테스팅 도구 개발의 프러덕 라인화 방안

천은정* 최병주
이화여자대학교 컴퓨터학과
{ myring, bjchoi}@ mm.ewha.ac.kr

A Method for Testing Tool Development in a Product Line Practice

Eunjung Chun* Byoungju Choi
Dept. of Computer Science & Engineering, Ewha Womans University

요 약

다양한 자동화된 테스팅 도구들이 존재하지만 프러덕 라인(product line)에서 요구하는 다양한 레벨과 방법론을 수용하기에는 부족하다. 또한 기존 테스팅 도구는 특정 기법과 레벨만을 제공하기 때문에 사용자가 다른 기법을 사용하기 원할 경우 적용하고자 하는 기법이 적용된 다른 테스팅 도구를 사용해야 한다. 따라서 본 논문에서는 테스팅 도구 개발에 있어서 프러덕 라인화 방안을 제안한다. 프러덕 라인 개념에 따라 표준에 정의된 테스트 활동들의 공통점과 방법론과 기법에 따른 차이점을 도출하여 추상화 시킨 후 컴포넌트로 개발한다. 개발된 컴포넌트를 조립 시 컴포넌트로 구현된 차이점들을 리스트로 만들어 컴포넌트를 선택 가능하게 함으로써 생산된 제품이 변경 가능하도록 한다. 또한 개발하는데 드는 시간과 노력을 줄이기 위해 각 컴포넌트의 사용자 인터페이스를 실제로 구현하지 않고 명세하여 자동 생성되게 한다. 본 논문에서 제안하는 방안을 통해 사용자가 원하는 방법론과 기법이 적용된 테스팅 도구로 테스팅을 수행할 수 있게 될 뿐 아니라 개발된 컴포넌트를 재사용 할 수 있다.

1. 서론

프러덕 라인은 선택된 특정 마켓의 요구를 만족하면서 공통적으로 다룰 수 있는 특성을 공유하는 제품들의 집합이다[1]. 프러덕 라인 접근 방법을 통해 기업은 빠른 시간 내에 고객이 만족하는 고품질의 소프트웨어를 시장에 내 놓을 수 있으며 개발된 컴포넌트의 재사용을 극대화 할 수 있다. 프러덕 라인 프랙티스는 프러덕 라인을 구성하는 여러 제품들을 수정하고, 조립하고, 구체화하고, 생성하기 위해 소프트웨어 어셋들을 체계적으로 사용하는 것을 의미한다. 프러덕 라인 프레임웍을 구축하는 활동을 코어 어셋 개발 프로세스, 구축된 프레임웍을 통해 제품을 개발하는 활동을 제품 개발 프로세스라 한다[1].

기존 테스팅 도구들은 소프트웨어 생명주기 단계의 일부 혹은 전체를 지원하는 도구로 분류될 수 있다. 대부분의 도구들은 테스트 계획 단계에서부터 테스트 결과 분석까지의 과정을 자동화 한 도구이거나 특정 테스트 레벨과 특정 테스트 기법으로 테스트 데이터를 생성하는 도구이다. 이러한 도구들은 특정 레벨과 기법이 적용되어 구현되어 있으므로 사용자는 도구에서 제공하는 레벨과 기법만을 사용할 수 있다. 사용자가 다른 기법을 적용하여 테스팅을 수행하고 싶을 경우 해당 기법을 가진 도구를 설치해서 사용해야 한다. 이는 도구 설치 시 드는 시간과 노력 뿐 아니라 도구 구입 비용 및 도구 사용법을 습득하는 시간이 많이 들기 때문에 테스팅을 위해 준비하는 시간이 많이 소모되어 빠른 시간 내에 제품을 시장에 내놓기 힘들다. 또한 기존 도구 사이의 호환도 잘 되지 않아 사용자의 요구에 맞게 변경하거나 확장하기 힘들다.

