

# Agile기반 ALM 프로세스 따라하기

오픈소스컨설팅 정명훈 이사



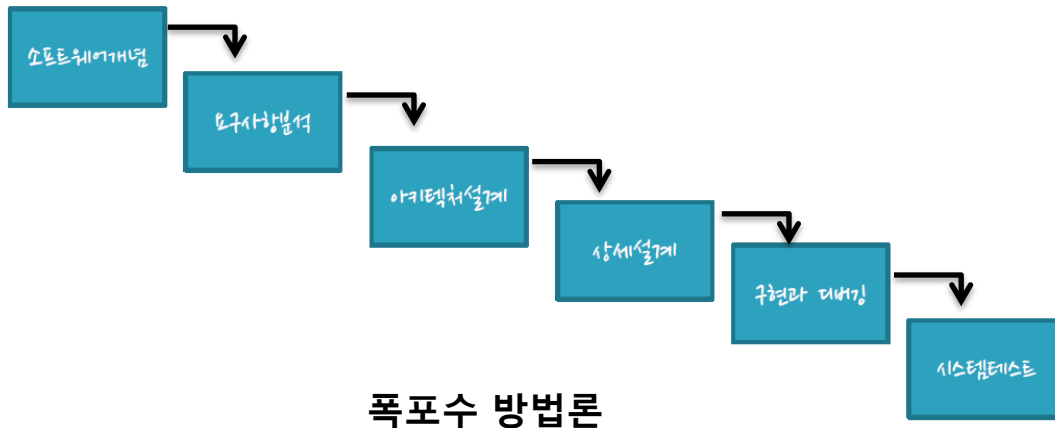


# Agile 개발 방법론

# 기존 폭포수 개발 방법론과 애자일 방법론의 특징

## ● 폭포수(Waterfall) 방법론

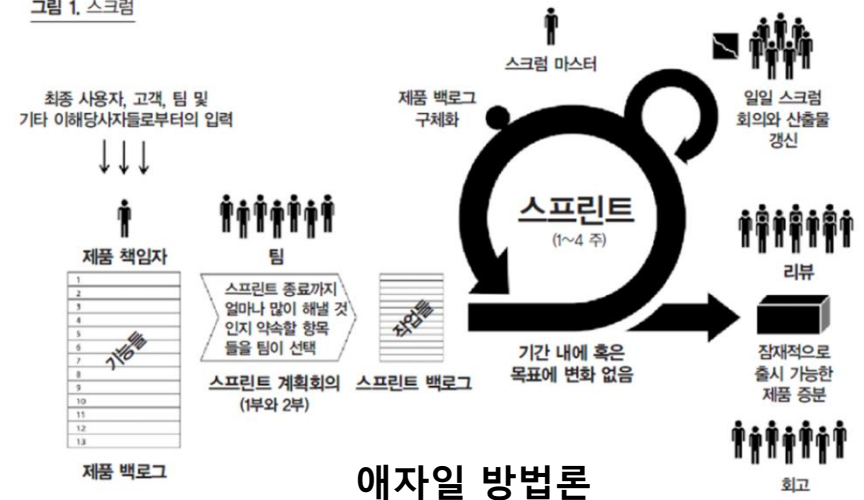
- 전통적인 대규모 프로젝트에 적합
- 프로젝트 기간을 "분석 → 설계 → 구현 → 테스트"의 주요 단계로 구분하여 각 시기 별로 해당 작업 수행
- 이 프로세스의 전제 조건은 다음 프로세스로 넘어가기 위해 **이전 단계가 완벽하게 끝나야** 한다는 것이다
- 실제로는 고객과 개발자 모두 프로젝트 진행 과정에 끊임없이 학습 하고, **학습의 결과로 "요구사항 변화"**



## ● 애자일(Agile) 방법론

- 제품/솔루션 개발 프로젝트에 적합
- 프로젝트 중에 **요구사항의 변경이나 개발자의 능력 진화(발전)를 전제로 한 변화를 수용**하는 실용주의 방법론
- 개발 과정을 짧은 조각(2~4주)으로 나누어 반복적으로 개발 - 반복적(Iterative) 개발 방법론
- **협업과 커뮤니케이션에 비중** - 스프린트 계획 회의, 일일 스크럼 회의 및 리뷰(데모) 회의
- 자동화 도구 - 협업, 개발, 빌드, 테스트 과정을 **통합된 환경(Application Lifecycle Management)에서 자동화**

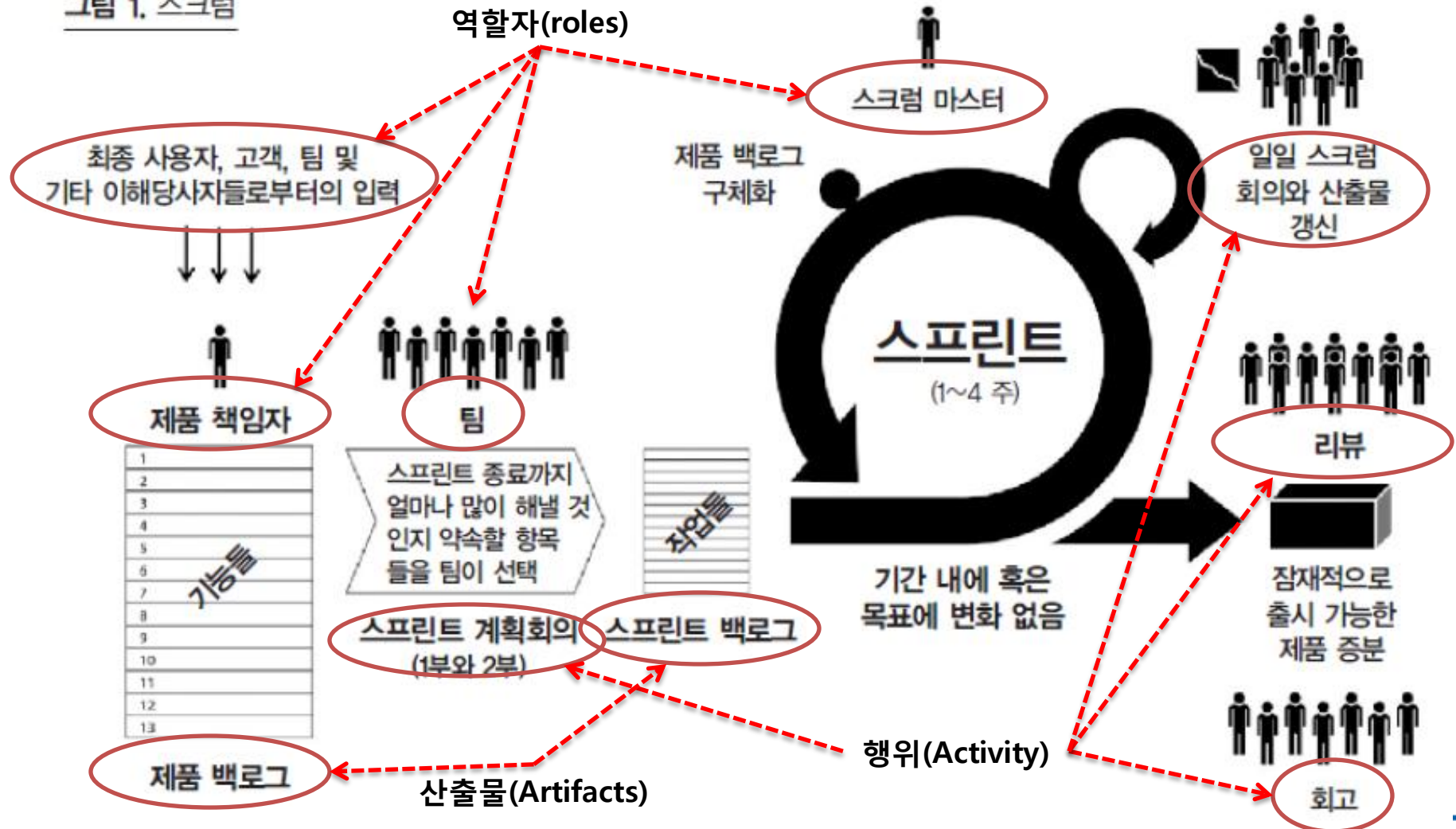
그림 1. 스크럼



# 스크럼 애자일(Scrum Agile) 개발 방법론

- 1990년대 초반부터 복잡한 제품 개발을 관리하기 위해 사용된 반복적이며 점진적인 개발 방법론
  - 스크럼은 몇 개의 반복(Iteration)으로 구성 (스프린트, Sprint라고 함)
  - 각 스프린트에는 1~4주 정도의 기간이 소요 (보통 2주 정도가 적당)

