10/2 보강 : 12/7까지 스냅보드강의

12장. SW 유지보수

숙명여자대학교
SOOKMYUNG WOMEN'S UNIVERSITY

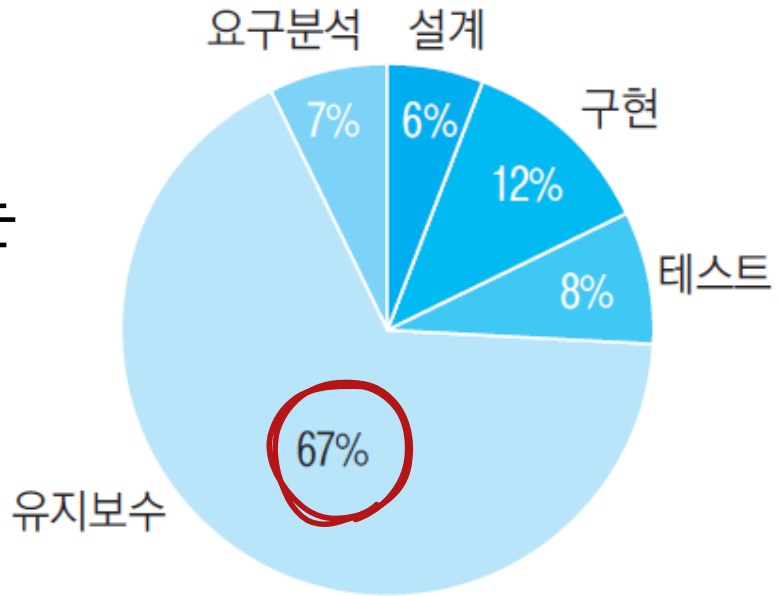
목차

1. 유지보수 개요
2. 유지보수 작업 과정
3. 소프트웨어 형상 관리
4. 역공학(Reverse engineering)
5. 깃허브 소개

1. 유지보수 개요 (1/5)

- SW 유지보수(maintenance)란 ↗ 버그제거, 기능·성능개선
  - 개발 후에 이루어지는 소프트웨어의 변경 작업
  - 소프트웨어가 유용하게 활용되는 기간
- 소프트웨어는 환경과 비즈니스 요구에 따라 진화함

- 유지보수에 드는 노력↑
  - 소프트웨어를 처음 개발하는 과정에서 유지보수를 좀 더 쉽게 할 수 있도록 설계



1. 유지보수 개요 (2/5)

- 레거시(legacy) 시스템 : 수십년전에 구축해서 지금까지도 사용중
  - 대체하려면 비용이 많이 든다.
  - 지식, 경험, 지능이 녹아 있음
  → 유지보수필요
- 소프트웨어 배포 이후 여러 가지 이유로 변경이 필요하게 됨
  - 버그 제거 (수정) → 변경된 부분만 리그레션테스트 (사용자)
  - 운영 환경의 변화
  - 정부 정책, 규례의 변화
  - 비즈니스 절차의 변화 e.g. 해킹대비 보안↑ → 이중인증
  - 미래 문제를 배제하기 위하여 변경 e.g. 신뢰성을 높이기 위한 수정

1. 유지보수 개요 (3/5)

- IEEE 1219 정의
  - SW 유지보수는 납품(delivery) 후 결함, 수정, 성능 개선, 제품을 변경된 환경에 적응시키기 위한 SW 제품 수정활동

- ISO 14764 정의
  - SW 유지보수는 문제(problem)나 개선의 필요성으로 인해 코드나 관련 문서 수정이 SW 제품에 수행되는 프로세스
  - SW 제품의 무결성(integrity)을 유지하면서 수정하는 것이 목적

과거제공했던기능에문제가생기지 않도록

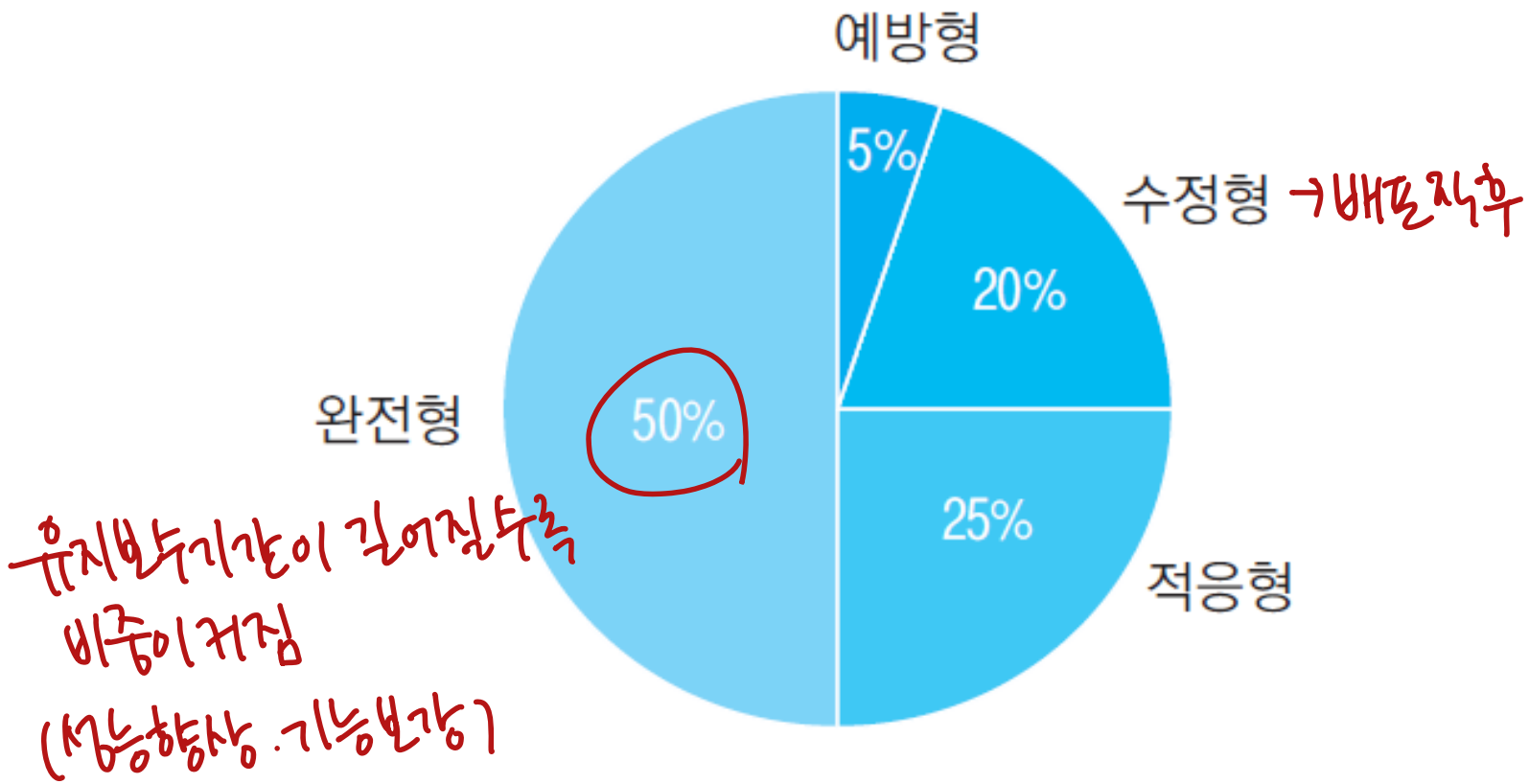
1. 유지보수 개요 (4/5)

