IE Interfaces
Vol. 17, No. 2, pp. 149-157, June 2004.

컴포넌트 소프트웨어를 위한 적합성 검증 방법

주운기[1†] · 김중배[2]

[1]선문대학교 지식정보산업공학과 / [2]한국전자통신연구원 지능형로봇연구단

A Conformance Test Procedure for the Enterprise JavaBeans

Un Gi Joo[1] · Joong-Bae Kim[2]

[1]Department of Knowledge and Industrial Engineering, Sunmoon University, Asan, 336-708
[2]Intelligent Robot Research Division, ETRI, Deajeon, 305-350

This paper considers a conformity testing problem on EJB(Enterprise JavaBeans). The EJB architecture is a component architecture for the development and deployment of component-based distributed business applications. The objective is to find an optimal test sequence for the conformity test between the EJB specification and an implemented one. For the test sequence, we formulate the problem as a rural Chinese postman tour one and use a linear programming formulation. Based upon the formulations, we suggest a conformance test procedure and show its efficiency by applying the procedure to the CMP(Container-Managed persistency) entity bean of the EJB.

Keywords: conformance test, Enterprise JavaBeans, UIO, rural Chinese postman tour

1. 서 론

소프트웨어 재사용에 대한 연구는 오래 전부터 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. 컴포넌트는 특정기능을 수행하기 위해 독립적으로 개발되고 잘 정의된 인터페이스를 가지며 다른 컴포넌트와 조립되어 응용시스템을 구축하기 위해 사용되는 소프트웨어의 단위를 뜻한다. 특히, 컴포넌트는 언제, 어디서나, 누구나 필요한 정보를 쉽게 얻을 수 있는 인터넷 환경이 보편화되면서 인터넷 상에서 다양한 소프트웨어 부품을 "Plug and Play" 형태로 조립하여 사용할 수 있게 됨에 따라 컴포넌트 기반 소프트웨어의 개발추세가 가속화되고 있다.

컴포넌트 모델 및 아키텍처의 기술로는 Sun 사의 JavaBeans와 EJB(Enterprise JavaBeans), Microsoft 사의 COM/DCOM과 OLE/ActiveX, OMG의 CORBA 2 Specification 등이 있다. 본 논문은 Sun 사의 EJB 명세에 대한 국내 E연구소 구현제품(Kim *et al.*, 2002)의 적합성 검증방법을 제시한다. 소프트웨어 개발을 위해서는 요구사항 분석을 통해 명세(specification)를 정의한 후에 이에 대한 타당성을 검토(validation)하고 명세에 따라 구현하며, 구현된 제품에 대해 적합성 검사(verification)를 실시한다. 명세를 정의한 후에 검토하는 단계인 타당성 확인(validation)은 요구사항이 명세에 잘 반영이 되었는지를 확인하는 과정이고, 적합성 검증(verification)은 소프트웨어가 명세에 맞게 구현되어 있는지를 검사하는 과정이다(Holzmann, 1991; Hailpern and Santhanam, 2002). EJB 명세는 자연어(영어)로 기술되어 있고, 특정 항목에 대한 설명이 명세서 전반에 걸쳐서 기술되어 있어서 일목요연하게 정리되지 않는 부분이 있으며, 엄밀한 정의를 하지 않은 채로 기술된 EJB 명세 부분은 정확한 의미를 파악하기 어렵게 한다. 그리고, EJB 구성요소에 대한 설명을 위해 기술되어 있는 처리절차나 코드 부분이 정형화된 규칙을 설명하는 것보다는 단지 하나의 실례를 설명하는 것에 그치고 있어서 설명된 실현 예의 일반화가 어려운 부분이 있다(Lee and

†연락저자 : 주운기 교수, 336-708 충남 아산시 탕정면 갈산리 선문대학교 지식정보산업공학과, Fax : 041-530-2926,
E-mail : ugjoo@sunmoon.ac.kr
2003년 3월 3일 접수, 1회 수정 후 2004년 3월 4일 게재 확정.

Krishnam, 2001; Sun Microsystems, 2003). 따라서, 명세와 구현제품 간에는 불일치가 발생할 수 있는데, EJB와 같은 분산형 컴퓨팅 환경을 운용환경으로 하는 경우는 다양한 이기종 간 통신이 원활하게 이루어질 수 있어야 하므로, 명세에 따라 각각 구현한 제품 간 원활한 연동을 위해서도 적합성 검증은 매우 중요하다.

EJB의 검증에 대한 기존 연구로는 Sousa and Garlan(2001)이 ADL(Architecture Description Language)의 일종인 Wright를 이용하여 EJB 컴포넌트의 클라이언트와 서버 간 관계에 대한 명세 Ver. 1.0을 기술하였고, 컴포넌트(component)와 커넥터(connector) 간 행위(behavior)에 대해 CSP(Communicating Sequential Processes)를 이용하여 정의한 후 모델확인 도구(model checker)로 FDR (Failure/Divergence Refinement)를 이용하였다. Nakajima and Tamai(2001)는 모델 확인도구의 일종인 SPIN를 위해 BMP(Bean Managed-Persistence) 엔티티 빈에 대한 EJB 명세 Ver. 1.1을 PROMELA(PROcess MEta LAnguage)와 LTL(Linear Temporal Logic)을 이용하여 기술하였다. 이 외에 Lee and Krishnam(2001)은 EJB 명세 Ver.1.1에 대해 정리증명기(theorem prover)의 일종인 PVS (Prototype Verification System)을 이용하여 검증할 수 있음을 보였다. 그러나, 이들 연구는 모두 특정 검증 도구를 EJB의 타당성(validation) 검증에도 활용할 수 있음을 보이는 데에 그치고 있고, 이들 방법을 활용하기 위해서는 각 검증도구가 필요하고 이에 대한 활용법을 익혀야 한다는 문제도 있다. 또한, EJB 컴포넌트 프로그래밍과 같이 내부와 인터페이스가 분리되어 있고, 사용자는 단지 인터페이스를 정의 또는 수정하여 프로그래밍을 하는 경우는 구현물이 명세(specification)에 따라 정확하게 구현되었는지를 검증하는 적합성 검증(conformity test)이 필수적이다.

