CBD 기반 소프트웨어에 대한 유스케이스 기반 테스트모델

A USE CASE based Test Model for CBD Based Software

유지호(Ji Ho You)**, 이남용(Nam Yong Lee)*

초 록

품질 좋은 소프트웨어란 성능이 뛰어나고 가격이 저렴한 소프트웨어를 일컫는 말이다. 또한, 시장 요구에 부응하는 서비스를 지원하면서도 오류가 적고 유지보수하기 쉬운 구조를 가져야 한다. 이 모든 것을 충족하는 소프트웨어를 개발하기 위한 방법으로 최근 CBD(Component Based Development)가 각광 받고 있다. CBD(Component Based Development) 확산과 더불어 컴포넌트 구성, 기능, 호환성, 신뢰성을 위한 컴포넌트 테스트를 수행하고 있으나 이것만으로는 CBD(Component Based Development) 기반 소프트웨어 품질을 보장할 수 없다. 높은 신뢰성, 유지보수성, 재활용성을 가지는 단위 컴포넌트라도 조립, 통합 후에는 의도한 요구사항을 만족시키지 못하고 문제를 일으킬 수도 있으므로 이에 대한 통합 테스트 기법이 필요하다.

본 논문은 UML (Unified Modeling Language)기반 소프트웨어 개발에 컴포넌트 통합 테스트를 위한 것 이고, 테스트 기법은 UML (Unified Modeling Language)의 순차다이어그램과 협력다이어그램을 참조하여 컴포넌트 사이 인터페이스에 존재하는 결점을 추출하는 테스트기법이다.

ABSTRACT

High quality software is the one that has an excellent performance with a low price, consumer's request satisfaction and less bug. In addition, it should have structure easily maintainable.

CBD(Component Based Development)technology is getting more popular as a method for software development which can satisfy all above conditions. Although there are a lot of test methods about component for composition, function, compatibility and reliability, it's not enough to ensure that the component has reliability, maintainability, and reusability, so an integated method are necessary.

In this paper, we are going to propose the solution model for integrating components, which are based on UML (Unified Modeling Language) technologies. This model can extract existing faults in the component interfaces by referencing sequence diagram and collaboration diagram.

키워드 : 컴포넌트기반개발, 소프트웨어 테스트, UML(Unified Modeling Language), USE CASE, 테스트모델
CBD(Component Based Development), Software Test, UML(Unified Modeling Language), USE CASE, Test Model

본 연구는 숭실대학교 교내연구지원으로 이루어졌음.

* 숭실대학교 컴퓨터학부
** 숭실대학교 컴퓨터학과

1. 서 론

컴포넌트에 기초한 소프트웨어 개발기술은 소프트웨어 위기를 극복할 만한 개발 생산성을 높일 수 있는 방법으로 주목 받고 있다. 소프트웨어 업계에서의 성공이란 수행 속도가 더 빠르고, 성능이 뛰어나면서도 저렴한 소프트웨어를 개발하는 것이다. 소프트웨어는 시장의 요구에 부응하여 현재의 요구사항 혹은 장래의 요구사항에 대한 서비스를 지원하면서도 수행 시 더 적은 오류를 포함하고 생산 및 운영 비용이 저렴해야 한다. 이러한 성공 요소들의 관점에서 CBD개발 프로세스의 확산, 반복적 개발프로세스의 확산, SI 산업구조의 변화에 따라 컴포넌트 기반의 개발(Component-Based Development)은 학계와 업계의 관심을 동시에 받고 있다.

이러한 CBD(Component Based Development) 개발의 확산에 따라, 컴포넌트의 구성 자체의 기능, 호환성, 신뢰성, 즉 품질을 보장하기 위한 수단으로 컴포넌트의 재사용 또는 조립단계에서 일어날 수 있는 오류를 철저히 예방할 수 있는 품질 평가 및 시험평가 방법에 대한 많은 연구와 노력이 진행되어 왔다. 그러나 이러한 고품질의 컴포넌트들이 조립, 통합되어 최종 소프트웨어 및 시스템이 개발되었을 경우의 품질을 확신하기에는 무리가 따른다. 따라서, 컴포넌트를 기반으로 개발된 소프트웨어 품질을 확신하고, 제고하기 위해서는 컴포넌트 기반 소프트웨어에 특성화되어 개발된 시험평가 시스템을 활용할 수 있는 방안을 확대하여 추진할 필요성이 있다. 그러나 현재 컴포넌트 기반 소프트웨어의 시험평가에 대한 체계적인 연구 및 추진사례가 없기 때문에 본 논문은 "컴포넌트 기반 소프트웨어의 통합 테스트 모델"을 개발하고 이에 대한 효용성을 검증하고 보완하기 위한 사례연구 수행을 그 목표로 한다. 또한 기존의 테스트케이스 선정과 본 연구에서 진행한 테스트케이스 선정 방법에 대한 차이점을 서술했다.

