제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

컴포넌트 계약을 기반으로 하는 컴포넌트 테스팅 기법

박세희*, 이병선* ,진영택*
*한밭대학교
e-mail:aluqard@hotmail.com
{lbs, ytjin}@hanbat.ac.kr

A Component Testing Technique based on Component Contract

Se-Hui Park*, Byung-Sun Lee*,Young-Taek Jin*,
* Hanbat National University

요 약

컴포넌트를 이용하여 작성된 소프트웨어의 테스팅은 컴포넌트의 다양한 특성 때문에 기존의 화잇박스 테스팅 기법을 적용하기 어렵다. 또한 컴포넌트의 통합으로 발생하는 인터액션 테스팅에 더 많은 비중을 둔다. 본 논문에서는 컴포넌트의 기본 및 행위 명세를 나타낸 컴포넌트 계약을 토대로 컴포넌트 테스팅을 수행하기 위한 방법을 제시한다. 컴포넌트 계약은 사전/사후 조건 및 불변 조건을 명시하는 OCL로 작성되며 테스트 케이스 생성과 인터액션 테스팅을 위해 이용된다.

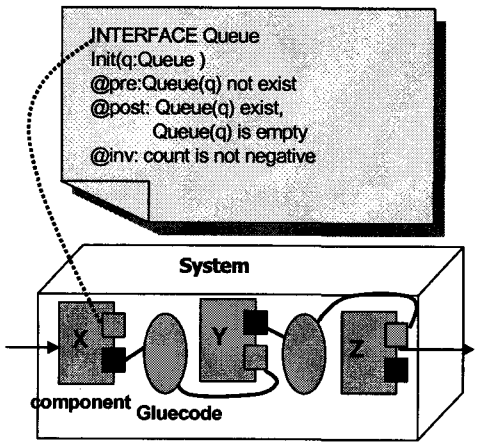
1. 서론

컴포넌트를 기반 소프트웨어 개발은 상용 컴포넌트를 획득하여 조립하므로서 단기간에 걸쳐 소프트웨어 제품을 만들 수 있는 유망한 방법이다. 이러한 개발은 기존의 방법과 달리 많은 노력이 요구 사항, 테스팅과 통합에 놓여지고 상대적으로 적은 노력이 코드 설계에 들어가는 다른 프로세스를 제시한다. 이러한 프로세스에서 소프트웨어 설계는 모듈 내부의 작업 보다는 컴포넌트의 조립 또는 통합과 관련한 아키텍처 문제에 더 중점을 둔다[1]. 컴포넌트의 개발 못지 않게 중요한 분야는 컴포넌트 기반 소프트웨어의 품질 보증을 위한 컴포넌트의 테스팅 분야이다[2]. 컴포넌트의 개발자 관점에서 볼 때 컴포넌트가 소프트웨어가 요구 사항에 맞게 구현되었는지 또는 컴포넌트의 행위와 기능이 적합한 수준에 있는지를 테스팅 할수 있는 방법이 필요하다. 컴포넌트를 구입하여 사용하는 사용자 관점에서는 통합할 컴포넌트의 품질에 대한 인증서와 나중에 합성될 때 컴포넌트가 적절히 동작한다는 것을 테스팅 하기 위한 기법이 요구된다. 특히, 컴포넌트는 소스코드가 유용하지 않은 블랙 박스 형태로 전달이 되기 때문에 단위 테스팅 또는 통합 테스팅을 위해 형식적 또는 비형식적인 형태의 컴포넌트 기능 명세와 컴포넌트 관련 정보를 이용한다. 이러한 컴포넌트의 명세는 컴포넌트의 합성 과정에서 컴포넌트 서비스의 부적절한 사용을 방지할 수 있으며 여러 테스팅 기법에서 다양하게 활용되고 있다[3,4,5]. 본 논문에서는 이러한 명세의 일종인 컴포넌트 계약을 이용하여 컴포넌트를 테스팅하기 위한 기법을 제안한다. 컴포넌트 계약은 컴포넌트에 의해 제공되는 서비스와 그런 서비스를 제공하기 위해 요구되는 환경과 의무를 설정한것으로서 OCL(Object Constraint language)[6]를 이용하여 기능에 대한 사전,사후 조건 및 불변조건으로 명시된다. 이와 같이 소프트웨어 개발 및 태스팅에 Design by Contract[7] 및 분석 계약[8]과 같은 다양한 종류의 계약을 사용하는 아이디어는 다양한 관점과 적용분야를 갖는다. 본 논문에서의 도입은 컴포넌트의 단위 블랙 박스 테스팅을 위해 컴포넌트 계약에 서술된 내용을 이용하여 테스트 케이스를 생성한다. 아울러 바람직 하지 않은 인터액션

및 컴포넌트 통합 이후에 유지되지 않는 여러 특성 검사를 수행하는 컴포넌트 통합 테스팅에 이러한 계약을 적용하고 테스팅을 자동화하는 관점에 초점을 둔다.

