韓國 OA學會 論文誌
第7卷 第4號, 2002. 12.

2002-7-4-1-6

이형 분산 컴포넌트 플랫폼간 상호 운영성 보장에 대한 연구

장 연 세*

A Study on the Interoperability between heterogeneous Component Platform

Yeun-sae Jang*

요 약

급변하는 IT 환경에서 시스템의 재사용성을 높여 라이프타임을 증가시키고 비용을 절감하기 위한 다양한 노력들이 이루어져 왔다. 구조적 프로그래밍 기법에서는 모듈에 기반한 아키텍처를 활용하여 생산성을 향상시켰다. 그러나 모듈들은 단순히 호출 빈도를 높일 뿐, 성장이나 진화를 하지 못하는 한계 상황에 직면하게되었다. 객체지향 기법은 클래스들을 상속시키거나 메소드를 재정의 함으로써 시스템의 성장과 진화를 가능케 하여 구조적 프로그래밍 기법의 한계를 극복하였다. 최근 CORBA, COM+와 EJB/J2EE 같은 분산 처리 기술과 객체 지향 기법이 융화되어 생성된 컴포넌트 아키텍처는 고도의 재사용성이나 라이프타임의 증가뿐만 아니라 플러그-앤-플레이(Plug-&-Play)도 지원한다. 그러나 이제 컴포넌트를 구축하는데 국한된 문제가 발생하는 것이 아니라 컴포넌트 플랫폼간 연동의 문제가 제기되고 있다. 본 연구에서는 이러한 문제를 해결하기 위해 이형의 컴포넌트 플랫폼간 상호 운영성 보장 방안을 제안한다.

Abstract

There has been several meaning full efforts to save costs and expand the life-time of a system in changeful IT circumstance. It was a module-based architecture that empower productivity at structured programming. It couldn't grow nor evolve, but could raise only calling frequency of module. But OOP or OO-method overcome limit of structured programing by class inheritance and/or overloading and/or over-riding. A component centric architecture, what is mixture of distributed systems, like CORBA, COM+ or EJB/J2EE with OOP, can support not only high reusability or expansion of life-time but also Plug-&-Play between component. In now day, It is not problem of component building but problem of interoperability between heterogeneous CBD Platform. At this study, the enhanced referential component architecture will be suggested.

* 수원과학대학 인터넷정보과 겸임 교수

논문접수 : 2002. 9. 23
심사완료 : 2002. 12. 7

Ⅰ. 서론

1990년 이후 중앙 집중적인 컴퓨팅 환경에서 벗어나 여러 대의 컴퓨터를 이용해 일정한 역할을 분배하여 네트워크 상에서 상호 협력하여 작업함으로써 언제 어디서나 필요한 정보를 얻을 수 있는 컴퓨터 시스템인 분산 컴퓨팅(Distributed Computing) 환경이 시작되었다. 소프트웨어 부문에서도 이러한 환경을 현실적으로 가능하게 하는 다양한 미들웨어(Middle/Ware)들이 발표된 후로 기존의 단순 형태에서 벗어나 분산 환경에서 다양한 서비스를 제공하는 분산 시스템이 활발히 전개 되었다.

분산 컴퓨팅 환경에서의 클라이언트/서버 기술은 인터넷 기술과 객체지향 패러다임이 정보기술의 모든 분야에 적용되면서 분산 객체 컴퓨팅(Distributed Object Computing)으로의 전환이 이루어지고 있다. 분산 객체 컴퓨팅은 단지 기술만의 변화가 아닌, 오늘날의 컴퓨팅 환경의 문제를 해결하기 위한 패러다임이며, 소프트웨어 공학 프로세스를 확장하고 적용시키는 촉진제인 것이다. 이러한 기술 발전에 대한 소프트웨어 개발 기술의 성공 요소는 재사용 아키텍처를 기반으로 하는 객체지향 기술(Object Oriented Technology)로서, 이에 대한 효율적인 활용 전략이 주요 이슈로 대두되고 있다.

