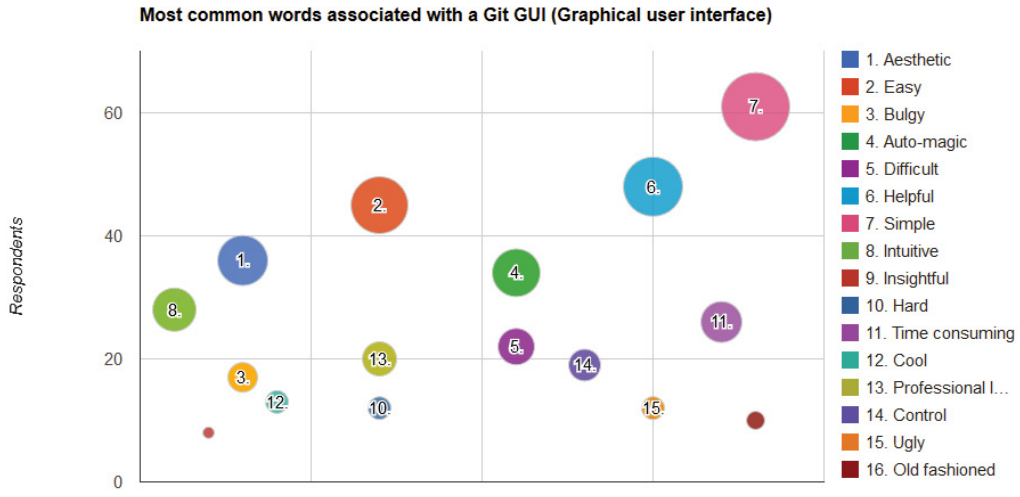


Figure A.9: Low experienced Git users - common words associated with Git GUI.

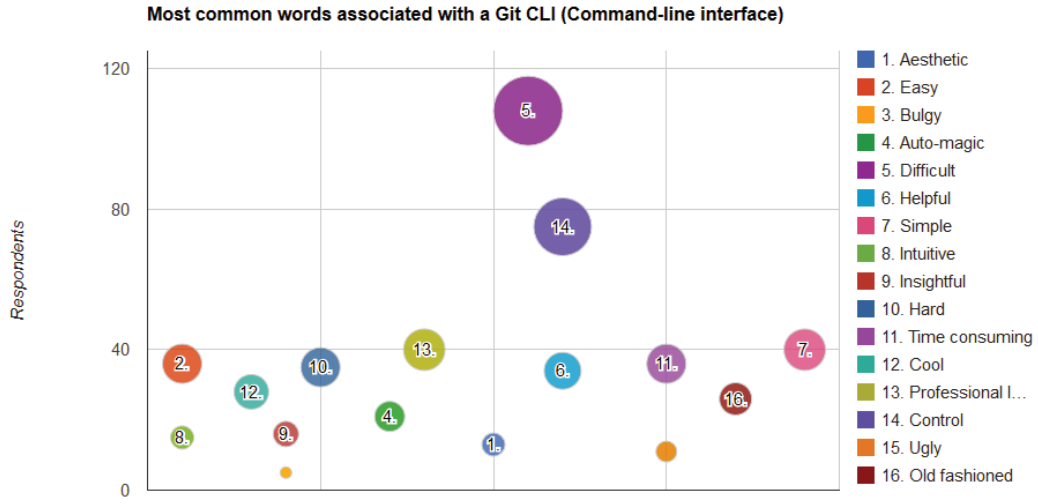


Figure A.10: Low experienced Git users - common words associated with Git CLI.

