Introduction to CTIP

김의섭

2016-03-04

목차

- CI
- CTIP
- CTIP 장단점
- CTIP 구성도
- Tools
- Team Projects

CI

Continuous Integration

- **소프트웨어 개발에서 Build*(Test-CTIP)* 의 프로세스를 지속적으로 수행하는 것.**
- 지속적으로 개발된 Unit 코드에 대한 Integration 작업을 수행하여, **항상 최신의 상태를 유지.**
- Agile 개발 방법론/프로세스 중의 하나인 XP(eXtream Programming)의 Practice 중의 하나.

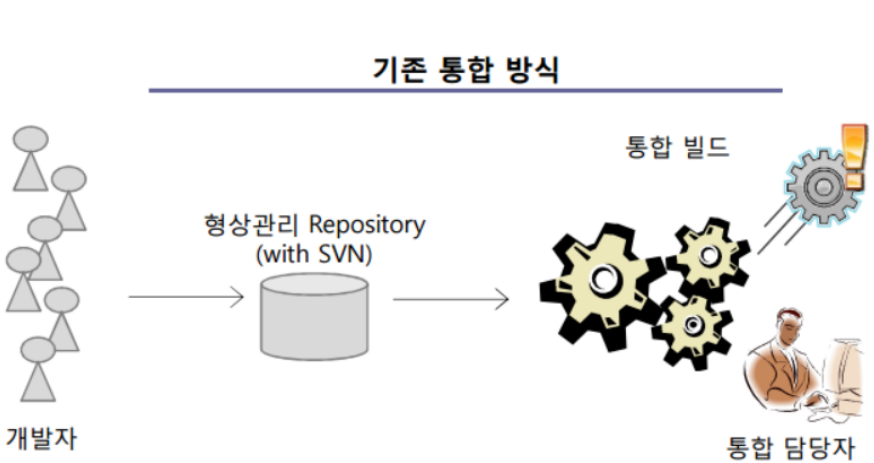
- **팀을 기반으로한 협동적인 개발**
  - 반례) 고전적인 소프트웨어 개발 방법론인 폭포수 모델 과 나선 모형

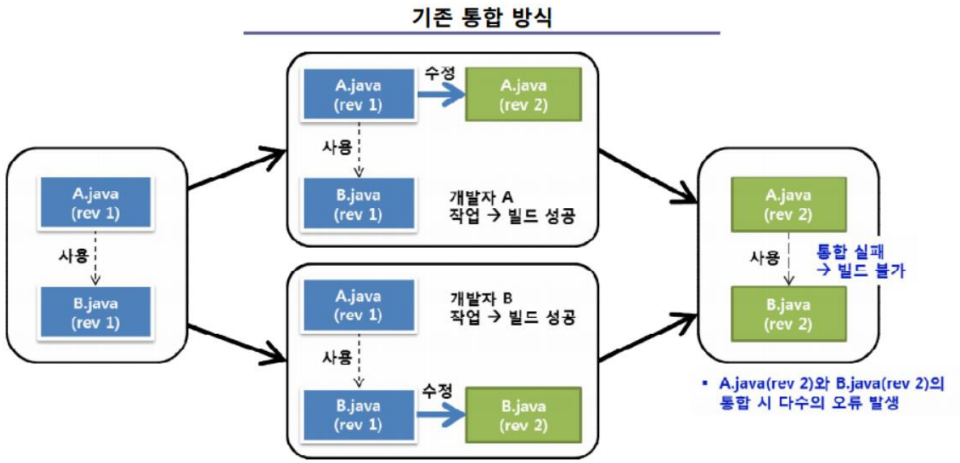
- 지속적인 통합(Continuous Integration)은
  팀의 구성원들이 자신들의 작업한 내용을 자주 통합하는 개발 지침를 말한다. (적어도 하루에 한번)
  - 마틴 파울러 (Martin Fowler)

기존 문제점

기존 문제점 1. 개발 막바지에 이르러 통합 작업 수행

- 개발자 간 잘못된 의사 전달로 인해 수많은 오류 발생
- 각종 오류를 한꺼번에 처리해야만 하는 Integration Hell 현상 발생
- 통합 시점에 이미 소스 코드가 방대해져 오류의 원인 파악 및 해결을 위해 많은 시간 할애