최근에는 객체지향 기술에 근간을 둔 컴포넌트 기반의 소프트웨어 개발 기술(CBD: Component Based Development)이 정립되어 가고 있으며, 또한 컴포넌트 기술의 프로임워크를 제공하는 업체간의 표준 전략은 기존의 DCE(Distributed Computing Environment) 기반의 환경에서, 객체 기술과 인터넷 기술이 접목된 MS사의 COM+와 OMG(Object Menegement Group)의 CORBA, SUN사의 EJB(Enterprise Java Bean) 및 J2EE(Java 2 Enterprise Edition)로 서로 보완적 관계를 유지하면서 발전하고 있다. CBD는 단순히 소프트웨어 개발 생산성에 목표를 둔 것이 아니라 비즈니스 객체를 통한 업무 재사용까지도 의미한다.

본 연구에서는 실질적 컴포넌트 프레임워크로 인식되고 있는 CCM(CORBA Component Model), EJB 및 J2EE와 COM+를 통해 구축할 수 있는 컴포넌트의 아키텍처를 검토하고 이를 바탕으로 상호 운영성 보장을 위한 방안을 제시하고자 한다. 이를 위해 제 2장에서는 컴포넌트의 형상과 세가지 프레임워크를 비교 분석하고 제 3장에서 각 프레임워크간 상호 운영성 보장을 위한 참조 모델을 제시한후 제 4장에서 본 연구의 결론과 향후 연구 과제를 기술한다.

Ⅱ. 컴포넌트 프레임워크

컴포넌트들은 일반적으로 그림 1과 같은 형상으로 구성되며 이에 대한 세부 사항은 다음과 같다 :

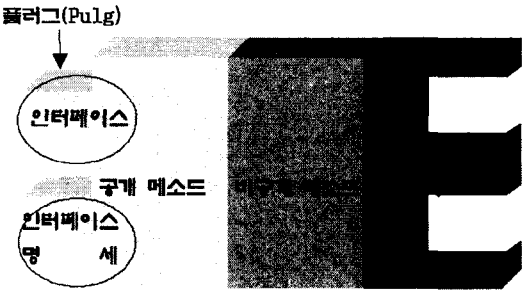

그림 2. 컴포넌트의 형상

(1) 플러그(Plug) : 다른 컴포넌트와의 조립을 위해 제공되는 인터페이스와 그에 대한 행위 명세(behaviour specification)
(2) 공개 메소드(Revealed behaviour) : 개발자가 컴포넌트를 채택함에 있어 보조하기 위해 필요한 측면에서의 공개가 요구되는 메소드와 플러그 대한 상세한 명세
(3) 비공개 메소드(Hidden behaviour) : 컴포넌트의 사용자와 무관한 구현 세부 사항과 설계 사항
(4) 실행부(Executable Part) : 컴포넌트의 기능을 실행

그림 1의 컴포넌트는 공개 메소드와 비공개 메소드를 동시에 갖기 때문에 그레이 박스(grey box)형의 컴포넌

트이다. 비공개 메소드가 없는 컴포넌트는 화이트 박스(white box)형이고 플러그를 제외하고 공개 메소드를 갖지 않는 컴포넌트는 블랙 박스(black box)형의 컴포넌트이다.[1]

컴포넌트 관련 기술의 세계적 표준화는 OMG (Object Management Group), Microsoft, Sun Micro- systems 세 조직을 중심으로 이루어지고 있다. OMG는 분산 객체 기술의 표준인 CORBA(Common Object Request Broker Architecture)를 기반으로 컴포넌트 기반 시스템의 구조를 정의한다. Microsoft는 COM(Component Object Model), OLE(Object Linking and Embedding) 등 Windows 플랫폼을 기반으로, Sun은 인터넷 기반의 Java 기술을 기반으로 컴포넌트 기반 시스템의 구조를 정의한다.