테스트케이스식별모델은 UML(Unified Modeling Language) 의 순차다이어그램과 협력다이어그램을 사용하여 통합 타겟 사이에서의 메시지로부터 통합테스트 아이템을 추출한다. 우리는 이와 같은 테스트케이스식별모델과 통합테스트기법을 정보검색시스템에 적용한다.

2. 관련연구

2.1 컴포넌트

CBD(Component Based Development) 기반 소프트웨어 테스트 모델을 언급하기에 앞서 CBD(Component Based Development) 기반 소프트웨어를 구성하고 있는 컴포넌트에 대한 이해가 필요하다. 컴포넌트에 대한 정의는 〈표 1〉과 같다.

즉, 컴포넌트는 '특정한 기능을 수행하기 위해 독립적으로 개발 보급하고 인터페이스를 통해 시스템을 조립 구축할 수 있는 소프트웨어 단위'라고 규정할 수 있다. 컴포넌트 특성을 정리하면 다음과 같다.

1) 컴포넌트는 모듈 단위이다.

〈표 1〉 주요 컴포넌트 정의

제안자	컴포넌트 정의 및 내용
Philippe Krutchen	아키텍처 상에서 특정 기능을 수행하는 시스템 내 독립적이고 대치 가능한 부분이며 인터페이스 집합에 대한 물리 구현을 제공한다.
Gartner Group	동적 바인드 할 수 있는 프로그램들을 한 단위로 관리하는 패키지이며 런타임 시에 인터페이스를 통해 접근 가능하다.
Clemens Szyperski	계약으로 명시한 인터페이스와 문맥 의존성을 가진 구성 단위로서 독립적으로 배포할 수 있다.
Wojtek Kozaczynski	자발적인 비즈니스 객체 또는 비즈니스 로직을 소프트웨어로 구현한 것이다.
D'Souza	프트웨어 구현을 갖는 패키지이고 독립적으로 개발 배포할 수 있으며 제공 인터페이스와 요구 인터페이스를 가지고 속성을 커스터마이즈 할 수 있다.

2) 컴포넌트는 인터페이스를 갖는다.
3) 컴포넌트는 논리모델이 아닌 실제 구동 가능한 모듈이다.
4) 컴포넌트는 아키텍처를 기반으로 한다.
5) 컴포넌트는 커스터마이즈가 가능하다.

2.2모델 기반 테스팅

CBD(Component Based Development) 기반 소프트웨어 테스트 모델을 개발하기 위해선 소프트웨어 테스트 모델에 대한 개념의 정립과 소프트웨어 테스트 모델 수립 목적을 살펴봐야 한다.

소프트웨어는 그 자체의 복잡함 때문에 테스트 설계(Test Design) 지원을 위한 모델 개발이 요구된다. 특히, 테스트 설계자는 소프트웨어 개발 과정 또는 최종으로 발생될 수 있는 입력 데이터를 이용하여 테스트 설계를 위한 가용적인 모델을 제안해야 한다. 이러한 소프트웨어의 완전한 검증(Verification)과 확인(Validation)을 위해 소프트웨어 테스트가 모델을 기반으로 제시되어야 하는 목적을 분석하면 다음과 같다.[1]

이러한 모델 기반 테스팅은 소프트웨어에 대해 확인하고자 하는 실질적인 관계를 식별하여 쉽게 테스팅이 개발되고 수행될 수 있게 한다.

모델 기반 소프트웨어 테스팅의 목적

① 모델기반 소프트웨어 테스팅은 입력과 상태의 조합, 모든 대상을 포함하도록 체계적인 목록을 제공
② 모델기반 소프트웨어 테스팅 모델은 버그가 발생하기 쉬운 조합을 제공
③ 모델기반 소프트웨어 테스팅은 자동화된 테스팅 기법을 제공

〈표 2〉 모델 기반 테스팅의 중요 구성요소

구성요소	내용
주제(Subject)	모델은 테스트 하고자 하는 소프트웨어의 효과적인 테스트 케이스를 선정하도록 도와준다
이론의 관점 (Point of Theory)	소프트웨어 테스트 모델은 요구된 행위를 표현하고, 버그가 예상되는 구조 또는 요소에 초점을 맞춘다. 테스트 케이스를 생성하고 평가하기 위해 필요한 정보를 제정해야만 한다
표현 (Representation)	소프트웨어 테스트 모델은 그래프를 사용한다. 테스트 모델의 일반적인 형태는 체크리스트이다. 예를 들면, Myer 의 동치 클래스 기준, Hamlet's suspicion, Marick's 테스트 요구사항 카탈로그를 포함시킬 수 있다.
기법 (Technique)	모델은 복잡한 산출물이다. 소프트웨어 테스트 모델링의 접근방법은 방대한 문헌 커버링 스타일과 발견적 지도법에 있다

〈표 2〉는 모델 기반 테스팅의 중요 구성요소이다.

3. 테스트케이스식별모델

컴포넌트의 특징이 가장 두드러진 테스트 단계는 통합테스트이다.

