테스트 프로세스 tailoring을 위한 방안

윤회진° , 최병주
이화여자대학교 컴퓨터학과

Tailoring Technique for the Test Process

Hoijin Yoon, Byoungju Choi
Computer Science & Engineering, Ewha Womans University ,

요 약

표준에 정의된 내용을 가지고, 각 프로젝트와 도메인에서 사용할 수 있는 수준의 프로세스를 구축하는일은 쉽지 않다. 표준에 근거하여, 어떤 부분을 어떻게 tailoring하여야 하는지에 대한 지침이 없기 때문이다. 따라서 표준은 표준대로 존재할 뿐, 실제 프로세스 정의에는 제대로 사용되지 못하고 있다. 본 논문에서는 표준에 대한 tailoring이 갖는 문제를 해결하기 위해, 콤포넌트가 갖는 메카니즘, 즉 customization과 composition을 이용한다. 표준에서 정의하는 프로세스를 추출하고, 그를 좀더 다양한 모습으로 변형시키기 위해 콤포넌트들로 정의한다. 나아가 이들을 실제 특정 도메인에 맞게 tailoring하기 위한 customization과 composition 방안에 대해 제안한다. 본 논문에서 제안한 프로세스 콤포넌트를 이용하여 표준의 테스트 프로세스를 도메인의 특성에 적합한 테스트 프로세스로 tailoring할 수 있다. 따라서 사용자는 프로세스 콤포넌트의 인터페이스를 제외한 나머지 부분에 대해서는 고려할 필요없이, 주어진 인터페이스와 플러그인들을 이용하여 customization과 composition만을 수행함으로써, 체계적으로 tailoring된 테스트 프로세스를 구축할 수 있다.

1. 개 요

ISO/IEC 14598[1]. ISO/IEC 9126[2]. ISO/IEC 15504[3] 등은 소프트웨어 제품과 프로세스에 관한 품질 향상을 위해 제시된 국제 표준이다. 이들 표준들은 특정 기술이나 방법론을 반영하지 않은 일반적인 프로세스를 제공하며, 그 수준 역시 개괄적이어서 실제 소프트웨어 개발에 적용하기에는 부족함이 있다. 따라서 표준을 각 프로젝트에서 이용하기 위해서는 개발 도메인에 맞도록 tailoring이 요구된다.

그러나 실제로 표준에 정의된 내용을 가지고, 각 프로젝트와 도메인에서 사용할 수 있는 수준의 프로세스로 tailoring하는일은 쉽지 않다. 표준에 근거하여, 어떤 부분을 이떻게 tailoring하여야 하는지에 대한 지침이 구체적이지 않기 때문이다. 따라서 표준은 표준대로 존재할 뿐, 실제 프로세스 정의에는 제대로 사용되기 어렵다.

본 논문에서는 표준에 대한 tailoring이 갖는 문제를 해결하기 위해, 콤포넌트가 갖는 메카니즘, 즉 customization과 composition을 이용한다. 표준에서 정의하는 프로세스를 추출하고, 그를 좀더 다양한 모습으로 변형시키기 위해 콤포넌트들로 정의한다. 나아가 이들을 실제 특정 도메인에 맞게 tailoring하기 위한 customization과 composition 방안에 대해 제안한다.

2장에서는 본 논문에서 tailoring방안의 기본으로 삼는 콤포넌트 기반 개발에 대해 알아보고, 3장에서는 표준으로부터 추출한 테스트 프로세스에서 추출한 콤포넌트를 정의한다 4장에서는 3장에서 정의한 프로세스 콤포넌트들을 이용하여 특정 도메인에 맞게 tailoring하여 새로운 프로세스를 구성하는 방법을 제안하고, 마지막 5장에서는 결론 및 향후 연구 과제를 기술한다.

2. 콤포넌트 기반 개발

프로세스 표준들을 실제 개발 도메인에 맞도록 tailoring하기 위해, 본 논문은 콤포넌트 개념을 이용한다. 이미 작성된 콤포넌트들을 새로운 목적으로 customize하고 composite하여, 새로운 기능을 하는 소프트웨어를 구축하는 과정이, 표준을 tailoring하여 특정 도메인에 맞는 프로세스를 구축하는 본 논문의 목적과 일치한다. 따라서 본 장에서는 콤포넌트 기반 개발에 대해 기술한다.

