韓國 OA學會 論文誌
第5卷 第2號, 2000. 6.

2000-5-2-1-6

Java Bean 기반 시스템에서 컴포넌트 통합을 위한 모델링에 관한 연구

소 경 영*, 박 종 구**

A Study on the Modeling for Component Integration in the Java Bean-based System

So Kyung-Young, Park Jong-Goo

요 약

CORBA에서 객체 기술은 분산 및 이기종 기계에 분산되어 있는 소프트웨어 컴포넌트의 통합된 구현을 용이하게 한다. CORBA와 유사한 객체 통합 기술들은 표준화된 컴포넌트 통합 및 상호 동작 모델을 정하고 호환 불가능한 컴포넌트 구현을 캡슐화하기 위한 객체지향 원리를 발전시켰다. 본 논문에서는 Java Bean에 기반을 둔 분산 시스템 환경에서 객체와 객체간에 관련성을 모델링하기 위해 컴포넌트, 연결자 및 컴포넌트 스키마로 구성된 구조화된 모델을 제시하고 구현한다. 특히 Java Bean 환경에서 객체간의 관련성을 모델링하기 위한 연결자의 구성에 중점을 둔다.

Abstract

Object technologies like the OMG's CORBA are enabling technologies the aim to facilitate integration implementation of diverse software components in distributed, heterogeneous environment. CORBA and similar object integration technologies define a standard component interconnection and inter-operation model, promote object-oriented principles to encapsulate incompatible component implementations. In this paper, we present a connector model for software architectural representation of complex component collaborations. Our Connector model is base on research in software achitecture and object-oriented modeling, and part of a design framework for modelig component-based system. We believe the connector concepts to be vary benificial for a clear expression of dependencies between multiple component in Java Bean-based system.

* 국립익산대학 컴퓨터과학과 부교수
** 원광대학교 컴퓨터공학과 교수

Ⅰ. 서 론

객체지향 소프트웨어 시스템은 상호 연관되어 있는 수많은 객체의 집합으로 구성되며 객체지향설계 및 분석 단계에서 객체에 대한 정보는 일반적으로 클래스 및 클래스간의 관계를 정형화된 기법을 적용하여 명확하게 기술한다. 또한 클래스 인스턴스로서의 객체와 객체간의 관계를 보다 객체지향 패러다임의 특성인 일반화, 구체화, 관련성(association) 등을 이용하여 보다 구체적으로 기술한다[1][7].

OMG(Object Management Group)의 CORBA에 기반을 둔 분산 객체 시스템에서도 서로 연관되고 상호 동작을 수행하는 수많은 객체의 집합으로 구성되어 있다. 하지만 CORBA에 기반을 둔 시스템에서는 객체간에 모델을 설정하고 설계를 하는데 있어 객체지향 기술에서 사용하였던 클래스에 기반을 둔 모델링 기술을 적용하는데 제약을 가진다. 왜냐하면, CORBA 객체를 기술하는데 있어 클래스 추상화를 이용하면 엄격한 인터페이스 부분과 구현 부분을 분리하기가 어렵기 때문에 충분하게 CORBA의 객체의 특성을 기술할 수 없다. 따라서 추상화와 CORBA 객체간의 관련성을 일반적인 클래스 관련성을 이용하여 표현하는데는 새로운 기법이 적용되어야 한다.

CORBA에서 객체 기술은 분산 및 이기종 기계에 분산되어 있는 소프트웨어 컴포넌트의 통합된 구현을 용이하게 한다. CORBA와 유사한 객체 통합 기술들은 표준화된 컴포넌트 통합 및 상호 동작 모델을 정하고 호환 불가능한 컴포넌트 구현을 캡슐화하기 위한 객체지향 원리를 발전시켰다. 모든 통합된 소프트웨어 엔티티는 객체와 유사한 인터페이스를 통해 정보를 전달하고 동기화된 메소드 호출을 통해 상호 작용을 하는 객체로 간주된다. 컴포넌트간에 상호 협력을 위한 진보되고 다양한 통신 방법 및 상호 작용 모델은 CORBAServices와 같은 일반적인 객체 서비스 기술을 통해 시작되었다[2].