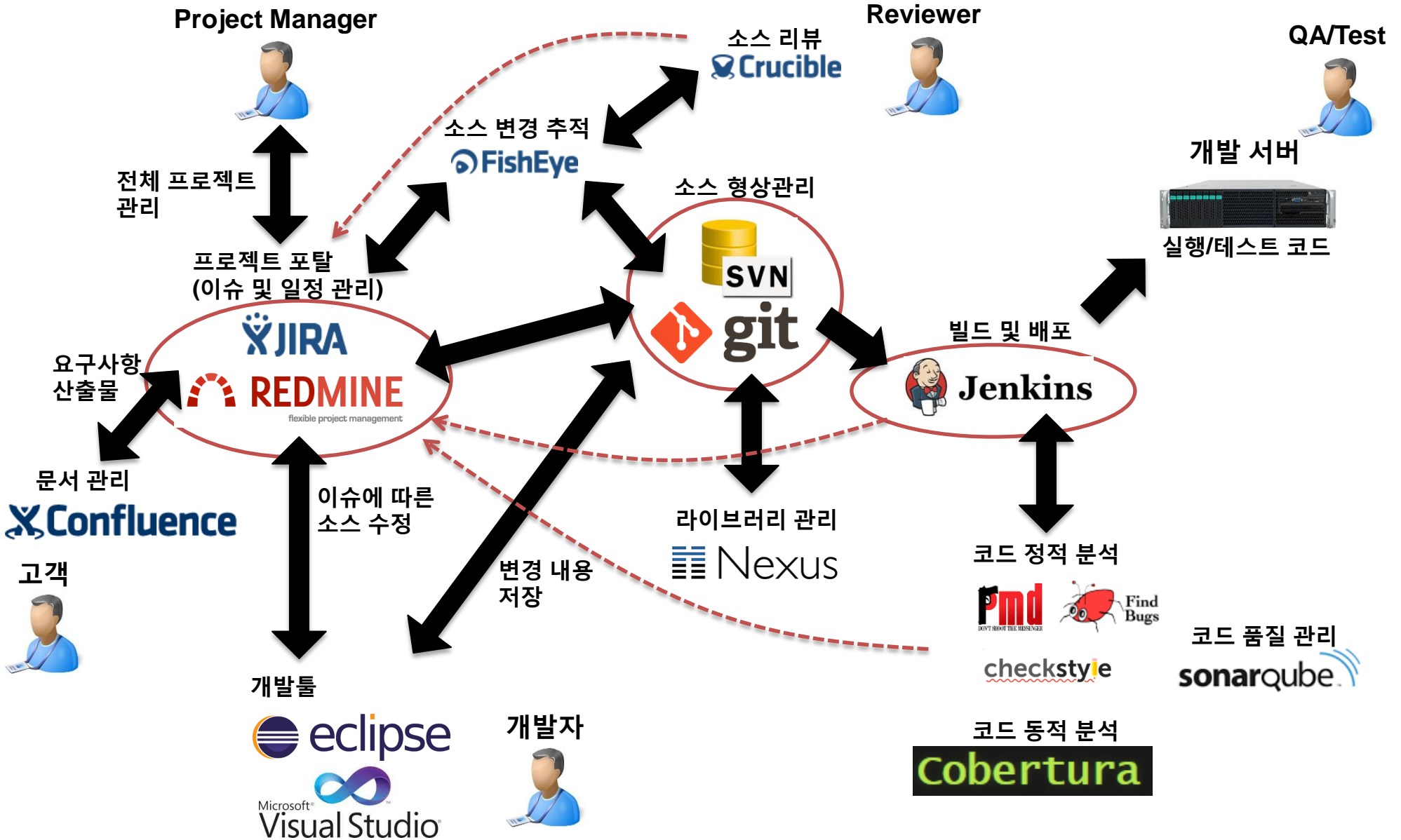
그림 1. 스크럼





# ALM 프로세스

# ALM 환경 사례




# ALM(Application Lifecycle Management) 기반 품질관리 방안

- 일관된 프로젝트 산출물(요구사항, 문서, 코드, 협업 내용 등) 관리
- 소스코드 품질 보장
- 개발 과정, 코드 품질, 이슈 및 위험도에 대한 가시성 제공

## System Dashboard

### Introduction



**Thanks for choosing JIRA.**  
Welcome to JIRA—the easy way to help your team track and deliver projects.

**Where do I start?**  
Learn more about configuring JIRA with the [JIRA 101 guide](#).  
You have no projects yet. [Create your first project](#).

### Guide for JIRA Administrators

#### Getting Started

- Create or import your first project to start tracking your work
- Create your first JIRA issue to get things done
- Build your team by adding users or inviting users
- Add your own style by customizing the look and feel

[I'm done, hide this list](#)

#### Do more with JIRA

- [Install JIRA Agile for project management. Free for 30 days!](#)
- [Install JIRA Capture, exploratory testing for agile teams. Free for 30 days!](#)

[Congratulations, you've completed everything on the list! Hide this list now.](#)

#### Administrator documentation

Browse through the [admin documentation](#) to learn more about [defining your workflows](#), [customizing issue fields and screens](#), [managing users](#) or the [time tracking features](#) of JIRA. If you are migrating from another issue tracker you might want to read our [migration guides](#).




### Assigned to Me

T	Key	Summary	P ↓
	QA-23	QA version 2.0 bacon issue fixed 2.0 - Would i get notified????	
	BULK-43	Sub-Task of Death!!!!!!	
	QA-40	would u show up in my inbox?	
	BULK-70	Description testing	