- 유지보수의 유형 (IEEE)
  - 수정형 유지보수(corrective maintenance)
    - 발견된 오류의 원인을 찾아 계획적으로 문제해결
  - 적응형 유지보수(adaptive maintenance)
    - 변경된 환경에서도 계속 사용할 수 있도록 새로운 자료나 운영체제, 하드웨어 환경으로 이식
  - 완전형 유지보수(perfective maintenance)
    - 성능이나 유지보수성을 개선하기 위한 변경 코드의 기능·효율성
  - 예방형 유지보수(preventive maintenance)
    - 오류 발생을 방지하기 위해 수행하는 유지보수

    e.g. 코드주고받기 (협업을 위해, 간결한 코드를 위해) - 리팩토링

1. 유지보수 개요 (5/5)

- 유지보수 유형별 분포



유지보수기간이 길어질수록
비중이 커짐
(성능향상, 기능보강)

2. 유지보수 작업 과정 (1/2)

① 현재 프로그램의 이해 (대부분 문서가 없거나 업데이트되지 않은 경우 多)
- 프로그램 로직을 추적하거나 요구, 설계 등에 대한 이해가 필요

② 변경 파악과 분석
- 필요한 변경을 파악하고 그 영향도와 소요 비용, 변경에 의한 리스크를 분석

③ 변경 영향 파악
- 이해당사자들에게 알리고 피드백을 얻음 (승인받아야함)

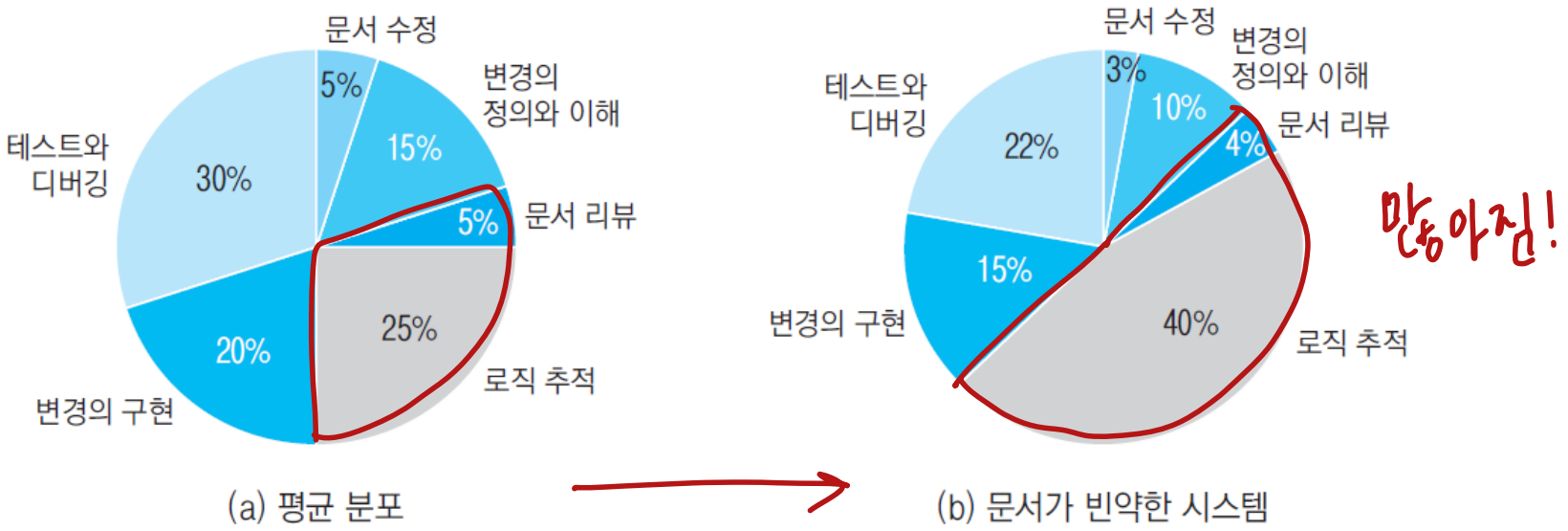
④ 변경 구현, 테스트, 설치 (현재운영환경에 통합)
- 시스템을 수정하고 확인 후 설치

설계가 변경되기도 함!

2. 유지보수 작업 과정 (2/2)

분석-설계-구현-테스트

- SW 개발은 코딩 중심의 작업이지만, 유지보수는 이해 중심의 작업, 통합된 작업 - 단기간에 포괄적으로



많아짐!