일반 소프트웨어의 개발프로세스는 1)요구사항분석 2)설계 3)구현 4)테스트 5)배포 단계로 진행되는 반면,

컴포넌트 기반 개발방법은 1)요구분석, 컴포넌트 획득 2)컴포넌트 기반 설계 3)컴포넌트 조립 4)통합시험 5)인수시험 5단계로 진행된다고 볼 수 있다.

따라서 두 방법을 비교해본 결과 컴포넌트의 테스트 기법 중 가장 중요하다고 생각되는 기법은 통합테스트이다. 컴포넌트 기반 개발방법은 단위테스트가 완료된 컴포넌트를 분석단계 후 획득하여 조립하는 방법이기 때문에 통합테스트를 중요하게 부각시킨 이유이다. 이와 같은 통합 테스트를 하는데 있어 적용할 테스트케이스를 선정하는 모델을 테스트케이스식별모델이라 하겠다.

테스트케이스식별모델은 UML(Unified Modeling Language)의 순차다이어그램과 협력다이어그램을 참조하여 수행하는 통합테스트를 위한 새로운 테스트모델이다. 단위테스트는 단위 안에서 테스트할 수 있으나 다른 단위 사이의 인터페이스에서 발생하는 결점들을 발견하지 못하고 지나칠 수 있다.

인터페이스에서 이들 결점은 통합테스트를 통하여 발견될 수 있고, 여기서 제안하는 테스트케이스식별모델은 인터페이스에서 결점을 검출하는 테스트 아이템을 식별할 수 있다.

3.1 UML모델과 테스트 기법

이번 단락에서는 개발 프로세스와 테스트 기법사이의 관계를 보여줄 것이다. 개발 프로

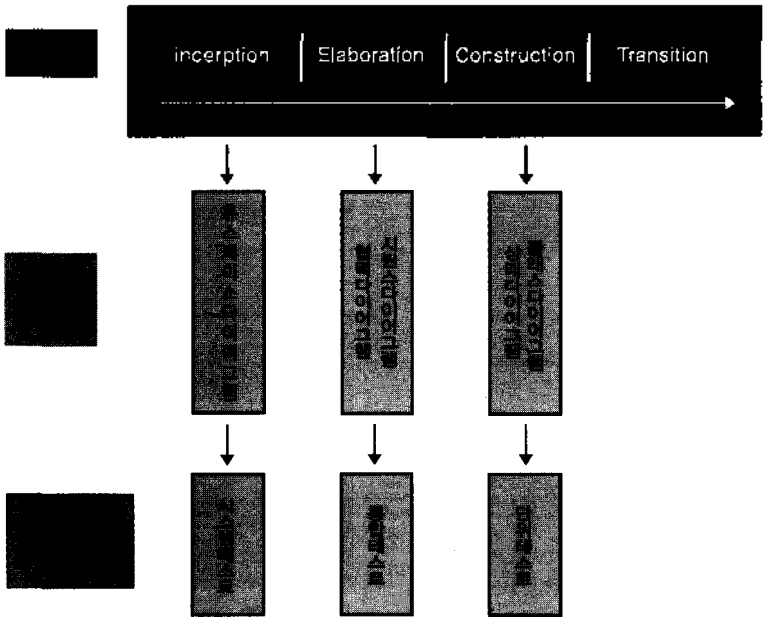
세스에는 RUP(Rational Unified process)를 사용할 것이며, RUP는 도입(INCEPTION), 상세(ELABORATION), 구축(CONSTRUCTION), 전이(TRANSITION) 4 단계로 구성 되어있다.

도입단계는 소프트웨어 요구사항과 명세서를 일일이 열거하고, 소프트웨어가 사용자 요구사항을 만족시키는지의 여부를 테스트하는 시스템 테스트 레벨을 위한 중요한 입력으로 사용되는 유스케이스다이어그램을 도출한다. 소프트웨어가 설계되는 상세단계는 순차 이벤트에 대한 순차다이어그램과 한 시나리오에서 발생하는 동시 이벤트에 대한 협력다이어그램을 도출한다. 이러한 다이어그램들은 상세 기능 단위의 통합 테스트 기법을 위한 중요한 입력으로 사용된다.

소프트웨어가 구현되는 구축단계는 각각 클래스의 상태를 위한 클래스와 상태다이어그램의 속성과 메소드를 위한 클래스다이어그램을 생성한다. 이들 다이어그램은 단위테스트기법에서 사용된다. 다음 〈그림 1〉은 개발프로세스, UML (Unified Modeling Language) 모델 그리고 테스트기법사이의 관계를 보여준다.

3.2 UML(Unified Modeling Language) 테스트모델의 명세서

본 논문에서의 테스트케이스식별모델은 다른 컴포넌트의 통합 장소인 인터페이스 사이에서 결함을 검출하는 테스트항목을 추출한다. 테스트케이스식별모델은 설계문서 중 하나인 순차다이어그램에서 통합 타겟을 표현하는 노드와 이들 노드 사이의 상호작용을 표현하는 메시지 흐름으로 구성되어 있다.



