Java Beans 환경에서 컴포넌트 연결자 모델링의 설계 및 구현

정 성 옥
광주여자대학교 정보통신학부
전화 : 062-950-3733 / 핸드폰 : 011-608-6686

Design and Implementation of Component Connector Modeling in the Java Beans Environment

Sung-Ok Jung
Gwangju Women's University, Division of Information Communication
E-mail : sojung@namkyung.kwu.ac.kr

Abstract

Components are abstractions of system level computational entities, connectors are abstractions of component interrelationships. we propose connectors as transferable abstractions of system level component interconnection and inter-operation. Connectors are architectural abstractions of component coordination in the abstract architecture of a system only. Connectors describe a collaboration rationale for component adaptations, which are then modeled in the concrete architecture of a system.

I. 서론

객체지향 소프트웨어 시스템은 상호 연관되어 있는 수많은 객체의 집합으로 구성되며 객체지향 분석 및 설계 단계에서 객체에 대한 정보는 일반적으로 클래스 및 클래스간의 관계를 정형화된 기법을 적용하여 명확하게 기술한다[1]. OMG(Object Management Group)의 CORBA(Common Object Request Broker Architecture)에 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 CORBA 객체의 특성을 기술할 수 없다. 따라서 추상화와 CORBA 객체간의 관련성을 일반적인 클래스 관련성을 이용하여 표현하는데는 새로운 기법이 적용되어야 한다.

본 논문에서는 Java Beans에 기반을 둔 분산 시스템 환경에서 객체와 객체간에 관련성을 모델링하기 위해 컴포넌트, 연결자(connector) 및 컴포넌트 스키마(scheme)로 구성된 구조화된 모델을 제시하고 구현한다. 특히 Java Beans 환경에서 객체간의 관련성을 모델링하기 위한 연결자의 구성에 중점을 둔다.

II. 분산 컴포넌트 기술

기존의 제 4세대 언어인 비쥬얼 베이직 또는 델파이에서는 컴포넌트에 대해 많이 언급되어 왔지만 시각적인 컴포넌트인 블랙박스 형태의 컴포넌트에 대해 정의하기는 많은 제한 사항을 가지고 있다. 아직까지 컴포

넌트에 대한 정의가 분명하지는 않지만 컴포넌트에 대한 여러 가지 정의들을 통해 공통적으로 컴포넌트가 의미하는 특성을 파악할 수 있다. 컴포넌트의 여러 가지 정의에 대해 컴포넌트는 공통적으로 모듈성(modularity), 동적 바인딩, 인터페이스, 아키텍처 의존성, 조립(composition), 주문화(customize) 등의 특성을 갖는다.

모듈성(modularity) : 컴포넌트는 시스템에 대치할 수 있는 부분으로 여러 시스템에 적용 가능한 부품이며 시스템에 매우 독립적이다. 모듈성은 일반성(generalization)과 비례 관계를 가지며 모듈성이 높아지면 가변성이 약해져 적용할 수 있는 범위는 넓어지겠지만 적용할 수 있는 컴포넌트의 크기는 작아진다.

동적 바인딩 : 컴포넌트는 실행 시에 인터페이스를 통해 동적으로 응용 프로그램에 연결될 수 있으며 다른 컴포넌트와도 동적으로 연결된다. 라이브러리 함수가 개발 시스템에서 재사용될 때 재컴파일 되어야 하는 반면 컴포넌트는 실행 시에 개발 시스템에서 재컴파일 없이 동적으로 바인딩 되어 사용될 수 있다.

인터페이스. : 컴포넌트는 블랙박스 형태의 재사용 단위로 개발 시스템에서 컴포넌트를 사용하기 위해서 사용자는 컴포넌트 외부 인터페이스만 알면 되며 내부 구현에 대한 자세한 모듈을 인식할 필요가 없다. 인터페이스 구현 시에 다양한 컴포넌트에 대한 인터페이스는 다른 여러 개발자에 의해 조작할 수 있고 벤더(vendor)들간에 재사용될 수 있도록 하기 위해 표준화 되어야 한다.

아키텍처 의존성 : 컴포넌트 기반 아키텍처는 새로운 컴포넌트를 첨가하거나 대치(replace)하여 기능적인 향상을 이룰 수 있도록 프레임워크 형태로 제공된다. 최근에는 컴포넌트가 특정 아키텍처에 종속적인 특징을 가지고 설계되지만 다른 아키텍처에서도 상호 운영될 수 있도록 하고 있다[2].

조립(composition) : 플러그 앤 플레이(plug and play) 할 수 있는 컴포넌트는 각 컴포넌트의 인터페이스를 통해 조립한다[3].

주문화(customize) : 개발자는 컴포넌트를 사용하여 어플리케이션을 개발할 때 컴포넌트의 속성을 변경하여 개발하려는 시스템에 컴포넌트를 적절하게 삽입하여 사용할 수 있다[4].

