컴포넌트에 대한 IDL 기반 품질평가에 관한 연구

김상영, 황선명
대전대학교 컴퓨터공학과
e-mail : jayusop@zeus.taejon.ac.kr
sunhwang@dragon.taejon.ac.kr

Study on Component Evaluation based on IDL

Sang-Young Kim, Sun-Myung Hwang
Dept of Computer Engineering, Taejon University

요 약

컴포넌트 소프트웨어의 개발은 재사용성의 확보로 인하여 개발기간 단축, 개발비 감소등과 같은 효과를 가져올수 있으나, 컴포넌트를 개발한 환경과 이를 적용하는 환경과의 차이점으로 인한 위험성은 절대적으로 고려해야할 사항 중의 하나이다. 또한 컴포넌트는 실행가능한 코드의 형태로 존재한다는 점 때문에 기존의 Whitebox Testing방법을 적용하기에는 무리가 있다. 따라서 기존의 소프트웨어와 컴포넌트와의 구별되는 요소를 찾아 이제 적용 가능한 테스트 방법의 연구가 요구되어 진다.

본 논문에서는 컴포넌트의 개념과 대하여 살펴보고 기존의 테스트방법을 응용하여 컴포넌트에 적용가능한 테스트 방법을 제안한도록 한다.

1. 서론

지난 수십년간 소프트웨어의 복잡성이 기하급수적으로 증가하면서 위기론이 대두되었고 그에 대한 대안으로 객체지향 개념이 제안되었다. 하지만 이러한 객체지향 소프트웨어도 방대한 양의 일을 적절히 대처하여 급격히 변화해가는 소프트웨어 개발 유지 보수를 감당하기에는 역부족이었다. 실제 개발자들에 의해 객체지향에서 제시하는 재사용성, 대규모 시스템 개발 능력, 확실한 정보 은폐의 능력이 부족함이 대두되면서 컴포넌트에 대한 관심이 대두되었다.

이러한 컴포넌트도 기존의 소프트웨어와 마찬가지로 품질에 대한 중요성은 그냥 지나칠수만은 없는 문제이다. 어느 제품에 있어서도 품질의 중요성은 아무리 강조해도 지나치지 않는다. 소프트웨어 테스팅은 소프트웨어의 품질을 확보하고 결함을 찾아내기 위해 수행되는 일련의 작업이다. 테스팅은 개발된 소프트웨어의 품질에 대한 평가와 품질 향상을 위한 수정 작업들을 포함한다

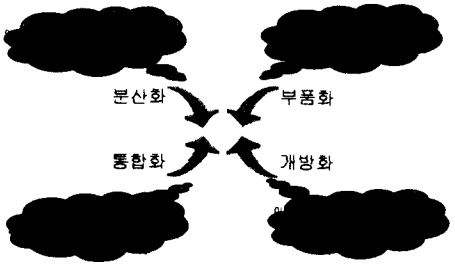
소프트웨어 테스팅에는 개발자는 물론이고 사용자, 독립적인 테스터 등이 참여한다. 소프트웨어에 대한 테스팅은 소프트웨어 품질 보증을 위한 마지막 단계이며, 따라서 품질 보증을 위한 마지막 보루라 할 수 있다. 이 단계에서 수정되지 않는 오류는 사용자가 사용하며 발견되어 이를 수정하는데 이때 많은 비용이 소요된다. 테스팅이 제대로 수행되지 않았을 때 소요되는 비용과 품질 저하, 고객의 불만족 등은 잘 계획되고 철저한 테스팅이 얼마나 중요한 것인가를 설명해주고 있다.

이러한 테스팅에 있어서 크게 두가지의 테스팅 방법을 적용할 수 있게 되는데 이들은 프로그램의 정적 분석을 통한 테스팅 방법(WhiteBox Testing)과 값에 의한 동적 테스팅(BlackBox Testing)으로 나눌 수 있다. 특히 정적 분석을 통한 테스팅 중에는 소프트웨어의 품질을 나타내는 여러 가지의 메트릭스를 사용하여 프로그램의 테스팅을 하기도 하는데 이러한 메트릭스를 소프트웨어 메트릭스(Software Metrics)라고 한다.

본 논문에서는 컴포넌트의 개념에 대하여 살펴보고 실제 응용되고 있는 컴포넌트 아키텍처를 살펴봄과 동시에 적용가능한 품질평가 기준에 대하여 연구하였다.

2. 컴포넌트의 정의

소프트웨어 기술의 발전은 (그림 1)과 같이 발전되어 지고 있으며 이러한 발전현황에 맞게 개발되어지는 것이 컴포넌트이고 컴포넌트의 정의는 아래와 같다.



(그림 1) 소프트웨어 기술의 발전[6]

2.1 Gartner Group

- A runtime software component is a dynamically bindable package of one or more programs managed as unit and accessed through documented interfaces that can be discovered at runtime.

2.2 Clemens Szyperski(Component Software)

- A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to the third-party composition.