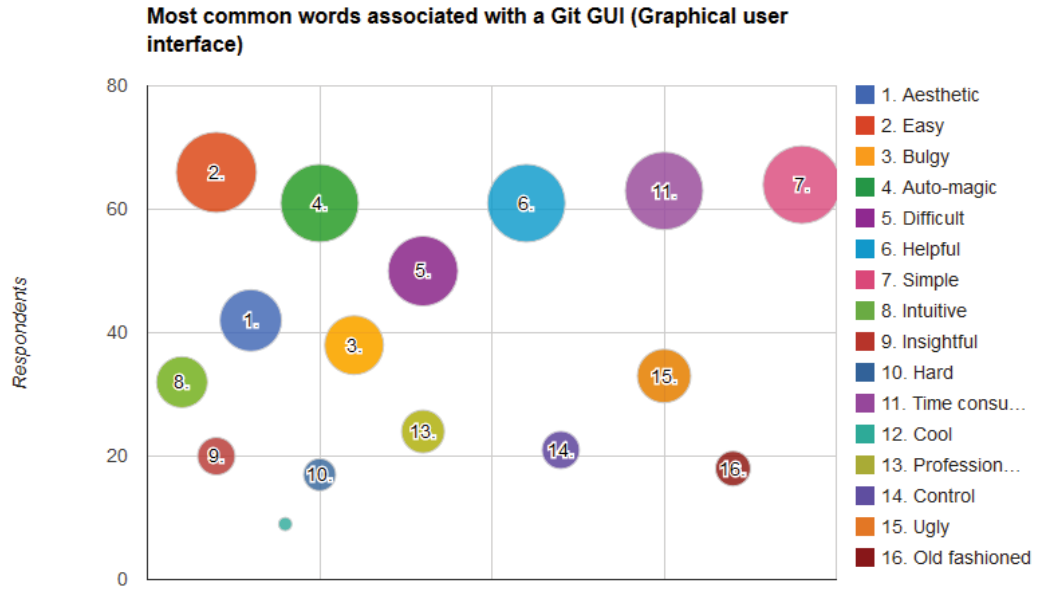


Figure A.11: Mid experienced Git users - common words associated with Git GUI.

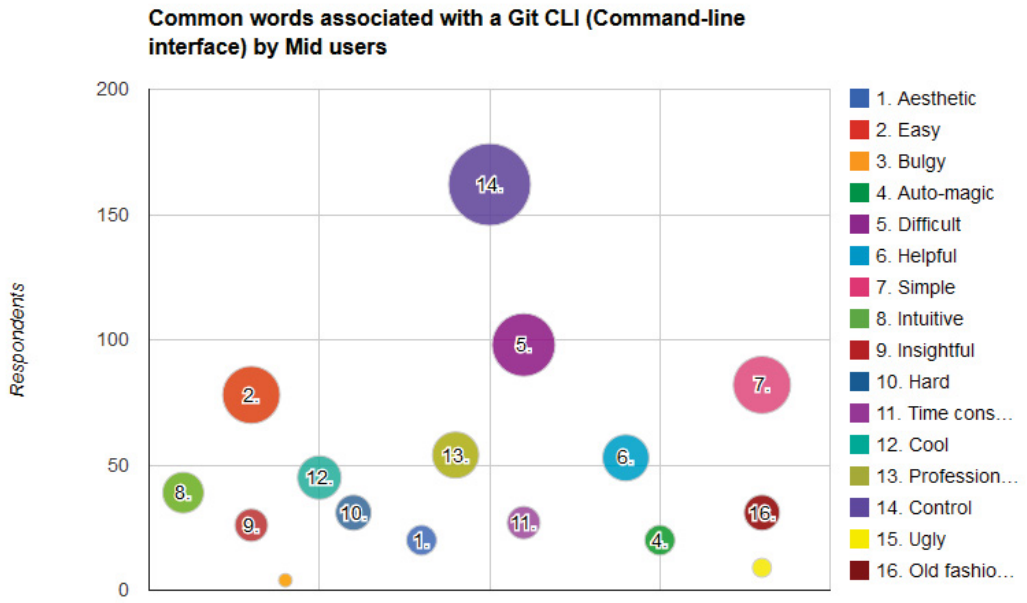


Figure A.12: Mid experienced Git users - common words associated with Git CLI.

