RUP기반 컴포넌트 품질 평가에 관한 연구

오 기 성[†]·류 성 열[††]

요 약

컴포넌트 기반 소프트웨어 개발에서 컴포넌트 각각의 품질은 전체 소프트웨어의 품질을 결정하는 중요한 역할을 하므로 컴포넌트를 체계적으로 테스팅할 수 있는 전략이 필요하다. 일반적으로 컴포넌트를 테스팅하는 관점은 크게 컴포넌트 생산자 입장과 구매자 입장으로 구분할 수 있다. 본 논문에서는 컴포넌트 구매자 입장에서도 생산자의 산출물을 테스팅 근거 자료로 활용할 수 있도록 테스팅 도메인을 컴포넌트 생산자 입장에서 전개한다. 컴포넌트의 품질을 평가하기 위해 반복(Iteration) 테스팅 지침(Testing Guideline)에 중점을 두고 있는 현재의 RUP 테스팅 프로세스를 개선하여 컴포넌트 단위 테스팅에 촛점을 맞춘 실질적인 프로세스를 구체적으로 제안하고 EJB 환경하에서 컴포넌트의 품질을 평가하는 사례 연구를 적용해 본다. 이를 통하여 본 논문에서 제시한 RUP기반 5단계 테스팅 프로세스가 컴포넌트의 품질평가를 위해 적용 가능한 것임을 보인다.

A Study on RUP based Component Quality Evaluation

Kie Sung Oh[†] · Sung Yul Rhew[††]

ABSTRACT

In Component-Based Software Development, the quality of indivisual component is play an important role in quality decision of the whole software. So we need the practical strategy for component testing. In general, component testing can divide focus into producer position and consumer position. In this paper, because the consumer position uses output of the producer position, testing domain is deployed in the producer position. We propose RUP based five step testing processes for component quality evaluation and implements a case study of EJB environment for appling our testing process. This paper shows that proposed five step processes are applicable to component quality evaluation.

키워드 : 컴포넌트(Component), 품질평가(Quality Evaluation), 테스팅(Testing), RUP(Rational Unified Process)

1. 서 론

최근 소프트웨어 산업은 컴포넌트를 조립하여 소프트웨어를 생산하는 방식으로 패러다임이 변해감에 따라 컴포넌트 산업이 새로운 산업으로 등장하고 있다. 세계 컴포넌트 산업의 평균 성장률은 49%로 예상되며 같은 기간 소프트웨어 산업의 평균 성장률 14.5% 보다도 높다[8]. 이러한 시대적 요구에 부흥하기 위해 많은 기관에서 컴포넌트를 개발하는 방법론이 연구되고 있으나 개발되어진 컴포넌트의 품질을 객관적으로 평가하는 테스팅 전략은 미약한 상황이다. 일반적으로 컴포넌트를 테스팅하는 관점은 크게 컴포넌트 생산자 입장과 구매자 입장으로 구분할 수 있으며 컴포넌트 생산자 입장에서 테스팅된 산출물은 구매자 입장에서도 참조, 활용하고 있다[3, 9].

연구범위는 컴포넌트의 품질을 평가하기 위해 여러 품질평가 요소 중에서 기능상에 결함이 없는 단위 컴포넌트 테스팅에 초점을 맞추어 컴포넌트의 정확성, 신뢰성, 무결성, 유용성, 재사용성이 향상되도록 고려하였다.

본 논문에서는 생산자 입장의 개발 컴포넌트들을 좀 더 효율적으로 테스팅할 수 있는 RUP기반 5단계 테스팅 프로세스를 구체적으로 제안하고 사례연구로 EJB 환경하에서 쇼핑관련 컴포넌트 빈들을 생성하고 테스팅 프로세스를 적용해 보았다. 이를 통하여 본 논문에서 제시한 RUP기반 5단계 테스팅 프로세스가 컴포넌트의 품질평가를 위해 적용 가능한 것임을 보였다.

RUP 프로세스는 점진적 소프트웨어 개발 프로세스에 기반을 두고 있으며 각 단계별 세부 사항은 구체적으로 명시되어 있지 않다. 정확한 테스트 케이스를 도출하기 위해 우선 Use Case Diagram과 Class Diagram, Component Diagram으로부터 XML 메타 데이타를 추출하고, 컴포넌트를 구성하는 메소드들의 입력 도메인을 동치영역과 경계값으로 구분하여 각 도메인 입력 데이타들의 조합을 테스트 케이스로 추출한다. 추출된 테스트 케이스는 테스트 드라이버에 의해 수행된 후 결과값을 가지고 비율검증을 통해 통계

† 정 회 원 : 동원대학 컴퓨터정보과 교수
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2001년 8월 20일, 심사완료 : 2001년 10월 19일

적으로 컴포넌트의 품질을 평가한다[2, 4].

