AutoTP : 테스트 프로세스 자동 생성 도구
(AutoTP : Automatic Test Process Generating Tool)

서 주 영 † 최 병 주 ††
(Jooyoung Seo) (Byoungju Choi)

요 약 소프트웨어 프로세스에 관한 표준을 각 프로젝트에서 이용하기 위해서 개발 도메인에 맞도록 테일러링하는 작업이 필요하다. 그러나 기존의 테일러링 작업은 체계적이지 않고, 개발 도메인과 방법론의 분석 없이는 불가능하며 유사한 다른 프로젝트로의 적용도 쉽지 않다.

본 논문에서는 "컴포넌트 기반 개발 개념을 활용한 테일러링 방안"을 기반으로한 체계적인 테일러링 단계와 테스트 프로세스 생성 자동화 알고리즘을 제안하고, XML을 이용하여 구현한 "테일러링 방안의 자동화 도구, AutoTP"를 제안한다. AutoTP 사용자는 프로세스 표준이나 방법론과 도메인을 분석하는 등의 작업 없이도 특정 개발 도메인에 적합한 테스트 프로세스를 자동생성할 수 있다.

Abstract Utilizing standards for software process to a specific project requires a tailoring process to meet the development domain. However, the existing tailoring schemes are not systematical and possible to use without analyzing the methodology and development domain. Also, it is not quite easy to apply them to similar projects.

This paper includes: 1) systematical tailoring steps and 2) an automatic algorithm for generating test process based on "a scheme of tailoring process using the component-based development paradigm"; 3) "an automation tool for tailoring, AutoTP" which is derived from XML techniques. Users can generate a tailored test process through our AutoTP automatically without analyzing standards, methodology and domain.

1. 서 론

최근 소프트웨어의 활용 분야가 다양화되면서 프로젝트의 특성에 적합한 개발 프로세스[1]에 대한 요구가 늘어나고 있다. 표준에 정의된 프로세스와 그에 따른 활동, 작업은 준수해야 할 필수적인 내용(what-to-do)만을 제시하고 있기 때문에, 실제 개발 현장에의 적용(how-to-do)을 위하여 개발 환경에 맞는 표준 프로세스의 선택과 특정 개발 방법 및 기술을 추가하는 그림 1과 같은 '표준의 테일러링'을 하도록 정의하고 있다 [1, 2]. ISO/IEC에서는 표준에 대한 지침을 ISO/IEC TR 15271[1]에서 제공하고 있으나, 실제 개발 프로젝트에 적용할 수준의 프로세스를 위한 지침에는 한계가 있다. 왜냐하면 표준의 어떤 부분을 어떻게 테일러링하여야 하는지에 대한 지침이 부족하기 때문이다. 사실 표준은 표준대로 존재할 뿐, 실제 프로세스 정의에는 제대로 사용되지 못하고 있다. 즉, 다양한 프로젝트의 특성을 반영하기 위하여 특정 방법론의 선택과 개발 방법 및 기술을 추가하는 '표준의 테일러링'을 위한 체계적인 방안이 요구된다.

일반적으로 프로세스 테일러링의 경우, 프로젝트 관리자는 수행하는 프로젝트의 특성에 적합한 방법론을 선

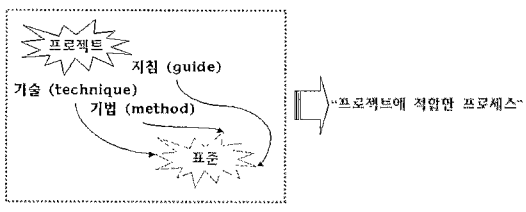

그림 1 표준의 테일러링

· 본 연구는 한국과학재단 목적기초연구(과제번호 : 2000-0-303-02-3) 지원으로 수행되었음.
† 비 회 원 : 이화여자대학교 컴퓨터학과
jyseo@ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
bjchoi@ewha.ac.kr
논문접수 : 2001년 1월 16일
심사완료 : 2001년 5월 31일

택한 후 다양한 체크리스트를 추출하여 이를 자신의 프로젝트 계획 단계에 반영하는 방식으로 테일러링을 수행한다[3,4]. 하지만, 이러한 방식은 프로젝트의 특성을 추출하기 위한 단계가 객관적이지 못하고 방법론을 이해하고 도메인을 분석하는 등의 비용과 시간이 많이 소비되며 유사한 다른 프로젝트에 재 사용할 수 없는 단점을 갖는다.

본 저자는 논문[5]에서 컴포넌트 기반 개발 개념(Component Based Development: CBD)을 적용하여 프로세스의 테일러링 방안을 제안하였다. 이 "CBD 기반의 테일러링 방안"에서는 "핵심 프로세스" 컴포넌트와 다양한 방법론과 도메인을 위한 "플러그인" 컴포넌트를 위한 프로세스 메타 모델을 제안하였다. 프로세스의 테일러링이란 핵심 프로세스 컴포넌트와 플러그인 사이에서 이루어지며, 이때 CBD의 맞춤(customization) 기법을 적용하여 프로세스의 테일러링 기법을 제안하였다.

본 논문에서는 논문[5]에서 제안한 "CBD 기반의 테일러링 방안"을 기초로 하여, 프로세스 생성 자동화 알고리즘을 개발하고 XML 기술을 적용하여 테스트 프로세스 생성 도구(**Auto**matic **T**est **P**rocess generating tool: AutoTP)를 개발함으로써 사용자가 원하는 방법론과 개발 도메인에 테일러링된 테스트 프로세스를 자동 생성 가능하게 한다. AutoTP는 다양한 핵심 방법론(절차적, 객체 지향적, 컴포넌트 기반…등)과 향후 유망할 핵심 도메인(전자 상거래, 금융, 생명과학, 통신, 제조, 응용기술, 운송, 시뮬레이션…등)에 대한 플러그인을 제공할 예정이고, 한번 개발된 플러그인들은 AutoTP내에서 재 사용되어 다양한 프로젝트 환경에 적합한 테스트 프로세스를 자동 생성할 수 있다.

2장에서는 CBD 개념을 활용한 테스트 프로세스 생성 자동화에 대하여, 3장에서는 AutoTP의 설계와 구현을 기술하며 Objectory 방법론과 전자 상거래 도메인을 대상으로 실행 사례를 기술한다. 마지막으로 4장에서는 결론 및 향후 연구 과제를 기술한다.

2. CBD 개념을 활용한 테스트 프로세스 생성 자동화

테스트 프로세스 생성 자동화는 그림 2에서처럼 "테스트 프로세스 메타 모델의 개발"과 "테스트 프로세스 메타 모델의 테일러링"의 두 단계로 이루어진다. 테스트 프로세스 메타 모델은 핵심 테스트 프로세스 컴포넌트와 플러그인으로 나뉘어 개발되며, 이들 테스트 프로세스 메타 모델은 방법론 및 개발 도메인 등의 프로젝트 요구사항에 따라 테일러링되어 프로젝트 요구사항이 반영된 테스트 프로세스가 생성되게 된다.