### Activity Stream

#### Your Company JIRA

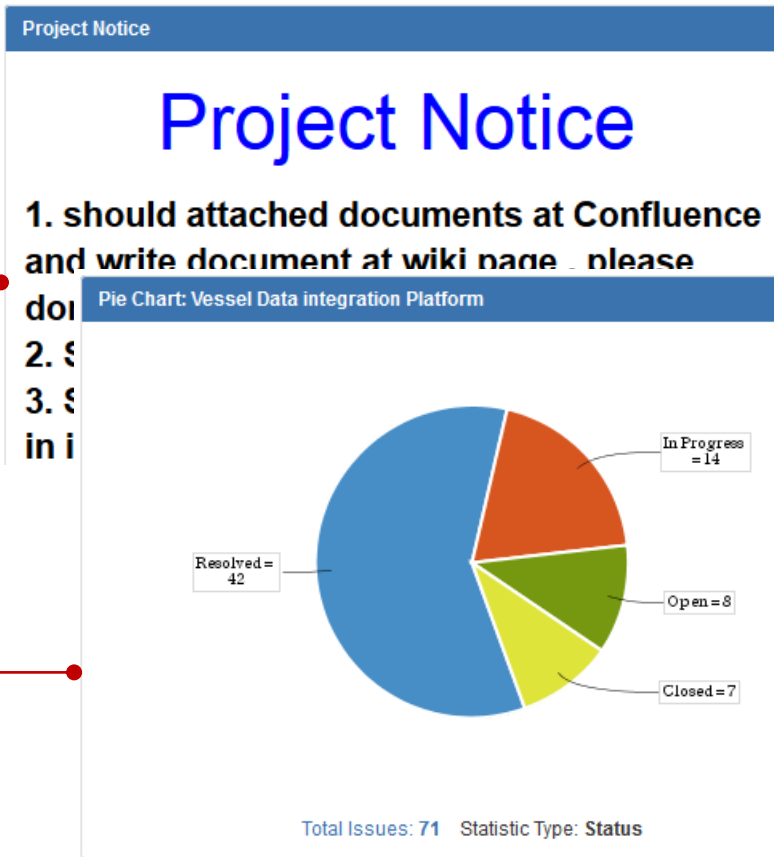
Today

-  **Audrey Cheng** commented on TST-284 - Login screen failure  
I've move this to Version Three. Only a minor issue and too late to get it into the next version. Thanks A  
Moments ago [Comment](#) [Watch](#)
-  **Kieran Williams** updated a link from TST-284 - Login screen failure to Kieran Script Five (Enterprise Tester)  
2 minutes ago [Comment](#) [Watch](#)
- Kieran Williams** updated a link from TST-284 - Login screen failure to Login screen failure (Enterprise Tester)  
2 minutes ago [Comment](#) [Watch](#)
-  **System Updated Incident 'Login screen failure'**  
2 minutes ago
-  **Audrey Cheng** updated 2 fields of TST-284 - Login screen failure  
- Added the Fix Version 'Version Three'

# ALM 시스템 소개 – 개발자 View

주요 프로젝트 공지 사항

각종 프로젝트 활동 로그



**Activity Stream**

Your Company JIRA

Today

- Audrey Cheng** commented on TST-284 - Login screen failure  
I've move this to Version Three. Only a minor issue and too late to get it into the next version. Thanks A  
Moments ago    Comment    Watch
- Kieran Williams** updated a link from TST-284 - Login screen failure to Kieran Script Five (Enterprise Tester)  
2 minutes ago    Comment    Watch
- Kieran Williams** updated a link from TST-284 - Login screen failure to Login screen failure (Enterprise Tester)  
2 minutes ago    Comment    Watch

**Assigned to Me**

T	Key	Summary	P ↓
📷	QA-23	QA version 2.0 bacon issue fixed 2.0 - Would i get notified????	🚫
📌	BULK-43	Sub-Task of Death!!!!!!!	🚫
📌	QA-40	would u show up in my inbox?	🚫
📌	BULK-70	Description testing	↑
📌	BULK-118	BULK-116 / Cross-Site Request Forgery	↑
📷	BULK-3	Unknown Unknown issue fixed 2.0	↑
📌	BULK-4	Version 1.1.1 cheese issue fixed 1.1.1why did summary vanish? saving	↑
📌	BULK-5	BULK-4 / version 2.0 bacon issue fixed 2.0	↑

주요 이슈들의 진행 상황

각 개발자에게 할당된 이슈



# ALM(Application Lifecycle Management) 기반 품질관리 방안

개발툴(이클립스)에서 소스 수정 후 Commit

**Commit**

**Enter a commit comment**  
 You can specify a new message or choose the previously entered one. Empty comments are allowed, but filling a comment message would help other people to understand the changes.

Comment  
 HHVIDIP-7 Fixed critical performance issue

Choose a previously entered comment or template:

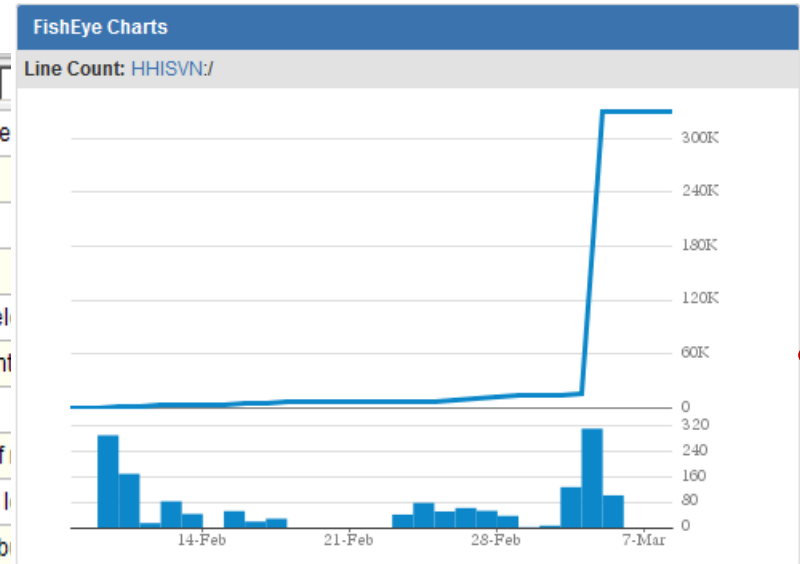
S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간
●	☀	<a href="#">PILOT-Sensor-driver-bundle</a>	14 hr - #14	1 day 14 hr - #3	23 sec
●	☀	<a href="#">PILOT-VDMConf-loader</a>	14 hr - #8	—	1 min 4 sec
●	☀	<a href="#">VAAS-Common-lib</a>	14 hr - #28	—	2 min 23 sec
●	☀☁	<a href="#">VAAS-SMTP-sender-bundle</a>	14 hr - #8	15 hr - #6	2 min 4 sec
●	☀☁	<a href="#">VAAS-Topic-listener-bundle</a>	14 hr - #24	15 hr - #21	1 min 48 sec
●	☀	<a href="#">VAAS-Udp-adapter-bundle</a>	14 hr - #30	6 days 20 hr - #7	1 min 43 sec

변경된 소스에 대한 자동 빌드

Changelog: [public:/contrib/jira/jira-fisheye-plugin/](#) Full changelog Search: [ ]

📄	tpettersen	Updating pom version to 2.1-SNAPSHOT for next development ite
📄	tpettersen	Tagged jira-fisheye-plugin-2.0
📄	tpettersen	Updated pom.xml for 2.0 release
📄	tpettersen	Updating atlassian-plugin.xml version number to 2.0
📄	tpettersen	FISH-89: Improved logging of upgrade tasks in preparation for rel
📄	tpettersen	FISH-89: Implemented fake ChangesIndexService stub to prevent
📄	tpettersen	FISH-89: Post-function upgrade now runs for all JIRA editions
📄	tpettersen	FISH-89: Post-function order is now maintained after upgrade. (if
📄	tpettersen	FISH-89: Fixed bug, log warn/info methods were logging at error l
📄	tpettersen	FISH-89: Made PerforceUpdateCreateJobFunctionTask more rob