콤포넌트 기반 개발, 즉 CBD(Component-Based Development)의 기본 단위가 되는 콤포넌트에 대한 정의는 각양각색이다. 이러한 다양한 정의들은 다음을 공통적으로 언급하고 있다. 콤포넌트는 응집력을 갖는 소프트웨어 구현으로서, 독립적으로 개발되고, 인터페이스에 대한 명시적이고 잘 정의된 스펙을 갖는다. 콤포넌트 그 자체를 수정하지 않고 다른 콤포넌트들과 composition될 수 있고, 그의 특성을 customization할 수 있다. 소스코드를 통하지 않고, 인터페이스를 통하여 콤포넌트를 다양하게 다룰 수 있다. 따라서 콤포넌트는 표1과 같은 요소들로 구성된 하나의 패키지라고 볼 수 있다[4].

표 1 콤포넌트의 구성요소

콤포넌트 패키지 구성요소	내용
Provided interfaces	콤포넌트가 다른 콤포넌트나 플러그인에 제공하는 서비스
Required interfaces	다른 콤포넌트나 플러그인으로부터 요구되는 서비스
External specification	콤포넌트의 주변에 어떠한 행위를 연결할 수 있는지, 그리고 콤포넌트가 놓여진 환경이나, 적용된 콤포넌트 기술
Executable code	콤포넌트의 기능을 수행하는 코드
Design	콤포넌트가 개발된 때의 설계 산출물

CBD는 크게 3개의 view를 갖는다[5]. 이미 만들어진 콤포넌트를 이용하여 새로운 소프트웨어를 구성하는 integrator관점, 재사용될 수 있는 콤포넌트를 만드는 vendor 관점, 그리고 vendor가 만들어놓은 콤포넌트를 integrator가 효과적으로 이용할 수 있도록 도와주는 기능을 하는 broker 관점이 그것이다. 이 가운데 주로 integrator관점에서 CBD를 보기 쉬우나, 완벽한 CBD가 되려면 이 세 가지가 모두 고려되어져야 한다.

본 논문에서는 표준에 따른 테스트 프로세스[6]를 대상으로 vendor관점에서 프로세스 콤포넌트를 추출하고, integrator관점에서 그들을 이용하여 각 도메인에 맞는 테스트 프로세스를 구축할 수 있도록 한다. 이렇게 CBD가 갖는 잇점들을 프로세스에도 도입함으로써, 프로세스를 다양한 도메인에 체계적으로 적용할 수 있도록 한다.

3. 테스트 프로세스 콤포넌트 개발

테스트 프로세스 콤포넌트를 개발하기 위해, 우선 테스트 프로세스와 관련된 내용을 정의하고 있는 다양한 표준들[1,2,3]을 참조하여, 테스트 프로세스를 정의하였다[6]. 이 테스트 프로세스는 MaRMI-II 객체지향 테스트 프로세스[6]로서, 자연어로 나열된 문서의 형태를 갖는다. 이를 콤포넌트로 기능적 분할을 하기 위해서 테스트 프로세스를 대상으로 UML과 Objectory를 기반의 객체지향 모델링을 수행한다.

객체지향 모델링 결과, 테스트 프로세스의 활동을 기준으로 작업은 클래스로, 절차는 오퍼레이션으로 변환되며, 그 외에도 해당 활동과 관련이 있는 입력물이나 산출물은 클래스로 존재한다. 각 절차를 수행하기 위한 기법들은 오퍼레이션 내에 위치하고, 특정 도메인을 위한 기법들은 플러그인으로 존재하여 customization을 통해 프로세스 콤포넌트에 반영된다. 테스트 프로세스를 이루는 산출물과 활동들은 클래스로 표현되어 클래스다이어그램을 구성하고, 클래스들과 오퍼레이션사이의 수행 순서는 순서도로 표현된다. 클래스다이어그램과 순서도에서 낮은 커플링과 높은 응집력을 갖는 영역을 묶어 하나의 콤포넌트로 정의한다 이때 정의되는 콤포넌트는 일반적인 '소프트웨어 콤포넌트'를 specialize한 '프로세스 콤포넌트'라고 할 수 있다 프로세스 콤포넌트는 표1의 패지지 구성요소를 표 2와 같은 내용으로 specialize하여 정의한다.

