Unified Change Management(UCM)을 이용한 Embedded Software 개발

조현[0] 김은영
삼성전자 CTO전략실 소프트웨어센터
(hcho, marine)@samsung.com

Embedded Software Development using Unified Change Management

Hyun Cho[0] EunYoung Kim
Software Center, CTO, Samsung Electronics

요 약

Embedded System이 고도화, 지능화되어감에 따라 이를 구동하는 Embedded Software의 역할과 크기 그리고 복잡도 역시 증가하고 있어 주어진 개발 기간동안 안정되고 품질이 보장된 Embedded Software 개발을 위해서는 기존의 방법보다 간편하고 유연하며 개발 프로세스를 지원하는 새로운 변경 관리 기법과 병렬 환경 구성이 방법이 필요하게 되었다. 이에 따라, 기존의 방법을 추상화하여 사용과 관리가 용이하며 프로세스 지원이 가능한 Unified Change Management(UCM)을 Embedded Software인 Handset Platform 개발에 적용하여 병렬 개발 환경 구축과 Change Control and Problem Tracking 측면에서 기존의 방법과 비교하여 그 효용성을 살펴보고자 한다.

1. 서 론

현재의 Embedded System은 고도화 지능화가 급속히 추진되고 있으며 이에 따라 Embedded Software 개발은 비즈니스 생존의 핵심이 되었다. 많은 개발자들은 증가하는 Embedded Software의 복잡도와 개발 라이프사이클의 단축이외에도 여러 내적/외적인 요인들로부터 품질이 보장된 Embedded Software 개발에 많은 어려움을 겪고 있으며 개발자 혼자의 능력으로 Embedded Software 개발의 모든 책임을 떠 안는 것은 이미 지난 일이 되었다. 이러한 문제들을 해결하기 위하여 많은 조직에서는 Embedded Software 개발을 위한 개발 프로세스를 도입하고 있으며 특히 품질을 예측하고 관리하기 위한 품질 보증 활동과 변경 관리에 많은 인력, 시간, 그리고 자금을 투자하고 있다.

Embedded Software는 소형 가전 제품에서부터 PDA, 휴대폰, 의료 기기, 우주 항공에 이르기까지 그 적용 범위가 넓고 크기와 복잡도 역시 적용 범위에 무관하게 날이 다르게 증가하고 있다. 또한, Embedded Software의 결함에 의해 발생된 문제는 그 결과를 예측하기 힘들며, 문제가 발생하였을 경우 인간의 생명에 치명적인 경우가 발생하게 된다.

그러므로, 안정되고 품질이 보장된 Embedded Software를 개발하기 위해 전체 개발 생명 주기 동안에 Software의 형상을 식별하고 형상 변경에 대한 관리를 체계적이고 계획적으로 수행하며 변경에 대한 추적성을 관리하는 것이 중요한 요소로 대두되었다[1].

따라서, 본 논문은 Embedded Software 개발을 위한 변경 관리 기법으로 Unified Change Management (UCM)을 적용하여 병렬 개발 환경 구축과 Change Control & Problem Tracking 측면에서 기존의 변경 관리 기법과 비교해 보고자 한다.[2][3].

2. Project 개요

UCM을 적용한 프로젝트는 휴대폰에서부터 PDA까지 다양한 하드웨어와 Embedded 운영체제를 지원할 수 있는 Handset Platform 개발 프로젝트로서 이는 Real Time운영 체제와 Embedded Device에 독립적이며 Target에서 실행 가능한 응용 프로그램 개발을 위한 API 제공이 가능한 Platform 개발을 그 목적으로 한다. 본 프로젝트에서 정의한 개발 범위는 표 1과 같다. 표 1에서 보는 것과 같이 본 프로젝트는 Target 의 범위가 넓고, 다양한 Component가 존재하며, Component의 개발 일정과 참여 인력이 유동적이어서 품질이 보장된 Embedded Software 개발을 위해 간편하면서 신뢰할 수 있는 병렬 개발 기법과 구현 단계와 테스트 단계에서의 변경 관리가 절실히 필요한

프로젝트이다.

표 1 Target Hardware

구분	최소 사양	최대 사양
Target	CDMA Phone	PDA
CPU	ARM	Dual CPUs
Display	B/W LCD	Color LCD
Memory	4M	8M
Component	12 개	12~
참여 인력	10명	40명
개발 기간	1~2년	

3. 병렬 개발 환경

병렬 개발은 하나의 Software에 대해 동시에 여러 개발자들이 개발을 진행해 나가며 각 개발자들별로 그 Software에 대한 버전 관리를 수행해 가나가는 것을 의미하며 이는 현재 변경 관리 Tool을 선택할 때 중요한 요소이며, 이러한 병렬 개발에는 두 가지 상충되는 요구 사항이 존재한다.

