컴포넌트 기반 개발 개념을 활용한 테스트 프로세스 tailoring

(Tailoring Test Process by using Component-Based Development Paradigm)

윤 회 진 † 최 병 주 ††
(Hoijin Yoon)(Byoungju Choi)

요 약 다양한 표준에 정의된 내용을 가지고, 각 도메인에서 사용할 수 있는 프로세스를 구축하는 일은 어렵다. 왜냐하면 표준의 어떤 부분을 어떻게 tailoring하여야 하는지에 대한 지침이 부족하기 때문이다. 사실 표준은 표준대로 존재할 뿐, 실제 프로세스 정의에는 제대로 사용되지 못하고 있다. 따라서 본 논문에서는 컴포넌트 기반 개발 개념을 이용하여 다양한 도메인에 맞게 프로세스를 tailoring하는 방법을 제안한다. 우선 테스트 프로세스를 언급하는 다양한 표준들로부터 테스트 프로세스 컴포넌트를 정의하고, 이들을 특정 도메인에 맞게 tailoring하기 위한 맞춤(customization), 조립(composition) 방안을 제안한다. 본 논문에서 제안한 방법은 테스트 프로세스 컴포넌트의 인터페이스만을 고려하고 주어진 플러그인들을 이용하여, 맞춤·조립만을 수행함으로써, 체계적으로 tailoring된 테스트 프로세스를 구축할 수 있도록 한다.

Abstract It is difficult to build processes, which can be used directly in each domain, with varied standards. This is because not many written guides on how one can tailor the standards to a specific domain does not exist. So standards are hardly used for defining the process in reality. We therefore propose a tailoring method using the component-based development paradigm. To tailor the test process to a specific domain, we first define the test process component from the varied standards, then propose the customization and composition method for the specific domain. In our method, the user systematically builds a test process through the test process component customization and composition by only considering the interface of the component and using the provided plug-ins.

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 tailoring하는 일이 요구된다.

그러나 실제로 표준에 정의된 내용을 가지고, 각 프로젝트와 도메인에서 사용할 수 있는 수준의 프로세스로 tailoring하는 일은 쉽지 않다. 표준에 근거하여, 어떤 부분을 어떻게 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, 컴포넌트 기반 개발이 갖는 개념을 이용한다. 이를 위해 우선 표준에서 정의하는 내용들을 토대로 기준이 되는 테스트 프로세스를 추출하고, 그를 분할하여 테스트 프로세스 컴포넌트를 개발한다. 다양한 도메인의 요구사항을 수용할 수 있도록 각 테스트 프로세스 컴포넌트에 대한 플러그인들도 개발한다. 테스트 프로세스 컴포

· 본 연구는 한국과학재단 목적기초연구(과제번호:2000-0-303-02-3)지원으로 수행되었음.
† 학생회원 : 이화여자대학교 컴퓨터학과
hoijin@cs.ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
bjchoi@mm.ewha.ac.kr
논문접수 : 2000년 2월 10일
심사완료 : 2000년 11월 2일

넌트와 그들의 플러그인들을 이용하여 컴포넌트 기반 개발의 맞춤(customization)과 조립(composition)을 수행하는 tailoring으로 테스트 프로세스를 구축하는 방안을 제안한다.

2장에서는 본 논문에서 tailoring방안의 기본이 되는 컴포넌트 기반 개발에 대해 알아보고, 3장에서는 테스트 프로세스 tailoring을 위해 사용자에게 제공되어져야 하는 테스트 프로세스 컴포넌트와 그에 대한 플러그인을 정의하고, 테스트 프로세스 tailoring을 수행하는 방안을 제안한다. 마지막 4장에서는 결론 및 향후 연구 과제를 기술한다.

2. 관련 연구

2.1 컴포넌트 기반 개발

본 논문에서는 프로세스 표준을 실제 개발 도메인에 맞도록 tailoring하는 방안으로써, 컴포넌트 기반 소프트웨어 개발의 개념을 이용한다. 따라서 본 장에서는 컴포넌트 기반 개발에 대해 핵심적인 내용만을 기술한다.

컴포넌트 기반 개발, 즉 CBD(Component-Based Development)의 기본 단위가 되는 컴포넌트에 대한 정의는 각양각색이다. 이러한 다양한 정의들은 다음을 공통적으로 언급하고 있다. '컴포넌트는 응집력을 갖는 소프트웨어 구현으로서, 독립적으로 개발되고, 인터페이스에 대한 명시적이고 잘 정의된 스팩을 갖는다. 컴포넌트 그 자체를 수정하지 않고 다른 컴포넌트들과 조립될 수 있고, 그의 특성을 맞춤할 수 있다[4].' 그러므로 컴포넌트는 반드시 인터페이스를 제공해야 하며 인터페이스를 통하여 소스 코드의 변경 없이 컴포넌트에 접근할 수 있다.

CBD는 크게 3개의 관점으로 나타낼 수 있다[5]. 이미 만들어진 컴포넌트를 이용하여 새로운 소프트웨어를 구성하는 통합자(integrator)관점, 재사용될 수 있는 컴포넌트를 만드는 제공자(vendor) 관점, 그리고 제공자가 만들어놓은 컴포넌트를 통합자가 효과적으로 이용할 수 있도록 도와주는 기능을 하는 중간자(broker) 관점등이 있다. 이 가운데 주로 통합자(integrator)관점에서 CBD를 보기 쉬우나, 완벽한 CBD가 되려면 이 세 가지가 모두 고려되어져야 한다.

2.2 표준에 대한 Tailoring

국제 표준 기관인 ISO/IEC에서는 소프트웨어 개발 프로세스에 대한 표준을 ISO/IEC 12207에서 제공하고 있다. 표준에서 정의된 프로세스는 개발 기관 및 개발 프로젝트의 크기에 상관없이 표준에서 정의한 프로세스를 개발 목적에 맞게 선택할 수 있도록 되어있다. 표준에서 정의한 프로세스는 개발 방법론(구조적, 객체지향, 컴포넌트 기반 등), 개발 및 테스트 기법, 개발 생명 주기 모델(폭포수, incremental, evolutionary 등), 개발 언어에 상관없이 모두 준수할 수 있도록 되어 있다. 즉, 표준에 정의된 프로세스와 그에 따른 활동(activity), 작업(task)은 준수해야할 필수적인 내용(what-to-do)을 제시하기 때문에, 실제 개발 현장에의 적용(how-to-do)을 위하여 개발 환경에 맞는 표준 프로세스의 선택과 특정 개발방법 및 기술을 추가하는 '표준의 tailoring'을 하도록 정의하고 있다. ISO/IEC에서는 표준에 대한 지침을 ISO/IEC TR 15271[8]로 제공하고 있다. 그러나 ISO/IEC TR 15271은 tailoring을 위한 활동을 정의하고 그 활동에 대한 설명을 해 놓았을 뿐, 그를 위한 구체적 기법을 제공하지 않는 추상적 지침에 불과하다. 따라서 기존의 표준들이 갖고 있던 문제와 마찬가지로 일반 사용자들이 이해하기 어렵다는 문제점을 여전히 갖고 있다.

