마우스를 이용한 Command 입력 git GUI tool 개발을 위한 모델

박성혁*, 강주호*, 주현석*, 황성진*, 이창훈*1)
*서울과학기술대학교 컴퓨터공학과
e-mail: nut982451@gmail.com

A model on git GUI tool development Using mouse for command input

Seonghyuk Park*, Juho Kang*, Hyeonseok Ju*, Seongjin Hwang*, Changhoon Lee*1)

*Computer Science and Engineering, SEOUL NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

요 약

git에 대한 관심이 증가하고 있는 요즘 각종 업계에서 Version Control System을 사용하는 인원들은 git을 svn보다 복잡하다고 생각한다. 물론, 전문적인 소프트웨어 개발자라면 컴퓨터 시스템에 대한 이해가 높기 때문에 이에 별로 구애받지 않을 수도 있다. 하지만, 그 외 디자이너, 학생과 같이 전문 개발자가 아닌 인원들은 git을 이해하기 어려워하고 있다. git을 처음 접하는 인원들이 여려움을 느끼는 이유는 개념의 이해와 명령어 사용이었는데, 이들에게 git 사용을 보다 편안하게하기 위한 git GUI tool 개발을 위한 모델을 제안을 한다.

1. 서론

최근 대다수의 공학 관련 회사에서는 VCS (Version Control System)을 사용한다. 그 이유는 하나의 일에 많은 분야의 사람들이 같이 작업을 해야하는 경우가 많기 때문이다. 특히 컴퓨터 소프트웨어 개발 업종에 일하는 인원들은 이 VCS를 더욱 더 많이 접하게 된다. VCS는 버전관리 시스템이라고 하는데 동일한 정보에 대한 여러 버전을 관리하는 것을 말한다. 많은 소프트웨어 개발업체에서는 VCS의 한 종류인 Subversion(이하 svn)을 많이 사용하고 있는데 최근 들어 git이라는 VCS가 많은 관심을 받고 있다.

많은 사람들이 git에 대한 관심이 증가함에 따라 많은 인원들이 svn을 사용하는 대신 git을 사용하는 사람들이 증가하고 있다. 하지만 그렇지 않은 사람들도 많다. 그러한 이유는 git의 개념이 보다 복잡하고 사용 tool의 종류가 적다는 것에 있다. 또한 처음 접하는 사람들도 git의 개념적 어려움과 각종 명령어들 사용에 대해 어려움을 느껴 쉽게 사용하지 못한다. 개발 모델 이전 단계인 현재 'Grigit!'은 Graph에 나타난 Commit 선택과 버튼 입력을 통한 repository 변화를 실시간으로 Graph로 표현 가능하다.

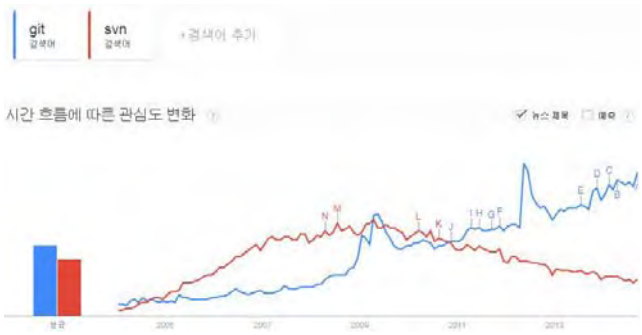
본 논문에서 제안하는 모델은 명령어를 통한 Graph 확인이 아닌 Graph에 나타난 Commit 드래그를 통한 Command 자동 입력이다. 또한 이 변화를 실시간으로 확인할 수 있는 git GUI tool이다. 본 논문에서 제안된 모델을 사용 한다면, git을 처음 접하는 사람과 git tool에 어려움을 느끼는 사람들 모두 조금 더 쉽게 git을 이용할 수 있다.

본 논문에서 2장에서는 git과 svn에 대한 분석을 하고 3장에서는 'Grigit!'의 구현과 개발 현황을 알아보고 제안 모델과 적용 방법을 설명한다.

