SPIN을 이용한 EJB 적합성 검증
A Verification Method for EJB Architecture

주운기* · 김중배**

* 선문대학교 지식정보산업공학과, E-mail : ugjoo@sunmoon.ac.kr
** 한국전자통신연구원 컴퓨터소프트웨어연구소, E-mail : jjkim@etri.re.kr

Abstract

This paper considers a verification problem on EJB(Enterprise JavaBeans). We select SPIN as a automatic verification tool and consider an instance management specification of CMP(Container Managed-Persistence) entity bean. By showing the verification procedure, we can conclude that SPIN can be used to verify EJB systems.

1. 서론

소프트웨어 부품을 재사용하는 방식의 소프트웨어의 개발은 오래 전부터 연구되고 시도되어 왔으나 최근 들어 컴포넌트 기반의 소프트웨어 개발(Component-Based Development : CBD)방법이 정착되면서 컴포넌트 및 CBD는 21세기 소프트웨어 산업을 이끌 핵심적인 화두로 등장하고 있다. 컴포넌트 모델 및 아키텍처의 기술로는 Sun사의 JavaBeans와 EJB(Enterprise JavaBeans), Microsoft사의 COM/DCOM과 OLE/ActiveX, OMG의 CORBA 2 Specification 등이 있는데, 서버측 컴포넌트 모델인 EJB(Enterprise JavaBeans)는 SUN 사가 1998년에 Release 1.0을 발표한 이래 Release 1.1을 거쳐 Release 2.1을 2002년8월에 발표하였다[6]. 객체지향 언어인 Java를 기반으로 하고 있으며, 엔터프라이스 규모의 대규모 컴포넌트 개발 및 실행을 위한 명세(specification)로 자바의 컴포넌트 모델인 JavaBeans에 서버에서 운영되는 컴포넌트에 필요한 인터페이스 등을 갖추고 있다.

EJB에 대한 명세가 인터넷 등을 통해 공포되어 있지만, 자연어(영어)로 서술되어 있으므로, 서술된 명세 자체가 구현자의 해석 방법 여부에 따라 다르게 구현될 수 있다. 즉, 구현물 들은 개발자 나름대로 표준을 해석하여 구현하는 부분이 있게 마련이므로 많은 해석상의 오류 및 구현 절차상의 오류가 있을 수 있으며 이로 인하여 실제 구현물은 표준대로 구현되지 않을 수 있다. 더구나, 요구하는 기능의 복잡성의 증가 및 많은 선택사항이 있는 명세의 구현 경우는 구현 제품간의 연동성에 문제가 발생할 수 있다([5], [6], [8], [9]).

EJB 컴포넌트 프로그래밍과 같이 복잡한 내부와 인터페이스가 분리되어 있고, 사용자는 단지 인터페이스를 정의 및 수정하여 프로그래밍을 하는 경우는 구현물이 명세(specification)에 따라 정확하게 구현되었는지를 검증하는 과정이 필수적인데, 이와 같이 구현물이 명세에 따라 정확하게 구현되었는지를 검증하는 것을 적합성 검증(conformity test)이라고 한다[1]. 적합성 검증은 요구하는 기능이 복잡해지고 개발 기간 단축, 고 품질의 소프트웨어 수요의 증가에 따라 중요하게 인식되고 있다. 그러나, EJB와 같은 분산객체 처리 방식의 이종 망 이종 장치간 분산 처리 환경에서는 예상하지 못하는 오류가 더 많이 발생할 수 있으므로 적합성 검증이 어렵다. 복잡한 시스템의 검증을 위해서는 자동화 도구의 사용이 바람직한데, 본 연구에서는 자동화 검증도구의 하나인 SPIN(Simple Promela INterpreter)을 이용한 EJB 검증 방안을 다룬다.