이러한 문제를 해결하기 위하여 본 논문에서는 표준을 일반적인 개념인 객체지향적으로 표현하여 컴포넌트화하고, 그의 tailoring에 CBD를 적용하여, 체계적이고 구체적인 tailoring 기법을 제시하여, 일반 사용자가 표준에 대한 최소의 지식으로 다양한 도메인을 위한 tailoring을 수행할 수 있도록 한다.

2.3 컴포넌트 기반 개발에서의 주요 활동

컴포넌트를 기반으로 소프트웨어를 개발할 때, 컴포넌트는 현재 개발자와는 다른 개발자에 의해 다른 목적으로 작성되는 특성을 가지므로, 개발자가 모든 코드를 새로 개발하는 기존의 개발 방법과는 다르게 새로운 활동이 요청된다. 그림 1은 CBD에서 요구되는 주요 활동들을 나타낸다.

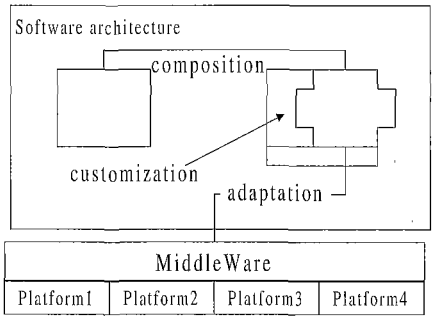


그림 1 CBD에서의 주요 활동

첫째, 컴포넌트는 컴포넌트 내의 세부 구현 사항들을 외부에 노출시키지는 않지만, 컴포넌트 사용자들이 자신의 목적에 맞도록 컴포넌트가 지니는 속성이나 메소드들을 변경할 수 있도록 지원되어야 한다. 즉, 일반적인 요구

사항에 의해 만들어진 컴포넌트들을 특정 요구사항에 맞추어 수정하는 작업이 요구된다. 이렇게 컴포넌트의 인터페이스를 통해 컴포넌트를 새로운 어플리케이션에 맞추는 작업을 컴포넌트 맞춤[4]이라고 한다. 컴포넌트 맞춤은 두 가지 범위로 수행될 수 있는데, 그 하나가 주어진 속성 집합에서 선택을 하는 맞춤이고, 또 다른 하나는 컴포넌트의 기능을 구체적으로 확장하거나 변경하기 위해 새로운 코드를 추가하는 맞춤이다.

둘째, 이미 이종의 컴포넌트 아키텍처에서 개발된 컴포넌트를 새로운 컴포넌트 아키텍처에 맞추는 작업이 요구되는데, 이를 컴포넌트 개조(adaptation)[4,5]라고 한다. 컴포넌트 개조는 EJB, CORBA등의 컴포넌트 아키텍처, 즉 컴포넌트가 개발된 컴포넌트 아키텍처와 컴포넌트가 사용될 컴포넌트 아키텍처의 특성을 정확하게 이해하고 수행되어져야 한다.

셋째, 맞춤과 개조가 이루어진 컴포넌트들을 하나의 시스템으로 묶어주는 활동이 컴포넌트 조립(composition)이다[6]. 컴포넌트 조립은 서로 다른 컴포넌트들이 연결되어 좀 더 큰 기능 수행할 수 있도록 해 주는 활동으로서 컴포넌트들의 인터페이스를 통해서 이루어진다. 두 컴포넌트들의 인터페이스들은 쉽게 연결되지 않는다. 이들 사이의 불일치가 존재할 수 있는데, 이를 해결하기 위해 중간에 연결자(connector)를 두어야 한다. 이 연결자의 패턴을 추출하여 연결자 모델을 구축하는 연구[7]가 진행되고 있다.

위와 같은 컴포넌트 기반 개발에 대한 다양한 개념들을 정의하고 프로세스를 제시하고 있는 대표적인 방법론으로 Catalysis를 들 수 있다. Catalysis는 객체와 프레임워크 개념을 사용한 컴포넌트 기반 개발을 지원하는 방법론이다[4]. Catalysis의 특징은 네가지로 볼 수 있다. 첫째, 프레임워크를 이용한 컴포넌트 기반 개발을 지원한다는 것이다. 비즈니스 모델, 요구사항, 설계, 코드등이 모두 일반적인 컴포넌트와 프레임워크를 재사용하여 구축될 수 있다. 둘째, 명확한 개발 프로세스를 제공한다는 것이다. 셋째, UML, Java, COM, CORBA등 기존의 기술들 간의 관계를 정의하였다. Catalysis가 제공하는 개발 프로세스는 개발 산출물들이 구성되고, 개발되고, 진화되는 방법을 기술한다. Catalysis는 단일 프로세스를 제공하기보다는 '프로세스 패턴'을 제공하여, 개발 프로세스가 다양한 프로젝트에 적용될 수 있도록 하는 장점을 갖는다. 반면에 catalysis는 너무 많은 표기법을 사용하므로, 이해하기 어려운 단점이 있다.

3. 테스트 프로세스 Tailoring

본 논문은 표준의 내용을 tailoriing하는 방법으로 표준에 기반한 테스트 프로세스를 객체지향적으로 표현하여 컴포넌트화하고, 컴포넌트에 CBD의 맞춤과 조립 개념을 적용하는 방안을 제안한다.

본 논문은 컴포넌트 기반 개발 개념을 적용하여 테스트 프로세스의 tailoring 방안을 제안한다. ISO/IEC 12207은 테스트를 위한 프로세스를 정의하고 있으나, 그 내용이 불충분하여, IEEE 테스트 표준[9,10,11,12]들을 추가 참고하여 개발한 MaRMI-II[13] 테스트 프로세스를 기초로 하여, 테스트 tailoring 방안을 기술하겠다.