2.1 OMG의 CORBA Component Model(CCM)

CCM은 CORBA 3.0 명세에 정의되어 있으며, CORBA기반의 분산 엔터프라이즈 어플리케이션의 개발 패턴을 종합한 모델이다. CCM의 표준 명세를 정의하기 위한 RFP(Request for Proposal)가 1997년 6월에 OMG로부터 제안되었고, IBM, Oracle, BEA Systems 등 여러 회사가 참여하여 제안작업을 수행하여 현재 거의 완성된 명세가 나와 있는 상태이다.

CORBA 컴포넌트는 CORBA의 새로운 메타 타입으로 객체 메타 타입을 확장한 것이다. 컴포넌트 타입은 IDL component문으로 정의하며 인터페이스 저장소에 저장된다. 컴포넌트는 객체 참조자로 표현되는 컴포넌트 참조자를 갖는다.

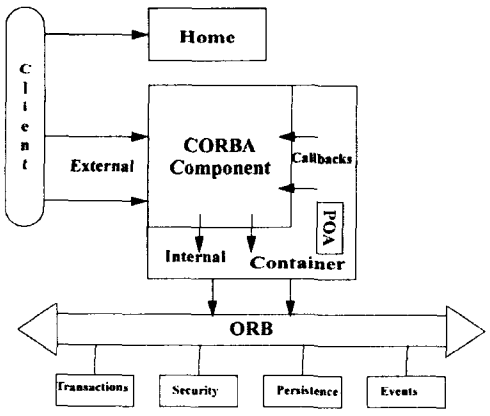

그림 3. CCM의 컨테이너 모델

컨테이너는 CORBA 컴포넌트 구현을 위한 서버의 런타임 환경으로 CORBA 애플리케이션을 쉽게 만들도록 하는 API 프레임워크를 제공한다. 컨테이너 프로그래밍 모델은 다음과 같은 4개의 요소로 구성된다. 그림 2는 CCM을 활용한 프로그래밍시 컨터이너 프로그래밍을 위한 요소와 다른 CORBA 요소와의 관계를 보여 준다.[5]

2.2 마이크로소프트사의 COM+

마이크로소프트사의 COM+는 COM과 MTS (Microsoft Transaction Server)를 통합시킨 컴포넌트 모델이다. 윈도우즈 NT를 위한 미들웨어 컴포넌트 모델인 COM에 기반을 두고, 그 위에 MTS와 MSMQ (Microsoft Message Queue)등의 서비스를 통합하였다.

COM+의 클래스는 한 개 이상의 COM+ 인터페이스를 구현한 코드의 집합체이다. 클래스는 자신이 지원하는 인터페이스를 구현한 실제 함수를 제공한다. 인터페이스가 유일한 IID로 서로 구별되는 것처럼 COM+ 클래스도 유일한 식별자인 CLSID를 가지고 있다. 다른 컴포넌트와 교류하기 위해서는 클라이언트는 적어도 한 클래스의 CLSID와 그 클래스가 지원하는 IID를 알아야 한다. 이 정보를 이용하여 클라이언트는 COM+에게 객체를 생성과 인터페이스 포인터를 요청할 수 있다. 한 개 또는 그 이상의 클래스가 결합하여 서버를 구성하는 방식으로 구현 된다.[4]

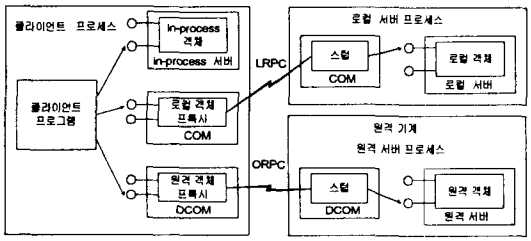

그림 4. COM+의 클라이언트와 서버 구조

2.3 Sun Microsystems사의 Enterprise JavaBeans

EJB(Enterprise JavaBeans)는 서버 컴포넌트 모델을 정의하며, 엔터프라이즈 레벨의 컴포넌트를 생성하는 방법에 대한 메커니즘을 제공한다. EJB는 엔터프라이즈 어플리케이션을 개발하기 위한 표준 플랫폼인 J2EE(JavaTM2 Platform, Enterprise Edition)의 컴포넌트 모델이다.

