소규모 조직의 소프트웨어 프로세스 구현을 위한 이클립스 플러그인의 통합 모델 개발

도성룡*, 한혁수**, 이상은***, 이혁재***, 배문식***
*상명대학교 일반대학원 컴퓨터과학과
**상명대학교 컴퓨터과학부
***한국소프트웨어진흥원
e-mail : imdsr@smu.ac.kr

Developing Integrated Model of Eclipse Plugins for Software Process Implementation of Small Organizations

Sung Ryong Do*, Hyuk Soo Han**, Sang Eun Lee ***, Hyuk Jae Lee***, Moon Sik Bae***
*Dept. of Computer Science, SangMyung University
**School of Computer Science, SangMyung University
***Korea SW Industry Promotion Agency

요 약

소프트웨어 프로세스는 소프트웨어와 이에 관련된 산출물을 개발, 유지하기 위해 사용하는 활동, 방법, 절차의 집합이라고 할 수 있다. 프로세스를 기반으로 작업하는 조직은 필요한 프로세스들을 파악하고, 각 프로세스들을 구현하기 위해, 담당자를 할당하고, 수행 활동들을 정의한 후, 이를 기반으로 작업을 수행한다. 이 때 보다 효과적으로 작업하기 위해 적절한 도구들을 활용하기도 한다.

소프트웨어 개발에서 도구의 활용은 이미 그 효과가 검증되었고, 많은 상업용 제품들이 개발되어 현장에서 사용되고 있다. 이러한 도구들 중에는 독자적으로 하나의 프로세스를 지원하는 독립형 (Standard Alone) 도구들과 여러 프로세스를 지원하는 통합형 도구들이 있다. 통합형 도구들은 여러 프로세스를 연결하고 통합 관리하기 때문에 효과가 크지만, 주로 가격이 비싼 상업용 제품들이고, 대규모 프로젝트에 적합한 복잡한 기능이 많아 소규모 조직이 채택하기에는 어려운 경향이 있다. 독립형 도구들은 통합형 도구보다 상대적으로 기능이 복잡하지 않고, 공개 소프트웨어로도 제공되고 있기 때문에 소규모 조직들도 사용해 왔지만 통합형 도구와 같은 효과를 내기는 쉽지 않았다.

본 논문에서는 이클립스 플랫폼 기반에 독립형으로 존재하는 플러그인들을 통합하여, 여러 프로세스를 지원하는 이클립스 플러그인 모델을 개발하고, 그 효과를 살펴보았다.

1. 서론

소프트웨어의 요구사항이 복잡해지고 규모가 커지면서 정해진 기간 내에 고품질의 소프트웨어를 개발하는 것은 점점 더 어려워지고 있다. 과거에는 이러한 문제를 기술의 향상과 우수 인력 확보만을 통해 해결하려고 하였다. 그러나 최근에 들어서 대형 프로젝트를 성공시킨 조직들의 우수 사례(Best Practice) 분석을 통해 소프트웨어 프로젝트 성공에는 프로세스도 중요한 역할을 한다는 것이 인식되었다.

CMM/CMMI 의 기초를 만든 Watts S Humphrey 는 " 소프트웨어 제품의 품질은 그 제품을 만들기 위해 사용된 프로세스의 품질에 의해 결정된다." 라고 프로세스의 중요성을 강조하였다. 현재는 많은 조직에서 소프트웨어 개발에 효과적인 프로세스를 도입하기 위해 노력하고 있다.

CMM 에서는 소프트웨어 프로세스를 소프트웨어의 생산 및 진화에 사용되는 활동, 방법 및 실무 활동들의 집합이며, 원하는 산출물을 생산하기 위해 인력, 절차, 방법, 장치 및 도구들을 통합하는 수단으로 정의하고 있다.

소프트웨어 생명주기 프로세스 표준인 ISO/IEC 12207 은 크게 기본 생명주기 프로세스, 지원 프로세스 그리고 조직 생명주기 프로세스로 분류할 만큼 소프트웨어 개발에는 많은 프로세스들이 존재한다. 각각의 프로세스들을 정의하기 위해, 조직들은 프로세스의 담당자를 정하고, 절차를 만들고, 그에 필요한 산출물 양식도 정의한다[1].

