2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

분산 컴포넌트 시스템에서 컴포넌트 구현 재사용에 관한 연구

임 성 진*, 이 상 준***, 서 성 채**, 김 병 기**
* 전남대학교 정보통신 협동과정
** 전남대학교 전산학과
*** 서남 대학교 정보통신학과
e-mail : sjlim@superse.chonnam.ac.kr

Research about component implementation reuse in distributed component system

Seong-Jean Lim* , Sang-Joon Lee***, Sung-chae Seo**,
Byung-Gi Kim**
*Interdisciplinang Program of Information and Telecommunication
chonnam national university
**Dept of Computer Science, chonnam national university
**Dept of Computer Science, Seo-Nam national university

요약

컴포넌트의 조립만으로 소프트웨어를 생산하는 컴포넌트 기반 개발 방법론이 크게 대두되고 있다. 잘 정의된 인터페이스와 정형화된 컨텍스트를 통한 컴포넌트의 조립을 통한 기존의 시스템과의 연계를 통해 사용자 요구사항을 만족하는 소프트웨어의 개발이 가능해 졌다. 분산 컴포넌트 환경에서 클라이언트 컴포넌트와 서버 컴포넌트와의 결합이 기존의 컴포넌트의 변경이 없이 컴포넌트 구현을 재사용 한다면 어플리케이션의 구현은 시간과 더불어 유지 보수 비용의 절감에 커다란 효과를 가져올 수 있을 것이다.

1. 서론

소프트웨어 부품을 재사용 하여 개발비용 및 개발 시간을 단축시키고자 하는 개발 방법이 오래 전부터 연구되고 시도되어 왔고, 최근 들어 컴포넌트 기반의 소프트웨어 개발(Component-Based Development) 방법이 크게 대두되고 있다. 컴포넌트는 특정 기능을 수행하기 위해 독립적으로 개발되고 잘 정의된 인터페이스를 가지며 다른 컴포넌트와 조립되어 응용시스템을 구축하기 위해 사용되는 소프트웨어의 단위이다. 그러나 컴포넌트의 조립 생산에는 모듈화, 표준화, 독립화 등의 문제를 효율적으로 해결하지 못하여 일부 기능을 재사용 하는데 그치고 있다. 잘 조립된 컴포넌트를 사용하여 소프트웨어를 재사용하기 위한 핵심 기술은 표준화된 인터페이스를 갖추는 기술, 컴포넌트들을 유연하게 조립하는 기술등이며 초기 소프트웨어 개발 요구사항 분석 시 재사용 될 컴포넌트와의 효율적인 조립 여부를 판가름하여 재사용 컴퍼넌트를 사용해야 할 것이다.

2. 컴포넌트 기술

소프트웨어 컴포넌트는 서드 파티들에 의한 조합을 목적으로 독립적으로 배포될 수 있다.[1] 즉 컴포넌트는 표준화된 인터페이스를 바탕으로 독립적 재사용이 가능한 단위를 말한다. 컴포넌트 모델은 일반적으로 시스템의 서버측 재사용 아키텍처를 말한다.

2.1 소프트웨어 컴포넌트

소프트웨어 컴포넌트는 구성하여 통합할 수 있고, 나눌 수 있으나 수정은 할 수가 없다.[3] 클래스 라이브러리나 다른 소스 코드들은 수정할 수 있는 형태로 존재하기 때문에 소프트웨어 컴포넌트의 정의 안에 넣을 수가 없다. 소프트웨어 컴포넌트는 이진 형태의 포맷으로 이루어진 소프트웨어의 독립적인 단위이기 때문에 본질적으로 논리적으로나 물리적으로 분산객체이다. Brad J.Cox는 소프트웨어 컴포넌트를 소비자들에게 전달하기 위한 하나의 방법으로 코드의 패키지화한 기술이라고 여기고 있다고 했다.[3] 소프트웨

어 컴포넌트는 실행 파일이나(EXEs) 동적 링크 라이브러리(DDLs)와 같은 형태로 구현이 되어진다. 소프트웨어 컴포넌트가 실행이 되어 지면 하나나 다수의 인터페이스를 가진 객체들을 노출시킨다. 컴포넌트에 의해 노출된 객체들은 객체들의 인터페이스를 통해 접근할 수 있다.