- 분석 설계 과정에서 문서작업을 소홀히 하면 안된다고 강조하는 이유

2.1 유지보수 프로세스 모델

- 임시방편
- 코드먼저수정
  → 나머지에반영

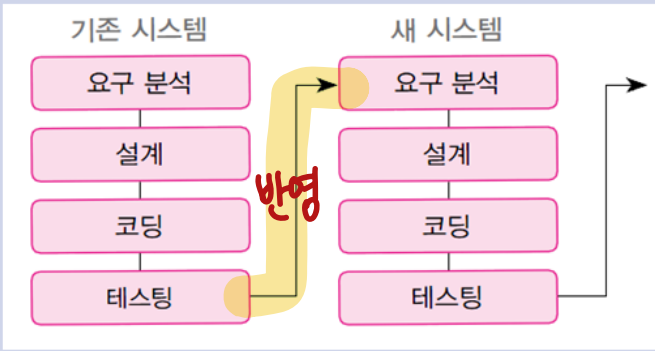
즉시수정모델

장) 빠르다

단) 여러번 반복되면 원래의구조가 무너질 수 있음 → 문제발생

- 개발프로세스와 비슷함
- 문서먼저

반복적개선모델



장) 지속적인 문서 업데이트 → 장기적 유지보수성↑

단) 시간이 많이걸림 (주기가길다)

- 재사용기반 SW 개발 프로세스와비슷함
- 재사용요소를반영

재사용중심모델

장) 옛날 버전 재사용 → 제품라인이 있는경우 적합 (ex. MS 시리즈) 컴포넌트가 쌓여 비용효과적

단) 초기 저장소 구축비용

2.2 프로그램 이해

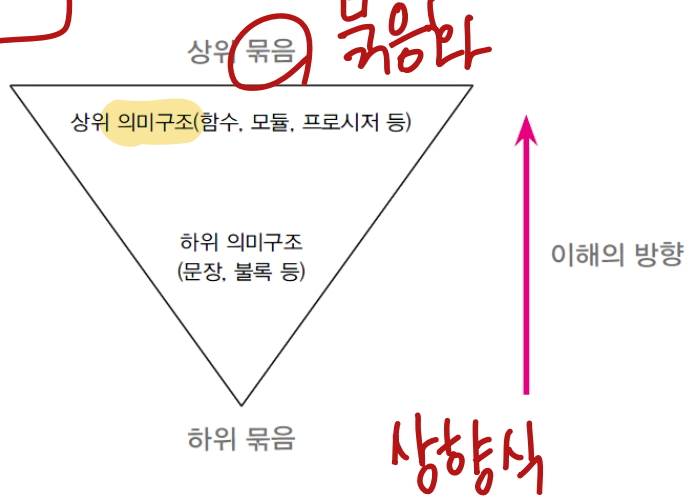
사람들이 대상에 대해 갖고 있는 개념 → 개인의 경험, 가치관 반영

- 원시코드로부터 설계나 명세를 추출하여 멘탈 모델로 표현하는 작업
- 개발 프로세스와 반대로 추상성을 추구하는 방향
- 상향식 이해 모델
  - 상향식(bottom-up)
  - 묶음화(chunking) → 0000 00

묶음

```
maxValue = Table[1];
For index = 2 To 100 Do
    If Table[index] > maxValue Then
        maxValue = Table[index];
    End;
End;
```

→ 경험이 많을수록 이 묶음의 의미를 빨리 알 수 있음!



2.3 변경 파악과 분석

잘해야비용↓

- 변경 요구를 기초로 어떤 부분을 변경할지 찾아냄

+변경방법(ex. 고치는게사람이너무많이들면 상용컴포넌트구매하자)

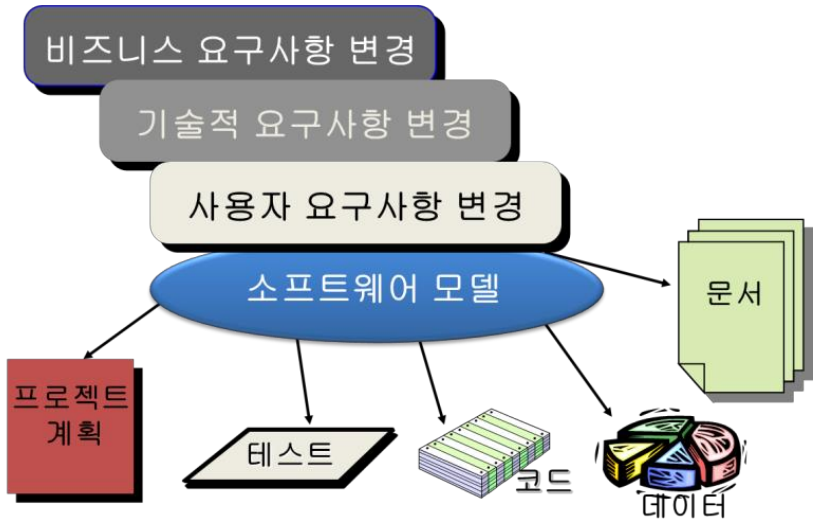
- 변경 분석
  - 변경 효과 분석
  - 변경을 구현하고 테스트하는 데 드는 비용, 시간의 예측 → 너무많이들면 다른방법찾기
  - 리스크 파악

- 객체지향 소프트웨어
  - 변경 효과 : 클래스 사이의 의존관계로 파악

3. 형상관리

↳원래 HW 개발용어 (부품변경관리)

- 형상 관리(Configuration Management)
  - 개발 주기 동안 생성된 문서를 관리하고 소프트웨어 시스템과 컴포넌트의 상태를 추적하는 작업
- 문서와 결과물에 대한 변경이 잘 조정되지 않는다면 불일치 발생 ⇒ 변경은 철저히 관리해야