본 논문에서는 Java Bean에 기반을 둔 분산 시스템 환경에서 객체와 객체간에 관련성을 모델링하기 위해 컴포넌트, 연결자 및 컴포넌트 스키마로 구성된 구조화된 모델을 제시하고 구현한다. 특히 Java Bean 환경에서 객체간의 관련성을 모델링하기 위한 연결자의 구성에 중점을 둔다. 본 연구에서 제시된 연결자 모델은 Java Bean 기반 분산 시스템 환경에서 다양한 객체간의 의존성을 명확하게 표현하는데 효과적이며 분산되어 있는 컴포넌트를 정형화된 방법으로 통합할 수 있는 효과를 가진다.

본 논문의 구성은 제 2장에서 CORBA 환경과 Java Bean 환경에서 분산 컴포넌트 기술에 대해 고찰하며 제 3장에서는 Java Bean 환경에서 분산 컴포넌트 통합을 위한 모델링 방법을 제시한다. 마지막으로 제 4장에서는 결론과 향후 연구 방향에 대해 기술한다.

Ⅱ. 분산 컴포넌트 기술

2.1 분산 컴포넌트

최근에는 Java 언어를 이용하여 기존에 작성된 컴포넌트를 발전시켜 새롭게 개발된 것이 Java Bean으로서 Java 언어를 사용하여 Java 컴포넌트를 구축할 수 있는 시스템 도구이다. Java Bean은 자바 언어의 플랫폼 독립성을 활용하여 이기종 시스템 환경에서 동작되는 보편적인 컴포넌트로서 성장을 기대하고 있다[3].

Java Bean 자체는 단일 기계상의 컴포넌트를 주로 고려하고 있지만 실제 컴포넌트는 복잡한 클라이언트/서버 시스템에서 생성되는 분산 컴포넌트이며 이러한 분산 컴포넌트들을 구축하는 것은 객체 웹의 핵심 요소이다. 따라서 CORBA 컴포넌트나 Enterprise Java Bean(EJB)의 구상이 자연스럽게 등장하였다. CORBA Bean은 초창기에 단순히 TCP/IP 기반 네트워크에서 ORB간의 통신을 위한 표준 프로토콜인 IIOP(Internet Inter-ORB Protocol)를 통해 클라이언트 빈에서 호출할 수 있는 CORBA 객체로 존재하였다. 이러한 CORBA 컴포넌트는 Java Bean과 함께 결합되었을 때의 발생되는 상승 효과로 인해 현재 OMG에서 CORBA 컴포넌트의 기본 모델로 Java Bean을 채택하고 있다. 실제 CORBA는 Java Bean 컴포넌트에 Bean을 위한

분산 서비스 하부 구조를 제공하는 것과 CORBA에 메타 데이터, 이벤트, 패키지화에 관련된 다양한 도구를 제공하는 효과를 제시하고 있다. 실제 CORBA 프로그래밍은 소수의 전문가들에 한정되어 있으므로 일반 개발자들이 비주얼 개발 도구나 스크립트 언어를 이용하여 비즈니스 어플리케이션을 쉽게 제작할 수 있는 환경을 구축하기 원했고 이를 위해서는 표준 컴포넌트 기반 구조를 정의해야만 했다. 그래서 URL, LDAP(Lightweight Directory Access Protocol) 등을 이용하여 객체에 명명할 수 있는 구조, 보안 지원 구조, 기존의 Java나 Java 스크립트와 같은 언어를 통한 객체 접근 및 제어 등의 필요성을 제안하고 있다. 따라서 Java Bean 컴포넌트를 CORBA 컴포넌트의 기본 모델로 채택하여 지원하고 있으며 이러한 모델의 요소로 Java Bean 설계 패턴, 이벤트 속성, 패키지, 메타 데이터, 도구 등이 있다. 또한 현재의 주요한 개발 환경인 비주얼 도구의 페러다임을 그대로 적용하고 있다. 이러한 도구를 가지고 속성(property) 편집기를 호출할 수 있고 컴포넌트간 상호 연결 페러다임을 제공하며 CORBA 컨테이너에서 컴포넌트들이 계층적으로 구축될 수 있고 컨테이너는 하나 이상의 컨테이너를 구축할 수 있다[2][3][6].