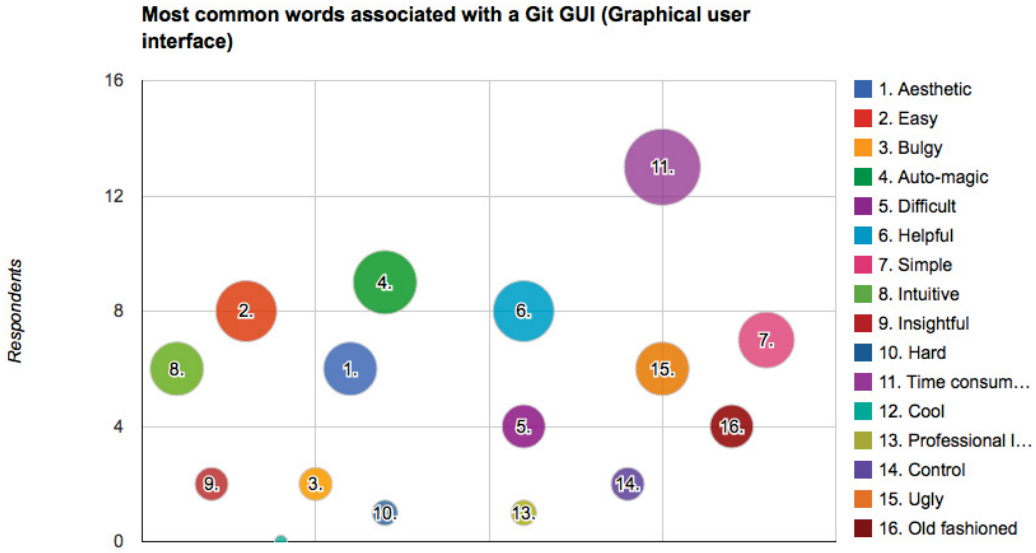


Figure A.13: Highly experienced Git users - common words associated with Git GUI.

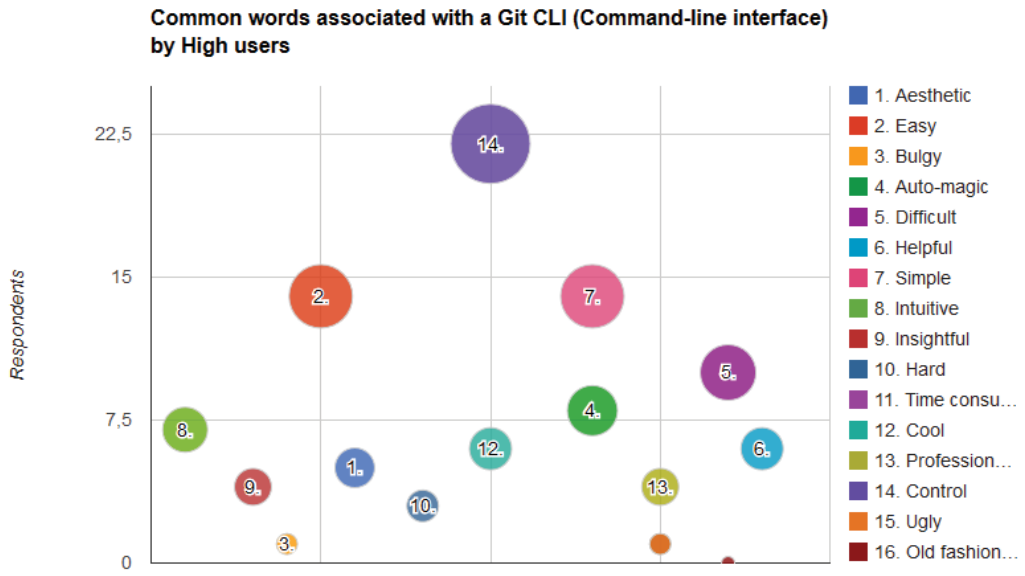


Figure A.14: Highly experienced Git users - common words associated with Git CLI.

Appendix B

Correlation Analysis & Cross tabulations

		Level	Percentage
Level	Pearson Correlation	1	1,000*
	Sig. (2-tailed)		,016
Percentage	Pearson Correlation	1,000*	1
	Sig. (2-tailed)	,016	

*. Correlation is significant at the 0.05 level (2-tailed).

