Product line 개념에 따른 소프트웨어 테스트 프로세스 재사용 방안

이윤정 최병주
이 화 여 자 대 학 교 컴 퓨 터 학 과
{yoonjung, bjchoi}@mm.ewha.ac.kr

A Scheme on Software Test Process Reuse for Product Line Practice

Yoonjung Lee Byoungju Choi
Dept. of Computer Science & Engineering, Ewha Womans University

요 약

본 논문에서는 product line 개념을 활용하여 체계적으로 각 어플리케이션에 적합한 소프트웨어 테스트 프로세스를 생성할 수 있도록 하는 1) "소프트웨어 테스트 프로세스의 재사용 방안"을 제안하고, 2) 이를 자동화한 "소프트웨어 테스트 프로세스 생성 도구"의 프로토타입을 제시한다.

"소프트웨어 테스트 프로세스의 재사용 방안"은 product line 개념에 따라 표준, 방법론과 도메인을 위한 개발 프로세스들의 공통점과 차이점을 분석하여 core asset들을 CBD개념의 컴포넌트들로 개발하고, 이 core asset들을 가지고 컴포넌트의 맞춤 패턴을 이용하여 손쉽게 각 어플리케이션에 적합한 테스트 프로세스를 생성할 수 있도록 한다.

"소프트웨어 테스트 프로세스 생성 도구"는 "소프트웨어 테스트 프로세스 재사용 방안"의 core asset 개발 단계에서 개발된 core asset들을 저장소에 재사용을 목적으로 저장하며, 프로덕트 개발 단계를 자동화하여 각 어플리케이션에 적합한 테스트 프로세스를 생성한다.

1. 서론

소프트웨어 재사용과 컴포넌트 기반 소프트웨어 개발은 빠르게 변화하는 소프트웨어 시장의 특성을 고려하여, 고품질의 소프트웨어를 보다 빠른 시간과 적은 비용으로 생성하고자 하는 요구에 의한 것이다. 최근 이러한 요구는 소프트웨어 제품 자체 뿐만이 아니라 소프트웨어 제품을 개발하기 위한 프로세스, 소프트웨어 아키텍처, 테스트 케이스 등에 이르는 소프트웨어 전 분야로 확장되고 있다. 이러한 추세는 특정 도메인에 속하는 어플리케이션 간의 재사용을 극대화 하고자 하는 개념인 "product line"[1,2]으로 발전하고 있다.

최근엔 소프트웨어 활용 분야가 점점 다양화되면서 어플리케이션의 특성에 적합한 테스트 프로세스에 대한 요구가 늘어나고 있다. 표준에서 제안하는 프로세스들은 기본적인 핵심 프로세스들은 유사하나 방법론이나 어플리케이션이 속하는 도메인에 따라 서로 다른 다양한 특징을 갖는다. 같은 도메인에 속하는 어플리케이션이라도 그 요구사항에 따라 조금씩 다른 테스트 프로세스를 필요로 하기도 한다. 이러한 소프트웨어 프로세스의 공통점과 차이점을 잘 분석하여 재사용할 수 있는 컴포넌트로 개발한다면, 손쉽게 다양한 어플리케이션 각각에 맞는 적절한 테스트 프로세스를 생성할 수 있다.

본 논문에서는 각 어플리케이션에 적합한 소프트웨어 테스트 프로세스를 생성 할 수 있도록 하는 "소프트웨어 테스트 프로세스 재사용 방안"을 제안하고, 이를 위하여 product line 개념을 활용한다. Product line은 도메인의 공통 요구 사항들(프로세스, 프레임워크, 패턴 등)을 도출하여 추상화 시킨 후, 도메인내의 여러 어플리케이션과의 차이점과 공통점을 분류하는 분석을 통하여 도메인을 위한 core asset을 개발하는 단계와, 이를 맞춤하여 프로덕트를 개발하는 단계로 이루어진다.

본 논문에서는 생성하고자 하는 프로덕트를 소프트웨어 프로덕트가 아닌 소프트웨어 테스트 프로세스로 정의한다. 즉, 본 논문의 "소프트웨어 테스트 프로세스 재사용 방안"은 product line개념에 따라 표준 프로세스, 방법론과 도메인에서의 공통점인 핵심 프로세스와 차이점인 특징들을 분석하여 core assets으로 구현하고, 이들을 특정 어플리케이션을 위한 테스트 프로세스로 맞춤하는 core asset 개발과 프로덕트 개발 단계를 갖는다.