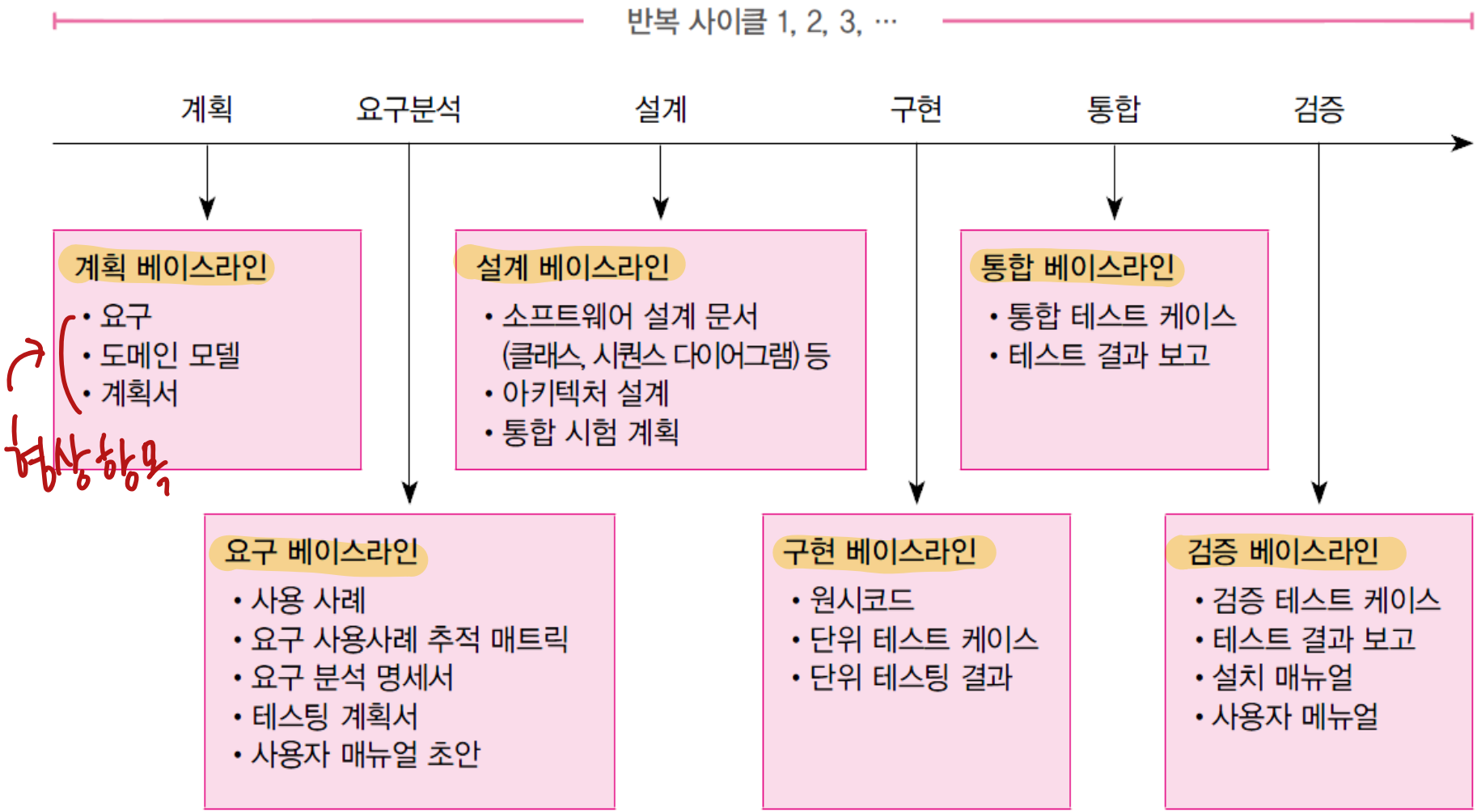
3.1 베이스 라인 (1/2)

- 베이스 라인(baseline)
  - 소프트웨어 개발 과정 중 산출물의 집합이 만들어지는 특정 시점 진도측정
  - 프로젝트 성공을 위해 반드시 거쳐야 하는 중요한 지점
  - 소프트웨어 형상 항목(configuration item)의 집합으로 구성
    ↳ 형상관리대상
- 목적
  - 프로젝트의 중요한 상태 정의
  - 프로덕트가 특정 상태에 이르렀는지를 나타냄
  - 계속되는 개발, 유지보수 작업의 기준 - 변경의 수용시점
  - 형상 항목에 대한 변경을 제어하는 메커니즘

베이스라인전까지는 변경가능!

3.1 베이스 라인 (2/2)

단계별설정!



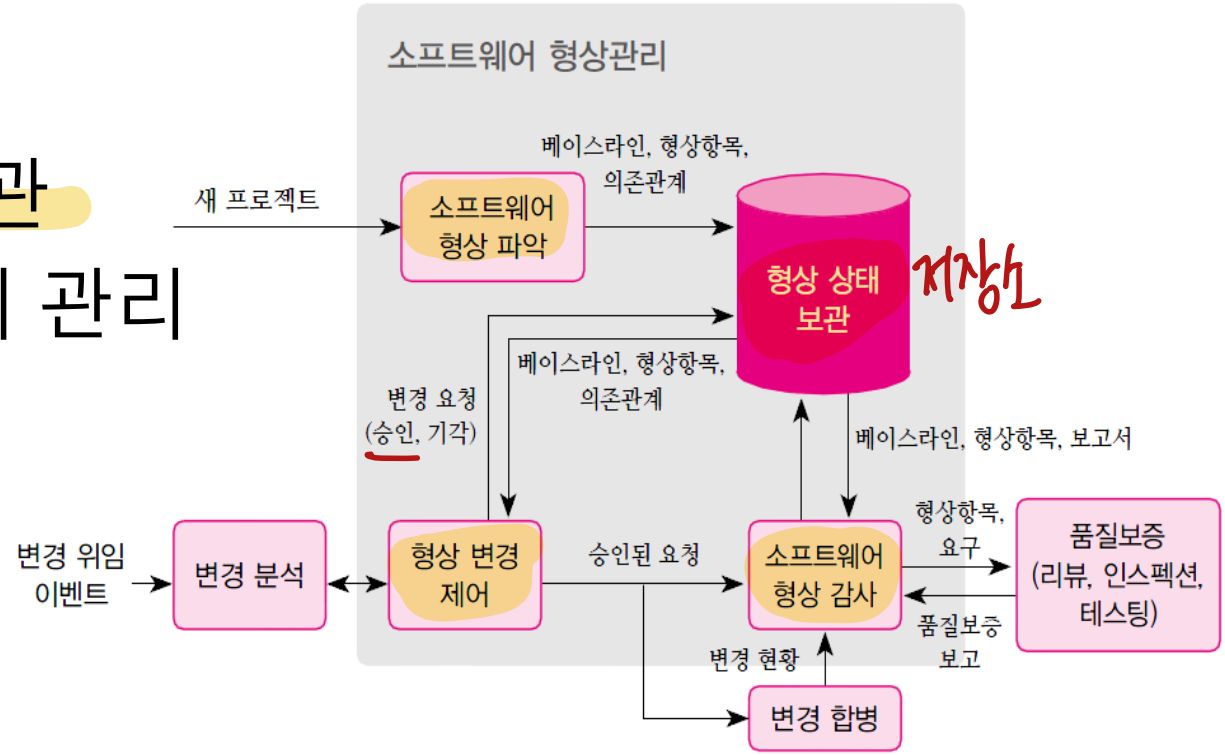
형상항목

3.2 형상관리 절차

4가지

1. 소프트웨어 형상파악 프로젝트가 새로 시작될때
2. 형상변경제어 : 변경을 수행할지 평가하고 결정하는 절차
3. SW 형상 감사
4. SW 형상 상태 보관

- 형상통제위원회의 관리



3.3. 형상 항목 정의

모든문서가 형상관리를 받는것은아님!