〈그림 1〉 UML모델과 테스트레벨

노드는 하나의 자동화시스템기능(ASF) 상태를 표현하고, 메시지흐름은 자동화시스템기능(ASF)의 절차를 표현한다. 자동화시스템기능(ASF)은 한 컴포넌트에서 객체 사이의 메시지흐름으로 구성되어 있는 컴포넌트를 실행시키는 단위를 말하며, 이와 같은 노드 사이의 메시지흐름으로 절차를 표현한다.

3.3 컴포넌트 통합테스트기법

컴포넌트의 범위를 어떻게 결정할 것인지가 여전히 컴포넌트 기반 소프트웨어 공학에서 중요한 이슈이다. 우리는 하나의 이벤트의 흐름으로부터 컴포넌트의 통합까지를 통합테스트 타겟으로 식별한다. 컴포넌트 통합테스트 기법은 첫째로 각각의 이벤트 흐름별로 테스트케이스식별모델을 구축하고 모델상에서 통합테스트 케이스를 선택한다. 예를 들어, 〈그림 2〉와 같이 이벤트 흐름상의 컴포넌트 그룹 A를 위한 테스트케이스식별모델이 구축되고 통합된 테스트케이스가 모델상에서 선택된다. 우리의 컴포넌트 통합테스트기술은 아래와 같이 기술했다.

단계 1 : 테스트케이스식별모델 구축

서브순차단계를 가진 이벤트 흐름상에 컴포넌트를 위한 테스트케이스식별모델을 구축한다.

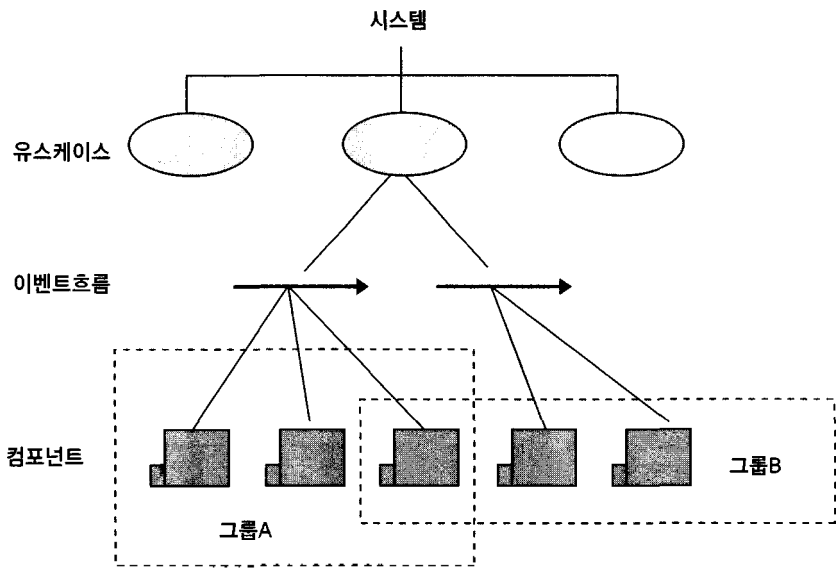
단계 1.1 : 순차다이어그램을 추출

이벤트의 각 흐름을 위한 정상흐름과 예외흐름을 보여줄 수 있는 순차다이어그램을 추출한다.

단계 1.2 : 협력다이어그램을 추출

동시 발생하는 케이스에서 각 이벤트 흐름을 위한 협력다이어그램을 추출한다.

단계 1.3 : 자동시스템기능 단위에서 순차다이어그램과 협력다이어그램을 분리



〈그림 2〉 통합 테스트 타겟

컴포넌트 사이에서 발생하는 메시지 변이에 따라서 자동시스템기능 단위의 순차다이어그램과 협력다이어그램을 분리한다.

단계 1.4 : 테스트케이스식별모델을 기술

자동시스템기능(ASF)으로부터 노드와 메시지 흐름을 기술함으로써 테스트케이스식별모델을 완성한다.

단계 2 : 통합테스트케이스 선택

테스트케이스식별모델에서 테스트케이스 선택기준을 적용하여 테스트케이스를 선택한다.

3.4 기존 테스트케이스 선정과 차이점

컴포넌트 기반 개발 방법론이 발전하는 속도에 비례하여 테스트 기술의 발전도 필요하다. 전통적인 방법과 비교하여 컴포넌트 기반 소프트웨어의 테스트는 매우 차이가 난다. 전통적 패러다임에 의한 프로그램은 프로시저나 함수를 테스트 단위로 하여 입력 데이터 값들을 테스트케이스로 선정하는데 비하여 컴포넌트 기반 소프트웨어의 테스트는 컴포넌트간의 연결고리를 인터페이스로 하기 때문이다.

기존의 방법과 차이점을 논하기 전에, 컴포넌트 기반 개발이 일반 소프트웨어 개발방법과 어떻게 다른지 알아 볼 필요가 있다.

