Combacsa's SPARCS Web Seminar

SOFTWARE DEVELOPMENT #5: VERSION CONTROL SYSTEM

Version Control

Version Control System

Recall) 문서, File

- 문서
  - … 설마 모르는 사람?
- File (파일)
  - …… 설마 모르는 사람?

Version Control (버전 관리)

- 정의 (영어)
  - Management of **changes** to documents, programs, and other **information stored as computer files**.
- 번역 (한국어)
  - "관리 대상" 인 파일들의 **버전**을 관리하는 것
  - "관리대상" 인 **파일**들에 일어나는 **변화**들을 관리하는 것

Version (버전, 판)

- 정의
  - "관리 대상" 에 있는 파일이 변화를 겪을 때마다 그 파일에게 매겨지는 일련 번호

- 예를 들면
  - 국기에 대한 맹세
    - "조국과 민족의 무궁한 영광을 위하여" … 판
    - "자유롭고 정의로운 대한민국의 무궁한" … 판

    ➔ 같은 문서였지만 개정되었다. 새 "판" 이 생긴 것.

국기에 대한 맹세 Version

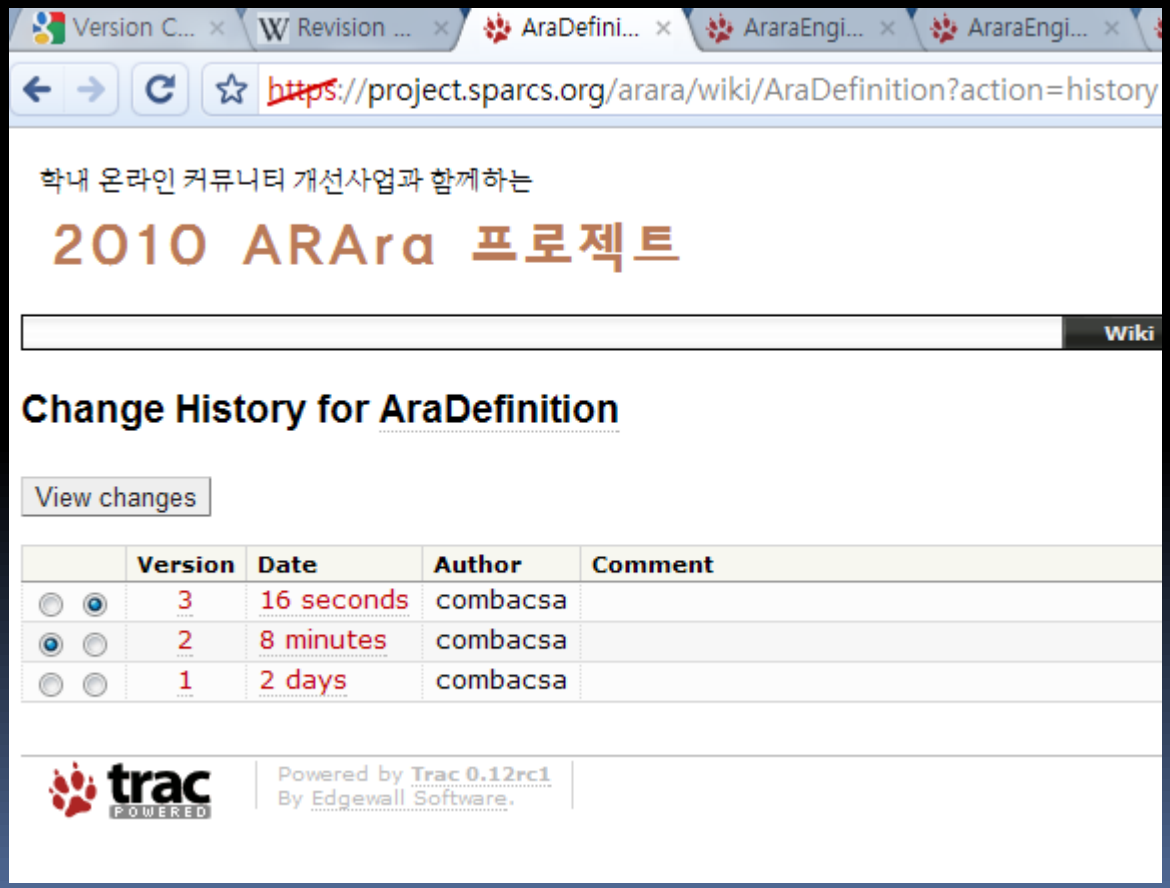
- Version 1 (1968.03)
  - 나는 자랑스런 태극기 앞에 조국의 통일과 번영을 위하여 정의와 진실로서 충성을 다할 것을 다짐합니다.
- Version 2 (1972)
  - 나는 자랑스런 태극기 앞에 조국과 민족의 무궁한 영광을 위하여 몸과 마음을 바쳐 충성을 다할 것을 굳게 다짐합니다.
- Version 3 (2007)
  - 나는 자랑스러운 태극기 앞에 자유롭고 정의로운 대한민국의 무궁한 영광을 위하여 충성을 다할 것을 굳게 다짐합니다.