형상라인과정
. 베이스라인설정
. 형상항목정의

- 형상항목을 정의
  - 고유 식별자 : 고유번호
  - 이름 : 예) 도서대출유스케이스, 도서대출 시퀀스 다이어그램..
  - 문서 종류 : 예) 요구분석명세서, 설계서, 테스트케이스 ..
  - 문서 파일 : 파일의 이름과 경로
  - 저자
  - 생성 날짜, 목표 완성일
  - 버전 번호
  - 업데이트 이력
  - 설명
  - SQA 담당자 : 형상항목 품질보증 책임자 (quality assurance)
  - SCM 담당자 : 형상항목 체크 책임자 (configuration management)

4. 역공학*

↳원래 HW에서 제품복제 목적(경쟁사제품)

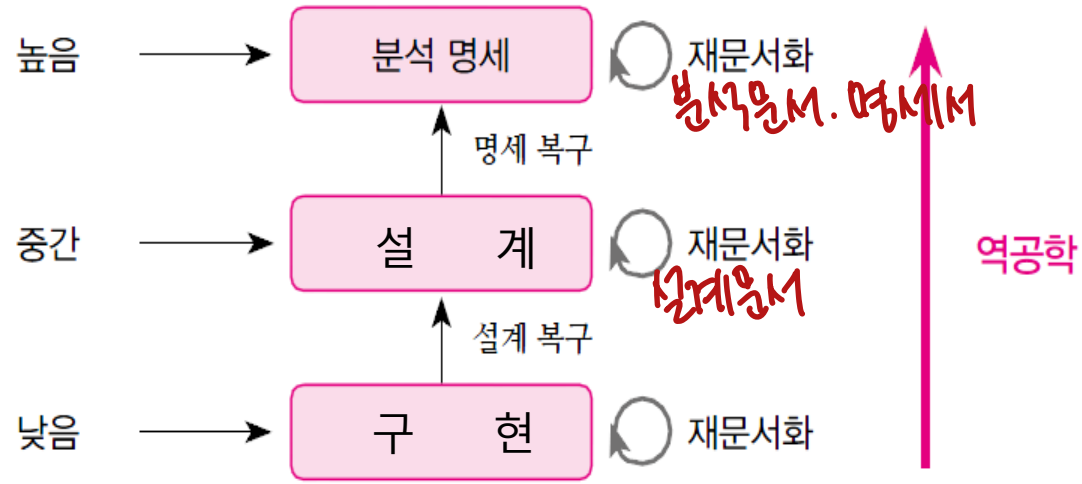
- 역공학(Reverse Engineering)의 정의
  - 대상 시스템을 분석하여 시스템의 컴포넌트와 관계를 찾아내어 같은 수준의 다른 표현이나 더 높은 수준의 표현으로 만드는 작업

↳ 역공학으로 코드구조, 파일구조 추출 → 리팩토링할때 이용

↑ 추상화수준 (추상 수준)

↳ 코드관점으로 (개발 결과물)

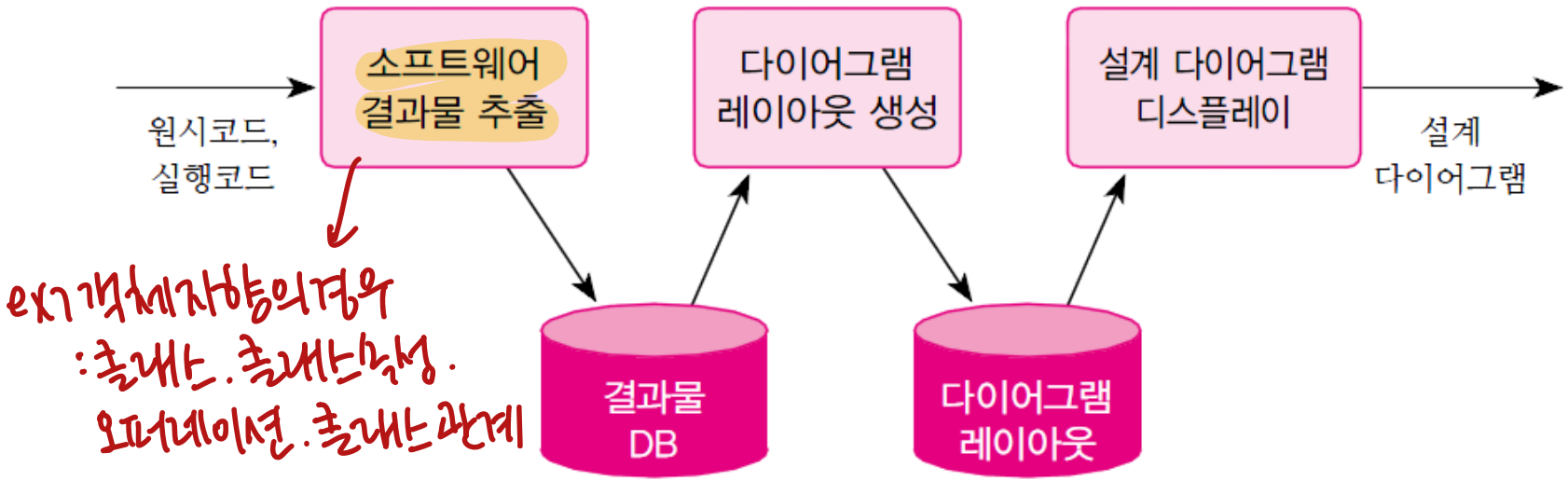
어떤것이나 구조는 어떤지



4.1 역공학 작업순서

- 역공학은 도구를 이용한 자동화가 가능
- 원시코드를 입력 받아 설계명세서를 추출

- 역공학 도구의 구성



4.2 역공학의 용도

- 복원된 다이어그램은 다음 여러 방면에 사용
- 프로그램 이해
  - 소프트웨어의 구조, 기능, 동작 이해 용이
- 정형적 분석(정형기법) → 수리논리기반, 명세&검증
  - 소프트웨어에 존재할 수 있는 문제를 감지
- 테스트 케이스 생성
  - 흐름도의 경로 : 테스트케이스 설계에 도움
- 리엔지니어링(re-engineering)
  - 재구축 과정 (SW의 일부를 재설) ← 구조파악에 활용

다시만들다!