2. 컴포넌트 계약

소프트웨어 컴포넌트는 계약상에 명시된 인터페이스와 명백한 배경 종속성을 가진 합성 단위이며 독립적으로 채택될수 있고 합성하기 쉽다[9]. 합성은 컴포넌트를 활용하여 소프트웨어 시스템을 구축하는 핵심 기법이며 주로 인터페이스를 통하여 이루어진다. 인터페이스는 컴포넌트의 서비스를 명시하기 위해 사용되는 오퍼레이션의 집합이다. 컴포넌트의 인터페이스를 통하여 적절한 컴포넌트를 찾고, 기능 및 제약 사항을 이해하게 되므로서 컴포넌트의 합성과 커스터마이징을 이룰 수 있다. 합성은 [그림 1]에서 제시된 바와 같이 연결 중심의 활동이며 정적인 측면과 동적인 측면으로 나눌 수 있다.



[그림 1] 컴포넌트에 의한 시스템의 작성

정적인 합성은 한 컴포넌트가 다른 컴포넌트의 서비스를 필요로 할 때 발생하며 컴포넌트의 제공 및 요청 인터페이스의 연결을 토대로 한다. 동적 합성은 컴포넌트의 인터액션 동안 발생하며 시스템의 비기능적 요구와 관련된다. 즉, 어디에 요구가 놓이고 공유 자원에 대한 동시 액세스를 조정하며 적법한 수행순서를 설정하는 것이 이러한 비기능적 요구로 고려될 수 있다. 이러한 합성 과정에서 개별 컴포넌트에서 발생하지 않은 바람직하지 않은 현상이 발생할 수 있다. 컴포넌트 사이에 바람직하지 않은 인터액션이 발생할 수 있으며 단위 컴포넌트에 유지되었던 특성들이 합성 이후에는 존재하지 않을 수 있다. 따라서 컴포넌트 계약이 요구되며 이를 토대로 컴포넌트들 사이의 인터페이스 불일치와 같은 오류를 테스팅 할 수 있다. 컴포넌트 계약은 내용과 범위에 따라 여러가지 유형이 존재한다. 기본적인 계약 또는 구문 계약(syntactical contract)은 컴포넌트에 대한 기본 정보를 나타내며 컴포넌트가 요청하는 입.출력 파라메터, 컴포넌트에 의해 수행되는 오퍼레이션 및 오퍼레이션의 수행동안 제기될 수 있는 예외조건을 명시한다. 행위 계약은 사전/사후 조건과 불변조건(invariant)으로 오퍼레이션의 행위를 명세한다. 동기화 계약(synchronization contract)은 메소드 호출사이의 동기화로서 객체의 전체 행위를 명세한다. 또한 순서와 병행성과 같은 컴포넌트에 의해 제공되는 서비스 사이의 종속성을 표현한다. 이러한 계약은 형식적 또는 비형식적으로 다양한 형태로 표현된다[10]. 본 연구에서는 기본 내용과 행위 수준의 계약을 채택하고 표현 기법으로 [그림 1]에 제시된 바와 같이 OCL을 채택하였다. 그 이유는 컴포넌트 인터페이스 및 행위를 간결하게 표현할 수 있으며 컴포넌트 계약에 명시된 제약 조건을 실행시간에 검증하고 테스팅하기 위해 OCL을 Java와 같은 언어로 변환시키는 도구[11]를 활용할 수 있기 때문이다.