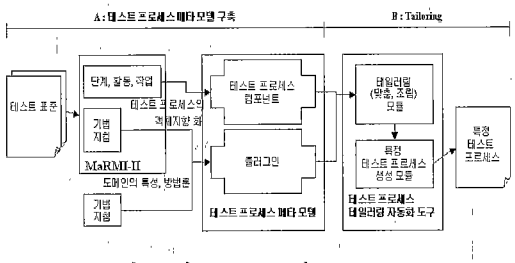


그림 2 테스트 프로세스 tailoring

제안하는 테스트 프로세스 tailoring의 기본 아이디어는 그림 2와 같다. 테스트 프로세스 메타모델을 구축하고, 테스트 메타모델의 tailoring을 통하여 특정 개발 도메인을 위한 테스트 프로세스를 생성할 수 있도록 한다. MaRMI-II 테스트 프로세스를 객체 지향적으로 모델링하여 테스트 프로세스 메타모델을 구축한다. 테스트 프로세스 메타모델은 테스트 프로세스 컴포넌트와 다양한 개발 도메인을 위해 필요한 기법과 지침을 위한 플러그인으로 구성된다. 테스트 프로세스 메타 모델의 tailoring은 컴포넌트 기반 개발에서의 맞춤 기술을 도입하여 해결한다. 즉, 테스트 프로세스 메타 모델을 구성하는 테스트 프로세스 컴포넌트와 플러그인은 맞춤과 조립의 과정을 거쳐서 특정 도메인을 위한 테스트 프로세스가 생성되며 이는 테스트 프로세스 테일러링 자동화 도구에 의해 구현된다.

3.1 테스트 프로세스 메타 모델 구축

테스트 프로세스 메타 모델은 테스트 프로세스 컴포넌트와 플러그인으로 구성한다.

(1) 테스트 프로세스 컴포넌트

테스트 프로세스 자체도 소프트웨어 산출물이기 때문에 테스트 프로세스도 CBD에서의 컴포넌트화가 가능하다. 그러나, 일반적인 컴포넌트들이 실행코드 또는 설계 산출물 등의 특정 대상을 위한 정형적인 명세에 기반하

고 있다. 즉 테스트 프로세스 tailoring에 컴포넌트 기반 개발 패러다임을 적용하기 위해, 표준에 기술된 핵심적인 테스트 프로세스에 관련된 내용들을 컴포넌트의 모습으로 구성한 것이, 테스트 프로세스 컴포넌트이다. 본 논문은 테스트 프로세스를 이루는 활동, 작업과 산출물을 객체지향적으로 표현한 클래스도와 순서도 형태의 테스트 프로세스 메타 모델을 아래와 같이 정의하였다.

<정의 1> 테스트 프로세스 컴포넌트

테스트 프로세스 컴포넌트는 표준에 따라 반드시 수행되어야 하는 수정 불가능한 '블랙박스' 영역과 tailoring을 가능하도록 하는 인터페이스로 그림 3과 같이 구성한다. 테스트 프로세스를 이루는 활동, 작업과 산출물은 클래스도로써, 그리고 테스트 프로세스에 따른 클래스들의 오퍼레이션들 사이의 상호작용은 순서도로 표현한다.

테스트 프로세스 컴포넌트를 MVC[14]에 따라 추출하였다. MaRMI-II 테스트 프로세스에서 MVC의 모델에 해당하는 TestModel 패키지를 추출하고, 모델을 이용하여 테스트 작업을 수행하는 MVC의 콘트롤로

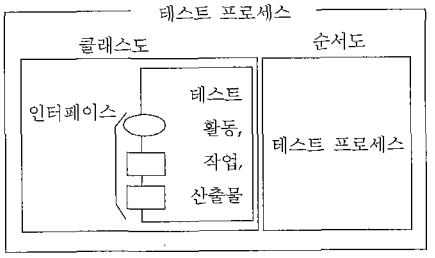


그림 3 테스트 프로세스 컴포넌트의 구조

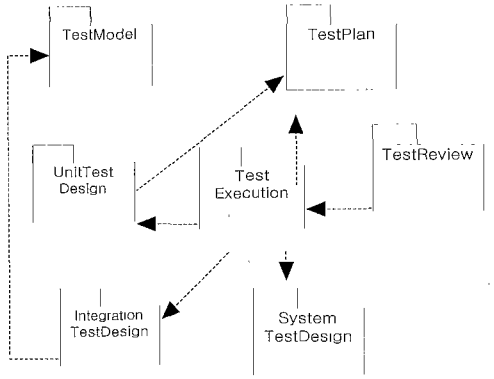


그림 4 테스트 프로세스 메타 모델의 패키지도

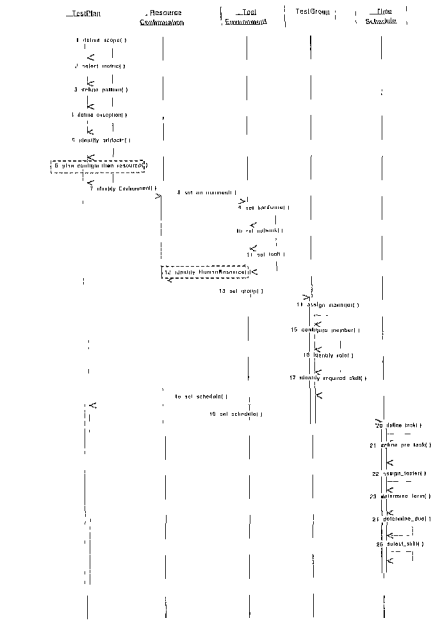


그림 5 TestPlan 컴포넌트

서 TestPlan, UnitTestDesign, IntegrationTest Design, SystemTestDesign, TestExecution, Test Review 패키지를 추출하였다. MVC의 뷰는 개발 프로세스에서 생성되는 각종 다이어그램과 문서들로 보고, 본 테스트 프로세스에서는 고려하지 않았다.

이들 각 패키지는 테스트 프로세스 컴포넌트로 다음과 같이 정의된다. 클래스도와 순서도에서의 점선 상자는 컴

포넌트의 인터페이스를 나타낸다.

1) TestPlan 컴포넌트

각 테스트 단계를 위한 계획을 수립하는 컴포넌트로서, MaRMI-II 테스트 프로세스의 '테스트 계획 수립'에 해당한다. 그림 5는 TestPlan 컴포넌트의 클래스도와 순서도이다. 정의 1에 따라 TestPlan 컴포넌트는 클래스도와 순서도로 표현된다. MaRMI-II 테스트 프로세스에서 테스트 계획에 해당하는 활동, 작업, 산출물을 객체지향적으로 표현한 것이 그림 5의 클래스도이고, 클래스도의 각 메소드들을 MaRMI-II 테스트 프로세스에서 명시한 순서대로 표현한 것이 그림 5의 순서도이다.