따라서 본 논문에서는 테스팅 도구 개발의 프러덕 라인화 방안을 제안한다. 제안한 방안을 통해 다양한 테스팅 기법을 수용할 수 있는 테스팅 도구를 개발 할 수 있으며 동시에 이를 재사용할 수 있을 것이다.

2. 관련 연구

2.1 프러덕 라인

프러덕 라인은 선택된 특정 마켓의 요구를 만족하면서 공통적으로 다룰 수 있는 특성을 공유하는 제품들의 집합이다. 프러덕 라인 프랙티스의 필수적인 활동은 2단계로 코어 어셋 개발 단계와 개발된 코어 어셋을 이용해서 제품을 개발하는 제품 개발 단계가 있다. 코어 어셋 개발 단계에서는 도메인의 공통 요구 사항들을 도출하여 추상화 시킨 후 도메인 내의 여러 어플리케이션과의 공통점과 차이점을 분류하는 분석을 통하여 도메인을 위한 코어 어셋을 컴포넌트로 구현한다. 이 단계에서는 프러덕 라인의 범위, 코어 어셋들과 생산 계획을 정의한다. 제품 개발 단계는 생산 계획에 따라 코어 어셋들을 맞춤 하여 특정 제품을 개발한다. 즉, 코어 어셋 개발 프로세스의 결과물인 프러덕 라인 범위, 코어 어셋들, 생산 계획과 각 제품의 요구사항을 입력으로 반복적인 프로세스를 통해 제품을 생산한다.

3. 테스팅 도구 개발의 프러덕 라인화 방안

프러덕 라인 프로세스에 따른 테스팅 도구 개발은 다른 제품의 개발과 마찬가지로 프러덕 라인 프로세스를 따른다. 즉

코어 어셋을 개발 한 후 개발된 코어 어셋으로 제품을 개발 한다. 각 단계에서 수행해야 할 활동은 다음과 같다.

Step 1 코어 어셋 개발
  Step 1.1 아키텍쳐 개발
  Step 1.2 코어 어셋 개발
  Step 1.3 생산 계획 개발
Step 2 제품 개발
  Step 2.1 코어 어셋을 생산 계획에 따라 제품으로 개발

제안하는 방안은 다음과 같다. 즉 테스팅 활동의 공통성과 차이점을 분류, 분석하여 추상화를 시킨 후 이를 컴포넌트로 구현 한다. 컴포넌트로 구현된 다양한 테스트 활동들을 사용자의 요구에 맞는 테스팅 도구로 맞춤하기 위해 컴포넌트로 만들어진 기법들의 리스트를 작성하여 등록한다. 각 기법들은 독자적인 컴포넌트로 각 컴포넌트마다 사용자 인터페이스가 있다고 가정한다. 컴포넌트의 사용자 인터페이스는 따로 구현하지 않고 사용자 인터페이스 마크업 언어(User Interface Markup Language, UIML)로 명세 한다. 명세 된 사용자 인터페이스를 자동 생성하여 다른 테스트 활동 컴포넌트들과 상호작용 할 수 있도록 함으로써 제품을 개발한다.

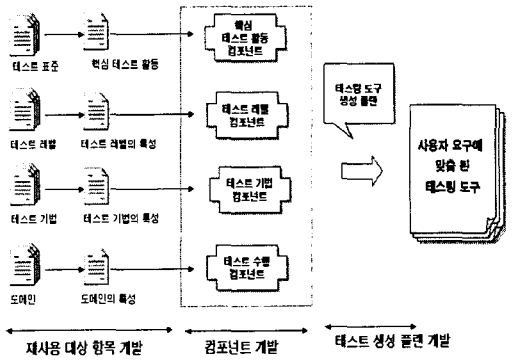

그림 1 테스팅 도구 개발의 프러덕 라인화 방안

3.1 테스트 활동의 공통점과 차이점 추출

3.1.1 공통점

테스트 프로세스는 테스트 계획, 테스트 디자인 생성, 테스트 케이스 생성, 테스트 수행, 테스트 결과 분석으로 이루어진다. 대부분의 테스팅 도구들은 테스트 프로세스의 전체 혹은 일부를 지원하기 위한 도구들이다. 소프트웨어가 복잡해지고 방대해짐에 따라 테스트 프로세스 전체를 지원하는 도구의 필요성이 커지고 있다. 또한 프러덕 라인에서 소프트웨어 개발 시에도 요구사항 모델의 확인에서 사용자가 제품을 인수한 후 수행되는 검증작업까지 모두 포함한다. 따라서 테스트 프로세스를 수행하면서 다양한 테스트 기법을 수용하는 테스팅 도구를 개발할 필요성이 증대되고 있다.

