이종의 컴포넌트 미들웨어 프레임워크 간의 상호운용성을 위한 모델 설계

김경민[O] 김태웅 김태공 최항묵
인제대학교 전산학과
{kmkim[O], twkim, ktg, hmchoi}@cs.inje.ac.kr

A Design of the Model for Interoperability among Variant Component Middleware Frameworks

Kyung-Min Kim[O] Tae-Woong Kim Tae-Gong Kim Hang-Mook Choi
Dept. of Computer Science, Inje University

요 약

현재 소프트웨어 재사용과 생산성, 시스템 품질관리 등에 대한 해결책으로 부품화와 재사용의 특징을 가지는 컴포넌트기반의 소프트웨어 개발 방법론이 대두되고 있다. 경험이 많은 개발자에 의해 개발된 컴포넌트는 소프트웨어 재사용이 뛰어나고, 이미 많은 곳에서 사용 중이므로 안정성 및 신뢰성이 인정된다. 그러나 이러한 컴포넌트는 하나의 컴포넌트 미들웨어 프레임워크에서 개발해야 하는 한계를 가지고 있으며 이러한 결과는 결국 소프트웨어 개발비용에 결정적인 영향을 미친다. 이에 본 논문에서는 서로 다른 컴포넌트 미들웨어 프레임워크에서 개발된 컴포넌트를 호출하고 사용하기 위해 HTTP와 XML를 이용하여 이종의 컴포넌트 미들웨어 프레임워크들 간의 상호운용성을 위한 모델을 설계하고자 한다.

1. 서 론

소프트웨어 개발에 있어서 가장 큰 관심은 소프트웨어 개발 생산성을 높이는 것이며, 이 생산성을 높이기 위해서는 소프트웨어를 이루고 있는 여러 가지 구성단위(컴포넌트)들을 재사용하는 방법이 가장 효과적이다. 현재 컴포넌트의 개념이 활성화되면서 소프트웨어 개발 생산성은 비약적으로 높아지게 되었으며, 컴포넌트 기술의 발전과 인터넷의 급속한 보급으로 여러 플랫폼 상에 분산된 컴포넌트들을 설계, 구축 및 조립하는 컴포넌트 기반의 개발 기술들이 필요로 하게 되었다.

이로 인해서 개발된 것이 이종의 컴퓨터 환경 하에서 컴포넌트 간의 연동 및 조립 구조를 제공하는 미들웨어 프레임워크이며, 분산된 컴포넌트들은 이 미들웨어 프레임워크를 통해 서로 통신하고 작동이 가능하게 된다. 대표적인 미들웨어 프레임워크 제품으로는 OMG(Object Management Group)의 CORBA(Common Object Request Broker Architecture)[1] 제품들, SUN사의 J2EE(Java 2 Platform Enterprise Edition)[2] 제품들 그리고 Microsoft사의 .NET[3] 제품들이 있다. 이들 미들웨어 프레임워크의 등장으로 인해 동종의 시스템에서 뿐 아니라 이종의 시스템에서도 컴포넌트의 재사용이 가능해졌다.

그러나 여러 종류의 미들웨어 프레임워크가 등장함에 따라 또다시 컴포넌트가 하나의 컴포넌트 미들웨어 프레임워크에서 개발해야 하는 미들웨어 프레임워크간의 연동 문제가 발생하게 되었다. 객체 표준화 기구인 OMG에서는 CORBA 사양을 중심으로 미들웨어 프레임워크 간의 연동 구조를 정의하고 있으나 아직 .NET과의 호환은 이루어지지 않고 있다[4].

이처럼 재사용성이 좋은 컴포넌트 기반의 소프트웨어 개발이라고는 하지만 모든 컴포넌트들이 자신의 개발 미들웨어 프레임워크에 의존적이고 이종의 모델에서 개발된 컴포넌트와는 유기적인 결합을 이루지 못하고 있다.

이에 본 논문에서는 시스템에 독립적인 문서 XML[5]과 기존의 HTTP[6] 프로토콜을 이용하여 이종의 컴포넌트 미들웨어 프레임워크 간의 상호운용성을 위한 모델을 설계하고자 한다.