일반 소프트웨어의 개발프로세스는 1)요구사항분석 2)설계 3)구현 4)테스트 5)배포 단계로 진행되는 반면,

컴포넌트 기반 개발방법은 1)요구분석, 컴포넌트 획득 2)컴포넌트 기반 설계 3)컴포넌트 조립 4)통합시험 5)인수시험 5단계로 진행된다고 볼 수 있다.

두 방법간의 가장 두드러진 차이점은 선정과 조립단계의 추가이다. 컴포넌트 기반 개발은 단위테스트가 끝난 상용 컴포넌트를 조립하고 조립된 컴포넌트에 대한 통합테스트가 가장 중시된다. 따라서 조립되고 나서 이를 테스트하기 위해 컴포넌트를 구입할 때 설계문서도 함께 획득하여 이를 대상으로 테스트케이스를 선정하는 것이 가장 두드러진 차이점이다.

좀 더 자세히 설명하면, 기존의 테스트케이스 선정 방법은 데이터 흐름 다이어그램이나 데이터 컨트롤 다이어그램을 가지고 경우의 수를 측정하여 테스트 케이스를 선정하는 방법으로 수행했었다. 이러한 방법으로 테스트케이스를 선정할 때 선정하는 사람에 따라 테스트케이스의 수와 종류가 다양하게 식별될 수 있다. 이는 테스트를 수행하는 테스터에 따라 테스트 결과가 다르게 나온다는 것을 의미한다. 그러나 본 연구에서 제시된 테스트케이스식별모델을 테스트에 적용하여 테스트케이스를 선정할 때는 다양한 테스터에 따른 테스트케이스 식별결과가 거의 같게 나온다. 이는 테스트에 대한 신뢰성이 상당히 높아 질 수 있다는 것을 의미한다. 어느 누가 테스트를 한다고 해도 테스트케이스가 거의 유사하게 선정된다면 테스트케이스 선정 자동화에 대해 상당한 기여를 할 수 있을 것이다.

3.5 사례연구

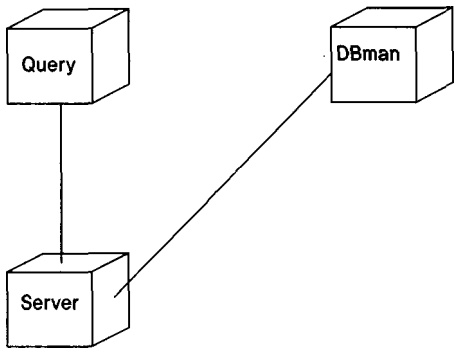
이번 단락에서는 간단한 정보를 찾아 보여주는 사스템의 테스트케이스식별모델과 컴포넌트 통합 테스트기술을 적용할 것이다.

3.5.1 문제정의

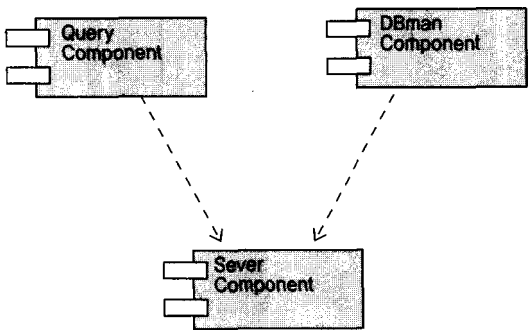
UML에 기반하여 설계된 정보검색시스템은 개인적인 주소, 전화번호 그리고 다른 개인적인 자료를 찾고 관리하는 시스템이다. 3개의 분산된 객체로 이루어져 있는 이 시스템은 데이터 베이스를 운영하는 객체인 DBman, 서비스를 제공하기 위한 객체인 Server 그리고 정보를 검색하기 위한 객체인 Query로 이루어져 있다. <그림 3>은 정보검색시스템의 분산된 구조를 보여주는 배치다이어그램이다. 그리고 <그림 4>는 이 시스템의 컴포넌트를 보여주는 컴포넌트다이어그램이다.

정보검색시스템은 2개의 이벤트흐름으로 이루어져있다. 첫번째는 DBman 컴포넌트와 Server 컴포넌트를 사용하는 데이터베이스 관리이고, 두번째는 Query 컴포넌트와 Server 컴포넌트를 사용하는 데이터베이스 사용이다. 이들 관계는 <그림 5>에 표현되었다.