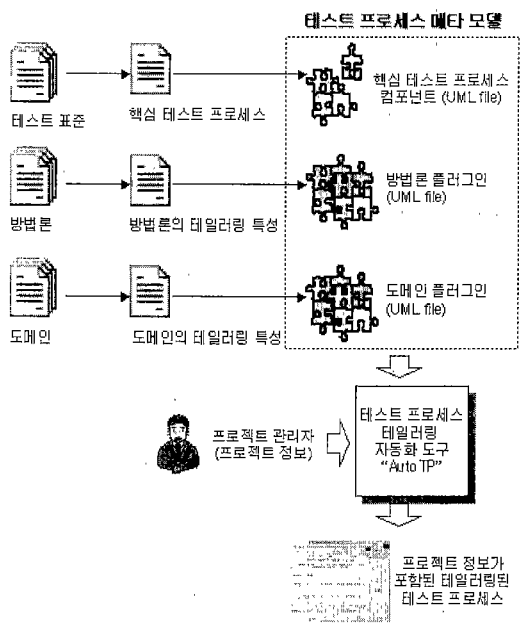

그림 2 테스트 프로세스 생성 과정

논문[5]에서는 이들 프로세스 메타 모델을 정의하였고, CBD의 맞춤 기법을 적용하여 이 메타모델의 테일러링 기법을 제안하였다. 그러나 테일러링에 대한 기본 아이디어는 있으나 프로세스 생성 자동화를 위하여는 보다 체계적이고도 구체적인 테스트 프로세스 생성 자동화 방안이 필요하다. 테스트 프로세스 생성 자동화 방안을 위하여, 테스트 프로세스 메타모델과 테일러링 기법을 간략히 정리한 후, 프로세스 생성 자동화를 위하여 새롭게 제안하는 테일러링 단계와 구체화된 테스트 프로세스 생성 자동화를 기술하겠다.

(1) 테스트 프로세스 메타 모델

테스트 프로세스 메타 모델이란 텍스트 형태로 기술된 테스트 프로세스를 객체지향화하여 UML 표기법[6]으로 표현한 컴포넌트 모델이다. 테스트 프로세스 메타 모델은 그림 3처럼 표준으로부터 추출한 핵심 테스트 프로세스 컴포넌트와 특정 방법론을 반영한 방법론 플러그인과 특정 도메인의 공통 특성을 반영하는 도메인 플러그인으로 구성된다.

테스트 프로세스 메타 모델은 그림 3에서처럼 표준에

따라 반드시 수행되어야 하는 수정 불가능한 블랙박스 영역과 테일러링을 가능하도록 하는 인터페이스로 구성한다. 블랙박스 영역은 정적모형(structural feature)과 동적모형(behavioral feature)으로 구성하는데, 텍스트 형태로 기술된 테스트 프로세스를 테스트 프로세스 메타 모델로 변환하기 위해 컴포넌트 모델링 규칙(Component Modeling Rule: CMR)을 적용한다. 즉, CMR에 의해 테스트 프로세스의 정적 모형은 클래스와 패키지도로 표현한다. 작업은 패키지로 변환하고, 클래스도로 세분화되어 표현한다. 작업의 각 절차는 메소드로, 산출물은 클래스로 표현한다. 테스트 레벨에 따른 구체적인 테스트 프로세스는 인터페이스 클래스로 표현한다. 또한, 테스트 프로세스의 절차의 상호관계, 즉 동적 모형은 순서도로 표현한다.

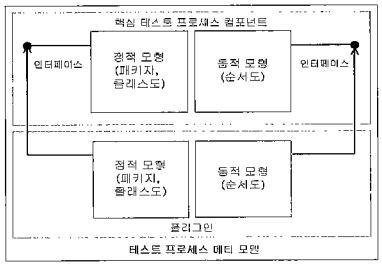

그림 3 테스트 프로세스 메타 모델

(2) 테일러링 기법

테스트 프로세스의 테일러링은 테스트 프로세스 메타 모델의 핵심 테스트 프로세스 컴포넌트와 플러그인들 사이에서 이루어지며, CBD의 맞춤 기법을 적용하여 테스트 프로세스의 테일러링 기법을 제안하였다. 컴포넌트 맞춤은 메소드, 클래스와 패키지의 세 단위별로 연관(association)관계와(또는) 상속(inheritance)관계에 따라 이루어진다[7]. 표 1에서처럼 이들 각 관계에 따라 4가지의 테일러링 패턴, tP1, tP2, tP3, 와 tP4를 정의하였다.

표 1 테일러링 패턴

단위	관계	패턴(tP)
메소드 사이	클래스내의 두 메소드 사이의 연관 관계	tP1
클래스 사이	두 클래스 사이의 연관 관계	tP2
	두 클래스 사이의 상속 관계	tP3
패키지 사이	두 패키지 사이의 연관 관계	tP4

CBD의 맞춤은 1) 직접 인터페이스를 수정하거나, 2) 플러그인을 연관관계로 연결하거나, 3) 상속관계로 연결함으로 수행될 수 있다. 따라서 표 2에서처럼 핵심 테스트 프로세스 컴포넌트와 플러그인의 맞춤을 네가지의 테일러링 패턴 별로 정의함으로써 테스트 프로세스 메타 모델의 테일러링 기법을 제안하였다.

표 2 테일러링 기법

패턴 (tP)	UML 표현	테일러링 기법
tP1	클래스도	새로운 메소드를 연관 관계로 추가
	순서도	새로운 메소드 호출을 관련 순서도에 추가
tP2	클래스도	새로운 클래스를 연관관계로 추가
	순서도	새로운 메소드 호출을 관련 순서도에 추가
tP3	클래스도	새로운 클래스를 상속관계로 추가
tP4	패키지도	새로운 패키지를 연관관계로 추가
	순서도	새로운 순서도를 추가

(3) 테스트 프로세스 생성 자동화

프로젝트의 특성을 반영하는 체계적인 테스트 프로세스의 테일러링을 위하여 그림 4와 같은 세 단계 테일러링 단계를 제안한다. 제1단계는 프로젝트에 적용될 방법론 테일러링이며, 제2단계는 프로젝트가 속하는 도메인에 대한 공통 특성을 테일러링하는 단계이고, 제3단계는 특정 프로젝트만의 특성들을 맞춤 하는 단계를 거쳐서 실제 프로젝트 개발 환경에 적합한 테스트 프로세스를 완성한다. 테일러링 단계를 3 단계로 세분화함으로써 플러그인의 재사용성을 높이고 보다 정확한 테스트 프로세스를 생성할 수 있다.