다시. Version Control 이란…

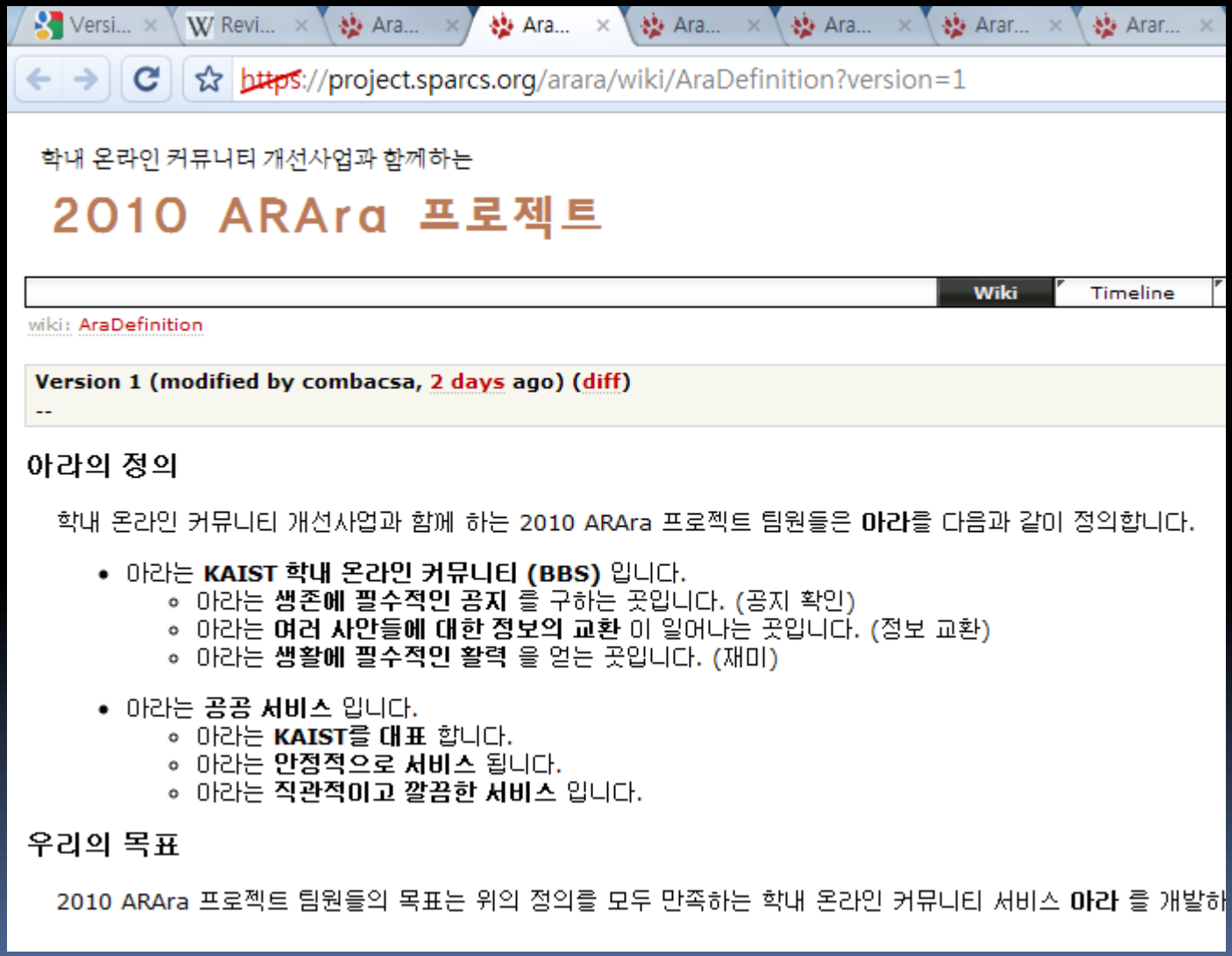
- Version 의 생성
  - 새로 생긴 관리 대상 문서의 본문을 해당 문서의 "초판" (Version 1) 으로 설정
  - 문서에 변화가 일어나면, 변화가 일어난 이후의 문서를 "새로운 판" (Version 2) 으로 설정
  - 이후 변화가 일어날 때마다 자동으로 Version 을 매겨나감
- 옛날 Version 의 열람
  - 해당 문서의 옛 Version 을 다시 볼 수 있게 해줌