첫째, 각 개발자들은 다른 개발자들의 변경에 영향을 받지 않으면서 독립적으로 자신에게 주어진 기능을 구현할 수 있어야 한다. 둘째, 각 개발자들은 다른 개발자들이 구현하거나 변경한 부분을 자신이 개발하는 소스에 반영되어 개발을 진행해 나갈 수 있어야 한다.

첫번째 요구 사항은 Branch로 해결할 수 있다. 기존의 Branch 기법을 이용한다면 최소한 2~5개의 기능을 수행하여야 한다. 또한, 개발자들은 Branch 개념과 기능을 파악하여야 하고 Branch 생성, 유지, 삭제에 대한 책임과 권한을 가지게 된다. 그러나, UCM에서는 Branch 생성과 유지를 추상화하여 개발자들이 Branch에 대한 개념이 미흡하더라도 프로젝트 관리자가 설정한 Baseline에 개발자는 단순히 Join Project라는 기능을 한번 수행함으로써 자신을 위한 Branch가 생성되며 다른 개발자의 변경에 영향을 받지 않고 개발을 진행할 수 있다.

두번째, 요구 사항인 개발자 서로간의 변경된 부분을 반영하는 것 역시 기존의 방법에서는 Merge를 이용하여 복잡하게 얽힌 Branch 사이에서 신중히 수행하여야 하는 매우 성가신 문제였으나 UCM에서는 Branch 생성과 마찬가지로 Merge 기능을 추상화하고 개발자들이 변경한 부분을 반영하는 것을 그림 1과 같이 프로세스로 구현한다.

자신의 Branch에 저장된 변경 사항을 다른 개발자들에게 반영시키기 위해 Main Branch로 변경을 Delivery하고, 다른 개발자의 변경을 나에게 반영하기 위해서는 Main Branch의 새로운 Baseline에서 Rebase 한다.

UCM의 이러한 추상화된 병렬 개발 환경 구축 기능은 기존의 방식에 비해 간단하여 개발자들이 병렬 개발 환경 구축에 들이는 노력을 경감시킨다.

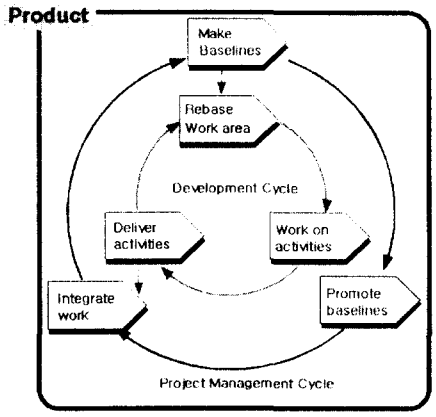

그림 1 추상화된 Merge 프로세스

병렬 개발 환경 구축측면에서 UCM을 적용할 때 그림 2에서와 같이 UCM을 적용하지 않았을 때 보다 병렬 개발 환경 구축을 위해 필요한 학습 시간을 50%정도 단축시킬 수 있어 UCM이라는 새로운 기법 적용으로 인한 개발 지연은 최소화할 수 있다.

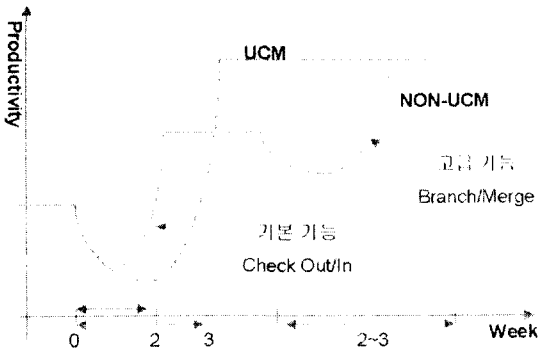

그림 2 Learning Curve

4. Change Control & Problem Tracking

Embedded Software 개발시 결함을 수정하였음에도 불구하고 다음 Iteration이나 Target 변경으로 인해 동일한 문제가 다시 나타나는 경우가 종종 발생함으로 기존의 수정된 부분과 현재 상황을 비교하여 그 차이점을 파악할 필요가 있다. 따라서, 변경이 발생할 때마다 누가, 언제, 무엇을, 어떻게, 변경하였는가, 그리고 누가

그 변경에 대해 승인을 하였나를 관리할 필요가 있다.