본 논문은 EJB의 적합성 검증을 위한 최적의 시험열(test sequence) 발생방법을 제시한 것으로, 여기서 제안하는 시험열 생성방법을 이용하면, 특별한 검증도구가 없더라도 EJB 구현물의 적합성 검증을 효율적으로 수행할 수 있다. 시험열을 발생시키는 방법으로는 Uyar and Dahbura(1986)가 시스템의 각 상태(state)를 확인할 수 있는 수단과 시스템의 초기 상태로 언제든지 되돌릴 수 있는 기능이 있는 시스템의 경우에 대한 시험열 생성방안을 제시하고, 이 방법을 ISDN(Integrated Service Digital Network) Q.931 프로토콜의 시험에 적용하였다. Aho *et al.*(1991)는 시스템의 상태 확인이나 초기 상태로 즉시 되돌릴 수 있는 기능이 없는 시스템을 위한 시험열 생성방안을 제시하였는데, 여기에서는 시험열 생성을 위해 minimum-cost maximal flow 문제의 해법을 이용하여 ISDN Q.931 프로토콜의 시험을 위한 시험열을 구성하였다. 그리고, Csondes *et al.*(2002)는 적합성 검증을 위한 시험열의 일부를 효율적으로 구하기 위한 유전(genetic) 알고리즘을 제시하였다. 본 논문은 EJB 명세에 따른 적합성 검증을 다룬 것으로, Aho *et al.*와 마찬가지로 시스템의 상태 확인이나 초기 상태로 즉시 되돌릴 수 있는 기능이 없는 시스템에 대해서도 적용 가능하도록 UIO(Unique Input-Output) 시험열을 이용하였지만, EJB 적합성 검증을 위한 시험열을 선형계획법(Linear programming) 모형을 통해서 구한다. 여기서 제안하는 선형계획법 모형은 Aho *et al.*의 minimal- cost maximal flow 모형을 이용하는 것보다 효율적으로 EJB에 대한 적합성 검증을 할 수 있다. 본 논문의 구성은 다음과 같다. 제2장에서는 EJB의 구조 및 특징을 기술하였고, 제3장에서는 EJB 적합성 검증방법을 제시하고, 제안한 시험열 발생방법의 유용성을 보이기 위해 EJB의 CMP(Container-Managed Persistence) 엔티티(entity) 빈에 적용한 예를 보였다. 마지막으로, 제4장에서는 결론 및 추후 과제를 기술하였다.

2. EJB

컴포넌트 기반개발(CBD)은 소프트웨어 개발방법의 가장 최근의 패러다임으로, 컴포넌트의 오퍼레이션 및 상호 오퍼레이션을 정의하는 명세(specification)의 개발, 객체나 컴포넌트들로부터 컴포넌트의 구축(construction), 컴포넌트들을 이용해 애플리케이션을 조립(assembly)하는 등의 활동을 필요로 한다. CBD에서 컴포넌트들은 외부에 제공하는 자신만의 서비스와 컴포넌트로서 꼭 가져야 하는 서비스 등으로 이루어져 있는 소프트웨어 단위로, 컴포넌트들 간에 서로 직접 통신할 수도 있지만 컴포넌트 아키텍처 시스템에 플러그인(plug-in)되어 동작하는 것이 더 효과적이다. 즉, 컴포넌트들을 직접 연결하지 않고 EJB 아키텍처와 같은 컴포넌트 아키텍처를 통해 연결하면, 컴포넌트 간 연관성이 작아지므로 컴포넌트들을 외부에 배포하여 재사용하기가 용이해지고, 코드의 변경, 추가, 삭제가 쉽다는 장점이 있다. 이를 위해서 컴포넌트들은 외부 환경인 컴포넌트 아키텍처에 플러그인할 수 있는 잘 정의된 인터페이스(interface)에 대한 명세가 필요하지만, 컴포넌트 기능의 구현은 사용자에게 드러나지 않아도 된다. 이와 같이 컴포넌트의 명세와 구현의 분리는 애플리케이션 개발의 신속성, 용이성, 플랫폼 독립성 등을 제공함으로써 개발비용의 절감, 제품의 시장진입의 신속성, 생산성 향상 및 품질 향상을 기할 수 있기 때문에 CBD가 소프트웨어 위기를 극복하는 방법으로 최근 관심이 고조되고 있다.

2.1 EJB 구조

EJB(Enterprise JavaBeans)는 엔터프라이스 애플리케이션(Enterprise Application)의 일종으로, Sun 사가 개발한 웹 응용 서버 규격인 J2EE(Java 2 Enterprise Edition)의 비즈니스 로직을 컴포넌트 형태로 작성한 서버측 컴포넌트 프로그래밍 모델이다 (Sun Microsystems, 2003). EJB 명세는 Ver1.0을 1998년 3월에 발표한 이래로, 1999년 12월에 EJB Ver1.1 Final Release, 2001년 4월에 EJB Ver2.0 Proposed Final Draft, 2001년 8월에는 27개의 장과 5개의 부록으로 구성된 총 572 페이지 분량의 EJB Ver.2.0 Final

Release을 인터넷에 공포하였고, 현재에도 Sun 사에서는 새로운 기능 및 명세 보완에 대한 내용을 수시로 발표하고 있다. EJB 아키텍처는 객체지향 분산 엔터프라이즈 애플리케이션의 개발 및 분산배치를 위한 컴포넌트 아키텍처로, EJB 아키텍처를 이용해 구축한 애플리케이션은 확장성과 트랜잭션을 보장하며, 다수의 대규모 분산 사용자 환경에서도 안전하다는 장점을 갖는다.

EJB 애플리케이션은 <그림 1>과 같이 클라이언트(client)와 서버(server)로 구성되어 있고, 서버는 BJB 컴포넌트, EJB 컨테이너(container), EJB 서버의 크게 3가지 구성요소로 되어 있다.

클라이언트는 서버 컴포넌트들을 호출하여 해당 사용자들에게 보여주는 코드로 구성되어 있고, EJB 컴포넌트들을 호출하며 그 결과 값을 받을 수 있다. EJB 서버는 컨테이너에서 필요로 하는 여러 서비스들을 제공해 주는 운영체제 개념의 것으로, 데이터베이스나 트랜잭션 등의 서비스들을 컨테이너가 요청할 때 제공해 주는 역할을 한다. 서버의 컴포넌트인 엔터프라이즈 빈(Enterprise Bean)은 실제 비즈니스 로직을 담은 메소드(method)들로 구성되어 있고, Java 언어로 작성된다. 서버에서 동작하는 업무를 수행하며 관련 데이터를 처리하는 소프트웨어 모듈로, 각각의 엔터프라이즈 빈은 원격 인터페이스(Remote Interface), 홈 인터페이스(Home Interface) 및 빈 클래스(Bean Class)로 구성되어 있다.

원격 인터페이스는 빈이 어떤 일을 수행하기 위해서 외부 세계에 제공하는 빈의 비즈니스 메소드를 정의하는 역할을 하며, EJB 컨테이너에 의해서 자동으로 구현된다. 홈 인터페이스는 새로운 빈을 생성하고, 제거하고, 찾아내는 등의 빈의 사이클 메소드를 정의한다. 원격 인터페이스와 마찬가지로 EJB 컨테이너에 의해서 자동으로 구현되게 된다.

EJB 컴포넌트들을 호출할 때 EJB 컨테이너를 거쳐야 한다. 일단 실행된 EJB 컴포넌트는 EJB 컨테이너의 관리하에서 생성, 실행, 소멸의 과정이 수행된다. EJB 서버 및 EJB 컨테이너는 EJB 컴포넌트를 설치할 수 있는 환경 역할을 한다. 컨테이너에 EJB 컴포넌트가 설치되면 보안, 트랜잭션, 안전성 등의 시스템 레벨의 기능을 컨테이너에서 처리하고 관리해 주므로, 프로그래머의 작업량 단축, 코드 재사용성의 증가로 인한 코드 관리용이, 시장진입 시간단축의 효과가 있고, 프로그래밍 전문가의 필요성이 감소되는 이점이 있다.

