Object-Z를 이용한 EJB 컴포넌트 정형명세

박기창*, 김병기*
*전남대학교 전산학과
e-mail:kcpark@superse.chonnam.ac.kr

Formal Specification of EJB Component using Object-Z

Ki-Chang Park*, Byung-Ki Kim*
*Dept of Computer Science, Chonnam National University

요약

컴포넌트 명세는 컴포넌트와 클라이언트 사이의 계약 으로 볼 수 있다. 하지만 현재의 컴포넌트 명세는 인터페이스 명세 언어(Interface Definition Language)와 정확성이 결여된 자연어로 작성된 명세에 의존하고 있다. 이러한 명세는 모호성, 불완전성, 모순등을 포함할 수 있다. 따라서 컴포넌트 명세는 이러한 점을 충분히 고려하여 진행되어야 하는데 본 연구에서는 EJB 컴포넌트의 코드 템플릿을 반영하여 요구사항을 Object-Z로 명세하고 이러한 명세로 부터 컴포넌트를 구현하는 단계를 제시하였다.

1. 서론

소프트웨어 규모는 커지고 있으나, 소프트웨어 개발 기술의 발전 속도는 하드웨어 발전 속도에 비해 낙후된 실정이다. 대부분의 소프트웨어 프로젝트들은 개발 기간이 지연되거나 예산을 초과하거나 고객이 원하는 고품질의 소프트웨어를 생산하지 못하고 실패하는 경우가 많다. 이러한 소프트웨어 위기에 대한 대안으로 소프트웨어 재사용 기술이 등장하였다. 대표적인 재사용 기술로 컴포넌트 기반 개발 방식(Component Based Development)이 있는데 이는 소프트웨어를 컴포넌트라는 작은 모듈로 나누어 마치 하드웨어처럼 모듈들을 결합하여 원하는 기능을 하는 소프트웨어를 개발하는 방식이다. 컴포넌트에 대한 일반적인 명세는 IDL(Interface Definition Language)과 자연어로 설명된 메뉴얼, 예제 파일로 배포된다. 이러한 컴포넌트 모듈의 명세는 구문적인 측면은 잘 정의되어 있으나, 의미적인 측면은 자연언어로 기술된 비 정형적인 문서에 의존하고 있다.[1]

즉, 기존의 컴포넌트 명세는 구문적인 측면을 기술하는 IDL의 사용으로 컴포넌트의 구문적인 정합성만을 보장하고 컴포넌트 명세의 의미적인 측면을 보장해 주지는 못한다.[1] 이는 곧 재사용 측면이 강한 컴포넌트 기반 개발 방법에서 모호성, 불완전성, 모순등으로 인한 부작용을 발생 시킬 수 있다. 컴포넌트 기반 개발 방식은 결국 작은 컴포넌트 모듈이 각각의 요구사항을 만족하는 기능을 발휘해야 결국 이러한 컴포넌트의 묶음으로 이루어지는 소프트웨어 시스템도 품질을 인정받을 수 있기 때문에 각 컴포넌트에 대한 기능적인 정확성은 특히 중요하다.

따라서, 본 연구에서는 표준 컴포넌트 아키텍처인EJB(Enterprise Java Beans) 코드 템플릿을 반영하여 Object-Z를 사용해 컴포넌트의 의미적인 측면

※ 이 논문은 정보통신연구진흥원 2001년도 대학 기초 연구 지원사업 지원에 의하여 연구되었음.

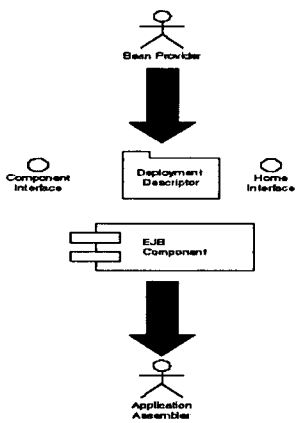
을 명세하고 이러한 명세를 이용하여 EJB 아키텍처를 따르는 컴포넌트로 구현되는 단계를 제시한다.

2. 관련연구

2.1 EJB Component Architecture

EJB는 분산 컴포넌트 환경에서 사용할 수 있는 서버측 컴포넌트 모델이다. EJB 에서는 Java Servlet과 Java Server Page와 같은 Web Component와 Java Class파일의 묶음으로 구성된 Enterprise Bean 의 두 가지 형태의 컴포넌트를 제공한다.