2. 이론적 배경

2.1 전통적 소프트웨어 테스팅 방법론

기존의 테스팅 방법론은 크게 두가지 전략들로 접근한다. 그 하나는 프로그램의 원시코드를 분석하지 않고 명세에 크게 의존하여 프로그램을 기능 중심으로 테스팅하는 명세 기반 테스팅이고 또 다른 하나는 원시코드의 내부 구조를 분석하여 테스팅하는 프로그램 기반 테스팅이다.

2.2 객체지향 소프트웨어 테스팅 방법론

객체지향 구조의 특성으로 인해 기존의 테스팅 방법을 달리할 필요가 발생되었다. 상속성, 다형성, 캡슐화와 같은 객체지향이 가지는 장점이 테스팅 분야에서는 여러 가지 요인을 고려해야만 하는 사항을 발생시킨다. 초기의 테스팅 방법론들은 객체 개념으로 인해 블랙박스 테스팅을 주로 사용하였으나 객체 내부의 소스코드에 대한 화이트박스 테스팅 방법도 적용되고 있다. 객체지향 테스팅에 관한 연구는 다음과 같다[5].

2.2.1 Weyuker가 제안한 적합성 테스팅 기준

Weyker가 제시한 공리들은 테스팅 수행자들이 직관적이고 경험적인 관점에서 공감하는 것들을 형식화한 것이다.

예) 공리1 : 모든 프로그램에는 적합한 테스트 집합이 있다.
공리2 : 프로그램 P가 테스트 집합 T에 의해 적합하게 테스트되었을 때 T는 철저한 테스트 집합은 아니다.
공리3 : 만약 테스트 집합 T가 프로그램 P에 적합하고 $T \subset T'$이면 T'는 P에 적합하다.

2.2.2 Perry와 Kaiser의 테스팅 방법

Weyuker의 적합성 공리를 클래스에 적용하여 객체지향 프로그램의 테스팅을 제안하였고 메소드를 상속받을 시 테스팅 문제, 메소드의 재정의시 테스팅 문제등이 테스팅 대상이다.

2.2.3 Harrold와 McGregor의 테스팅 방법

상속성 구조를 가지는 특성에 따라 기본 클래스의 테스팅 정보를 재 사용하여 상속된 클래스의 테스팅을 수행한다.

2.2.4 Smith의 FOOT

객체지향프로그램 구조에서 테스팅을 실시해야할 여러 가지 전략을 소개하고 FOOT라는 객체지향프로그램 테스팅 구조를 소개하였다.

2.3 컴포넌트 소프트웨어 테스팅 방법론

컴포넌트의 시험이 기존의 소프트웨어 시험과 차이를 보이기 때문에 컴포넌트 시험에 적용되는 시험 기준 또한 차이를 보이게 된다. Resenblum은 컴포넌트의 정형화된 모델을 정의하여 컴포넌트 시험의 타당한 기준을 정의하려고 하였고 Ghosh는 인터페이스의 뮤테이션을 통해 시험 기준을 마련하고자 하였다[6].

2.4 RUP(Rational Unified Process)프로세스

RUP에서는 반복적인 개발 방법을 제안한다. 소프트웨어의 개발은 여러 번의 반복(Iteration)을 거치며 각각의 반복은 요구사항 분석, 설계, 구현 및 테스트, 평가 과정을 포함하고 있어 자체로서도 하나의 개발주기를 이룬다. 이러한 반복적인 개발 방법에서는 반복마다 실행 가능한 릴리즈가 산출되고 이는 반복이 거듭될수록 향상되어 결국 최종 시스템으로 발전된다.

전통적인 개발 프로세스와 비교했을 때 장점은 다음과 같다. 초기의 위험요소를 줄일 수 있으며 변경에 대한 관리가 용이하고 보다 높은 수준의 재사용이 가능하다. 또한 프로세스가 진행됨에 따라 프로젝트 팀원의 기술이 향상되며 높은 품질의 제품을 얻을 수 있다.

본 논문에서는 컴포넌트의 품질을 테스팅하기 위해 RUP 기반 5단계 프로세스를 구체화시키고 사례연구를 통해 통계적으로 검증해 보았다. 특히 논문에서 제안한 연구 도메인은 6개월 미만의 프로젝트로 반복(Iteration) 2회 이하로 운영하는 소규모 프로젝트 테스팅을 대상으로 하며 컴포넌트 단위 테스팅에 촛점을 맞추고 있다.