빈 클래스(Bean Class)는 실제로 빈의 비즈니스 메소드를 구현하는 것으로, 빈 클래스는 반드시 원격 인터페이스에 정의된 메소드들과 대응하는 메소드들을 가져야 한다. 빈 클래스는 빈의 역할에 따라 세션 빈(Session Bean)과 엔티티 빈(Entity Bean)으로 나누어진다. 세션 빈(Session Bean)은 비즈니스 프로세스를 나타내는 컴포넌트이고, 엔티티 빈(Entity Bean)은 비즈니스 프로세스가 사용하는 데이터들의 집합으로, 영구적으로 저장되어야 할 데이터베이스의 테이블을 나타낸다. 엔티티 빈은 엔티티 빈 내의 필드 관리는 누가 맡게 되느냐에 따라 Contai- nerManaged Persistence(CMP) Entity Bean과 Bean-Managed Persis- tence(BMP) Entity Bean의 두 가지로 나눌 수 있다. CMP 엔티티 빈은 개발자가 프로그래밍을 하지 않더라도 컨테이너가 직접 엔티티 빈의 필드 관리를 맡는다. 클라이언트가 특정 엔티티 빈을 탐색하는 작업을 수행한다면 서버측에서 이를 데이터베이스에서 검색하여 엔티티 빈의 필드로 값을 전달하고 클라이언트로 이를 리턴하게 되는데, 이 데이터베이스와의 연동작업을 컨테이너가 알아서 해준다. 반면에 BMP 엔티티 빈은 직접 이 코드를 작성해 줘야 한다.

2.2 인스턴스 관리기능 모듈

2.2.1 EJB 규격

국내의 E연구소에서는 EJB 규격 Ver 2.0에 따른 EJB 서버 시스템을 개발하였고, 그 중의 한 기능 모듈인 CMP 엔티티 빈의 인스턴스(instance) 관리기능은 EJB 서버에서의 원활한 기능수행을 위한 기본적인 기능 중 하나이다. EJB 규격 Ver 2.0의 CMP(Container Managed-Persistence) 엔티티 빈에 대한 빈 인스턴스의 생성에서 삭제까지의 상태도는 <그림 2>와 같다.

엔티티 빈은 다음 세 가지 상태 중에서 하나의 상태를 가지게 된다: 존재하지 않는 상태(does not exist); pool에 저장되어 있는 상태(pooled); 준비상태(ready). 여기서, '존재하지 않는 상태'는 엔티티 빈 인스턴스가 아직 생성되지 않은 상태이고, 'pool에 저장되어 있는 상태'는 엔티티 빈 인스턴스가 아직 엔티티

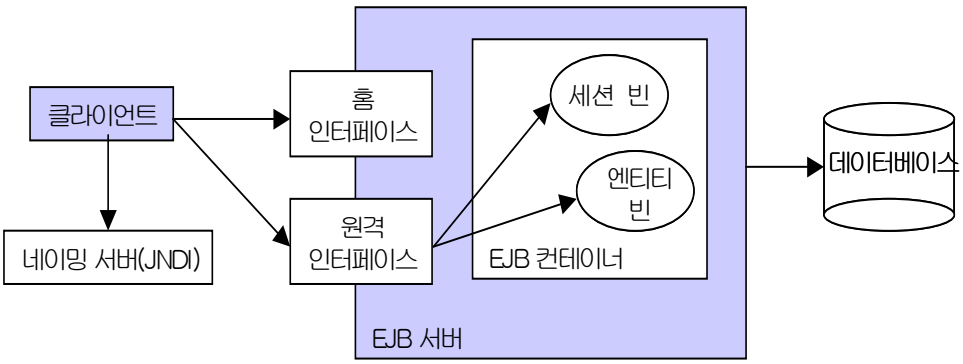

그림 1. EJB 구조.

객체 식별성(identity)을 가지지 않은 상태를 의미하며, '준비상태'는 엔티티 개체 식별성이 할당되어서 메소드 호출을 기다리는 상태를 나타낸다. CMP 엔티티 빈은 <그림 2>와 같이 각각의 상태에서 컨테이너로부터 특정한 입력을 받으면 이에 해당하는 처리를 한 후 상태천이를 한다. 예를 들면, 엔티티 빈 인스턴스는 컨테이너가 엔티티 빈의 newInstance( ) 메소드를 호출하면서 생성된다. 다음에는 엔티티 빈의 setEntityContext( ) 메소드를 호출해서 이미 만들어진 EntityContext를 엔티티 빈에 할당한다. 이 과정을 거친 엔티티 빈 인스턴스는 'pool에 저장된 상태'로 들어가게 된다. 각 엔티티 빈의 자신만의 풀(poll)을 가지고 있으며, 풀에 들어 있는 엔티티 빈 인스턴스는 아직 사용되지 않은 상태이고, 풀에 속한 인스턴스들은 모두 동일한 것으로 간주한다. 풀된 상태에서 컨테이너는 엔티티 빈의 finder 메소드(ejbFind<METHOD>)나 엔티티 빈의 홈 메소드(ejbHome<METHOD>)를 호출할 수 있다. 이 상태에서 컨테이너는 풀에 포함된 엔티티 빈 인스턴스 중에서 하나를 선택해서 클라이언트의 요청을 처리하도록 한다.

'pool에 저장된 상태'에서 '준비상태'로의 전이는 ejbCreate METHOD>( )와 ejbPostCreate<METHOD>( ), 또는 ejbActivate( )의 호출에 의해 수행된다. 인스턴스가 '준비상태'에 있는 경우 인스턴스는 엔티티 객체에 대한 식별성을 가지게 된다. 또한 컨테이너는 엔티티 빈 인스턴스와 데이터베이스에 있는 내용과 일치시키기(synchronize) 위해서 ejbLoad( )와 ejbStore( ) 메소드를 여러 번 호출할 수 있다. ejbLoad( )는 데이터베이스의 내용을 읽어서 엔티티 빈의 멤버 필드에 값을 할당하기 위한 메소드이고, ejbStore( )는 엔티티 빈의 내용을 데이터베이스에 저장하기 위한 메소드이다. '준비상태'에서는 비즈니스(business) 메소드들이 여러 번 호출될 수 있고, 비즈니스 메소드에서 필요한 경우에 ejbSelect <METHOD>( ) 메소드가 호출될 수 있다.

2.2.2 구현제품

E 연구소에서 구현한 EJB 서버는 현재 상용화를 준비하는 단계에 있으나, 이에 대한 상세설명서나 소스 파일 설명서가 없는 상태여서 내부의 처리절차 분석에는 어려움이 있으므로, 본 논문에서는 black-box 시험방식을 택하였다. 구현한 EJB의 인스턴스 관리기능은 인스턴스 풀, 인스턴스 캐쉬, 재생기(recycler), 생명주기 관리자의 기능 모듈로 구성되어 있다(Kim *et al*., 2002). 인스턴스 풀은 BJB 빈 인스턴스를 일정 개수만큼 미리 생성하여 클라이언트가 인스턴스를 다 사용하면 다른 클라이언트가 해당 인스턴스를 재사용할 수 있도록 보관하는 기능을 수행한다. 인스턴스 캐쉬는 현재 사용중인 EJB 빈 인스턴스를 보관하는 기능을 수행하고, 재생기는 일정 시간마다 타임 아웃된 빈을 검사하여 타임 아웃된 빈을 메모리에서 제거하는 역할을 한다. 그리고 생명주기 관리자는 ejbActivate( ), ejbPassivate( ), ejbLoad( ), ejbStore( ) 등과 같은 요청을 빈에게 전달하는 기능을 수행하도록 구현하였다.