본 논문에서는 "소프트웨어 테스트 프로세스 재사용 방안"을 자동화하는 "소프트웨어 테스트 프로세스 생성 도구"의 프로토타입을 제안한다. "소프트웨어 테스트 프로세스 생성 도구"는 core asset 개발 단계에서 개발된 core asset을 저장소에 저장하고, 프로덕트 개발 단계를 자동화 하여 사용자가 원하는 각 어플리케이션에 적합한 테스트 프로세스를 자동 생성한다.

본 논문은 product line의 개념에 컴포넌트 기술을 적용하여 소프트웨어 테스트 프로세스를 재사용할 수 있는 체계적인 방안과 자동화 도구를 제안함으로써, 쉽게 다양한 어플리케이션에 적합한 소프트웨어 테스트 프로세스를 생성할 수 있는 방법을 제시한다.

2장에서는 관련 연구로써 "Product line"을 기술하며, 3장에서는 본 논문에서 제안하는 Product line 개념을 활용한 "소프트웨어 테스트 프로세스 재사용 방안"을 기술하고, 4장에서는 "소프트웨어 테스트 프로세스 재사용 방안"을 자동화하는 "소프트웨어 테스트 프로세스 생성 도구"에 대해 기술한다. 마지막으로 5장에서는 결론 및 향후 연구 과제를 기술한다.

2. 관련연구

2.1 Product Line

최근엔 특정 도메인내의 유사 어플리케이션간의 재사용을 극대화 하고자 하는 노력으로 product line이 활성화되고 있다. 본 논문에서는 소프트웨어 테스트 프로세스를 재사용하는 방안으로써, product line 개념을 이용한다. Product line 개발 프로세스는 core asset 개발과 프로덕트 개발의 주요 단계로 구축된다

(1) Core asset 개발

Core asset 개발 단계는 도메인의 공통 요구 사항들을 도출하여

추상화 시킨 후 도메인내의 여러 어플리케이션과의 차이점과 공통점을 분류하는 분석을 통하여 도메인을 위한 core asset을 컴포넌트로 구현한다.

(2) 프로덕트 개발

프로덕트 개발 단계는 production plan에 따라 core asset들을 맞춤하여 특정 프로덕트를 개발하는 단계이다.

3. 소프트웨어 테스트 프로세스 재사용 체계

본 논문에서 제안하는 "소프트웨어 테스트 프로세스 재사용 방안"은 그림 1에서처럼 "*Core asset* 개발"과 "프로덕트 개발"의 2단계로 이루어지며, 전체 프로세스는 "Management"에 의해 통제된다. 그림 1의 화살표가 양방향인 것은 core asset이 프로덕트 개발 단계에 이용될 뿐만 아니라, 개발된 프로덕트도 새로운 core asset으로 core asset 개발 단계에 사용할 수 있음을 의미한다.

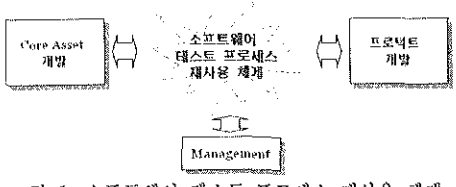

그림 1. 소프트웨어 테스트 프로세스 재사용 체계

본 논문에서 core asset의 관리와 프로덕트 개발 단계의 전체 과정은 "Management"에 의해 관리되며, 이를 자동화하는 도구의 프로토타입에 대해서는 4장에서 기술하겠다.

3.1 Core Asset 개발

본 논문에서의 "소프트웨어 테스트 프로세스 재사용 방안"의 Core asset 개발단계는 1) reuse architecture 개발, 2) core asset 개발, 3) production plan개발의 3단계로 구성한다.

*그림2*는 *core asset 개발* 단계를 입력물과 출력물을 중심으로 표현한 것이다. 표준, 방법론, 도메인의 특성, 테스팅 기법은 core asset 개발을 위한 입력으로 사용되며, 맞춤 패턴과 생성 기법은 production plan 작성을 위한 입력으로 사용된다.

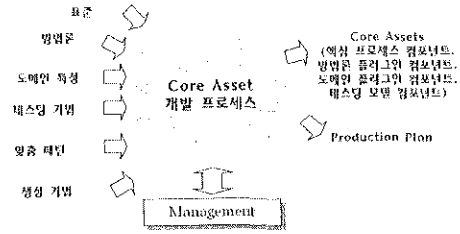

그림 2. Core Asset 개발 프로세스

3.1.1 Reuse architecture 개발

본 논문에서는 체계적인 소프트웨어 테스트 프로세스의 재사용을 위하여 5단계의 reuse architecture를 제안한다. 제1단계는 프로세스 표준에 대한 재사용, 제2단계는 다양한 방법론에 대한 재사용, 제3단계는 개발 도메인에 대한 공통 특성들에 대한 재사용이고, 제4단계는 방법론, 개발 도메인, 프로젝트 특성에 따른 테스팅 모델의 재사용이다. 마지막 단계는 특정 어플리케이션의 특성을 맞춤하기 위한 단계이다.