3. 구체적 RUP 기반 5단계 테스팅 프로세스

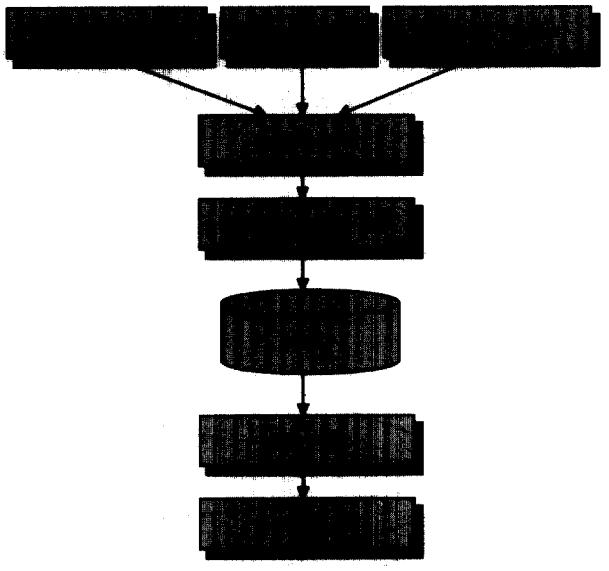
RUP기반 테스팅 프로세스는 일반적으로 Elaboration phase의 첫 번째 반복 단계부터 행해지며 Construction phase, Transition phase로 진행됨에 따라 아래 5단계 테스팅 프로세스 과정을 반복적으로 수행한다[1].

1 단계 : Test Plan

반복(Iteration)단계에 맞는 테스팅 목적, 범위 및 테스팅 자원, 툴, 요구사항에 따른 테스팅 수행 전략등을 정의하고 Planning Sheet에 요약 정리한다. 참고로 본 논문의 테스팅 범위는 컴포넌트 단위 테스팅으로 제한한다.

2 단계 : Test Design

1단계의 Test Plan을 기반으로 테스트 드라이버, 테스트 케이스와 관련된 테스트 모델을 정의한다. 본 논문에서는 컴포넌트 단위 테스팅에 필요한 테스트 케이스를 생성하기 위해 우선적으로 각 반복 단계의 컴포넌트 분석, 설계 다이어그램을 XML 메타 데이터로 변형하고 그 내용을 웹 브라우져로 확인해 볼 수 있도록 XSL을 이용하여 HTML로 요약, 정리한다.



(그림 1) Test Design 단계의 전체적인 흐름도

<표 1>은 다이어그램 산출물을 XML 메타데이타로 변형하기 위한 DTD이다.

〈표 1〉 다이어그램 산출물을 정의하기 위한 DTD

```
<!DOCTYPE Component [
<!ELEMENT Component (Method+)>
<!ELEMENT Method (Description,
                  Component-name,
                  method-name,
                  method-params)
>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Component-name (#PCDATA)>
<!ELEMENT method-name (#PCDATA)>
<!ELEMENT method-params (method-param+)>
<!ELEMENT method-param (#PCDATA)>
]>
```

3 단계 : Test Implement

2단계에서 정리된 메소드별 XML 메타 데이타로부터 테스트 수행에 필요한 실질적인 테스트 케이스를 추출한다. 컴포넌트의 기능성은 이를 구성하는 메소드들을 통해 표현되므로 메소드 입력 도메인을 동치영역과 경계값으로 구분하고 각 도메인 입력 데이타들의 조합을 테스트 케이스로 추출한다. 동치테스트는 테스트 사례의 개수를 최소화하는 블랙박스 테스팅 방법으로 크게 2단계로 구현된다.

[1 단계] 입력조건에 따라 유효동치영역과 무효동치영역으로 나눈다. 설정기준은 다음과 같다.

① 입력조건이 범위를 나타날 때
최소 1개의 유효동치영역과 2개의 무효동치영역으로 정의

② 입력조건이 특정한 값일 때
최소 1개의 유효동치영역과 2개의 무효동치영역으로 정의

③ 입력조건이 집합일 때
최소 1개의 유효동치영역과 1개의 무효동치영역으로 정의

④ 입력조건이 부울값일 때
최소 1개의 유효동치영역과 1개의 무효동치영역으로 정의

[2 단계] 다음 기준에 따라 테스트케이스를 정한다.

① 정의된 동치영역에 고유번호를 붙인다.