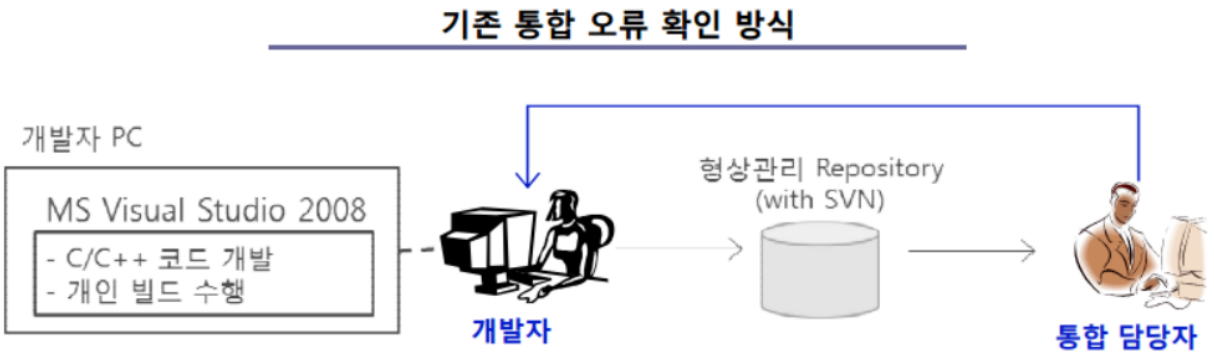


[출처] C 환경에서의 CI(지속적통합) 구축 사례
/ JAVA 환경에서의 CI(지속적통합) 구축 사례

기존 문제점

기존 문제점 2. 통합 오류에 대한 빠른 대처 부족

- 통합 빌드 오류를 통합 담당자가 각 개발자에게 직접 전달
- 개발자가 해당 소스코드를 개발한 후 많은 시간이 흘러 통합 오류 확인
- 통합 오류에 대하여 신속한 대응 체계를 갖추지 못함



[출처] C 환경에서의 CI(지속적통합) 구축 사례
/ JAVA 환경에서의 CI(지속적통합) 구축 사례

Continuous Test & Integration Platform

- Continuous Integration **+ Continuous Test**

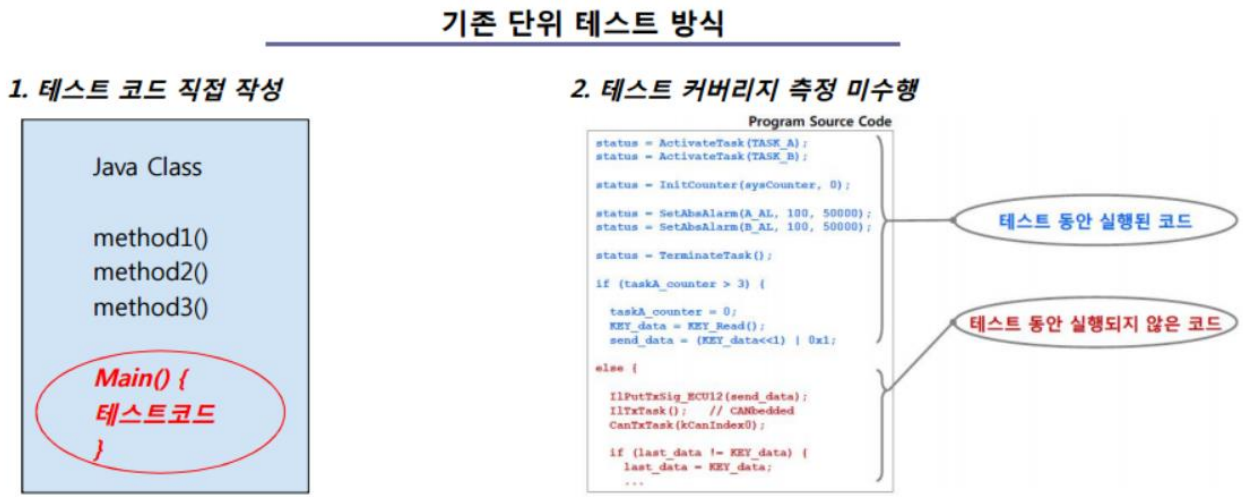
- CI (Continuous Integration)
  개발기간 동안 개별 개발된 모듈에 대한 빌드를 지속적으로 수행하기 위해 XP(eXtream Programming)에서 도입한 **빌드 자동화 개념**