2) TestModel 컴포넌트

개발 산출물로부터 테스트 항목과 테스트 케이스 추출등의 테스팅 정보 수집을 위해 개발된 UML 테스트 모형[15]이다. 그림 6은 TestModel 컴포넌트의 클래스도와 순서도이다. 이들도 TestPlan 컴포넌트의 클래스도, 순서도와 마찬가지로 클래스도는 MaRMI- II 테스트 프로세스에서 테스트 모형과 관련된 활동, 작업, 산출물들을 객체지향적으로 표현하였고, 순서도는 이 클래스도의 메소드들이 MaRMI-II 테스트 프로세스에서 수행되는 순서를 표현하였다. TestModel 컴포넌트에서 ASF란 Atomic System Function의 약자로서 입력과 출력 이벤트를 기준으로 묶이는 시스템 기능 단위를 의미한다.

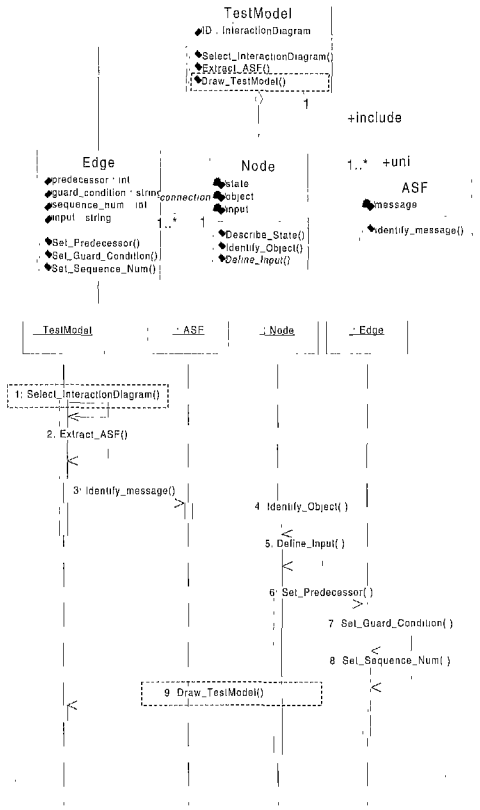


그림 6 TestModel 컴포넌트

3) UnitTestDesign 컴포넌트

기본 단위 테스트 설계를 위한 테스트 항목과 테스트 케이스를 추출하는 컴포넌트로서, MaRMI-II 테스트 프로세스의 '클래스 테스트 설계'에 해당한다. 그림 7은 UnitTestDesign 컴포넌트의 클래스도와 순서도이다.

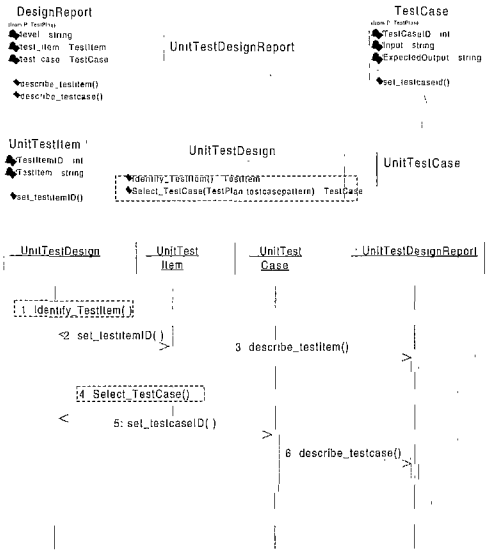


그림 7 UnitTestDesign 컴포넌트

4) IntegrationTestDesign 컴포넌트

통합 테스트 설계를 위한 테스트 항목과 테스트 케이스를 추출하는 컴포넌트로서, MaRMI-II 테스트 프로세스의 '통합 테스트 설계'에 해당한다. 그림 8은 IntegrationTestDesign 컴포넌트의 클래스도와 순서도이다.

5) SystemTestDesign 컴포넌트

시스템 테스트 설계를 위한 테스트 항목과 테스트 케이스를 추출하는 컴포넌트로서 MaRMI-II 테스트 프로세스의 '시스템 테스트 설계'에 해당한다. 그림 9는 SystemTestDesign 컴포넌트의 클래스도와 순서도이다.

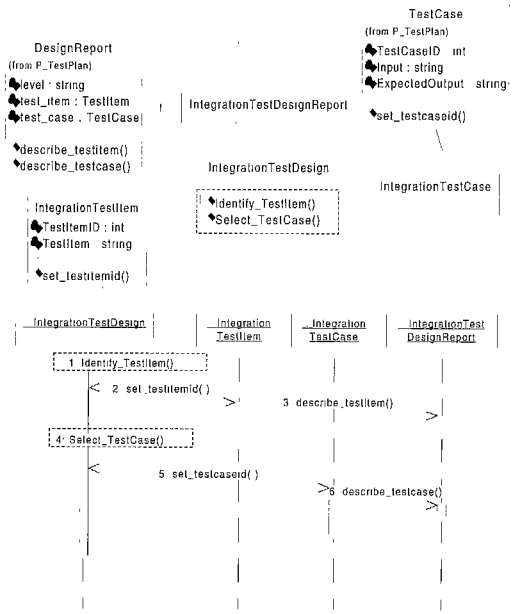


그림 8 IntegrationTestDesign 컴포넌트

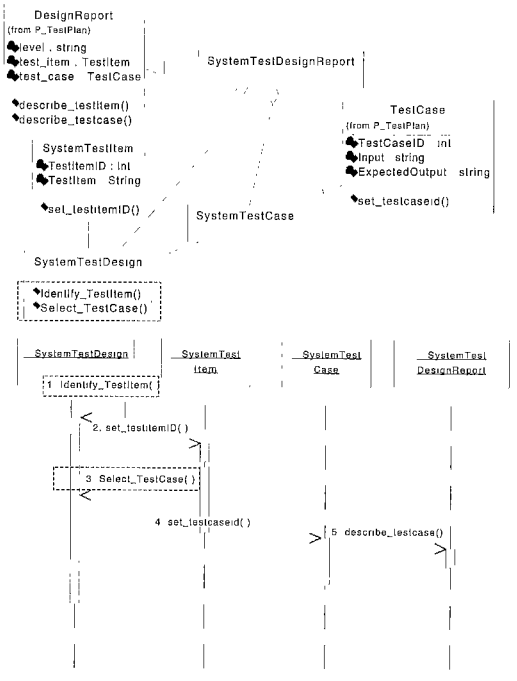
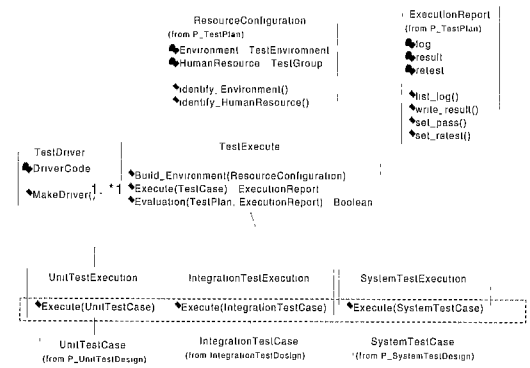


