UML/OCL을 이용한 기업형 컴포넌트의 자동화 시험 환경

김상운[+] ○ 마유승[+] 강제성[+] 배두환[+] 권용래[+]
[+]한국과학기술원 전산학과
{ swkim, ysma, jskang, bae, kwon}@salmosa.kaist.ac.kr

Generating Automated Testing Environment for Enterprise Components using UML/OCL

Kim, Sang-Woon[+] ○ Ma, Yu-Seung [+] Kang, Jae-Sung [+] Bae, Doo-Hwan [+] Kwon, Yong-Rae[+]
[+]Department of Computer Science, KAIST

요 약

기업형 정보 시스템을 개발하는 데 클라이언트 계층, 어플리케이션 서버 계층, 데이터베이스 계층으로 구성된 3계층 아키텍처가 널리 사용되고 있다. 따라서 기업형 컴포넌트의 올바른 행위를 시험하기 위해서는 3계층 아키텍처를 고려한 시험 기법이 요구된다. 하지만 기존의 대부분의 컴포넌트 시험 기법들은 클라이언트 계층과 어플리케이션 서버 계층 사이의 관계만을 대상으로 하고 있어서 기업형 컴포넌트 시험에 부족하다.

논문에서는 기업형 컴포넌트의 시험을 위해 클라이언트 계층과 어플리케이션 서버계층 간의 관계만이 아니라 어플리케이션 서버계층과 데이터베이스 서버계층과의 관계를 포함한 시험 기법을 제안한다. 이를 위해 3계층 아키텍처를 반영하는 시험모델을 제안했으며 UML/OCL를 컴포넌트의 명세로 사용하여 시험모델을 추출한 뒤 자동으로 시험을 수행하는 시험 환경을 제안했다. 제안된 시험 환경은 일반적인 시험 단계의 뒷부분으로 테스트 케이스를 분석하여 생성하는 것보다는 생성된 시험 자료를 수행시켜 자동으로 시험 과정을 수행하는데 관심을 두고 있다. 제안된 시험환경은 기존의 연구와 달리 3계층 아키텍처를 반영하고 산업체 표준인 UML/OCL을 이용하므로 기업형 응용 프로그램의 생산성을 증가시켜 줄 것으로 보인다.

1. 서론

기업형 정보 시스템을 개발하는 데 사용자 인터페이스를 구성해주는 클라이언트(Client) 계층과 시스템의 기능(Business Logic)을 제공해주는 어플리케이션 서버(Application Server) 계층, 그리고 시스템에서 사용하는 데이터와 관련기능을 제공하는 데이터베이스 서버(Database Server) 계층으로 구성된 3계층 아키텍처가 널리 사용되고 있다[1]. 3계층 아키텍처는 계층 간의 독립성을 최대한 유지함으로써 데이터베이스를 바꾼다거나, Business Logic을 교체할 때의 비용이 적어지게 되고, 자원의 효율적인 재사용이 가능해진다.

현재 사용되는 서버측 컴포넌트 아키텍처로는 Microsoft의 DCOM, Sun Microsystems의 J2EE, OMG의 CORBA가 있다. 이들은 3계층 아키텍처를 지원하는 컴포넌트 아키텍처로서, 이를 기반으로 구현되는 기업형 시스템은 3계층의 사용의 장점뿐만 아니라 컴포넌트의 사용으로 인한 장점들도 얻을 수 있다.

본 논문에서는 3계층 컴포넌트 아키텍처를 따르는 컴포넌트를 기업형 컴포넌트(Enterprise Component)라 정의한다. 기업형 컴포넌트는 넓은 의미의 컴포넌트보다 3계층에 따른 행위가 강조되므로, 기업형 컴포넌트의 올바른 행위를 시험하기 위해서는 3계층 아키텍처를 고려한 시험 기법이 필요하다. 하지만 기존의 컴포넌트 시험 기법들은 클라이언트 계층과 어플리케이션 서버계층의 두 계층 사이의 상호관계만을 대상으로 하고있다.

본 논문에서는 기업형 컴포넌트의 시험을 위해 기존의 컴포넌트 시험 기법을 바탕으로 어플리케이션 서버계층과 데이터베이스 서버 계층과의 관계도 시험 대상으로 포함하는 시험 기법을 제안한다. 그리고 제안된 시험 기법에 따라 자동으로 시험을 수행하는 시험 환경의 구축을 목표로 한다. 제안하는 시험 기법은 산업계 표준 모델링 언어인 UML/OCL을 컴포넌트의 명세로 받아들이고, 명세로부터 컴포넌트와 데이터베이스와의 관계를 추출한다. 그리고 테스트 케이스를 수행하면서 추출된 관계가 올바른지를 감시하는 것까지를 포함한다. 자동화 가능한 테스트 케이스의 수행 및 감시 기능은 일반적인 시험 과정의 후반부(Back-End)에 해당한다.