표 2. 프로세스 콤포넌트의 구성요소

콤포넌트 구성요소	프로세스 콤포넌트 구성요소
Provide interfaces	다른 프로세스 콤포넌트에 서비스를 제공하는 작업
Required interfaces	프로세스의 작업들 가운데 외부의 입력을 요구하는 작업
External specification	프로세스 콤포넌트가 어떠한 행위를 연결할 수 있는지, 그리고 콤포넌트가 놓여진 환경이나, 적용된 콤포넌트 기술 등
Executable code	콤포넌트를 이루는 클래스들의 오퍼레이션들을 순서도에 따라 기술한 테스트 프로세스 (자연어 또는 기계어로 기술되어짐)
Design	프로세스 콤포넌트 내부의 활동들을 클래스로 표현한 클래스 다이어그램 프로세스 콤포넌트의 실행 코드는 콤포넌트를 이루는 활동과 작업으로 이루어진 순서도

이때 프로세스 콤포넌트의 소스코드는 순서도에 따라 콤포넌트를 이루는 클래스들의 오퍼레이션들이 수행될 수 있도록 구성되어진 프로그램이고, 그에 대한 실행코드는 소스코드 프로그램을 컴파일하여 사용자가 직접 수행할 수 있도록 자연어로 기술된 테스트 프로세스일 수도 있고, 아니면 테스트 프로세스의 자동화를 위해 실제 기계에서 돌아가는 기계어로 기술될 수도 있다.

MaRMI-II 테스트 프로세스[6]를 프로세스 콤포넌트로 분할한 결과, 다음의 7개 프로세스 콤포넌트를 정의할 수 있었다

- TestPlan
- UnitTestDesign
- IntegrationTestDesign
- SystemTestDesign
- TestExecution
- TestReview
- TestModeling

7개의 프로세스 콤포넌트 가운데 'IntegrationTestDesign' 프로세스 콤포넌트는 그림1 과 같다.

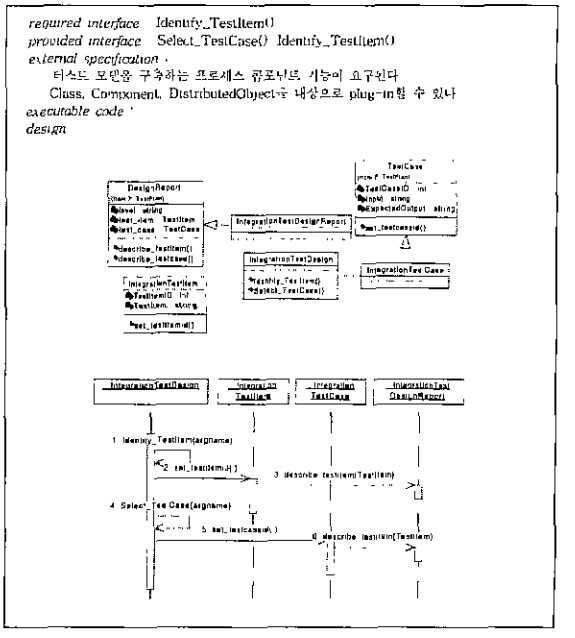

그림 1 'IntegrationTestDesign' 프로세스 콤포넌트

프로세스 콤포넌트는 특성에 따라 그림 2와 같이 세 가지 영역을 갖는다. 표준에 따라 반드시 수행되어야 하는 부분으로 수정 불가능한 '블랙박스'영역과, 특정 도메인을 위한 기법수준의 내용이나 도메인에서 요구되는 특정 작업들을 정의한 '플러그인'영역, 그리고 다른 프로세스 콤포넌트 또는 플러그인과 서비스를 교환해야 하는 부분인 '인터페이스'영역이다.