이러한 프로세스의 구현은 간단한 문서도구를 활용하여 진행할 수도 있지만, 자동화 도구를 사용하여 담당자의 부담을 줄여주기도 한다[2].

도구의 활용은 CASE(Computer Aided Software Engineering)도구가 활발히 도입되던 1970 년대부터 시스템 분석, 설계, 코딩 및 유지보수에서 많이 이루어져 왔다[3].

IDE(Integrated Development Environment)와 같이 코딩과 디버깅을 지원하여 개발자들에게 없어서는 안될 기능들을 제공해왔고, UML(Unified Modeling Language) 지원 도구들을 통해 객체지향 패러다임에서의 도구의 활용이 이루어져 왔다.

이러한 도구들의 사용은 일관성 있는 체계를 제공하여, 참여자들간에 소통을 원활히 하고, 도구 내에서 문서를 관리해줌으로 유지보수를 용이하게 하는 장점을 가지고 있다.

이러한 도구들 중에는 독자적으로 하나의 프로세스를 지원하는 독립형 도구들과 여러 프로세스를 지원하는 통합형 도구들이 있다.

통합형 도구들은 여러 프로세스를 연결하고 통합관리하기 때문에 효과가 크지만, 주로 상업용 제품들이라 가격이 비싸고, 기능이 많아 소규모 프로젝트를 주로 하는 조직에서는 채택하기에 어려움이 많았다.

독립형 도구들은 특정 프로세스만을 지원하기 때문에 기능 규모가 크지 않아, 공개 소프트웨어로도 제공되고 있어서 소규모 조직들도 채택하여 사용해 왔다. 하지만 통합형 도구와 같은 효과를 내기는 쉽지 않았다.

이클립스 플랫폼은 독립형으로 동작하는 기존의 도구들을 플러그인 형태로 통합할 수 있도록 설계된 개발 환경이다. 하나의 플러그인이 다른 플러그인들과 쉽게 연계될 수 있도록 설계되었기 때문에, 플러그인들 간의 통합이 용이하다[4][5].

소규모 조직이 프로세스를 정립하고 이를 지원하기 위한 플러그인들을 선택하여, 이클립스 플랫폼 기반의 통합환경을 구축하면, 그 조직과 프로젝트에 적합한 통합형 도구를 갖추는 것과 유사한 효과를 나타낼 수 있다.

이를 증명하기 위해 본 논문에서는 요구사항 개발 도구인 JFeature 와 테스팅 도구인 JUnit 그리고 형상관리 도구인 Subclipse 를 활용하여 통합환경을 구축하고, 그 효율성을 입증하였다.

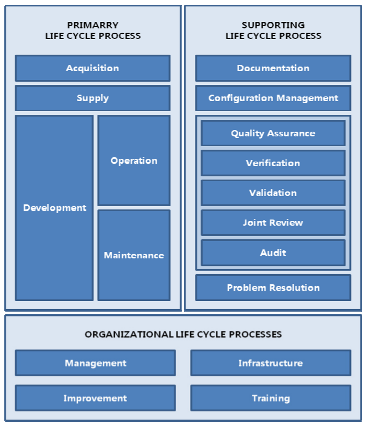
2. 관련연구

2.1 소프트웨어 프로세스

소프트웨어 프로세스는 소프트웨어와 이에 관련된 산출물을 개발, 유지하기 위해 사용하는 활동, 방법, 절차의 집합이라고 할 수 있다.

소프트웨어 개발 조직들은 소프트웨어 개발을 일관성 있게 수행하고, 제품의 품질을 유지하기 위해 소프트웨어 프로세스를 정의하고 프로젝트들로 하여금 정해진 프로세스에 따라 개발을 진행하도록 한다.