J2EE는 EJB에서 사용하는 서비스의 API 및 표준 클라이언트를 포함하는 인프라 스트럭쳐를 정의한다.[5]

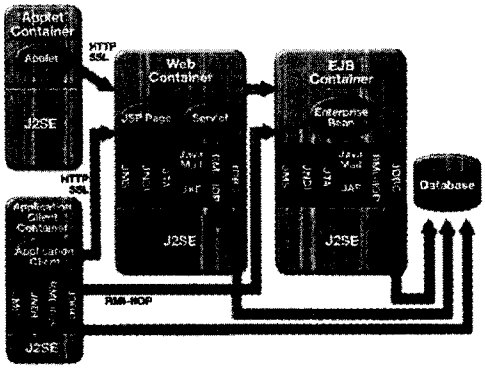

그림 4. J2EE 구조

EJB는 분산 트랜잭션 지향 엔터프라이즈 어플리케이션의 컴포넌트로서 한 클래스가 하나의 빈이 된다. 모든 엔터프라이즈 자바빈은 빈의 생성과 관련된 기능을 가지고 있는 홈(Home)과 비즈니스 로직을 가지고 있는 원격(Remote) 인터페이스들을 가지며, 클래스가 지원하는 기능과 상태, 자료의 특징에 따라 세션 빈과 엔티티 빈으로 나뉘게 된다.

EJB는 클라이언트와 컨테이너 사이, 그리고 엔터프라이즈 빈과 컨테이너 사이에 계약을 가지고 그 계약 내에서 각 역할들을 지원된다. 클라이언트는 엔터프라이즈 자바빈의 홈 및 원격 인터페이스를 접근하고, 객체 식별자를 얻는다. 또한 엔터프라이즈 빈의 동적인 호출을 위한 메타자료 인터페이스가 지원된다. 컴포넌트 계약 차원에서는 엔터프라이즈 자바빈의 인스턴스에 관한 생명 주기 등이 컨테이너에 의해 관리되고, 모든 컨테이너의 서비스 목록들이 엔터프라이즈 빈들에게 제공된다.

Java라는 프로그래밍 언어로 출발하여 컴포넌트 플랫폼으로 확장된 J2EE, 여러 프로그래밍 언어와 운영 체제 환경에서 작성된 시스템들을 통합할 목적으로 출발한 CCM과 Windows라는 운영 체제의 일부분으로 출발한 COM+는 각기 유래가 다르기 때문에 특징이나 기능적 메카니즘이 서로 다르다.

그러나 근본적으로 모두 RPC 계열의 프로세스 호출 메카니즘을 갖고 있으며 분산 객체 환경을 지원하고 있다. 또한 표준화된 인터페이스를 통해 클라이언트가 서버

표 1. 컴포넌트 플랫폼 비교표

Comparison Criteria	CORBA	EJB	COM+
Complexity of Specification	Complex(Extended IDL)	Somewhat complex	The most complex
Comp. Spec. Language	CIDL	None(tools, Java)	IDL
Separation of Public I/F	Separated IDL interface	No separated interface	Separated through interface
Communication Protocols	IIOP	RMI/IIOP, JNDI	RPC
Customizability	Fully Supported(Dynamic binding, Plug-in Objects)	The same as left colm. (at Deployment time)	Limited supported (containment, aggregation)
Scalability	Fully scalable and distributable	Fully scalable and distributable	Not supported in Win 2000
Interoperability	Source code level through IDL	Binary level within a specific EJB server	Binary level (Windows platform)
Required Runtime Engines	CIF Engine	EJB container, EJB server	COM Library
Runtime Efficiency	Not yet tested	moderate	Superior for C/S Application
Development Tools/CASE	Not available yet	A few(Web Logic,..)	Most MS Visual Studio
Market Share	Some	A small number	Thousand of products
Market Prediction	Most traditional, academic	Emerging for large grained components	Greatest market for small-grained components

나 구현된 컴포넌트를 호출하고 이를 서비스하기 위한 부가 기능을 갖는 등 유사성도 많이 있다. 표 1에서는 세 개의 컴포넌트간 유사성과 차별성을 도식화 하여 비교 정리한다.[1][2][7]

Ⅲ. 상호 운영성 보장 참조 프레임워크

3.1 제안된 컴포넌트 프레임워크

CCM은 EJB와 마찬가지로 패키징과 배포에 관한 정보를 나타내기 위해 XML(eXtensible Markup Language)을 사용한다. XML은 전자문서의 내용을 구조화하기 위한 국제 표준이므로, 이기종의 플랫폼에서 인식이 가능하다.