Figure B.1: Correlation between experience and usage of the CLI. Shows that as experience (in the analysis referred to level) increase the number of CLI users increase. Sig-value of 0.016 which shows a statistically significant correlation.

		Level	Percentage
Level	Pearson Correlation	1	-,980
	Sig. (2-tailed)		,128
Percentage	Pearson Correlation	-,980	1
	Sig. (2-tailed)	,128	

Figure B.2: Correlation between experience and usage of the GUI. Shows that as experience (in the analysis referred to as level) increases the number of GUI users decreases. Sig-value of 0.128 which shows that there is no statistically significant correlation.

		Level	Percentage
Level	Pearson Correlation	1	,583
	Sig. (2-tailed)		,604
Percentage	Pearson Correlation	,583	1
	Sig. (2-tailed)	,604	

Figure B.3: Correlation between experience and usage of both the CLI and GUI. Shows that as experience (in the analysis referred to as level) increases the number of users, that use both a GUI and the CLI, increases. Sig-value of 0.128 which shows that there is no statistically significant correlation.

Interface * Alternative Crosstabulation

			Alternative						
			Colleague or friend	Open source community	Online tutorials	Book	Course	Self taught	Total
Interface	Command-line interface	Count	27	23	38	4	8	34	134
		% within Alternative	57,4%	85,2%	69,1%	80,0%	53,3%	63,0%	66,0%
	Graphical user interface	Count	12	2	7	0	5	8	34
		% within Alternative	25,5%	7,4%	12,7%	0,0%	33,3%	14,8%	16,7%
	Both equally	Count	8	2	10	1	2	12	35
		% within Alternative	17,0%	7,4%	18,2%	20,0%	13,3%	22,2%	17,2%
Total		Count	47	27	55	5	15	54	203
		% within Alternative	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%

Figure B.4: A cross tabulation showing how many that learnt Git through each of the options available in question 4, in the survey, and what interface they are using now.

Interface * Alternative Crosstabulation

			Alternative			
			Basic change tracking	Collaborative work	Large scale version control with branch management and code reviewing through git	Total
Interface	Command-line interface	Count	38	32	16	86
		% within Alternative	58,5%	50,8%	53,3%	54,4%
	Graphical user interface	Count	19	20	3	42
		% within Alternative	29,2%	31,7%	10,0%	26,6%
	Both equally	Count	8	11	11	30
		% within Alternative	12,3%	17,5%	36,7%	19,0%
Total		Count	65	63	30	158
		% within Alternative	100,0%	100,0%	100,0%	100,0%

Figure B.5: A cross tabulation showing what respondents use Git for and what interface they are using for their work.

Interface * Alternative Crosstabulation

			Alternative				
			Windows	Mac	Linux	Other	Total
Interface	Command-line interface	Count	73	52	109	2	236
		% within Alternative	49,3%	61,9%	82,0%	100,0%	64,3%
	Graphical user interface	Count	48	13	2	0	63
		% within Alternative	32,4%	15,5%	1,5%	0,0%	17,2%
	Both equally	Count	27	19	22	0	68
		% within Alternative	18,2%	22,6%	16,5%	0,0%	18,5%
Total		Count	148	84	133	2	367
		% within Alternative	100,0%	100,0%	100,0%	100,0%	100,0%

Figure B.6: A cross tabulation showing what OS respondents use and what interface they are using.

Appendix C

Quality Analysis of interviews

C.0.1 Bagge

The command that the respondent uses the most is git grep, a command that the respondent is unsure about if it exists in a GUI.

The respondent has used graphical user interfaces in the past, but it would appear as if it doesn't occur that often. But he knows about tortoiseGit and GitK, he also talks a little bit about tig. So it's safe to say that he has some knowledge about what's out there in terms of GUIs.

The respondent also explains why he's not working in a GUI environment by talking about the fact that he doesn't have a workflow for it so he doesn't really know what to do.