- CTIP (Continuous Test & Integration Platform)
  CI 개념에 **테스트 자동화 개념을 더함**
  빌드 도구, 테스트 도구 및 기타 개발 시 유용한 도구들이 유기적으로 동작하도록 묶은 도구 집합 (Tool Chain) 환경

기존 문제점

기존 문제점 3. 수작업에 의한 테스트 수행

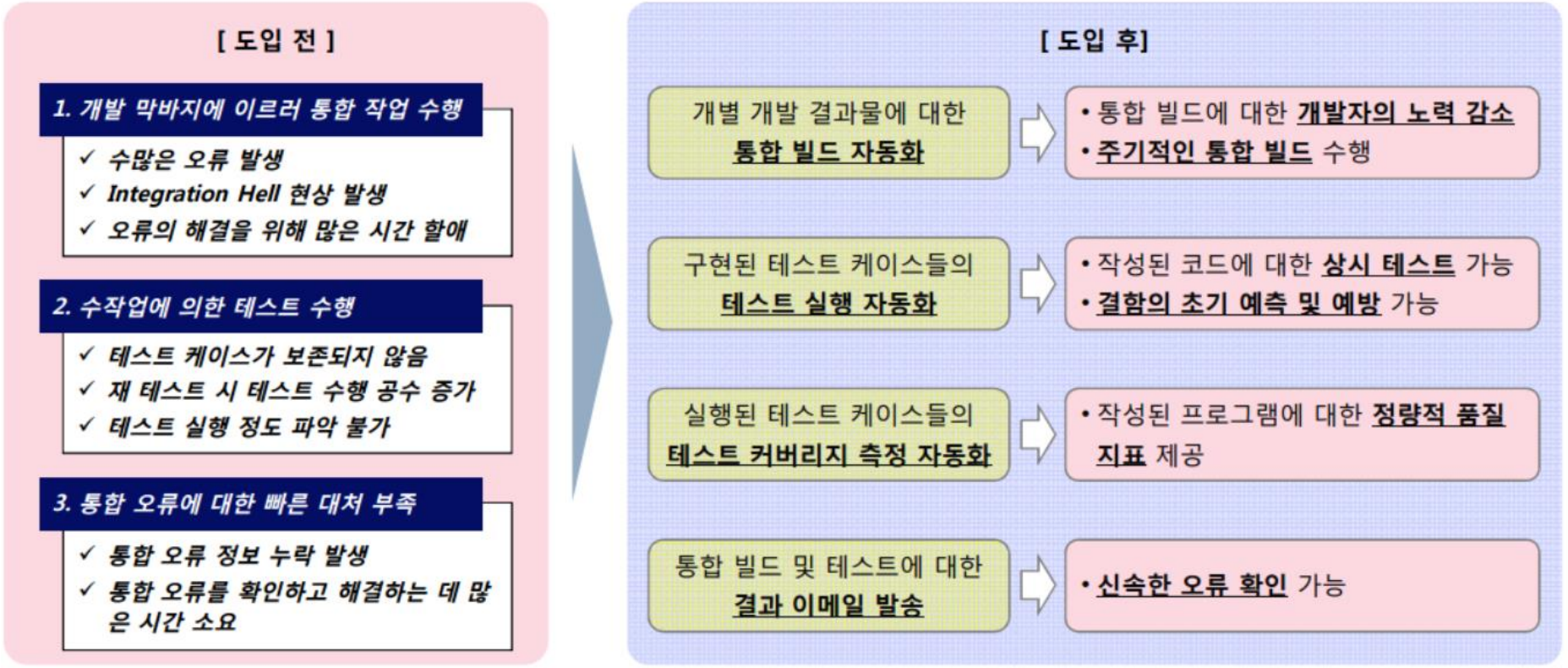
- 개별 단위 테스트는 수행되지만 테스트 케이스가 보존되지 않음
- 별도의 테스트 커버리지를 측정하지 않아 테스트 수행 정도를 가늠하기 어려움
- 회귀 테스트* 수행 시 테스트 케이스를 새로 작성해야 하기 때문에 막대한 공수를 필요로 함
  - * 회귀 테스트: 개발과 테스트를 완료한 모듈에서 오류 제거 및 수정과 같은 변경이 일어날 경우, 이러한 변경에 의해 유입된 오류가 없는지를 확인하는 일종의 반복 시험



[출처] C 환경에서의 CI(지속적통합) 구축 사례
/ JAVA 환경에서의 CI(지속적통합) 구축 사례

CTIP 장단점?