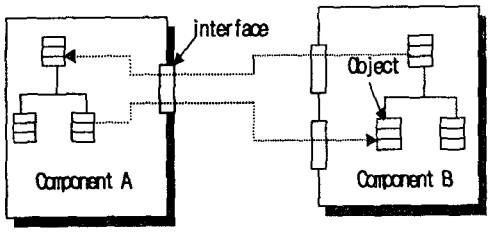


그림 1 .컴포넌트, 객체들 ,인터페이스

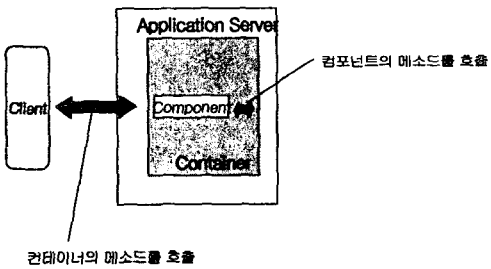
2.2 컴포넌트 기술 구조

컴포넌트 기술 구조는 다음과 같은 세 가지 기본 구조를 가지게 된다.

- 컴포넌트 : 표준화된 인터페이스를 가지고 있으며 표준화된 컨텍스트를 가지게 된다. 컨텍스트란 컴포넌트가 제공하는 기능이나 역할등의 내용이며 인터페이스는 IDL(Interface Definition Language)을 사용하여 명세화 된다.

- 컨테이너 : 서버측 런타임 환경으로서 컴포넌트를 실제적으로 구현하게 된다. 클라이언트에서 요청한 서비스를 제공하기 위해 컴포넌트의 참조 서비스를 제공하며 컴포넌트 인스턴스의 생성 및 소멸을 관리한다.

-어플리케이션 서버 : 네이밍(Naming) 서비스 및 트랜잭션 서비스를 제공하고 컨테이너 관리등도 함께 한다.



(그림 2) 컴포넌트 모델의 기본 구조

2.2 컴포넌트 모델의 종류

컴포넌트 관련 기술의 표준화는 OMG, Microsoft, Sun Microsystems의 세 조직을 중심으로 이루어지고 있다.

Microsoft 사의 COM/DCOM등은 Windows 플랫폼 기반이며, Sun은 Java 기술을 기반으로 하는 EJB를 제안하고 OMG 에서는 분산 객체 기술의 표준인 CORBA를 기반으로 컴포넌트 기반 시스템의 구조를 제안한다.

2.2.1 OMG의 CORBA Component Model(CCM)

OMG(Object Management Group)에서 분산 컴퓨팅과 객체지향 기술을 하나로 합친 표준 아키텍처에 의해 제안되었으며 분산객체 기술의 선도 역할을 하고 있다.지금 현재로는 CORBA 3.0 스펙이 재정되고 있으며 다음과 같은 특징이 지원되고 있다.

· 인터넷과의 통합

· QoS(Quality of Service의 확보)