When talking about tig, he mentions that the CLI lacks a visualization of the log. He also mentions that you could create your own visualization of the log in a CLI but that it's hard to do.

When asked how he feels about working using the CLI in comparison to a GUI, he says that the CLI feels like home. It is in the CLI where he does his works.

C.0.2 Bjorner

The respondent mainly uses Git for branch managing and has used a GUI quite often in the past. He thinks that the GUI is great for log visualizations, he thinks that he can easily get a good overview of the repository history. He uses the CLI as his primary interface.

The respondent believes that the GUI is a good starting point for new Git users that aren't used to working in a CLI.

He, himself, is very comfortable working in the CLI because it provides a feeling of control.

The respondent doesn't think that he gains anything by using a GUI when working alone but feels that it's useful when working together with someone else when - for example - browsing through repository history.

C.0.3 Eliasson

The main functionality the respondent uses is the git clone command, to clone repositories. He does not have much experience with working with GUIs, he primarily uses a CLI.

The respondent does not have much experience working with a CLI either, but feels like it's hard to tell when something goes wrong when using a CLI. He says that; going from a GUI to a CLI is a little bit wierd, you don't always keep track of where you are (assumingly where you are in terms of Git events, directories etc.). So one thing he misses from the GUI is some kind of indication of where he is and what he's doing in the CLI.

Appendix D

Transcripts of the interviews

D.0.1 Bagge

sebastian: Vilken funktionalitet använder du oftast? eller vilka funktionaliteter är det du vanligtvis använder?
respondent: Gissningsvis är det kommandot som jag mest använder git grep.
sebastian: Git grep?
respondent: Ja, för att leta i källkoden liksom efter någonting.
sebastian: Men det git grep är ingenting som egentligen finns tillgängligt i grafiska gränssnitt, va?
respondent: Oj, det har jag ingen aning om.
respondent: Det hade ju varit rätt praktiskt om jag öppnade min källkod i en sån källkodsbrowser-grej, för det blir det ju i det fallet, så skulle jag ju kunna tänka mig göra grep i den liksom.
sebastian: Hur upplever du att det är att arbeta med ett grafiskt interface emot Git? Jag gissar att du har gjort det någon gång åtminstone.
respondent: Ja, det händer ju. Det beror ju lite på vad man lägger det emot förstås. Men om man tar dem strikt grafiska typ tortoiseGit och GitK och...
sebastian: SourceTree?
respondent: Det har jag aldrig hört talas om, eller har hört talas om det men aldrig använt det. Det är väll egentligen tortoiseGit och GitK som jag använt. Sen har man dem som är borderline typ tig, som är ganska så trevligt när man gör vissa grejer. Jag kan egentligen inte arbeta i dom (grafiskt interface mot Git), är väll det rent praktiska i det.
sebastian: Okej, på vilket sätt?
respondent: Jag har inget arbetsflöde för det så jag vet inte hur jag ska göra. Så jag har använt det någon gång då och då för att titta på olika saker av olika skäl. Det är inget för mig. Tig har jag använt för att den har någon visualisering av loggen som är helt underbar liksom. Men det va längesen jag använde det senast. Så på något vis har jag väll inte behövt det, det är väll typ när man ska ner i loggen och leta långt bak och så ska man försöka förstå hur det hänger ihop så har den ett litet läge för att visa loggen med olika bakgrunder med histogram och sånt. Så att den har en funktion där som saknas lite på kommandoraden, eller den är svår att göra på kommandoraden utan att knäcka fingrarna.