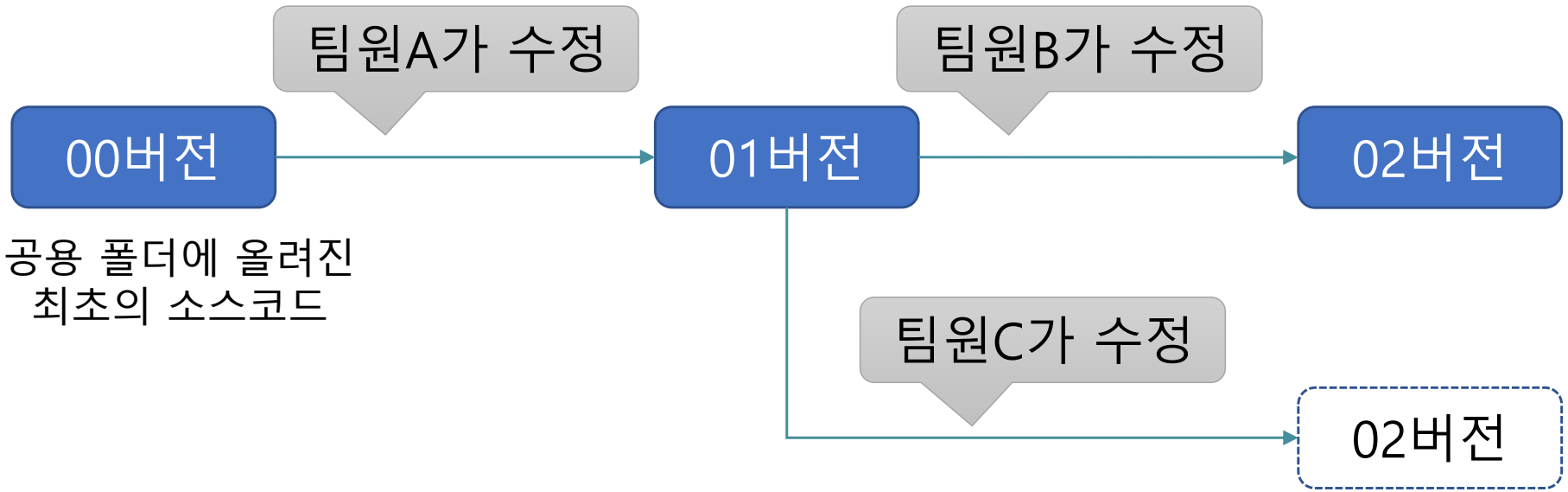
4.3 설계 복구

- Design Recovery
  - 원시코드를 자세히 검토하여 의미 있는 추상성 높은 표현을 찾아내고 추출하는 작업

- 복구된 설계
  - 원시코드 이해에 도움이 될 수 있음
  - 향후 유지보수 또는 역공학을 위한 베이스라인
  - 유사한 다른 애플리케이션을 위하여 사용될 수도 있음
- 프로그래밍 언어 구조에 크게 좌우 → 객체지향 : UML로 자동추출가능 아닌경우 어려움.
- 도메인 지식이 필요할 수도 있음

5. 깃허브 소개 (1/2)