2. 명세 기술

SPIN은 분산알고리즘을 시뮬레이션하고 검증하기 위해 Bell Labs의 G.J. Holzmann가 개발한 도구로, 명세언어로는 PROMELA((PROcess MEta LAnguage)를 이용하고, 명세의 성질을 기술위해서 LTL(Linear Temporal Logic)을 이용한다([1], [2], [3], [4]). PROMELA로 기술된 시스템에 대한 각 프로세스의 동작 묘사에서, 통신을 위한 각 프로세스는 FIFO(First-In-First-Out) 통신 채널, 랑데뷰(rendez-vous) 또는 공유변수(shared variables)를 가질 수 있다. SPIN은 유한 채널(bounded channel)을 통한 오토마타를 이용하여 모델을 기술하므로 timed system이나 페트리네트와 같은 무한 상태시스템을 처리하지는 못한다. 그러나, SPIN은 상태 축약(state compression), on-the-fly 검증, 해싱기법(hashing technique) 등의 상태 공간 감소기법을 사용할 수 있으므로 100만개의 상태가 있는 문제에 대해서도 적용이 가능하다. 그리고, SPIN은 실행 결과를 미리 확인할 수 있도록 하기 위해 random/interactive/guided의 시뮬레이션(what-if simulation) 기능을 가지고 있는데, PLTL(Propositional LTL)과 같은 LTL로 기술한 모델의 성질(property)은 그 성질의 만족여부를 모든 도달 상태 조사방식 또는 모든 상태 실행방식으로 검증한다.

명세 S에 대한 구현물 I의 검증은 'I $\Rightarrow$ S'임을 증명함으로서 수행할 수 있다. 여기서 명세 S에 대한 정형화된 기술을 위해 로직을 이용할 수 있고,

정형 명세에 대해서는 모델체커의 하나인 SPIN을 이용하여 명세의 타당성(validation) 검토를 수행할 수 있다. 또는 추가적으로 구현물 I에 대한 로직을 이용한 표기를 하면, 적합성 검증(verification)을 수행할 수 있다.

PROMELA는 다수의 사용자가 정의한 처리 template 또는 proctype 정의와 하나 이상의 process instantiation으로 구성되어있다. template는 다양한 유형의 process 에 대한 동작을 정의하는데, process 수행 시에 정의된 template를 이용하여 비동기 프로세스를 처리하는 구조로 되어 있으며, 이를 위해 message channel, process, variable을 이용하여 다음과 같은 형태로 기술한다.

```
proctype process_name(arguments)
    ( variable 선언부;
      behavior 기술부;
    }
```

proctype으로 시작된 문장은 process_name의 process를 정의한다는 의미이고, 이를 위한 argument는 process 수행을 위한 입/출력 parameter를 말한다.

변수의 유형에는 byte, bit, , proctype, chan, short, int 등이 있다. bit는 단일 비트의 정보를 가진 변수를 표시하고, byte는 C의 unsigned char과 같은 유형의 변수를 말한다. short 및 int도 C의 변수 유형과 같은 형태이다.

동작 선언 부는 보통 'do ~ od'의 하나의 loop로 다음과 같이 구성된다.

```
do
  :: option_11 -> option_12
  :: option_21 -> option_22
     ...
  :: option_n1 --> option_n2
od
```

여기서, 기호 '::'는 각 선택 문장의 시작임을 의미하고, 각 선택문에서의 첫 번째 문장을 'guard'라 한다. 따라서, 실제의 특정 시점에서의 process_name의 동작은 실행가능한 guard에 해당되는 선택문이 수행된다. PROMELA에서의 문장 분리자는 '->'나 ';'를 사용하므로 "option_11 -> option_12"는 "option_11; option_12" 와 같은 역할을 한다. 각 option 문에는 fromA?data(c,d), toB!data(f,g,h) 등과 같이 ?와 !를 사용하여 각각 입력 및 출력을 표시할 수 있다. 즉, fromA?data(c,d)는 fromA에서 data(c,d)를 입력받음을 의미하고, toB!data(f,g,h)는 toB에 data(f,g,h)를 출력함을 의미한다. 또한 제어문을 이용한 기술도 가능한데, if 문은 다음과 같은 구조를 가진다.

```
if
  :: option_k1
  :: option_k2
     ...
  :: option_km
fi
```

검증대상인 성질의 만족되는 지 여부를 확인위해 PROMELA에서는 주장문(assert statement), 타당성 레이블(validation label), 시간청구(temporal claim)와 같은 3가지 방법을 사용할 수 있다.