결국 이러한 컴포넌트들은 J2EE(Java2 Enterprise Edition) Applicaiton으로 구성되고 J2EE Server에 배치되어 그 기능을 제공하게 된다. (그림 1)과 같이 실제로 사용자의 요구사항으로 부터 Bean Provider가 해당 컴포넌트를 작성하고 컴포넌트와 관련된 컴포넌트 인터페이스, 홈 인터페이스, 배치 디스크립터, EJB 컴포넌트를 Application Assembler에게 전달하고 Assembler가 조립을 하게 된다.



(그림 1) J2EE Application Assembly and Deployment

배치 디스크립터는 컴포넌트의 정보를 XML 형식으로 작성되고 이는 컴포넌트의 배치 정보를 Assembler에게 전달한다. 문서의 양식은 (그림 2)와 같다. 따라서, Bean Provider는 컴포넌트의 요구사항을 정확히 파악할 필요가 있고 Bean Provider가 컴포넌트를 작성하는데 있어서 컴포넌트의 정형명세를 입력 산출물로 사용할 경우 모호성, 모순, 불완전성 등의 문제점을 해결할 수 있게 된다.

```
<?xml version="1.0" encoding="MS949"?>

<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN"
 "http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">

<ejb-jar>
 <description>no description</description>
 <display-name>CartJAR</display-name>
 <enterprise-beans>
  <session>
   <display-name>CartBean</display-name>
   <ejb-name>CartBean</ejb-name>
   <home>CartHome</home>
   <remote>Cart</remote>
   <ejb-class>CartEJB</ejb-class>
   <session-type>Stateful</session-type>
   <transaction-type>Bean</transaction-type>
  </session>
 </enterprise-beans>
</ejb-jar>
```

(그림 2) Deploy Descriptor

2.2 Object-Z

Object-Z는 모델 기반 정형 명세언어 Z의 객체지향적인 확장언어이다.[2] Object-Z는 일반 개발자들이 비교적 이해하기 쉬운 수학적인 모델을 이용하며, 특히 인터페이스 사이의 상호작용을 명세하는데 적합하다. Object-Z를 이용한 객체지향 사용자 인터페이스 설계[3]와 객체지향 설계와 구현에 관해 연구[4]하는 활동들이 있다. 본 연구에서는 Object-Z를 사용하여 EJB 컴포넌트의 코드템플릿 요소들을 중심으로 요구사항을 명세 한다. 이러한 정형명세는 정확한 해석을 허용함으로써 무엇이 명세되었는지에 대한 논쟁의 가능성이 없어지고, 시스템이 추상적인 용어로 정의되는 것을 허용함으로써 개발자들은 어떻게 성취되는지에 대하여 미궁에 빠지기 전에 자세하게 시스템이 무엇을 하는지를 살펴 볼 수 있다. 또한 추상적인 명세로부터 잘 정의된 규칙들을 이용하는 완정한 명세로의 정제를 허용하기 때문에 정형명세로부터 자동적으로 프로그램의 생성 가능성을 유도할 수 있다.

3. Object-Z를 이용한 EJB 컴포넌트 정형명세

3.1 EJB 컴포넌트 코드 템플릿

Enterprise Bean은 크게 Entity Bean, Session Bean, Message Driven Bean등 3가지로 구분되는데 각각의 Bean 코드는 템플릿을 갖는다. 본 연구에서는 Session Bean의 명세를 예로 사용하며 Session Bean의 코드 템플릿은 (그림 3)과 같다. Bean은 제공해야할 비즈니스 메소드를 실제로 구현하는 자바 클래스로써 명세단계에서 이러한 Bean의 코드 특성을 반영하면 그 만큼 구현과 명세의 간격을 줄일 수 있는 효과가 있다. Object-Z로 명세할 때의 반영할 수 있는 요소로는 (그림 3)에서 < >로 표현된 부분

즉, 빈 클래스의 이름, 비즈니스 메소드 이름, 리턴 타입, 파라미터 그리고 비즈니스 정의 예외가 반영될 수 있다.