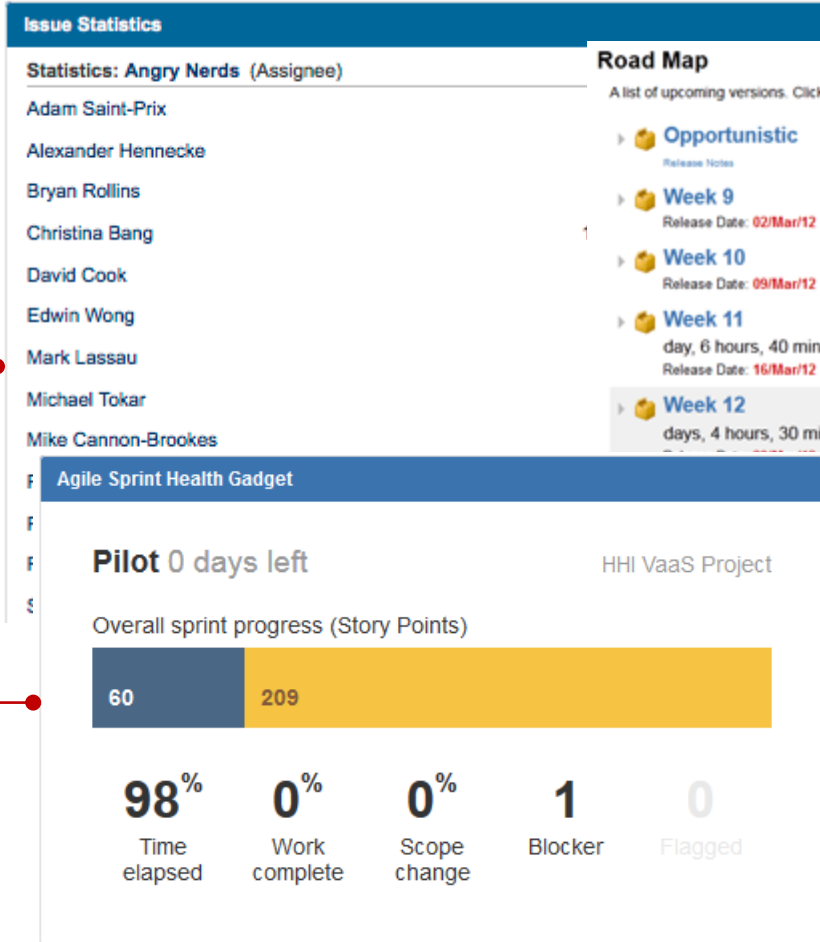
소스 수정 및 Commit 이력



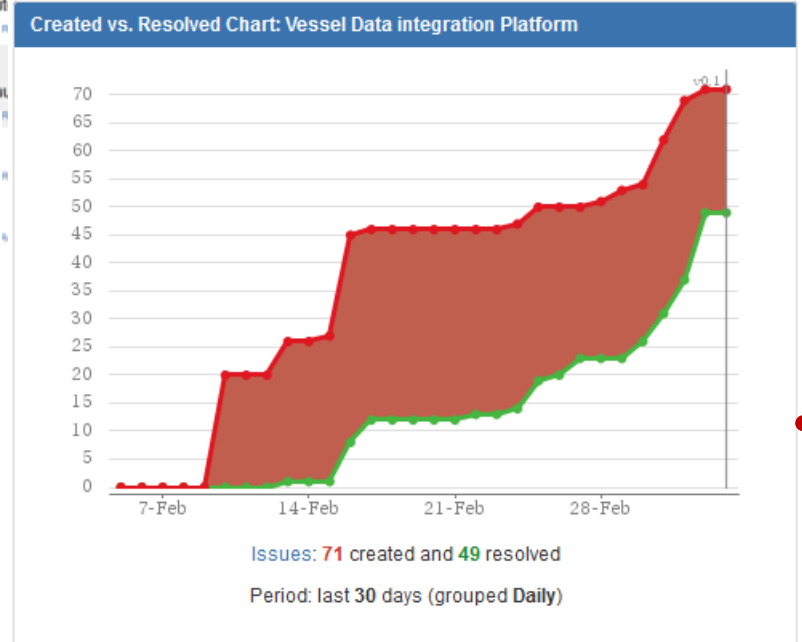
소스 증가 추이

# ALM(Application Lifecycle Management) 기반 품질관리 방안

## 개발자 별 이슈 진행 상황



## 프로젝트 주요 릴리즈 일정



## 주요 이슈들의 진행 상황

## 이슈 생성/해결 상황 그래프

# ALM 시스템 소개 – 소스에 대한 품질 관리

- 소스에 대한 정적 분석, 동적 분석, 복잡도 등에 대한 자동 관리 및 시각화

Version 1.0.0 - 2015/03/05 16:31:18 Time changes...

Lines Of Code 849 ↗	Files 20 ↗	Functions 74 ↗
Java	Directories 20 ↗	Lines 1,574 ↗
	Classes 20 ↗	Statements 342 ↗
	Accessors 22	

**중복률**

Duplications  
6.2% ↗

Lines 98 ↗	Blocks 6 ↗	Files 2 ↗
---------------	---------------	--------------

**복잡도(중복 의존성 등) 관리**

Complexity  
2.0 /function  
8.0 ↗ /class  
8.0 ↗ /file  
Total: 159 ↗

SOALE Rating  
**A**

Technical Debt Ratio  
7.2% ↗

Technical Debt  
3d 6h ↗

Issues  
105 ↗

Blocker	0
Critical	0
Major	60 ↗
Minor	39 ↗
Info	6

**소스 정적 분석 결과**

Directory Tangle Index  
0.0%

Dependencies To Cut  
Between Directories: 0  
Between Files: 0

Cycles  
>0

테스트 커버리지

단위 테스트 성공률

Unit Tests Coverage  
60.3% ↘

Unit Test Success  
100.0%

Line Coverage 65.0% ↘	Condition Coverage 43.5% ↘	Failures 0	Errors 0	Tests 22 ↗	Execution Time 1.9 sec ↗
--------------------------	-------------------------------	---------------	-------------	---------------	-----------------------------

Events All

2015/03/05 Version 1.0.0

# ALM 시스템 소개 - 소스 리뷰

- 소스에 대한 상호 리뷰와 커뮤니케이션을 통해 소스의 품질 관리 및 리스크 제거

The screenshot shows a code review interface for a file named 'nerds.html'. A review comment by Glencore Lionetti is highlighted with a red box. The comment text is: "Consider changing this as that is not a typical resolution." Below the comment, there is a 'Create Issue' button. An arrow points from the text '수정 권고 사항을 바로 이슈로 등록' to the 'Create Issue' button. Another text label '소스에 대한 리뷰 및 커멘트' is placed over the review comment area.

소스에 대한 리뷰 및 커멘트

수정 권고 사항을 바로 이슈로 등록

# Agile/ALM 프로젝트에서 산출물 전략

- 스크럼 애자일 기법과 ALM 환경 기반 **일관성 있는 산출물**
  - ALM 시스템 (Wiki, Jira, SVN, Jenkins 등)을 통해 이루어지는 개발 작업과 커뮤니케이션을 산출물화 함으로써 "요구사항 → 산출물" 화 하는 과정의 일관성 유지
- 프로젝트의 원활한 **커뮤니케이션 도구로 활용**
  - 각 컴포넌트 별 일정 및 체크포인트에 따른 산출물 작성을 통해 커뮤니케이션의 수단으로 활용 → 리스크 제거 및 프로젝트 가시성 확보를 위한 수단
- **지속적인 보완**
  - 애자일 기법의 특징처럼 산출물을 통해 프로젝트의 리스크를 발견하고, 반대로 해결된 문제를 산출물에 반영함으로써 지속적인 보완

# 산출물 관리 목표 예시

항목	요구 사항	목표	관리 도구
일정 관리	프로젝트 내 주요 스케줄 관리 및 공유	프로젝트 내 주요 스케줄 공유	Team Calendar
문서 관리	요구사항, 설계, 이슈 등에 대한 상세 기술 내용	상세 내용을 효과적으로 기술하고 멤버 간 공유	Wiki
이슈 관리	요구사항, 작업 내용, 문제점 등에 대한 트래킹	이슈에 대한 체계적인 트래킹과 리스크 확인	Jira/Redmine
소스 관리	소스 형상 관리	소스와 소스에 대한 리뷰	SVN
빌드 관리	개발 중인 소스에 대한 자동 빌드	일일 빌드를 통해 소스 완성도 관리	Jenkins
소스 품질	코드 커버리지	40%	SonarQube
	주석률	30%	SonarQube
	중복률	0%	SonarQube
제품 릴리즈 관리	릴리즈 별 산출물 관리	각 릴리즈 별 산출물(코드, 패키지, 노트 등) 버전 관리	JIRA, SVN, FishEye, Jenkins