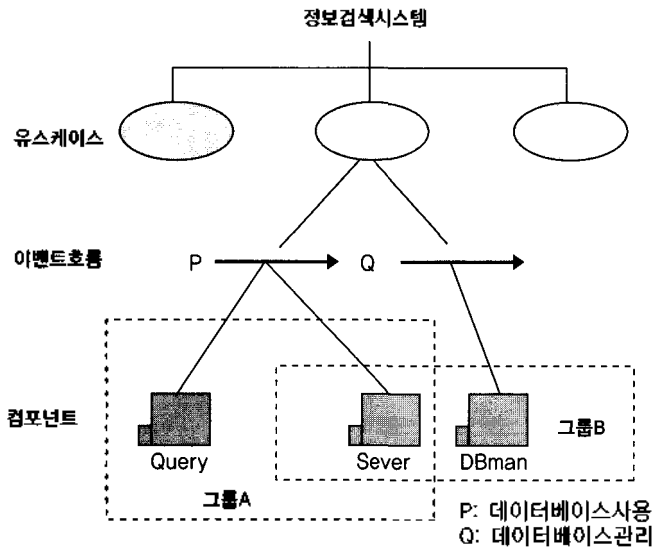
그런 이유로, 통합테스트는 <그림 5>의 정보검색시스템에서 그룹 A와 그룹 B를 위해 필요하다. 사례에서 컴포넌트 통합테스트 기술을 적용해서 Query 컴포넌트와 Server 컴포넌트를 구성하고 있는 그룹을 테스트한다. 첫



<그림 3> 배치다이어그램



<그림 4> 컴포넌트 다이어그램



〈그림 5〉 정보검색시스템을 위한 통합테스트 타겟

째로 DB사용 이벤트흐름을 위한 테스트케이스식별모델을 구축한다. 그리고 나서 테스트케이스를 선정한다. 이 시스템에 대해 선정된 테스트케이스 적용에 의해 우리는 DB사용 이벤트흐름을 위한 Query 컴포넌트와 Server 컴포넌트의 통합을 테스트한다.

3.5.2 컴포넌트 통합테스트 기술의 적용

Query 컴포넌트와 Server 컴포넌트가 단위테스트 레벨에서 테스트된 이후, 이들 컴포넌트에 결함이 없다고 가정하는 것은 위험하다. 단위테스트가 완료되더라도 단위 컴포넌트들이 조립된 후 결함이 검출될 수도 있기 때문이다.

아래의 단계들은 DB사용 이벤트흐름을 위한 컴포넌트 통합테스트 기술을 적용한 절차를 나타낸 것으로써, 컴포넌트 사이 인터페이스에 대한 테스트를 보여준다.

단계1 : 테스트케이스식별모델을 구축

단계 1.1

우리는 정상흐름과 비정상흐름을 위한 순차다이어그램을 추출한다. 정상흐름은 어떠한 결과가 질문에 합당할 때 발생한다. 그리고 비정상흐름은 다음의 3가지 케이스일 때 발생한다.

첫째, 다중결과가 제시될 때,

둘째, 결과가 없을 때

셋째, 다중결과를 위한 부 결과가 없을 때이다.

단계 1.2

이 사건흐름이 동시에 발생하지 않을 때 우리는 협력다이어그램을 추출하지 않는다. 그래서 테스트케이스식별모델을 위한 입력으로 순차다이어그램만을 사용한다.

단계 1.3

〈그림 6〉은 정상흐름을 위한 순차다이어그램이다. 〈그림 6〉에서 MainWindow 객체와 QueryByName 객체는 Query 컴포넌트에 포함된다. 그리고 information_i 객체는 Server 컴포넌트에 포함된다. 우리는 Query 컴포넌트와 Server 컴포넌트 사이의 전송되는 메시지에 따라서 〈그림 6〉의 ASF들로 순차다이어그램을 분해한다. 세 가지의 비정상의 흐름을 위한 다른 순차다이어그램들도 또한 ASF들로 분해된다.