2. 상호운용성을 위한 모델 설계

2.1 모델의 전체 구조

본 논문은 이종의 컴포넌트 미들웨어 프레임워크 간의 상호운용성을 위해 그림1과 같은 모델을 제안한다. 이 모델은 네트워크 전송수단으로써 분산 환경 및 공동작업 환경에서 하이퍼텍스트 전송을 위해 사용되는[7] HTTP를 이용한다. 그리고 컴포넌트의 특정 인터페이스 메소드 요청/응답 인코딩에는 일관된 방법으로 데이터를 형식화하고 전송을 위한 구조적인 데이터 형식을 제공하는 메타-마크업 언어인[8] XML을 사용함으로서 컴포넌트 간의 상호운용성을 높인다.

모델은 컴포넌트 미들웨어 프레임워크마다 각각 해당 컴포넌트로의 접근 변환을 담당하는 데몬을 가진다. 이 데몬에서 제공하는 서비스에 의해서 사용자는 사용할 컴포넌트의 미들웨어 프레임워크가 무엇인지 알 필요 없이, 컴포넌트의 존재여부 및 컴포넌트 이름, 인터페이스

이름, 메소드 이름, 파라미터 등 호출시 필요한 정보만을 이용하여 이종의 컴포넌트로의 접근이 가능하게 된다.

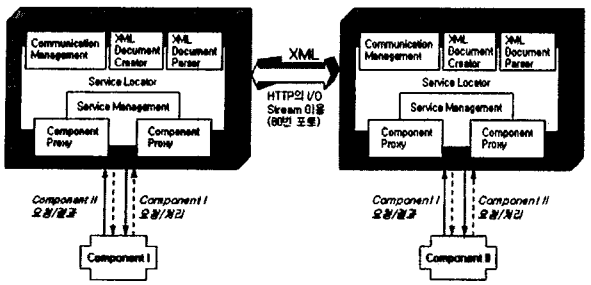

그림1 브리지 서비스 전체 구조

각 데몬들은 미들웨어 프레임워크 간의 상호운용성을 위한 서비스를 제공하기 위해 내부 구성 요소들로, 전송된 메시지들을 분석하는 통신 관리자(Communication Management)와 컴포넌트 요청과 그 처리 결과에 대한 XML 문서 생성기와 번역기(XML Document Creator /Parser), 요청된 컴포넌트를 찾는 서비스 결정자(Service Locator), 데몬 내부에서 생성된 프로세스들의 생명주기와 네이밍 서비스를 담당하는 서비스 관리자(Service Management), 실제 이종의 데몬들 간의 통신을 담당하는 컴포넌트 프록시(Component Proxy)를 포함한다.

2.2 유즈케이스 다이어그램

모델의 전체 구조를 기반으로 하여 액터(Actor)로서 이종의 컴포넌트1과 컴포넌트2, 해당 컴포넌트의 미들웨어 프레임워크1 데몬, 미들웨어 프레임워크2 데몬을 추출하였고 각각의 액터들과 관계를 가지는 22개의 유즈케이스를 추출하였다. 이를 바탕으로 UML[9] 표기법을 이용하여 유즈케이스 다이어그램을 작성하면 그림 2와 같다.

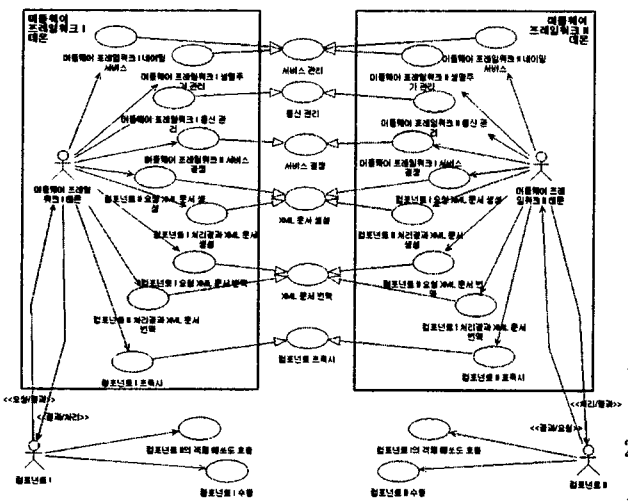

그림2 브리지 서비스의 유즈케이스 다이어그램

2.3 시퀀스 다이어그램

그림3, 그림4는 컴포넌트1에서 이종의 컴포넌트2의 특정 인터페이스 메소드 호출시, 해당 데몬에서 수행되는 절차를 미들웨어 프레임워크1 데몬(MF1 Daemon) 중심과 미들웨어 프레임워크2 데몬(MF2 Daemon) 중심으로 수행되는 절차를 나타내는 시퀀스 다이어그램이다.