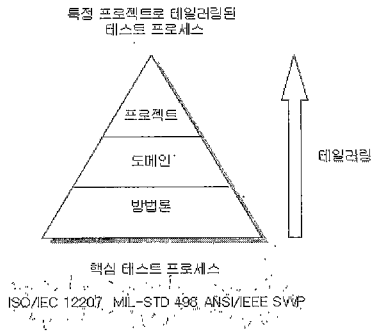

그림 4 테일러링 단계

테스트 프로세스 생성 자동화는 핵심 테스트 프로세스 컴포넌트, 방법론과 도메인 플러그인을 위한 테스트 프로세스 메타모델의 개발과 테일러링의 세 단계에 따른 테스트 프로세스 메타모델의 테일러링으로 아래와 같이 이루

어진다. 테스트 프로세스 메타 모델을 구성하는 각 컴포넌트 및 테일러링 과정의 개발 사례는 3장에 기술하겠다.

<알고리즘> 테스트 프로세스 생성 자동화

Input: 프로젝트 요구사항 (방법론, 개발도메인, 특정 프로젝트 정보)
Output: 프로젝트 요구사항이 반영된 테스트 프로세스
Step:
1 테스트 프로세스 메타 모델 개발
  1.1 핵심 테스트 프로세스 컴포넌트 개발
  1.2 방법론 플러그인 개발
  1.3 도메인 플러그인 개발
2 테스트 프로세스 메타 모델 테일러링
  2.1 핵심 테스트 프로세스 컴포넌트와 플러그인의 테일러링
  2.2 특정 프로젝트 정보 적용
  2.3 프로젝트 요구사항이 반영된 테스트 프로세스 생성

3. AutoTP - 테스트 프로세스 생성 도구

본 논문에서는 2장에서 기술한 테스트 프로세스 생성 자동화 알고리즘을 기반으로 테스트 프로세스 생성 도구인 AutoTP를 개발하였다. AutoTP의 프로토타입을 그림 5에 나타내었는데, 테스트 프로세스 메타 모델의 저장소와 프로세스 테일러링을 수행하는 다섯 개의 모듈로 구성한다. AutoTP는 외부로부터 프로젝트 요구사항인 방법론, 개발도메인, 특정 프로젝트 정보를 입력받고, 테스트 프로세스 메타 모델의 저장소로부터 이들 요구사항에 적합한 텍스트 형태의 테스트 프로세스를 출력한다.

AutoTP는 Windows NT 4.0 환경에서 개발하였으며, 개발 언어로는 Java2 SDK 1.2.2 (JDK1.2.2)으로 구현하였다. 테스트 프로세스 메타 모델은 UML로 작성된 컴포넌트 모델이다. AutoTP는 그림 5에서 처럼 그래픽 정보인 UML의 테스트 프로세스 메타 모델을 처리하기 용이한 XML로 변환한다. XML은 다른 포맷의 자료사이의 변환과 정보 교환에 유리한 장점을 갖고 있다. 따라서 AutoTP는 최종 결과 테스트 프로세스를 텍스트 형태의 다양한 포맷(.예: HTML, TEXT...등)으로 제공할 수 있다. 이를 위하여 XML 1.1 및 Unisys Rose/XML Interchange package[8]를 이용하여 UML을 XML으로 변환하였으며, XMI 포맷을 준수하는 'uml.dtd'를 그대로 수용한다. 프로세스 테일러링 자동화에 사용된 데이타 베이스로는 JDBC를 이용하여 구현하였다.

AutoTP는 테스트 프로세스 생성 자동화 알고리즘의 Step 1인 테스트 프로세스 메타 모델 저장소 개발과 Step 2인 프로세스 테일러링 수행으로 이루어지는데, 본 장에서는 이 두 단계와 AutpTP 수행 사례를 기술한다. 이때 방법론으로는 Objectory 방법론과 도메인으로는 전자 상거래 도메인을 예제로 한정하여 기술한다.

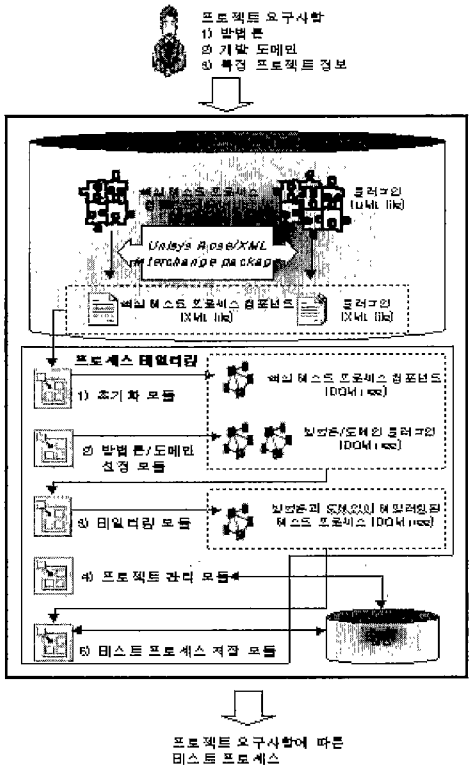

그림 5 AutoTP 프로토타입

3.1 테스트 프로세스 메타 모델 저장소

테스트 프로세스 메타 모델의 개발은 핵심 테스트 프로세스 컴포넌트 개발과 방법론 플러그인 개발 및 도메인 플러그인 개발로 이루어지며, 이들은 테스트 프로세스 메타모델 저장소에 저장된다.

(1) 핵심 테스트 프로세스 컴포넌트 개발

핵심 테스트 프로세스 컴포넌트 개발은 핵심 테스트 프로세스 추출 단계와 핵심 테스트 프로세스 컴포넌트 구현 단계의 2 단계로 진행한다.

1) 핵심 테스트 프로세스 추출

핵심 테스트 프로세스는 테스트 관련 표준들, ISO/IEC 12207, MIL-STD-498 [9], ANSI/IEEE SVV등을 기반으로 테스트에서 반드시 요구되는 핵심의 테스트 프로세스로만 구성한다. 표준으로부터 추출한 핵심

테스트 프로세스는 그림 6에서처럼 작업, 절차와 산출물과 같은 프로세스 요소들로 구성한다. 작업은 새로운 산출물을 생성하는 수준으로서 세부적인 절차를 갖는다.