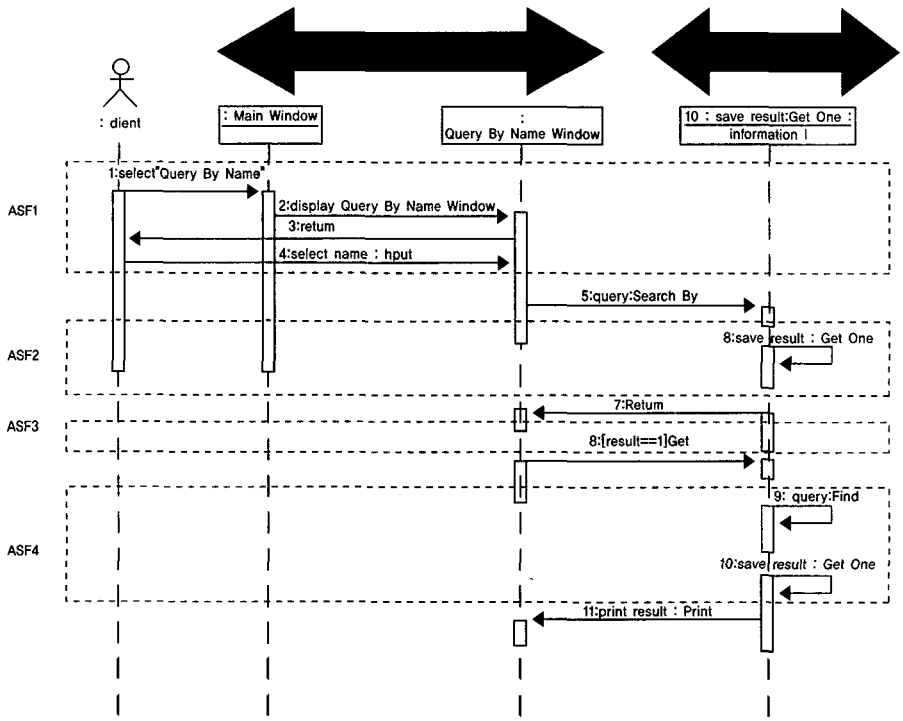
단계 1.4

하나의 ASF는 노드이고 두가지 컴포넌트들 사이의 전송된 메시지는 우리의 테스트케이스식별모델을 위한 메시지 흐름이다. 우리는 하나의 정상흐름과 셋의 비 정상흐름들을 위한 순차다이어그램들로부터 노드들과 메시지흐름들의 추출에 의해 하나의 테스트케이스식별모델을 제시했다. Query 컴포넌트와 Server 컴포넌트의 통합테스트를 위한 테스트케이스식별모델은 〈그림 7〉과 같다.

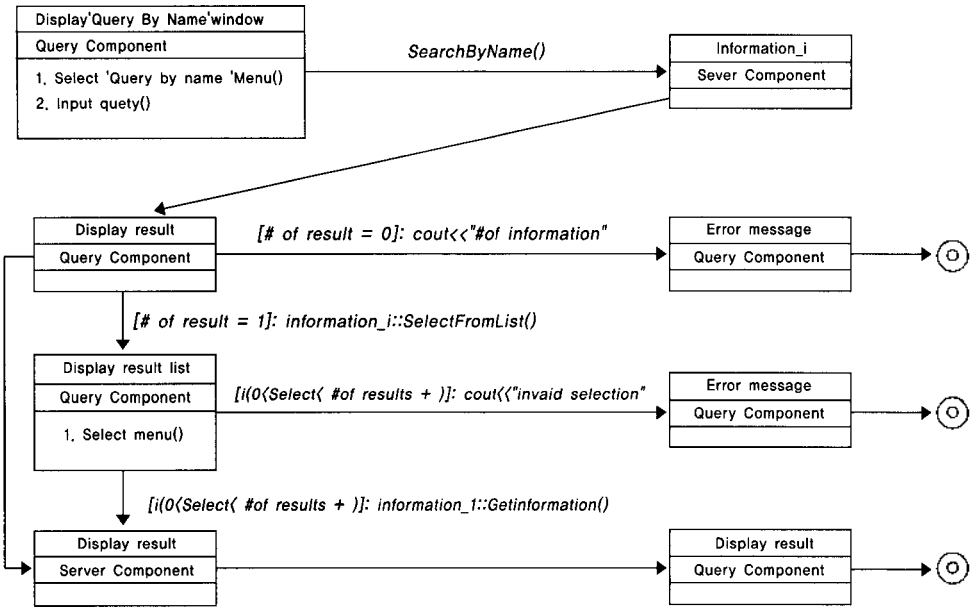
단계2 : 통합테스트케이스를 선정

〈그림 7〉의 테스트케이스식별모델을 위한 모든 테스트케이스선정기준을 적용한다.

테스트케이스식별모델의 모든 경로가 테스트케이스 선정기준이다. 전체 테스트케이스 선정기준은 다이어그램의 모든 부분을 포함한다. 테스트케이스는 방법의 순서와 입력데이터, 예상되는 출력, 이렇게 3가지 아이템으



〈그림 6〉 정상흐름의 순차다이어그램을 위한 ASFs



〈그림 7〉 테스트케이스식별 모델

로 구성된다. 선정된 경로는 방법의 각 순서를 생성한다. 그런 이유로 테스트케이스 TC1, TC2, TC3, TC4는 〈표 3〉과 같이 구분할 수 있다.

3.5.3 사례에 대한 분석

본 논문에서 정보검색시스템에 대한 테스트케이스들의 적용에 대한 결과를 보여주기 위해 〈표 3〉에서 예상되는 출력을 비교했다. TC1의 예에서 결과는 예상되는 결과가 홍길동일 때 홍길동 대신에 다른 사람의 정보가 된다. 그런 이유로 우리는 Query 컴포넌트와 Server 컴포넌트의 통합에 결점이 있다는 것을 검출했다. Query 컴포넌트가 'result = new FullInfo' 상태를 가지고 메모리로 할당되고 나서 GetInformation(...,fullinfo)에 'result = new FullInfo' 상태로 메모리에 두 번째로 할당되게 만드는 Server 컴포넌트의GetIformation(...,fullInfo)를 호출한다.

