공개소프트웨어를 활용한 지속적인 통합 (CI)시스템 구축 및 테스트 방안

이상복 · 송기평 · 신석규

한국정보통신기술협회

A Study of Test Method and Implement Continuous Integration System using Open Source Tools

Sang-Bok Lee · Gi-Pyeong Song · Seck-Kyoo Shin

Software Quality Evaluation Center Telecommunications Technology Association

e-mail : [jangpo, gpsong, skshin]@tta.or.kr

요 약

소프트웨어 개발 프로젝트에서는 다양한 개발 방법론을 적용해서 소프트웨어가 개발되고 있으며, 최종 목표 소프트웨어 및 시스템에 따라 최적의 개발방법론이 적용한다. 개발프로세스 단계에 따라 모듈을 개발하며 최종적으로 통합 단계에서 하나의 소프트웨어로 통합하고 있다. 하지만 프로젝트가 복잡하고 모듈별 의존도가 높을 경우 각 모듈 별로 개발된 코드는 많은 요구사항의 변경, 형상관리 미흡 및 표준 미 준수 등으로 모듈 통합이 완벽하게 진행되지 않고 있으며 그로인해 프로젝트 기간 안에 완료하지 못하는 상황이 빈번하게 발생한다. 통합이 완료되지 못하면 소프트웨어 품질은 낮아지고 프로젝트 비용 및 시간은 늘어나 결국 프로젝트가 실패 할 가능성이 높아진다. 각 모듈의 통합을 예측 가능하고 성공적으로 하기 위해 지속적인 통합 시스템을 구축하여 프로젝트에 적극적으로 적용하고, 공개소프트웨어 도구를 활용하여 자동화 시스템을 구축해야 한다. 국내에서는 지속적인 통합방법의 중요성을 인식하고 있지만 프로젝트 비용. 시간, 인식부족 및 도구의 부재 등으로 인해 소프트웨어 프로젝트에 활용되고 있지 않은 실정이다.

이에 본고에서는 공개소프트웨어 도구를 이용하여 지속적인 통합 환경을 구축하고 소프트웨어 품질개선에 활용할 수 있는 테스트 방안을 제안한다.

ABSTRACT

In this paper propose open source software the tools to build and continuous integration environment that can be used to improve software quality testing

키워드

지속적인 통합, 개발 방법론, 공개소프트웨어, 소프트웨어 품질 테스팅

Ⅰ. 서 론

소프트웨어 개발 프로젝트에서는 폭포수 모델, 애자일 모델 등 다양한 개발 방법론이 적용되어 소프트웨어를 개발하고 있다. 각각의 개발 방법론은 단계별 순서와 방법 등이 다를 뿐 최종적인 목적은 소프트웨어 및 시스템을 개발하는 것이고 단계별로 개발된 모듈을 통합하여 최종적인 소프트웨어 및 시스템을 구축한다. 하지만 현실에서는 많은 프로젝트들이 통합 단계의 문제로 인해 프로젝트 기간이 연장되고 비용이 증가하여 결국 기간 안에 완료하지 못하는 현상이 발생한다. 또한 개발된 소프트웨어의 품질은 저품질 개발되어 진다. 왜 이런 문제가 발생하는 것일까 ? 각 모듈을 통합하는데 왜 문제가 발생하는 것일까 ? 프로젝트가 복잡해지고, 요구사항이 변경되고, 소스

코드는 변경되고, 모듈별 의존도가 높아지면서 모듈별 통합이 정상적으로 되지 않고 있기 때문이다. 이런 문제점을 해결하기 위해 생겨난 것이 지속적인 통합 방법이고 프로젝트 관리자, 개발자, 테스터 등 관련된 모든 사람들이 기준으로 삼고 있어야 하는 내용이다. 하지만 지속적인 통합 방법을 적용하기 위해서는 많은 것들이 준비되어야 한다. 또한 지속적인 통합 방법에서 소프트웨어 품질을 향상 시킬 수 있는 테스트 방법들을 찾아 적용해야 한다. 이에 본고에서는 지속적인 통합 방법을 통해 프로젝트를 성공적으로 수행하고 품질 높은 소프트웨어 및 시스템을 개발하는 방법을 제시하고자 한다.