2.2 CORBA 컴포넌트 모델

일반적인 어플리케이션은 다양한 방법으로 설계가 가능하며 이러한 어플리케이션은 하나의 큰 블럭이 된다. 이것은 모듈 단위의 결합도가 높아 모듈 단위로 분리하기가 매우 어려우며 이것은 코드의 재사용이나 확장에 많은 제약을 가하게 된다. 이러한 제약을 해결하기 위해 CORBA는 컴포넌트 개념을 도입하게 되었다. 이러한 컴포넌트 모델은 플러그 앤 플레이 CORBA 객체의 개발을 위한 프레임워크를 지정하여야 한다. 이 CORBA 컴포넌트 모델은 CORBA-기반 컴포넌트 모델과 이러한 컴포넌트를 다른 컴포넌트 모델과 매핑을 위해 인터페이스와 메카니즘을 제공해야 한다. 이와 같은 기능을 제공하면 객체지향 소프트웨어 요소를 표현하는 기능과 이것들을 어플리케이션으로 만들기 위한 더욱 완벽한 메카니즘을 제공할 수 있다. 또한 CORBA 컴포넌트 모델은 Java Bean과 Active-X 컴포넌트를 포함한 다른 컴포넌트 기술과 강력한 통합이이 가능한다. 즉, 시스템의 확장성과 재사용성이 증가할 수 있다. CORBA 3.0 명세서에서는 이러한 CORBA 컴포넌트 모델을 위하여 Java Bean을 채택하였다.

2.3 Java Bean에서의 분산 컴포넌트

Java Bean은 이러한 컴포넌트 소프트웨어 어셈블리 패러다임을 새로운 수준으로 향상시키는 기술로서 최근에 등장한 중요한 기술이다. Java Bean은 동적인 컴포넌트를 생성하고 사용하기 위한 아키텍처이자 플랫폼 중립적인 API로서 기존의 제품들에 의해 정립된 컴포넌트 어셈블리 개발 모델의 장점을 기반으로 하여 구축되어 우수한 성능을 가지고 있다. 어플리케이션 개발자들은 다양한 개발 도구를 사용해서 이기종 기계로 이식가능한 Java Bean으로부터 사용자에 요구에 적합한 어플리케이션을 조합하여 다른 어플리케이션을 생산할 수 있다[4].

Java는 일반적으로 완벽하게 이기종 기계로 이식가능한 인터넷 및 기업 인트라넷 애플릿과 어플리케이션을 구축하기 위한 업계 표준 플랫폼으로 확고히 자리잡고 있다. Java 플랫폼은 이러한 유형의 어플리케이션을 개발하는 개발자에게 다음과 같은 여러 가지 장점을 제공한다.

완벽하게 이식 가능한 플랫폼 : 브라우저 영역에서 지원되는 다양한 언어와 라이브러리, 가상 기계의 급속한 파급 효과를 보이고 있는 Java 플랫폼은 빠른 시간 내에 운영체제 영역에서도 급속히 확산되고 있다. 따라서 개발자들은 어플리케이션 기능을 한 번만 작성하면 다양한 운영체제와 하드웨어 환경에서 그 어플리케이션을 수행할 수 있다.

컴포넌트 모델(component model)이란 사용자의 요구에 따른 임의의 어플리케이션을 생성할 때 동적으로 결합시킬 수 있는 소프트웨어 컴포넌트 개발자들이 규정할 수 있게 하는 아키텍쳐이면서 API라고 할 수 있다. 컴포넌트 모델은 컴포넌트와 컨테이너(container)의 2가지 주요 요소로 구성된다. 컴포넌트는 크기나 기능면에서 다양하여 버튼과 같은 작은 GUI 위젯부터 뷰어 같은 애플릿 크기, 텍스트 어플리케이션의 HTML, 브라우저와 같이 규모가 큰 어플리케이션 등이 있다. 컴포넌트는 버튼과 같이 시각적으로 표시될 수 있고 데이터 공급 모니터링 컴포넌트와 같이 시각적으로 보이지 않을 수도 있다. 컨테이너는 어셈블리나 관련 컴포넌트를 보유하는 데 사용된다.