Version Control 을 이용한 예

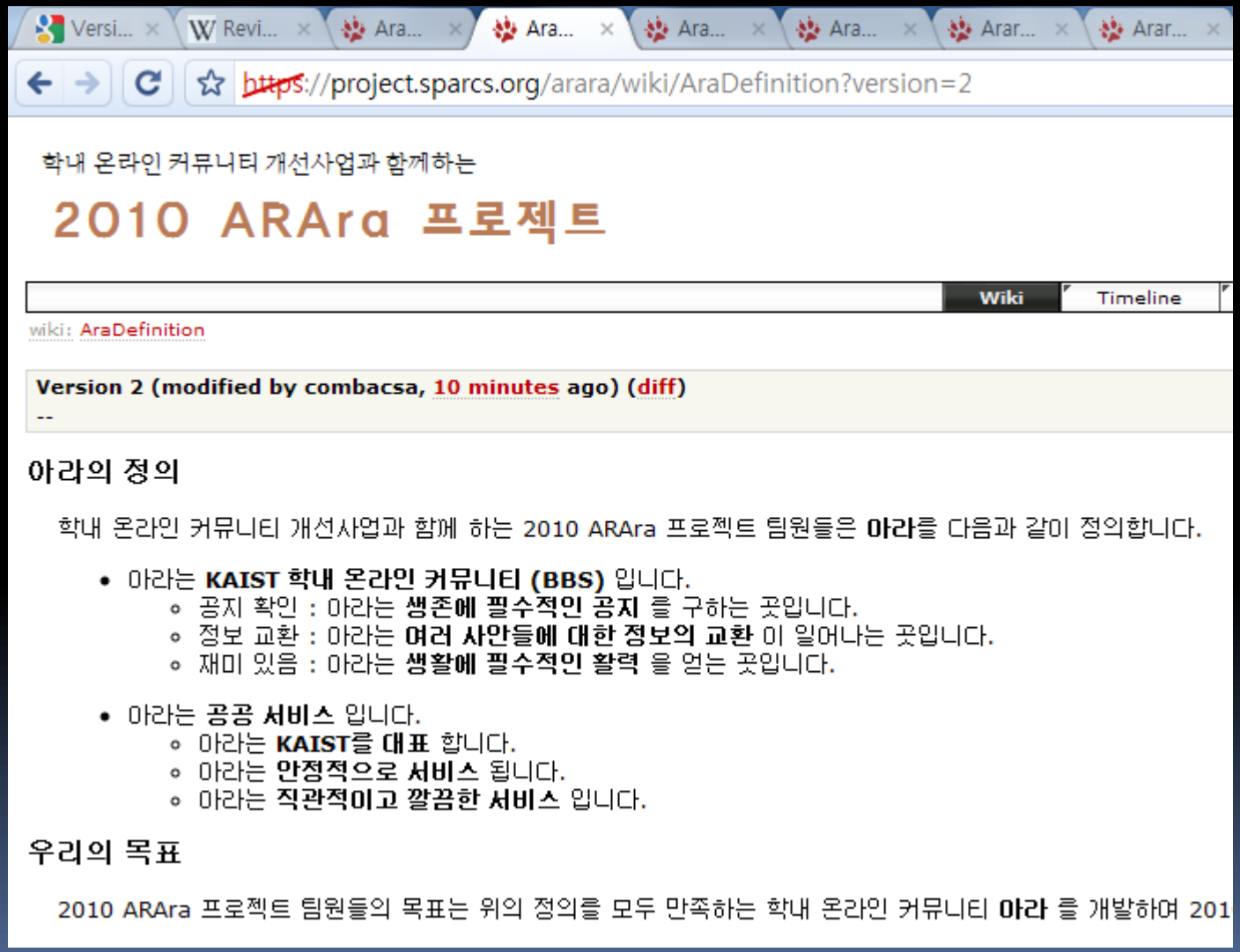
- ARAra 프로젝트의 "아라의 정의" 페이지



Version 1


https://project.sparcs.org/arara/wiki/AraDefinition?version=1
학내 온라인 커뮤니티 개선사업과 함께하는
2010 ARAra 프로젝트
Wiki
Timeline
wiki: AraDefinition
Version 1 (modified by combacsa, 2 days ago) (diff)
--
아라의 정의
학내 온라인 커뮤니티 개선사업과 함께 하는 2010 ARAra 프로젝트 팀원들은 아라를 다음과 같이 정의합니다.
아라는 KAIST 학내 온라인 커뮤니티 (BBS) 입니다.
아라는 생존에 필수적인 공지 를 구하는 곳입니다. (공지 확인)
아라는 여러 사안들에 대한 정보의 교환 이 일어나는 곳입니다. (정보 교환)
아라는 생활에 필수적인 활력 을 얻는 곳입니다. (재미)
아라는 공공 서비스 입니다.
아라는 KAIST를 대표 합니다.
아라는 안정적으로 서비스 됩니다.
아라는 직관적이고 깔끔한 서비스 입니다.
우리의 목표
2010 ARAra 프로젝트 팀원들의 목표는 위의 정의를 모두 만족하는 학내 온라인 커뮤니티 서비스 아라 를 개발하

Version 2


https://project.sparcs.org/arara/wiki/AraDefinition?version=2
학내 온라인 커뮤니티 개선사업과 함께하는
2010 ARAra 프로젝트
Wiki
Timeline
wiki: AraDefinition
Version 2 (modified by combacsa, 10 minutes ago) (diff)
--
아라의 정의
학내 온라인 커뮤니티 개선사업과 함께 하는 2010 ARAra 프로젝트 팀원들은 아라를 다음과 같이 정의합니다.
아라는 KAIST 학내 온라인 커뮤니티 (BBS) 입니다.
공지 확인 : 아라는 생존에 필수적인 공지 를 구하는 곳입니다.
정보 교환 : 아라는 여러 사안들에 대한 정보의 교환 이 일어나는 곳입니다.
재미 있음 : 아라는 생활에 필수적인 활력 을 얻는 곳입니다.
아라는 공공 서비스 입니다.
아라는 KAIST를 대표 합니다.
아라는 안정적으로 서비스 됩니다.
아라는 직관적이고 깔끔한 서비스 입니다.
우리의 목표
2010 ARAra 프로젝트 팀원들의 목표는 위의 정의를 모두 만족하는 학내 온라인 커뮤니티 아라 를 개발하여 201

Version 3


https://project.sparcs.org/arara/wiki/AraDefinition?version=3
학내 온라인 커뮤니티 개선사업과 함께하는
2010 ARAra 프로젝트
Wiki
Timeline
wiki: AraDefinition
Version 3 (modified by combacsa, 3 minutes ago) (diff)
--
아라의 역사
⇨아라의 역사
1991년 - 1999년 : pirate BBS - Eagle BBS
1999년 - 2006년 : NeoARA (Eagle BBS with News Server)
2006년 - 2009년 : WebARA with NeoARA
2009년 - 2010년 : ARAra Engine
아라의 정의
학내 온라인 커뮤니티 개선사업과 함께 하는 2010 ARAra 프로젝트 팀원들은 아라를 다음과 같이 정의합니다.
아라는 KAIST 학내 온라인 커뮤니티 (BBS) 입니다.
공지 확인 : 아라는 생존에 필수적인 공지 를 구하는 곳입니다.
정보 교환 : 아라는 여러 사안들에 대한 정보의 교환 이 일어나는 곳입니다.
재미 있음 : 아라는 생활에 필수적인 활력 을 얻는 곳입니다.
아라는 공공 서비스 입니다.