3. EJB(Enterprise JavaBeans)

3.1 EJB 개요

EBJ란 서버컴퓨터에서 실행되는 자바로 구현된 컴포넌트 모델로, 기업용 애플리케이션을 개발할 때 따라야하는 프로그래밍 규칙(specification)이다. 서버측 컴포넌트 모델인 EJB(Enterprise JavaBeans)는 SUN 사가 제안한 것으로, 2002년 8월 현재 Release 2.1을 발표하였다[6]. EJB 스펙에 의하면, EJB 아키텍처는 객체지향 분산 엔터프라이즈 애플리케이션의 개발 및 분산 배치를 위한 컴포넌트 아키텍처로 정의되어 있다. 엔터프라이즈 자바빈즈 아키텍처를 이용해 만들어진 애플리케이션은 확장성이 있거나 트랜잭션을 보장하며, 다수 사용자 환경에서도 안전하다. 이들 EJB 애플리케이션은 한 번 작성되면 엔터프라이즈 자바빈즈 스펙을 지원하는 어떤 서버 플랫폼에서도 배치되고 운영될 수 있다. 객체지향 언어인 Java를 기반으로 하고 있으며, 엔터프라이스 규모의 대규모 컴포넌트 개발 및 실행을 위한 명세(specification)로 자바의 컴포넌트 모델인 JavaBeans에 서버에서 운영되는 컴포넌트에 필요한 인터페이스 등을 갖추고 있다. EJB 컴포넌트 아키텍춰는 애플리케이션 개발과 배치의 라이프사이클에 있어서 빈 제공자(bean provider), 컨테이너 제공자(container provider), 서버 제공자(server provider), 애플리케이션 조립자(application assembler), 배치자(deployer), 시스템 관리자(system administrator)의 6개의 구별되는 역할을 정의하여, 각 EJB 역할분담자의 생산물이 다른 EJB 아키텍처 역할분담자의 생산물과 호환 가능하도록 보장하는 계약을 지정하고있고, EJB 스펙은 엔터프라이즈 빈의 개발과 배치를 지원하는 계약에 초점을 맞추고 있다.

EJB 애플리케이션은 클라이언트와 서버로 구성되어 있고, 서버는 BJB 컴포넌트, EJB 컨테이너, EJB 서버의 크게 3가지 구성요소로 되어 있다. 여기서, EJB 서버와 EJB 컨테이너는 EJB 컴포넌트를 설치할 수 있는 환경 역할을 한다. 컨테이너에 EJB 컴포넌트가 설치된다면 여러 가지 시스템 레벨의 기능들은 컨테이너에서 알아서 처리해 준다. EJB 컨테이너는 EJB Component의 설치 시 제공되는 Deployment Descriptor을 검사하여 설치되는 EJB 컴포넌트의 작동환경을 설정한다. 또한 클라이언트의 요청에 따라 EJB 컴포넌트를 이용하여 작동하는 기능을 가지고 있다.

클라이언트는 서버컴포넌트들을 호출하여 해당사용자들에게 보여주는 코드로 구성되어 있고, EJB 컴포넌트들을 호출하며 그 결과 값을 받을 수 있다. EJB 컴포넌트들을 호출할 때 EJB 컨테이너를 거쳐야 한다. 일단 실행된 EJB 컴포넌트는 EJB 컨테이너의 관리 하에서 생성, 실행, 소멸의 과정이 수행된다.

EJB 서버는 컨테이너에서 필요로 가능 여러 서비스들을 제공해주는 운영체제의 개념이다. 데이터베이스나 트랜잭션 등의 서비스들을 컨테이너가 요청할 때 제공해 주는 역할을 담당한다. 실제로 SUN에서 발표한 EJB 명세서에서는 컨테이너와 서버의 역할을 명시적으로 분할하고 있지 않다. 이는 EJB 서버 개발 업체에 구현 유연성을 제공하기 위함이다. 실제로 EJB 서버 구현업체, EJB 컨테이너 구현업체 이렇게 따로 있는 경우는 없으며 대부분의 EJB 컨

테이너 구현업체에서 서버까지 같이 제공한다.

일반적으로 EJB 어플리케이션 서버들은 J2EE(Java 2 Enterprise Edition : 엔터프라이즈 환경에 필요한 프로그래밍 API집합)를 지원하며 구체적으로 다음과 같은 기능을 공통적으로 제공한다 :EJB 컴포넌트 배치 및 실행; JDBC, RMI, JNDI 등 EJB 컴포넌트 사용을 위한 기본적인 API 지원; 보안; 동시동작, 패일오버(fail over), 트랜잭션 및 무결성 보장; 멀티쓰레드 안정성 보장; HTTP 지원; 서블릿, JSP 등 웹개발 환경 지원.