2. 관련 연구

많은 개발자들이 svn을 사용하고 있다. 정확히 말하면 많은 개발 회사에서 svn을 사용하고 있는 실정이다. 하지만 최근 들어 많은 개발자들 사이에서 프로그램 등의 소스 코드 관리를 위한 분산 버전 관리 시스템의 일종인 git에 대한 관심이 증가하고 있다.

1) 교신저자

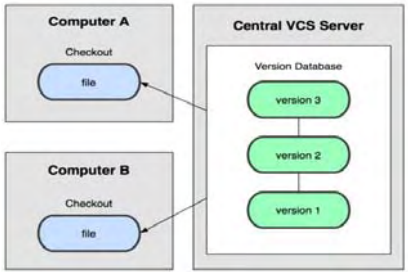


(그림 1) Google Trends에서의 git과 svn 검색률 변화 [1]

Google Trands를 통한 git과 svn의 검색률 Graph((그림 1) 참고)에서 볼 수 있듯이 많은 기업체에서 자리를 잡고 있던 svn에 비해 git의 검색률이 월등하게 증가하고 있는 모습을 볼 수 있다.

2.1. Subversion(svn)

svn은 중앙집중식 버전 관리 시스템(Centralized Version Control System; CVCS)의 일종이다. CVCS란 중앙 서버를 두고 다수의 클라이언트에서 파일을 보내는 방식이다[2].



(그림 2) Centralized Version Control System Diagram [2]

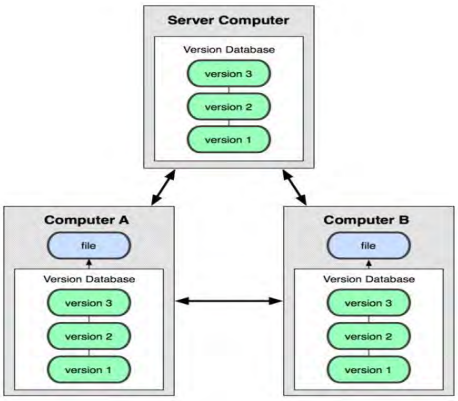
CVCS의 장점은 다른 사람들이 무엇을 하고 있는지 누구나 알 수 있고, 관리자는 누가 무엇을 할 수 있는지 관리를 할 수 있다는 것이다. 또한 개념적으로 쉬워 처음 접하는 사람들도 마음 편히 다가갈 수 있다. 다양하고 사용하기 편한 tool 들이 존재하고 기업의 시각에서는 CVCS를 통한 관리의 이점들이 존재하므로 아직까지 많은 기업에서 svn을 사용하고 있다[2][4].

2.2. git

git은 분산 버전 관리 시스템(Distributed Version Control System; DVCS)의 일종이다.

DVCS는 CVCS와 다르게 클라이언트가 파일들의 마지막 스냅샷을 가져오는 대신 저장소(repository)를 보유하고 있다. 즉, 각 local repository에 파일들의 버전(branch)을 관리하다가 중앙 서버에 기록을 하는 방식이다[3].

이를 통해 CVCS와 다르게 항상 중앙 서버로 업로드할 필요 없이 계속해서 기록된 repository를 통해 읽어오거나 local로 버전관리를 할 수 있다[2][4].



(그림 3) Distributed Version Control System Diagram [2]

2.3. Subversion과 git의 차이점

앞에 설명된 Subversion과 git의 가장 큰 차이점은 앞서 나왔던 CVCS와 DVCS의 차이점으로 저장 방식의 차이이다.

CVCS에는 치명적인 단점이 존재하는데 그것은 바로 중앙 서버 문제 발생 시 모든 클라이언트 유저들에게도 모든 것이 잘못 된다는 것이다. 문제 발생 시 서버가 복구 될 때까지 다른 사람과의 협업도, 진행중이던 작업을 버전 관리하는 것도 불가능해 진다.

이런 문제점을 해결한 것이 바로 DVCS이다. 각 local repository를 통해 중앙서버에 문제가 생겨도 각자 맡은 부분을 진행할 수 있고 계속 기록 관리를 할 수 있다. 또한 복구가 되었을 때 각 자의 repository의 내용들을 합쳐 프로젝트를 쉽게 이어 갈 수 있다. 이 부분이 DVCS의 가장 큰 장점이고 svn을 사용하는 인원들이 git에 관심을 가지는 이유 이다[6][7][8].