- [출처] JAVA 환경에서의 CI(지속적통합) 구축 사례

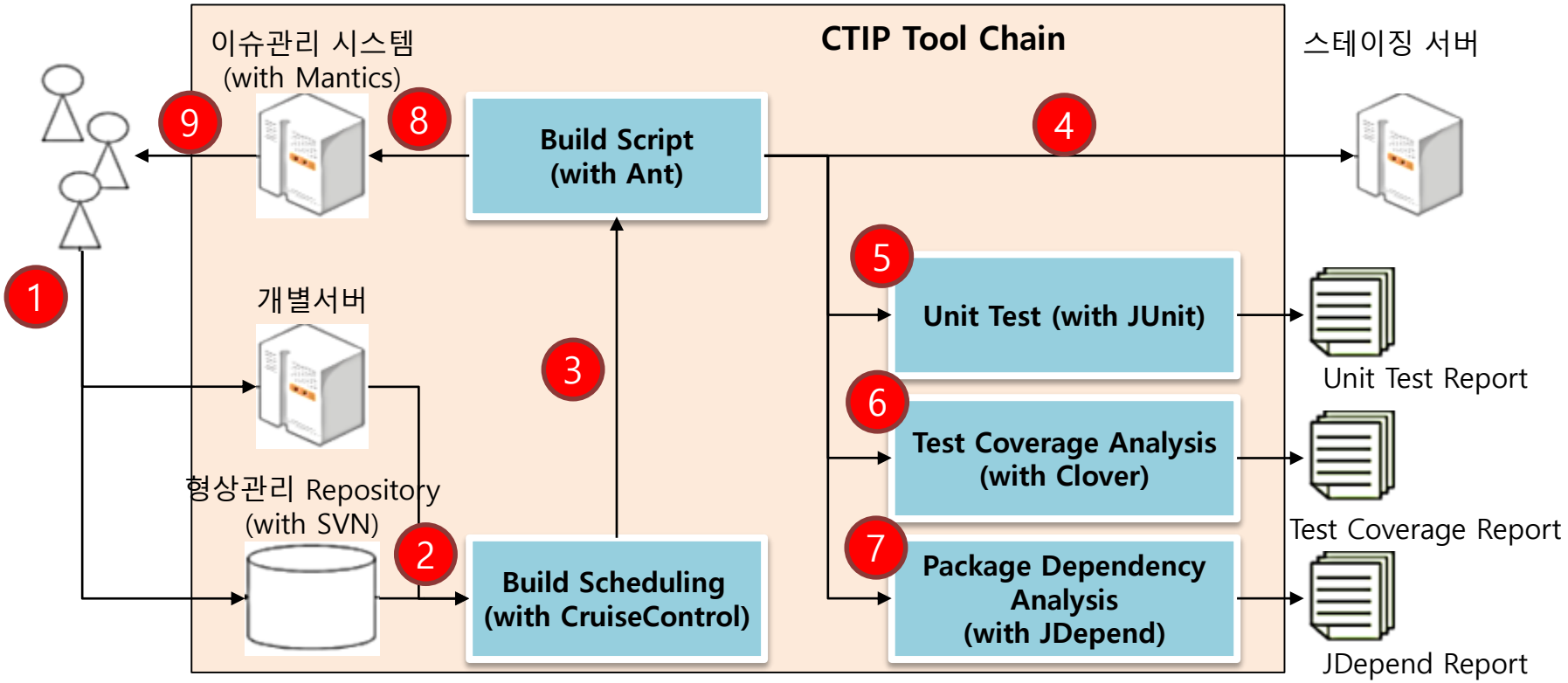


CTIP 장단점?