소프트웨어 개발을 위해서는 많은 프로세스들이 정의되어야 하는데, ISO/IEC 12207 에서는 소프트웨어 개발 및 관리에 적용될 수 있는 프로세스들을 (그림 1)과 같이 정의하고 있다.



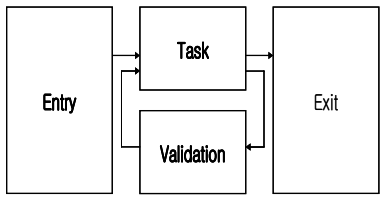
(그림 1) ISO/IEC 12207 표준의 생명주기 구조[6]

전체 프로세스를 기본 생명주기 프로세스, 지원 프로세스 그리고 조직 생명주기 프로세스로 분류하고, 각 프로세스는 하나 이상의 액티비티들로 이루어지며, 이들 액티비티는 하나 이상의 태스크들로 이루어진다.

CMMI 에서는 총 22 개의 프로세스 영역을 2 단계부터 5 단계까지 각 단계별로 이루어야 하는 목표에 맞게 할당하였다[7].

2.2 프로세스의 구현

조직들이 프로세스를 구현하기 위해서는 각 프로세스마다 (그림 2)의 ETVX(Entry-Task-Validation-Exit) 프로세스 모델을 이용하여 프로세스 내의 태스크를 중심으로 시작 조건과 종료 조건 등을 명시하고, 작업 흐름을 정의한다.



(그림 2) ETVX 프로세스 모델 [8]

2.3 소프트웨어 공학 도구

소프트웨어 프로세스를 수행하는데 있어서 도구의 활용은 작업의 품질과 생산성을 향상시키고자 하는 노력의 일환이다.

전체 소프트웨어 개발 프로세스에 사용되고 있는 도구들을 분류하면, 다음과 같다.

- 프로젝트 관리도구: 프로젝트를 효율적으로 진행하기 위해 일정 및 자원을 체계적으로 관리하고, 위험요소를 예측하여 해결할 수 있도록 도와주는 도구이다. 전체 프로젝트를 지원하는 통합형 도구들도 존재하고, 산정 및 WBS 작성 등만을 지원하는 독립형 도구들도 있다.
- 지원도구: 형상관리, 측정 및 분석 등 독립형 도구들이 주로 사용되고 있다.
- 개발도구: 요구사항 개발도구, 설계-모델링 도구, 구현 도구 그리고 테스팅 도구 등이 독립형 도구로 존재하고, 이들 영역을 모두 지원하는 통합형 도구도 사용되고 있다.

독립형 도구들은 공개 소프트웨어로도 제공되고 있어서 소규모 조직들도 채택하여 사용해 왔지만 여러 프로세스를 지원하는 통합형 도구가 요구되는 경우가 많이 있다. 이러한 어려움을 지원할 수 있는 개발 환경 중 하나가 이클립스 플랫폼이다.

2.4 이클립스 플랫폼

이클립스 플랫폼은 이클립스 프로젝트에서 개발한 개발환경으로, 플러그인을 통해 기능을 확장할 수 있다. 또한 플러그인들간의 연계가 가능하여 보다 강력한 통합 환경을 제공한다.

본 논문에서는 이클립스 플랫폼의 이러한 장점을 활용하여, 특정 소규모 조직을 위한 이클립스 플러그인 모델을 정의하였다. 요구사항 개발, 테스팅 그리고 형상관리의 대표적인 플러그인들을 기반으로 이클립스 플러그인 모델을 구성하여, 효과적인 테스팅과 산

출물의 형상관리를 용이하게 하였다. 3 장에서는 정의된 이클립스 플러그인 모델에 대해 설명하고, 4 장에서는 변경관리 프로세스를 기반으로 이 모델의 효과를 검증하였다.