· CCM(CORBA Component Model) 아키텍처

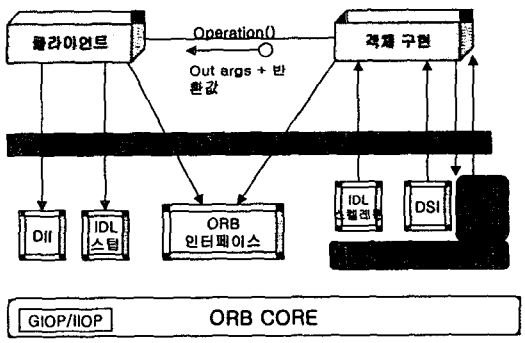
CORBA 클라이언트는 서버기계에 실제로 구현한 객체에 대한 프록시(Proxy) 역할을 하는 스텁을 생성한다. 클라이언트는 인터페이스를 구현한 객체와 직접 상호 작용하듯이 스텁에 접근하게 된다. 스텁은 클라이언트에 설치된 ORB(Object Request Broker)소프트웨어를 통해 인터페이스를 호출한다. CORBA 서버에서 ORB 소프트웨어는 인터페이스 호출을 생성된 스켈레톤에 자동으로 넘겨 준다. 스켈레톤은 BOA(Basic Object Adapator)를 통해 ORB 소프트웨어와 통신을 하게 된다. 스켈레톤은 BOA를 사용해 객체를 등록하고 여기에 객체의 범위와 객체가 인스턴스화해 클라이언트와 반응하게 되는 시기등을 결정한다.

CORBA 컴포넌트의 주요 부분은 다음과 같다.

· 트랙잭션, 보안 , 영속성 및 인터페이스 제공, 이벤트 처리 패키지 등을 컨테이너 환경에 제공

· EJB(Enterprise JavaBean)와의 통합

· CORBA 컴포넌트 소프트웨어 시장을 형성할 수 있도록 하기 위한 소프트웨어의 분산 형태



(그림 3) 기본적인 CORBA ORB의 동작 원리

CORBA 컴포넌트의 컨테이너 환경은 기존의 서비스에 비해 더 추상적인 상위 단계에서 패키지화한 뒤에 제공된다. CCM에서는 EJB가 CORBA 컴포넌트와 동

일하게 동작하며, CORBA 컴포넌트 컨테이너에 설치할 수 있다는 점이다. 분산 객체 기술로서 CORBA가 가지고 있는 장점중 하나는 OMG 내부에서 객체 지향 설계 및 디자인을 담당할 UML(Unified Modeling Language)을 표준으로 정하고 UML과 객체를 연동할 규격을 내놓았다.

2.2.2 Sun Microsystems의 Enterprise JavaBeans(EJB)

EJB는 멀티티어의 분산형 객체지향 자바 어플리케이션을 개발하고 보급하기 위한 컴포넌트 아키텍쳐이다. EJB를 사용함으로써, 복잡한 분산 객체 프레임워크에 대한 작성 없이 확장성과 신뢰성이 높고 안전한 어플리케이션 작성이 가능하게 된다. 시스템 수준의 세부적인 기능을 숨김으로써, 어플리케이션의 작성을 용이하게 하며,한번 작성된 어플리케이션들은 다른 환경에서도 바로 실행 되게 한다. 따라서 구현된 엔터프라이즈 EJB 컴포넌트는 복수의 플랫폼에서 설치되어 사용될 수 있다. EJB는 자바 API 와의 호환성을 제공하고, 엔터프라이즈 자바빈과 자바외의 언어로 구현된 어플리케이션과의 상호연동 또한 지원을 하며, CORBA 프로토콜과의 호환도 지원한다. EJB는 분산 트랜젝션 지향 엔터프라이즈 어플리케이션의 컴포넌트로서 한 클래스가 하나의 빈이 된다. EJB의 모든 엔터프라이즈 빈은 빈의 생성과 관련된 기능을 가지고 있는 홈 인터페이스와 비즈니스 로직을 가지고 있는 리모트 인터페이스로 이루어져 있으며, 클래스가 지원하는 기능과 상태, 자료의 처리 능력에 따라 크게 세션 빈과 엔티티 빈으로 나누어진다. EJB는 클라이언트와 컨테이너 사이, 그리고 엔터프라이즈 빈과 컨테이너 사이에 계약을 자기고 그 계약 내에서 각 역할들을 지원한다. 클라이언트는 엔터프라이즈 자바 빈의 홈 인터페이스와 리모트 인터페이스를 접근하고, 객체 식별자를 얻는다. EJB의 홈과 리모트 인터페이스는 RMI(Remote Method Invocation)에 의해 운영됨에 따라, EJB의CORBA와동성은CORBA객체와 RMI/IIOP(Internet INter-ORB Protocol)을 사용해 상호 작동 할 수 있어 연동하는데 불편이 없다. EJB 스펙이 1.1로 바뀌면서부터 XML 과의 연동을 취함으로서 EJB는 이식성 있는 모든 비즈니스 로직에 대해 표준을 정의하고 XML은 이식성 있는 데이터에 대해 표준을 정의하게 되었다.