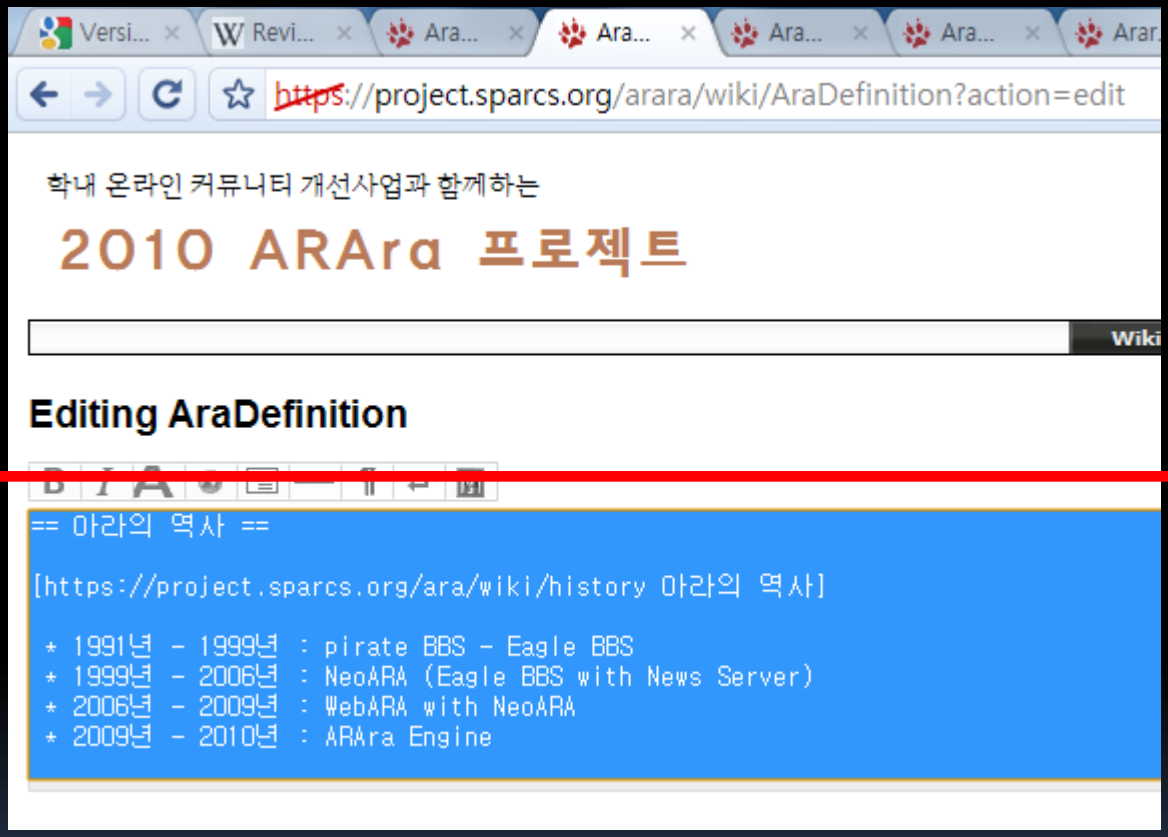
Version == 문서 발전의 기록?

- 새로운 버전의 생성
  - 문서에 계속 새로운 내용 추가
  - 전반적으로 점점 깔끔하게 정리
  - 버전이 높아질 수록 더 나은 문서가 된다
- 따라서
  - **버전 관리 == 문서 발전의 기록**
  - 문서 발전의 기록이 필요없다면
    굳이 버전 관리 할 필요가 없다는 생각이 든다

Version != 문서 발전의 기록

- 그러나
  - 문서가 어떻게 발전해나갔는지의 역사가 궁금하지 않더라도 Version 은 충분히 유용한 도구!
- 예를 들어
  - 다음과 같은 상황을 생각해 보자.
  - 한참 AraDefinition 문서를 편집하던 도중 …