2.4. Version Control System(VCS) tools.

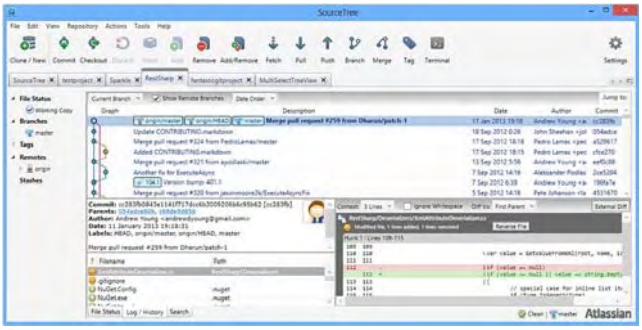
VCS의 종류는 앞서 나왔던 svn과 git 이외에도 CVS, Perforce, Mecurial, Bazaar, Darcs 등 많은 종류가 있는데 소프트웨어마다 이를 쉽게 사용할 수 있는 각종 tool이 존재한다. git을 처음 접하는 사람들이 겪는 어려움은 git의 개념이 svn에 비해 복잡하고, 이를 쉽게 활용할 수 있는 tool이 svn에 비해 적다는 것이다.

VCS를 사용하는 대상자들 중 현업에서 개발자로 일하는 사람들만이 아니다. 그래픽 디자이너와 같이 개발이 목적이 아닌 인원들과 각종 tool에 익숙치 않은 학생들도 많기 때문에 쉽게 접근할 수

있고 쉽게 사용할 수 있는 tool을 개발하려고 한다.

3. 제안 모델

본 논문에서 제안하는 git GUI tool 모델은 학생이나 디자이너같이 전문적인 현업 개발자들이 아닌, git을 처음 접하는 인원들에 대해 조금 더 git의 개념을 쉽게 이해하고 사용 할 수 있게 하기 위해서이다.



(그림 4) GUI tool Sourcetree [9]

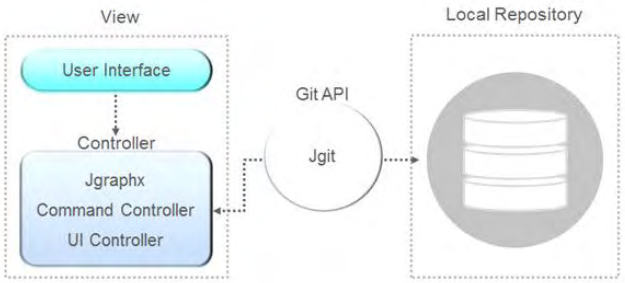
본 논문에서 제안하려는 tool의 목표는 Sourcetree ((그림 4) 참고)와 같이 Command를 통해 진행되는 git이 아닌, 마우스를 통한 Command 입력이다.

3.1. 개발 현황

'Grigit!'은 Graph로 표현된 Commit을 선택하고 Command 버튼을 이용해 실시간 Graph 변화를 파악 할 수있는 GUI tool이다. 'Grigit!'에는 Commit 선택과 실시간 Graph 변화 기능은 구현되어있다.

(그림 5) GUI tool 'Grigit!'

'Grigit!'의 마우스를 이용한 Commit 선택과 버튼을 통한 Command 입력은 Jgraphx와 버튼 View 그리고 Jgit을 통한 API의 상호작용이 원활이 이루어져야 가능하다. 또한, Command 입력 창에서의 Jgit API를 통한 repository변화를 Jgraphx View에 바로 나타낼 수 있어야 한다.

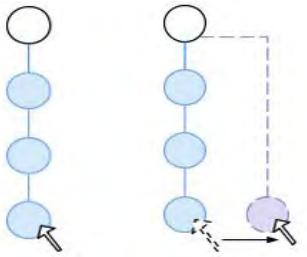


(그림 6) 시스템 구성도

시스템 구성도((그림 6) 참고)를 보면 View의 GUI를 통해 액티비티 발생 시 각 Controller들을 통해 메시지를 git API로 보내 메시지에 따라 필요 자료를 Local Repository에 기록하고 가져오는 방식으로 구성된다. 또한 Local Repository에서 git API를 통해 읽어 들인 정보는 View를 통해 나타나게 된다.