3. 컴포넌트 구현 재사용

소프트웨어 컴포넌트 명세적 (component -specific)인 재사용은 컴포넌트가 구현하는 것뿐만 아니라 소프트웨어 컴포넌트들의 특별한 면을 재 사용할 수 있도록 해야 한다. 비록 개발자들은 소프트웨어 컴포넌트를 만들기 위해 코드를 재사용 하지만, 여기서는 독립적인 단위로서의 소프트웨어 컴포넌트에 관해 이야기한다.

3.1 소프트웨어 컴포넌트의 결합

소프트웨어 컴포넌트의 결합은 한 컴포넌트의 입력 인터페이스(incoming interface)와 출력 인터페이스(outgoing interface)와의 결합이다. [3] 한 객체의 입력 인터페이스는 다른 객체의 출력 인터페이스의 호출에 의해 사용이 되어진다. 결합은 컴포넌트의 속성에 영향을 미치는 매개변수의 한 조이다.

소프트웨어 컴포넌트의 통합은 이들 컴포넌트들간의 호출자 관계들의 구성을 필요로 한다. 소프트웨어 컴포넌트를 구성하기 위해 CORBA 나 DCOM 과 같은 분산, 객체 지향적인 아키텍처를 사용할 때는 클라이언트 컴포넌트는 CORBA의 stub이나 proxy 와 같은 대리인(surrogate)들을 사용하거나 서버 컴포넌트에 접근하기 위해 CORBA Dynamic Invocation Interface 나 Microsoft OLE Automation를 사용하게 된다. 동적 접근 방법은 일반적인 메소드 호출들을 구성하기 위해 실행시에 인터페이스에 대한 정보들을 함유하고 있어야 한다. 그러나 일반적인 메소드 호출을 위해서는 클라이언트 컴포넌트가 이에 관한 정보들을 미리 알고 있어야 하는 어려움이 있다.그러므로 동적 접근 방법을 사용하여 컴포넌트에 접근을 한다 하여도 클라이언트 컴포넌트는 내부적으로 접근 하고자 하는 컴포넌트에 관한 정보를 담고 있어야 한다. 다른 컴포넌트에 의해 표현된 객체들의 정적인 관점들을 포함하기 위해서는 이들이 똑 같은 개발자에 의해 개발이 되어졌거나, 클라이언트 컴포넌트는 접근하고자 하는 서버 컴포넌트에 관한 사전 정보를 가지고 있어야 한다. 그러나 실제로 자체 개발된 컴포넌트를 가지고 기존에 이미 구입한 컴포넌트와 결합하거나, 서버 컴포넌트와 클라이언트 컴포넌트는 동시에 개발되거나 똑 같은 표준이나 명세아래에서 개발되어야 한다는 요구사항이 동반된다.

3.2 컴포넌트 연결자

서버 컴포넌트와 이에 접근하고자 하는 클라이언트 컴포넌트의 이런 구현 재사용의 복잡도를 감소시키기 위해 컴포넌트 연결자를 제안한다. 컴포넌트 연결자는 컴포넌트에 의해 노출된 객체들간의 호출자 관계를 구성한다. 컴포넌트 연결자는 다른 컴포넌트들의 명세적인 관점을 표현하고 CORBA의 stub과 같은 역할을 한다. 그러나 CORBA의 stub과 다른 점은 클라이언트 컴포넌트의 구현의 변경이 없이 동적인 재구성을

할 수 있다는 점이다. 클라이언트 컴포넌트는 서버 컴포넌트의 입력 인터페이스에 접근 할 수 있다. 컴포넌트 연결자의 각 인터페이스 구성 요소들은 서버 컴포넌트에 의해서 제공되어지는 하나나 다수의 구현들과 사상된다.