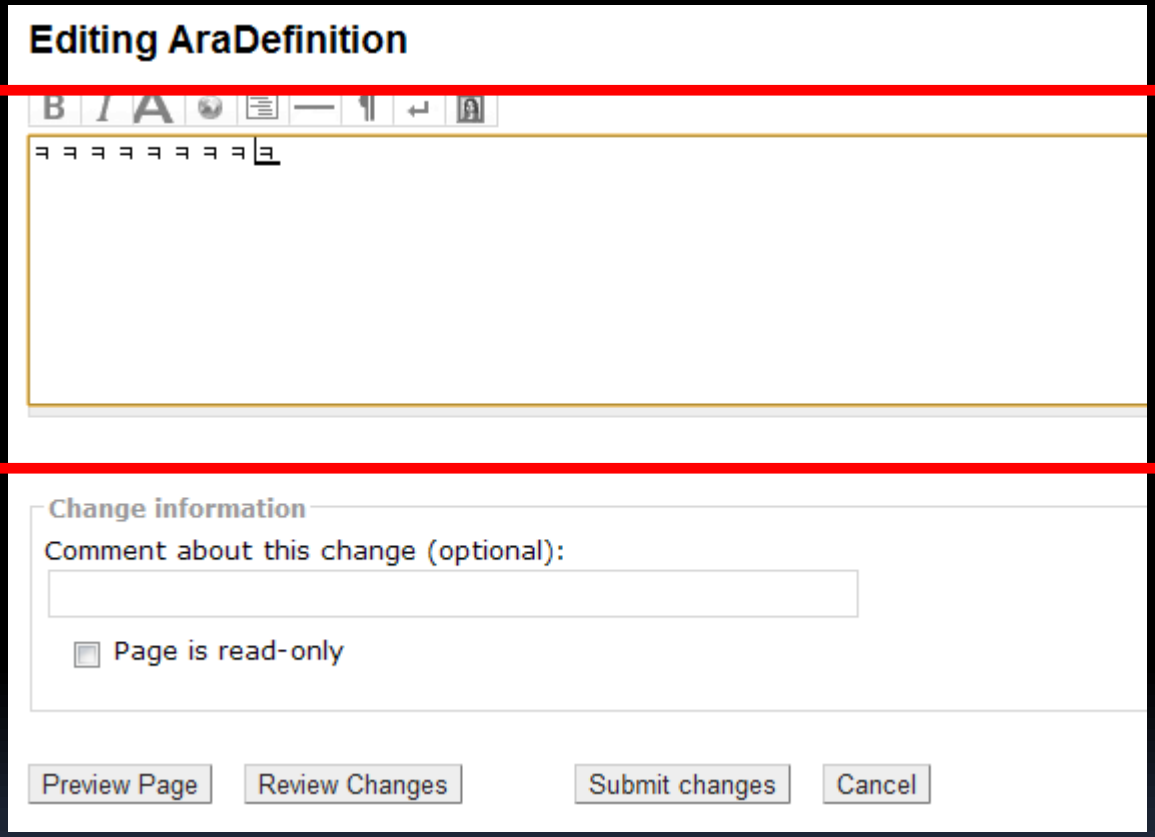
상황 1



"야 오늘도 세미나 일정이 변규홍되었대!!"

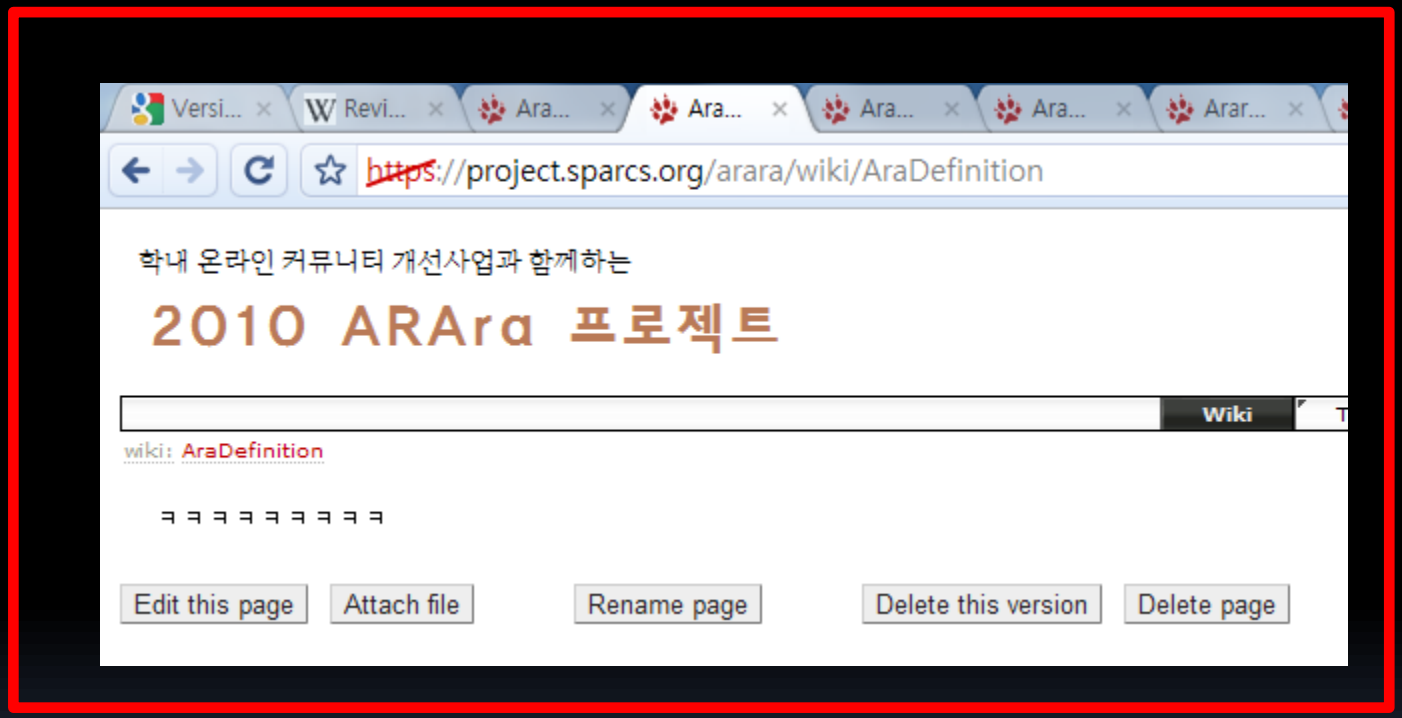
뭐? ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ

상황 2



ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ 아 웃겨 …
아 맞다, 문서 고치던 중이지. Submit Changes 해야지.

상황 3



헉 !!! 나 뭐한거임? ㅠㅠㅠㅠㅠㅠ 미쳤나봐 ㅠㅠㅠㅠ

일반화된 상황 설명

- 문서를 편집하던 중 실수로 내용을 지웠다
  - 경우 1 : 아직 편집 프로그램을 끄지 않았다
    - 대책 : Ctrl + Z (실행 취소)
  - 경우 2 : 이미 프로그램을 껐는데 다행히도 옛날에 복사해 둔 파일이 있었다
    - 대책 : 옛날에 미리 복사해 둔 파일에서 지워버린 부분을 복원한다
  - 경우 3 : 이미 프로그램을 껐는데 복사본 없ㅋ엉ㅋ