3.2. 제안 모델

본 논문에서 제안하는 모델은 'Grigit!'에서 나아가 Commit 선택과 버튼 입력이 아닌 Commit 드래그를 통한 자동 Command 입력이다.



(그림 7) 마우스 커서를 이용한 Branch 생성

(그림 7)과 같은 마우스 드래그를 통한 Command 입력은 각 Command 별 마우스 입력이 구분 가능하여야 한다. 마우스 좌표 인식만으로는 모든 Command에 대한 입력을 구분하기에는 어려움이 따른다.

<표 1> Mouse Control을 이용한 주요 Command 처리 방법.

Command	Mouse Control
Commit	Drag and drop same line.
Branch	Head drag and drop empty space.
Rebase	Commit drag and drop another line.
Cherry-Pick	Commit select and drag.
Merge	Commit drag and drop on commit.
Tag	Double Click.

하지만, <표 1>에서 제시한 방법으로 Commit, Branch, Rebase, Cherry-Pick, Merge, Tag와 같은 주요 Command는 충분히 표현이 가능하다.

'Grigit!'에 본 논문의 제안 모델을 적용시켜 손으로 쉽게 각종 Command를 다루고 실시간으로 Graph 변화를 눈으로 확인 할 수 다면 git을 사용하는 사람들이 손쉽게 git에 접근 가능할 것이다. 또한, 학생들의 git 학습용으로도 사용이 용이할 것이다.

4. 결론

기존에 많은 기업체에서 사용되던 svn은 아직까지도 많이 사용되고 있다. 물론 중앙집중식 버전관리의 문제점 때문에 svn보다 git을 사용하는 인원이 증가하는 추세이다. 이런 환경에서 git을 조금 더 사용하기 쉽고 공부하기 쉬운 tool을 사용한다면 신규 사용자가 이해하기 어려운 기존의 tool들보다 조금 더 쉽게 git에 접근이 가능하다. 하지만 모든 명령어를 마우스 움직임만으로 표현하기 힘든 문제점이 있기 때문에 기존의 git Command를 Command line을 통해 100퍼센트 활용 가능하면서도 마우스 커서만으로 많은 Command를 표현 할 수 있어야한다. 이런 부분에서는 보안이 필요하다. 이러한 문제점들을 보완하고 GUI tool인 만큼 그래픽적 요소를 깔끔하게 만든다면 git의 사용이 점점 더 많아짐에 따라서 사용자들은 이 tool을 이용해 조금 더 편의를 찾게 될 것이다.

참고문헌

[1] Google Trends. (http://www.google.com/trends). (2014.09.20.).

[2] Scott Chacon. "Pro Git". Getting Started - About Version Control. (http://git-scm.com/book). (2014.08.12.).

[3] Scott Chacon. "Pro Git". Git Basics. (http://git-scm.com/book). (2014.08.12.).

[4] "Wikipedia". 버전관리. (http://ko.wikipedia.org/wiki/%EB%B2%84%EC%A0%84_%EA%B4%80%EB%A6%AC). (2014.09.11.).

[5] "GithubGist". 버전 관리 시스템 유랑기, 그리고 Git 적응기. (https://gist.github.com/benelog/2922437). (2014.09.16.).

[6] "Notes for developers". (2013.04.02.). Git 과 svn 비교 - svn 대비 git의 차별점. (http://seungzzang.blogspot.kr/2013/04/git-svn-svn-git.html). (2014.09.11.).

[7] "stackoverflow". (2012. 03. 22.). Why is Git better than Subversion?. (http://stackoverflow.com/questions/871/why-is-git-better-than-subversion) (2014.09.12.)

[8] "KiSSFLOW". (2013.10.03.). GITvsSVN - The real reason to move away from SVN. (http://kissflow.com/kissu_kissu/git-vs-svn/). (2014.09.12.)

[9] "SourceTree". (http://www.sourcetreeapp.com). (2014.09.20.)