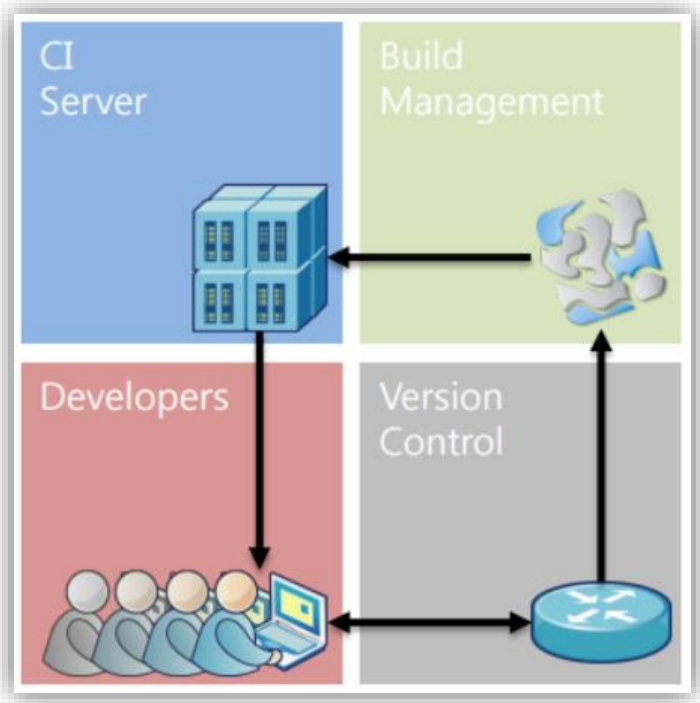
- 통합 소요시간 감소
- 개발자간 코드 충돌 조기 발견
- 통합 후 오류 발생류 감소
- 지속적인 통합을 통해 문제의 조기 발견
- 수동적인 반복 작업 감소
- 빌드 결과의 배포 및 관련자에게 통보(Feedback)
- 테스트 케이스 재사용으로 인한 개발 시간 단축
- 테스트 코드 커버리지 증가
- 품질 도구들을 통한 코드 품질 검토(테스트 및 정적 분석)
- 유지보수성 증가 (기능 변경, 결함 수 정에 대한 공수 감소)

- 초기 환경 구축의 어려움
- Well-developed test-suite이 필요 (테스팅 품질 향상을 위해)

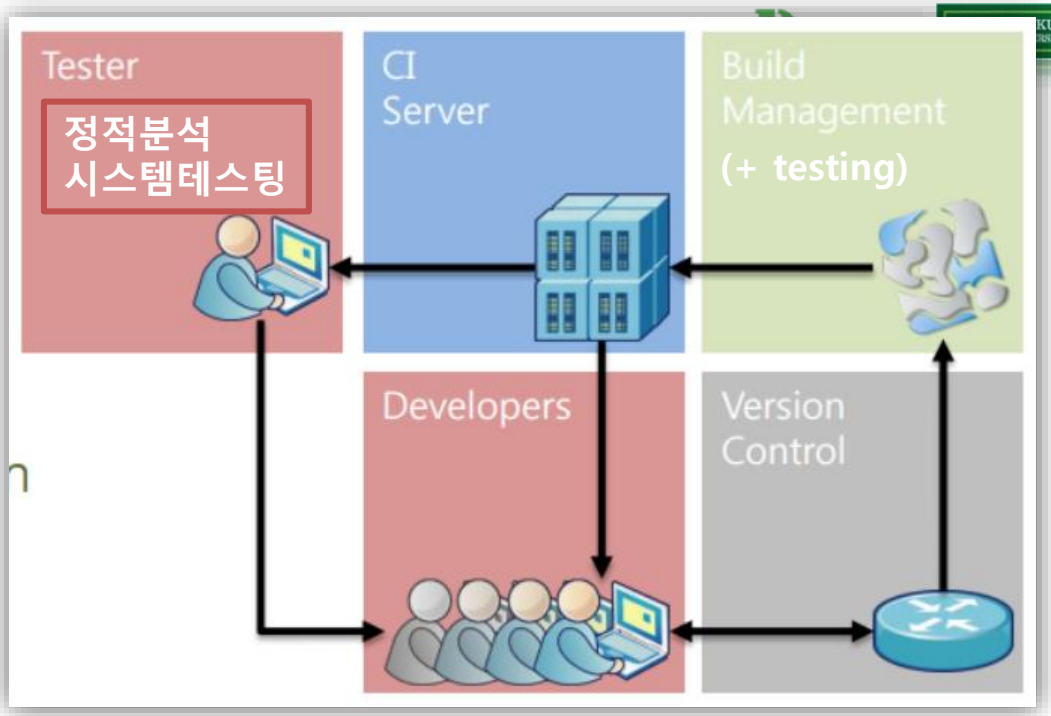
CTIP 구성도 예



1. 소스코드 개발 및 관리
2. 최신 소스코드 자동 체크아웃(업데이트)
3. 정해진 스케쥴링에 따라 빌드 요청
4. 빌드 자동 수행 및 배포
5. 빌드 시 단위 테스트 자동 수행
6. 빌드 시 커버리지 분석 자동 수행
7. 빌드 시 의존성 분석 자동 수행
8. 빌드 후 문제점 확인 및 등록 (통합 담당자)
9. 등록된 오류 확인 및 해결 (개발자)