sebastian: Nästa fråga då, ur upplever du att det är att arbeta med ett CLI emot Git?
respondent: Det är där jag bor liksom, jag jobbar med kommandoradsprylar. I det som jag producerar på jobbet så är det ju kommandoradsprylar. Att då jobba i kommanrodaden känns då som.. det vore lite löjligt att sitta och skriva kommandoradsprogram och sen jobba på ett annat sätt. Eftersom jag ändå sitter i en terminal när jag ändå ska prova koden så varför ska jag inte göra Git kommandon där i samma terminal. Någonstans där finns det en connection. Sen förövrigt så finns det ett flöde där som jag har vant mig vid.
sebastian: Vad saknar du ifrån ett GUI som du skulle vilja ha möjlighet att använda i ett CLI?
respondent: Det finns ju en grej som jag saknar när det kommer till loggen som jag egentligen inte tycker finns i någon utav dem som jag har provat. Det är att om jag vill titta på loggen så vill jag smidigt kunna följa ändringarna och sånt och det är komplicerat. Det är ju egentligen för att Git är byggt så som det är med förändringarna och sånt, så det är nog komplicerat att skriva ett kompetent UI oavsett vilken väg man väljer. Om du har den här ändringen kan du visa allt som har med den att göra med en snygg kedja som blir logisk för människohjärnan, för tyvärr är det ju så att göra det datalogiskt med hur dom hänger ihop är ju inte så komplicerat men det är inte så dom hänger ihop för mig som människa när jag tittar på ändringarna. Så ändringar kan komma fel, rent strukturellt, i mitt huvud jämfört med hur dom händer i trädet. Det finns en önskan om att loggen kanske kunde varit lite mer tillgänglig.
sebastian: Finns det någon tidsaspekt som du skulle kunna optimera om du hade ett grafiskt gränssnitt istället för ett CLI?
respondent: svaret är nej, för min del. Jag sitter aktivt och skriver raderna, men jag är ganska snabb på det för att jag vet vilken ordning allt ska vara i.
sebastian: Ja, det är väll muskelminne vid det här laget.
respondent: Ja.
sebastian: Som summering på sista frågan så kan man säga att du har en del alias, du har tre totalt.
respondent: Ja, det är dom som jag använder iallafall.
sebastian: Som gör dig lite smidigare, även om ett utav dom åtminstone är inte att rekommendera. Det är det du tycker gör saker och ting lättare och du saknar ingenting annat ifrån Guit.
respondent: Nej, inte medvetet iallafall.
sebastian: Förutom just då Tig-grejerna.
respondent: Ja, på sätt och vis så skulle det kunna vara så att jag har ju inte använt det så jag vet ju inte.

D.0.2 Bjorner

sebastian: Vilken funktionalitet i Git anser du att du jobbar med oftast, eller vilka funktionaliteter arbetar du mest med?
respondent: Jag skulle väll säga att jag för det mesta arbetar, just inom Git, med branchning för att lätt kunna experimentera loss i projekt jag sitter i utan att behöva ta sönder någonting i master. Jag använder det för att kunna ha en egen kopia utav våran källkod på jobbet, som jag arbetar med, utan att jag stör någon annan. Att, ja, implementera.
sebastian: Så branchmanagement i Git är det du pysslar med mest?
respondent: Ja, om man ändå bortser från det vanliga med commits.
sebastian: Hur upplever du att arbeta med ett GUI? Hur ser du på arbetet med det?
respondent: Jag har själv kört en del GUIn till Git. Inte så mycket GitGTK som är det vanliga, som man får med Git utan jag har använt SoureTree som är skapat av Atlassian. Jag tycker att GUIn är väldigt bra för att notera data, så du får snabbt en bra överblick över hur t.ex. hur ser din historik i Git. Du får data presenterat på ett väldigt vänligt sätt i GUIt, tycker jag överlag.
sebastian: Är det någonting mer du vill tillägga på GUI delen?
respondent: Ja, asså om man ska kolla på det hela så även med lite mer avancerade verktyg som har en terminalsida till det så tror jag att GUIn är en väldigt bra inkörningsport om man inte är så bekväm vid att sitta i en terminal.
sebastian: Samma fråga, fast riktat mot CLI.
respondent: Ja, jag trivs väldigt bra i terminalen för det känns som att jag har mycket mer kontroll över vad som händer i och med att verktygen ofta automatiserar långa kedjor utav kommandon baserat på vart du trycker. Om du trycker på en knapp så kan det vara fem eller sex kommandon som exekveras i ett GUI medans här får man göra allting verkligen för hand. Det kan ta mycket längre tid, det kan ta kortare tid beroende på hur van man är. Men jag känner att jag har mer kontroll i terminalen.
sebastian: Vad saknar du, om något, i det gränssnitt som du använder idag? Jag gissar på att det är CLI du använder idag?
respondent: Jajamen, det är det. Det jag saknar är väll lite som jag pratade om när jag pratade GUI, att snabbt kunna få en bra visuell representation utav saker. Jag kan inte, ur ren standard, få en bra representation utav min historik utan att kunna massa specialkommandon i Git tillexempel.
sebastian: Så dem grafiska egenskaperna är något som du känner gör arbetet med det mer lättar att hantera och lite effektivare?
respondent: Det beror helt på, om jag sitter själv och jobbar så tycker jag inte riktigt att det riktigt tjänar mig någonting att ha något grafiskt. Men om man ska sitta tillsammans med någon och kanske förklara någonting eller gå igenom en historik, då känner jag att det är mycket lättare att ha ett grafiskt gränssnitt att arbeta igenom.

