Chapter 6

Related Work

6.1 Design Case Studies

6.1.1 VCSs and Conceptual Design

Santiago Perez de Rosso used Conceptual Design Theory and applied it to version control systems. In his paper, *Purposes, Concepts, and Misfits in Git*, he studied Git to understand why it seems to fall short of users' expectations [3]. He analyzes those issues by using Conceptual Design to explain Git's design issues as operational misfits of its underlying concepts. This paper focuses on fixing Git's specific issues, while Snapstore is a more general solution to version control systems.

This study of Git [2] goes on to enumerate a set of purposes for version control systems. These purposes cover all of the existing concepts in every version control system and were used in the creation of Snapstore. They provide the benefit of allowing the designer to simplify some concepts and remove others; this makes the entire system easier to model cognitively.

Xiao Zhang performed another conceptual analysis case study, this time focusing on Dropbox[8]. She first researched and polled users for the areas of Dropbox they find most confusing. She then used Conceptual Design Theory to find the operational misfits that caused this confusion. The result was a remade conceptual model of Dropbox, one that was cleaner and easier to understand. Again, this study focuses

on one file sharing system, while Snapstore is a more general solution.

6.1.2 Other Design Theories

Other design theories and tools exist such as the cognitive dimensions framework [4]. This framework asseses a design and shows cognitive consequences for various design decisions. It is used as tool early and often throughout the design process to evaluate design decisions that have been made. Conceptual theory, employed by Snapstore, is not a tool used to look back at decisions or systems for evaluation [5], it is a tool that should be used from the very beginning preemptively to guide decisions.

Heuristics for user interface design have been used often[6]. These are general rules of thumb, not specific guidelines. Conceptual theory, on the other hand, has strong rules that cannot be broken, and it can be used in all dimensions of software design. Snapstore benefits from conceptual design guiding both its back-end and front-end development.

Requirements engineering [7] is a design theory that pertains to an entire system, much like conceptual design theory. However, requirements engineering chooses whichever conceptual framework is most convenient for its designers at the time of design. This leniency leads to inconsistent designs. Snapstore's conceptual design was a given before the design process started.

6.2 Version Control and File Syncing Tools

Using the knowledge from the Git study and the resulting version control purposes, *Gitless*[2] was created. *Gitless* is an experimental version control system, designed to fix the operational misfits of Git. *Gitless* is built on top of Git and fulfills all the purposes of a version control system, while fixing the conceptual design flaws found in Git. Snapstore, on the other hand, is build from scratch.

Before the conceptual design of Snapstore, we took a high level look at the version control and file sync space. Various systems such as Dropbox, Google Drive, Git, Mercurial and more were studied from a user's perspective to see how they accomplished

various tasks that Snapstore would cover.

The acts of grouping changes and recording coherent points in development is muddled by many of these systems. Continuous saving is supported in systems like Google Drive and Dropbox, but they do not allow a user to record coherent points and group changes. Google Drive and Dropbox users can leave comments on changes but cannot group or label changes that exist across multiple files. Snapstore accomplishes this with the group and tag concept. Git and Mercurial combine the acts of grouping changes together and recording coherent points. Git, for instance, combines the commit with the act of labeling the commit. Snapstore decouples these purposes by allowing users the granularity of a single snapshot and by allowing them to later group and tag those groups.

Creating branches and copies of files varies greatly between version control systems and file syncing systems. Git allows a user to create a branch, a fork, or a clone. This array of options is simplified to just a branch in Snapstore. File syncing services allow copies to made of individual files, but these files must be manually merged later. Snapstore allows branches to be made from existing branches so that later their histories can be merged together.

The act of merging is not well supported by many file syncing systems. Google Drive preserves every edit made on a shared document through a process called operational transformation. When a conflict arises, a newline is inserted. This is fine for some text documents, but it would break code. Dropbox does not allow files to be merged at all. Snapstore simplifies the shared document by keeping the line of development perfectly linear, so as not to adversely affect white space sensitive documents. It also allows file merging between branches to give it the power of a version control system.

The file model of these systems is typically file name dependent. In Git, if a user renames a file without using the "git" command, Git will see that as deleting an old file and creating a new one. It is similarly handled in Gitless. This limits the history of commits and history of the file. Dropbox is similar. Renaming a file offline will subsequently upload an entirely new file with a new history. Snapstore achieves a

seemless history by watching the user's filesystem for name changes. It is able to log renames to files as rename snapshots and keep the snapshot history consistent.

Finally, while file syncing systems typically are able to save without a direct command from the user, version control systems cannot. Git and Mercurial require explicit commands to do so. The push/pull model of Git is unnecessary. Snapstore automatically pushes out and pulls in changes to a branch.