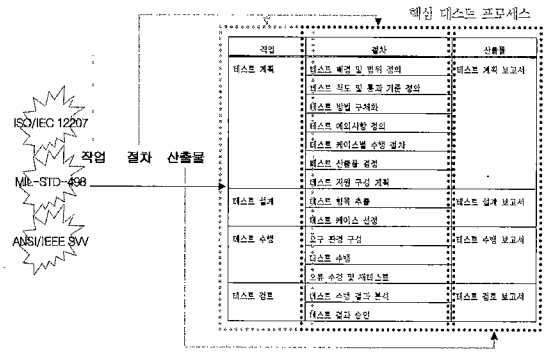

그림 6 핵심 테스트 프로세스 추출

2) 핵심 테스트 프로세스 컴포넌트 구현

그림 6의 핵심 테스트 프로세스의 작업, 절차, 산출물을 CMR에 따라 핵심 테스트 프로세스 컴포넌트로 정적 모형, 동적 모형으로 구성되는 컴포넌트의 블랙박스 영역과 인터페이스로 구분하여 다음과 같이 개발한다.

정적 모형

핵심 테스트 프로세스는 테스트 계획, 테스트 설계, 테스트 수행과 테스트 검토의 4개의 작업들로 구성한다. 핵심 테스트 프로세스의 정적 모형은 패키지와 클래스도로 표현한다. 즉, 그림 6의 4개의 작업들은 CMR에 따라, TestPlan, TestDesign, TestExecution과 Test Review의 4개의 패키지들로 구성하는 패키지도로 변환된다. 각 작업에 대한 패키지의 정적 모형은 클래스도로 세분화되어 표현된다.

예를 들어, 테스트 수행작업에 해당하는 Test Execution 패키지에 대한 클래스도는 그림 7의 정적 도형 부분과 같다. (그 외 나머지 패키지에 대한 클래스도들은[10] 참조). 테스트 수행 작업의 세부 절차들은 TestExecute 클래스의 메소드들로 변환되고, 테스트 수행 작업의 테스트 수행 보고서는 ExecutionReport 클래스로 변환된다.

동적 모형

핵심 테스트 프로세스의 동적 모형이란 작업내의 절차와 산출물의 관계를 나타내며, UML의 순서도로 표현한다. 예를 들어 그림 7의 작업내의 절차들 사이의 상호작용은 그림 7의 동적 모형에서처럼 순서도로 표현된다.

인터페이스

핵심 테스트 프로세스의 인터페이스란 테일러링을 위한 영역이며, 인터페이스 클래스로 표현한다. 예를 들어, 테스트 수행 작업의 경우, 그림 7에서처럼 테스트 레벨을 반영하는 테일러링을 위해 TestExecute 클래스를 인터페이스 클래스로 표현한다.

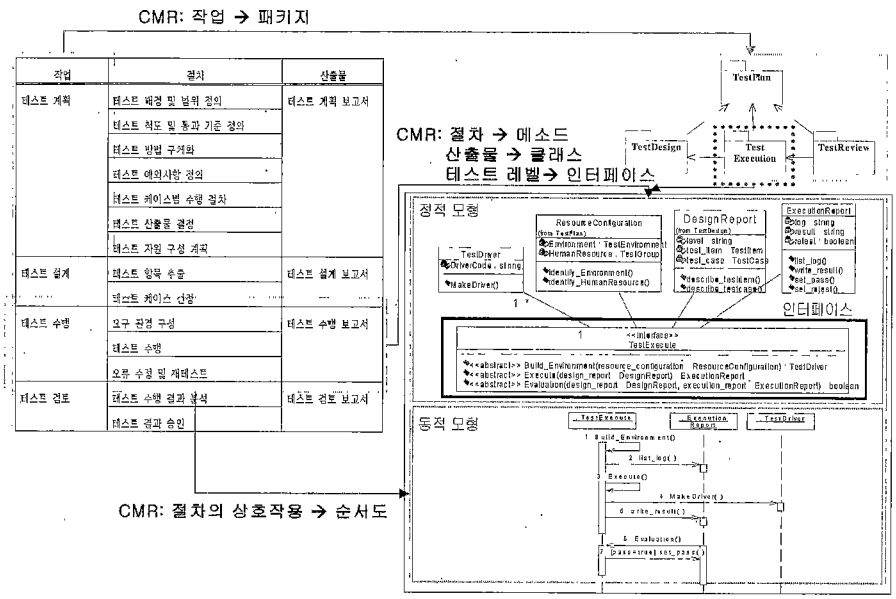

그림 7 핵심 테스트 프로세스 컴포넌트 구현(예: 테스트 수행 작업)

(2) 방법론 플러그인 개발

다양한 개발 방법론의 핵심 프로세스 내용을 지원하는 플러그인을 개발하여 테스트 프로세스 메타 모델 저장소에 저장한다. 각 방법론에 대한 특성 추출 단계와 방법론 플러그인 구현 단계의 2 단계로 진행한다. 본절에서는 Objectory의 개발 방법론에서의 테스트 프로세스를 대상으로 방법론 플러그인의 개발을 예로 기술하였다.

1) 방법론 특성 추출

그림 8은 Objectory 테스트 프로세스의 테스트 레벨, 작업, 절차, 산출물과 관련하여 간략히 기술한 것이다. 이때 굵은 테두리 영역은 Objectory의 방법론의 특성에 해당하는 부분이며, 이 부분은 Objectory를 위한 플러그인으로 개발된다. 나머지 부분은 앞에 기술한 핵심 테스트 프로세스에 해당한다.

2) 방법론 플러그인 구현

방법론 플러그인을 정적 모형, 동적 모형으로 구성되는 컴포넌트의 블랙박스 영역으로 구분하여 다음과 같이 개발한다. 그림 8의 Objectory 방법론의 특성 중 테스트 구현 작업은 Plugin_Objectory_Imp 플러그인으로 개발한 것이며, Objectory의 테스트 수행 작업에 대한 통합과 시스템 테스트 레벨을 반영하는 Plugin_Objectory_Exe 플러그인을 개발한다.

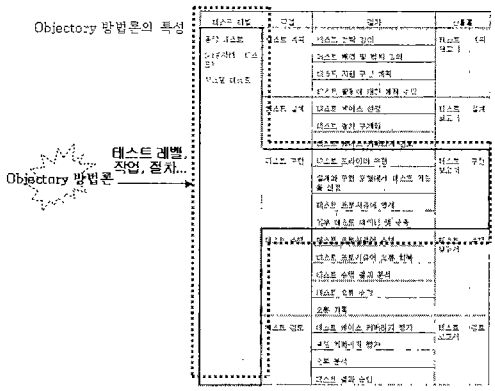

그림 8 Objectory 방법론 특성 추출

정적 모형

방법론에 따른 테스트 구현 작업은 CMR에 따라 그림 9의 정적 모형처럼 TestImplement 패키지와 클래스도로 표현된다. 테스트 구현 작업의 세부 절차들은 TestImplement 클래스의 메소드들로 표현되고, 테스트 구현 보고서는 TestSurvey 클래스로 표현된다. 그림 8의 Objectory 테스트 프로세스의 "테스트 구현"작업은 그림 9의 Plugin_Objectory_Imp의 플러그인으로 개발

