User Research Results, System Requirements and Problem Definition

of CSC318 Group 3

B.Hu - benj.hu@mail.utoronto.ca

D.Cho - dawing.cho@mail.utoronto.ca

R.Qiu - rex.qiu@mail.utoronto.ca

P.Bilodeau - philip.bilodeau@mail.utoronto.ca
M.Huang-Hobbs - maxwell.huang.hobbs@mail.utoronto.ca

TA: Eugene Cheung

March 1st, 2015

PART I: User Research Results

Summary of Research Results

Users found that the command names did not relate to the task or function they accomplished. Generally, users who were not used to using CLIs found the interface complicated and had not the slightest inkling of how to use it without any given direction (e.g. the help sheet). Inexperienced users did not like memorizing commands and arguments, and were confused upon the error messages that followed from invalid input, noting it to be unhelpful and often confusing. Furthermore, novice users had a hard time determining whether a task had been successfully completed, since the shell sometimes has no output after a successful command (e.g. creating a new directory, moving a file, etc.).

Experienced users' main gripes largely consisted of quality of life issues rather than any fundamental problem with the interface itself (e.g. syntax highlighting, colour customization, command completion, more concise man pages, etc.).

On the other hand, users were impressed by the power and capabilities of the interface and what could potentially be accomplished using it after having gained sufficient experience.

Overall, most users felt that CLIs are not user-friendly to all levels of users, particularly amateurs. They noted a steep learning curve from the time they used it/learned it initially, and that it was not intuitive to use.

User Needs List (Prioritized)

- Command names should be able to be recalled by some means as it may be difficult to remember them and the example usage even with experience.
- Command names should better relate to the function they accomplish, for example, with more intuitive information.
- Error messages should not be obtuse and should be helpful in all cases.
- Visual cues to help guide use (i.e. some spatial awareness with respect to where what your scope is currently).
- Should be more friendly to beginning users in general.
- Auto-correction / auto-complete of arguments and commands.
- Should minimize eye fatigue since staring at a CLI for hours can be a very tired experience.
- Output message after successful command.

Stakeholders' Descriptions

- Beginner Users
    - Users who have little or no experience with command-line interfaces but wish to learn, for examples, computer science students in universities. Having a more accessible shell will make the experience transitioning to using a shell easier during the learning process.
    - Users who have never used a command-line interface before. Having access to an enhanced shell may bridge the difference between non-user.
- Instructors/Educators of computer science. Having access to a better "learning shell" may make the process of teaching and learning easier/more streamlined for them and their students.
- All other existing CLI users. Especially those whose usage may be sparse or those with a limited command vocabulary/repertoire.

Primary Personas

Our primary personas are a group of second or third year computer science students in university. They usually understand the abstractions typical to their operating system of choice (files, folders, processes) but have never had to look into the underlying systems. However they have only briefly worked in a command-line shell in the past, some of them have no experience at all. However, a few of their required courses need a certain level knowledge in using a CLI. Therefore their basic **goal** is to make it through those required courses with as little pain as possible and become comfortable with using CLIs.

Behaviors:

- They find it difficult to remember the order arguments to commands and the command names themselves.
- They also find it difficult to find what commands to use and how to use them.
- When encountering a problem, they typically turn to google to find an answer and ask questions on the course forum since they don't know what other resources to turn to.

Attitudes:

- They usually think of CLIs as 'frustrating', unintuitive, or complicated and so, generally almost in all cases, prefer GUIs.