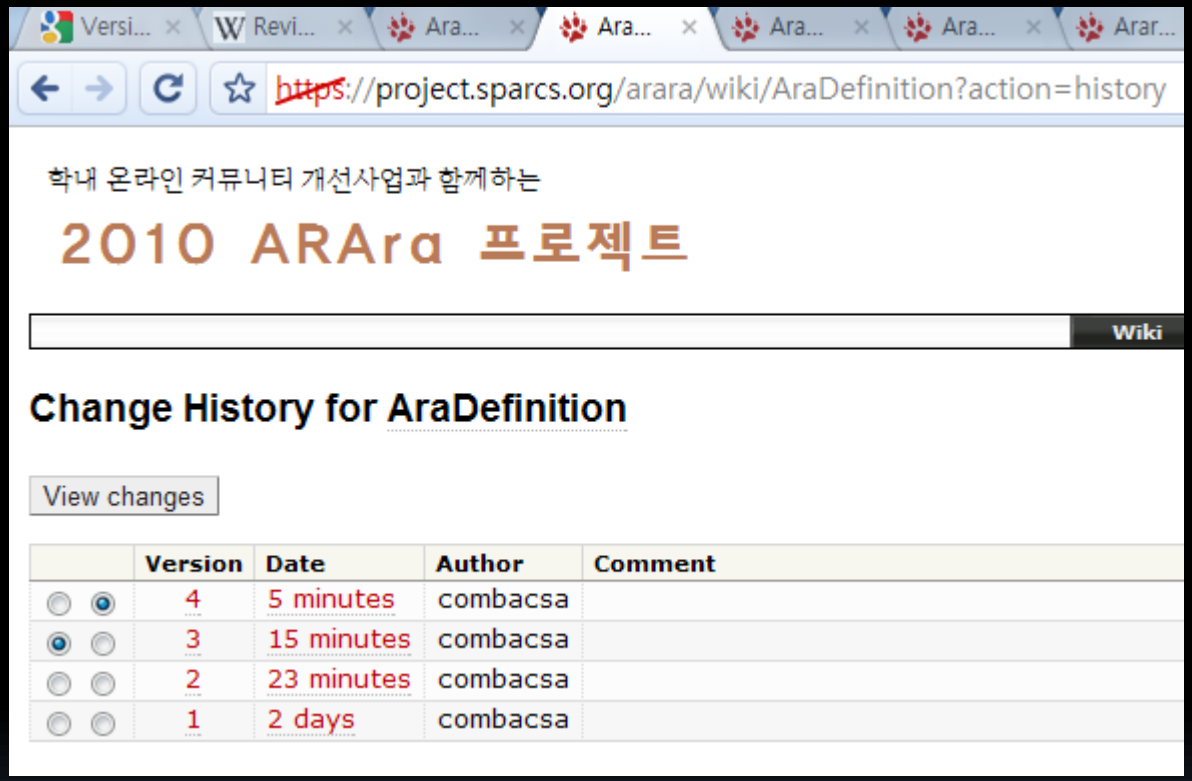
실수로 작업물 날릴 때 대책은?

- **1. 정신줄을 절대 놓지 않는다?**
  - 아무리 웃기는 일이 있어도 ㅋㅋㅋ 치지 않는다
    - *근데 인간적으로 인간이 이게 가능할 리가 없다*
- **2. 작업할 때 중간 중간에 파일 백업한다?**
  - 메일에 첨부하든 다른 폴더에 모아놓든 한다
  - 중간에 날리면 거길 뒤져서 원본을 복원해낸다
    - *근데 수작업으로 일일이 파일 복사하는 건 끔찍해.*
    - *그리고 만약 파일이 여러 개라면 …… 꺄악!!!*

2. 가 바로 버전 관리!

- 버전 관리가 있다면 아무 문제 없다
  - **작업 중간 중간에 "버전" 을 생성한다**
  - **실수로 문서의 내용을 잘못 수정했을 경우, 버전 관리되고 있었으므로 해당 문서의 옛 버전으로 되돌려버리면 끝**
- 즉 버전 관리란 단순히 "발전 기록" 이 아닌
  - 언제든지 "옛날 버전" 을 열람할 수 있도록 하는 유용한 기술이다

앞의 상황의 예



- 문서 내용을 그냥 옛 Version (3번) 으로 되돌리면 그만이다.

Summary

Version 이란 하나의 파일에 지속적으로 일어난 변화들에 대하여, 특정한 변화가 일어난 뒤의 파일의 상태를 일컫는 말이다.

Version Control 은 파일에 새로운 Version 이 생기면 이를 보관하고, 이미 저장된 Version 들의 상태로 파일을 되돌려주는 등의 역할을 통해 파일의 Version 을 관리해주는 것을 말한다.

Version Control 을 이용하면 파일을 편집하다 발생하는 다양한 실수들에 능동적으로 대처할 수 있으며, 필요시 언제든 옛날 Version 을 불러올 수 있으므로 매우 편리하다.

Version Control

Version Control System

Version Control System

- 정의
  - Version Control 을 해 주는 프로그램
- Version Control System 을 쓰는 이유
  - 복잡한 일은 버전 관리 시스템이 알아서 한다
    - 새로운 버전이 생길 때 파일 일일이 백업하기,
    - 백업하면서 버전 숫자 알아서 매겨주기 등
  - 우리는 다만 "버전 생성" 기능을 실행하면 끝!

VCS 없이 Version Control?

- 버전의 생성?
  - 새로운 버전을 만들기 위해서는 파일을 저장할 때마다 "기존의 파일" 을 따로 복사하여 적당한 이름으로 보존해야 한다
  - 또한 이 "보존" 하는 파일들을 적당한 장소에 수작업으로 모아두어야 한다
- 옛 버전으로 되돌리기?
  - 자신이 각 버전을 어떤 파일 이름으로 저장했는지 기억해 두던가 따로 적어둬야 한다

Verson Control System Rocks!

- 버전 관리 시스템을 쓰면
  - 귀찮은 영역은 프로그램이 전적으로 알아서!
- 인간이 신경써야 할 것은 오직
  - "버전 생성" 기능을 사용할 타이밍 고민 뿐!