제1단계의 프로세스 표준에 대한 재사용은 ISO/IEC 12207, MIL-STD 498과 ANSI/IEEE SVVP등의 프로세스 관련 표준들 [3,4]을 기반으로, 반드시 요구되는 테스트 프로세스를 작업, 절차, 산출물과 같은 프로세스 요소들을 기준으로 핵심 프로세스로 추출하여 이를 재사용을 목적으로 core asset으로 개발한다. 제2단계의 방법론 재사용은 절차적, 객체지향적, 컴포넌트 기반 등의 다양한 방법론의 특징들을 테스트 프로세스 요소들을 기준으로 추출하여 core asset으로 개발한다. 제3단계 도메인 재사용은 향후 유망할 도메인인 전자상거래, 금융, 통신 등의 개발 도메인의 특징들을 지침과 기법의 기준으로 추출하여 core asset으로 개발한다. 제4단계의 테스팅 모델의 재사용은 방법론, 개발 도메인, 프로젝트 특성에 따라 테스팅 기법을 적용한 테스팅 모델을 추출하여 core asset으로 개발한다. 제5단계의 현재 생산하려고 하는 어플리케이션 맞춤에서는, 어플리케이션의 특성을 어플리케이션 요구사항으로부터 추출하여 프로덕트 개발 단계의 입력으로 이용한다.

Reuse architecture를 5단계로 세분화함으로써 core asset의 재사용성을 높이고 보다 정확한 프로세스를 생성할 수 있다. 즉, reuse architecture는 표준을 준수하며, 특정 방법론의 프로세스 특성을 반영하고 도메인에 따른 지침과 테스팅 기법을 반영하여, 각 어플리케이션에 적합한 테스트 프로세스를 생성할 수 있도록 하는 기반을 제공한다.

3.1.2 Core asset 개발

본 논문에서는 3.1.1절에서 제안한 reuse architecture에 따라 재사용의 대상이 되는 core asset을 정의하고, 이를 컴포넌트로 개발한다. core asset은 그림 2처럼 표준을 반영한 핵심 프로세스 컴포넌트, 방법론을 반영한 방법론 플러그인 컴포넌트와 도메인 특성을 반영한 도메인 플러그인 컴포넌트 및 테스팅 기법을 반영한 테스팅 모델로 구성된다.

본 논문에서는 소프트웨어 테스트 프로세스 재사용을 위한 core asset을 개발하는데 소프트웨어 제품 구현 기술로만 활용되어 온 CBD 기술을 적용한다. CBD 기술이 진행적인 소프트웨어 제품뿐만아니라 "소프트웨어 테스트 프로세스"에도 적용 가능하기 때문이다. 일반적인 컴포넌트들이 설계 산출물을 기반으로 한 실행 코드인 것에 비해, 본 논문에서 정의하는 소프트웨어 테스트 프로세스 재사용을 위한 core asset의 컴포넌트는 자연어로 기술된 소프트웨어 테스트 프로세스를 객체 지향적으로 모델링하여 UML [5]로 기술한 메타 모델이다. 소프트웨어 테스트 프로세스를 위한 core asset에 대한 정의는 정의 1과 같다.

정의 1: Core asset

Core asset은 표준에 따라 반드시 수행되어야 하는 수정 불가능한 '블랙 박스'영역과 프로덕트의 특성에 따라 수정 가능하도록 하는 '인터페이스'로 구성한다. 본 논문에서는 core asset을 UML로 기술한 메타 모델로 표현하는데, 그림 *3*과 같이 *core asset*의 정적모형(structural feature)은 패키지, 클래스도로, core asset의 동적모형(behavioral feature) 은 순서도로 표현한다.

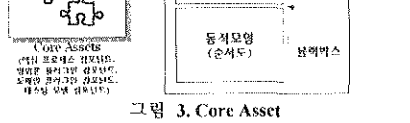

그림 3. Core Asset

그림2에서 보듯이 core asset은 각각 표준, 방법론과 도메인의 특성, 도메인, 방법론, 프로젝트에 따른 테스팅 모델로부터 특성을 추출한 뒤, core asset 개발 과정을 통해 각각의 컴포넌트로 개발된다.