3. EJB 적합성 검증

적합성시험은 구현물(I; Implementation)이 명세(S; Specification)에 합당하게 구현되었는지를 시험하는 것으로, 주어진 명세 S에서 시험 케이스(test case)를 생성하고, 이를 구현물 I에 적용한 결과가 명세 S의 결과와 일치하는지를 검증하기 위한 모델링 도구로 유한상태기계(FSM; Finite State Machine)를 사용할 수 있다. FSM은 명세의 제어부분을 유한 개의 상태(state)와 상태 간

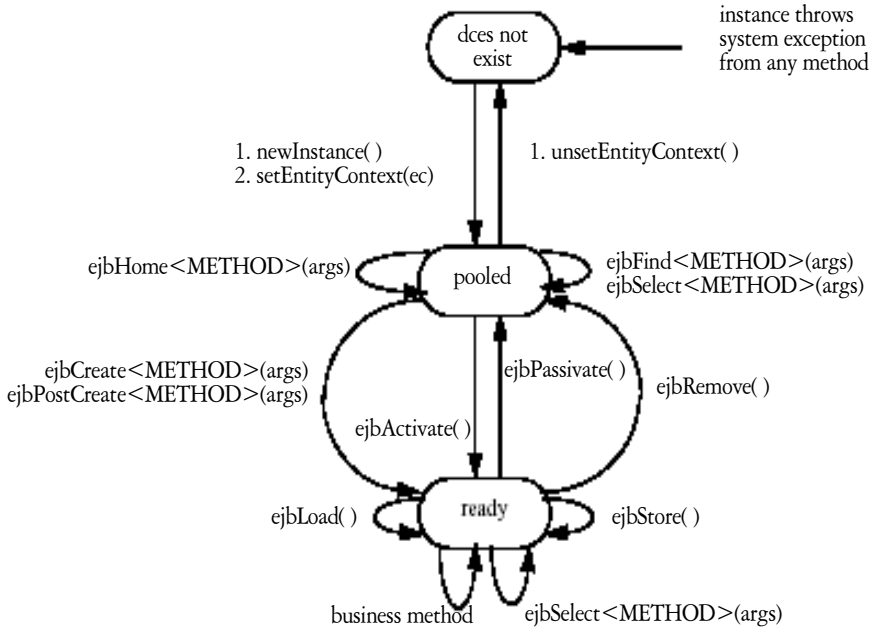

그림 2. CMP 엔티티 빈의 상태도.

천이(state transition)로 표현하는 것으로, 상태(state)는 프로세스가 진행되는 동작과정중에 발생하는 특정의 안정된 경우를 나타내고, 상태천이는 프로세스가 허용하는 동작(action), 일어나기를 기대하는 사건(event), 사건에 대응하기 위한 방법 등으로 정의한다. FSM은 표현하려는 대상이 유한 개의 상태를 가져야 한다는 제약이 있지만, 간결하고 이해가 쉬우며 사용이 편리한 장점이 있으므로(Holzmann, 1991), EJB 규격 기술을 위해서도 유용한 모델이다.

EJB 명세는 공개되어 있지만, 이에 따른 구현제품의 소스 파일은 비공개되는 것이 일반적이다. E 연구소의 구현제품의 경우에도 구현제품에 대한 상세설계 및 구현 설계서가 없는 상태이므로, black-box로 간주한 구현물이 명세에 일치하는지를 결정하기 위해 사용되는 시험 스위트 생성을 위한 검증절차를 다음과 같이 구성하였다.

검증절차

단계 1: 명세에 대한 FSM S와 구현물에 대한 FSM I를 작성

단계 2: FSM S와 FSM I 사이에 적합성 관계 정의

단계 3: FSM S로부터 각 상태 S_k에서 시험계열 TS_i 적용 시, 기대되어지는 시험계열 출력 TS_0 및 천이상태 S_l 정보를 도출

단계 4: TS_i를 black-box로 취급되는 구현물 FSM I의 입력부분에 적용

단계 5: FSM I의 출력부분에서 실제 출력부분 TS_a를 관찰

단계 6: FSM S와 FSM I의 적합성을 결정하기 위하여 TS_0와 TS_a 및 천이 후 상태를 비교하여 적합성을 판정

시험계열(test case)은 입력과 입력에 따른 출력값들의 한정된 길이의 시험열을 말하는데, black-box인 구현 FSM I의 임의의 상태 I_k에 TS의 입력 TS_i를 적용한 후의 출력값 TS_a이 기대되는 출력값 TS_0와 같고 기대되는 상태로 천이하였는지를 검증절차의 단계 4~6과 같이 확인함으로써 적합성을 검증할 수 있다. 그러나, 검증절차의 단계 4와 단계 5의 적합성 시험단계에는 일반적으로 두 가지의 어려움이 존재한다. 이는, 단계 4에서 구현 FSM I의 상태를 상태 I_k로 위치시키기 어렵다는 문제가 있고, 두 번째는 단계 5에서 입력 시험열을 적용한 후에 원하는 출력이 나온다 하더라도 구현 FSM I의 천이 후의 현 상태가 기대되는 상태와 동일한 것인지를 확인하기가 쉽지 않다는 점이다. 이러한 문제에 대해, 첫 번째 문제를 관리한계(controllability limit)라 하고 , 두 번째 문제를 관찰한계(observability limit)라 한다. 일반적으로 관리한계 문제를 해결하기 위해서 초기상태(S_1)로부터 시작하여 원하는 상태까지 가장 짧은 경로(shortest path)를 이용하여 해당 상태에 도착한 후 해당 천이를 시험하게 되고, 관찰한계 문제를 해결하기 위해 시험하는 천이 후에 도착한 상태의 유일한 시험 시퀀스를 시험계열에 포함시켜 적용한 후 구현 FSM I의 결과상태를 확인하는 방법을 사용한다. 이러한 시험열로는 UIO(Unique Input- Output), DS(Distinguishing Sequence), CS(Characterization Set) 시퀀스 등이 있다(Uyar and Dahbura, 1986; Sidhu and Leung, 1989; Holzmann, 1991; Aho *et al.*, 1991). UIO 시퀀스란 FSM의 각 상태에 대해서 유일한 입력・출력 정보를 가지는 시험열이다. 즉, 각 상태는 자신만의 UIO를 가지고 있으며, 이 UIO가 다른 상태에 적용되었을 때는 기대하는 출력행위가 아닌 다른 출력행위를 가지게 된다. 따라서, UIO는 한 천이에 대한 적합성시험을 할 때 천이의 도착상태를 확인할 수 있다. DS나 CS 시퀀스는 존재를 위한 기본조건이 모델 FSM 자체가 완전하게 명세화되어야 하나, UIO 시퀀스는 이러한 요구조건이 없다. 그리고, UIO를 이용한 시험열의 길이가 DS나 CS 시퀀스를 이용하는 것에 비해 짧기 때문에 결과적으로 짧은 시험 스위트(test suite)를 생성하므로(Sidhu and Leung, 1989), 본 논문에서는 UIO 시퀀스를 사용하여 관찰한계 문제를 해결하고자 한다.