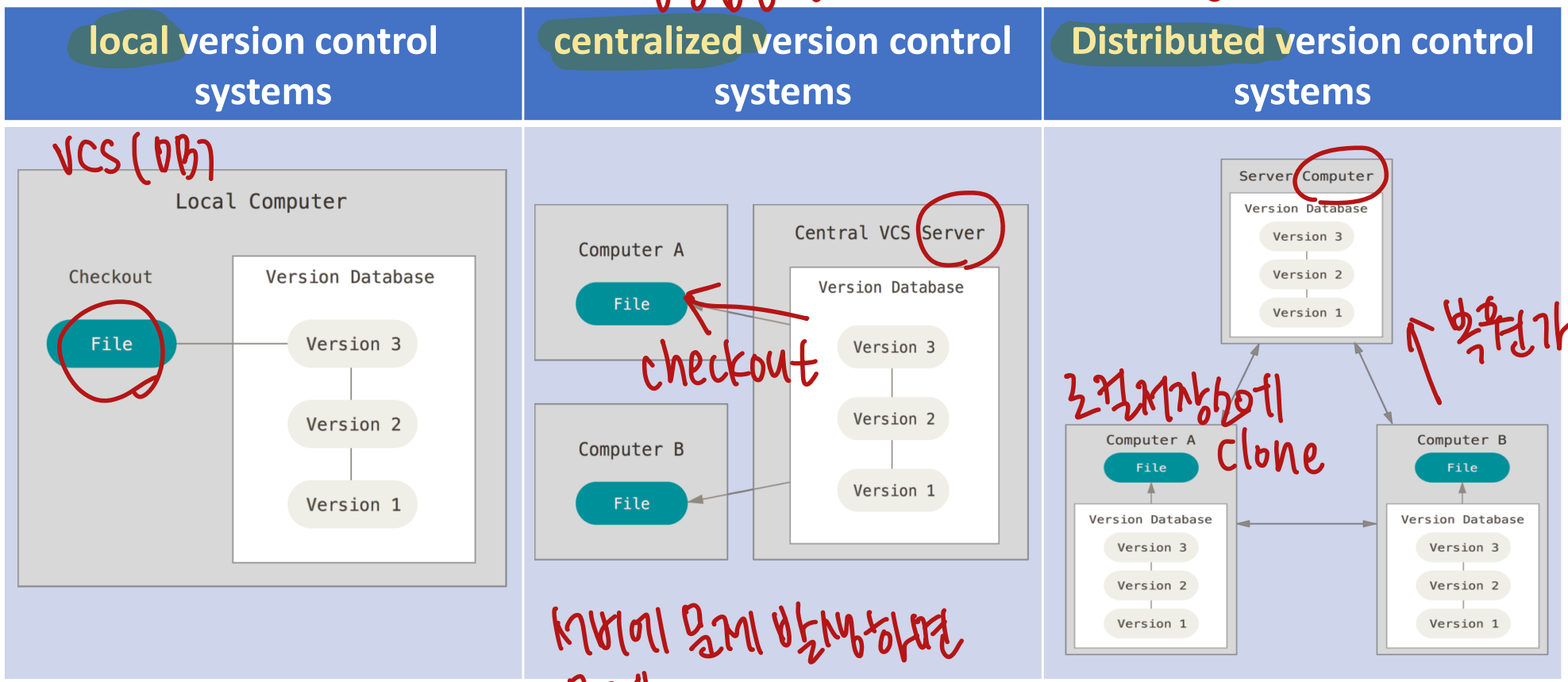
- 버전관리



- 여럿이 함께 작업하는 협업 프로젝트에서 버전관리는 필수

5. 깃허브 소개 (2/2)

- 버전관리시스템의 유형



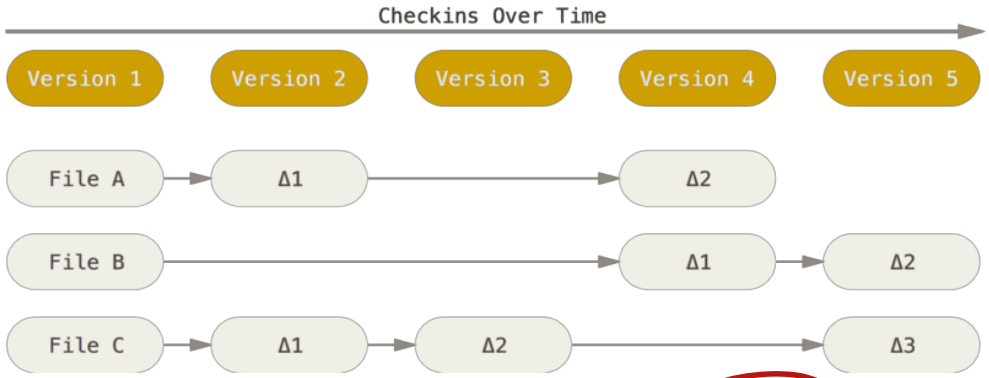
5.1 GIT이란 (1/2)

기존의 버전관리 시스템과의 차별성 : 가볍고 빠르다 (데이터를 다르게 다룸)

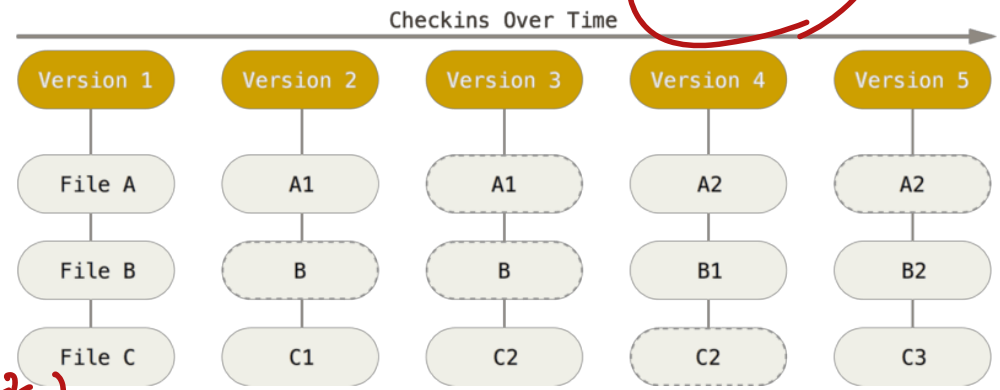
- 차이(differences)가 아닌 스냅샷(snapshots)

기존

- 버전관리는 각 파일의 기본 버전에 대한 변경 사항으로 데이터 저장



- Git = 시간 경과에 따른 프로젝트의 스냅샷으로 데이터 저장



- 커밋된 상태값들만 저장 (크기가 작음)
- 파일의 변경이 없으면 이전 상태의 링크만 가져옴.
- 서버 고장에 이어서 로컬에서 해결 (빠름)

5.1 GIT이란 (2/2)

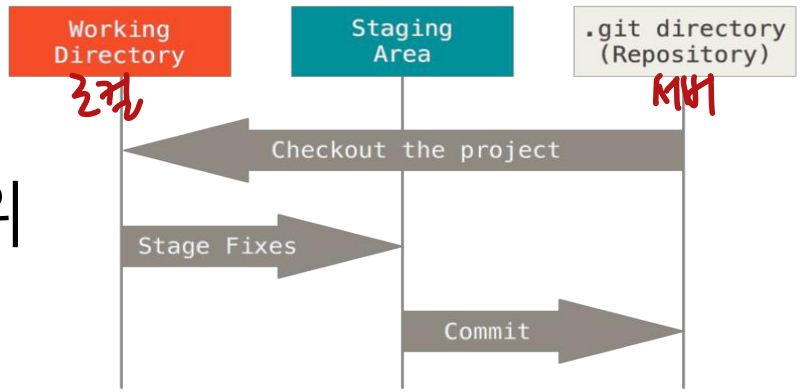
- 깃(Git)
  - 많은 개발자들이 협업할 수 있도록 만들어 놓은 소스코드 버전관리 시스템
- 깃허브(GitHub)
  - Git으로 관리하는 프로젝트를 올려둘 수 있는 Git 호스팅 사이트 중 하나

Git 호스팅사이트	모기업	특징	가격정책
GitHub.com	GitHub Inc. (MS에서 인수)	세계 최대 규모의 Git 호스팅 사이트	공개저장소 생성 무료. 비공개저장소는 작업자 3인 이하는 무료. 엔터프라이즈는 월 $21
GitLab.com	GitLab Inc.	NASA, Sony 등 10만개 이상의 조직이 사용. GitLab 프로젝트 자체가 오픈소스	공개 및 비공개저장소 생성 무료. 소스코드 빌드 도구 사용에 따라 월 $4 ~ $99
BitButcket.org	Atlassian	Atlassian이 만든 이슈관리 시스템인 Jira와 연동이 용이	5명 이하 팀은 공개 및 비공개 저장소 생성 무료. 그 이상은 월 $2 ~ $5