EJB 아키텍처는 컴포넌트 기반 분산 컴퓨팅을 위한 아키텍처이다. 엔터프라이즈 빈은 분산 트랜잭션 기반 엔터프라이즈 어플리케이션의 컴포넌트라 할 수 있다. 이와 같이 서버 컴포넌트인 엔터프라이즈 빈(Enterprise Bean)은 실제 비즈니스 로직을 담은 메소드들로 구성되어 있고, Java 언어로 작성된다. 서버에서 동작하는 업무를 수행하며 관련 데이터를 처리하는 소프트웨어 모듈로, 각각의 엔터프라이즈 빈은 리모트 인터페이스(Remote Interface), 홈 인터페이스(Home Interface) 및 빈 클래스(Bean Class)로 (표 1)과 같이 구성되어 있다. 빈 클래스는 빈의 역할에 따라 세션빈(Session Bean)과, 엔티티빈(Entity Bean)으로 나누어진다.

(표 1) EJB 컴포넌트의 구성

리모트 인터페이스	클라이언트가 호출하는 비즈니스 메소드를 정의
홈 인터페이스	EJB 객체의 생성, 식별, 제거 메소드 정의
Deployment Description	빈 클래스를 위한 서비스 설정, EJB 컴포넌트 구조 및 조립 정보

3.2 EJB 정형 검증 방안

정합성 검증은 프로토콜에 필수적인 "일반적인 정확성" 특성을 만족하는지에 대한 프로토콜 사양을 분석하는 것으로, 검증 결과 구현물의 설계가 구문적, 기능적 두 특성을 만족하면 정확한 것으로 간주한다. 일반적으로 정형 검증에서 대상으로 하는 것은 정확성(correctness), 안전성(safeness), 일관성(consistency), 필연성(liveness)의 성질이다. 그러나, EJB 명세는 특정 항목에 대한 설명이 명세서 전반에 걸쳐서 기술되어 있어서 일목요연하게 정리되지 않는 부분이 있고, 엄밀한 정의를 하지 않은 채로 기술된 부분이 있어서 빈이 무엇인지를 정확히 파악하기 어려운 부분이 있고, EJB 구성 요소 설명을 위해 기술되어 있는 처리절차나 코드 부분이 정형화된 규칙을 설명하는 것 보다는 단지 하나의 실례를 설명하는 것에 그치고 있어서 설명된 실현 예의 일반화가 어려운 부분이 있다.

EJB에 대한 기존 연구로는 Sousa와 Garlan[9]이 ADL(Architecture Description Language)의 일종인 Wright를 이용하여 EJB 컴포넌트의 클라이언트와 서버간 관계에 대한 명세(Ver. 1.0)를 기술하였고, 컴포넌트(component)와 커넥터(connector)간 행위(behavior)는 CSP(Communicating Sequential Processes)를 이용하여 정의한 후 모델확인 도구(model checker)로는 FDR(Failure/Divergence Refinement)를 이용하였다.

Nakajima와 Tamai[5]는 모델확인 도구로 SPIN를 위해 BMP 엔티티 빈에 대한 EJB 명세(Ver. 1.1)를 PROMELA를 이용하였고, 명세의 성질(property)을 검증하기 위해서는 LTL(Linear Temporal Logic)을 이용하였다.

3.3 SPIN을 이용한 EJB 검증

SPIN은 인터넷에서 무료로 다운 받아서 사용할 수 있는데, 현재 UNIX 용, LINUX용 및 PC용(Win 95/98/Me/2000/NT/XP)이 있다[4]. SPIN을 통한 EJB의 검증대상으로 CMP(Container Managed-Persistence) 엔티티 빈의 인스턴스 수명주기(EJB Spec.의 Figure 23)을 선택하여, PROMELA로 표현하면 다음과 같다.

```
#define NBUF 1
chan cTe = [NBUF] of short;
chan eTc = [NBUF] of short;
chan excp = [NBUF] of short;

#define newInstance 110
#define setEntityContext 111
#define EntityContext 112
#define ejbActivate 110
#define ejbPassivate 110
#define ejbHome 113
#define ejbCreate 114
#define ejbPostCreat 115
#define BM 112
#define Void 3
#define None 10
#define Pool 20
#define Ready 30

byte state=None;
proctype EJB()
    end:do
        :: (state==None &&
              cTe?[newInstance]) ->
              cTe?setEntityContext; state=Pool
        :: (state==Pool &&
              cTe?[ejbHome])->eTc!Void
        :: (state==Pool &&
             cTe?[ejbCreate]) ->
             cTe?ejbPostCreat;state=Ready
       od
init run EJB()
```

