분산 시스템 환경에서 Java Beans 컴포넌트 통합에 관한 연구

정성옥

광주여자대학교 정보통신학부

A Study on the Java Beans Component Integration in the Distributed System Environment

Sung-Ok Jung

Kwangju women's University, Division of Information Communication

E-mail : sojung@namkyung.kwu.ac.kr

요 약

현재의 소프트웨어 아키텍처에 관한 연구는 컴포넌트 집합과 같은 소프트웨어 시스템을 구성하는 객체 또는 컴포넌트의 상호 동작 및 관련성을 보다 효과적으로 연결할 수 있는 다양한 기법이 제시되고 있다. 본 논문에서는 Java Beans에 기반을 둔 분산 시스템 환경에서 객체와 객체간에 관련성을 모델링하기 위해 컴포넌트, 연결자 및 컴포넌트 스키마로 구성된 구조화된 모델을 제시하고 구현한다. 특히 Java Beans에서 객체간의 관련성을 모델링하기 위한 연결자의 구성에 중점을 둔다. 본 연구에서 제시된 연결자 모델은 Java Beans기반 분산 시스템 환경에서 다양한 객체간의 의존성을 명확하게 표현하는데 효과적이며 분산되어 있는 컴포넌트를 정형화된 방법으로 통합할 수 있는 효과를 가진다.

ABSTRACT

This Current research for software architecture views and models a software system as a set of components and connectors. Components are abstractions of system level computational entities, connectors are abstractions of component interrelationships. In his paper, we focus attention on connectors for the Java Beans-based systems that are built using object integration technologies like CORBA. We present connector model in Java Beans-based system for object-oriented component integration. We start with a discussion of related work of software architecture research and of object-oriented modeling that focuses on the description of component collaborations. 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. 서 론

현재의 소프트웨어 아키텍처에 관한 연구는 컴포넌트 집합과 같은 소프트웨어 시스템을 구성하는 객체 또는 컴포넌트간의 상호 동작 및 관련성을 보다 효과적으로 연결할 수 있는 다양한 기법들이 제시되고 있다. 따라서 이러한 객체 또는 컴포넌트를 보다 효과적으로 연결할 수 있는 연결자들의 모델을 설정하고 조명하는데 집중되어 있다[1]. 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 CORBA 객체의 특성을 기술할 수 없다. 따라서 추상화와 CORBA 객체간의 관련성을 일반적인 클래스 관련성을 이용하여 표현하는 데는 새로운 기법이 적용되어야 한다. 본 논문에서는 Java Beans에 기반을 둔 분산 시스템 환경에서 객체와 객체간에 관련성을 모델링하기 위해 컴포넌트, 연결자 및 컴포넌트 스키마로 구성된 구조화된 모델을 제시하고 구현한다. 특히 Java Beans 환경에서 객체간의 관련성을 모델링하기 위한 연결자의 구성에 중점을 둔다. 이를 비동기화 컴포넌트간 통신 모델링을 위한 객체, 사건-서비스 명세에 기술된 이벤트 통보 연결자를 구현한다. 본 연구에서 제시된 연결자 모델은 Java Beans 기반 분산 시스템 환경에서 다양한 객체간의 의존성을 명확하게 표현하는데 효과적이며 분산되어 있는 컴포넌트를 정형화된 방법으로 통합할 수 있는 효과를 가진다.