# 소스 형상 관리

**Subversion vs Git**

# 소스 형상 관리 시스템 용어

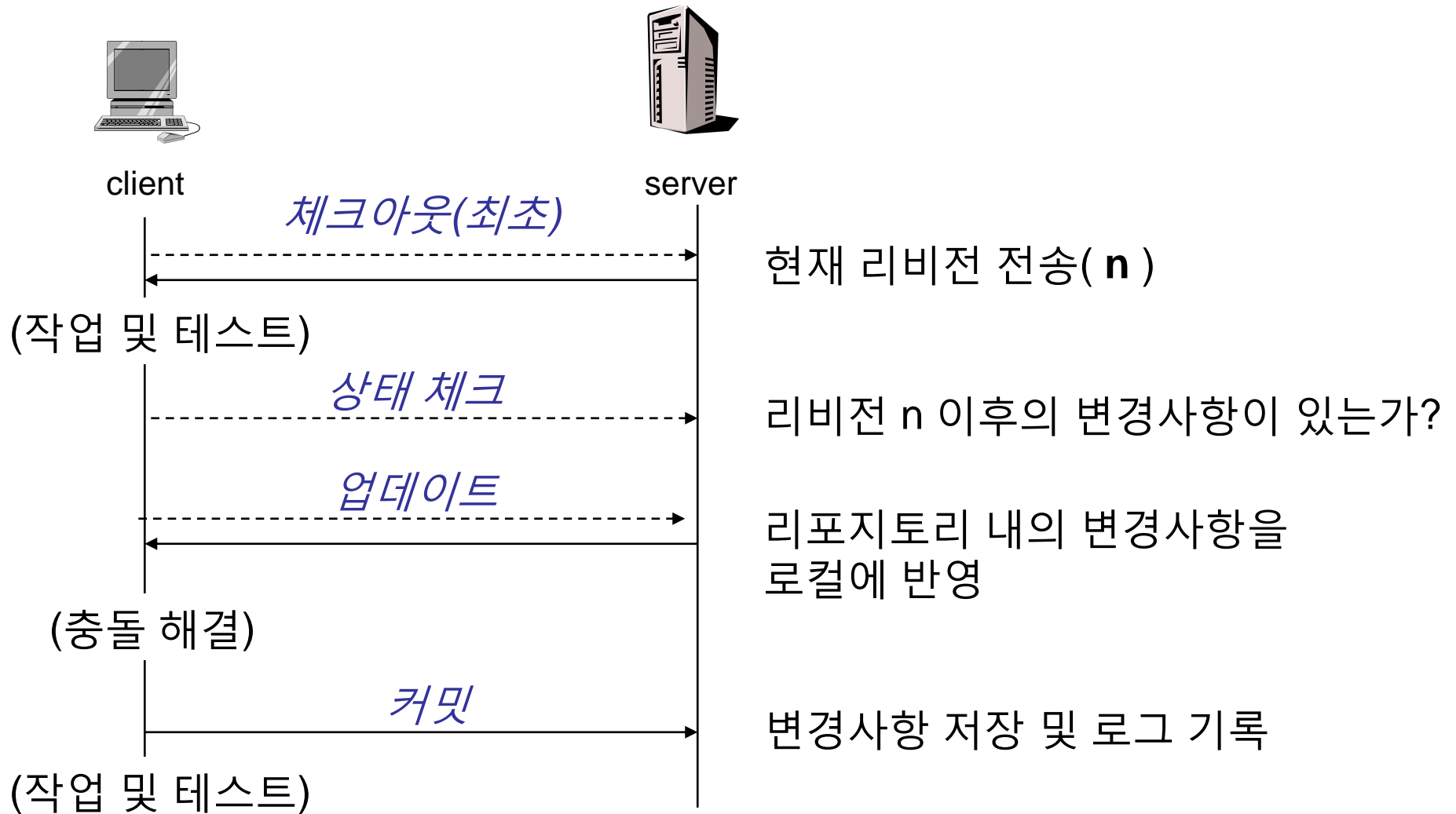
- 버전 관리 시스템에서 일반적으로 사용되는 용어는 아래와 같음

용어	설명
중앙 저장소 (Repository)	원본 소스를 저장하고 있는 저장소
작업 디렉토리 (Working Copy)	원본 저장소로부터 체크아웃을 통해 내려 받은 내 로컬 PC에 있는 작업 사본 디렉토리
커밋(Commit)	작업 디렉터리에서 변경, 추가 및 삭제된 파일을 원본 저장소인 서버에 적용하는 것
갱신(Update)	체크아웃을 받은 작업 디렉터리를 원본 저장소의 가장 최신 커밋된 버전까지 업데이트하는 명령어
리비전(Revision)	소스 파일을 수정하여 커밋하게 되면 일정한 규칙에 의해 숫자가 증가하며, 저장소에 저장된 각각의 파일 버전
롤백(Roll Back)	작업 디렉터리에 저장되어 있는 사본을 특정 리비전 또는 특정 시간으로 복원할 수 있도록 하는 명령어



# 일반적인 버전 관리 프로세스

- 중양 리포지토리 내에서 로컬 복사 후 변경한 사본을 커밋하는 방식으로 진행



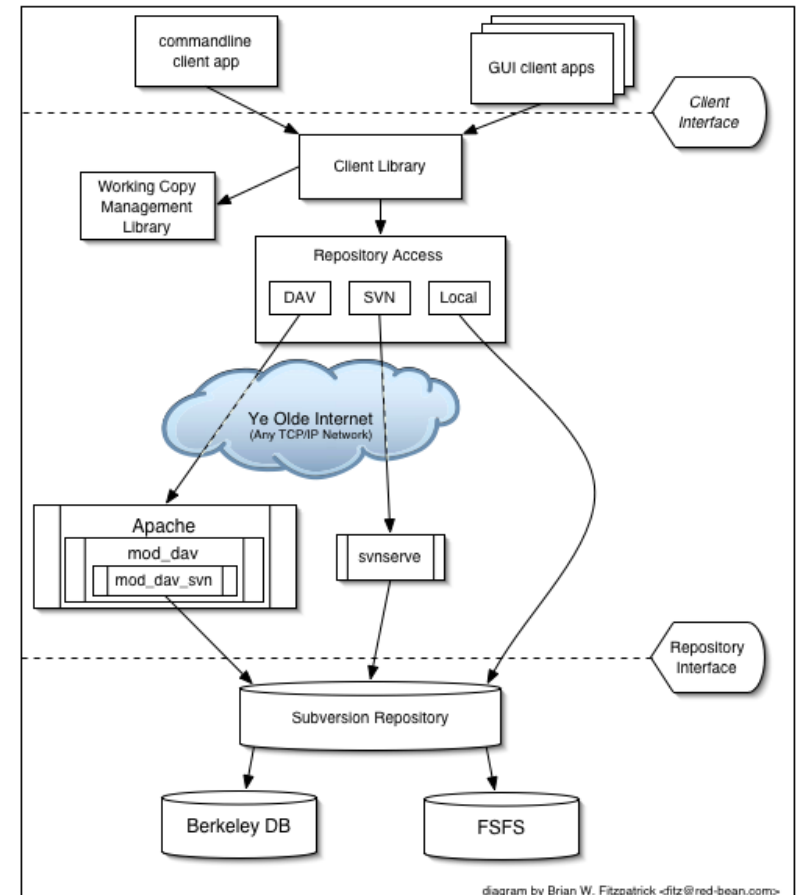
# Subversion, Git 간략 비교

구분	Subversion	Git
시초	CVS 문제점 <sup>1)</sup> 을 극복하기 위해 CVS 개발자들에 의해 2000년에 새롭게 시작	2005년 4월 BitKeeper 논쟁 <sup>2)</sup> 이후 리눅스를 만든 리누스 토발즈에 의해 시작
리파지토리 모델	클라이언트-서버 모델	분산형(Distributed)
동시성 모델	Merge, Lock	병합(Merge)
라이선스	Apache	GNU GPL 라이선스
지원 플랫폼	윈도우, 유닉스, 리눅스	윈도우, 유닉스, 리눅스
사용언어	C	C, shell scripts, Perl
보안	Numbers	SHA-1 hashes