② 하나의 테스트케이스가 가능한 많은 수의 동치영역을 커버하도록 한다.

테스트케이스 선택을 위한 좀 더 구체적인 기준은 다음을 참조한다[7].

① 포함정도 (coverage) : 모든 가능 입력은 동치 영역의 하나에 속하여야 한다.

② 비결합성 (disjointness) : 같은 입력이 동일 동 치영역에 존재하면 안된다.

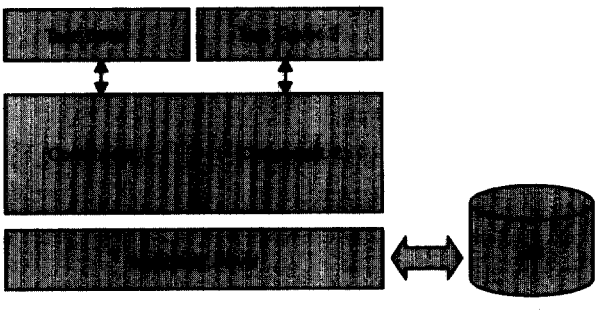
③ 표현성 (representation) : 특정 동치 영역의 멤 버가 입력으로 사용되었을 때 실행에 오류가 발생하면 같은 동치 영역의 멤버를 입력으로 사용하여 실행하였을 때 같은 오류가 발생한다.

경계값 테스트는 동치 테스트의 특정한 형태로 동치 영역에 있는 임의의 요소를 택하지 않고 경계로부터 선택한다.

본 논문에서는 2단계에서 정리된 XML 메타 데이터로부터 동치영역과 경계값에 근거한 테스트 케이스를 추출한다.

4 단계 : Test Execution

3 단계에서 추출한 테스트 케이스를 테스트 드라이버를 통해 (그림 2)와 같이 실행시키고 결과치를 얻는다. (그림 2)에서 각각의 컴포넌트들을 테스팅하기 위해 각각의 테스트 드라이버가 필요하며 컴포넌트별로 독립적인 결과가 도출된다.



(그림 2) 컴포넌트 단위 테스트를 위한 모형도

5단계 : Test Evaluation

4단계에서 정리된 결과치를 통계적으로 분석, 정리함으로써 테스트한 컴포넌트의 품질을 평가하며 분석, 정리를 위한 평가 도표 형식은 아래 <표 2>와 같다. 컴포넌트를 구성하는 메소드와 전체 컴포넌트 품질은 합격, 불합격 등급으로 평가한다. 비율검증을 위한 가설과 대립가설을 세우고 유의수준 α = 0.05를 기준으로 실시한다.

- 가설 H0 : 모집단의 만족비율이 90%이상
- 대립가설H1 : 모집단의 만족비율이 90%미만

검증은 통계적으로 분석하고 유의수준 α=0.05를 기준으로 테스트를 실시하여 컴포넌트 입력에 대한 결과로 기대값과 다른값이 나올 확률 즉, 오류를 범할 확률 P0값을 구한다. P0값에 따라 아래와 같이 합격, 불합격이 결정된다.

P0 > α 이면 H0를 기각하지 않는다. 즉, 합격
P0 <= α 이면 H0를 기각한다. 즉, 불합격

〈표 2〉 단위 테스트를 위한 평가표

입력데이타	결과값	기대값	평가항목	
			만족	불만족
파라미터 1				
파라미터 n				

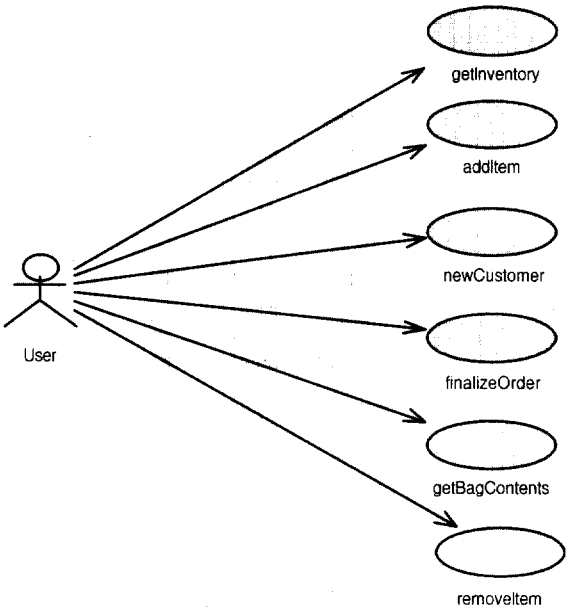
4. 사례 연구

테스팅 사례로 온라인 쇼핑을 하기 위한 Stateless Session Bean을 구현하고 평가해 볼 것이다.