II. Java Beans 컴포넌트와 CORBA와의 결합

객체지향 프로그래밍 기법을 적용하여 응용 프로그램을 개발해도 서로 다른 기계와 운영체제에서 최종적으로 목적기계 코드를 연결하는 것은 어려운 일이다. 따라서 서로 다른 시스템 사이에서 서로 다른 언어로 작성된 객체를 사용하는 것은 곤란하며 또한 상위 객체로부터 상속받을 때도 슈퍼 클래스까지 접근해야 하는 어려움을 가지고 있다. 그러므로 프로그램 내용이 변경되면 모든 소스 파일을 다시 컴파일하고 링크해야 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해 컴포넌트 방식으로 응용 프로그램을 작성하는 기법이 출현하게 되었다. 컴포넌트는 자신이 어플리케이션 개발에 필요한 환경을 갖는 하나의 독립된 개체로서 활동한다. 이 컴포넌트는 인터페이스를 외부 사용자에게 제공하여 사용자는 인터페이스를 통하여 컴포넌트를 사용할 수 있으며 최종 응용 프로그램을 개발할 때도 기존에 제작된 컴포넌트를 소프트웨어 프레임워크에 맞추어 개발하는 형태로 진행되고 있다. 이러한 컴포넌트를 이용해서 어플리케이션을 구성할 때 어플리케이션과 컴포넌트가 동일 주소 공간에 있는 것이 기본이고 이때 가장 최적의 성능을 발휘할 수 있다. 그러나 현재의 클라이언트/서버 환경으로 운영되는 네트워크 환경에서 운영되는 분산 컴포넌트들은 서로 다른 주소 공간, 네트워크를 통한 이기종 기계에서 동작되므로 이것을 효율적으로 묶는 인프라가 필요하게 되는데 이러한 컴포넌트간 통신 인프라를 제공하는 모델로 OMG의 CORBA, 마이크로소프트의 DCOM, SUN의 RMI 등이 있다. 기존의 소프트웨어 컴포넌트 개념으로 Visual Basic 컴포넌트와 객체지향 파스칼을 이용할 Delphi를 들 수 있다. 이러한 것은 윈도우 환경에서 사용될 수 있다. 최근에는 Java 언어를 이용하여 기존에 작성된 컴포넌트를 발전시켜 새롭게 개발된 것이 Java Beans로서 Java 언어를 사용하여 Java 컴포넌트를 구축할 수 있는 시스템 도구이다. Java Beans는 자바 언어의 플랫폼 독립성을 활용하여 이 기종 시스템 환경에서 동작되는 보편적인 컴포넌트로서 성장을 기대하고 있다. Java Beans의 주요 특징으로는 내부 관찰자(introspection)를 통해서 Beans의 속성, 메소드, 이벤트를 검색할 수 있다. 또한 디자인 패턴이나 BeanInfo 클래스를 사용하여 컴포넌트 내부를 관찰할 수도 있다. 그리고 속성 Beans는 설계 시 노출되어 있으므로 사용자의 요구에 따라 주문화(customization)가 가능하다. 서론 다른 Beans 사이에서 정보를 주고받기 위한 통신을 설정하기 위해서는 이벤트를 사용한다. 이벤트를 받기 원하는 Beans는 이벤트 소스 Beans에 등록할 수 있다. 지속성을 위해 객체 직렬화(object serializaton)를 사용해 Beans로 하여금 상태 저장 후 이후에 복구할 수 있도록 해준다 Java Beans 자체로는 단일한 가상 기계 상에서 구현되고 동작되게 되어있다. 이 Beans들은 여러 통신 프로토콜을 사용해 서버와 통신하게 된다. Java Beans 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 Beans(EJB)의 구상이 자연스럽게 등장하였다[2]. CORBA Beans는 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[3]. 이러한 CORBA 컴포넌트는 Java Beans와 함께 결합되었을 때의 발생되는 상승 효과로 인해 현재 OMG에서 CORBA 컴포넌트의 기본 모델로 Java Beans를 채택하고 있다. 실제 CORBA는 Java Beans 컴포넌트에 Beans를 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[4]. Java Beans 컴포넌트를 CORBA 컴포넌트의 기본 모델로 채택하여 지원하고 있으며 이러한 모델의 요소로 Java Beans

설계 패턴, 이벤트 속성, 패키지, 메타 데이터, 도구 등이 있다. 또한 현재의 주요한 개발 환경인 비주얼 도구의 패러다임을 그대로 적용하고 있다. 이러한 도구를 가지고 속성(property) 편집기를 호출할 수 있고 컴포넌트간 상호 연결 패러다임을 제공하며 CORBA 컨테이너에서 컴포넌트들이 계층적으로 구축될 수 있고 컨테이너는 하나 이상의 컨테이너를 구축할 수 있다.