Ⅲ. 컴포넌트 통합을 위한 모델링

3.1 컴포넌트 통합을 위한 모델링 설계

본 논문에서 Java Bean 시스템 환경에서 컴포넌트 통합을 위한 연결자 모델링을 설계하고 구현하기 위해 소프트웨어 아키텍처, 구조적 기술 언, 객체지향 모델링 기술, 분산 객체의 처리를 위한 참조 모델 등에 기반을 둔다.

이를 위해 Java Bean 기반을 둔 소프트웨어 아키텍처를 모델링하며 그림 1과 같이 구성된 컴포넌트, 연결자 및 컴포넌트 스키마로 구성된 프레임워크를 설계하고 구현한다.

컴포넌트는 Java Bean 환경에 존재하는 분산 객체의 집약적인 기술로서 인터페이스 집합인 인터페이스 명세 부분, 내부적인 객체지향 스키마인 표현(representation) 부분, 외부적인 객체지향 스키마인 표현 지도(representation map)를 포함하고 있다. 컴포넌트 인터페이스는 클라이언트에게 제공되는 컴포넌트 서비스를 기술하고 있으며 CORBA-IDL과 같은 인터페이스 기술 언어(Interface Description Language)를 이용하여 표현된다. 표현 지도 부분은 표현 부분과 인터페이스 명세 부분을 연관시키며 객체지향 스키마로서 표현된다.

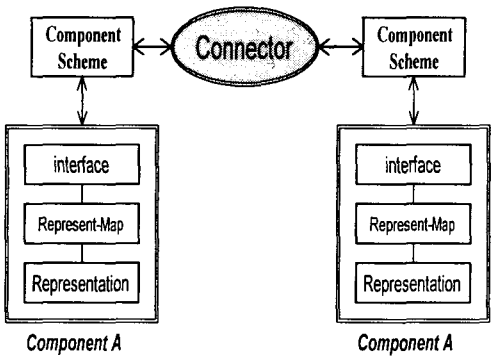

그림 1. 연결자 통합 모델링 구조

본 논문에서 설계한 계층적 구조를 가진 구조는 Java Bean 객체 모델에서 제안한 서로 다른 객체 단위와 보다 프로그램 지향적인 객체 모델을 연결하는 역할을 한다. 즉, 컴포넌트는 논리적이며 수많은 프로그램 수준의 내부 객체와 외부 객체로 구성된 분산 객체들이다.

연결자는 컴포넌트의 논리적인 관련성을 갖는 통합과 객체의 동적인 관계를 나타내는 컴포넌트의 상호작용을 추상화하며 연결자는 기능(role), 기능 인터페이스(role interface), 상호 작용 프로토콜(interaction protocol)의 기술을 포함하고 있다. 또한 연결자는 컴포넌트 사이의 상호 동작을 위해 컴포넌트로부터 요구되는 행위의 형태를 기술한다. 컴포넌트 스키마는 특정 컴포넌트를 위한 컴포넌트의 추상화를 번역한다. 컴포넌트는 CORBA 객체 모델과 같은 객체 모델링 방법에 의해 직접 지원되지 못하는 단점을 가지고 있다. 그러므로 연결자는 컴포넌트에 직접 연결(mapping)되어야 한다. 즉, 컴포넌트 사이의 상호 작용을 위한 책임성이 컴포넌트와 CORBA 분산 객체 구조와 같은 연결자의 번역을 위해 컴포넌트를 구성하는 원소들에게 분산되어야 한다.

3.2 이벤트 통보 연결자

객체의 동작이 동기화된 수행의 결과로 메소드 호출이 발생되는 표준화된 CORBA 객체 통신 모델링 방법에 반대로 이벤트는 다양한 객체사이에서 비동기화 방법으로 객체간에 통신을 한다. OMG의 이벤트 서비스는 표준화된 인터페이스와 "publish/subscribe"에 기반을 둔 이벤트-참여(participation)를 위한 객체 상호 작용 모델을 지정한다.