3. 프로세스 구축을 위한 이클립스 플러그인 모델의 활용

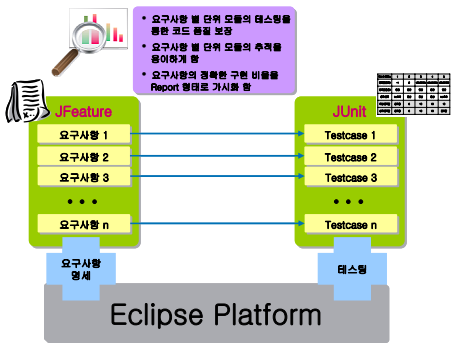
소규모 조직 중 이번 연구에 참여한 K 사의 내부프로젝트들은 규모가 크지 않고, 개발 기간이 짧기 때문에 코딩과 디버깅을 위한 도구만을 사용하는 것이 일반적인 경우였다. 품질은 주로 테스팅을 통해서 이루어져 왔는데, 문서도구를 통해 요구사항 정의서를 작성하고, 정의서 내의 요구사항 항목들을 기반으로 테스트 케이스를 만들었다. 요구사항이 변경되었을 때, 요구사항 정의서를 수정하는 작업이 지연되어 무결성이 유지되지 못 하는 경우가 빈번하고, 테스트 케이스를 빠뜨리는 경우, 테스팅 중에 요구사항의 문제점이 발견되는 경우 등 품질에 영향을 미치는 문제점들이 많았다.

또한 산출물들의 형상관리를 수작업으로 하기 때문에, 덮어쓰기 또는 불일치 버전의 통합 등 제품의 품질에 영향을 미치는 실수들이 빈번히 일어났다. 이러한 문제점을 해결하고, 품질과 생산성을 향상시키기 위해 요구사항 개발 도구인 JFeature 와 테스팅 도구인 JUnit, 그리고 형상관리 도구인 Subclipse 를 연계하여, 통합 환경을 구축했다. 각 도구의 기능은 다음과 같다.

- JFeature: 사용자 요구사항을 명세하고, JUnit 의 테스트 메소드와 매핑 되어 요구사항의 구현 정도를 Reporting 기능을 통해 제공한다[9].
- Eclipse 자체 Editor: Java 코딩 및 디버깅 기능을 제공한다.
- JUnit: 단위 테스팅 도구로, 테스트 케이스 생성 및 수행 기능을 제공한다. 또한 JFeature 와 통합되어 수행된다[10].
- Subclipse: Subversion 의 Eclipse 버전으로 각종 산출물의 버전관리 및 팀 개발 기능을 제공한다[11].

3.1 JFeature 와 JUnit 의 결합

요구사항 명세 도구인 JFeature 와 테스팅 도구인 JUnit 이 통합된 (그림 3)과 같은 이클립스 플러그인 모델을 활용하여, 연관된 문서간의 무결성을 유지하고, 보고서 출력 기능 등을 제공한다.



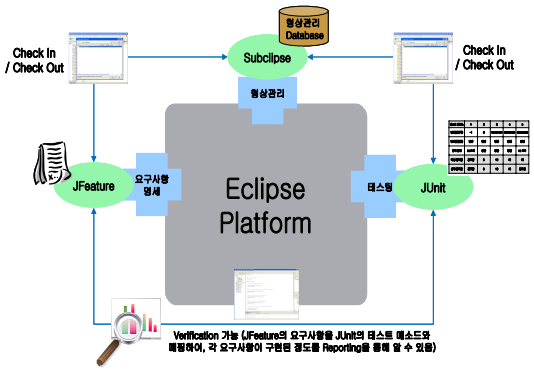
(그림 3) JFeature 와 JUnit 의 결합

요구사항 기능 항목과 테스트 케이스가 함께 매핑된 테이블을 생성하기 때문에 교차 검토가 가능하며, 요구사항의 구현 비율을 가시적으로 확인할 수 있고, 테스트 케이스가 생성되지 않은 기능 항목의 파악이 가능하다. 또한 테스팅 결과 보고서를 자동 생성함으로써 그에 수반되는 노력을 줄일 수 있다.