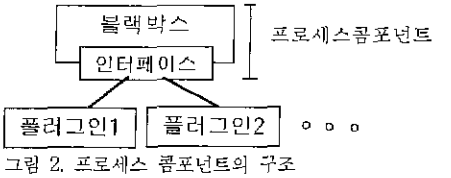

그림 2. 프로세스 콤포넌트의 구조

위에서 정의한 7개의 프로세스 콤포넌트는 표준에 따른 테스트 프로세스에서 추출한 프로세스 콤포넌트로서, 이들은 다양한 도메인의

특성을 표현한 플러그인을 갖을 수 있다. 플러그인은 프로세스 콤포넌트의 절차들에서 도메인의 특성에 따라 수행할 수 있는 구체적인 기법 수준의 내용을 담고 있다. 플러그인을 콤포넌트에 끼워서 도메인의 특성을 반영하도록 customize한다. 본 논문에서 정의한 프로세스 콤포넌트들도 표 3과 같은 플러그인들을 갖을 수 있다

표 3. 프로세스 콤포넌트의 plug-ins

프로세스 콤포넌트	플러그인
UnitTestDesign	ClassTestDesign
	ComponentTestDesign
	DistributedObjectTestDesign
IntegrationTestDesign	InterClassTestDesign
	CompositionTestDesign
	InterDistributedObjectTestDesign
TestModelign	ClassTestModeling
	ComponentTestModeling
	DistributedObjectTestModeling

4. 테스트 프로세스 콤포넌트 적용

3장에서는 CBD의 vendor관점에서 프로세스 콤포넌트를 개발하였다 프로세스 콤포넌트를 특정 환경에서 이용하기 위해서는 이미 개발된 콤포넌트들을 CBD의 integrator입장으로 도메인에 맞추는 방법이 요구된다. 콤포넌트를 다루는 방법에는 customization과 composition이 있다.

그림 1의 'IntegrationTestDesign'프로세스 콤포넌트의 경우, 그림 3에서 처럼 객체지향 클래스를 통합 테스트하는 특정 기법을 갖는 'InterClassTestDesign' 플러그인을 프로세스 콤포넌트의 인터페이스에 연관관계로 연결하거나(그림 3의 B), 인터페이스로 공개된 오퍼레이션을 수정함으로써(그림 3의 A) customization을 수행할 수 있다.

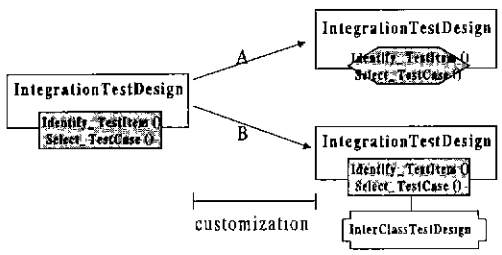

그림 3. 프로세스 콤포넌트 customization

테스트 프로세스를 구축하기 위해서는 하나 이상의 프로세스 콤포넌트들을 필요로 한다. 이들을 콤포넌트의 composition 메카니즘을 이용하여 서로 연결하여 보다 큰 기능을 수행하도록 해야 한다. 프로세스 콤포넌트도 일반적인 콤포넌트와 마찬가지로, 'component-port-connector' 모델[4]로 콤포넌트 architecture를 구성한다 port는 한 콤포넌트의 required interface와 또 다른 콤포넌트의 provided interface가 되고, port들 사이의 메시지 흐름을 자연스럽게 연결하는 역할을 하는 것이 connector이다[4]

예를 들어, 'IntegrationTestDesign' 프로세스 콤포넌트는 'TestModelng' 프로세스 콤포넌트와 composite되어져야 한다. 그림 4에서 처럼, 'IntegrationTestDesign' 프로세스 콤포넌트의 port는 required interface인 Identify_TestItem()이고, 'TestModeling' 이에 대한 프로세스 콤포넌트의 port는 그의 provided interface인 Draw_TestModel()이 된다. 이 둘을 연결하는 connector는 'TestModelng' 프로세스 콤포넌트에 의해 그려진 Test Model을 'IntegrationTestDesign' 프로세스 콤포넌트에 전달하는 역할을 한다.