하나 이상의 객체들은 이벤트 데이터를 발생시키고 이벤트 제공자(supplier)로서 동작한다. 또한 수많은 다른 객체들은 이벤트 데이터를 받고 이벤트 소비자(consumer)로서 동작한다. 본 논문에서 일반적인 이벤트 통신 방법과 반대가 되는 모든 이벤트 공급자는 동적으로 이벤트를 전송하는 push 형태이고 모든 클라이언트는 이벤트의 전송을 기다리는 push 형태 또는 동적으로 이벤트 발생에 대한 질의를 하는 pull 형태를 갖는 이벤트 채널로서 포괄 이벤트(generic event) 통신 방법으로 모델링된다.

연결자 EventNotification의 명세서에는 이벤트 통보를 집약적으로 수행하기 위해 객체사이의 상호 의존성을 기술하며 객체의 기능(role), 기능 인터페이스(role interface), 상호 작용 프로토콜에 의해 구조화된 형태를

갖게된다. 사건 통보(EventNotification) 연결자를 위한 기능 및 기능 인터페이스는 그림 2와 같다.

```
Connector EventNotification
Role :
    EventPushSupplier,
    EventPushConsumer,
        . . .
Role Intrface :
EventPushSupplier.CosEventComm::PushSupplier;
EventPushConsumer.CosEventComm::PushConsumer;
. . .
```

그림 2. 사건 통보 연결자 기능 및 인터페이스

기능 부분과 다른 기능의 인터페이스 사이에서 서로 다른 방향을 가진 사용 관련성이 존재하는데 이러한 과정은 그림 3과 같이 인터페이스 사용 다이어그램으로 표현된다. 인터페이스를 가리키는 화살표는 화살표의 출발점이 인터페이스의 클라이언트를 나타내는 기능을 나타낸다. 즉, 기능 역할을 나타내는 컴포넌트는 일정한 시간에 하나의 포인트를 갖으며 인터페이스를 사용한다. 기능 부분이 어떠한 순서로 인터페이스를 이용하며 협력 관계를 유지하는 어떠한 상태에서 인터페이스를 이용하는지에 대한 정보는 주어지지 않는다.

협력관계를 유지하는데 있어서 연결자의 실제적인 동작을 나타내는 행위적인 면은 인터페이스 프로토콜을 통해 지정되어 진다. 인터페이스 프로토콜은 선행 조건(pre-condition) 및 후위 조건(post-condition)을 고려하여 협력 관계에 대한 동작과 동작의 순서를 기술한다.

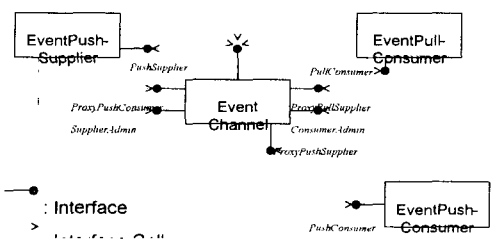

그림 3. 기능 인터페이스

연결자는 객체간 의존성의 집약적인 기술을 위하여 사용되며 이러한 연결자의 번역을 위하여 연결자 스키마(connector scheme)를 이용한다.. 시스템 모델링에 있어서 특정한 컴포넌트는 특정한 컴포넌트를 위한 연결자의 번역을 통하여 이벤트 통보와 같은 연결자를 이용하여 통합되어질 수 있다.

3.3 구조적인 관련성(relationships)

객체간의 관련성을 위한 기본적인 수단으로서 CORBA는 객체 참조(object reference)를 기반으로 하여 Java Bean에서 동일한 방법으로 정의한다. 객체 참조는 객체들이 다른 객체에 대한 요청을 발생시키기 위해 참조된다. 하지만 객체간의 관련성이 속성을 유지하거나 다중 방향성을 가지며 높은 우선 순위를 갖는 매우 복잡한 관계를 갖는다면 관련성 서비스(relationships service)가 이용된다. 3단계 계층적으로 구조화된 이러한 서비스는 엔티티-관련성 모델링에 기반을 둔 동작의 추상화를 위하여 인터페이스와 메카니즘을 정의한다. 엔티티-관련성 모델링은 기능 부분, 관련성(relationship), 연관된 객체의 탐색 그래프, 실행 시간에 서브-그래프로 정의한다.

본 논문에서는 수많은 다양한 객체와 인터페이스들이 Java Bean 기반 시스템에서 ER-like 관련성의 모델링을 위한 관련성 서비스를 위하여 적용되어 진다. 또한 Relation과 Traversing과 같은 두 개의 연결자가 구현된다.