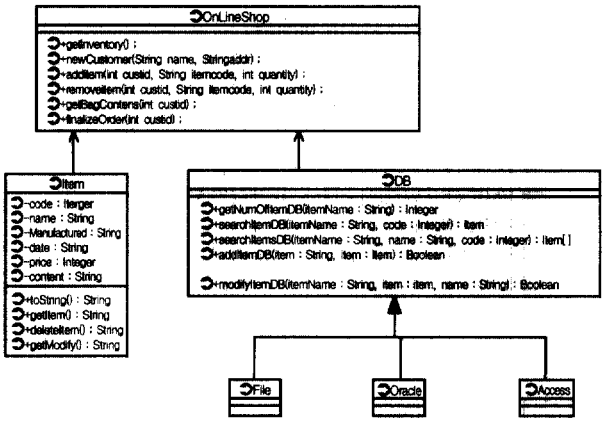
요구사항은 아래와 같으며 쇼핑한 상품을 짐수레 대신 가방에 넣을 것이다.

① 상품 목록을 가져오는 방법을 사용자에게 제공해야 한다.
② 사용자는 하나 또는 그 이상의 상품을 주문 할 수 있어야 한다.
③ 사용자는 쇼핑 가방으로부터 주문한 상품을 취소할 수 있어야 한다.
④ 상품 목록 데이터베이스와 상호작용을 할 수 있어야 한다.
⑤ 사용자는 시스템에게 쇼핑 가방의 현재 내용을 질의할 수 있어야 한다.
⑥ 사용자는 만일 상품 재고가 없으면 주문을 그 장소에서 할 수 없어야 한다.
⑦ 사용자는 데이터베이스에서 계산서 정보를 처리 할 수 있어야 한다.

본 논문에서 사례연구로 사용한 데이터베이스의 등록된 아이디는 800명이고 Item code수는 10개, Item code별 재고량은 최대 200개이다. (그림 3)과 (그림 4)는 사례연구 분석, 설계 산출물이며 <표 3>은 사례연구에서 사용한 Item code Table이다.



(그림 3) 사례연구 Use Case Diagram



(그림 4) 사례연구 Class Diagram

〈표 3〉 사례연구 Item Code Table

코드	내역	코드	내역
K1-00	본체	K1-01	모니터
K1-02	키보드	K1-03	마우스
K1-04	프린터	K1-05	스캐너
K1-06	스피커	K1-07	모뎀
K1-08	보안경	K1-09	CD-ROM

1단계 : Test Plan

컴포넌트 단위 테스팅에 기반을 둔 테스팅 프로젝트 계

획을 총괄적으로 정의한다.

2 단계 : Test Design

Use Case Diagram과 Class Diagram 산출물을 분석하여 아래와 같은 XML 메타 데이타를 생성하고 HTML로 요약, 정리한다. <표 4>는 사례연구 중 addItem() 메소드에 대한 분석 내용을 XML 메타데이타로 생성한 결과이다.

〈표 4〉 XML 메타데이타

```
< Component >
< Method >
< Description >
Method : public void addItem(int custid, String itemcode, int
        quantity) ;
< /Description >
< Component-name > OnLineShop < /Component-name >
< method-name > addItem < /method-name >
< method-params >
< method-param > int < /method-param >
< method-param > custid < /method-param >
< method-param > String < /method-param >
< method-param > itemcode </method-param >
< method-param > int < /method-param >
< method-param > quentity < /method-param >
< /method-params >
< /Method >
< /Component >
```

<표 5>는 <표 4>의 내용을 기반으로 XSL을 적용하여 HTML로 요약정리하기 위한 코드이며 (그림 5)는 웹브라우저로 확인한 결과이다.