2.3 Philippe Krutchen(Rational Software)

- A Component is a nontrivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interface

2.4 '99 ICSE CBSE Workshop

- Executible : 논리적인 모델이 아니라 실제 구동될 수 있도록 만들어진 모듈로서 원칙적으로는 Binary code에 기반을 둔다.
- 3rd-party composable : 독립적으로 보급되며 컴포넌트의 속성이나 메소드의 변경이 가능해야 한다.
- Has explicit interfaces : 명확한 인터페이스가 정의되어져야 하며 세부적인 구현을 숨기고 인터페이스에 의해 접근해야 한다.
- Conforms to a component model : 아키텍처 기반

3. 컴포넌트 아키텍처

컴포넌트 아키텍처는 컴포넌트 상호간 또는 수행환경에 인터페이스(Interface)를 명세하며, Run-time시에 컴포넌트 상호간에 통신할 수 있는 기본 메카니즘을 공급하는 한편 IDL(Interface Definition Language)를 사용해 서로 다른 프로그래밍 언어로 작성된 소프트웨어간 통신을 가능하게 한다.

3.1 COM/DCOM

COM/DCOM은 MS의 모든 제품들의 근간이 되고 있는 기술이다. 특히 분산객체기술의 표준으로 CORBA가 대두되면서 COM/DCOM은 CORBA에 대응하는 MS의 희망이라고까지 불리우고 있으며 윈도우즈라는 운영체제의 대중성에서 COM/DCOM의 가능성은 높게 평가되고 있다. 1997년 초, 마이크로소프트 트랜잭션 서버(MTS)를 NT 플랫폼에 탑재하였으며 이 MTS는 트랜잭션 가능한 COM 컴포넌트를 위한 서버측 컴포넌트 조정자(coordinator)이며 DCOM 단독으로는 다소 부족한 확장성과 컴포넌트 사이에 안정감 있는 조화를 제공한다.

3.2 JavaBeans

Java 언어로 만들어졌으며 Java가 가지고 있는 Platform free, 네트워크, 보안등의 장점들을 그대로 간직하고 있으면서 재사용이 가능한 컴포넌트 모델이다.

3.3 CORBA

CORBA(Common Object Request Broker Architecture)는 OMG(Object Management Group) 컨소시움의 산물물로서 객체버스 내에 존재하고 있는 컴포넌트의 형태와 상호운용방식을 정의한다. 또한 공개 객체 버스를 선택함으로서 컴퓨터 업계에서는 동시에 컴포넌트들을 위한 공개 활동 무대를 제공하고 있다. CORBA는 기존의 객체를 애플리케이션 버스에 가져가기 위한 단일화된 메타포어(Metaphor)로서 객체를 사용한다.

3.4 EJB(Enterprise JavaBeans)

JavaSoft는 일년 이상의 개발을 거쳐 1997년 12월에 EJB(Enterprise JavaBeans) Spec 공개 드래프트를 발표하였는데 EJB는 JavaBeans에 대한 서버 컴포넌트를 정의하고 있다. EJB는 서버에서 수행하는 특화된, Visual 하지 않은 Java Beans이고 JavaBeans와 마찬가지로 새로운 애플리케이션을 생성하기 위해 사용하는 재사용 가능한 컴포넌트이며 원격지로부터 플랫폼에 제한없이 재조립되어질수 있다. EJB 명세서는 서버측 JavaBeans와 새로 나온 컴포넌트 조정자(OTM) 사이의 계약을 정의하며 이 조정자는 서버 측 Beans를 트랜잭션, 상태관리, 자동 활성화와 비활성화를 이용하여 확장 가능한 방법으로 관리한다. 또한 EJB는 100% CORBA와 호환 가능하게 설계되어 있으며 서버 측 컴포넌트와 이들의 컨테이너 사이의 인터페이스를 정의함으로써 CORBA를 확장하였다.

4. 컴포넌트의 품질평가 요소

컴포넌트는 기존의 소프트웨어와는 구별되어지는 몇가지 특성을 가지고 있으며 이러한 품질평가 요소들을 이용하여 개발자들은 컴포넌트에 접근할 수 있으며 품질구성 요소는 (그림 2)와 같다.

컴포넌트 품질 요소	
ISO 9126 요소	정확성(Correctness)
신뢰도(Reliability)	견고성(Robustness)
기능성(Functionality)	인터페이스(Interface)
효율성(Efficiency)	확장성(Extensibility)
유지보수성(Maintainability)	재사용성(Reusability)
이식성(Portability)	호환성(Compatibility)
유용성(Usability)	적시성(Timeliness)

(그림 2) 컴포넌트 품질요소[2][5]

5. 컴포넌트에 적용가능한 테스팅

'99 ICSE CBSE Workshop에서 정의한 컴포넌트의 특징 중 하나인 Executible 특징으로 소스코드가 제공되지 않고 바이너리코드 형태로 존재하는데, 이로 인하여 기존의 메트릭스를 적용할 수가 없게 되지만 본 논문에서는 컴포넌트의 개발자 입장에서 소스코드의 테스팅을 포함한다.