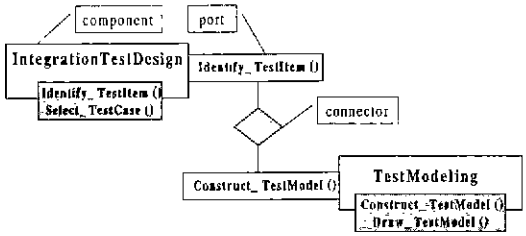

그림 4 프로세스 콤포넌트 composition

프로세스 콤포넌트는 표준에서 반드시 수행하도록 정의된 내용을 블랙박스로하고, 변형이 가능한 부분을 인터페이스로 한다. 따라서 표준의 내용을 특정 도메인이나, 개발 소프트웨어의 특성에 따라 tailoring하기 위하여, 프로세스 콤포넌트의 인터페이스만을 고려하는 콤포넌트 customization과 composition기술을 적용한다 이렇게 함으로써 표준에서 정의한 테스트 프로세스를 사용자사 쉽게 또 체계적으로 tailoring할 수 있다.

5. 결론 및 향후 연구 과제

표준의 내용들은 매우 추상적이어서 각 개발 환경에 따라 tailoring하여 사용하도록 되어 있다. 그러나 우선 표준에서 정의해 놓은 테스트를 위한 다양한 부분들을 하나의 테스트 프로세스로 기술하는데 어려움이 있고, 나아가 테스트 프로세스를 각 특징적인 도메인에 적합하도록 tailoring하는데 어려움이 있다. 첫 번째 문제는 MaRMI-II에서 정의한 테스트 프로세스로 해결하였고, 두 번째 문제는 본 논문에서 제안한 프로세스 콤포넌트로 해결할 수 있다.

본 논문에서 제안한 프로세스 콤포넌트를 이용하여 표준의 테스트 프로세스를 도메인의 특성에 적합한 테스트 프로세스로 tailoring할 수 있다. 따라서 사용자는 프로세스 콤포넌트의 인터페이스를 제외한 나머지 부분에 대해서는 고려할 필요없이, 주어진 인터페이스와 플러그인들을 이용하여 customization과 composition만을 수행함으로써, 체계적으로 tailoring된 테스트 프로세스를 구축할 수 있다.

본 논문의 프로세스 콤포넌트는 표준에 기반한 테스트 프로세스[6]를 대상으로 정의되었으나, 나아가 전체 개발을 위한 프로세스도 표준으로부터 추출하여, 개발 프로세스 콤포넌트를 개발한다면, 테스트를 포함한 전체 개발 프로세스가 다양한 도메인의 특성에 맞게 tailoring될 수 있을 것이다 이를 실제로 다양한 개발 환경에서 이용하기 위해서는 각 프로세스 콤포넌트마다 다양한 플러그인들 - 분산 환경을 위한, 콤포넌트로 구성된 소프트웨어를 위한, 실시간 시스템을 위한, 금융도메인을 위한, 등등 -을 개발하는 일이 요구된다.

참고문헌

[1] ISO/IEC 14598 : Information Technology - Software Product Evaluation

[2] ISO/IEC 9126 : Information Technology - Software Quality Characteristics and Metrics

[3] ISO/IEC 12207 : Information Technology - Software Life Cycle Process.

[4] Desmond Francis D'Souza and Alan Cameron Wills, *Objects, Components, and Frameworks With Uml The Catalysis Approach*. Addison-Wesley Object Technology Series, Oct, 1998.

[5] Mikio Aoyama, "New Age of Software Development : How Component-Based Software Engineering Changes the Way of Software Development?," proceedings of ICSE workshop on Component-Based Software Engineering, Apr, 1998.

[6] 최병주, "객체지향 소프트웨어 개발을 위한 테스트 방안에 관한 연구" 최종보고서, SERI, 1998