제 2장에서는 지속적인 통합에 대한 정의를 설명하고 제 3장에서는 통합에 사용할 수 있는 공개소프트웨어 도구를 소개하고 4장에서는 지속적인 통합 시스템을 구축 및 테스트 방안에 대해 기술하고 제 5장에서는 결론을 제시하였다.

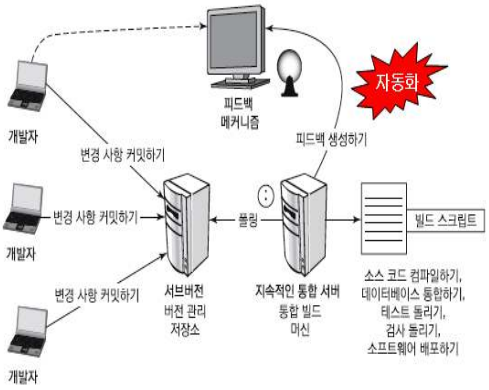
Ⅱ. 지속적인 통합(Continuous Integration)

2.1 지속적인 통합 소개

지속적인 통합은 팀 구성원들이 자신이 한일을 자주 통합하는 소프트웨어 개발 실천 방법이다. 프로젝트에서 소프트웨어를 개발하다 보면 많은 모듈별로 이루어지고, 통합 단계에서 모듈을 하나로 통합하여 전체 프로그램이 완성되게 된다. 하지만 개별적으로 개발된 모듈을 통합하는 것은 어려움이 존재하며 제품의 품질을 떨어뜨리는 원인이기도 하다. 정의 하자면 소프트웨어에 연관된 모듈을 통합할 때 통합단계에서 단번에 완수하는 것이 아니라 일정 기준에 맞춰 모듈이 개발되면 지속적으로 모듈 통합을 거쳐 소프트웨어를 개발하는 것이라 보면 된다. 그리고 이런 통합 단계를 수동적으로 수행하게 되면 시간 및 비용의 문제가 발생하고 소스코드에 대한 형상관리가 어려우므로 자동화 도구를 이용한 지속적인 통합 시스템을 구축하는 것이 좋다[1].

2.1.1 지속적인 통합 시스템

지속적인 통합 시스템은 개발자가 개발한 소스코드를 통합적으로 자동 빌드하는 시스템이다. 그러므로 소스코드에 대한 표준코딩, 형상관리, 통합 빌드, 피드백 등이 꼭 필요하다. 현재 상업적으로 판매되는 제품도 있지만 Hudson, Cruise Control등과 같은 공개소프트웨어로 개발된 통합 서버들이 존재하며 많은 프로젝트에서 활용되고 있다. 또한 통합 시스템을 구축 시 소프트웨어의 품질을 높이기 위해 자동 테스트 도구가 연동되어 사용되고 있으며 PMD, JUnit 등과 같은 공개소프트웨어 도구가 사용되고 있다. 아래 [그림1]은 지속적인 통합 시스템을 보여준다.



[그림1] 지속적인 통합 시스템

위 [그림1]에서 보듯이 지속적인 통합 시스템은 절차가 존재하고 자동화 도구들이 유기적으로 구성된다. 형상관리 도구, 빌드 도구, 통합 도구, 테스트 도구 등이 있다.

지속적인 통합을 활용하여 소프트웨어를 개발하면 아래와 같은 이익을 발생한다.

- 소프트웨어 위험요소 감소
- 반복적인 수작업 감소
- 언제든 배포 가능한 소프트웨어 생성
- 프로젝트의 가시성 증가
- 개발팀의 프로젝트에 대한 자신감 증가

또한 지속적인 통합을 실천하기 위한 개발자의 역할은 다음과 같다.

- 코드를 자주 커밋하기
- 깨진 코드 커밋 금지
- 빌드 실패시 즉시 수정
- 자동화된 테스트 케이스 작성
- 테스트와 검사 무조건 통과
- 개인 빌드 수행
- 깨진 코드 버리기

Ⅲ. 지속적인 통합 도구

지속적인 통합 자동화 시스템을 구축하기 위해는 단계별로 도구가 사용 된다. 소스코드를 관리하기 위한 형상관리 도구, 소스코드를 빌드하기 위한 빌드 도구, 통합적으로 관리하는 통합도구, SW품질을 위한 테스트 도구 등이 있다. 상용도구를 사용할 경우 비용이 발생하기 때문에 공개소프트웨어 도구를 많이 사용하고 있는 실정이며 각 단계에서 사용되는 공개소프트웨어 도구들도 많은 검증을 거쳐 안정적으로 운영됨을 확인되어 많은 프로젝트에서 사용되고 있다.