2.1 단위 컴포넌트 테스팅

컴포넌트 기반 시스템의 테스팅에서 단위 테스팅의 목적은 단위 컴포넌트가 명세에 부합하고 기능적 요구를 충족시키는지 테스트한다. 테스트 케이스의 생성은 카테고리-분할 방법[12]을 이용하고 가능한 테스트 케이스 생성의 자동화를 위해 컴포넌트 계약에 명시된 조건을 이용하였다. 즉, 카테고리의 식별에서 사전/ 사후 조건 및 불변조건을 각각 하나의 카테고리로 설정하고 각 카테고리를 상호 배타적인 초이스로 분할하였다. 그 다음에 테스트될 초이스(choice)의 조합인 테스트 프레임을 개발하고 테스트 프레임을 테스트 케이스로 대응 시켜 입력값을 생성하였다. 결과의 측정 및 비교를 위해 컴포넌트 계약에 명시된 행위 명세의 실행을 위한 방법은 세가지가 고려되었다. 첫째 방법은 OCL로 작성된 컴포넌트 계약을 특정한 구현 언어(예, Java)로 변경하여 컴포넌트와 별도로 실행시키는 방법이다. 두 번째 방법은 테스트 할 컴포넌트에 제약 조건을 해당 컴포넌트가 구현된 언어로 변환한 후 삽입하여 수행시키는 방법이다. 이 방법은 소스코드가 유용하지 않는 컴포넌트의 특성 때문에 고려대상에 제외되었다. 세 번째는 컴포넌트에 제공되는 원래의 오퍼레이션

을 랩퍼 메소드(wrapper method)로 변환하고 이 오퍼레이션에 대한 호출을 랩퍼 메소드에 대한 호출로 재 작성하여 수행하는 방법이다. 본 연구에서는 세 번째 방법을 채택하여 실행 시간에 제약 조건의 테스팅을 수행하였다.

2.2 통합 테스팅

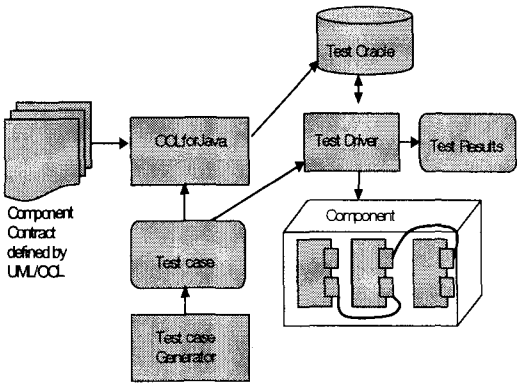
컴포넌트 기반 소프트웨어는 상호 작용하는 여러 컴포넌트들로 이루어지기 때문에 이러한 상호 작용에 대한 테스트가 요청된다. 대부분의 상호 작용 오류는 독립적으로 컴포넌트를 테스팅해서는 밝혀지지 않는 오류를 밝힌다. 가장 기본적인 통합 테스팅은 단위 컴포넌트에 유지되었던 특성이 통합 이후에도 유지되는지를 검사한다. 또한 인터액션이 인터페이스를 통해 발생하므로 인터페이스 내부의 오퍼레이션의 호출 순서 및 다른 인터페이스 사이의 연결에 따라 발생하는 오류를 밝힌다. 한 인터페이스 내의 오퍼레이션의 상호 작용을 테스트하기 위한 테스트 케이스 생성은 컴포넌트 계약에 명시된 사전/사후 조건을 이용하여 항상 만족스러운 결과와 그렇지 않은 결과를 생성하는 호출 순서를 토대로 하여 생성한다. 인터페이스의 사이의 상호 작용은 컴포넌트 A가 컴포넌트 B에 의해 제공되는 서비스를 필요로 할 때 컴포넌트 A가 컴포넌트 B의 메소드를 호출하므로서 상호 작용이 발생한다. 따라서 테스팅은 상향식으로 이루어지며 테스트 케이스의 생성은 다양한 인터액션 즉, 메소드 호출 순서에 따라 다양한 형태로 작성된다,

i) 클라이언트 컴포넌트가 사전 조건을 위반하는 메시지를 서버 클라이언트에 보낸다.
ii) 서버 컴포넌트의 오퍼레이션 실행 순서의 제약조건을 위반하는 메시지를 보낸다.
iii) 오퍼레이션에 잘못된 인수 또는 부정확한 인수값을 제공한다

통합 테스팅의 목적이 컴포넌트가 서로 정확히 상호 작용하고 인터페이싱 할 때 실패를 발생시키는 컴포넌트 오류를 밝히고자 하는 것이다. 따라서 컴포넌트 계약을 이용하여 통합될 컴포넌트 인터페이스 사이의 오류를 밝히는 테스팅을 수행한다.