〈표 5〉 HTML로 보기 위한 XSL

```
< ?xml version = "1.0" encoding = "EUC-KR"? >
< xsl : stylesheet
 xmlns : xsl = "http : //www.w3.org/TR/WD-xsl" >
< xsl : template match = "/" >
< HTML >
< HEAD > < /HEAD >
< BODY BGCOLOR = "YELLOW" >
< CENTER >
< H2 >XML MetaData Table < /H2 > < P > < /P >
< TABLE BORDER = "2" cellpadding = "2" >
< TR >
< TD BGCOLOR = "LIGHTBLUE" > method-name < /TD >
< TD BGCOLOR = "LIGHTBLUE" > method-param1 < /TD >
< TD BGCOLOR = "LIGHTBLUE" > method-param1 < /TD >
< TD BGCOLOR = "LIGHTBLUE" > method-param2 < /TD >
< TD BGCOLOR = "LIGHTBLUE" > method-param2 </TD >
< TD BGCOLOR = "LIGHTBLUE" > method-param3 < /TD >
< TD BGCOLOR = "LIGHTBLUE" > method-param3 </TD >
< /TR >
< DIV >
< xsl : for-each select = "Component" >
< xsl : apply-templates select = "Method" / >
< /xsl : for-each >
< /DIV >
< /TABLE > < /CENTER > < /BODY >
< /HTML >
< /xsl : template >
< xsl : template match = "Method" >
< TR >
< TD ALIGN = "CENTER" >
< xsl : value-of select = "method-name" / > < /TD >
< xsl : for-each select = "method-params" >
< xsl : apply-templates select = "method-param" / >
< /xsl : for-each >
< /TR >
< /xsl : template >
< xsl : template match = "method-param" >
< xsl : apply-templates / > < /xsl : template >
< xsl : template match = "text()" >
< TD ALIGN = "CENTER" >< xsl : value-of / > < /TD >
< /xsl : template >
< /xsl : stylesheet >
```

3 단계 : Test Implement

2 단계의 결과를 기반으로 동치영역과 경계값에 기반을 둔 테스트 케이스를 추출한다.

<표 6>의 정상문자는 유효동치영역에 포함되는 데이터이며 이탤릭문자는 무효동치영역에 포함되는 데이터이다. 입력범위에 매겨진 일련번호는 동치영역을 분류한 것이다.

- parameter : custid
  정상적인 입력범위 : ① DB에 등록된 id
  비정상적인 입력범위 : ② DB에 등록되지 않은 id,
  ③ 0, ④ 음수,
  ⑤ 문자, ⑥ 스트링

- parameter : itemcode
  정상적인 입력범위 : ① 코드테이블에 있는 항목
  비정상적인 입력범위 : ② 코드테이블에 없는 항목

- parameter : quantity
  정상적인 입력범위 : ① 재고가 있는 경우
  비정상적인 입력범위 : ② 재고가 없는 경우,
  ③ 재고보다 많은 경우,
  ④ 음수, ⑤ 0, ⑥ 스트링

- parameter : name, addr
  정상적인 입력값 : ① 스트링일 경우
  비정상적인 입력값 : ② 스트링이 아닌 모든 경우

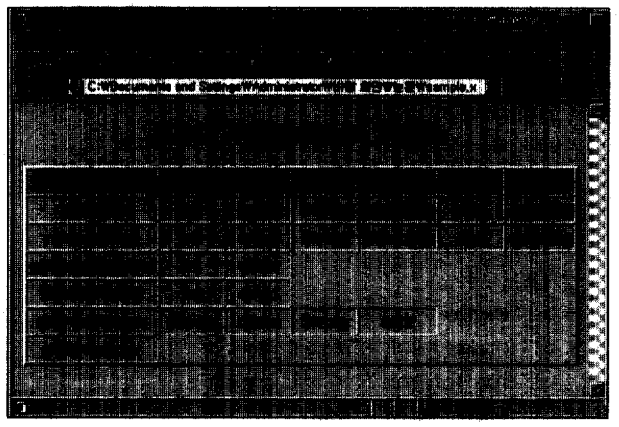
4 단계 : Test Execution

3 단계에서 정리된 메소드별 입력 데이터들을 (그림 2)와 같은 테스트 환경에서 실행시킨다.

5 단계 : Test Evaluation

<표 7>에서 <표 11>은 컴포넌트를 구성하는 메소드들

에 대한 비율검정 결과값이다.



(그림 5) XML 메타데이타 테이블

〈표 6〉 테스트 케이스 테이블

method-name	method-parameter
addItem() removeItem()	100, "K1-00", 200
	900, ***"AA-00"***, ***1000***
	900, ***"AA-00"***, ***0***
	900, ***"AA-00"***, 200
	900, "K1-00", ***1000***
	900, "K1-00", ***0***
	900, "K1-00", 200
	100, ***"AA-00"***, ***1000***
	100, ***"AA-00"***, ***0***
	100, ***"AA-00"***, 200
	100, "K1-00", ***1000***
	100, "K1-00", ***0***
	-100, ***-100***, ***-100***
	-100, ***-100***, 200
	-100, "K1-00", ***-100***
	-100, "K1-00", 200
	100, ***-100***,, ***-100***
	100, ***-100***,, 200
	100, "K1-00", ***-100***
	0, "K1-00", ***0***
	"OK", "K1-00", ***"OK"***
	100, ***100***, 200
getBagContents() finalizeOrder()	100
	900
	0
	"OK"
	-100
newCustomer()	"KSOH", "SEOUL 77"
	100, ***100***
	100, "SEOUL 77"
	"KSOH", ***100***
	-1, ***-1***