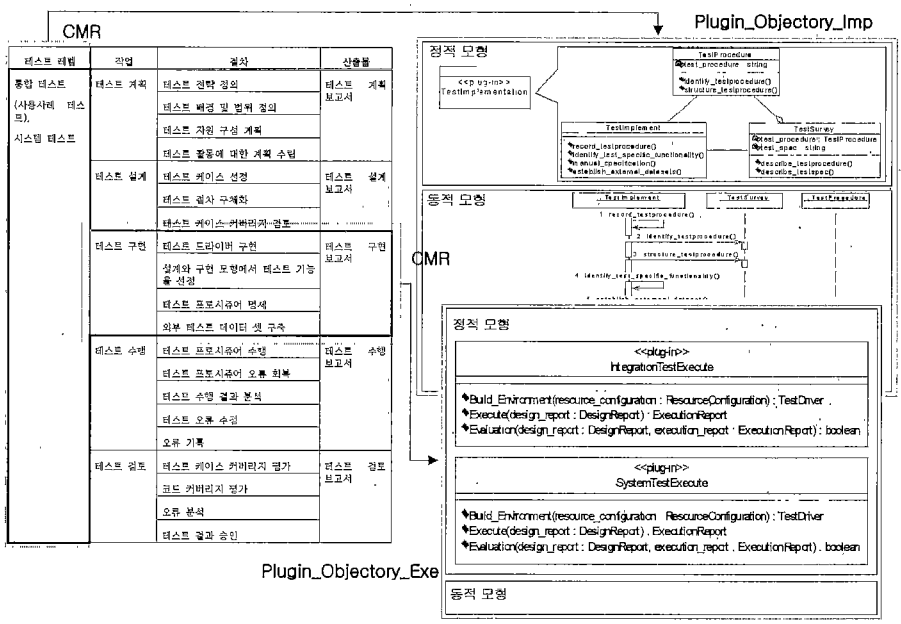

그림 9 Objectory 방법론 플러그인 구현

(예: 테스트 구현 작업(Plugin_Objectory_Imp)과 통합/시스템 테스트 레벨(Plugin_Objectory_Exe))

되었다.

Objectory 테스트 프로세스는 통합 테스트와 시스템 테스트의 2 단계의 테스트 레벨을 갖는다[11]. 핵심 테스트 프로세스 컴포넌트에는 작업별로 테스트 레벨을 반영하기 위해 인터페이스 클래스를 표현하였는데, Objectory 프로세스를 위한 통합 테스트와 시스템 테스트는 각각 이들 인터페이스 클래스를 상속받는 클래스로 구현된다. 그림 9의 정적 모형은 각각 테스트 수행 작업의 통합 테스트와 시스템 테스트를 위한 클래스도이며, Plugin_Objectory_Exe로 개발되었다. (그 외 나머지 작업의 테스트 레벨을 반영하는 클래스들은 [12] 참조)

동적 모형

Plugin_Objectory_Imp의 경우, 테스트 구현 작업내의 절차들과 산출물 사이의 상호작용은 그림 9의 동적 모형처럼 순서도로 표현된다.

Plugin_Objectory_Exe의 경우, 통합 테스트와 시스템 테스트를 위한 플러그인들은 핵심 테스트 프로세스 컴포넌트의 테스트 레벨을 반영하기 위한 인터페이스 클래스를 상속받기 때문에 추가의 순서도 개발은 필요하지 않다.

(3) 도메인 플러그인 개발

개발 도메인에 속하는 여러 응용 프로그램 사이의 공통점과 차이점을 분류하여 도메인 플러그인을 개발한다. 즉, 공통점은 블랙박스 영역으로써, 차이점은 인터페이스로 구현한다. 도메인 플러그인 개발은 도메인 특성 추출 단계와 도메인 플러그인 구현 단계의 2 단계로 진행한다.

본 절에서는 전자상거래 도메인을 대상으로 도메인 플러그인 개발 사례를 기술하겠다. 일반적으로 전자상거래 소프트웨어는 전자상거래 인프라를 중심으로 개발되어진다[13,14]. 전자상거래 인프라는 네트워크 인프라, 멀티미디어 및 네트워크 출판 인프라, 정보 분배 인프라, 공통 업무 서비스 인프라의 4가지 주요 인프라와 이들 사이의 인터페이싱을 위한 인프라로 구성한다. 본논문에서는 전자상거래 이들 4가지 인프라와 인프라 상호간의 인터페이스를 테스트하는 작업을 중심으로 기술하겠다.

1) 도메인 특성 추출

방법론에서 테스트 작업, 절차, 산출물과 같은 프로세스 요소들은 고려되어지므로 도메인을 위한 특성은 테스트 지침과 테스트 기법을 기준으로 추출한다. 따라서, 전자상거래의 도메인 특성은 네트워크 인프라, 멀티미디어 및 네트워크 출판 인프라, 정보 분배 인프라, 공통 업무 서비스 인프라의 4가지 테스트 기법과 인프라 인터페이싱 기법의 총 5가지 테스트 기법이 추출될 수 있다. 이들은 각기 도메인 플러그인으로 개발된다.

2) 도메인 플러그인 구현

정적 모형

테스트 작업의 구체적인 테스트 기법은 플러그인으로 개발되어 핵심 테스트 프로세스 컴포넌트의 인터페이스를 통하여 테일러링된다. 이들 기법은 클래스로 구현되어 핵심 테스트 프로세스 컴포넌트의 인터페이스 클래스를 통하여 상속받아 구체화된다. 그림 10은 테스트 설계 작업의 테스트 항목 추출 절차와 테스트 케이스 선정 절차를 위한 Plugin_E-Commerce_Infratest이다. (그 외 나머지 테스트 수행 작업의 인프라 테스트 기법을 반영하는 클래스들은 [15] 참조)

동적 모형

인프라 테스트 설계 기법을 위한 플러그인은 핵심 테스트 프로세스 컴포넌트의 인터페이스 클래스를 통하여 상속받기 때문에 추가의 순서도 개발은 필요하지 않다.