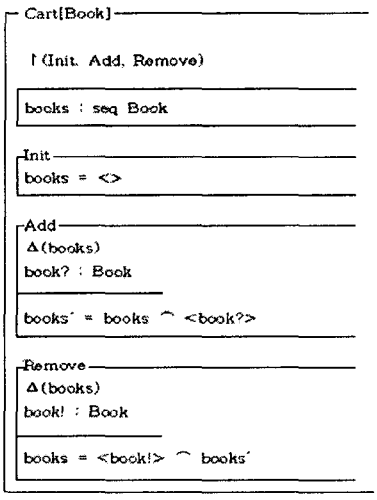
```
Public class <빈 클래스 이름>Bean implements javax.ejb.SessionBean
{
        public <RMI 호환형> <메소드 이름>(<RMI 호환형 파라미터>)
                throws [비즈니스 정의 예외>]... java.rmi.RemoteException
        {
                구현코드
                ...
        }

        public void ejbCreate(){}
        public void ejbRemove(){}
        public void setSessionContext(SessionContext sc){}
        public void ejbActivate(){}
        public void ejbPassivate(){}
}
```

(그림 3) Session Bean 코드 템플릿

3.2 Object-Z를 이용한 정형명세

컴포넌트 개발은 특정 도메인에서 재사용을 고려하여 설계 구현되어야 한다. 본 연구에서는 일반적인 인터넷 서점에서 볼 수 있는 기능, 즉 책을 주문하는 과정에서 장바구니에 추가하는 기능, 장바구니에서 삭제하는 기능을 정형명세로 표현한다.

(그림 3)과 같은 코드 템플릿에서 빈 클래스 이름은 Object-Z의 클래스 이름 표기에 대응하고 비즈니스 메소드 이름은 Object-Z의 인터페이스 이름과 대응 된다. 위에서 열거한 이러한 기능을 Object-Z로 표현하면 아래 (그림 4)와 같다.



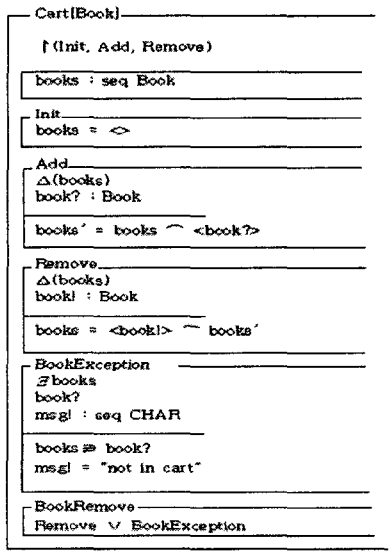
(그림 4) Object-Z를 이용한 초기 명세

스키마 이름은 Cart로 주어지고 입력값은 Book으로 되어있다. 초기에는 주문한 책이 한권도 없으므로 books는 비어있는 상태로 시작하고 add되는 경우에는 기존의 주문한 도서목록에 새로 한 권 추가된 상태로 remove한 경우에는 기존의 목록에서 해당되는 책이 삭제된 상태로 바뀌게 된다.

초기 명세에 장바구니 목록에 존재하지 않는 책을 삭제할 수 없다라는 예외 사항이 있다고 가정하자.그러면 초기의 명세는 다시 확장되고 확장된 명세는 (그림 5)와 같다. 이와 같은 명세를 통해 장바구니 목록에서 책을 삭제하는 연산은

$$BookRemove \cong Remove \; \mathcal{U} \; BookException$$

으로 정의 되며 결국 정상적인 Remove가 발생하거나 BookException과 같은 예외를 발생시키거나 하는 둘 중의 하나의 연산이 실행 됨을 알 수 있다.



(그림 5) 확장된 정형명세

3.3 명세로 부터의 EJB 컴포넌트 모델 작성

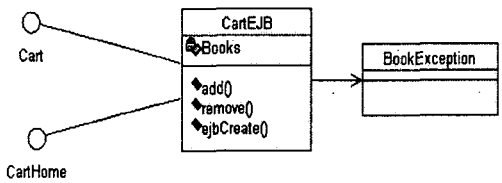
이러한 명세를 바탕으로 실제 컴포넌트 제공자는 컴포넌트를 작성할 때 참조모델로 사용하여 개발할 수 있는데 위의 기능을 제공하기 위한 컴포넌트의 인터페이스는 (그림 6)과 같이 표현되고 컴포넌트의 구조는 아래 (그림 7)과 같은 클래스 다이어그램으로 나타낼 수 있다.