또한, 변경은 테스트 단계에 이르러서야 발견되는 결함에 의해서 뿐만 아니라 구현 단계에서의 기능 추가, 변경, 삭제 등에 의해서도 변경이 발생하게 된다. 그러므로 테스트 단계뿐만 아니라 구현 단계에서의 변경도 관리되어져야 하나, 기존에는 테스트 단계에서 발생한 결함에만 초점이 맞춰져 있어다. 그러나, UCM에서는 Activity라는 개념을 도입하여 구현 단계에서 발생하는 변경을 BaseCMActivity(Base Change Management Activity)로, 테스트 단계에서는 결함으로 관리한다.

그림 3과 4는 각 Activity 처리를 위해 적용한 프로세스이다. 변경 관리를 Activity를 근간으로 프로세스화 함으로써 구현 단계에서 발생하는 변경을 프로젝트 관리자가 개발자별 기능별로 관리할 수 있게 되어 일정 관리나 기능 구현에 따른 위험 관리가 구현 단계에서 가능하여졌다. 또한 테스트 단계에서 Target의 상이함에 의해 반복되어 발생되는 결함에 대해 이전의 해결 방법을 검색하여 그 해결 방법을 적용함으로써 결함에 대한 처리 시간을 단축시킬 수 있었다.

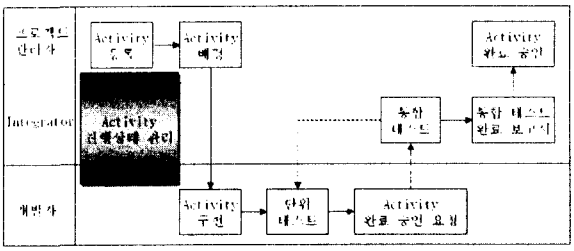

그림 3 Change Activity Process

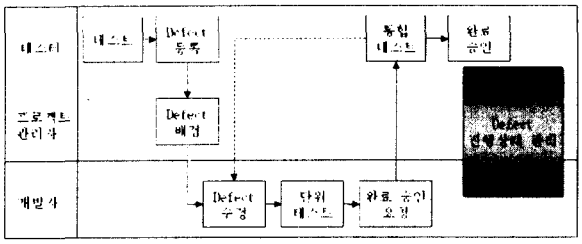

그림 4 Defect Activity Process

5. 결 론

UCM을 Target의 범위가 넓고, 다양한 Component가 존재하며, Component의 개발 일정과 참여 인력이 유동적인 Embedded Software인 Handset Platform 개발에 적용하였다.

병렬 개발 환경 구축 측면에서 UCM은 기존의 변경관리 기법을 추상화함으로써 병렬 환경 구축을 위한 기능을 간소화 시키고 프로세스화 하였다. 이렇게 간소화되고 프로세스화된 병렬 환경 구축으로 모든 개발자들은 다른 개발자의 영향을 받지 않고 자신이 개발하는 기능을 구현할 수 있었으며, Delivery, Baseline, Rebase를 통해 기능이 검증되고 안정적인 소스를 자신의 개발 환경으로 가지고 올 수 있어 다른 개발자의 오류로 인한 개발 지연이 최소화 되었다. 또한, 이러한 병렬 개발 환경을 구축하기 위해 필요한 학습 시간을 기존의 방법에 비해 50%이상 감소시켜 새로운 환경 적용으로 인한 과다한 학습 시간이 필요하지 않게 되었다. .

구현 단계와 테스트 단계에서 발생한 모든 변경 원인을 모두 Activity로 생성, 유지, 관리함으로써 프로젝트 관리자는 일정 관리는 물론 Embedded Software와 같이 Embedded 환경과 하드웨어에 의한 예기치 못한 변수로 인하여 기능 구현이 지연될 때 지연 원인을 보다 빨리 파악할 수 있어 일정 지연 또는 기술적 문제에 따른 위험 관리가 가능하여졌다.

Unified Change Management(UCM)의 이러한 추상화되고 프로세스화된 병렬 개발 환경과 구현 단계에서부터 테스트 단계까지 프로세스화된 변경 관리로 인하여 품질이 보장된 Embedded Software인 Handset Platform 개발이 가능 하였다.

6. 참고 문헌

1. Dart, S.A. 1991, *Concepts in Configuration Management System*, Proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, June, 1991, ACM

2. *Working with UCM*, Rational

3. *Managing Software Projects with ClearCase*, Rational