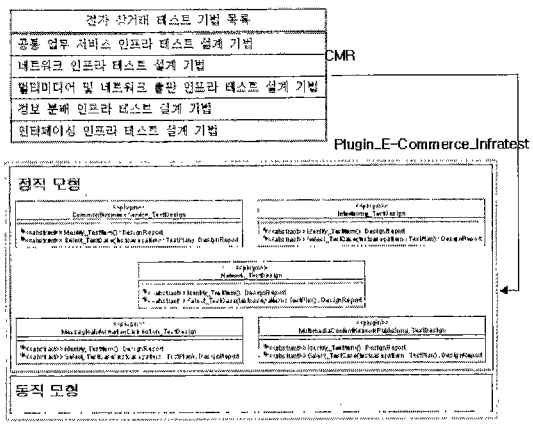

그림 10 전자상거래 도메인 플러그인 구현
(예: 인프라 테스트 설계 기법)

3.2 프로세스 테일러링

테스트 프로세스 메타 모델의 테일러링은 2장의 테스트 프로세스 생성 자동화 알고리즘에서와 같이 핵심 테스트 프로세스 컴포넌트와 플러그인의 테일러링으로 이루어진다. 이 테일러링 작업은 그림 5와 같이 AutoTP 초기화 모듈, 방법론/도메인 선정 모듈, 테일러링 모듈, 프로젝트 관리 모듈과 테스트 프로세스 저장 모듈의 5가지 주요 모듈로써 구현되었다. 각 모듈간의 자료 교환과 테일러링 작업을 위한 내부 포맷으로는 XML DOM 트리가 사용되었다. 즉, 초기화 모듈과 방법론/도메인

선정 모듈을 통하여 결정된 방법론과 도메인 플러그인의 XML 파일들을 읽어들여 각각의 XML DOM 트리로 파싱한 후, 각 플러그인들의 테일러링 패턴에 따른 테일러링 기법으로 XML DOM 트리를 조작하여 테일러링된 하나의 XML DOM 트리를 생성한다. AutoTP의 주요 모듈에 대한 간단한 설명은 아래와 같다.

초기화 모듈: AutoTP의 초기화 모듈은 핵심 테스트 프로세스 컴포넌트와 방법론과 도메인의 플러그인들의 테스트 프로세스 메타 모델에 대한 정보를 읽고, 핵심 테스트 프로세스 컴포넌트에 대한 XML DOM 트리를 생성한 후, 이것을 데이타베이스에 연결한다.

방법론/도메인 선정 모듈: AutoTP의 사용자는 자신의 프로젝트에 적합한 방법론과 도메인을 AutoTP의 메뉴를 통하여 입력한다. 방법론/도메인 선정 모듈은 사용자에 의해 선택된 방법론과 도메인에 대한 플러그인을 테스트 프로세스 메타 모델 저장소에서 추출하여 해당 XML DOM 트리를 생성한다.

테일러링 모듈: 초기화 모듈과 방법론/도메인 선정 모듈로부터 파싱된 핵심 테스트 프로세스 컴포넌트와 방법론과 도메인 플러그인의 XML DOM 트리들을 각각의 테일러링 패턴에 따른 테일러링 기법으로 처리하여 테일러링된 하나의 XML DOM 트리를 생성한다.

프로젝트 관리 모듈: 방법론과 도메인이 테일러링된 테스트 프로세스에 사용자로부터 입력받은 프로젝트가 갖는 고유 특성을 반영한다. 즉, 사용자로부터 프로젝트에 관련된 정보를 입력받아 데이타베이스에 저장한다. 프로젝트 관리 모듈을 통하여 AutoTP의 최종 결과물인 프로젝트에 특화된 테스트 프로세스를 생성할 수 있게 된다.

테스트 프로세스 저장 모듈 : AutoTP의 최종 결과물인 프로젝트 요구사항을 반영한 테스트 프로세스를 HTML 문서로 저장한다. 즉, 테일러링 모듈을 통하여 테일러링된 하나의 XML DOM 트리를 HTML 문서 포맷으로 변환하여 저장한다.

3.3 AutoTP의 실행 사례

본 절에서는 AutoTP가 자율적으로 테스트 프로세스를 테일러링하여 프로젝트에 적합한 테스트 프로세스의 생성 시나리오를 그림 11로 나타내었다. 즉, AutoTP 사용자는 1) AutoTP를 수행시키고, 2) 자신의 프로젝트에 적합한 방법론과 도메인을 입력하고, 3) 테일러링을 지시하며, 4) 프로젝트 관련 정보를 입력하고, 5) 최종 테일러링 결과를 파일에 저장하는 5단계를 거쳐서 프로젝트에 맞춤된 테스트 프로세스를 획득할 수 있다.

(1) AutoTP 시작 (Invoke "AutoTP")

사용자가 AutoTP를 실행시키면, AutoTP는 내부적인 초기화 작업을 수행한다. 특히, 이 단계에서 AutoTP는 테스트 프로세스 메타 모델을 구성하는 컴포넌트들 중 핵심 테스트 프로세스 컴포넌트를 읽어들여서 XML DOM 트리로 파싱한다.

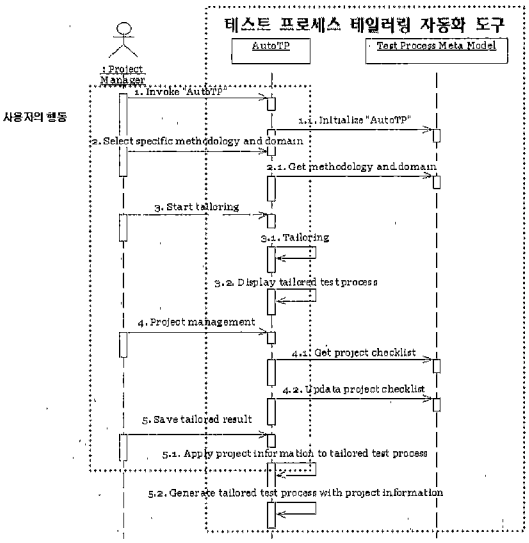

그림 11 AutoTP 사용 시나리오

(2) 방법론과 도메인 입력(Select specific methodology and domain)

사용자는 자신의 프로젝트의 특성에 적합한 방법론과 도메인을 그림 12의 도메인 선정 다이알로그 박스에서 입력한다. 본 사례에서는 Objectory 방법론(Rational