〈표 7〉 getBagContents() Method 평가표

입력데이타	결과값	기대값	평가항목	
			만 족	불만족
100	정상	정상	●	
900	정상	정상	●	
0	정상	정상	●	
"OK"	정상	정상	●	
-100	정상	정상	●	

Binomial Test

	Category	N	Observed Prop.	Test Prop.	Exact Big (1-tailed)
VAR00001 Group 1	1.00	5	1.0	.9	.590
Total		5	1.0		

P0 = 0.590, getBagContents() Method : 합격

〈표 8〉 addItem() Method 평가표

입력데이타	결과값	기대값	평가항목	
			만족	불만족
100, "K1-00", 200	정상	정상	●	
900, ***"AA-00"***, ***1000***	정상	정상	●	
900, ***"AA-00"***, ***0***	정상	정상	●	
900, ***"AA-00"***, 200	정상	정상	●	
900, "K1-00", ***1000***	비정상	정상		●
900, "K1-00", ***0***	비정상	정상		●
900, "K1-00"200	정상	정상	●	
100, ***"AA-00"***, ***1000***	정상	정상	●	
100, ***"AA-00"***, ***0***	정상	정상	●	
100, ***"AA-00"***, 200	정상	정상	●	
100, "K1-00", ***1000***	정상	정상	●	
100, "K1-00", ***0***	비정상	정상		●
-100, ***-100***, ***-100***	정상	정상	●	
-100, ***-100***, 200	정상	정상	●	
-100, "K1-00", ***-100***	정상	정상	●	
-100, "K1-00", 200	정상	정상	●	
100, ***-100***,, ***-100***	정상	정상	●	
100, ***-100***,, 200	정상	정상	●	
100, "K1-00", ***-100***	정상	정상	●	
0, "K1-00", ***0***	비정상	정상		●
"OK", "K2", ***"OK"***	정상	정상	●	
100, ***100***, 200	정상	정상	●	

Binomial Test

	Category	N	Observed Prop.	Test Prop.	Exact Big (1-tailed)
VAR00001 Group 1	1.00	18	.8	.9	.172[2]
Group 2	.00	4	.2		
Total		22	1.0		

a.alternative hypothesis states that the proportion of cases in the first group < .9.

P0 = 0.172, addItem() Method : 합격

〈표 9〉 removeItem() Method 평가표

Binomial Test

	Category	N	Observed Prop.	Test Prop.	Exaot Big (1-tailed)
VAR00001 Group 1	1.00	19	.9	.9	.380[2]
Group 2	.00	3	.1		
Total		22	1.0		

a.Alternative hypothesis states that the proportion of oases in the first group < .9.

P0 = 0.380, removeItem() Method : 합격

〈표 10〉 finalizeOrder() Method 평가표

Binomial Test

	Category	N	Observed Prop.	Test Prop.	Exaot Big (1-tailed)
VAR00001 Group 1	1.00	5	1.0	.9	.590
Total		5	1.0		

P0 = 0.590, finalizeOrder() Method : 합격

〈표 11〉 newCustomer() Method 평가표

입력데이타	결과값	기대값	평가항목	
			만 족	불만족
"KSOH", "SEOUL 77"	정 상	정상	●	
100, "SEOUL 77"	정 상	정상	●	
"KSOH", ***100***	비정상	정상		●
100, 100	비정상	정상		●
-1, -1	비정상	정상		●

Binomial Test

	Category	N	Observed Prop.	Test Prop.	Exaot Big (1-tailed)
VAR00001 Group 1	1.00	2	.4	.9	.009[2]
Group 2	.00	3	.6		
Total		5	1.0		

a.Alternative hypothesis states that the proportion of oases in the first group < .9.

P0 = 0.009, newCustomer() Method : 불합격

<표 12>는 사례연구 OnLineShop Component에 대한 최종 결과값이다.