III. 컴포넌트 통합을 위한 연결자 모델링 설계

본 논문에서는 분산 시스템 환경을 기반으로 하여 객체 및 컴포넌트 사이의 관련성을 기술하고 상호 연관 동작을 갖는 컴포넌트를 연결하기 위해 그림 1과 같은 역할을 하는 연결자를 설계하고 구현한다. 또한 연결자를 이용하여 분산 시스템 환경에서 원활하게 컴포넌

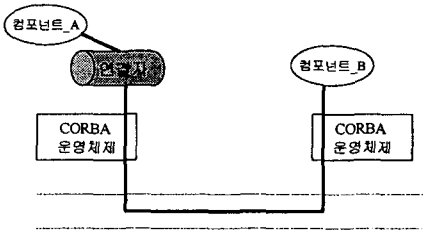

그림 1. 연결자의 역할

트간 연결 및 상호 작용을 지원하기 위하여 그림 2와 같이 컴포넌트, 컴포넌트 스키마, 연결자를 갖는 시스템 모델을 설계하였다.

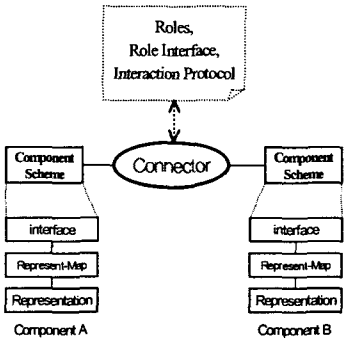

그림 2. 컴포넌트, 연결자, 컴포넌트 스키마

컴포넌트는 클라이언트에게 정보를 제공하는 인터페이스 부분, 인터페이스에 대한 내부 구현 부분으로 연결해주는 표현-지도 부분, 실질적으로 컴포넌트의 행위를 기술한 표현 부분으로 구성하였다.

본 논문에서 Java Beans 환경에서 컴포넌트 통합을 위한 연결자 모델링을 설계하고 구현하기 위해 소프트웨어 아키텍처, 구조적 기술 언어(architectural description language), 객체지향 모델링 기술, 분산 객체의 처리를 위한 참조 모델 등에 기반을 둔다. 이를 위해 Java Beans에 기반을 둔 소프트웨어 아키텍처를 모델링하며 그림 2와 같이 구성된 컴포넌트, 연결자 및 컴포넌트 스키마로 구성된 프레임워크를 설계하고 구현한다. 컴포넌트는 Java Beans 환경에 존재하는 분산 객체의 집약적인 기술로서 인터페이스 집합인 인터페이스 명세 부분, 내부적인 객체지향 스키마인 표현(representation) 부분, 외부적인 객체지향 스키마인 표

현-지도(represent-map)를 포함하고 있다. 연결자는 컴포넌트의 논리적인 관련성을 갖는 통합과 객체의 동적인 관계를 나타내는 컴포넌트의 상호작용을 추상화하며 연결자는 기능(roles), 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.

Ⅳ. 컴포넌트 통합을 위한 연결자 구현

4.1 이벤트통보 연결자 구현

객체의 동작이 동기화 된 수행의 결과로 메소드 호출이 발생되는 표준화된 CORBA 객체 통신 모델링 방법에 반대로, 이벤트는 다양한 객체사이에서 비동기화된 방법으로 객체간에 통신을 한다. OMG의 이벤트 서비스는 표준화된 인터페이스와 publish/subscribe에 기반을 둔 이벤트-참여를 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 채널(channel)로서, 포괄 이벤트(generic event) 통신 방법으로 모델링 된다.

연결자 EventNotification의 명세서에는 이벤트 통보를 집약적으로 수행하기 위해 객체사이의 상호 의존성을 기술하며 객체의 기능(roles), 기능 인터페이스, 상호 작용 프로토콜에 의해 구조화된 형태를 갖게된다. 기능(roles) 부분은 컴포넌트를 구성하기 위해 협력하는 참여자를 말한다. 이벤트 통보 연결자의 경우 "EventPushSupplier", "EventPushConsumer", "EventPullConsumer", "EventChannel"과 같은 4개의 객체 기능이 존재한다. 이러한 객체들은 확장된 수준에서 특정 컴포넌트에 의해 동작되어진다. 또한 각각의 컴포넌트는 서로 같거나 다른 연결자에 의해 하나 이상의 기능이 수행될 수 있으며 각각의 기능에 대해 하나 또는 그 이상의 기능 인터페이스가 기술될 수 있다.

4.2 기능 인터페이스

기능 인터페이스 부분에서는 서로 협력관계를 유지하는 컴포넌트가 지원해야할 서비스 명세를 기술한다. 기능 인터페이스 부분은 그림 3과 같이 OMG에서 지원하는 일반적인 이벤트 명세서에 정의된 IDL 인터페이스 기준을 따르며 다음과 같은 문법 규칙을 가지고 있다.