3.2 JFeature, JUnit 그리고 Subclipse 의 결합

개발 과정에서 만들어지는 산출물들의 무결성을 위한 형상 관리 작업을 위해, JFeature 와 JUnit 모델에 Subclipse 를 연계하였다. 소프트웨어 개발 단계에서 산출된 요구사항 문서, 소스 코드, 테스트 소스 등은 Subclipse 를 통해 형상관리 DB 에 저장되어 관리된다. 여러 사람이 동일한 데이터를 수정하고 관리하는 협업이 가능하며, 변경 전 버전과 변경 후 버전의 비교, 잘못된 변경사항의 원상태로의 복구 등을 제공하여, 소프트웨어 개발의 산출물 관리를 용이하게 한다.

(그림 4)는 이클립스 플러그인 모델의 연관도를 보여준다.



(그림 4) JFeature, JUnit 그리고 Subclipse 의 결합

이클립스 플러그인 모델의 각 단계별 흐름에 대한 설명은 다음과 같다.

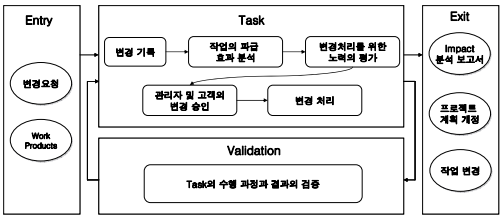
- JFeature 를 통해 사용자의 요구사항을 명세 한다.
- 명세된 각각의 요구사항을 Eclipse 자체 Editor 를 이용하여 단위 모듈로 구현한다.
- JUnit 을 이용하여, 단위 모듈별 테스트 메소드를 구현한다.
- JFeature 에서 명세한 사용자의 요구사항과 단위 모듈별 테스트 메소드를 매핑한다.
- JUnit 을 이용하여 단위 테스팅을 수행한다.
- JFeature 의 Reporting 기능을 통해 요구사항의 구현의 정도를 Coverage 로 보여준다.
- 공통적으로 각 단계마다 산출된 요구사항 문서, 소스 코드, 테스트 케이스 등은 Subclipse 를 통해 형상관리 DB 에 저장되어 관리된다.

4 변경관리 프로세스의 구현을 통한 플러그인 모델의 검증

K 사의 프로세스 구현을 위해 구축된 (그림 4)의 이클립스 플러그인 모델을 기반으로 변경관리 프로세스를 정의하였다.

ETVX 프로세스 모델을 기반으로 변경관리 프로세

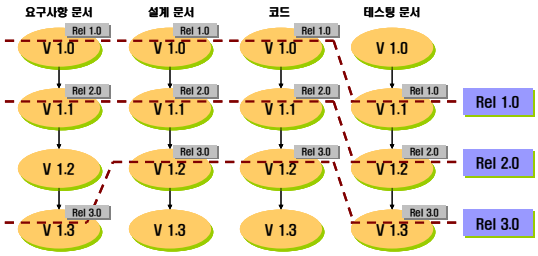
스를 도식화하면 (그림 5)와 같다.



(그림 5) 변경관리 프로세스

고객의 변경요청을 받은 개발팀은 변경을 기록하고, 이에 따른 파급 효과를 분석한 뒤, 변경 처리에 따른 노력을 평가한다. 그리고 관리자 및 고객간의 합의를 통해 변경 유무를 결정하여, 변경이 가능한 경우에는 변경을 실시한다. 이러한 모든 과정과 결과는 지속적으로 검증된다.

변경관리에서 중요한 활동은 (그림 6)과 같이 요구사항 문서, 설계 문서, 소스 코드, 테스팅 문서의 무결성을 유지하는 것이다.

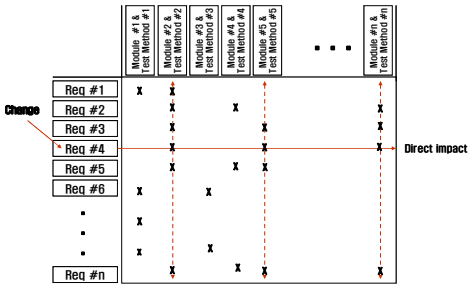


(그림 6) 개발 산출물의 무결성 유지[1]

이렇게 개발 산출물간에 무결성을 유지하기 위해서는 문서간 추적성이 가능해야 한다.