〈표 12〉 OnLineShop Component 평가표

Component Methods	평 가	
	합 격	불합격
addItem()	●	
removeItem()	●	
getBagContents()	●	
finalizeOrder()	●	
newCustomer()		●

Binomial Test

	Category	N	Observed Prop.	Test Prop.	Exaot Big (1-tailed)
VAR00001 Group 1	1.00	4	.8	.9	.410[2]
Group 2	.00	1	.2		
Total		5	1.0		

a.Alternative hypothesis states that the proportion of oases in the first group < .9.

P0 = 0.410, OnLineShop Component : 합격

5. 결 론

소프트웨어 제작에 대한 패러다임 변화로 이를 구성하는 컴포넌트 개발에 대한 방법론은 여러 기관에서 연구되고 있으나 개발되어진 컴포넌트의 품질을 객관적으로 평가하는 테스팅 전략은 미약한 상황이다.

본 논문에서는 이러한 미약한 문제점들을 체계적으로 보완할 수 있도록 현재의 RUP 기반테스팅 프로세스 지침을 실질적으로 구체화 시켰으며 이를 객관적으로 검증하기 위해 EJB 환경하에서 사례연구를 통하여 테스트 케이스를 추출하고 테스트 드라이버를 수행시켰다. 또한 수행된 결과를 비율검증을 통해 통계적으로 컴포넌트의 품질을 평가하였다. 단, 본 연구에서 추출한 테스트 케이스는 컴포넌트 단위 테스팅에 기반을 두고 있으며 통합 테스팅, 시스템 테스팅에 기반을 둔 성능 테스팅은 포함시키지 않았다.

향후 연구방향은 컴포넌트 분석, 설계 다이어그램으로부터 XML 메타데이타를 자동으로 추출할 수 있는 자동화 도구의 생성과 테스트 케이스 생성의 자동화가 필요하며 시스템 테스팅에 기반을둔 성능 테스팅의 기준을 마련해야 할 필요가 있다. RUP기반 5단계 테스팅 프로세스가 컴포넌트를 제작, 생산하는 여러 기관에서 객관적으로 사용할 수 있도록 지금까지의 테스팅 프로세스 과정을 좀 더 보완하고 자동화 해 나갈 계획이다.

참 고 문 헌

[1] Roger Fournier, "A Methodology for Client/Server and Web Application Development," pp.397-444, 1999.

[2] Jose Javier Dolado, "A Validation of the Component-Based Method for Software Size Estimation," IEEE Software Engineering, Vol.26, No.10, Oct. 2000.

[3] E. J. Weyuker, "Testing Component Based Software : A Cautionary Tale," IEEE Software, Vol.15, No.2, Sep. 1998.

[4] J. M. Voas, "Certifying Off-The-Shelf Software Components," IEEE Computer, June, 1998.

[5] 한규정, 김치수, "객체지향 소프트웨어의 테스팅 방법론", 정보통신부 시스템공학연구소 최종보고서, pp.15-23, 1996.

[6] 오승욱, 마유승, 배두환, 권용래, "EJB 컴포넌트 시험평가 체계 및 시험 환경 구축에 관한 연구," 한국정보과학회지, 제19권 제2호, pp.22-30, 2001.

[7] 최은만, "소프트웨어공학론", 사이텍미디어, 2001.

[8] 이상덕, 정효택, 신석규, "공용 컴포넌트 개 발 및 기술개발 전략", 정보처리학회지, Vol.7, No.4, pp.10-17, 2000.

[9] 윤희진, 최병주, "EJB 컴포넌트의 맞춤 테스트 기법", 정보과학회논문지 제28권 제3호, 2001.

오 기 성

e-mail : ksoh@tongwon.ac.kr

1989년 청주대학교 전자계산학과 졸업 (학사)

1991년 숭실대학교 대학원 컴퓨터학과 (공학석사)

2000년 숭실대학교 대학원 컴퓨터학과 (박사수료)

1991년-1996년 (주)다우기술 연구소 근무

1996년-1998년 (주)마이크로소프트사 근무

1998년-현재 동원대학 컴퓨터정보과 조교수

관심분야 : 소프트웨어공학, 컴포넌트 테스팅, 컴포넌트 품질평가, 컴포넌트 개발

류 성 열

e-mail : syrhew@computing.soongsil.ac.kr

1996년 아주대학교 컴퓨터학부(공학박사)

1997년~1998년 George Mason Univ. 교환교수

1981년~현재 숭실대학교 컴퓨터학부 교수

1998년~현재 숭실대학교 정보과학대학원 원장

1998년~현재 숭실대학교 전자계산원 원장

관심분야 : 리엔지니어링, 분산 객체 컴퓨팅, 소프트웨어 재사용, 전자상거래, 소프트웨어 테스팅