그림 9 SystemTestDesign 컴포넌트

6) TestExecution 컴포넌트

테스트 케이스를 이용하여 테스트를 수행하는 컴포넌트로서, MaRMI-II 테스트 프로세스의 '클래스 테스트 수행', '통합 테스트 수행', '시스템 테스트 수행'에 해당한다. 그림 10은 TestExecution 컴포넌트의 클래스도와 순서도이다.



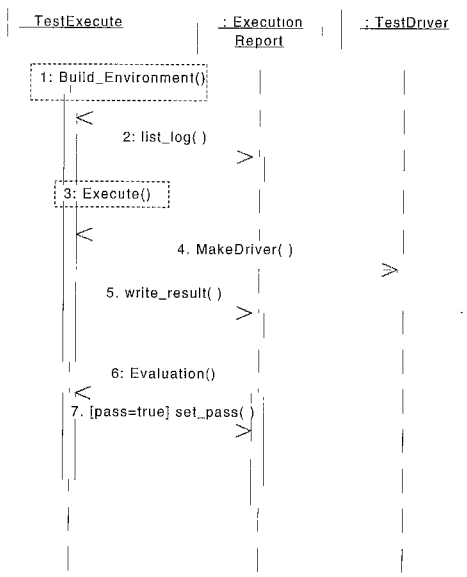


그림 10 TestExecution 컴포넌트

7) TestReview 컴포넌트

테스트 수행 결과를 검토하는 컴포넌트로서, MaRMI-II 테스트 프로세스의 '클래스 테스트 검토', '통합 테스트 검토', '시스템 테스트 검토'에 해당한다. 그림 11은 TestReview 컴포넌트의 클래스도와 순서도이다.

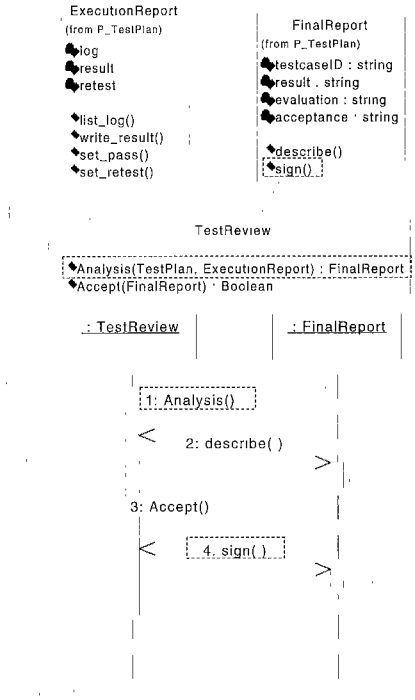


그림 11 TestReview 컴포넌트

(2) 플러그인

테스트 프로세스 컴포넌트는 인터페이스를 통하여 플러그인을 적용하여 프로세스의 tailoring이 이루어진다. 플러그인은 일반적 요구사항에 맞춘 컴포넌트를 특정 요구사항에 맞추어 특정화하기 위한 구체적 구현단위로서, 적절한 플러그인을 선택함으로써, 컴포넌트 맞춤화를 수행할 수 있다. 본 논문에서의 플러그인은 각 테스트 프로세스 컴포넌트의 인터페이스의 오퍼레이션을 구체화할 수 있는 기법이나 지침으로써 클래스로 표현된다. 표 1은 IntegrationTestDesign 컴포넌트의 기법에 대한 플러그인의 목록의 예이다.

표 1 IntegrationTestDesign 테스트 프로세스 컴포넌트의 플러그인의 예

테스트프로세스 컴포넌트	플러그인	설명
Integration Test Design	InterClassTest Design	클래스단위의 통합테스트의 테스트 케이스 선정 기법을 표현한다.
	CompositionTest Design	컴포넌트단위의 통합테스트의 테스트 케이스 선정 기법을 표현한다.
	InterDistributed ObjectTestDesign	분산객체단위의 통합테스트의 테스트 케이스 선정 기법을 표현한다.

3.2 테스트 프로세스 Tailoring

3.1절에서는 CBD의 제공자 입장에서 테스트 프로세스 컴포넌트를 개발하였다. 프로세스 컴포넌트를 특정 도메인에서 이용하기 위해서는 이것을 CBD의 컴포넌트 통합자 입장에서 도메인에 맞추는 tailoring이 필요하다. Tailoring은 테스트 프로세스 컴포넌트의 맞춤과 조립(composition)을 통하여 이루어지고, 이로써 그림 2에서처럼 특정 테스트 프로세스가 생성된다. 이 과정은 테스트 프로세스 테일러링 자동화 도구를 통하여 자동화된다.

(1) 테스트 프로세스 컴포넌트 맞춤

테스트 프로세스 컴포넌트는 실제 개발 현장에의 적용을 위하여 개발 환경에 맞는 표준 프로세스의 선택과 특정 개발방법 및 기술 을 추가하는 tailoring 작업을 위하여 CBD의 맞춤 기법을 적용할 수 있다. 맞춤은 테스트 프로세스 컴포넌트의 인터페이스에 기법과 지침을 갖는 플러그인을 끼워 넣음으로써 이루어진다. 본 논문에서 플러그인은 주어진 테스트 프로세스 컴포넌트의 인터페이스를 특정화하는 내용을 담고 있어서, 이를 인터페이스에 연결시킴으로써, 컴포넌트의 행위에 변형을 줄 수 있는 역할을 한다.