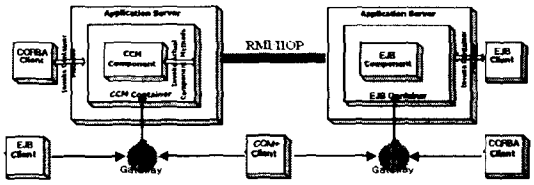

그림 6. 상호 운영성 보장 참조 프레임 워크

따라서 각 플랫폼간의 콘테이너 정보를 XML을 이용하여 표준 게이트웨이(Gateway)로 선언하고, 이 정보를 IDL내에 선언하면 CCM 클라이언트 스텁과 EJB/J2EE의 RMI 스텁내에 함께 선언되어 상호 호출 가능하게 된다.

CCM에서는 동일한 어플리케이션에서 EJB와 CORBA 컴포넌트를 통합하여 사용할 수 있도록 하기 위한 구조를 제공한다. 각 EJB 컴포넌트를 관리하기 위한 컨테이너를 따로 정의하며, EJB 정의와 매핑이 가능하도록 IDL을 확장하였다. CCM과 EJB가 혼합된 모델에서는 게이트웨이(Gateway)가 각기 다른 시스템의 인터페이스간의 매핑을 가능하게 해준다. 그림 5 CCM 클라이언트에서 EJB 컴포넌트를 참조하기 위해서는 EJB 정의가 CORBA IDL로 매핑되어야 하고, 런타임 시 CCM 클라이언트에 의해 호출된 CCM 메소드는 브리지에 의해 EJB 메소드로 변환된다.

제안된 플랫폼을 이용하여 컴포넌트를 패키징하고 배포(deployment)하기 위한 과정은 다음과 같다 :

1) 컴포넌트 어셈블리와 패키징

CCM과 EJB/J2EE 환경에서 컴포넌트 패키지는 단일 컴포넌트, 컴포넌트 어셈블리 패키지는 연관성을 가진 다수의 컴포넌트를 배포하기 위한 매개체이다. 컴포넌트들은 zip 압축파일 형태로 패키지화 되며, 각각의 내부 정보를 XML로 기술한다. 패키징 정보에 대한 규약은 CCM과 J2EE가 모두 동일하고 또한 모두 XML을 사용하기 때문에 XML로 작성된 패키징 정보는 상호 인식 가능하다.

2) 컴포넌트 배포

컴포넌트 패키지와 어셈블리 패키지는 배포 툴이나 어플리케이션을 사용하여 네트워크상에 타겟 호스트에 배포된다. 어셈블리 패키지 내에 있는 컴포넌트들은 서로 다른 위치에 배포가 가능하다.

XML로 기술된 패키지 내부 정보와 사용자가 제공하는 정보가 조합되어 배포 툴의 입력 정보가 되고, 배포 툴은 그 정보를 바탕으로 컴포넌트 서버, 컨테이너, 컴포넌트 홈과 인스턴스를 생성하고 활성화한다.

3.2 게이트웨이(Gateway)의 메시지 전달

XML게이트웨이(Gateway)는 이형 플랫폼에 대한 모든 메시지 호출을 XML형태로 변환하고 이를 해당 컨테이너에게 전달하는 역할을 수행한다.