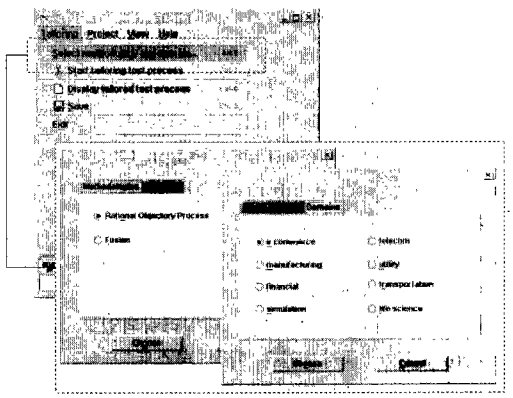

그림 12 방법론, 도메인 선정 작업

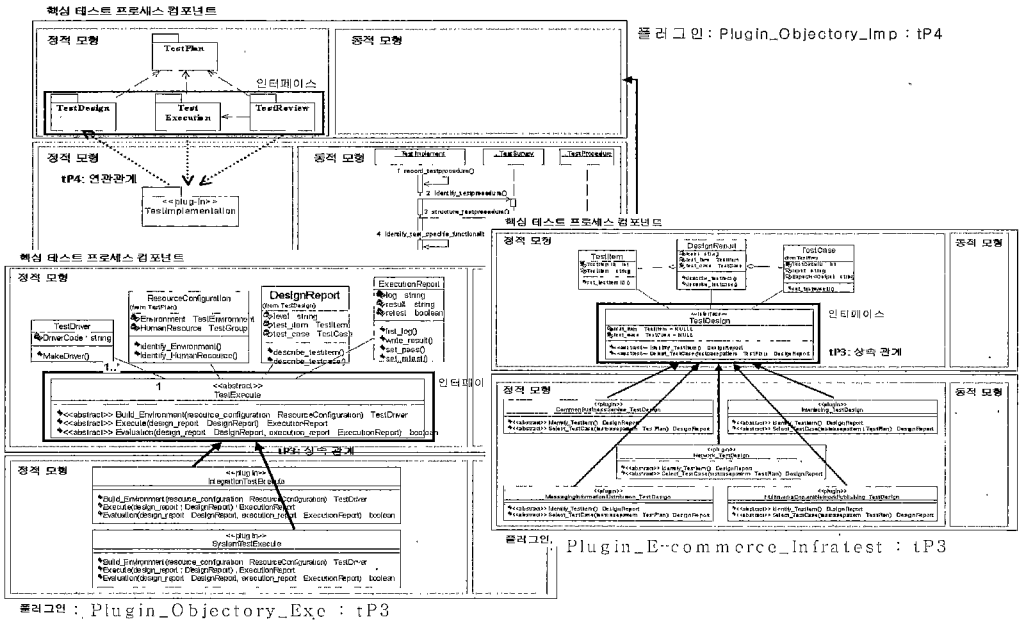

그림 13 테스트 프로세스 메타 모델의 테일러링

Objectory Process)과 전자상거래 도메인(e-commerce)를 선택한다. AutoTP는 테스트 프로세스 메타 모델을 구성하는 컴포넌트들 중에서 사용자가 선택한 Objectory 방법론과 전자 상거래 도메인에 해당하는 플러그인들을 읽어들여서 각각 XML DOM 트리로 파싱한다.

(3) 테일러링 (Start tailoring)

사용자는 테스트 프로세스 테일러링을 위하여 그림 14의 "Start tailoring test process" 메뉴를 선택한다. 그러면 AutoTP는 핵심 테스트 프로세스 컴포넌트와 Objectory 방법론과 전자 상거래 도메인 플러그인에 대한 XML DOM 트리들을 각각의 테일러링 패턴에 따라 테일러링을 수행한다.

그림 13은 Objectory 방법론과 전자 상거래 도메인의 테일러링 과정 가운데 앞 3.1절의 예로 든 3 개의 플러그인 Plugin_Objectory_Imp, Plugin_Objectory_Exe, Plugin_E-Commerce_Infratest 가 테일러링된 결과를 보여준다. (전체 결과는 [16]를 참고) 즉, Plugin_Objectory_Imp 플러그인은 핵심 테스트 프로세스 컴포넌트의 TestDesing, TestExecution, TestReview 패키지에 tP4에 의해 연관관계로 테일러링되었고, Plugin_Objectory_Exe 플러그인은 핵심 테스트 프로세스 컴포넌트의 TestExecute 인터페이스 클래스에 tP3에 의해 상속관계로 테일러링되었다. 또한, Plugin_E-Commerce_Infratest 플러그인은 핵심 테스트 프로세스 컴포넌트의 TestDesign 인터페이스 클래스와 tP3에 의해 상속관계로 테일러링되었다.

위와 같은 테일러링 단계를 거쳐, 그림 14와 같은 결과 화면이 나타난다. 결과 화면은 사용자가 미리 선정한 방법론과 도메인에 대해 테일러링된 테스트 프로세스를 중간 결과 포맷인 XML의 DOM 트리로 보여주며, 아래

그림 14 테일러링 작업

단의 메시지 창을 통해서는 테일러링 과정을 보여준다. 테일러링된 테스트 프로세스가 XML 포맷의 중간 결과로 저장되는 것은 사용자에게 전달되는 최종 결과물인 프로젝트에 특화된 테스트 프로세스가 다양한 형태의 문서 포맷으로 지원 가능하게 하기 위함이다.

AutoTP는 방법론과 도메인의 두 단계로 테일러링이 진행되었으며 그림 15의 메시지 창을 보면, 이 시나리오에 대한 결과는 Objectory 방법론과 전자 상거래 도메인을 대상으로 테일러링 되었음을 알 수 있다.

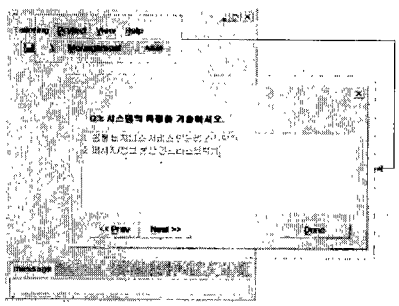

그림 15 프로젝트 관리 작업

(4) 프로젝트 관련 정보 입력 (Project management)

사용자는 특정 프로젝트의 정보를 포함하는 보다 정확한 테스트 프로세스를 생성하기 위하여 그림 15처럼 프로젝트 관리 메뉴를 선택하고 프로젝트 관리 다이알로그 박스와의 상호작용을 통하여 프로젝트 정보를 전달한다. 프로젝트 관리 다이알로그 박스는 데이타베이스에 저장되어있는 핵심 테스트 프로세스 컴포넌트와 방법론, 도메인 플러그인에 관련된 체크리스트에 대해 해당 프로젝트의 정보를 입력할 수 있도록 하며, 이렇게 입력된 프로젝트 정보는 최종 테스트 프로세스 생성에 적용된다.

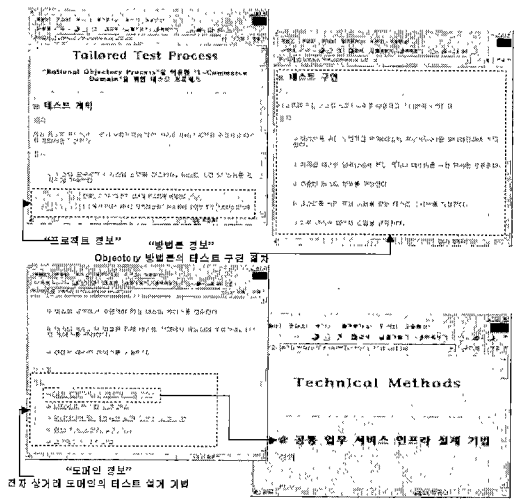

그림 16 최종 테일러링된 테스트 프로세스

(5) 테스트 프로세스 저장 (Save tailored result)

사용자를 위한 최종 테스트 프로세스는 웹 기반 도큐먼트로 작성되는데 사용자가 선택한 도큐먼트 이름으로 최종 테일러링된 HTML 문서의 테스트 프로세스가 생성된다. 최종 테일러링된 테스트 프로세스는 그림 16과 같다.