도구의 도움 없이는 문서 도구를 활용하여, 각 문서들의 테이블을 만들고, 변경 때마다 수작업으로 그 테이블의 내용들을 일치시켜야 한다.

본 논문에서 제안한 이클립스 플러그인 모델은 상용 도구에서만큼의 추적성은 제공하지 않지만, (그림 7)과 같이 요구사항과 관련된 단위 모듈 및 테스트 메소드를 매핑 시킬 수 있다.



(그림 7) 요구사항과 단위모듈 및 테스트 메소드의 매핑

이에 따라 특정 요구사항이 변경 시, 개발자는 관련된 단위 모듈과 테스트 메소드를 확인할 수 있고, 이들을 수정하여 보다 효과적으로 변경사항을 관리할 수 있다.

5 결론

소프트웨어는 사람에 따라 다양한 방식으로 개발될 수 있다. 소프트웨어 개발 조직들은 소프트웨어 개발을 일관성 있게 수행하고, 제품의 품질을 유지하기 위해 소프트웨어 프로세스를 정의하고 이에 따라 개발을 진행한다. 프로세스를 구현하기 위해서는 각 프로세스의 담당자를 할당하고, 수행 활동들을 정의하고 그에 따라 작업을 수행해야 하는데, 이때 작업 수행에 도움이 되는 필요한 도구들을 활용한다.

이러한 도구들 중에 통합형 도구들은 여러 프로세스를 연결하고 통합 관리하기 때문에 효과가 크지만, 주로 상업용 제품들이라 가격이 비싸고, 기능이 복잡하여 소규모 조직이 채택하기에는 어려움이 많았다.

본 논문에서는 이클립스 플랫폼 기반의 플러그인으로 여러 독립형 도구들을 통합하는 방식으로 여러 프로세스의 구현을 지원하는 통합 모델을 구축하고 그 효과를 살펴보았다.

소규모 조직들이 적절한 도구를 선택하여 통합하여 활용한다면, 작업을 위해 소요되는 시간과 노력을 줄일 수 있을 뿐만 아니라, 품질은 높고, 재 작업량은 줄어, 생산성을 높일 수 있을 것이다.

향후 연구로는 이클립스 플러그인 방식을 효과적으로 활용하기 위해서 플러그인 저장소를 구축하고, 조직이 원하는 방식대로 플러그인들을 통합할 수 있는 지침들에 대한 연구가 이루어져야 하겠다.

* 본 연구는 지식경제부 및 한국소프트웨어진흥원의 소프트웨어공학물 공용저장소 구축 및 운영 사업의 연구 과정 중 수행되었음

참고문헌

[1] 한혁수, “소프트웨어 공학의 소개”, 홍릉과학출판사, 2008
[2] 김영걸, “CASE 도구의 활용에 의한 소프트웨어 개발의 생산성 및 품질향상 방법론에 관한 연구”, 정보과학회논문, 1997
[3] V.J. Mercurio, B.F. Meyers, A.M. Nisbet, G. Radin, AD/Cycle strategy and architecture, IBM System Journal, 1990
[4] Zhihui Yang, Wayne Zage, Dolores Zage, “The Eclipse Platform for Tool Integration and Development”, IBM Systems Journal, 2004
[5] 김성훈, 이홍창, 박양수, 이명준, “이클립스 통합 개발 환경에서의 원격 작업 공간 지원”, 한국정보과학회, 2006
[6] ISO/IEC 12207: Software life cycle process
[7] Mary Beth Chrissis, Mike Konrad, Sand Shrum, “CMMI Second Edition: Guidelines for Process Integration and Product Improvement”, Addison-Wesley, 2006
[8] Watts S. Humphrey, Managing the Software Process, Addison-Wesley, 1989
[9] “http://www.technobuff.net/”, JFeature
[10] “http://junit.org/”, JUnit
[11] “http://subclipse.tigris.org/”, Subclipse