III. 컴포넌트 통합을 위한 연결자 모델링 설계

객체지향 소프트웨어 시스템은 서로 연관되고 상호 작용을 하는 많은 객체들로 구성된다. 객체지향 모델링 및 설계 단계에서 객체에 대한 집약적인 기술은 일반적으로 클래스의 인스턴스들과 객체간의 관계를 포함하고 있다. 일반화 및 명세화 등과 같은 클래스에 대한 집약적인 기술은 엔티티-관련성의 모델링과 객체지향 프로그래밍 언어 등에 기반을 두고 있다. OMG의 CORBA에 기반을 둔 분산 객체 시스템도 서로 연관되고 상호 작용을 하는 많은 객체들로 구성되어 있다. 하지만 COBRA에 기반을 둔 시스템을 모델링하고 설계하기 위해서는 일반적으로 객체지향 모델링 기법을 적용하는 클래스에 기반을 둔 모델링 언어를 직접 적용하는데는 많은 문제점을 가지고 있다. 클래스 추상화 기법은 CORBA의 객체 특성을 집약적으로 기술하는데 있어서 인터페이스 부분과 구현 부분을 엄격하게 구분하거나 CORBA 객체의 단위(granularities)를 엄격하게 구분하는데 문제점을 가지고 있다. 따라서 CORBA 객체간의 관계를 기술하고 추상화를 위해 일반적으로 객체지향 패러다임에서 지원하는 클래스간의 관련성을 지원하기 위해서는 보다 많은 연구가 진행되어야 한다. CORBA 객체와 객체 사이의 관련성을 집약적으로 기술하기 위해서는 컴포넌트 통합을 위한 모델링 기법이 제시되었다. 본 논문에서는 본 논문에서는 CORBA 객체 사이의 관련성을 위한 연결자의 추상화 기법을 이용하여 Java Beans 시스템 컴포넌트의 통합을 위해 컴포넌트, 연결자(connector) 및 컴포넌트 스키마(component scheme)로 구성된 구조화된 모델링 기법을 설계하고 구현한다. 특히 Java Beans 환경에서 객체간의 관련성 및 통합을 위한 연결자의 구성에 중점을 둔다. 기존의 OMG의 OMA(Object Management Architecture)는 비동기적인 기법으로 객체사이의 통신, 이벤트-서비스 명세에 이벤트 통보(notification)에 관한 모델링 연결자가 설계되고 구현되었다. 본 논문에서 구현된 연결자 모델링 기법은 Java Beans 기반 시스템에서 다양한 객체사이의 의존성 관계를 명확하게 표현하는데 용이할 것을 기대된다. 본 논문에서 Java Beans 환경에서 컴포넌트 통합을 위한 연결자 모델링을 설계하고 구현하기 위해 소프트웨어 아키텍처, 구조적 기술 언어, 객체지향 모델링 기술, 분산 객체의 처리를 위한 참조 모델 등에 기반을 둔다. 이를 위해 Java Beans에 기반을 둔 소프트웨어 아키텍처를 모델링하며 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 표현-지도(representation-map)를 포함하고 있다. 컴포넌트 인터페이스는 클라이언트에게 제공되는 컴포넌트 서비스를 기술하고 있으며 CORBA-IDL(Interface Description Language)과 같은 인터페이스 기술 언어를 이용하여 표현된다. 표현-지도 부분은 표현 부분과 인터페이스 명세 부분을 연관시키며 객체지향 스키마로서 표현된다. 본 논문에서 설계한 계층적 구조를 가진 구조는 Java Beans 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.