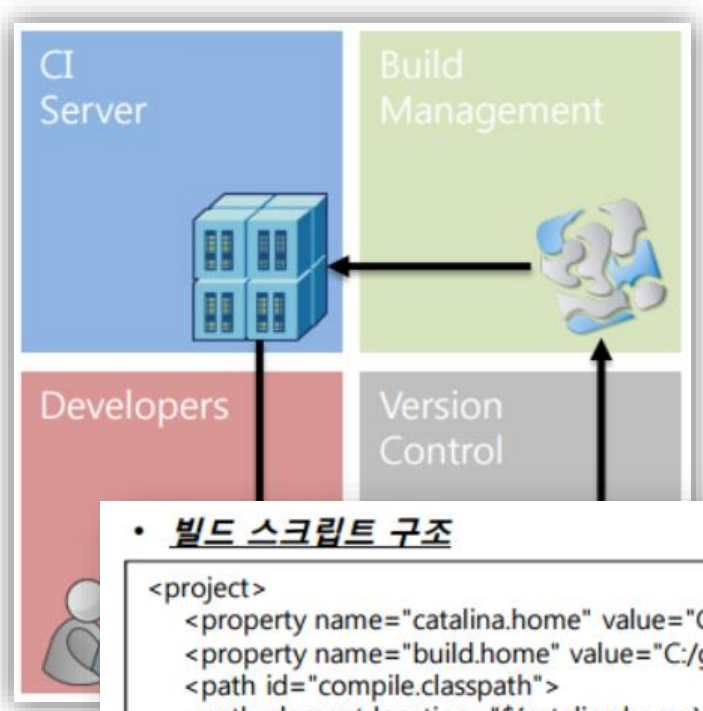


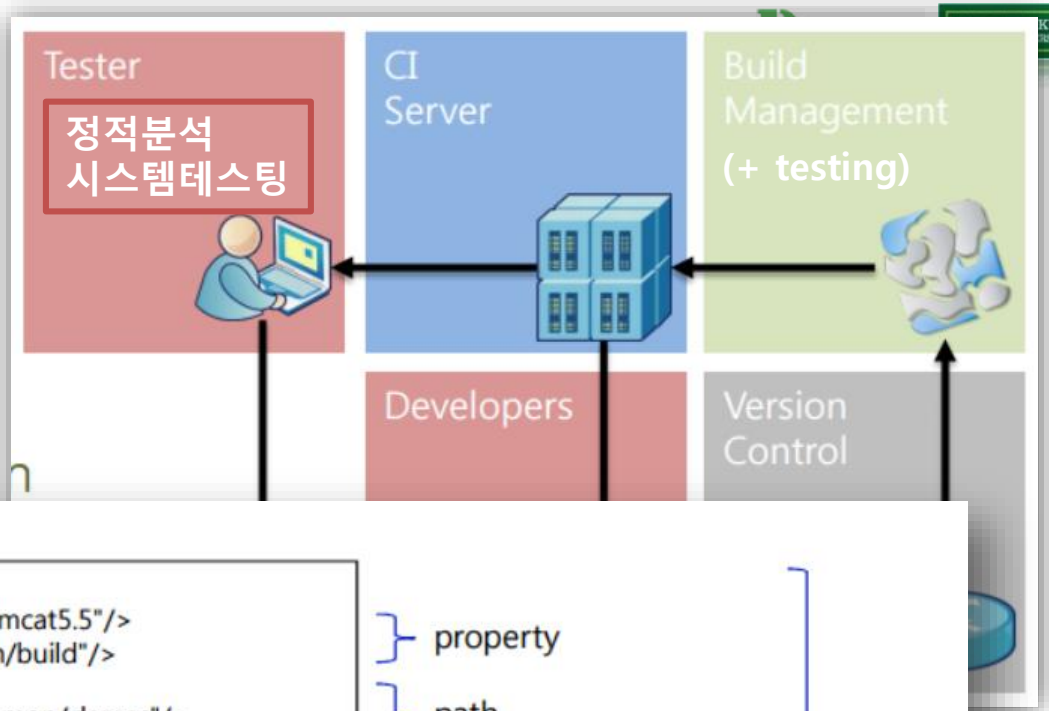
CI



CTIP

- CI 도구(CI Server)
  - 전체 Build 프로세스를 관리 - Ex) Jenkins
- 버전 관리 시스템(SCM)
  - 항상 동일한 최신의 베이스라인 코드를 가지고 작업 필요. - Ex) Git, SVN
- 빌드 도구(Build Tool)
  - 개발된 코드를 컴파일 과정을 거쳐서 서비스 가능한 형태로 만드는 빌드 과정을 수행 할 도구
  - Ex) Maven, Ant
- 테스트 도구(Test Tool)
  - 단위테스트, 통합테스트, 사용자 테스트, 회귀 테스트 등을 자동으로 수행 할 수 있는 도구
  - Ex) Junit





- **빌드 스크립트 구조**

```
<project>
    <property name="catalina.home" value="C:/Tomcat5.5"/>
    <property name="build.home" value="C:/graph/build"/>