Scenarios

1. [Does not know what to do] John Doe is a second year computer science student who is learning to use the command-line. One day he wanted to find only the file names with the .java file extension in a messy folder. He does not know about the grep or find commands. John uses google to search “bash finding java files only” and is unable to find exactly what he wants: just to get the file names and instead sees how to get full paths among other unrelated information. He quickly becomes overwhelmed.
2. [No offline documentation] Dirk is a student working on his laptop on the subway ride to class. He is trying to learn the command-line to make use of his time more efficiently, but cannot remember the command for displaying a given command’s manual page. Without access to the internet from the subway, Dirk’s learning comes to a halt.
3. [Managing file names] Stallman the specialist wants to automatically manage file names of things dumped into a shared network file system. He knows from his experience that he can do that with `incron`, but can’t remember exactly how. Stallman reads the man page, sets up an `incrontab` user table for a custom renaming script, and feels good about his automatically managed files.
4. [Forget the order to issue a set of commands] Eric the newbie spent hours in his programming assignment, and finally finished it before the deadline. He was using svn to version control his files. He first wanted to move the files to the local copy folder. But unluckily he forgot the command and the detailed location of it. So he opened Finder to look for the address, afterwards he googled the internet and found the command he needed was ‘mv dir1 dir2’. Then he typed ‘svn commit -m’ (which was wrong), but the error message returned. He typed ‘svn commit m-’ (without changelog). Of course, the commitment is still unsuccessful, because he forgot the changelog. After retyping it again, because he did not know he can use up arrow to repeat previous command, adding the changelog ‘Updated assignment 1’, the commitment failed again, indicating his files are not under version control. Eric struggled with this problem for a while, finally resolved it with the help from one of his classmates. It turned out that he forgot to ‘svn add filename’ before committing.

PART II: Design Requirements

Problems to address

- The complexity of the shell for new or casual users with limited experience.
- The difficulty of learning and remembering command names and techniques.
- The limited feedback provided by current command-line interfaces (mainly recovery from error and acknowledgement of success)
- Quality of life improvements that experienced users may appreciate

Design Principles

- Simplicity and clarity without sacrificing usability or power.
- Intuition-building.
- Fully-customizable.
- Minimalist and lightweight in design.
-

User Needs

- Command names should be able to be recalled by some means as it may be difficult to remember them and the example usage even with experience.
- Command names should better relate to the function they accomplish, for example, with more intuitive information.
- Error messages should not be obtuse and should be helpful in all cases.
- Visual cues to help guide use (i.e. some spatial awareness with respect to where what your scope is currently).
- Should be more friendly to beginning users in general.
- Auto-correction / auto-complete of arguments and commands.
- Should minimize eye fatigue since staring at a CLI for hours can be a very tired experience.
- Output message after successful command.
-

Environmental Requirements

- Physical environments vary and do not affect usage
- Environment is contained within the computer screen, and has the same look and feel as other command-line interfaces

Functional Requirements

- Improve and provide easier to use command-line interfaces
- Requires users to have an understanding of computers

Technical Requirements

- Must be compatible with or have similar functionality as existing libraries of commands
- Should be relatively lightweight (similar to current interfaces) in terms of system resource usage
- Should be able to be run natively or be easily accessible from a Microsoft Windows operating system as this is the most common operating system used by our target user base.

Usability Requirements

- Must be easier to use and learn than a regular command-line interface
- Provide features that will make it easier to use
- Error correction or suggestions on incorrect input / Helpful, specific error messages when required.
- Easily “undo” commands if a mistake is made (e.g. rm’ing the wrong batch of files)

PART III: Short Form Creative Brief

Project Objective

Our project objective is to improve parts of the command-line interface, including the learning curve, design, and other quality of life features so that more users would be inclined to use them. We are aiming to enhance the experience for users that are relatively new with command-line interfaces but familiar with technology and computers. The primary focus is to flatten the learning curve and allow users to have an easier time familiarizing themselves with CLIs. Through our design, new users will be able to learn more quickly and effectively, becoming more likely to consider CLIs as a valid alternative or supplement to graphical interfaces.

Key Personas

One of our key personas is Arthur, a computer science student who has finished the first two years of his program. He enjoys coding and understands key programming concepts, but has never used a command-line interface before. Arthur isn't looking forward to working in the command-line for a systems programming class this next semester, but the syllabus requires that he learn shell scripting and work in the shell for version control. He plans to try and find a GUI program that will accomplish the same tasks.