공통점을 추출 시 공통점에 전체 테스트 프로세스를 포함한다. 공통점을 추출하기 위해 테스트 프로세스가 명시된 표준을 분석하여 테스트 활동을 추출한다. IEEE standard 1012-1998 표준에 명시된 테스트 활동은 테스트 계획 생성, 테스트 디자인 생성, 테스트 케이스 생성, 테스트 절차 생성, 테스트 수행이다. 각 테스트 활동의 산출물은 IEEE standard 829-1983을 참고로 하여 정의한다.

3.1.2 차이점

테스팅에 있어서 각 방법론에 따른 테스트 레벨과 기법은 다양하게 제시되었다. 테스트 레벨은 소프트웨어 개발 주기에 따라 단위, 통합, 시스템, 인수 테스팅으로 나뉠 수 있으며 혹은 방법론에 따라 다른 레벨을 가질 수 있다. 예를 들어 컴포넌트 기반 개발 방법론에서는 컴포넌트 테스트, 자격 테스트, 맞춤 테스트, 조립 테스트, 보증 테스트의 레벨이 있을 수 있다. 테스트 기법의 경우 화이트 박스 테스팅 기법뿐 아니라 블랙박스 테스팅 기법이 있으며 각 기법에도 세부적으로 다양한 기법들이 있다. 테스팅 수행 시 소프트웨어의 특성과 개발 방법론에 맞는 적절한 기법을 사용한다. 테스트의 수행의 경우 각 도메인의 특성과 환경, 테스트 레벨에 따라 테스트 요구 환경을 구성하는 것이 달라질 수 있다. 또한 테스트 수행에 포함되는 테스트 수행 결과 분석은 분석 방법에 따라 다른 형태로 보여지게 된다.

이러한 차이점들을 분석하여 추출한다. 추출한 공통점과 차이점을 UML로 추상화 한다.

3.2 코어 어셋 개발

코어 어셋을 개발하는 단계는 아키텍처를 정의하고 정의된 아키텍쳐에 명세 된 각 컴포넌트의 명세에 따라 코어 어셋 을 개발한 후 생산 계획을 개발하는 3단계로 구성된다.

아키텍쳐 개발 단계에서는 테스트 활동의 관계를 분석해서 정의하여 논리적인 컴포넌트로 이루어진 아키텍쳐를 정의한다. 코어 어셋 개발 단계에서는 논리적인 컴포넌트를 실질적인 컴포넌트로 구현한다. 생산 계획 단계에서는 개발된 코어 어셋을 기반으로 제품을 생산하기 위한 생산 계획을 생성한다.

다음 그림은 아키텍처를 나타낸다.

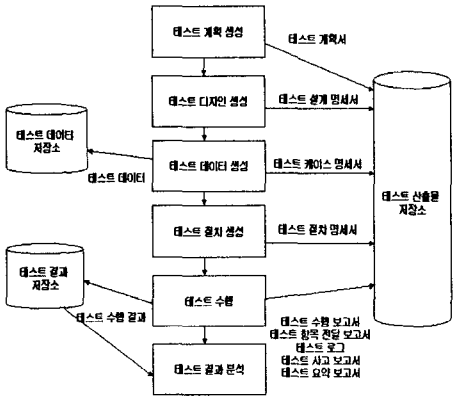

그림 2 아키텍처

테스트 활동으로 이루어진 아키텍처는 테스트 계획, 테스트 디자인 생성, 테스트 데이터 생성, 테스트 절차 생성, 테스트 수행이 각각 컴포넌트로 구성 된다. 각 단계의 진행 순서는 테스트 프로세스와 동일 하다. 각 단계에서 산출되는 산출물, 예를 들어 분서들은 테스트 산출물 저장소에 저장되며 테스트 데이터는 테스트 데이터에 저장된다. 테스트 결과는 테스트 데이터와 따로 저장되며 테스트 결과는 테스트 결과 분석에 사용된다.