IV. 컴포넌트 통합 연결자 모델링 구현

객체의 동작이 동기화된 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 공

급자는 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 "EventNotification"의 명세서에는 이벤트 통보를 집약적으로 수행하기 위해 객체사이의 상호 의존성을 기술하며 객체의 기능, 기능 인터페이스, 상호 작용 프로토콜에 의해 구조화된 형태를 갖게된다. 기능부분은 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. 기능 인터페이스 부분에서는 서로 협력 관계를 유지하는 컴포넌트가 지원해야할 서비스 명세를 기술한다. 기능 인터페이스 부분은 다음과 같이 OMG에서 지원하는 일반적인 이벤트 명세서에 정의된 IDL 인터페이스 기준을 따르며 다음과 같은 문법 규칙을 가지고 있다.

문법

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

기능 부분과 다른 기능의 인터페이스 사이에서 서로 다른 방향을 가진 사용 관련성이 존재한다. 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. "Event supplier"는 첫 번째로 "EventChannel"에 이벤트를 전송하고 "EventChannel"은 전송 받은 정보를 모든 등록되어 있는 "PushConsumer"에게 전파(propagate)시킨다. 동시에 모든 "PullConsumer"를 위하여 전송 받은 정보를 임시 저장한다. 또한 "Event Supplier"는 "EventChannel"과 모든 "PushConsumer"사이에 있는 상호 작용과 독립적으로 저장되어 있는 Event들을 요청한다. 또한 이벤트 데이터 교환 상호 작용 프로토콜 역시 각각 기능이 컴포넌트에 의해 동작되어져야 하는 선행 조건을 지정한다. 이러한 선행 조건 "Connecting Push-Consumer to Channel"과 같은 다른 상호 작용 프로토콜을 이용하여 채워질 수 있다. 이러한 것은 상호 작용 프로토콜 사이에서의 상호 의존성을 기술한다. 이러한 경우에 이벤트 데이터의 교환은 OMG 이벤트 서비스에 정의된 것처럼 양방향으로 이벤트 채널에 등록되어 있는 이벤트 소비자를 요구한다. 서로 다른 수준에서 상호 작용을 기술하기 위하여 상호 작용 프로토콜 명세서는 특성화되어야 한다. 예를 들어, 이벤트 데이터의 교환은 "pulling consumer"를 가진 것으로부터 "pushing consumer"를 가진 상호 작용을 분리하기 위하여 두 개의 상호 작용으로 분리될 수 있다. 그리고 나서 자세한 명세서는 기법들을 차별화 할 수 있으며 "pulling consumer"는 pull() 과 try_pull() 동작을 이용할 수 있다. 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.

V. 결 론

본 연구는 기존의 객체 및 컴포넌트를 위한 연결 기법을 기반으로 하여 연결자와 컴포넌트를 원활하게 연결하기 위한 컴포넌트 스키마를 도입하였으며 이를 CORBA 환경에서 Java Beans에 적용하였다. 또한 연결자를 사용하며 기존의 동기화 방식에 추가적으로 비동기화 방식으로도 컴포넌트 연결을 위한 정보를 주고받을 수 있도록 하였다. 본 논문의 특징은 컴포넌트 통합 및 연결을 요구하는 사용자는 컴포넌트 스키마 정보만을 정형화된 방법으로 기술하여 원하는 컴포넌트를 생성할 수 있으며 비동기화 방식으로 컴포넌트 연결을 위한 정보를 주고받을 수 있는 장점을 가진다. 향후에 본 연구를 보다 보완하여 다양한 응용분야에 존재하는 객체 및 컴포넌트에 대한 연결 및 통합을 진행하고자 한다.

참고문헌

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

[2]R. Monson, *Enterprise Java Bean*, O'Reilly, 2000

[3]Kurt Wallnau, Nelsin Weiderman, *Distributed 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]Joao Pedro Sousa and David Garlan, "*Formal Modeling of the Enterprise Java Beans TM Component Integration Framework*", FM '99 : World Congress on Formal Methods, Sep, 1999