3.1.3 Production plan 개발

프로덕트 개발 단계에서 core asset을 기반으로 어플리케이션에 적합한 테스트 프로세스를 생성하기 위해서는, 체계적인 production plan이 필요하다. 각 어플리케이션에 적합한 테스트 프로세스의 생성은 표준을 준수하는 핵심 프로세스 컴포넌트에 특정 방법론과 개발 도메인의 플러그인 컴포넌트를 컴포넌트 맞춤을 함으로써 이루어진다.

본 논문에서는 [6]에서 제안한 컴포넌트 맞춤 패턴과, 각 맞춤 패턴별 생성 기법에 따라, 각 어플리케이션에 적합한 테스트 프로세스를 생성하기 위한 production plan을 작성한다. 즉, core asset을 이루는 핵심 프로세스 컴포넌트와 방법론/도메인 플러그인 컴포넌트를 표1의 맞춤 패턴에 따라 연관/상속 관계로 연결하는 생성 기법을 기반으로 production plan을 수립한다. Production plan은 프로덕트 개발 단계에서 각 어플리케이션에 적합한 테스트 프로세스를 생성하는데 이용된다.

3.2 프로덕트 개발

본 논문에서의 "소프트웨어 테스트 프로세스 재사용 방안"의 프로덕트 개발 단계는 그림 4에서처럼 3.1절에서 개발된 core asset과 어플리케이션 요구사항을 기반으로 production plan에 따라 각 어플리케이션에 적합한 테스트 프로세스를 생성하는 작업을 수행한다.

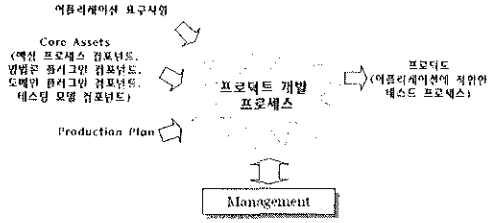

그림 4. 프로덕트 개발 프로세스

Core asset의 컴포넌트 중 특정 어플리케이션에 적합한 방법론, 해당 도메인에 대한 플러그인 컴포넌트 및 테스팅 모델을 production plan에 따라 핵심 프로세스 컴포넌트에 맞춤한다. 이렇게 방법론, 도메인 및 테스팅 모델 단계까지 맞춤하여 생성된 소프트웨어 테스트 프로세스에 해당 어플리케이션의 요구사항들을 입력 받아서 어플리케이션에 적합한 테스트 프로세스를 생성한다.

4. MANAGEMENT

3장에서 기술한 "소프트웨어 테스트 프로세스 재사용 방안"의 전체 프로세스는 "Management"에 의해 통제된다. "소프트웨어 테스트 프로세스 재사용 방안"의 Management는 core asset 개발 단계에서 생성되는 core asset을 저장소에 관리하며, 프로덕트 개발 단계의 전체 프로세스, 특정 어플리케이션에 적절한 방법론, 도메인 및 테스팅 모델에 해당하는 core asset의 컴포넌트를 선정하여 각 어플리케이션에 적합한 테스트 프로세스를 생성하는 프로세스를 관리한다.

본 논문에서는 "Management"를 자동화하는 "소프트웨어 테스트 프로세스 생성 도구"를 제안한다. "소프트웨어 테스트 프로세스 생성 도구"의 프로토타입을 그림 5에 나타내었는데, core asset을 관리하는 저장소와 프로덕트 개발단계를 관리하는 다섯 개의 모듈로 구성한다. "소프트웨어 테스트 프로세스 생성 도구"는 외부로부터 어플리케이션 요구사항인 방법론, 개발 도메인 및 특정 어플리케이션 정보를 입력 받고, core asset 저장소로부터 이들 요구사항에 적합한 소프트웨어 테스트 프로세스를 출력한다.

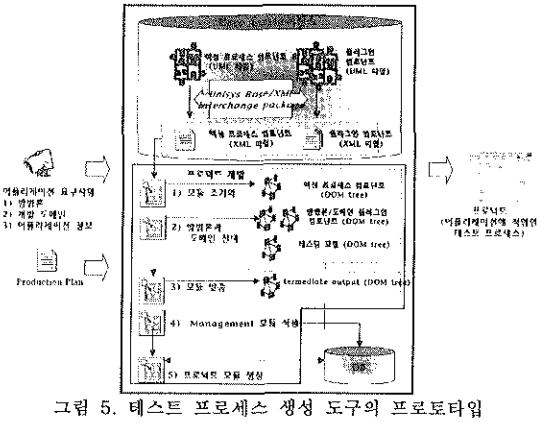

그림 5. 테스트 프로세스 생성 도구의 프로토타입

5. 결론 및 향후 연구 과제

본 논문에서는 product line 개념을 이용하여 "소프트웨어 테스트 프로세스의 재사용 방안"을 제안하였다. 본 논문의 "소프트웨어 테스트 프로세스 재사용 방안"은 core asset 개발과 프로덕트 개발의 2단계로 세분화되었다.