- 1) 디렉토리, 파일 이름 변경이 불가능하고, 속도가 느리며 커밋 실패시 롤백이 지원되지 않아 소스 관리의 어려움
- 2) 2002년에 리눅스 커널 프로젝트는 상용 DVCS 시스템인 bitKeeper를 사용하기 시작했다. 2005년에는 리눅스 커널을 개발하는 커뮤니티와 BitKeeper를 개발하는 커머셜 회사간에 관계에 문제가 생기면서, 툴의 무료 사용권이 사라지게 됨. 이 사건으로 BitKeeper 기능을 토대로 새로운 버전관리 툴을 만드는 계기가 됨

# Subversion 개요

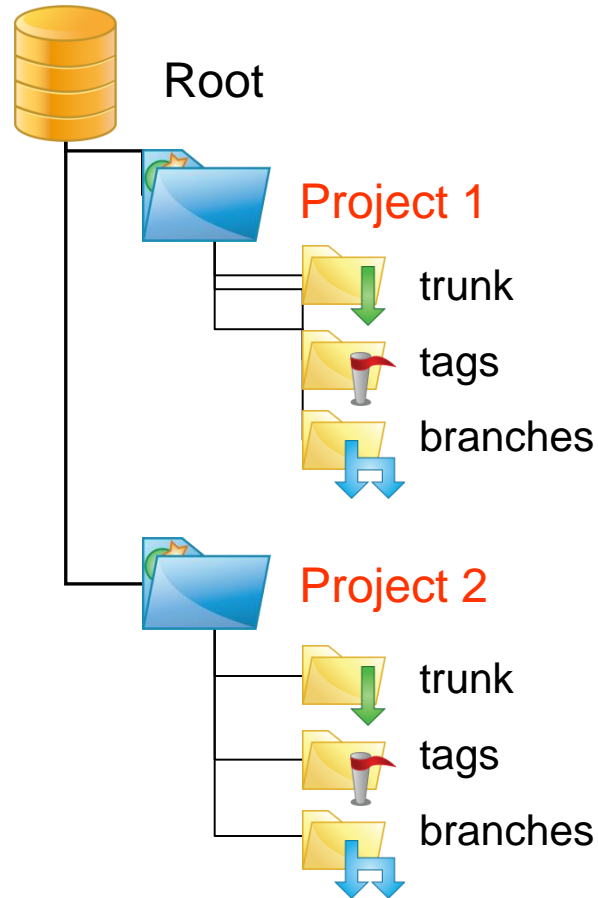
- CVS를 개발하던 개발자들이 CVS가 가지고 있던 문제를 해결하고자 완전히 새로운 아키텍처를 적용하여 새롭게 만든 솔루션
- 커밋 단위가 파일이 아니라 체인지셋
  - CVS에서라면 여러 개의 파일을 한꺼번에 커밋 하더라도 각각의 파일마다 리비전이 다름
  - Subversion에서는 파일별 리비전이 없고 한번 커밋할 때마다 변경 사항별로 리비전이 하나씩 증가
- CVS에 비해 엄청나게 빠른 업데이트/브랜칭/태깅 시간
- CVS와 거의 동일한 사용법으로 기존 사용자 유도
- 파일 이름변경, 이동, 디렉터리 버전 관리도 지원
- 원자적(atomic) 커밋
  - CVS에서는 여러 파일을 커밋하다가 어느 한 파일에서 커밋이 실패했을 경우 앞의 파일만 커밋이 적용되고 뒤의 파일들은 그대로 남음
  - Subversion은 여러 개의 파일을 커밋하더라도 커밋이 실패하면 모두 이전 상태로 회귀
- 양방향 데이터 전송으로 네트워크 트래픽 최소화
- 트리 별, 파일 별 접근 제어 리스트. 저장소 쓰기 접근을 가진 개발자라도 소스 변경 제어가능
- 저장소/프로젝트 별 환경 설정 가능