Our other key persona is Bob Doh. Bob is a fourth year computer science student who considers himself an advanced intermediate with the command-line. Bob however finds he ends up not using the command-line for long periods of time at a time and often forgets commands or argument order between uses. Using the man pages is more time consuming and annoying to Bob than he wishes, however it proves to be the easiest method for him to get back into using the command-line smoothly.

Key Scenarios

One scenario is of Arthur and his initial experiences with a CLI. After fumbling with the CLI in his first lab of his systems course, Arthur finds that he has difficulty remembering command names, and the ordering of additional arguments. On top of this, he finds it difficult to find new commands to use, or if he is even able to accomplish certain tasks given the commands in his repertoire. Arthur does not know what resources he should use to learn to use the shell, as Googling yields varied results which are not always relevant and the textbook is dense. Despite the learning curve, Arthur is convinced he can become much faster by using the command-line, though his productivity has actually dropped since he started using it.

In another scenario, Arthur is trying to rename his files, and move them to another folder. He searches for the command on Google once again, but is confused as to why renaming and moving are listed as both the same command. He uses the man page in an attempt to figure out the syntax, but is overwhelmed by the wall of text. He knows that he can accomplish the same task with a GUI in a matter of seconds, and so he becomes increasingly frustrated, starting to question the effectiveness of the man pages, and the CLI itself.

The last scenario for Arthur includes commands not native to the shell, but other software. Arthur is trying to pull a git repository for the first time, but is having a hard time because of merge conflicts and difficulty understanding the commands that he needs to input. His professor has given him a help sheet for using the command-line, including information on basic file navigation and `man`. However, the help sheet doesn't include much information on git. He looks at the git manpage, but after a few minutes of looking through commands, can't find how to pull a repository.

Our scenario for Bob features his experience using command-lines. Bob wants to automatically manage file names of things dumped into a shared network file system. He knows from his experience that he can do that with `incron`, but can't remember exactly how. Bob reads the man page, sets up an `incrontab` user table for a custom renaming script, and feels good about his automatically managed files.

Key Principles

We have several key principles that we believe are important, and aim to ensure as we are designing. One principle is simplicity and clarity without sacrificing usability – i.e., the design should improve on making the CLI simpler to use, without affecting the power and flexibility of CLIs. We are looking to build on the CLI, and not take away features that are present. Another principle we have is to allow users to build intuition naturally, on their own – we want to decrease the reliance on outside sources and keep the work done inside the command-line itself.

PART IV: Experience Map

User Process:
Open Command Line -> Orient with interface -> Determine task needed to be done -> Enter commands to complete task -> Exit

Entice:

- Powerful, quick and flexible alternative to GUIs
- Using a CLI potentially makes users cool while using them
- Prospect of an easier time learning to use CLIs for new users, especially students with little experience using CLIs
- Helps users along the way as they are using it
- New and improved interface, with more intuitive features and usage, over a regular CLI

Enter:

- Prompt, waiting for user command input
- Text only, with responsive output
- Introduction screens, brief tutorial to CLIs for new users, and skippable for experienced users
- Guides and orients all users to features that are commonly used, as well as additional key features that have been added
- Similar look and feel to a regular CLI so users feel the same when using the new CLI

Engage:

- Quick feedback and output from commands
- Short command names, making quick typing advantageous
- Clear and detailed information being displayed
- Organized and easier to sift through text
- Helpful messages and tips
- Less intimidating messages when mistakes are made
- Unobtrusive markers and sections that indicate things such as command history, command usage, etc.
- Keeps usage in the command-line, and offers help

Exit:

- Messages upon successful command input, and forgiving error messages
- May prompt user for confirmation depending on command-line
- Simply closes

Extend:

- Quick, and powerful experience completing tasks, ease of completion for specific jobs, leading to repeat usage
- Remind users of the features of the new CLI and some advantages they may have encountered
- Possible tips and tricks that users may find useful next time they use the new CLI