VCS Everywhere

- Wikipedia (위키백과)
  - 누군가 고칠 때마다 편집 기록을 모조리 보관
    - 위키백과 이외의 다른 Wiki 대부분에서 제공
      - 예) 아까의 AraDefinition 문서 …
- Google Docs (Google 문서도구)
  - 문서 내용을 수정할 때마다 수정 이력 보관
    - 누가 어떻게 고쳤는지 언제든 알 수 있다

보편적인 버전 관리 방식

- **버전 관리 대상 폴더**를 설정한다
- 다음 작업 중 하나를 한다
  - **새로운 파일**을 버전 관리 대상으로 (폴더 속으로)
  - 이미 버전 관리 대상인 파일을 편집하고 **파일의 새로운 버전의 생성**을 명령
  - 관리 대상인 파일을 **버전 관리 대상에서 제외**
  - 버전 관리 대상인/이었던 **파일의 옛날 버전 열람** 또는 복원

VCS AKA

- Revision Control System (RCS)
  - Version 을 넘어 Revision 을 관리해준다
    - Revision? (뒤에 설명)
- Source Configuration Management (SCM)
  - 프로그램 Source 코드의 상황을 관리한다
    - VCS 는 SCM 의 기능의 일부분이라고 보는 견해도
    - 엄밀한 구분이 되기는 하는 모양이다

Change

- 정의
  - 버전 관리되고 있는 파일들에 일어난 변화
  - 보통 파일 / 문서 단위로 매긴다
    - 하나의 문서 내에 여러 곳이 변화한 것도 일단은 1개의 변화라고 보는 편
- Change 없으면 새 Version 도 없다
  - 바꿔 말하면 Change 가 있으면 Version 생성 가능

Changeset

- 정의
  - 버전 관리되고 있는 파일들에 일어난 여러 Change 들을 하나로 합친 것
- 예를 들어
  - 파일 1번에도 Change 가 있고
    파일 2번에도 Change 가 있을 때
    이 둘을 합쳐서 하나의 Changeset 이라고 볼 수 있다

Commit

- 정의
  - Change 혹은 Changeset 으로부터 새로운 Version 을 만들어 내는 것
- 즉
  - 변화가 일어났다고 하더라도 Commit 을 하기 **전**까지는 새로운 Version 이 생기지 않는다
    - 새로운 Version 의 생성은 명시적인 Commit 시에만!

Revision

- 정의
  - Commit 에 의해 저장소 자체에 매겨지는 Version
- 즉
  - 파일들에게는 각 파일마다의 버전이 있고
  - 저장소는 각 Changeset 에 의해 "여러 파일" 이 한꺼번에 바뀌는 경우가 있어서
    - 그러한 한꺼번에 바뀜이 저장소 입장에서는 "하나" 의 Version 의 생성이라고 볼 수 있으니까.
      - 예제를 보자

Revision 의 예

- 버전 관리중인 두 파일
  - A.txt 와 B.txt 에 대하여

- Changeset 1 : A.txt change
- Changeset 2 : A.txt change, B.txt change
- Changeset 3 : B.txt change
- 이제
  - A.txt, B.txt 에게 각각 2개의 버전이 생겼다

Revision 의 예 (Cont'd)

- 그러면 저장소 입장에서는

- Revision 1 : A.txt 버전 1
- Revision 2: A.txt 버전 2, B.txt 버전 1
- Revision 3: A.txt 버전 2, B.txt 버전 2

- 이렇게 된다!

왜 Revision 이 필요한가?

- 저장소 차원에서의 "단위 변화" 는 "파일 하나" 보다 큰 단위여야 할 수 있다!
  - 판타지소설 작가가 소설 파일들을 버전 관리한다고 치자. (챕터마다 파일을 나눠둠)
  - 주인공 이름을 바꾼다고 하자. 그럼 이 변화는 "한 파일" 에만 적용되어야 할까, 여러 파일 모두에 동시에 적용되어야 할까?
    - 당연히 모든 파일에 대해서 동시에 바뀌어야 한다! 어느 파일에는 옛날 이름을 쓰고 어느 파일에는 새 이름을 쓰면 "소설" 의 레벨에서는 문제 발생!

Revision 개념이 없는 VCS

- Changeset 개념이 없는 경우
  - "한번" 에 동시에 일어나야 하는 변화, 라는 개념을 인정하지 않으면 Changeset 개념도 없고 따라서 Revision 개념도 존재할 수 없다
- 예
  - 위키 백과사전에서는 각각의 서로 다른 항목들은 "독립"적인 항목들이므로 굳이 "여러 항목"에 걸친 변화를 고려해줘야만 할 필요는 없다고 볼 수도 있다 (수동으로 맞춰야 한다는 의미)