컴포넌트 맞춤 패턴은 크게 '컴포넌트 맞춤 구문(syntactic) 패턴'과 '컴포넌트 맞춤 의미(semantic) 패턴'으로 나뉜다[17,18]. 컴포넌트 맞춤 구문 패턴은 '어떻게' 컴포넌트 맞춤을 하는지에 대한 패턴으로써 1)직접 인터페이스를 수정하거나, 2)플러그인을 연관관계로 연결하거나, 3)상속관계로 연결할 수 있다. 맞춤 의미 패턴은 컴포넌트를 '왜' 컴포넌트 맞춤을 하는지에 대한 패턴으로써, 컴포넌트 '속성(property) 변경 맞춤'과 '기능 추가 맞춤'으로 구분된다. 속성 변경 맞춤은 인터페이스를 통해서 컴포넌트의 속성 변수의 값을 수정하거나 조건을 부여하여 변경시키는 경우이며, 기능 추가 맞춤은, 개발 도메인에서 요구되는 특정한 기능을 컴포넌트에 플러그인을 사용하거나 새로운 코드를 작성하여 특정한 기능을 추가하는 경우이다. 컴포넌트 맞춤 패턴은 기존의 설계 패턴[19]을 기반으로 추출하였다.

그림 12는 테스트 프로세스 컴포넌트의 맞춤과 조립을 표현한다. 테스트 프로세스 tailoring에서 맞춤의 경우, 그림 12처럼 테스트 프로세스 컴포넌트의 인터페이스에 플러그인을 연관관계로 연결하여, 특정 기법 혹은 테스트 작업에 해당하는 플러그인이 연결될 수 있도록 한다. 테스트 프로세스 컴포넌트 맞춤은 컴포넌트 기반 개발 개념에 따라서 주어진 인터페이스만

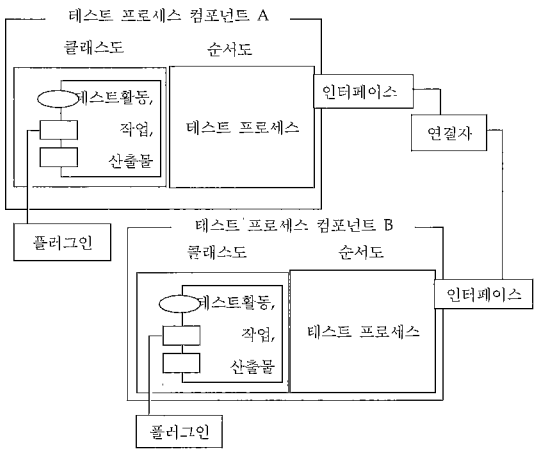


그림 12 테스트 프로세스 컴포넌트 맞춤 및 조립

을 변형할 뿐 그 나머지 부분은 변형할 수 없다.

플러그인을 연결할 경우, 테스트 프로세스 컴포넌트 맞춤은 해당되는 인터페이스에서 플러그인을 호출하도록 인터페이스를 변형시킨다. 즉, 인터페이스에 해당하는 오퍼레이션을 수행하게 되면 연결된 플러그인은 그 오퍼레이션내에서 호출되어 수행된다. 또한 클래스도와 함께 컴포넌트를 이루는 순서도는 테스트 프로세스 컴포넌트의 활동 및 작업들 사이의 순서관계를 표현하며. 플러그인은 테스트 프로세스 컴포넌트에 포함되는 것이 아니라 단지 맞춤을 통해 연결될 뿐이다. 따라서 본 논문에서 정의한 순서도의 의미상 플러그인이 순서도에 추가되지 않으며, 순서도에 추가되는 작업없이 인터페이스에 해당하는 오퍼레이션이 순서도에서 수행될 때, 연결된 플러그인이 수행될 것이다. 즉, 인터페이스에 해당하는 오퍼레이션을 수행하게 되면 연결된 플러그인은 그 오퍼레이션내에서만 호출되고 호출된 후 반드시 해당 오퍼레이션으로 복귀하여 마치 인터페이스의 오퍼레이션이 플러그인 내용을 포함하고 있는 것처럼 수행된다. 모든 플러그인은 해당 인터페이스만을 구체화시키므로 플러그인 수행후 연결된 인터페이스로 복귀할 뿐 플러그인에서 컴포넌트의 다른 곳으로 순서가 진행되지 않는다. 컴포넌트의 클래스도에 플러그인이 연관관계로 삽입되고, 순서도에는 변화가 없다. 인터페이스 클래스의 오퍼레이션이 순서도에서 수행될 때, 각 오퍼레이션은 연관 관계에 의해 플러그인의 오퍼레이션을 호출하게 된다. 따라서 클래스도에 플러그인을 연관관계로 삽입함으로써, 순서도의 수정없이 플러그인의 내용을 반영할 수 있다.

예로써, 테스트 프로세스 컴포넌트 가운데, IntegrationTestDesign 컴포넌트를 표 1 InterClassTest Design 테스트 기법 플러그인을 적용하여 맞춤을 수행한 결과를 그림 13에 나타내었다. InterClassTest Design 플러그인은 IntegrationTestDesign 테스트 프로세스 컴포넌트의 인터페이스에 있는 Identify_ Test