4. AutoTP의 효과 및 결론

소프트웨어 개발을 위한 표준은 매우 추상적이어서 각 개발 환경에 따라 테일러링하여 사용하도록 규정되어 있다. 본 논문에서는 이러한 테일러링 작업을 CBD의 맞춤 개념을 이용하여 자동화함으로써, 개발 특성에 테일러링된 프로세스를 자동 생성할 수 있도록 하였다. 특히 테스트 프로세스의 자동생성을 위하여 방법론, 도메인과 프로젝트로 세분화되는 테일러링 단계를 제안하였고, 이에 따라 체계적으로 테일러링 작업이 수행될 수 있는 테스트 프로세스 생성 자동화 알고리즘을 정의하였고 이를 AutoTP로 구현하였다.

테스트 프로세스 생성 자동화 알고리즘은 테스트 프로세스 메타모델의 개발과 테일러링 단계로 이루어진다. 본 논문에서는 표준에서 추출한 핵심 테스트 프로세스와 방법론과 개발 도메인을 위한 플러그인 들로 구성하는 UML 형태의 테스트 프로세스 메타 모델을 XML로 변환하여 개발하였다. 따라서 표준으로부터 추출한 핵심 테스트 프로세스로부터 사용자가 원하는 방법론과 도메인으로 테일러링된 테스트 프로세스를 HTML 문서를 생성할 수 있었다.

핵심 테스트 프로세스 컴포넌트와 방법론, 도메인을 위한 플러그인들은 AutoTP의 테스트 프로세스 메타 모델 저장소에 저장되어 재 사용될 것이다. 즉, 어떠한 프로젝트라도 손쉽게 테스트 프로세스 메타 모델 저장소 내에 저장된 플러그인들을 재 사용하여 특정 테스트 프로세스를 생성할 수 있으며, 프로젝트에 따라 방법론을 변경하고 싶은 경우에도 단순히 AutoTP의 메뉴를 통하여 원하는 방법론을 변경하여 선택함으로 손쉽게 변경할 수 있다.

본 AutoTP를 통하여 표준을 준수하면서도 현장에서 바로 사용할 수 있는 수준의 테스트 프로세스 테일러링

작업을 보다 체계적으로 자동화할 수 있었다. AutoTP 사용자는 프로세스 표준이나 방법론과 도메인을 분석하는 등 작업 없이도 AutoTP의 메뉴를 통하여 원하는 방법론을 선택하고 자신의 프로젝트가 속하는 도메인을 선정하는 작업만으로 시간과 비용을 절약하면서 테스트 프로세스의 테일러링 작업을 완성할 수 있다.

본 논문에서는 소프트웨어 개발 프로세스 가운데, 테스트 프로세스로 제한하여 테스트 프로세스의 테일러링 기법을 제안하였다. 그러나 테일러링 기법의 기본 아이디어는 전체 개발 프로세스를 위한 테일러링에 무리 없이 적용될 수 있다. 그러나 실제로 다양한 개발 환경에서 이용하기 위해서는 각 개발 특성에 따른, 전자 상거래, 통신, 제조, 금융, 시뮬레이션, 생명과학…등, 다양한 플러그인들을 개발하는 일이 필요하다.

참 고 문 헌

[1] ISO/IEC JTC1/SC7 N1894, TR15271: Information Technology-Guide for ISO/IEC12207, 1998.

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

[3] Faye C.Budlong, Paul A.Szulewski, Ralph J.Ganska, "Process Tailoring for Software Project Plans," The Process Management Technologies Team The Software Technology Support Center(STSC), January 1996.

[4] Mark P. Ginsberg, Lauren H. Quinn, "Process Tailoring and the Sortware Capability Maturity Model," CMU/SEI-94-TR-024 ESC-TR-94-024, November 1995.

[5] Jooyoung Seo, Byoungju Choi, "Tailoring Test Process by using the Component Based Development Paradigm and XML Technology," 7th Proceeding of Asia-Pacific Software

[6] Martin Fowler and Kendall Scott, UML Distilled : Applying the Standard Object Modeling Language, Addison-Wesley, Aug. 1997.

[7] Wolfgang Pree, Design Patterns for Object-Oriented Software Development, Addison-Wesley, 1995.

[8] Unisys/Rational XML Interchange Package v.4.0.1, ftp://ftp.rational.com/public/rose/rose_extras/RoseXmi4.0.1.zip

[9] MIL-STD-498, Software Development and Documentation.

[10] PART 1: Core-Test Process Component, http://selab.ewha.ac.kr/apsec2000.html

[11] Rational Objectory Process: Process Manual 4.1

[12] PART 2: Objectory Plug-in, http://selab.ewha.ac.kr/apsec2000.html

[13] Efrain Turban, Jae Lee, Jae Kyu Lee, David King, H,Michael Chung, "Electronic Commerce : A Managerial Perspective," Prentice Hall, 1999. Engineering Conference (APSEC) 2000 in Singapore, pp.356-363, Dec. 2000.

[14] Ravi Kalakota, "ELECTRONIC COMMERCE A Manager' s Guide," Addison Wesley, 1997.

[15] PART 3: E-Commerce Plug-in, http://selab.ewha.ac.kr/apsec2000.html

[16] PART 4: Tailoring, http://selab.ewha.ac.kr/apsec2000.html

서 주 영

1993년 2월 이화여자대학교 전자계산학과 학사. 1998년 3월 ~ 현재 이화여자대학교 대학원 컴퓨터학과 석사과정. 관심분야는 소프트웨어 공학, 객체지향 소프트웨어 테스팅, 분산 컴포넌트 테스팅

최 병 주

1979년 ~ 1983년 이화여대 수학과 학사. 1986년 ~ 1988년 Purdue Univ. 전산석사. 1987년 ~ 1990년 Purdue Univ. 전산학(소프트웨어공학 전공) 박사. 1991년 ~ 1992년 삼성종합기술원 선임연구원. 1992년 ~ 1995년 용인대학교 전자계산학과 전임강사. 1995년 ~ 현재 이화여자대학교 컴퓨터학과 조교수. 관심분야는 소프트웨어공학, 소프트웨어 테스트, 소프트웨어 품질 측정, 테스트 케이스 적정성 측정 알고리즘