Relation 연결자는 관련성을 설정하고 유지하기 위한 컴포넌트를 위해 RelatedObject와 Relationship과 같은 두 개의 컴포넌트를 포함한다. 또한 객체 사이에 관련성을 가질 수 있는 컴포넌트 관련성의 검색을 위하여 NavigatingClient 기능을 도입하였다.

Relation 연결자로부터 탐색 연결자의 구성을 위해 노드와 그래프로 구성된 관련성 서비스 명세서(relationship service specification)의 수준에 기반을 둔 Java Bean 객체 관련성을 위해 기능, 기능 인터페이스 및 상호 작용 프로토콜을 설계하여 탐색 연결자를 구현하였다.

연결자는 연관된 객체의 탐색 그래프(traversing graph)를 위하여 RelatedObject나 Relationship의 기능에 존재하는 여러 개의 컴포넌트들과 Traversing Cient 또는 TraversalObject의 기능에 존재하는 컴포넌트들을 이용한다.

Ⅳ. 결론 및 향후 과제

CORBA에서 객체 기술은 분산 및 이기종 기계에 분산되어 있는 소프트웨어 컴포넌트의 통합된 구현을 용이하게 한다. CORBA와 유사한 객체 통합 기술들은 표준화된 컴포넌트 통합 및 상호 동작 모델을 정하고 호환 불가능한 컴포넌트 구현을 캡슐화하기 위한 객체지향 원리를 발전시켰다. 모든 통합된 소프트웨어 엔티티는 객체와 유사한 인터페이스를 통해 정보를 전달하고 동기화된 메소드 호출을 통해 상호 작용을 하는 객체로 간주된다. 컴포넌트간에 상호 협력을 위한 진보되고 다양한 통신 방법 및 상호 작용 모델은 CORBAServices와 같은 일반적인 객체 서비스 기술을 통해 시작되었다.

본 논문에서는 Java Bean에 기반을 둔 분산 시스템 환경에서 객체와 객체간에 관련성을 모델링하기 위해 컴포넌트, 연결자 및 컴포넌트 스키마로 구성된 구조화된 모델을 제시하고 구현한다. 특히 Java Bean 환경에서 객체간의 관련성을 모델링하기 위한 연결자의 구성에 중점을 둔다. 본 연구에서 제시된 연결자 모델은 Java Bean 기반 분산 시스템 환경에서 다양한 객체간의 의존성을 명확하게 표현하는데 효과적이며 분산되어 있는 컴포넌트를 정형화된 방법으로 통합할 수 있는 효과를 가진다.

참 고 문 헌

[1] James Rumbaugh, Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[2] Robert Orfall, Dan Harkey, Client/Server Programming with JAVA and CORBA, John Wiley and Sons Inc.

[3] Kurt Wallnau, Nelsin Weiderman, Distrubuted Object Technology With CORBA and Java : Key Concepts and Implications, Technical report CMU/SEI-97-TR-004, June, 1997.

[4] Susanne Busse, Stefan Tai, Software Architectural Modeling of the CORBA Object transaction Service, Software Corp, 1997.

[5] Philippe Kruchten, Modeling Component Systems with the Unified Modeling Language,relation Software Corp, 1997.

[6] S. Voloski, CORBA: Integration Diverse Applications Within Distributed Hetegeneous Environments, IEEE Communications, Vol. 4, No. 2, 1997.

[6] 왕창종, 이세훈, Inside CORBA 3 프로그래밍, 대출판사, 1999.

[7] 이상구 외 3인, CORBA&Java 분산객체 기술, 교학사, 1998.

저 자 소 개

소 경 영

1986년 원광대학교 전자공학과 졸업(공학사)

1990년 원광대학교 컴퓨터공학과 졸업(공학석사)

1998년 원광대학교 컴퓨터공학과 박사 수료

1991년~현재 익산대학 컴퓨터과학과 부교수

관심분야 : 소프트웨어공학, 분산시스템, 전문가시스템

박 종 구

1999년 동국대학교 통계학과 졸업(이학박사)

1981년~현재 원광대학교 컴퓨터공학과 교수

관심분야 : 소프트웨어신뢰성공학, 전문가시스템, 시스템프로그래밍