5.2 Git 메커니즘 (1/2)

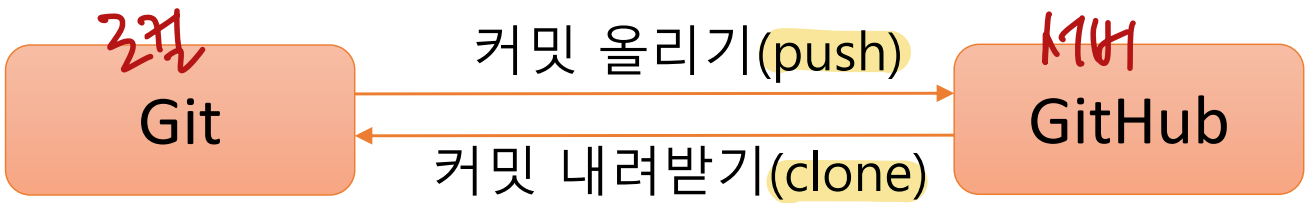
- modified : 수정만
- staged : 커밋할것이라고표시
- commited : 커밋(저장) 완료

- Git에는 파일이 상주할 수 있는 세가지 상태가 있음
  - modified, staged, committed
  - 커밋(commit) : 버전관리를 통해 생성된 파일, 또는 그 행위
    ↳ 수정됐다고등록하는것



- 로컬저장소와 원격저장소

로컬저장소를 깃이라고하기도함

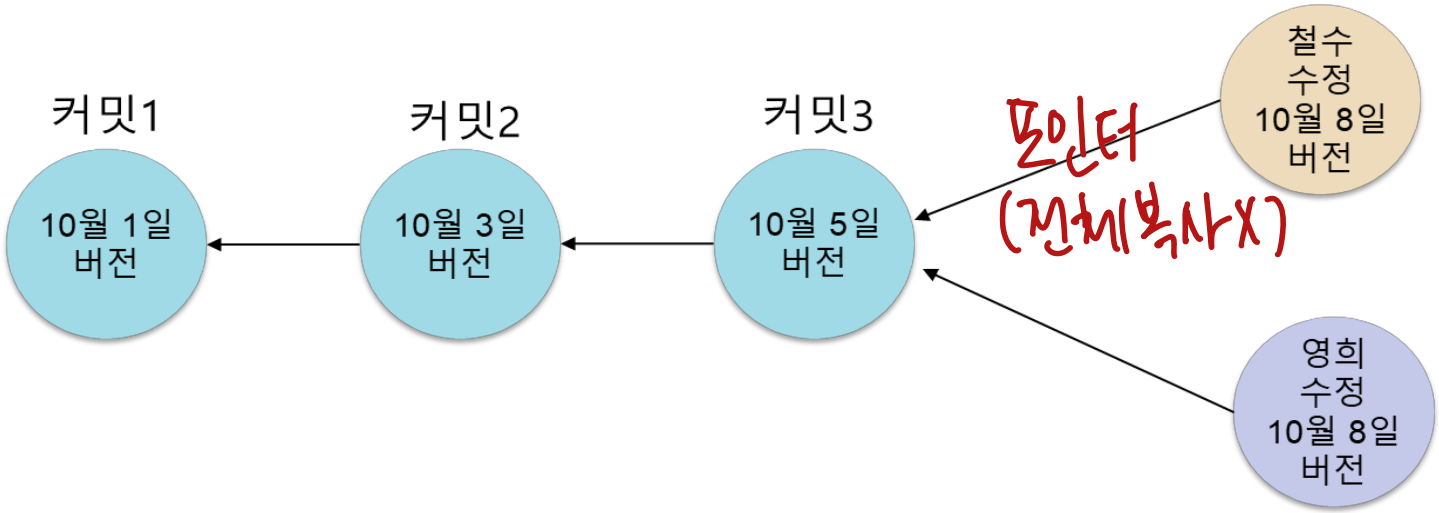


1. 내 컴퓨터에 깃 설치
2. 로컬저장소 생성 → git디렉토리가생성됨
3. 커밋만들기

1. 원격저장소(repository) 생성

5.2 Git 메커니즘 (2/2)

- Git의 커밋 관리 방식
  - 새로 만든 커밋은 기존 커밋 다음에 시간순으로 쌓임



  - 특정 기준에서 줄기를 나누어 작업할 수 있는 기능이 브랜치(branch) : 브랜치를 통해 협업이 가능해 짐

5.3 Gitbash

다기 source tree : GUI툴

명령어기반

- bash = Bourne Again Shell
  - 스티븐 본이 개발한 최초의 유닉스 쉘 프로그램인 sh의 확장판
  - git bash
    - 운영체제와 상관없이 사용할 수 있는 리눅스 기반 터미널용 git

```
ykimh@DESKTOP-CJOCC88 MINGW64 ~/My Documents/gitEx (master)
$ git log
commit 773295c5d056ab11777908a02b2499170a2e0d17 (HEAD -> master)
Author: Yukyong Kim <ykim.be@sookmyung.ac.kr>
Date:   Mon Nov 1 12:37:50 2021 +0900

    첫 번 째  커 밋  - 1절

    애 국 가  1절 의  첫 줄  입 력

commit b290f4cde827466176defd344409c2c02d20cdbc (origin/master, origin/HEAD)
Author: Yukyong Kim <44826830+Yukyong-Kim@users.noreply.github.com>
Date:   Mon Nov 1 11:50:25 2021 +0900

    Add files via upload

commit 5b1210c7f89a97f2093a348f1953b0dd2365ed38
```

5.4 GitHub 작업플로우

- 협업중심의 워크플로우로 Fork 해서 프로젝트에 기여하는 방식
- 토픽브랜치 중심으로 작업이 이루어짐
- 프로젝트를 Fork
- master 기반으로 토픽 브랜치를 만듦
- 수정하고 커밋
- 자신의 GitHub프로젝트에 브랜치를 Push
- GitHub에 Pull Request 생성
- 프로젝트 소유자는 Pull Request를 merge