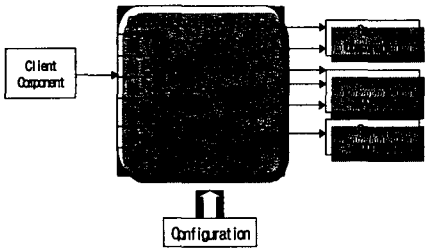


그림 4 . 컴포넌트 연결자의 구조

3.2.1 컴포넌트 연결자 구성

컴포넌트 연결자는 다음과 같은 단계에 의해 구성이 되어진다.

1) 인터페이스 정의
2) 인터페이스 연결
3) 컴포넌트 연결자의 생성

두 번째 단계에 있는 인터페이스 연결의 중요한 면은 매개 변수와 리턴 타입의 입력이다. 다음과 같은 두 개의 컴포넌트가 있다고 하자.

Component A

Incoming Interface

```
Interface Customer {
  String Name();
  Name (string Name);
  long CustomerID;
};
```

Outgoing Interface

```
Interface Store {
  storeCustomer (Customer cust);
};
```

Component B

Incoming Interface

```
struct strucst {
  long CustID;
  char* Name;
};
Interface StoreCust {
 store(strcust cust);
 strucst load(long CustID);
};
```

컴포넌트 A의 출력 인터페이스와 컴포넌트 B의 입력 인터페이스가 연결된다고 하면 함수인 Store::storeCusto
mer(...)는 StoreCust::store(...) 함수에 연결이 되어야 한다.

```
(A) c1=[Create Customer]
(A) // 속성들의 정의
(B) sc1=[Create struct struct]
(B) sc1.Name=Str2PChar(c1.Name())
(B) store(sc1)
```

위의 예제에서 보는 것처럼 컴포넌트 연결자는 컴포넌트 A의 출력 인터페이스와 컴포넌트 B의 입력 인터페이스에 사상이 된다. 만약 컴포넌트 A를 위해 다른 서버 컴포넌트가 선택된다면 또 다른 컴포넌트 연결자가 생성이 될 것이다.

4. 결론

분산환경에서 클라이언트의 컴포넌트와 서버 컴포넌트의 효율적인 합성을 연구해 보았다. 이런 개발로 컴포넌트의 재사용으로 인한 비용/개발 시간의 절감 효과와, 위치 투명성으로 인해 가져오는 장점들, 클라이언트와 서버의 효율적인 연결 관리 적절한 컴포넌트 배치를 통한 성능 향상등 분산 컴포넌트 환경에서 개발에 강력함을 더해 주기 위해 수행 되었다. 앞으로 이런 컴포넌트 연결자의 자동화 도구의 실현으로 좀 더 빠르고 강력한 클라이언트/서버 환경을 구축하고자 한다.

[참고 문헌]

[1]Clemens Szyperski ,"Component Software", Addison Wesley ,1998

[2] B.J.Cox,"Object-Oriented Programming-An evolutio
nary approach , Addison Wesley

[3]H.D.Hofmann "Componentware-Integration of software components in Distributed computing environment" M.Sc Thesis, Cork Institute of Technology ,http://members.tripod.com/~hofmann,

[4]http://www.javasoft.com/ejb,"EJB 1.1 specification"

[5]E.Roman, Mastering Enterprise Java Beans, John Wily & Sons, 1999

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

[7]Booch, G., "Object Solutions-Managing the Objet Oriented Project", Addison-Wesley

[8] Jon Siegel,"CORBA 3 Fundamentals and Programming", John Wiley & Sons, 2000