위와 같은 PROMELA 표현에 대해 SPIN을 적용한 결과는 다음과 같다.

```
root@localhost Project]# ./pan
hint: this search is more efficient if pan.c is
compiled -DSAFETY
(Spin Version 4.0.6 -- 29 May 2003)
        + Partial Order Reduction
Full statespace search for:
        never-claim             - (none specified)
        assertion violations    +
        acceptance   cycles     - (not selected)
        invalid endstates       +
State-vector 32 byte, depth reached 1, errors: 0
       2 states, stored
       0 states, matched
       2 transitions (= stored+matched)
       0 atomic steps
hash conflicts: 0 (resolved)
(max size 2^18 states)
1.573   memory usage (Mbyte)
unreached in proctype EJB
        line 27, state 2, "cTe?111"
        line 27, state 3, "state = 20"
        line 28, state 5, "eTc!3"
        line 29, state 7, "cTe?115"
        line 29, state 8, "state = 30"
        line 31, state 12, "-end-"
        (6 of 12 states)
unreached in proctype :init:
        (0 of 2 states)
```

SPIN을 이용하면, 위와 같은 방식으로 EJB의 타 규격에 대해서도 검증을 할 수 있다.

4. 결론

적합성 검증은 어떤 구현물이 해당 명세(표준)에 적합한지 여부를 시장 유통 전에 검토.확인하기 위해 수행하여, 관련자에게 해당 구현물이 관련 표준과 부합되어 상호 운용에 문제가 없이 유통될 수 있다는 근거를 제공한다.

본 연구는 EJB에 대한 적합성 검증에 대한 것으로, SPIN을 통한 규격의 명세화 및 정형 검증을 목적으로 하였다. EJB는 현재 Ver 2.1까지 발표된 상태로, 분산 환경 하에서의 서버 측 컴포넌트 모델에 대한 계속적인 수정/보완 과정을 거치고 있다. 그러나, 자연어(영어)로 서술된 명세 자체가 구현자의 해석 방법 여부에 따라 다르게 구현될 수 있다. 따라서, 서로 일치하지 않는 구현물 들 간에는 분산환경 하에서의 상호 운용성에 심각한 문제가 있을 수 있고, 컴포넌트의 재 사용성이 낮아지게 되므로, EJB의 적합성 검증은 매우 중요하다고 할 수 있다. 본 연구에서는 EJB의 적합성 검증을 위해 검증을 위한 자동화 도구의 하나인 SPIN을 활용하는 예를 보였다. SPIN은 예로 보인 CMP 엔티티 빈 외에도 EJB의 타 규격에 대한 검증 뿐 아니라 유사 소프트웨어의 개발 시에도 SPIN을 유용하게 활용될 수 있을 것이라 기대된다. 그러나, SPIN은 기본적으로 reachability analysis를 통한 검증을 수행하므로, 상태 폭발 문제(state explosion problem)가 발생할 수도 있다는 문제가 있다. 따라서, 향후 정리증명(theorem proofing) 방식을 활용하는 검증 방식에 대한 연구가 필요하다.

참고문헌

[1] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, and P. McKenzie, *Systems and Software Verification : Model-Checking Techniques and Tools*, Springer, 2001.

[2] H. B. Enderton, *A Mathematical Introduction to Logic*, Harcourt Academic press, 2001.

[3] G.J. Holzmann, "The Model Checker SPIN", *IEEE Transactions on Software Engineering*, 1997, Vol23, No.5, pp.279-295.

[4] On-the-fly, LTL Model Checking with SPIN, *http://netlib.bell-labs.com/netlib/spin/whatispin.html*, 2003.

[5] S. Nakajima and T. Tamai, "Behavioural Analysis of the Enterprise JavaBeans Component Architecture", *Lecture Notes in Computer Science(LNCS)*, No.2057, 2001, pp.163-182.

[6] Enterprise JavaBeans(TM) Specification, *http://java.sun.com/products/ejb*, 2003.

[7] K.W. Lee and P. Krishnan, *Towards a Formal Framework for JavaBean and Enterprise JaveBeans*, Reports in Computer Science, University of Canterbury, 2001.

[8] R. Silaghi and A.Strohmeier, "Critical Evaluation of the EJB Transaction Model", *LNCS* 2604, 2003, pp.15-28.

[9] J.P. Sousa and D. Garlan, "Formal Modeling of Enterprise JavaBeans Component Integration Framework", *Information and Software Technology*, 2001, Vol.43, pp.171-188.