3.1 FSM 작성

적합성 검증을 위해서는 검증절차 단계 1과 같이 주어진 명세와 구현물에 대해 각각 FSM으로 모델링을 한 후, 검증절차 단계 2의 두 FSM I와 S의 관계가 동치(equivalence) 관계에 있음을 보임으로써 적합성을 확인하게 된다. 여기서, I의 임의의 상태 I_i에서의 모든 가능한 입력에 대해 동일한 출력을 갖는 FSM S 내의 상태(동치상태) S_j가 최소한 한 개 이상 존재하고, FSM S의 임의의 상태 S_j에 대해 동치관계에 있는 I의 상태 I_i가 최소한 한 개 이상 존재하면, 두 FSM I와 S는 동치라고 한다.

먼저, 명세와 구현물에 대한 FSM으로의 모델링은 메시지를 주고받는 송신개체와 수신개체 간 제어(control)부분을 상태 테이블(state table) 또는 상태천이 다이어그램(state-transition diagram)으로 표현할 수 있다. 상태천이 다이어그램으로는 유방향 그래프(directed graph) G = (N, A)를 사용하여 그래프 상의 노드집합 N 내의 각 노드(node)는 임의의 상태를, 그리고 가지(arc)집합 A 내의 각 가지는 상태천이를 나타내며, 송신측과 수신측 사이의 상호작용 관계를 표현한다. 본 연구에서 다루는 CMP 엔티티 빈에 대한 명세 FSM S는 <그림 3>과 같이 표현할 수 있다. 여기서, 상태 S_1은 엔티티 빈 객체가 생성되지 않은 상태를 나타내고, 상태 S_2는 생성되었지만 식별자가 부여되지 않은 비활성화 상태를 나타내며, 마지막으로, 상태 S_3는 고유한 식별자가 부여되어 메소드의 호출을 대기하는 상태인 활성화 상태를 나타낸다. 상태 간의 천이에 관련된 입・출력 데이터는 각 가지 위에 표시된 a_i/b_i로, 여기서 입력 데이터 a_i는 각각 <그림 2>의 상태천이 메소드인 a_1=newInstance.SetEntityContext, a_2=ejbHome, a_3=ejbFind.ejbSelect, a_4=ejbActivate, a_5=ejbLoad, a_6=ejbSelect, a_7=ejbStore, a_8= ejbPassivate, a_9=ejbCreate.ejbPostCreate, a_{10}=ejbRemove, a_{11}= unsetEntityContext이다. 각 상태 간 천이를 위한 출력 값 b_i는 EJB 명세에 규정되어

있지 않는 것들도 있으나, 이에 대한 입 · 출력 값이 서로 다르게 구현된 후 이들 값은 무시하는 형태로 실행되도록 하면 되므로 본 논문에서는 모든 입 · 출력 간 상이한 경우를 고려한다. 즉, $a_i \neq a_j$이고 $b_i \neq b_j$, $i \neq j$인 경우를 다룬다. 그러나, 각 입 · 출력 쌍 간 동일한 것이 존재한다 하더라도, 여기서 제시되는 시험열 생성절차를 수정 없이 적용 가능하다. <그림 2>에서 보는 바와 같이 명세 FSM S는 강하게 연결(strongly connected)되어 있고, 상태 수는 최소(minimal)이며, 결정형(deterministic)인 FSM이다. 여기서, 강한 연결망이란 네트워크 내 임의의 두 노드 간 유 방향(directed) 경로가 존재하는 망을 말한다.

3.2 시험열 생성

시험 계열을 적용한 결과 발생할 수 있는 오류는 '출력오류'와 '천이오류'이다. 출력오류는 주어진 '입력'에 대해 예기치 않은 '출력'이 구현 FSM으로부터 도출되는 경우이고, 천이오류는 주어진 '입력'에 대해 예기치 않은 상태로 천이가 일어나는 경우이다. 이러한 오류의 발생 여부를 확인하는 과정에서 발생하는 관찰한계(observability limit)의 문제를 해결하기 위해 본 논문에서는 UIO 시퀀스를 이용한다. FSM의 한 상태 i에서의 UIO인 UIO_i는 상태 i에 적용한 입력의 결과로 얻어지는 출력이 유일하게 결정되게 하는 입 · 출력 시퀀스를 말한다. 따라서, 구현 FSM I의 각 상태 i에서의 입력 시험열 T_{ij}에 대한 출력 TS_a 및 천이상태 S_l에 대해, $TS_0 = TS_a$이고 상태 l에 대한 UIO_i의 입력을 적용한 후, 결과로 관측되는 출력이 해당 UIO UIO_l의 출력과 동일하다면, UIO_l를 적용하기 전의 FSM 상태는 l이었음을 확인할 수 있게 된다. 즉, 검증절차의 단계 3을 통해 얻어지는 시험열의 구조는 다음과 같다 : (현 상태가 상태 i가 되게 하기 위한 입력 시퀀스) · (상태 i에서 인접상태 j로의 천이를 위한 입력 데이터) · (현 상태가 상태 j임을 확인하기 위한 UIO UIO_j), 여기서 ' · '는 연결(concatenation)을 의미한다. 즉, TS_{ij}를 상태 i에서 j로의 천이를 확인하기 위한 시험열이라 하고, T_{ij}를 상태 i에서 j로의 천이를 위한 입력이라고 하면, $TS_{ij} = T_{ij} \cdot UIO_l$가 된다.

CMP 엔티티 빈에 대한 명세 FSM인 <그림 3>에서의 각 상태의 UIO는 다음과 같다.

상태 S_1 : $UIO_1 = a_1/b_1$, *상태* S_2 : $UIO_2 = a_2/b_2$, *상태* S_3 : $UIO_3 = a_7/b_7$

각 상태 쌍 i에서 j간 천이를 확인하기 위한 시험계열은 $TS_{ij} = T_{ij} \cdot UIO_j$ 이므로, <그림 3>의 TS_{ij}는 다음과 같이 구해진다.

$$TS_{12} = T_{12} \cdot UIO_2 = a_1 \cdot a_2,\ TS_{22} = T_{22} \cdot UIO_2 = a_2 \cdot a_2,$$

$$TS_{22} = a_3 \cdot a_2,\ TS_{23} = a_4 \cdot a_7,\ TS_{33} = a_5 \cdot a_7,\ TS_{33} = a_6 \cdot a_7,$$

$$TS_{33} = a_7 \cdot a_7,\ TS_{32} = a_8 \cdot a_2,\ TS_{23} = a_9 \cdot a_7,\ TS_{32} = a_{10} \cdot a_2,$$

$$TS_{21} = a_{11} \cdot a_1$$

위에서 보는 바와 같이 <그림 3>의 상태 간 여러 개의 천이가 존재하는 경우, 각 천이별 시험열이 필요한데, 예를 들면, TS_{33}는 $a_5 \cdot a_7$, $a_6 \cdot a_7$, $a_7 \cdot a_7$의 3개의 재귀천이(self-loop)에 대해 각각 시험열이 필요하다.