실제 클라이언트는 Cart와 CartHome 인터페이스를 통해 컴포넌트의 비즈니스 메소드를 이용할 수

있다.

```
import java.util.*;
import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Cart extends EJBObject {

   public void add(Book book) throws RemoteException;
   public void remove(Book book) throws BookException, RemoteException;
}
```

(그림 6) 인터페이스 코드



(그림 7) 클래스 다이어 그램

위의 명세를 바탕으로 CartEJB 클래스에는 add(), remove()와 같은 비즈니스 메소드가 구현되어 있고 예외 사항을 처리하기 위한 BookException 클래스가 작성되었다. 결과적으로 도메인에 대한 정형명세를 EJB 컴포넌트로 매핑을 한 형태로 작성이 되었고 컴포넌트가 제공하는 인터페이스는 명세에 표현된 기능을 담당해야 한다. 위의 (그림 7)의 CartEJB 클래스의 경우 add() 메소드는 새로운 book의 title을 입력으로 받아 기존의 목록에 추가하는 기능을 하고 remove()의 경우 기존의 장바구니에서 입력값으로 들어온 title을 갖는 book을 삭제하는 기능을 담당하게 된다. 만일 해당되는 book이 존재하지 않을 경우 BookException 클래스에서 정의된 예외를 발생시킨다. 결국 클라이언트 개발자는 이러한 정형명세를 통해 컴포넌트의 각 인터페이스의 역활을 확실히 인지할 수 있고 Provider는 구현에 참조모델로 이용할 수 있다.

4. 결론

컴포넌트 기반 개발방법에서 정형기법은 도메인 분석을 모델화하여 여러 Provider가 정형기법을 토대로 해당 컴포넌트를 구현하는데 도움을 줄뿐만 아니라 이미 존재하는 컴포넌트를 선정하는 과정에서도 원하는 컴포넌트를 선정하는 모델로 활용할 수 있으며, 제작된 컴포넌트가 요구사항을 만족하는지의 검증에도 사용될 수 있다. 본 연구에서는 여러 정형언어 중 Object-Z를 통해 요구사항을 명세하고, 명세를 바탕으로 EJB SessionBean을 구현하는 예를 보였다.

컴포넌트 기반 개발에서 컴포넌트의 각 기능이 올바로 작동해야 실제로 여러 개의 컴포넌트로 조립된 시스템도 사용자가 원하는 대로 동작을 하기 때문에 그 만큼 신뢰도가 있어야 한다. 단순한 인터페이스 명세나 자연어 명세로는 컴포넌트의 명세가 완전하지 못하고 여기서 정형명세 기법을 도입하여 활용 할 수 있다. Object-Z의 정형명세는 수학적인 표기법을 사용하므로 자연어에 비해 모호성이나, 완전성등이 보완되어 표현된다. 이러한 정형기법은 컴포넌트의 의미적인 명세를 표현하는데 적합하고 오류를 찾거나 검증에도 이용될 수 있다. 추후 이러한 정형명세를 토대로 자동화된 코드 생성기나 컴포넌트 검증에 관한 연구가 필요하다.

참고문헌

[1] 박찬진, 우치수 "컴포넌트 정형명세", 정보처리 제7권 제4호, 2000.

[2] Graeme Smith "The Object-Z Specification Language", Kluwer Academic Publishers, 2000.

[3] Andrew Hussey "Formal Object-Oriented User-Interface Design", IEEE, 2000.

[4] Kihn Nguyen "Bridging the Gap Between Object Oriented Design and Implementation", IEEE

[5] Sun Microsystems "Java™2 Platform Enterprise Edition Specification" Version 1.3, 2001.

[6] Sun Microsystems "Enterprise JavaBeans™ Specification" Version 2.0, 2001.

[7] Ed Roman "Mastering Enterprise JavaBeans", John Wiley & Sons, 1999.

[8] 장종표 "컴포넌트 품질향상을 위한 정형 명세 프로세스", 2002. 2 전남대학교 박사학위 논문