    <path id="compile.classpath">
    <path element location="${catalina.home}/common/classes"/>
    </path>

    <target name="clean">
    <delete dir="${build.home}"/>
    </target>

    <target name="compile">
    <mkdir dir="${build.home}/classes"/>
    <javac srcdir ="${build.home}/src" destdir ="${build.home}/classes "/>
    </target>
    </project>
```

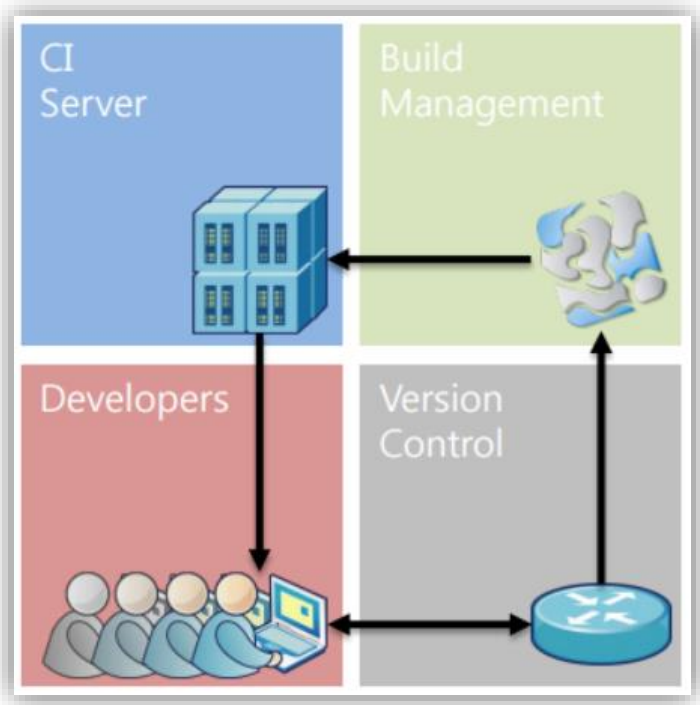
property
path
target
task
target
project

- CI
  -
- 버
  -
- 빌
  -
  - Ex) Maven, Ant
- 테스트 도구(Test Tool)
  - 단위테스트, 통합테스트, 사용자 테스트, 회귀 테스트 등을 자동으로 수행 할 수 있는 도구
  - Ex) Junit