문법

<Role>.<Module>::<Interface>
{:<Module>::<Interface> }

Connector EventNotification

Role :

EventPushSupplier,
EventPushConsumer,
EventPullConsumer,
EventChannel,

Role Interface :

EventPushSupplier.CosEventComm::PushSupplier;
EventPushConsumer.CosEventComm::PushConsumer;
EventPullConsumer.CosEventComm::PullConsumer;
EventChannel.CosEventChannelAdmin
::ProxyPushSupplier:CosEventComm::PushSupplier;
EventChannel.CosEventChannelAdmin
::ProxyPushConsumer:CosEventComm::
PushConsumer ;
EventChannel.CosEventChannelAdmin
::ProxyPullSupplier:CosEventComm::PullSupplier;
EventChannel.CosEventChannelAdmin::
ConsumerAdmin;
EventChannel.CosEventChannelAdmin::SupplierAdmin;
EventChannel.CosEventChannelAdmin::EventChannel;

그림 3. 이벤트통보 연결자의 기능과 기능 인터페이스

4.3 상호작용 프로토콜

협력관계를 유지하는데 있어서 연결자 내부의 실제적인 동작은 상호작용 프로토콜을 통해 지정되어 진다. 인터페이스 프로토콜은 선행 조건(precondition) 및 후행 조건(postcondition)을 고려하여 협력 관계에 대한 동작과 동작의 순서를 기술한다. 그림 4에서와 같

이 이벤트 데이터 교환(exchanging event data)을 위해서는 4개의 동작으로 표현된다. 인터페이스 프로토콜을 기술할 때 오름차순 순서를 갖는 숫자는 수행되는 동작의 순서를 의미하며, 알파벳 문자의 순서는 동시에 수행되는 병행 동작을 의미한다.

Connector EventNotification

Protocol :

- Interaction Exchanging Event Data :

Precondition :

each role is played by a component using the interaction

Creating EventChannel, Connecting PushConsumer to Channel, . . .

Postcondition :

event data is transmitted from an EventPushSupplier to all registered EventPushConsumer and buffered for EventPullConsumers

Actions :

1 EventPushSupplier sends event to EventChannel

2a EventChannel sends event pushes data to EventPushConsumer

2b EventChannel buffers event for EventPullConsumer

2b/3 EventPullConsumer quries

FventChannel, EventChannel delivers buffered event

- Interaction Creating EventChannel :

. . .

- Interaction Connecting PushSupplier to Channel :

. . .

- Interaction Connecting PullConsumer to Channel :

. . .

- Interaction Connecting PushConsumer to Channel :

Precondition :

an EventChannel and a Consumer component exist

Postcondition :

Consumer is connected to EventChannel according to the push-model

Actions :

1 PushConsumer obtains ConsumerAdmin factory via EventChannel interface

2 PushConsumer obtains ProxyPushSupplier reference via the factory ConsumerAdmin

3 PushConsumer connects to channel via ProxyPushSupplier interface

그림 4. 이벤트통보 연결자의 상호작용 프로토콜

V. 결 론

본 연구는 기존에 객체지향 환경 및 분산 환경 시스템에서 제시된 객체간 및 컴포넌트간의 연결 및 통합을 위한 기법을 기반으로 하여 현재 분산 시스템 환경에서 적용할 수 있도록 활용 범위를 확대하였다. 따라서 본 논문의 특징은 기존의 객체 및 컴포넌트를 위한 연결 기법을 기반으로 하여 연결자와 컴포넌트를 원활하게 연결하기 위한 컴포넌트 스키마를 도입하였으며, 이를 Java Beans 환경에서 분산 컴포넌트간에 연결자를 사용하여 기존의 동기화 방식에 추가적으로 비동기화 방식으로도 컴포넌트 연결을 위한 정보를 주고받을 수 있도록 하였다.

향후에 연구 과제로는 보다 보완 된 상용화된 컴포넌트의 통합 및 연결이 이루어질 수 있도록 하기 위하여, 다양한 응용 분야에 존재하는 객체 및 컴포넌트에 대한 연결 및 통합이 진행되어야 한다.

참고문헌

[1] Joao Pedro Sousa and David Garlan, *"Formal Modeling of the Enterprise JavaBeansTM Component Integration Framework"*, FM '99 : World Congress on Formal Methods, Sep, 1999.

[2] Robert Orfall, Dan Harkey, *Client/Server Programming with JAVA and CORBA*, Guide, John Wiley & Sons Inc.,1997.

[3] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Logensen, *Object-oriented Modeling and Design*, Prentice-Hall International Editions, 1991.

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

[5] Robert Orfall, Dan Harkey, Jeri Edwards, *The Essential Distributed Objects Survival Guide*, John Wiley & Sons Inc., 1996.

[6] Mary Campione, Kathy Walrath, *The Java Tutorial : Object-oriented Programming*, Addison Wesly, 1996.

[7] Ken Arnoldm James Gosling, *The Java Programming Language*, Addison Wesly, 1999.