5.1 구조적/객체지향 메트릭스

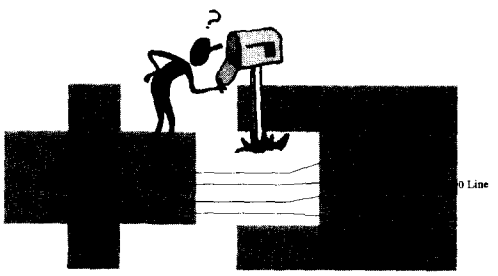
컴포넌트는 OOP에대한 요소를 골고로 갖추어져 있다. 이러한 컴포넌트에 메트릭스를 적용하는데는 아무리 시스템이 객체지향적으로 잘 구성되어져 있는 시스템이라고 할지라도 반듯이 구조적인 부분이 존재하게 된다. 그렇기 때문에 Object-Oriented하게 개발된 프로그램이라도 구조적/객체지향적 메트릭스를 선별적으로 적용해서 적용해야 한다.[8]

5.2 IDL(Interface Definition Language) Testing

Component는 임피던스 미스매치(Impedance Mismatch)를 처리하기 위하여 IDL(Interface Definition Language)을 이용하여 타 프로그래밍 언어에 대한 인터페이스의 정의를 할 수 있도록 되어져 있는데, 컴포넌트의 내부 구조가 아무리 고품질화 되어 있다고 하더라도 IDL이 조잡하게 되어 있으면 컴포넌트에 대한 접근시 문제가 야기될 소지가 있다.

- pctidl(Percent of IDL)

'99 ICSE CBSE Workshop에서 거론된 컴포넌트의 특징 중 하나인 3rd-party composable의 기능을 수행하기 위해서 사용되어지는 것이 IDL(Interface Definition Language)인데 컴포넌트 전체 소스코드에 대한 IDL의 interface 선언문의 비중이 많게 되면 (그림 3)과 같이 기능성이 줄어들고 속이 비어있는(empty) 상태의 컴포넌트로 존재할 가능성이 있다.



(그림 3) 기능이 비어있는 컴포넌트

- 측정방법

$$pctidl = \frac{IDL\text{내에정의된인터페이스선언부에대한소스코드}}{\text{컴포넌트의전체소스코드}} * 100$$

5.3 JARS(Java Applet Rating Service)[3][4]

Sun Microsystems의 자바 개발자 사이트에서는 기 개발된 자바 프로그램에 대한 품질등급을 JARS의 평가결과를 인용하여 사용되어지고 있으며 이는 보통의 자바 Applet뿐만 아니라 컴포넌트 아키텍쳐의 하나인 EJB(Enterprise JavaBeans)에 대하여도 평가되어지고 있다. 등급결정 기준은 (표 1)과 같으며 이에대한 평가 등급은 (표 2)와 같다.

등급 결정 요소	점수
Presentation	200점
AppletPerfect	300점
Functionality/Performance	200점
Originality	300점

(표1) 등급 결정 기준

점수(1000점 만점)	평가등급
990점 이상	Top 1% Web Applet
950점 이상	Top 5% Web Applet
750점 이상	Top 25% Web Applet

(표 2) 평가등급

6. 결론

소프트웨어의 발전은 현재 컴포넌트라는 개념에 이르게 되었고 관련된 연구가 많이 이루어지고 있다. 다른 소프트웨어들과 마찬가지로 컴포넌트의 품질에 대한 관심도 많이 다루어지고 있으나 기존의 소프트웨어처럼 적용 가능한 메트릭스가 있지를 않은 실정이고, 그나마 연구되어지고 있는 내용은 ISO 9126에서 정의한 메트릭스 적용요소에 추가적으로 컴포넌트의 특징적 요소를 적용한 정도이다.

본 논문에서는 아직까지는 컴포넌트에 대한 정의가 내려져 있지 않은 국내 실정에 맞지는 않지만 대표적인 컴포넌트 아키텍쳐상에서 공통적으로 접근해 볼 수 있는 IDL에 대한 메트릭스를 적용하여 보았다.

향후 연구과제로서는 컴포넌트에 대한 국외 동향이나 국내 한국소프트웨어컴포넌트컨소시엄(KCSC)의 표준화 동향을 살펴보고 실제 적용 가능한 품질평가 메트릭스의 개발과 이를 자동화해 적용할 수 있는 지원도구의 설계를 목표로 한다.

참고문헌

[1] Roger S. Pressman "Software Engineering A Practltiners' Approach" 3rd Ed. McGraw Hill

[2] K.-H.Möller & D,J.Paulish "Software Metrics A practitioner's guide to improved prosuct development", Chapman&Hall Co., New York, 1993.

[3] http://www.jars.com/

[4] http://java.sun.com/

[5] ISO/IEC 9126 : Information technology - Software product evaluation - Quality characteristics and guidelines for their use, ISO, 1991

[6] 컴포넌트 기반 소프트웨어 개발을 위한 기술 및 전략, 한국과학재단, 1999

[7] http://www.etnews.co.kr/

[8] 김상영, 정지환, 황선명,"자바 프로그램의 정적 분석을 위한 테스팅 도구에 관한 연구", KCSE-2000, 2000