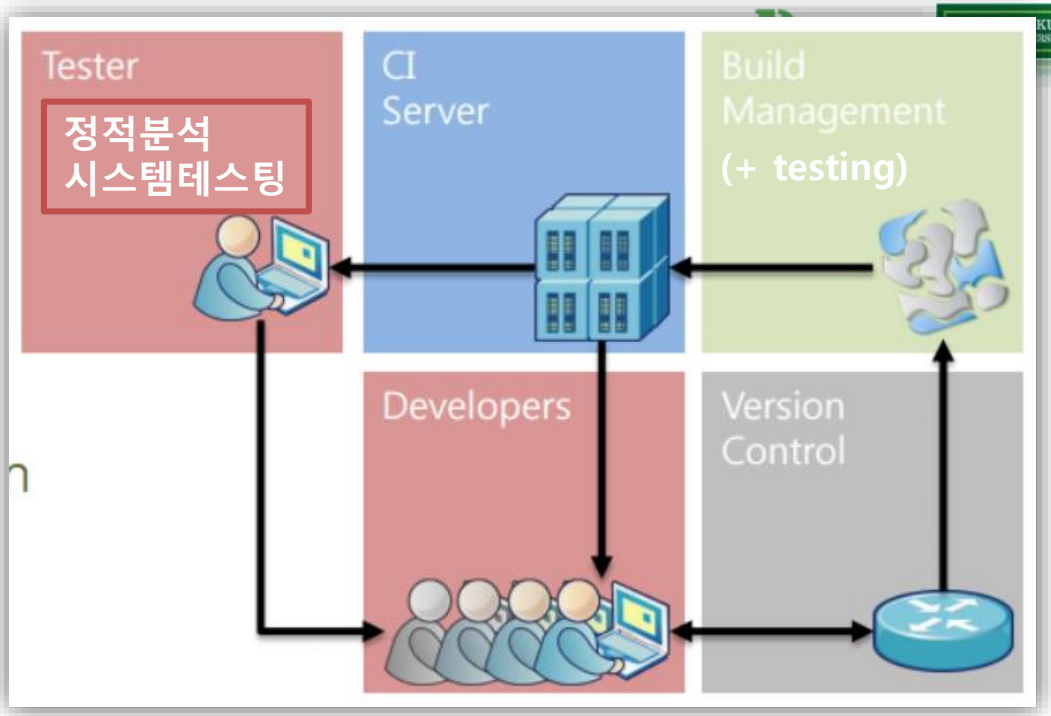
Tools?

Tool	Name
Project Management	OpenProj, GanttProject, Mylyn, Hsarepoint, ·····
Quality Management	Splint, PMD, JDepend, Checkstyle, ·····
Requirement Analysis & Management	Jfeature, JRequisite, ·····
Design & Modeling	StarUML, VioletUMLEditor, ArgoUML, ·····
Configuration Management & Version Control	Trac, Jenkins (Hudson), Maven, CVS, SVN, Bugzillia, Mantis, TortoiseCVS, Mercurial, Cruise Control, Git, Ant, RedMind, Berkshelf, Perforce, Nagios, Sensu, Vagrant, Foodcritic, ·····
Implementation	Eclipse, Valgrind, Netbeans, ·····
Testing	TestLink, CPPUnit, OProfile, HttpUnit, EMMA, Jmeter, Nunit, SoapUI, FitNesse, ·····
Static Analysis	Coverity, dart, sparrow, cppcheclipse, Resort, sonar, HP Fortify Static Code Analyzer, Checkmarx CxSuite, Syhunt Sandcat, Parasoft {Jtest, dotTEST, ···), VeraCode Static, Coverity Static Analysis,·····

Niapa: https://www.swbank.kr/helper/tool/toolMain.do



CI



CTIP

팀발표 #1	팀발표 #2	팀발표 #3	팀발표 #4	팀발표 #5	팀발표 #6
Junit , Eclipse , 및 빌드환경	Mantis , SVN , JFeature 및 CTIP	정적분석 도구 , Testlink 및 시스템테스트 도구	1st Testing - System Test	2nd Testing - System Test - Static Analysis	Final Presentation

2016 Software Verification

6 Team Practices

- TP#1(03.18): Junit , Eclipse , 빌드환경
- TP#2(04.01): Mantis , SVN 및 CTIP
- TP#3(04.15): 정적분석도구, Testlink 및 시스템 테스트 도구
- *TP#4(05.19): 1st System Testing*
- *TP#5(06.02): 2nd System Testing & Static Analysis*
- *TP#6(06.09) : Final Presentation - English*

8	04.21 / 04.22	Midterm Exam.	
9	04.28 / 04.29	이론 강의	Team Practice #4 - 각 팀별로 SMA에게 CTIP 환경 전수

5개팀 (19명 - 3/4/4/4/4)

Team Projects 2016 Software Verification

3학년 "소프트웨어 모델링 및 분석" 수업과 연동해서 진행합니다.

- 3학년 수업의 개발 결과물에 대해서 시스템 테스팅을 수행한 후 결과를 **Mantis**를 사용하여 공유합니다.
- 3학년 수업에서 단위 테스팅 및 관련 분석을 수행할 수 있도록 CTIP 환경을 제공합니다.
- 테스팅 결과에 대한 의견교환은 **Mantis** 등을 사용합니다.
  (다른 도구를 사용하셔도 됩니다.)
- 테스팅을 위한 소스코드는 **SVN**을 통해 공유합니다.
- NIPA SW 뱅크에서 제공하는 다양한 도구를 사용할 수 있습니다.