Version Control

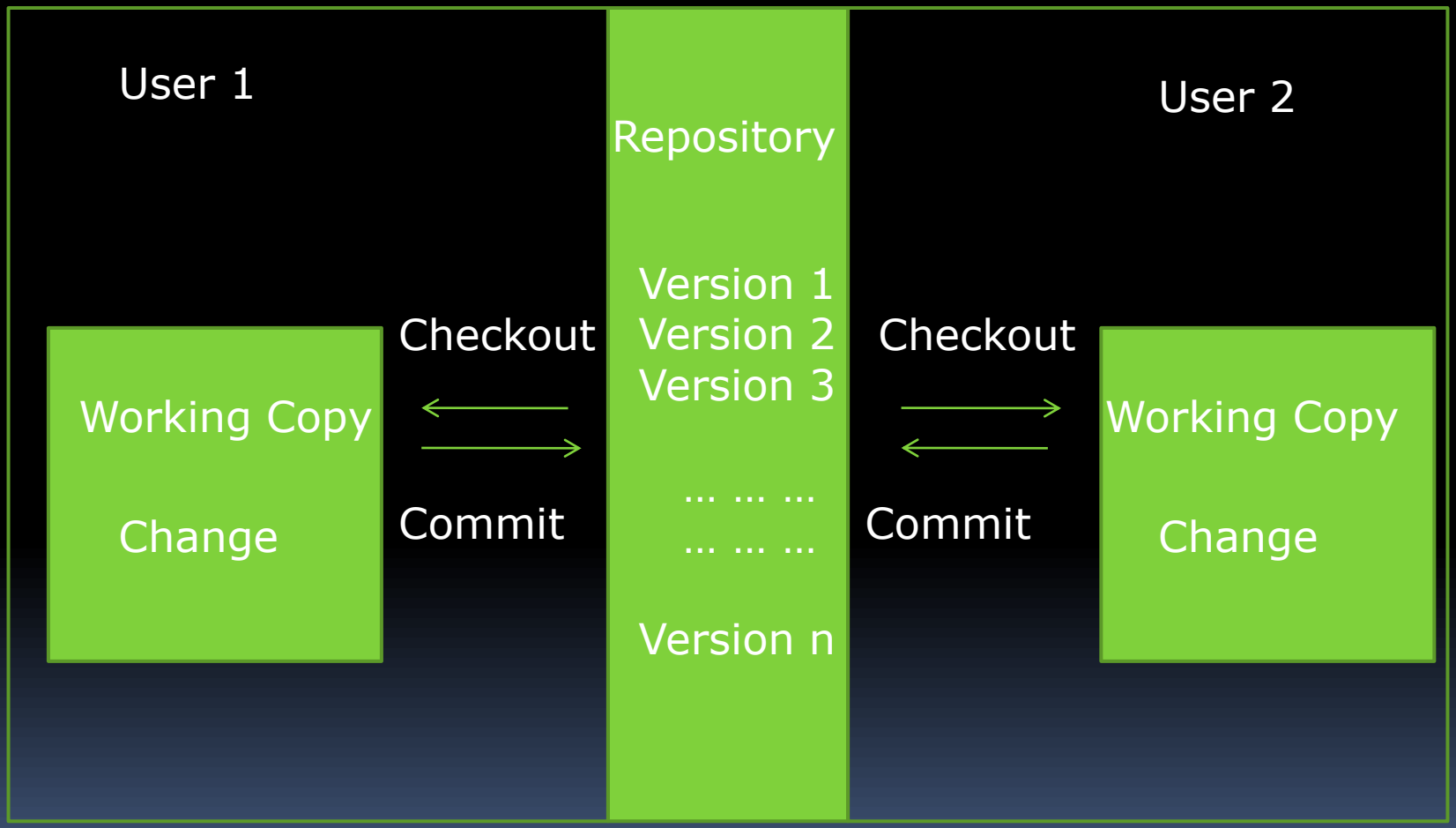
Version Control System

중앙 집중형 VCS

중앙 집중형 버전 관리 시스템

- 목적
  - 여러 사람이 공동으로 작업하면서 **하나의 버전 관리 시스템을 공유**한다
    - 즉, 여러 명이 동시에 작업하는 파일들을 하나의 버전 관리 시스템에 집어넣고, 모든 사람들이 추가하는 버전들을 모든 사람들이 공유한다!

중앙 집중형 VCS 의 작동방식



주요 용어 (1/2)

- Repository (저장소)
  - "관리 대상" 의 파일들과 파일들의 각 버전에 대한 정보가 모여 있는 폴더
- Working Copy (작업 사본)
  - 각 사용자들이 "저장소" 에 있는 파일들의 최신 버전의 복사본을 가져와서 작업하는 폴더
    - 작업 사본에서 어떤 삽질을 해도 저장소에는 영향을 바로 끼치지는 않음.
    - 따라서, 저장소는 안전하게 유지된다

VCS 주요 용어 (2/2)

- Checkout
  - 저장소에서 작업 사본을 만들어 낸다
- Checkin
  - Change 를 Repository 에 반영
    - 말하자면 Commit 이다

중앙 집중형 VCS 가 있으면

- 여러 사람이 작업하기 쉽다!
  - 각자가 일으킨 Change 를 Repository 경유하여 손쉽게 공유할 수 있다
  - 누가 일으킨 Change 때문에 문제가 발생했는지 손쉽게 추적할 수 있다
- 실수에 대응하기 쉽다!
  - 버그를 만들어냈을 때 언제든지 버그가 발생하기 이전의 소스코드로 돌아갈 수 있다

중앙 집중형 VCS 가 없다면

- 프로젝트 참여자들은 서로 각자의 컴퓨터에서 소스 코드를 편집하다가 합쳐야 하는 상황이 발생했다
  - E-Mail 을 이용해서 그때 그때 파일을 서로에게 전송하면 서로 서로 "바뀐 부분" 을 자기 소스코드에 반영하자!
    - **결과) 헷갈리기 시작한 날부터 모든게 엉망이 됨**
- **아까도 말했지만,**
  헷갈리면 말짱 황이다. 그냥 VCS 쓰자.

대표적인 VCS

- 중앙 집중형 VCS
  - **Subversion** (압도적!)
  - Perforce (Google 도 쓰는)
  - CVS (Current Version System) – 오래된!
- 분산형 VCS
  - **Mercurial** (동아리 프로젝트!)
  - Git (압도적인 지지율!)

Summary

Version Control System 은 여러 파일에 대하여 Version Control 을 쉽게 할 수 있도록 도와주는 프로그램이다.

VCS 를 통해 여러 사람이 하나의 프로젝트를 좀더 쉽고 편리하게 작업할 수 있다.

VCS 에는 중앙 집중형 VCS 와 분산형 VCS 가 있다.

대표적인 중앙 집중형 VCS 로는 Subversion, Perforce, CVS 등이 있으며 분산형 VCS 로는 Mercurial, Git 등이 있다.