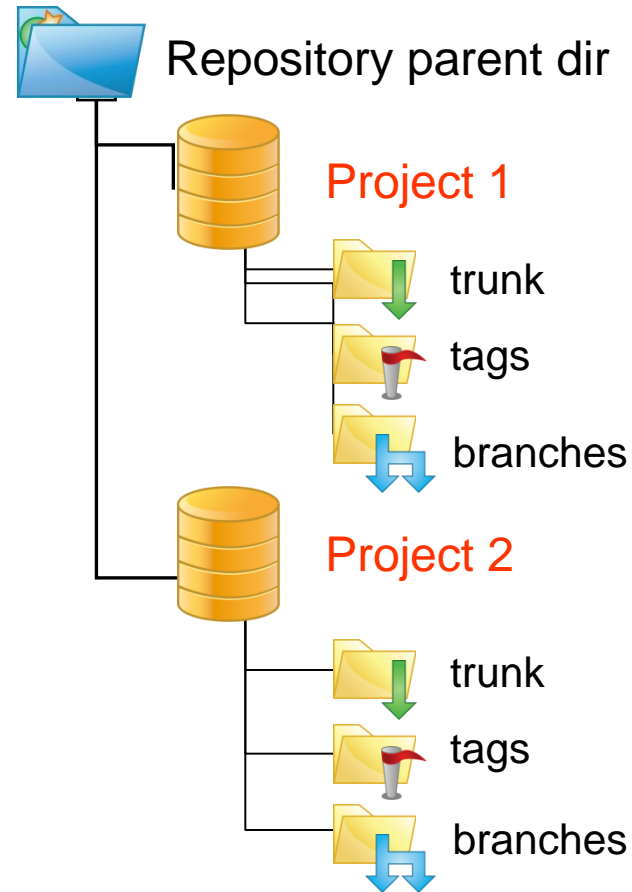
# Subversion 리포지토리 구조

- 여러 가지 방식의 리포지토리 구조를 생성하여 사용하는 것이 가능함

One repository, many projects

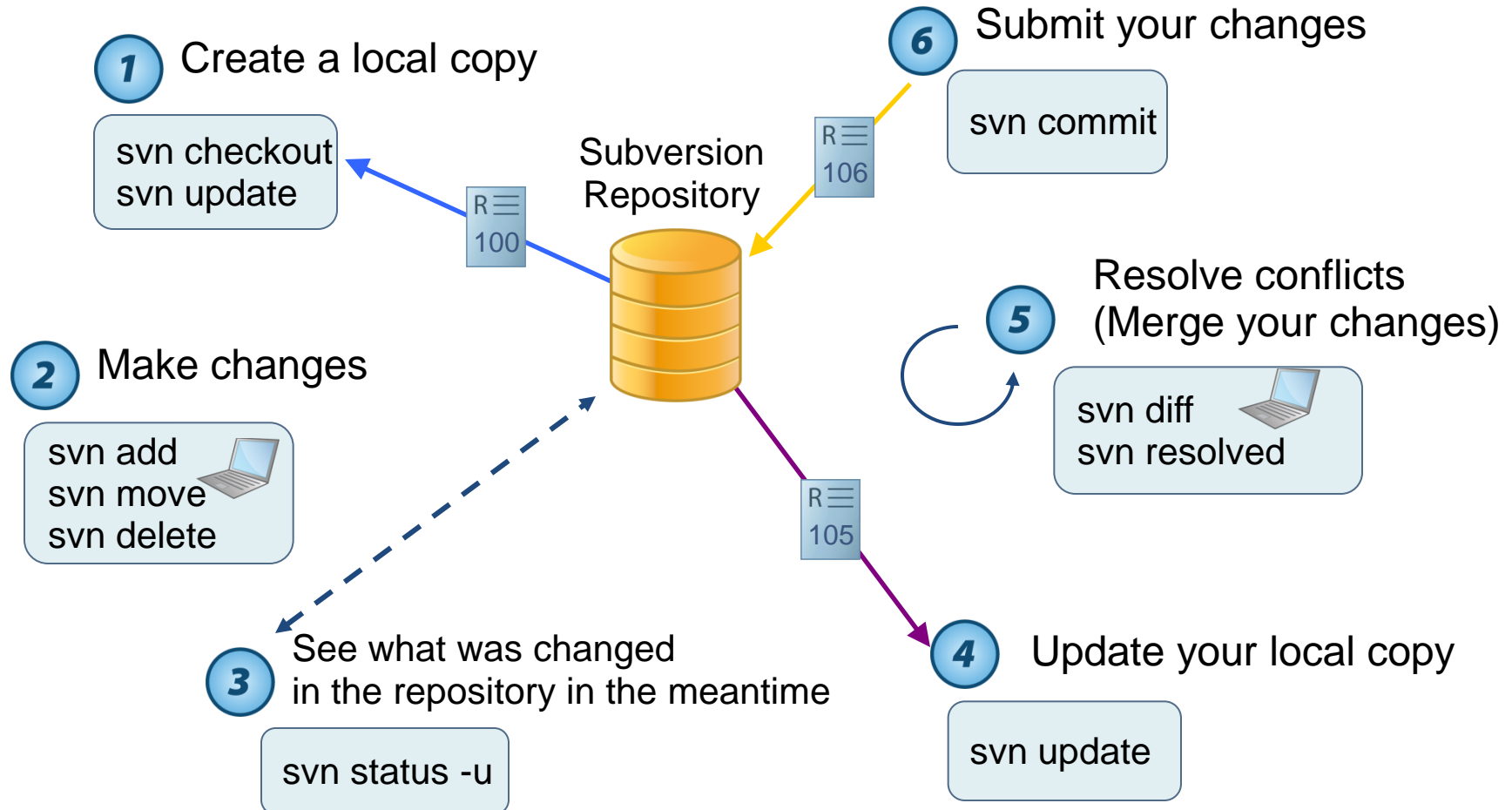


One project per repository



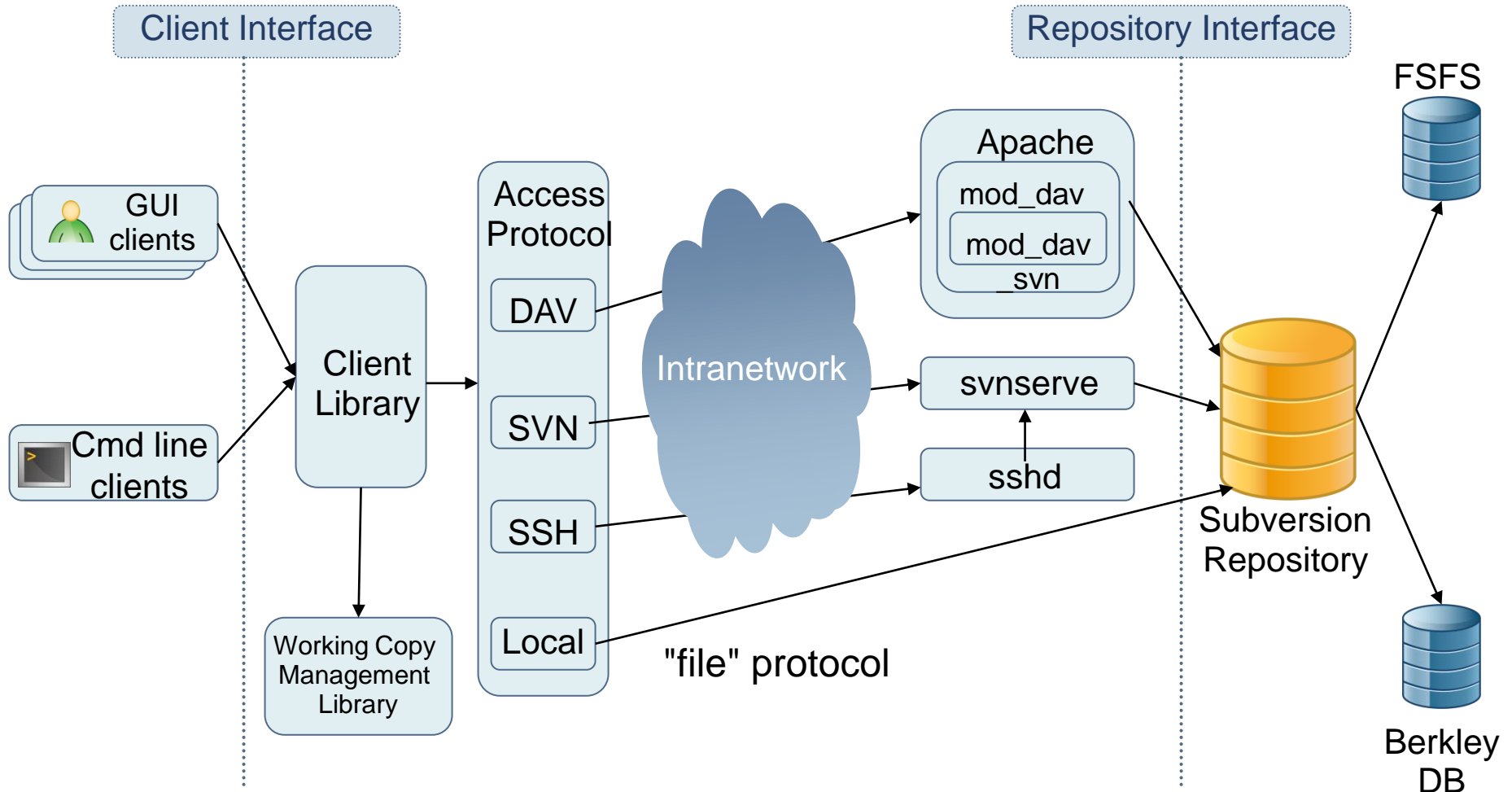
# Subversion 작업유형

- 중양 리포지토리 내에서 로컬 복사 후 변경한 사본을 커밋하는 방식으로 진행



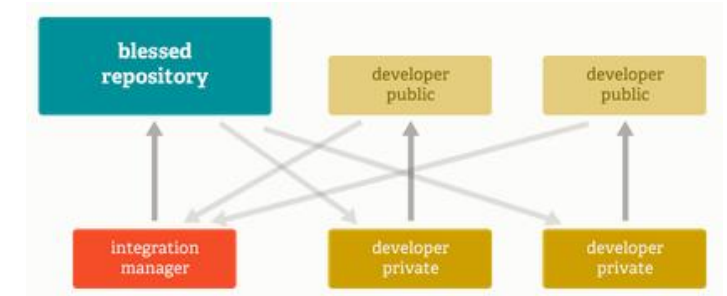
# Subversion 아키텍처

- 다양한 클라이언트 인터페이스를 통해 각 리포지토리에 접근하여 작업 가능



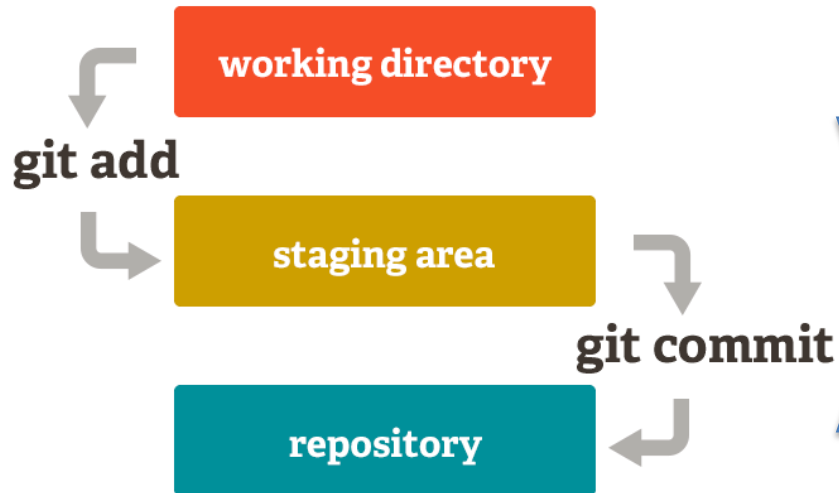
# Git 개요

- 리눅스 커널 커뮤니티와 BitKeeper를 개발하는 커머셜 회사간에 관계에 문제가 발생하면서 리눅스 토발즈에 의해 새롭게 만들어 버전 관리 솔루션
- Branching and Merging
  - 새로운 패치, 아이디어를 위해 브랜치를 생성하고 커밋 후 다시 메인 코드로 병합이 가능
- Small and Fast
  - 모든 동작들을 우선적으로 로컬에서 실시하게 된다. 따라서, 이것은 서버와 연동하기 위해 네트워크를 타지 않아도 되기 때문에 속도면에서 여러 가지로 엄청난 장점
- 분산형 개발 모델
  - 각각의 작업환경에서 별도로 작업한 소스들은 언제든지 메인서버로 업로드하고 merge가 가능