3. 테스팅 절차

[그림 2]에 제시된 바와 같이 컴포넌트 계약은 OCL 수식으로 컴포넌트 분석 및 설계 시간에 모델링 과정에서 작성된다. 실행 시간에 OCL을 평가하기 위해 OCL이 OCLforJava[11]도구를 활용하여 Java 코드로 변환이 되고 수행된다. 카테고리 분할 방법을 토대로 한 테스트 케이스 생성기는 컴포넌트 계약을 측정하기 위한 OCLforJava 어플리케이션에 대한 입



[그림 2] 컴포넌트 테스팅 환경

력 및 테스트 드라이버에서 사용되는 테스트 케이스를 생성한다. 테스트 드라이버는 이 테스트 케이스를 가지고 변환된 Java 코드 및 컴포넌트를 구동시키고 결과를 평가한다. 사례 연구로서 본 연구에서는 컴포넌트를 이용하여 네트웍 게임 서버를 구축하고 컴포넌트의 통합 테스팅을 실시하였다.

4. 결론 및 향후 과제

컴포넌트를 기반으로 한 소프트웨어 개발은 미리 작성된 컴포넌트를 재사용하여 특정한 요구에 맞게 채택하여 사용한다. 제삼자에 의한 개발된 컴포넌트를 사용하는 경우 각 컴포넌트가 수행할 작업에 대한 신뢰 및 품질이 보증되어야 한다. 또한 컴포넌트가 기존 객체지향 어플리케이션과 통합될 때 테스팅 방법 및 품질 평가를 위한 환경이 요구 된다. 본 연구는 컴포넌트 인터페이스에 주어지는 기본 및 행위 명세인 컴포넌트 계약을 단위 테스팅과 통합 테스팅에 적용하였다. 이러한 컴포넌트 계약은 테스트 케이스의 자동 생성에 활용이 되었으며 제약 조건의 검사를 통해 컴포넌트 기반 소프트웨어의 신뢰성을 확보할 수 있었다. 많은 컴포넌트 서술 언어[13]가 존재하고 OCL의 제한점이 제시됨에도 불구하고 OCL을 채택한 이유는 이 언어의 보편성 및 유용성을 고려하였다. 향후 과제로서 아키텍쳐 명세[14]를 토대로 컴포넌트 상호 작용에 대한 테스트 케이스를 자동으로 생성하고 관리하기 위한 기법이 요청된다.

참고문헌

[1]M.Morisio and C.B.Seaman et al, Investigating and improving a COTS-based software development process, ICSE 2000, pp.31-40.2000.
[2]Jeffrey M. Voas, Certifying off-the-Shelf Software Components, IEEE computer,pp.53-59. June 1998.
[3] Allessandro Orso, Mary Jean Harrold and David Rosenblum. Component metadata for software engineering tasks, proceedings 2nd International Workshop on engineering distributed objects, Nov.2000.
[4] J.Han. An approach to software component specification, proceedings of the 1999 international Workshop on component-based software engineering, May 1999.
[5] J.Han. A comprehensive interface definition framework for software components, proceedings of the 1998 Asia-Pacific software engineering conference, pp.110-117, IEEE Computer society press.
[6] Jos Warmer and Anneke Kleppe,The Object Constraint Language-precise modeling with UML, Addison-wsley ,1999.
[7] Bertrand Meyer,Applying "Design by Contract" ,IEEE Computer,pp.40-51,Oct. 1992.
[8] L.C.Briand and Y.Labiche and H.Sun, Investigating the use of Analysis Contracts to support fault isolation in object oriented Code, Technical Report, Carleton University, Version 4, SCE-01-10 , to appear in Software - Practice and Experience, Wiley, 2003
[9]Clemens Szypersky, Component Software-beyond object-oriented programming, Addison-wsley ,1998.
[10] R.B.Findler et al, Behavioral Contracts and Behavioral Subtyping, Foundations of Software Engineering 2001,
[11]OCLforJava, http://dresden-ocl.sourceforge.net/
[12] Paul Ammann and Jeff Offut, Using formal Methods to derive test frames in Category - partition Testing.
[13] Joseph R.Kiniry, CDL: A component Description Language, COOTS 1999.
.[14] David Garlan, Acme: Architectural description of component-based systems, Foundations of component -based systems, Cambridge University Press, pp.47-67, 2000.