이 결점은 각각의 컴포넌트에 적용되는 단위테스트에서는 검출되지 않을 수 있다. 왜냐하면 단위테스트는 한 컴포넌트에서 부분적으로 테스트 하기 때문이다. 그러나 컴포넌트의 통합에 존재하는 결점은 여기서 제안한 테스트케이스식별모델을 구축하고 통합테스트를 수행하여 검출될 수 있다. 본 논문에서 우리는 완전하게 테스트될 수 있는 블랙박스로 통합 타겟을 취함으로써 통합타겟들 사이에 인터페이스를 위한 테스트케이스를 선정한다. 유스케이스와 이벤트흐름을 위한 통합 테스트는 컴포넌트들의 통합테스트만큼은 중요하지 않다. 이것은 이벤트흐름과 유스케이스 사이에서 발생하는 메시지 전송이 없기 때문이다.

4. 결 론

본 논문에서 우리는 CBD(Component Based Development)기반 소프트웨어 통합테스트모델을 제안했다. 이 모델은 테스트를 위한 입력으로 UML(Unified Modeling Language)의 순차다이어그램과 협력다이어그램을 사용한다면 준비시간이 줄어들고 소프트웨어 개발 프로세스의 전반적인 단계상에 일관된 테스트 케이스가 선정 될 수 있다. 가장 중요한 이점은 일관된 테스트케이스가 선정된다는 것이다. 예컨대 테스트케이스 선정에 대하여 기존의 방법들은 테스터에 따라 주관적인 테스트케이스가 선정되는 것이 당연시 되었으나, 본 테스트케이스식별모델을 도출하여 테스트케이스를 선정한다면 각기 다른 테스터가 테스트케이스를 선정한다 해도 같은 결과가 나올 것이다. 그리고 테스트케이스식별모델은 UML(Unified Modeling Language) 기반 개발 프로세스와 테스트 프로세스가 자연스럽게 연결되게 만든다. 그런 까닭에 테스트케이스식별모델을 구축하고 테스트케이스를 선정하는 테스트 기법의 자동화가 가능해진다. 기존의 테스트와 차이점은 컴포넌트에 대한 테스트 모델이라는 것, 즉 컴포넌트의 인터페이스를 대상으로 테스트한다는 것이다.

본 모델이 제시됨으로써 컴포넌트 기반 기술의 개발 및 응용 컴포넌트 개발 분야에 기여하게 된다. 즉, 한 번 작성된 모델은 약간의 조정을 통하여 재사용 될 수 있으므로 테스트케이스식별모델은 장기적으로 시간과 노력을 더욱 줄일 수 있는 경제적인 방법이다.

앞으로 본 모델의 완전성을 위해서 보다 규모가 큰 CBD(Component Based Development) 기반 소프트웨어를 대상으로 실제 적용을 해야 하고, 시스템테스트나 인수테스트에 적용할 테스트 모델을 개발할 필요성이 있다. 또한 테스트모델의 범위를 확대하여 다양한 품질 특성에 따른 테스트모델에 대한 연구가 필요할 것이다.

참 고 문 헌

[1] 류성렬, 이남용, 오기성, Software Testing, 도서출판 글로벌, 2002

[2] B. Jeng, E.J.Weyuker, "Analyzing Partition Testing Strategies", IEEE Trans.Soft Eng, 17(7):703-711, July 1991.

[3] D. Mandrioli, S. Morasca, A. Morzenti, "Generating Test Cases for Real-Time Systems from Logic Specifications", ACM Trans. Computer Systems, Vol. 13, No. 4, pp. 365-398, Nov. 95

[4] Peter Zadrozny, J2EE Performance Testing with BEA Weblogic Server, EXPERT PRESS, 2002

[5] Principles of Functional Testing, RATIONAL SOFTWARE, 2000

[6] 정기원, 강래성, A Test Case Generation Technique from Component Specification, 석사학위 졸업논문, 2002

[7] Arshad Jhumka, Martin Hiller, Neeraj Suri, "An Approach to Specify and Test Component-Based Dependable Software", IEEE Software, 8(2):58-65, March 1996.

[8] J. Zhuo, P. Oman, R. Pichai, S. Sahni, "Using Relative Complexity To Allocate Resources In Gray Box Testing Of Object-Oriented Code", Proc.4th Intl Software Metrics Symposium, pp. 74-81, 1997

[9] Elaine. J. Weyuker, Testing component-Based Software : A Cautionary Tale, IEEE Softeare, 1998

저 자 소 개

유지호 (E-mail : yj789@beans.soongsil.ac.kr)
2001. 상명대학교 소프웨어공학과(공학사)
현재 숭실대학교 일반대학원 컴퓨터학부 소프트웨어공학(공학석사)
관심 분야 : CBD, S/W Testing, S/W Metrics, Object Oriented Technology, Middleware 등

이남용 (E-mail : nylee@computing.ssu.ac.kr)
1980. 숭실대학교 컴퓨터공학(공학사)
1984. 고려대학교 MIS(석사)
1994. 미시시피주립대학교 MIS(박사)
2000. ~ 현재 숭실대학교 컴퓨터학부 교수
관심 분야 : 객체기술(UML, RUP), 컴포넌트기술(OOAD, CBD), 테스팅기술 (Functionality, Performance, Reliability Test)