- 데이터 안전
  - 모든 파일은 체크섬 검사를 하고, 다운을 받을 때에도 다시 실시를 하여 Git에서 모든 bit까지 보장
- 스테이징 모드
  - 모든 파일은 스테이징(로컬) 영역에 추가되고 서버의 리포지토리에 커밋되는 두 단계로 구성



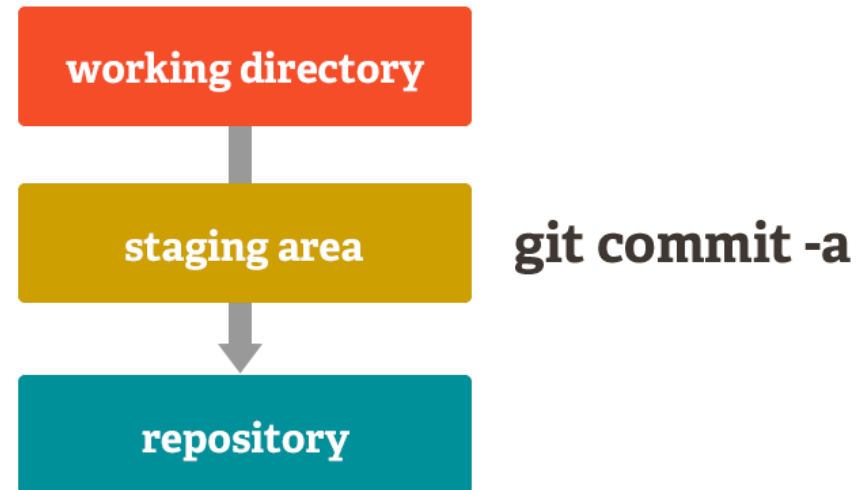
# Git 작업 유형

- 기존의 버전 관리 시스템과는 다른 로컬 영역의 저장소의 2단계를 가짐



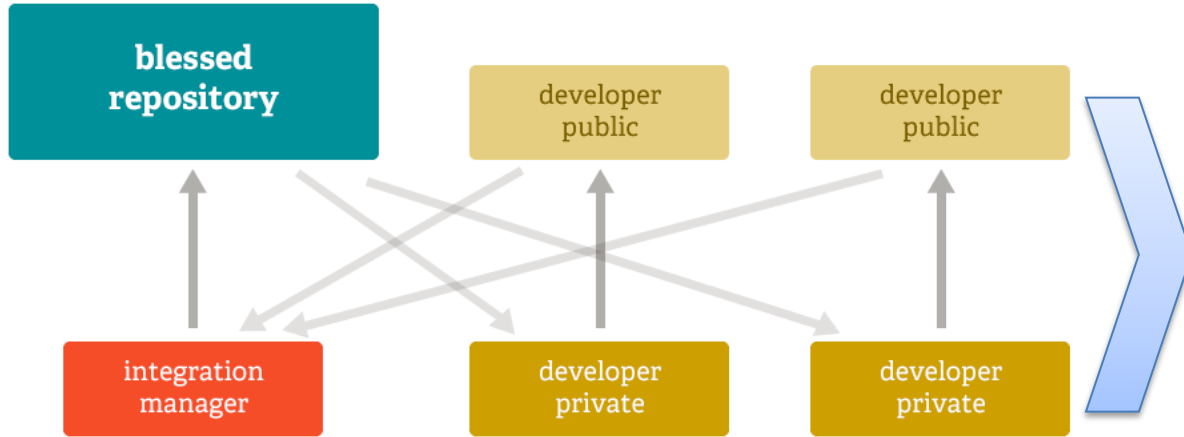
- 명령어들을 로컬에서 실시하고 나중에 서버에 올리는 방식을 사용
- 기존의 svn에서 commit이 1. 소스를 수정하고, 2. 저장소로 업로드 하는 2 단계를 각각의 명령어로 분리하여 중간에 로컬에서 일하다가 저장소에 적용하기 전의 잠재적인 한 번의 단계를 거침

- 한 번의 명령으로 바로 적용되는 SVN과는 달리 2번의 명령을 실행해야 마침내 서버에 있는 저장소로 업로드
- "git commit -a" 의 명령을 통해서 다른 버전관리 프로그램들과 같이 이용하는 것도 가능





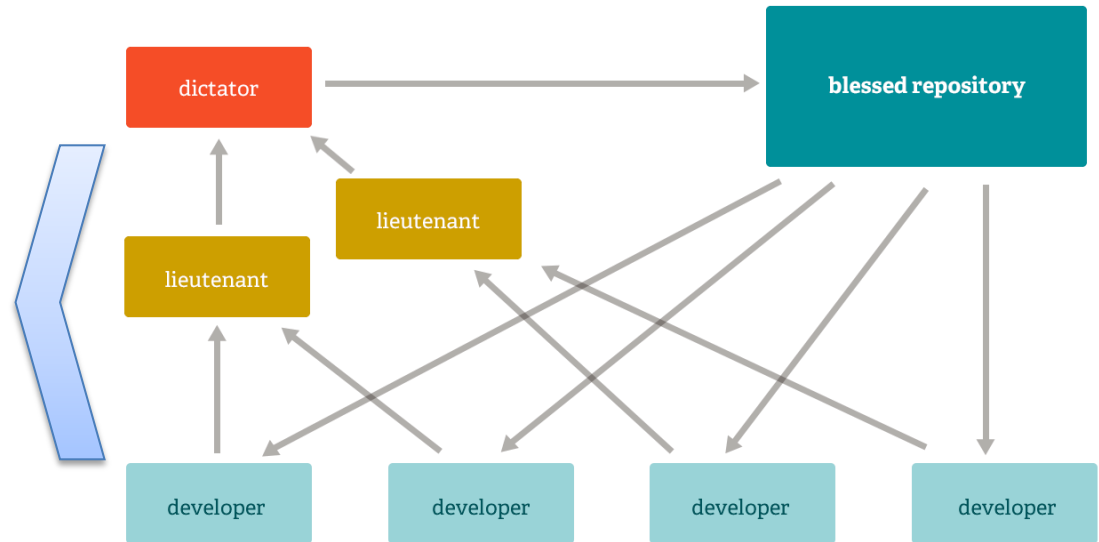
# 분산형 개발 아키텍처



- 각 팀원의 역할을 분담해서 한 명은 통합을 전담으로 수행
- 다른 개발자는 각자 부분 개발에 집중을 하는 식으로 역할 분담

<통합 관리 워크플로우 모델>

- 팀이 커질수록 복수의 repository를 쉽게 구성 가능
- 하위 팀들을 묶어주는 통합 관리자, 그리고 전체를 통합하는 관리자로 더 세분화해서 구성하는 것이 가능





# ALM 실습