명세에 대한 성질이 구현물에서 정확히 반영되었는지를 확인하기 위해서는 FSM의 각 천이가 바르게 수행되는지를 확인해야 한다. 즉, FSM으로 모델링된 명세의 모든 천이를 최소한 한 번 이상 검토해야 하는 것으로, 효율적인 시험이 되기 위해서는 FSM 내의 각 가지를 최소한 한 번 이상 거치면서 총 시험열의 길이(비용)가 최소가 되는 시험방안을 찾아야 한다. 이러한 시험방안은 바로 중국 우편배달부 문제(Chinese Postman Problem)가 된다. 즉, <그림 3>의 FSM의 중국 우편배달부 경로(CPT: Chinese Postman Tour)가 가장 효율적인 시험방안을 제공하게 된다. 그러나, 관찰한계의 문제를 해결하기 위해서는 검

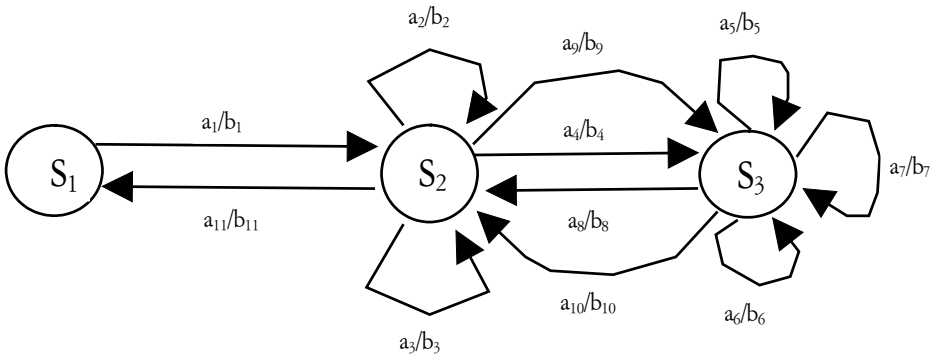

그림 3. CMP 엔티티 빈 인스턴스 생명주기 관리기능의 FSM S.

증절차의 단계 3에서의 시험열은 TS_{ij}을 포함하여야 하므로, <그림 3>을 확장한(augmentation) FSM S'를 고려해야 한다. 확장된 FSM S'는 <그림 3>의 FSM에 각 상태 간 시험열 { TS_{ij} }에 해당되는 가지 및 레이블을 추가하여 구성하는 것으로, 이를 G'=(N', A')의 그래프로 표현하면, <그림 3>의 그래프 G=(N, A)에 대해 A'=A $\bigcup A_C$ 이고 N'={A'에 포함된 모든 노드}의 관계를 가지는데, 여기서 가지 집합 A_C는 시험열 { TS_{ij} }에 해당하는 가지들로서 그래프 G"=(N', A_C)는 <그림 4>와 같다.

따라서, 효율적인 시험열을 생성하는 것은 그래프 G'에서 가지 집합 A_C에 대한 중국 시골 배달부 문제(Rural Chinese Postman Problem)를 푸는 것과 같다. 즉, 검증절차의 단계 3을 위해서는 다음과 같은 세부절차가 필요하다.

검증절차

단계 3.1: 상태 i에서 상태 j로의 천이를 확인하기 위한 시험 스위트 TS_{ij}를 구성, TS_{ij}= T_{ij} · UIO_j

단계 3.2: 명세 FSM S의 각 가지를 최소한 한 번 이상 확인하면서 최소의 시험경로(비용)을 가지는 시험 스위트를 RCPP (Rural Chinese Postman Problem)을 풀어서 구함

그래프 G'에 대한 RCCP 경로를 찾기 위해 다음과 같은 사실(Minieka, 1978)을 이용한다.

성질 1. 유 방향 망이 오일러 경로(Euler tour)를 가지기 위한 필요충분조건은 유 방향 망이 강 연결된(strongly connected) 대칭(symmetric) 망이다.

주어진 유 방향 망에서 오일러 경로가 존재한다면, 이는 최소의 길이를 가진 중국배달부 경로가 되므로, 오일러 경로는 중국 우편배달부 문제의 답이 된다. 그러나 강 연결이지만 대칭은 아닌 유 방향 망의 경우는, 성질 1에 의해 중국 배달부 경로가 오일러 경로가 아니므로 이 경로 상의 어떤 가지의 경우는 두 번 이상이 포함된다는 것을 알 수 있다. <그림 4>의 그래프 G"는 상태 S_2나 S_3에서 상태 S_1로 도달할 수 있는 경로가 존재하지 않으므로 강 연결망이 아니고, 상태 S_1에서의 (출력 가지수) − (입력 가지수) = 1이므로 대칭망도 아니다. 따라서, G"=(N', A_C)에서 G"의 가지집합 A_C의 모든 가지를 한 번씩만 거치는 오일러 경로는 존재하지 않는다. 그러나, 그래프 G'=(N', A $\bigcup A_C$)에는 <그림 3>의 가지집합 A를 포함하고 있으므로 강한 연결망이 되고, 여기에 추가의 임시가지를 적절히 중복시키면 대칭망을 형성할 수 있다.

본 연구의 목적인 시험열 생성은 위에서 기술한 바와 같이 그래프 G'의 RCPT(Rural Chinese Postman Tour)를 구하는 문제가 되고, 이를 위해서는 다음과 같은 사실(Minieka, 1978)을 이용한다.

성질 2. G'의 대칭확장(symmetric augmentation) 그래프 $\bar{G}$에 대한 오일러 경로는 G'의 CPT(Chinese Postman Tour)가 된다.

성질 1에 의해 대칭인 강 연결망은 오일러 경로를 가지므로, 성질 2를 이용하여 $\bar{G}$의 오일러 경로를 구하면 G'의 CPT가 얻어지는데, 이와 유사하게 G'의 가지집합 A_C에 대한 RCPT (Rural Chinese Postman Tour)를 구하는 문제는 대칭확장 그래프 $\bar{G}$에서 가지집합 A_C를 포함하고 A_C에 관련된 모든 노드집합 N_C에 의해 구성되는 시골대칭확장(rural symmetric augmentation) 그래프 G* = (N_C, $A_C \bigcup A_0$), $A_0 \subseteq$ A를 고려할 수 있다. 즉, 시골대칭확장망 G*는 G'의 가지집합 A에 포함된 가지는 0번 이상 포함시키고, G'의 가지집합 A_C 내에 포함된 가지들은 적어도 1번 이상 포함시켜서 대칭성을 만족시키되, 중복시키는 임시의 가지 수가 최소인 형태이다. 여기서 만약 N_C =N'이라면, G*에서의 오일러 경로는 G'의 RCPT가 됨을 다음 성질과 같이 알 수 있다.