아키텍처에서 테스트 레벨과 기법에 의해 변경되는 부분

은 테스트 디자인 생성과 테스트 케이스 생성이며 테스트 도메인에 따라 달라지는 부분은 테스트 수행이며 분석 방법에 따라 달라지는 부분은 테스트 결과 분석 부분이다. 예를 들어 컴포넌트 맞춤 테스트 기법을 사용할 경우 테스트 데이터 생성 부분에서 컴포넌트로 구현된 컴포넌트 맞춤 테스트 컴포넌트가 추가된다.

아키텍처에서 정의된 논리적인 컴포넌트들을 코어 어셋 개발 단계에서 실질적인 컴포넌트로 구현한다.

생산 계획 단계에서는 생성된 테스트 컴포넌트를 조립하는 방안을 제시한다. 다양한 컴포넌트를 수용할 수 있기 위해 컴포넌트 저장소를 두어 컴포넌트 리스트를 만들고 컴포넌트들의 사용자 인터페이스를 분리하여 명세 한다.

예를 들어 테스트 디자인 생성 컴포넌트의 경우 실질적으로는 테스트 디자인 생성 뿐 아니라 테스트 케이스 생성까지 포함한다. 따라서 테스트 디자인 생성 컴포넌트가 다양한 테스트 기법을 포함하는 여러 테스트 데이터 생성 컴포넌트와 조립될 수 있어야 한다. 각 테스트 데이터 생성 컴포넌트는 각각의 기법으로 테스트 모델을 생성하고 테스트 데이터를 생성하는 독립적인 기능을 하며 컴포넌트에 해당되는 사용자 인터페이스를 가지는 컴포넌트로 가정한다.

독립적인 사용자 인터페이스를 가진 컴포넌트들을 구현 시 각 컴포넌트마다 사용자 인터페이스를 구현하는 것은 시간이 많이 들고 노력이 많이 필요하다. 따라서 컴포넌트의 사용자 인터페이스를 명세만 하고 시스템에서 자동으로 생성할 수 있도록 컴포넌트와 사용자 인터페이스를 분리한다. 사용자 인터페이스를 실제로 구현하지 않고 UIML로 명세한다. 명세 된 내용을 기반으로 자동으로 사용자 인터페이스가 생성 되도록 한다. 생성된 사용자 인터페이스와 사용자 인터페이스에서 값을 받아오는 컴포넌트간의 상호작용이 가능하도록 한다.

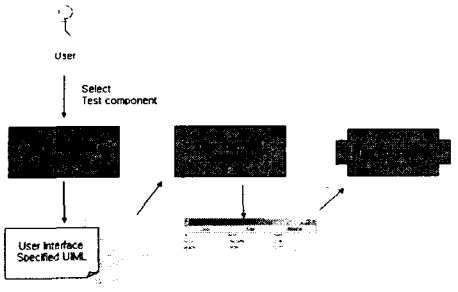

그림 3 제품 생산 계획

3.3 제품 개발

코어 어셋을 생산 계획에 따라 맞춤 하여 제품을 개발한다. 코어 어셋을 맞춤 하는 것은 소스코드 레벨에서 이루어지지 않고 컴포넌트로 구현된 다양한 기법 컴포넌트를 등록하고 해당 컴포넌트들의 사용자 인터페이스 명세를 등록하는 것으로 이루어진다. 즉 각 컴포넌트와 컴포넌트의 사용자 인터페이스가 명세 된 내용을 리스트로 작성한 후 작성된 리스트 중 원하는 컴포넌트를 선택한다. 선택된 컴포넌트에 해당하는 사용자 인터페이스 명세를 파싱하여 사용자 인터페이스의 인스턴스를 생성한다. 생성된 사용자 인터페이스를 통해 조립된 컴포넌트와 상호작용이 가능하다. 따라서 제품 개발 단계에서 생성 되는 것은 사용자가 선택한 기법이 반영된 컴포넌트 기반 테스팅 도구이다.