앞으로의 구성은 다음과 같다. 2장에서는 컴포넌트의 명세로 사용하고자 하는 OCL에 대해 살펴보고, 3장에서는 UML/OCL을 이용한 시험 기법과 컴포넌트 기반 소프트웨어의 자동화 시험 환경 구축에 대해 살펴본다. 4장에서는 기업형 컴포넌트의 행위를 분석하고, 이를 반영하는 시험 기법 및 자동화 환경에 대해 제안한다. 5장에서는 결론 및 향후 연구방향을 기술한다.

2. OCL(Object Constraint Language)

UML의 일부분으로 제안된 OCL은 객체 지향 모델이나 모델링의 산출물에 대해 UML으로 표현 불가능한 제약조건(Constraint)을 표현하기 위한 언어이다. 제약조건으로는 ***Invariant***, ***Precondition***, ***Postcondition***이 있다. Invari-

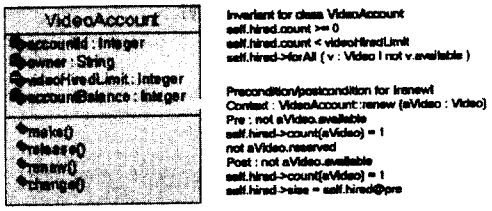

그림 1: Video Account 컴포넌트와 기술된 OCL 일부

ant는 클래스로 인해 생성된 객체가 항상 만족해야 하는 조건을 명시한다. Precondition과 Postcondition은 객체에서 제공하는 오퍼레이션에 대한 제약조건을 기술 할 때 사용되는데 Precondition은 오퍼레이션이 수행되기 위한 조건을 명시하고, Postcondition은 오퍼레이션이 끝난 후의 조건을 명시한다. 오퍼레이션이 올바른 행위를 수행하기 위해서는 오퍼레이션 수행 전에 Precondition이 참이어야 하고, 수행 후에 Postcondition이 반드시 참 값을 가져야 한다[2].
그림 1은 비디오 대여관리의 비디오 계정 컴포넌트에 대한 OCL 명세의 일부분을 보여주고 있다.

3. 관련 연구

3.1 UML/OCL를 이용한 시험 기법

OCL은 assertion형태로 수행중의 값을 분석해야 명시된 명세의 정확성 여부를 알 수 있으므로 제약조건의 진위를 감시(Monitoring)하는 행위 자체가 제약조건을 이용하는 시험 기법의 시험 오라클로 간주된다[3]. 즉 UML/OCL을 이용한 시험 기법들은 제약조건의 감시를 시험 오라클로 이용한다. 실제 여러 프로그램 언어를 기반으로 제약조건의 감시 환경을 구축하는 연구들이 90년대 후반부터 진행되어왔다. 이러한 연구들에서는 제약조건을 소스 코드 내부에 추가로 구현하는 방식을 따르는 특징이 있다[4].
최근에는 데이터베이스 응용프로그램의 분야에 한해서, Invariant를 응용 프로그램에서의 상태가 아닌 데이터베이스에서 감시하는 기법을 사용한다. 즉, Invariant를 데이터베이스의 무결성(Integrity)을 표현하는 제약조건으로 간주하는 방법이다[5].
본 논문에서는 앞선 연구에서와 같이 OCL의 제약조건을 감시하는데, Invariant 뿐만 아니라 오퍼레이션의 Precondition, Postcondition도 데이터베이스를 통해서 검사하는 기법을 사용한다.

3.2 컴포넌트 기반 소프트웨어의 자동화 시험 환경 생성

컴포넌트의 복잡도의 증가로 인해 컴포넌트의 시험 기법을 체계적화 하여 자동화하는 연구가 시도되었다[7]. 일반적으로 컴포넌트 시험은 명세 기반 시험이지만, 기존의 자동화 컴포넌트 연구에서는 크게 소스 코드의 수정 가능 여부에 따라 2가지로 구분한다.
소스 코드의 수정이 가능하면 시험에 필요한 정보를 알아내기 위한 코드를 삽입하고, 시험을 수행하고, 결과를 비교하는 시험 방법이 가능하다[6]. 다만 컴포넌트 개발자에게 소스 코드를 제공받아야 한다는 문제가 발생한다.
소스 코드에 대해 접근할 수 없는 경우에는 내부의 세세한 상태에 대해 알 수 없으므로 외부로 드러난 인터페이스에 의존하여 상태를 감시하거나, 인터페이스를 둘러싸는 중간 계층을 만들어 모니터링 하는 코드를 소스 코드 외부에 추가하여 감시하는 시험 방법이 사용된다[7].