지속적인 통합 시스템의 도입을 통해 정리한 위험 요소, 자동화 프로세스 및 자동화 도구의 연관관계에 따라 어떻게 구성되는지 아래 [표1]에서 보여준다.

[표 1] CI 연관 관계

위험요소	자동화	도구
o 배포 불가능 - 빌드 실패 - DB 동기화 실패 - 배포 실패	- 소스 컴파일 - DB Script 테스트 - 릴리즈	- ANT - Maven - Subversion
o 늦은 결함 발견 - 회귀 테스트 부재	- 테스트 - 커버리지 측정	- JUnit - TestNG - Selenium - Coberura
o 프로젝트 가시성 부재 - 자동화 피드백 부재	- 통합결과 피드백 - 이슈 관리	- Hudson -CruiseControl
o 저품질 SW - 코딩표준 미준수 - 설계지침 미준수 - 중복코드 - 복잡한 코드	- 코딩표준 검사 - 설계지침 검사 - 중복코드 검사 - 코드복잡도 검사	- Checkstyle - PMD - Findbugs

Ⅳ. 시스템 구축 및 테스트 방안

공개소프트웨어를 이용하여 지속적인 통합 시스템을 구축해 보고 자동화 테스트 도구를 활용하여 소스코드 및 소프트웨어의 품질을 향상시키는 방법을 제시한다.

o 시스템 환경 및 활용 도구

지속적인 통합 시스템을 구축하기 위해서는 필수적으로 필요한 자동화 도구가 존재한다. 기 구축한 시스템에서는 개발도구, 형상관리 도구, 테스트 도구, 통합관리 도구, 빌드 도구 등을 사용하였다

가. 개발도구 : eclipse(java)
나. 형상관리 도구 : SubVersion
다. 빌드도구 : ANT, Maven
라. 통합 도구 : Hudson
마. 테스트 도구 : JUnit, PMD, CheckStyle, Sonar

[그림2] 시스템 구축 화면

o 실천 방법

지속적인 통합 시스템을 자동화로 구축했다고 하여도 지속적인 통합 방법이 성공적으로 프로젝트에 적용될 수는 없다. 아래 실천 방법을 적용해야 최소의 효율과 효과를 볼 수 있다.

- 빌드를 자동화 하기
- 명령어 하나로 빌드를 수행하기
- IDE에서 빌드 스크립트 분리하기
- 형상관리 도구 사용하기
- 일관성 있는 디렉토리 구성하기
- 일일 빌드 하기
- 빌드 시간을 짧게 하기
- 빌드를 여러 단계로 나누기
- 어떤 환경에서도 빌드하기

o 테스트 방안

지속적인 통합 개발 환경에서는 개발자의 품질에 대한 의식이 중요하다. 대부분의 도구들이 개발자들이 직접 사용하는 테스트 도구들로 이루어 있으며 개발 소스와 연관되어 있다. 자동화 도구을 사용하게 되면 표준코드 준수여부, 소스코드 형상관리, MC/DC 등의 정보를 직관적으로 확인할 수 있다. 또한 FindBug, Simian, Javadoc 등과 같은 공개 소프트웨어로 제공되는 자동화 테스트 도구를 연동하여 결함을 사전에 발견 품질을 높이는 방법을 도입해서 소스코드의 품질을 향상시켜야 한다.

Ⅴ.결 론

본 논문에서는 지속적인 통합 방법에 대해 소개하고 통합 시스템 구축에 필요한 자동화 도구에 대해서 알아보았다. 또한 공개소프트웨어를 이용하여 지속적인 통합시스템을 구축하였으며 자동화 도구를 연동하여 품질을 높이는 테스트 방법을 제안하였다. 앞으로 더 많은 소프트웨어 개발 프로젝트에서 품질을 한 단계 높이는 지속적인 통합 방법이 적용되기를 바란다.

참고문헌

[1] 폴 M 듀발 “지속적인 통합(소프트웨어 품질을 높이고 위험을 줄이기”, 위키북스, ISBN 978-89-92939-13-2S