성질 3. 시골대칭확장망 G*의 노드집합 N_C가 G'의 노드집합 N'과 동일하다면, G*의 오일러 경로는 G'의 RCPT가 된다.

중복시키는 가지 수가 필요 이상으로 많은 형태로 G*가 구축된다면, G*의 오일러 경로는 필요 이상의 중복된 가지도 거치는 경로이어야 하므로 최적의 시험열이 생성되지 않게 된다. 즉, G'에서 최적의 시골대칭확장 그래프 G*를 구축하기 위한 문제는 G"의 노드집합 N'에 속한 각 상태 쌍간 몇 개씩의 가지를 중복시키는 것이 좋은가를 결정하는 문제로, 다음과 같이 선형계획법(LP: Linear Programming) 문제로 모델링할 수 있다.

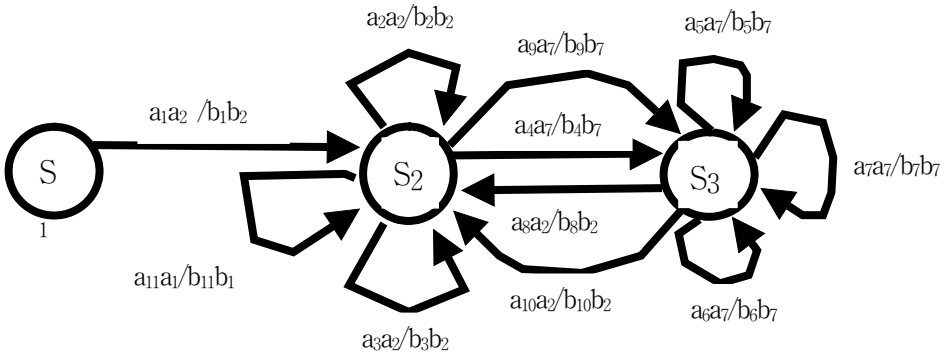

그림 4. FSM S 확장을 위한 가지 그래프 G".

$$\text{Min} \quad \sum_{i=1}^{|M|} \sum_{j=1}^{|M|} X_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{|M|} X_{ij} - \sum_{j=1}^{|M|} X_{ji} = n_i, \; i=1,2,..., |N'|$$

$$0 \le X_{ij}, \quad X_{ij} \in R, \; i,j = 1,2,..., |N'|,$$

여기서, X_{ij} : 노드(상태) i에서 j로의 추가 가지 수, i, j = 1, 2, ... , $|N'|$

n_i = G"의 노드 각 노드 i에 대해, (i로 입력되는 가지 수)-(i에서 나가는 가지 수)를 나타내는 상수(constant) 값

R = G'의 부분가지집합 A_C 내에 포함된 가지들에 해당되는 노드 쌍의 집합

위의 LP 모형에서 목적함수식은 추가되는 총 가지 수의 합이 최소화되어야 함을 나타내고, 제약식은 각 노드가 대칭이 되어야 한다는 것을 나타낸다. 여기서 결정변수인 X_{ij}는 추가되는 가지의 수를 나타내므로, 정수(integer)이어야 하지만, 이 문제의 제약식은 totally unimodular 성질을 만족하므로 선형계획법 문제로 모델링이 된다. 본 논문의 위와 같은 선형계획법 모형은 최대 N^2개의 결정변수를 위해 N개의 제약식으로 구성되었으나, 결정변수는 집합 R 내의 변수로, G'의 가지집합 A_C의 밀도가 작거나 통합시험이 불가능한 상태가 있는 경우에는 N^2보다 작아진다. 이는 Aho *et al.*(1991)이 모델링한 변수 수 N^2+2 및 제약식 수 $N+(N+1)^2$보다 훨씬 적으므로, G"에 대한 대칭확장망 G*을 효율적으로 구할 수 있다.

<그림 4>의 G"에서 각 상태 S_1, S_2, S_3에서의 $\{n_i\}$ 값을 구하면, n_1=-1, n_2=1, n_3=0이다. 결정변수 X_{ij}는 G"의 두 상태 간 가지 수에 추가할 가지 수를 나타내므로, G"에서 두 상태 간 하나의 가지라도 존재하는 경우인 R={(1,2), (2,2), (2,3), (3,3)}에 대한 X_{ij}만 고려하면 된다. 따라서, G*를 위한 LP 모형은 다음과 같다.

$$\text{Min} \quad X_{12} + X_{21} + X_{22} + X_{23} + X_{32} + X_{33}$$

$$\text{s.t.} \quad X_{12} - X_{21} = -1$$

$$(X_{21} + X_{22} + X_{23}) - (X_{12} + X_{22} + X_{32}) = 1$$

$$(X_{32} + X_{33}) - (X_{23} + X_{33}) = 0$$

$$X_{ij} \ge 0 \quad \forall i, j$$

이 문제에 대한 최적해를 LINDO(Lindo Systems, 2003)를 이용해서 푼 결과는 X^*_{21}=1이고 나머지 변수는 모두 0이다. 따라서, G'에 대한 대칭확장망 G*는 <그림 5>와 같이 G"에 상태 2에서 1로의 가지를 하나 추가하면 구성된다.

시험열 생성을 위한 마지막 단계는 그래프 G'에 대한 RCPT를 구하는 것으로, 이 문제는 일반적으로 NP-complete하다(Lenstra and Rinnooy Kan, 1976). 그러나, CMP 엔티티 빈의 인스턴스 관리기능의 경우는 <그림 5>의 시골대칭확장 망 G*에서의 노드집합 N_C가 G'의 노드 집합 N'와 동일하므로, 성질 3에 의해 G*의 오일러 경로를 구함으로써 G'의 RCPT를 구할 수 있다. <그림 5>의 G*는 대칭이고 강 연결된 망이므로, 성질 1에 의하면 G*의 오일러 경로는 존재가 보장된다. 그리고, 오일러 경로는 그래프 내의 모든 가지를 한 번씩만 거치는 폐환경로이므로, 다음 성질 4가 성립한다.

성질 4. 오일러 경로의 총 길이는 탐색 시작 노드에 관계없이 동일하다.