4. 결론 및 향후 연구 과제

일반적인 소프트웨어 개발 단계에서나 혹은 프러덕 라인에서 수행되는 개발 단계에서 수행되는 테스팅은 각 단계마다 다르다. 각 단계의 특성과 도메인의 특성상 여러 가지 테스팅 방법론이나 테스팅 레벨, 기법이 사용된다.

기존의 테스팅 도구는 이러한 다양한 단계의 테스팅을 수행하기에는 제공되는 기법이나 레벨이 한정적이다. 특정 레벨과 특정 기법을 가진 테스팅 도구만 존재하거나 전체 테스트 프로세스를 지원하는 테스팅 도구라 해도 도구에서 지원하는 방법론만을 사용할 수 있을 뿐 사용자의 요구를 반영할 수 없다. 다른 기법을 사용하기 위해서는 다른 도구를 구입하여 설치해야 하기 때문에 불편하고 비용적 부담이 든다. 테스팅에 시간이 많이 소요되어 전체적인 개발 시간도 길어진다. 또한 개발된 테스팅 도구는 복잡하고 방대하며 컴포넌트로 구현되지 않아 이를 재사용하기도 힘들다.

따라서 본 논문에서는 테스팅 도구 개발의 프러덕 라인화 방안을 제시하였다. 우선 표준에 명시된 테스트 활동을 추출하여 컴포넌트로 구현한다. 또한 방법론과 레벨에 따라 달라지는 테스팅 기법도 추출하여 컴포넌트로 구현한다. 구현된 컴포넌트를 생산 계획에 따라 조립하여 사용자 요구에 맞는 변경 가능한 테스팅 도구를 구현한다. 조립 시 컴포넌트의 사용자 인터페이스와 컴포넌트를 분리하여 명세한 후 사용자 인터페이스가 명세 된 내용에 따라 자동 생성될 수 있도록 UIML을 사용한다. 이 모든 과정은 프러덕 라인 개발 프로세스에 따라 진행된다.

본 논문에서 제시한 방안을 통하여 테스팅 도구 개발의 프러덕 라인화를 함으로써 개발자는 고품질의 테스팅 도구를 빠른 시간 안에 시장에 내놓을 수 있으며 컴포넌트들의 사용자 인터페이스를 개발하는 시간과 노력을 절약할 수 있으며 또한 사용자 요구 사항에 맞는 테스팅 도구를 소소코드 차원이 아닌 실행 시에 맞춤 할 수 있다. 컴포넌트로 구현되기 때문에 개발된 컴포넌트들은 재사용 될 수 있다. 향후에는 제안한 방안을 자동화하는 시스템을 구현할 예정이다.

참고 문헌

[1] Clements, Paul & Northrop, Linda. A Framework for Software Product line Practice, Version 3.0, SEI/CMU, March 2002

[2] CMU/SEI, Product Line Practice, http://www.sei.cmu.edu/plp, May 2002

[3] John D.McGregor, Testing a Software Product Line, December 2001

[4] Jooyoung Seo, Yoonjung Lee, Byoungju Choi, Sangduck Lee, Woohyun Jang, " A Scheme on Software Process Component Reuse using Product Line Practice" , Proceeding of ACIS 2nd International Conference on SNPD 2001, pp809-816, Nagoya, Japan, August, 2001

[5] IEEE Std 1012-1998 : IEEE Standard for Software Verification and Validation

[6] IEEE Std 829-1998 : IEEE Standard for Software Test Documentation

[7] UIML Version 2.0, http://www.uiml.org/specs, January 2000