4. 기업형 컴포넌트를 위한 자동화 시험 환경 구축

4.1 3가지 시험 모델의 제안

컴포넌트는 자신 내부의 상황을 외부에 드러내지 않으므로 인터페이스를 통하지 않고는 컴포넌트 내부의 상태나 오퍼레이션의 결과를 알아 낼 수 가 없다.
하지만 기업형 컴포넌트의 경우 그림 2에서와 같이 컴포넌트에서 외부로 제공하는 인터페이스 외에도 데이터베이스와 컴포넌트의 관계를 통해 컴포넌트의 내부의 상태나 오퍼레이션의 결과를 얻을 수 있다.

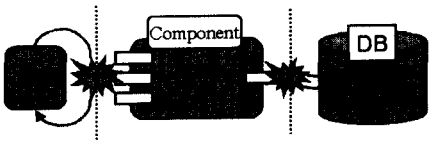

그림 2: 3계층 아키텍처에서 기업형 컴포넌트의 행위

본 논문에서는 컴포넌트의 행위를 시험하기 위해 클라이언트와 컴포넌트 사이의 관계와 컴포넌트와 데이터베이스의 관계를 시험하는 3가지 시험 모델을 제안한다.

- Database-Integrity 모델
  하나의 컴포넌트 내부에 존재 가능한 여러 클래스에 대해 기술 가능한, OCL로 표현된 Invariant를 추출한 모델이다. Invariant들은 SQL 코드로 변환되고, 변환된 코드는 시험 입력을 수행시키는 동안 데이터베이스에게 질의를 통해 제약조건이 올바르게 지켜지는지를 감시하는 데 사용된다.

- Database-Dynamic 모델
  컴포넌트에서 외부로 제공되는 각각의 오퍼레이션에 대해 기술한 Precondition과 Postcondition을 추출한 모델이다. 각각의 제약조건은 상응하는 SQL 코드로 변환되고, 변환된 SQL 코드는 테스트 케이스를 수행하는 동안 오퍼레이션의 수행 직전과 수행 직후에 제약조건이 성립하는지를 확인하는 데 사용된다.

- Return-Result 모델
  테스트 케이스를 수행하고 얻은 결과를 시험하기 위한 모델이다. 주로 Precondition과 Postcondition에서 추출하지만, 데이터베이스를 통한 SQL 질의로는 얻지 못하는 상태이므로 위의 모델과는 구분된다.
  현재는 Symbolic Execution으로 수행 가능한 간단한 제약조건을 추출하여 테스트 케이스의 수행이 완료된 후 얻은 시험 결과를 비교하는 데 사용된다.

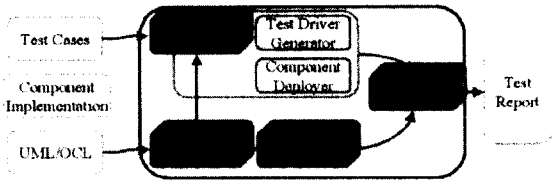

그림 3: 자동화 과정

4.2 자동화 시험 기법을 적용한 시험 환경의 구현

이 절에서는 앞 절에서 제안한 3가지 시험 모델을 이용한 자동화 환경을 기술한다. 자동화 시험 환경은 크게 테스트 케이스의 수행, 시험 모델의 생성, 시험 수행의 감시, 그리고 시험 결과의 비교 4가지로 나눌 수 있다.
그림 3은 제안하는 자동화 환경에 대해 보여주고 있다.

4.2.1 테스트 케이스의 수행

테스트 케이스의 수행은 주어진 테스트 케이스를 수행하는 환경을 구성하는 역할을 수행한다. 테스트 케이스에서 사용하는 오퍼레이션을 제공하는 컴포넌트에 대해 어플리케이션 서버에 배치(Deploy)시켜주는 컴포넌트 배치기능과 실제 오퍼레이션을 호출(Invoke)하는 클라이언트 프로그램을 생성 해 주는 시험 드라이버(Test Driver) 생성기능으로 나눌 수 있다. 배치 기능은 어플리케이션 서버의 종류를 고려해서 어플리케이션 서버와의 통신을 통해 수행되고, 시험 드라이버 생성 기능은 클라이언트 프로그램을 만드는 일정의 템플릿에 따라 Java 클라이언트 코드를 생성한다.