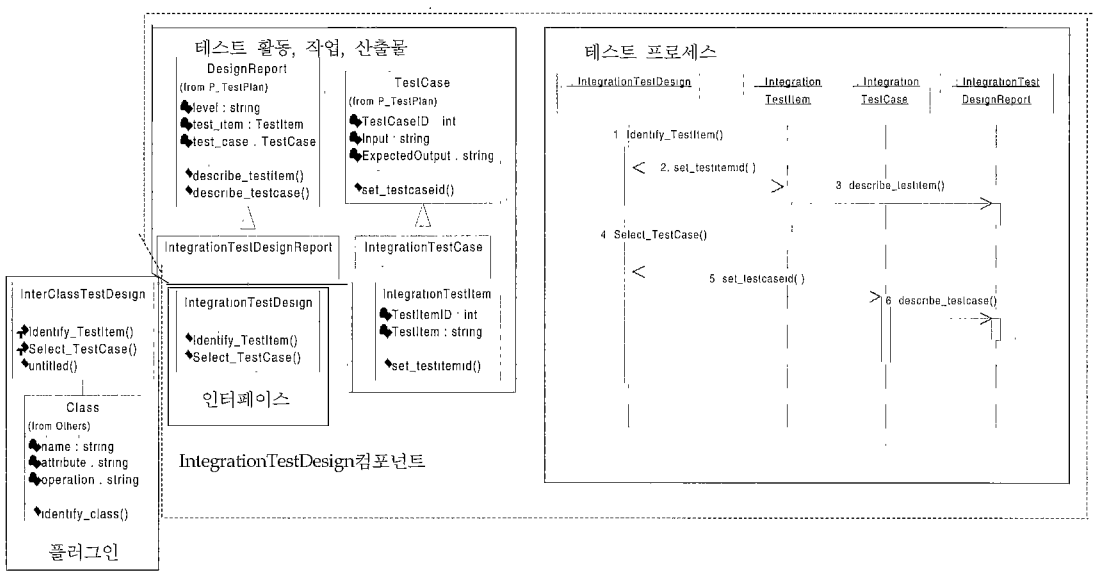


그림 13 InterClassTestDesign 플러그인을 이용하여 맞춤된 IntegrationTest Design 컴포넌트

Item()과 Select_TestCase() 메소드에 대한 구체적인 테스트 작업과 기법, 즉 Inter-Class 테스트를 위한 테스트 작업과 기법을 갖고 있다. 그림 13은 플러그인을 이용한 맞춤이 수행된 이후의 IntegrationTest Design 컴포넌트의 클래스도와 순서도이다.

(2) 테스트 프로세스 컴포넌트 조립

완전한 테스트 프로세스를 구축하기 위해서는 앞절에 기술한 7가지의 테스트 프로세스 컴포넌트들을 연결하여야 한다. 이들을 CBD의 조립 기술을 적용하여 해결할 수 있다. 즉, 테스트 프로세스 컴포넌트는 '컴포넌트-포트(port)-연결자(connector)'[4] 컴포넌트 아키텍처 모델을 구성하도록 한다. 본 논문에서는 한 컴포넌트의 'required' 인터페이스와 또 다른 컴포넌트의 'provided' 인터페이스를 조립을 위한 포트로 본다.

그림 12처럼 맞춤이 이루어진 두 개의 테스트 프로세스 컴포넌트들의 조립은 각 컴포넌트의 순서도를 이용하여 수행된다. 각 순서도에서 다른 컴포넌트와 상호작용을 하는 부분인 인터페이스들을 서로 연결하여 컴포넌트들을 조립한다. 이때 두 인터페이스들 사이에 간격을 연결자가 메꾸어준다. 조립 결과 생성되는 것은 맞춤되어진 컴포넌트들의 순서도들이 연결되어 생성된 하나의 순서도이다.

예를 들어 맞춤이 이루어진 두 개의 테스트 프로세스 컴포넌트: IntegrationTestDesign 컴포넌트와 TestModel 컴포넌트가 조립된 경우에서 Integration TestDesign 컴포넌트의 인터페이스는 Identify_TestItem()과 Select_TestItemID()이고, TestModel 컴포넌트의 인터페이스는 Select_InteractionDiagram()과 Draw_TestModel()이다. 이 경우 그림 14와 같이 두 컴포넌트 각각의 테스트 프로세스를 표현하는 순서도들이 서로 인터페이스를 통해 조립되어진다. 본 논문의 컴포넌트 조립 결과 생성되는 것은, 그림 2에 나타내었듯이, 특정 테스트 프로세스 모델이다.

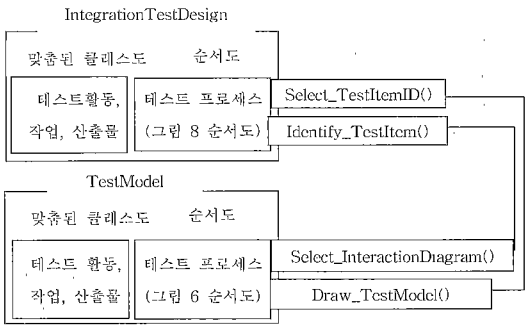


그림 14 IntegrationTestDesign 컴포넌트와 TestModel 컴포넌트의 조립

일반적으로 테스트 프로세스 컴포넌트들의 조립의 경우, 컴포넌트의 인터페이스들 사이의 전이가 자연스럽게 이루어지므로, 연결자는 필요하지 않다. 따라서 그림 14의 IntegrationTestDesign 컴포넌트와 Test Model 컴포넌트 조립에서는 연결자가 존재하지 않는다. 앞서 언급한대로 본 논문에서의 컴포넌트는 자연어로 기술한 테스트 프로세스를 tailoring을 위하여 컴포넌트의 형태로 구성한 것이다. 따라서 테스트 프로세스 컴포넌트들의 조립 결과는 두 개 이상의 테스트 프로세스 컴포넌트를 이루는 활동 및 작업들이 각각의 순서도에 따라 나열되어져 있고 이들이 단지 연결되는 모습이다. 본 논문의 테스트 프로세스 컴포넌트는 실행이 되는 구현 코드를 갖는 것이 아니므로 두 컴포넌트 사이의 이벤트 흐름이나 컴포넌트 사이의 불일치를 해결하는 랩핑 등을 위한 연결자는 반드시 고려하지 않아도 된다. 그러나 자연어로 기술된 행위를 갖는 연결자를 고안하여 테스트 프로세스 컴포넌트 조립을 위한 연결자 모델을 구축한다면, 좀더 체계적인 조립을 수행할 수 있을 것이다. 이에 대한 구체적인 연구를 현재 진행하고 있다.

(3) 테스트 프로세스 생성

앞에 기술한 테스트 메타 모델의 맞춤·조립 결과, 특정 도메인에 tailoring된 테스트 프로세스를 생성할 수 있다. 이를 위해 앞에서 제안한 테스트 프로세스 의 tailoring 기법을 위한 "테스트 프로세스 테일러링 자동화 도구" [16,20]의 프로토타입을 개발하였다. "테스트 프로세스 자동화 도구"를 통하여 표준으로부터 추출한 테스트 프로세스로부터 사용자가 원하는 도메인으로 tailoring된 테스트 프로세스를 자동 생성할 수 있게 된다. 테스트 프로세스의 주요 기능은 다음과 같으며, 그 프로토타입은 그림 15와 같다.

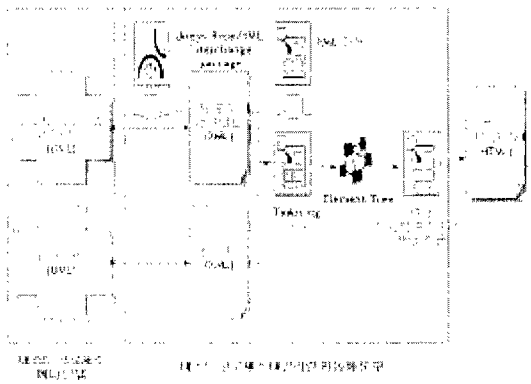


그림 15 테스트 프로세스 테일러링 자동화 도구

1) 테스트 프로세스 메타 모델을 위한 XML 문서 생성:
테스트 프로세스 메타 모델을 구성하는 테스트 프로세스 컴포넌트와 플러그인에 대한 XML 문서를 생성한다.