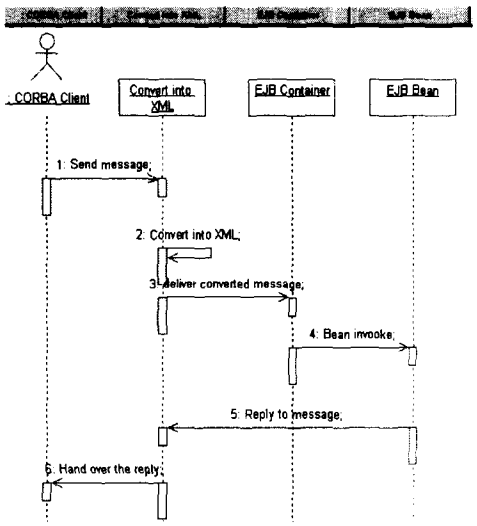

그림 7. Messaging Seq. Diagram

Ⅳ. 결론 및 향후 연구 과제

컴포넌트 플랫폼을 활용한 어플리케이션 작성시 기존의 타 플랫폼으로 작성된 레거시(legacy) 시스템을 연동할 수 있는 참조 모델을 제시하였다. 이로써 특정 영역의 어플리케이션을 개발하려 할 때 참조 모델에서 유도되는 소프트웨어 아키텍처를 이용함으로써 컴포넌트 기반 소프트웨어 개발을 효율적으로 수행할 수 있으며 동시에 타 플랫폼에서 작성된 컴포넌트들을 재활용 할 수 있는 밑바탕을 마련하였다. 이러한 환경을 제공함으로써 컴포넌트의 재사용성을 타 플랫폼까지 연장할 수 있게 되었다.

현재 제안된 참조 플랫폼을 이용하여 자동화된 게이트웨이를 생성하기 위한 추가 연구를 진행중이다. 이 연구가 완료되면 제안된 플랫폼과 연계 할 예정이다.

참고문헌

[1] 장연세,참조 컴포넌트 아키텍처 모델과 UML 명세화에 대한 연구,한국OA학회, Vol 6, No 3, 2001년 7월호

[2] 장연세,임승린,UML을 이용한 컴포넌트 명세화에 대한 연구, 수원과학대학기술연구소, 2002

[3] 장연세,컴포넌트의 품질 측정을 위한 메트릭에 대한 연구, 한국SI학회 2002년 춘계학술대회

[4] 장연세,금영욱,CORBA와 DCOM의 통합,한국정보과학회, 1999년 7월호

[5] 장연세,금영욱,CORBA 3 프로그래밍 바이블, 도서출판 그린, 2000년5월

[6] Stuart Kent, John Howse,Modelling Software Components, IEEE, 1998

[7] Re-Chi Lin,Reusing MS-Windows Software Applications Under CORBA Environment, IEEE, 1998

[8] Nunzio-Nicolo Savino,A UML-based Method to Specify the Structural Component of Simulation-based Queuing Network Performance Models, IEEE, 1999

[9] OMG, The Common Object Request Broker: Architecture and Specification 2.2, OMG, MA,1998,ftp://ftp.omg.org/pub/docs/formal/98-07-01.pdf

[10] 조은숙,김수동,류성열, UML을 기반으로 한 실무 중심의 객체지향 방법론, 한국정보과학회, 1999년

[11] 한정수,송영재,클래스 부품 재사용을 위한 객체의 추출과 이해, 한국정보과학회, 1999년

저 자 소 개

장 연 세

1993년 서울산업대학교 전자계산학과(공학사)

1995년 수원대학교 전자계산학과(이학석사)

1998년 아주대학교 컴퓨터공학과(박사과정 수료)

1998.2 ~ 1999.2 (주)필컴 기술연구소 팀장

1999.3. ~ 2001.4 (주)한국정보컨설팅 기술연구소장

2001.5 ~ 2002.4 (주)한국후테로시스템 기술연구소장

2001.5 ~ 현재 한국컴포넌트 컨소시엄 위원

2001.3 ~ 현재 수원과학대학 인터넷정보과 겸임교수

2002.5 ~ 현재 쿠도커뮤니케이션(주) 네트워크기술연구소장

관심분야 :
분산객체, CORBA, J2EE, 컴포넌트