Core asset 개발 단계에서는 테스트 프로세스의 재사용을 위하여 표준 프로세스, 방법론, 도메인, 테스팅 모델과 어플리케이션으로 세분화되는 reuse architecture를 제안하였고, 이에 따라 체계적으로 core asset 을 개발할 수 있는 core asset 개발을 정의하였다. 또한, 본 논문에서는 재사용 단위로 개발된 core asset을 완벽한 프로덕트인 각 어플리케이션에 적합한 테스트 프로세스를 생성하기 위하여 CBD의 맞춤 패턴에 기반을 둔 production plan을 제안하였다.

프로덕트 개발 단계에서는 개발된 core asset과 어플리케이션 요구사항을 기반으로 production plan에 따라 어플리케이션에 적합한 테스트 프로세스를 생성하는 방법을 기술하였다.

"소프트웨어 테스트 프로세스 재사용 방안"의 전체 프로세스는 "Management"에 의해 통제된다. 즉, core asset 개발 단계에서 생성되는 core asset을 관리하며, 프로덕트 개발 단계의 전 과정을 관리한다. 본 논문에서는 "Management"를 자동화하는 "소프트웨어 테스트 프로세스 생성 도구"를 제안하였다. 핵심 프로세스 컴포넌트와 방법론, 도메인을 위한 플러그인 컴포넌트 및 테스팅 모델들의 core asset은 "소프트웨어 테스트 프로세스 생성 도구"의 core asset 저장소에 저장되어 재사용될 것이다. 즉, 어떠한 어플리케이션이라도 손쉽게 core assets 저장소 내에 저장된 컴포넌트를 재사용하여 특정 소프트웨어 테스트 프로세스를 생성할 수 있으며, 어플리케이션에 따라 방법론을 변경하고 싶은 경우에도 단순히 "소프트웨어 테스트 프로세스 생성 도구"의 메뉴를 통하여 원하는 방법론을 변경하여 선택함으로 손쉽게 변경할 수 있다.

향후 실제로 다양한 어플리케이션 환경에 맞는 소프트웨어 테스트 프로세스를 생성하기 위해서, 제안한 "소프트웨어 테스트 프로세스 생성 도구"를 구현하고, 각 방법론에 따른(절차적, 객체지향적, 컴포넌트 기반...등), 또한 각 개발 특성에 따른(전자 상거래, 통신, 제조, 금융, 시뮬레이션, 생명과학…등) 다양한 플러그인 컴포넌트 및 테스팅 모델을 개발하는 연구를 진행할 예정이다.

참고문헌

[1] CMU/SEI, Product Line Practice, http://www.sei.cmu.edu/plp/, January 2001.
[2] John Bergey, Matt Fisher, Brian Gallagher, Lawrence Jones, Linda Northrop, "Basic Concepts of Product Line Practice for the DoD", CMU/SEI-2000-TN-001, February 2000.
[3] ISO/IEC 12207 : Information Technology – Software Life Cycle Process.
[4] MIL-STD-498, Software Development and Documentation.
[5] Martin Fowler and Kendall Scott, UML Distilled : Applying the Standard Object Modeling Language, Addison-Wesley, Aug. 1997.
[6] Seo, Jooyoung, and Choi, Byoungju, "Tailoring Test Process By using the Component Based Development Paradigm and XML Technology," 7th Proceeding of Asia-Pacific Software Engineering Conference (APSEC) 2000 in Singapore, pp.356-363, Dec. 2000
[7] Clements, Paul & Northrop, Linda. A Framework for Software Product Line Practice, Version 2.0, SEI/CMU, July 1999
[8] Desmond Francis D'Souza and Alan Cameron Wills, Objects, Components, and Frameworks With Uml : The Catalysis Approach, Addison-Wesley Object Technology Series, Oct. 1998.
[9] Rational Objectory Process: Process Manual 4.1.
[10] Unisys / Rational XML Interchange Package v.4.0.1, ftp://ftp.rational.com/public/rosc/rose_extras/RoseXmi4.0.1.zip
[11] Wolfgang Pree, Design Patterns for Object-Oriented Software Development, Addison-Wesley, 1995.
[12] W3C, Extensible Markup Language (XML) 1.1, http://www.w3c.org/XML, 1998.