2) 테스트 프로세스 메타 모델의 tailoring:
XML로 표현된 테스트 프로세스 메타 모델을 tailoring 기법에 따라 tailoring한다.

3) HTML형태의 문서로 자동 생성:
tailoring된 테스트 프로세스 메타 모델에 대한 도메인이 반영된 특정 테스트 프로세스를 HTML형태의 문서로 생성한다.

5. 결론 및 향후 연구과제

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 컴포넌트로 해결할 수 있다.

본 논문에서 개발한 테스트 프로세스 컴포넌트를 이용하여 표준의 테스트 프로세스를 도메인의 특성에 적합한 테스트 프로세스로 쉽게 tailoring할 수 있다. 따라서 사용자는 프로세스 컴포넌트의 인터페이스를 제외한 나머지 부분에 대해서는 고려할 필요없이, 주어진 인터페이스와 플러그인들을 이용하여 맞춤과 조립만을 수행함으로써, 체계적으로 tailoring된 특정 테스트 프로세스 모델을 구축할 수 있다. 나아가 이를 특정 테스트 프로세스로 구축하여 테스트에 적용한다. 특정 테스트 프로세스 모델로부터 특정 테스트 프로세스를 생성하는 것은 XML을 이용하여 자동화할 수 있는 부분으로서, 현재 프로토타입이 구현되어 있으며 보다 완벽한 tailoring 지원을 위하여 계속 연구가 진행중이다.

본 논문은 구현 기술로만 활용되는 컴포넌트 기반 개발 패러다임을 프로세스 tailoring에 적용하였다는 의미를 갖는다. 따라서 더욱 완벽한 기법을 구축하기 위해, 다음의 몇몇 과제들이 향후 진행되어져야 하며, 현재 이들을 순서대로 진행하고 있다.

첫째, 본 논문의 프로세스 컴포넌트는 표준에 기반한 테스트 프로세스를 대상으로 정의되었으나, 나아가 전체 개발을 위한 프로세스도 표준으로부터 추출하여, 개발 프로세스 컴포넌트를 개발한다면, 테스트를 포함한 전체 개발 프로세스가 다양한 도메인의 특성에 맞게 tailoring될 수 있을 것이다. 이를 실제로 다양한 개발 환경에서 이용하기 위해서는 각 프로세스 컴포넌트마다 다양한 플러그인들 - 분산환경을 위한, 컴포넌트로 구성된 소프트웨어를 위한, 실시간 시스템을 위한, 금융도메인을 위한, 등등 -을 개발하는 일이 요구된다. 현재 전자상거래 도메인을 대상으로 본 논문의 기법을 적용하는 사례연구가 진행중이다. 또한, 본 논문에서의 테스트 프로세스 컴포넌트는 표준화된 테스트 프로세스를 나타내는 많은 특성들 중에 활동, 작업, 산출물을 중심으로 정의되었고, 플러그인은 이에 대한 구체적인 기법이나 지침 등의 내용으로 정의되어 있으나, 현재 그 밖의 도메인에 따른 목적과 사용자 요구사항, 기술적인 작업 요구사항 등을 반영하는 플러그인 개발 단계와 이에 따른 tailoring 기법도 추가 정의하므로써 보다 체계적인 tailoring에 대한 연구를 진행 중이다.

둘째, 테스트 프로세스 컴포넌트 조립을 위한 연결자 모델에 대한 연구가 진행될 계획이다. 본 논문은 테스트 프로세스 컴포넌트의 조립이 컴포넌트가 갖는 자연어의 특성상 특정 연결자 없이도 가능하다고 보고 있다. 그러나 체계적인 조립이 이루어지기 위해서는 구체적인 연결자 모델이 구축될 필요가 있다. 이에 대한 연구를 진행할 계획이다.

셋째, 컴포넌트 기반 소프트웨어 개발에서의 테스팅[17]을 본 논문의 테스트 프로세스 tailoring에 적용하여, tailoring된 테스트 프로세스에 오류를 테스트하는 기법에 대한 연구도 진행할 계획이다.

참 고 문 헌

[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] Sherif Yacoub, "Model for classifying component interfaces," Int'l workshop on CBSE, 1999
[6] 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.
[7] Stefan Tai, "A Connector Model for Object0Oriented Component Integration," proceedings of Int'l

workshop on Component-Based Software Engineering, Apr. 1998
[8] ISO/IEC TR15271 : Information Technology - Guide for ISO/IEC 12207
[9] MIL-STD-498, Software Development and Documentation
[10] ANSI/IEEE Std 289-1983 IEEE Standard for Software Test Documentation
[11] ANSI/IEEE Std 1012-1986 IEEE Standard for Software Verification and Validation Plan
[12] ANSI/IEEE Std 1008-1987 IEEE Standard for Software Unit Testing
[13] 최병주, "객체지향 소프트웨어 개발을 위한 테스트 방안에 관한 연구" 최종보고서, SERI, 1998
[14] G. E. Krasner and S. T. Pope. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. JOOP, vol 1, no 3, August/ September, 1988, pp.26-49
[15] Hoijin Yoon, Byoungju Choi, Jin-Ok Jeon, "UML-based Test Model for Component Integration Test," Proceedings on Workshop on Software Architecture and Component, pp.63-70, Dec, 1999
[16] W3C, Extensible Markup Language (XML) 1.0, http://www.w3c.org/TR/REC-xml, 1998
[17] Hoijin Yoon, Byoungju Choi, "Interclass test technique between black-box class and white-box class for component customization failures," APSEC'99, Dec, 1999
[18] 윤희진, 최병주, "컴포넌트 맞춤 오류를 위한 테스트 기법", 정보과학회논문지, 27(2):148-156, 2000.2.
[19] Wolfgang Pree, *Design Patterns for Object-Oriented Software Development*, Addison-Wesley, 1994
[20] Jooyoung Seo, Byoungju Choi, "Tailoring Test Process by using the Component-Based Development Paradigm and the XML technology," APSEC'2000, Dec, 2000, accepted

윤 희 진
정보과학회논문지 : 소프트웨어 및 응용
제 27 권 제 2 호 참조

최 병 주
정보과학회논문지 : 소프트웨어 및 응용
제 27 권 제 2 호 참조