성질 4에 따라 polynomial 계산복잡도(complexity)를 가지는 오일러 경로해법(Minieka, 1978)을 이용하여 G*에 존재하는 오일러 경로를 구하면, 결국 CMP 인스턴스 관리기능 명세 S에 대한 시험열은 다음과 같이 구해진다.

$$a_1a_2/b_1b_2 \cdot a_2a_2/b_2b_2 \cdot a_3a_2/b_3b_2 \cdot a_{11}a_1/b_{11}b_1 \cdot$$
$$a_4a_7/b_4b_7 \cdot a_8a_2/b_8b_2 \cdot a_9a_7/b_9b_7 \cdot a_5a_7/b_5b_7 \cdot$$
$$a_7a_7/b_7b_7 \cdot a_6a_7/b_6b_7 \cdot a_{10}a_2/b_{10}b_2 \cdot a_{11}/b_{11}$$

위와 같이 구한 시험열은 검증절차의 단계 4~6과 같이 구현

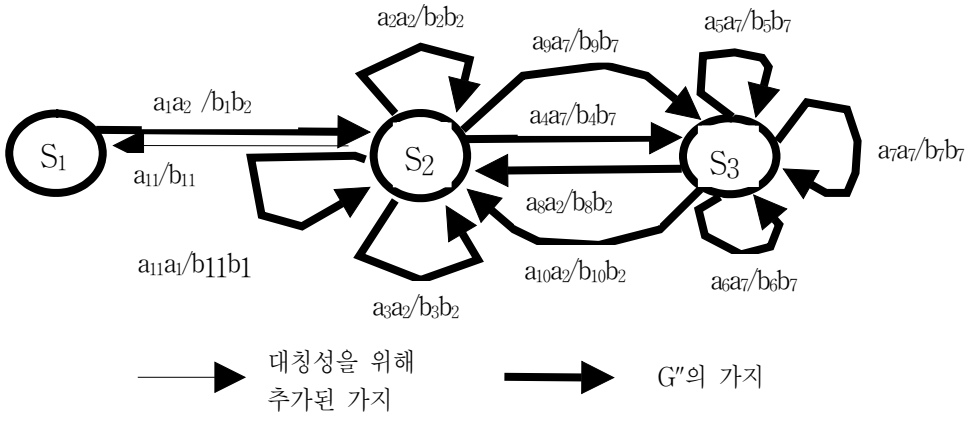

그림 5. 시골대칭확장망 G*.

물에 적용하여 결과를 비교함으로써 적합성 검증을 수행할 수 있다.

4. 결 론

소프트웨어 부품을 재사용하는 방식의 소프트웨어의 개발은 오래 전부터 연구되고 시도되어 왔으나 최근 들어 컴포넌트 기반의 소프트웨어 개발(CBD; Component-Based Development)방법이 정착되면서 컴포넌트 및 CBD는 21세기 소프트웨어 산업을 이끌 핵심적인 화두로 등장하고 있다. 서버측 컴포넌트 모델인 EJB(Enterprise JavaBeans)와 같이 재사용성 및 plug- and-play의 성질을 갖는 컴포넌트 기반 소프트웨어의 경우는 적합성 검증이 특히 중요한 부분이라 할 수 있다.

본 논문은 EJB 컴포넌트 모델의 적합성 검증에 대한 것으로, EJB 명세에 대한 구현물이 정확하게 구현되었는지를 검증하기 위한 효율적인 시험열 생성방안을 다루었다. 적합성 검증에 관련된 국내외의 기존 연구들은 모두 각기 특정 검증대상에 대한 효율적인 적합성 검증방안에 대한 것으로, EJB를 검증대상으로 하는 시험열 생성방법은 현재까지 다루어지지 않은 상태이다. 본 논문에서는 CMP 엔티티 빈에 대한 규격을 분석하여 적합성 검증방안을 적용하는 실제 예를 보였다. 적합성 검증을 위해서 명세를 FSM(Finite State Machine)으로 모델링하였고, UIO(Unique Input-Output)을 이용한 최적의 시험열을 생성하기 위해 중국 시골배달부 문제(rural Chinese postman problem) 및 선형계획법(Linear programming) 모형을 이용하였다.

본 연구에서 제시한 시험열 생성을 통한 검증방법은 CMP 엔티티 빈 인스턴스 관리 모듈의 검증에서는 polynomial 계산복잡도(complexity)를 가지지만, EJB 명세의 다른 기능 모듈에 대해서는 NP-complete한 계산복잡도를 가질 수 있다. 그리고 상태 수가 매우 많은 경우에는 적용이 용이하지 않다는 문제도 있다. 따라서, 본 논문에서 제안한 시험열 생성절차를 따르는 시험열 자동 발생도구의 개발이 필요하고, 상태 수가 많은 경우에도 적용이 가능한 정리증명기(theorem prover)와 같은 검증방법의 연구나 발견적 해법에 대한 연구도 필요한 상태이다.

참고문헌

Aho, A.V., Dahbura, A.T., Lee, D. and Uyar, M.U.(1991), An Optimization Technique for Protocol Conformance Test Generation based on UIO Sequence and Rural Chinese Postman Tours, *IEEE Transactions on Communications*, **39**(11), 1604-1615.

Csondes, T., Kotnyek, B. and Szabo, J.Z.(2002), Application of Heuristic Methods for Conformance Test Selection, *European Journal of Operational Research*, **142**, 203-218.

Hailpern B. and Santhanam, P.(2002), Software Debugging, Testing, and Verification, *IBM Systems Journal*, **41**(1), 4-12.

Holzmann, G.J.(1991), *Design and Validation of Computer Protocols*, Prentice-Hall.

Kim, S.-H., Roh, M.-C., Seo, B.-S., Jang, C., Jung, S.-W. and Kim, J.-B.(2002), E504 EJB(Enterprise Java Beans) Server System for Enterprise e-Business, *Proceeding of CALS/EC Korea 2002*, Korea Institute of CALS/EC, 478-485.

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

Lenstra J.K. and Rinnooy Kan, A.H.G.(1976), On General Routing Problems, *Networks*, **6**, 273-280.

Lindo Systems (2003), *LINDO System's Index Page*, http://www.lindo.com.

Minieka, E.(1978), *Optimization Algorithms for Networks and Graphs*, Marcel Dekker Inc.

Nakajima S. and Tamai, T.(2001), Behavioural Analysis of the Enterprise JavaBeans Component Architecture, *Lecture Notes in Computer Science(LNCS)*, **2057**, 163-182.

Sidhu, D.P. and Leung, T.-K.(1989), Formal Methods for Protocol Testing : A Detailed Study, *IEEE Transactions on Software Engineering*, **15**(4), 413-426.

Sousa J.P. and Garlan, D.(2001), Formal Modeling of Enterprise JavaBeans Component Integration Framework, *Information and Software Technology*, **43**, 171-188.

Sun Microsystems (2003), *Enterprise JavaBeans(TM) Specification*, http://java.sun.com/ products/ejb.

Uyar M.U. and Dahbura, A.T.(1986), Optimal Test Sequence Generation for Protocols : the Chinese Postman Algorithm Applied to Q.931, *Globecom'86*, 3.1.1-3.1.5.

주 운 기

성균관대학교 산업공학과 학사
한국과학기술원 산업공학과 석사
한국과학기술원 산업공학과 박사
한국전자통신연구원 선임/초빙연구원
현재: 선문대학교 지식정보산업공학과 부교수
관심분야: 통신 시스템, 일정계획, 생산정보 시스템

김 중 배

고려대학교 산업공학과 학사
한국과학기술원 산업공학과 석사
한국과학기술원 산업공학과 박사과정
대한항공(주) 시스템부
현재: 한국전자통신연구원 로봇소프트웨어 아키텍처연구팀장
관심분야: 미들웨어, 시스템 소프트웨어, 실시간 시스템