4.2.2 시험 모델의 추출

시험 모델의 추출에서는 제공받는 UML/OCL로부터 시험 모델을 추출한다. 컴포넌트의 명세로 사용하는 UML/OCL로부터 앞에서 제안한 3가지 시험 모델에 따른 OCL을 추출하고, 해당 모델에 따라 SQL 코드로 변환되거나 결과 비교 조건으로 변환된다.

4.2.3 시험 수행의 감시

시험 수행의 감시에서는 모델에 따른 감시 작업을 수행한다. Database-Integrity 모델에서는 데이터베이스의 무결성을 확인하는 SQL 코드를 소유하고 있기 때문에 컴포넌트의 모든 기능을 수행하기 전과, 수행한 후에 항상 검사를 시도한다. Database-Dynamic 모델은 주어진 오퍼레이션에 대해 Precondition과 Postcondition에 해당하는 SQL 코드를 데이터베이스에 수행하여 시험을 수행하는 도중의 상태를 감시한다. Return-Result 모델은 결과에 대한 모델로서 감시 기능이 필요하지 않으므로 시험 수행의 감시에서는 고려하지 않는다.

4.2.4 시험 결과의 비교

시험 결과의 비교는 시험 수행의 감시나 테스트 케이스의 수행의 결과로 나타나는 차이점을 정리한다. *시험 수행의 감시*를 통해 데이터베이스에서 수행된 SQL 코드의 결과와 *테스트 케이스의 수행*의 결과에 대해서 시험자에게 제시해준다. 결과에 대한 정확성 여부는 자동으로 판단하지 않고, 단지 시험자에게 시험의 정확성 여부를 판단할 수 있도록 자료를 제공한다. 시험자는 제공 받는 정보를 토대로 시험의 정확성을 판단하고, 그 결과를 시험 환경에서 관리하도록 한다.

5. 결론 및 향후 연구방향

본 논문은 기업형 컴포넌트의 시험을 위해서 3계층 아키텍처를 기반으로 하는 3가지 시험 모델을 제안하였다. 그리고 제안한 모델을 사용하여 자동으로 시험을 수행하는 시험 환경을 구축하였다.
제안된 모델은 기존의 연구와 달리 컴포넌트와 연계된 데이터베이스 환경을 수용하므로 기업형 컴포넌트의 시험에 적합하고, 구현한 시험 환경은 실제 산업계에서 이용하는 기업형 컴포넌트를 대상으로 하기에 기업형 응용 프로그램의 생산성을 증가시켜 줄 수 있다. 또한 컴포넌트 명세로 UML/OCL을 사용하는 시험 환경이므로 기존의 UML 기반의 시험을 추가하는 것이 가능하다.
현재 자동화 시험 환경은 시험 단계에서의 후반부를 중점적으로 구현한 것에 비해 전반부(Front-End)에 해당하는 테스트 케이스의 설계나 생성에 관한 부분이 생략 되어있다. 향후에는 테스트 케이스의 설계 연구와 생성에 관한 연구를 수행하고, 자동화 환경의 전반부를 추가하고자 한다. 결과적으로 시험 단계 전 과정을 포함하는 자동화 시험 환경을 만들고자 한다.

References

[1] Dolgicer M. "How Will Object Middleware Evolve¿', *Component Strategies*, May 1999

[2] Warmer J.B and Kleppe A. G, *The Object Constraint Language : Precise Modeling with UML* , Addison-Wesley, 1997

[3] Jazéquel J.M. *Object Oriented Software Engineering with Eiffel*, Addison-Wesley, March ,1996

[4] Plosch R. "Design by contract for Python", *Proc. of APSEC '97 and ICSC '97.* , 1997

[5] Demuth B. and Hussmann H. "Using UML/OCL Constraints for Relational Database Design", *Proc. UML '99*, 1999

[6] Bundell G.A., Lee, G., Morris, J., Parker, K. and Peng Lam "A Software Component Verification Tool", *Proc. Int. Conf. on Software Methods and Tools, 2000*, pp.137-146, 2000

[7] Edwards S. H. "A framework for practical, automated black-box testing of component-based software", *Software Testing, Verification and Reliability 2001*, vol. 11, pp. 97-111 , June 2001

[8] Fewster M. and Graham D. *Software Test Automation*, Addison-Wesley, 1999