D.0.3 Eliasson

Sebastian: Vilken funktionalitet i Git är det du använder oftast, vilken eller vilka?
Respondent: Det är mestadels för att klona grejer som andra har gjort.
Sebastian: Så bara hämta källkod för att kolla på?
Respondent: Ja.
Sebastian: Hur upplever du att det är att arbeta med Guit eller någon utav Guina som finns tillgänliga för Git?
Respondent: Jag har ju inte jobbat mycket alls med Guit, men det har ju varit bra hittills.
Sebastian: Är det någonting du tycker är extra trevligt i Guina eller extra kasst i Guina, som du har märkt.
Respondent: Nej, jag kan nog inte säga det.
Sebastian: Då går vi vidare till samma fråga fast för kommandoraden.
Respondent: Det har funkat bra, i och med att jag inte har använt det så mycket så har jag svårt att säga vad som är problematiskt. Man måste sitta lite med grejer för att märka av att det inte riktigt funkar som jag vill. Det är svårt att utmärka problem.
Sebastian: Det är ju problem ifrån din synvinkel, det ska inte vara buggar i systemet. Utan hur du uppfattar det.
Respondent: Nej, det förstår jag. Men jag har inte använt det tillräckligt mycket, känner jag.
Sebastian: Är det någonting du tycker är extra svårt, som typ branchhantering eller commithantering eller rebase?
Respondent: Det jag känner framför allt är att jag är ovan att skriva det på det sättet, men hittills har det funkat bra.
Sebastian: Hur mycket erfarenhet har du haft av kommandoraden tidigare innan du började använda Git?
Respondent: Jag läste ju Linux på gymnasiet.
Sebastian: De var en kurs eller?
Respondent: Det var en kurs, jag läste den cirka en termin. Om jag inte kommer ihåg helt fel så var jag rätt intresserad tillochmed. Men man tappar det snabbt, tyvärr.
Sebastian: Jo, så blir det tyvärr. När man inte jobbar med det mycket så försvinner det ganska fort. Men det är det rättvist att säga att du inte är jättevan vid att använda terminaler och sånt?
Respondent: Det börjar komma tillbaka, men jag är fortfarande ovan.
Sebastian: Vad saknar du, om något, i gränssnittet som du använder idag? Alltså vad saknar du i CLI:t för att bli mer effektiv, för att vara smidigare att använda, för att göra saker och ting lättare för dig i utveckling och arbetet med Git?
Respondent: Från ett GUI till en kommandorad blir lite sådär små konstigt,

man håller inte hela tiden koll på vart man är och så. Det är framför allt det jag känner att jag saknar, att jag vill ha någonting som indikerar vart jag är någonstans hela tiden.
Sebastian: Ja, precis. För man tappar bort sig litegranna ibland.
Respondent: Ja.
Sebastian: Jag gissar på att du inte har hållit på någonting med changeloggar och grafer och såntdär än?
Respondent: Nej.