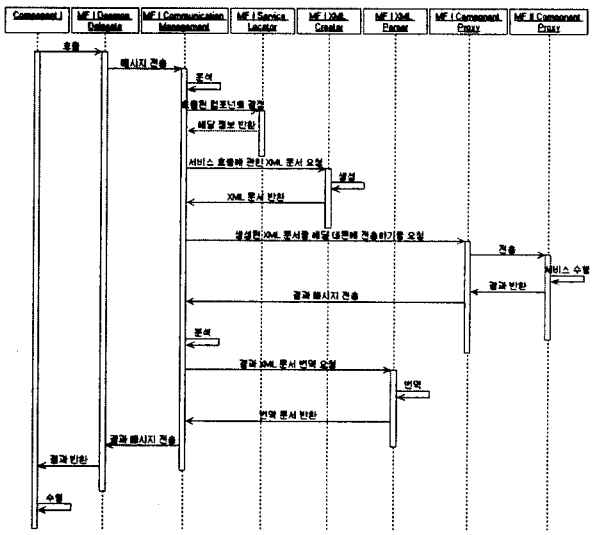

그림3 컴포넌트1에서 컴포넌트2 호출시 MF1 Daemon 중심의 시퀀스 다이어그램

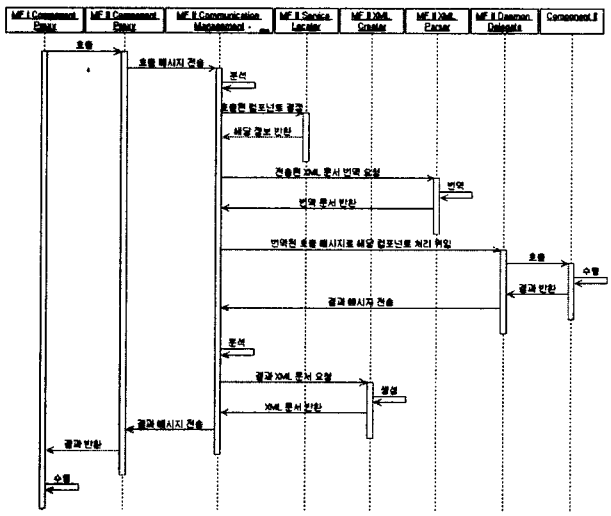

그림4 컴포넌트1에서 컴포넌트2 호출시 MF2 Daemon 중심의 시퀀스 다이어그램

2.4 XML DTD 정의

앞서 이종의 컴포넌트 사이의 인터페이스 메소드호출 정보 및 처리결과 정보를 XML로 나타냄으로서 사용하는 시스템에 상관없이 문서의 일관성을 유지하였다. 이

XML을 문서의 구조적 표현이 가능하고 요소 사이의 관계를 지정할 수 있는 DTD(Document Type Definition)로 정의함으로서 XML 문서의 유효성을 유지하고자 한다[10,11].

그림5와 그림6은 컴포넌트의 인터페이스 메소드 호출 정보를 기술하는 XML DTD와 그 호출에 대한 처리 결과 정보를 기술하는 XML DTD를 정의하고 있다.

```
<!ELEMENT Component (Name, Interface)>
<!ATTLIST Component id CDATA #REQUIRED>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Interface (Name, Method)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Method (Name, ReturnType, Parameter*)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT ReturnType (#PCDATA)>
<!ELEMENT Parameter (Name, Type)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
```

그림5 컴포넌트의 인터페이스 메소드 호출 정보를 기술하는 XML DTD

```
<!ELEMENT Result (ReturnType*, ReturnValue*)>
<!ATTLIST Component id CDATA #REQUIRED>
<!ELEMENT ReturnType (#PCDATA)>
<!ELEMENT ReturnValue (#PCDATA)>
```

그림6 컴포넌트의 인터페이스 메소드 처리 결과 정보를 기술하는 XML DTD

이와 같은 XML문서들은 HTTP 전송 시 그림7과 같은 헤더정보를 포함한다. 이 헤더 정보를 이용하여 데몬 내의 서비스 결정자에 의해서 요청된 서비스에 대한 해당 컴포넌트를 찾게 된다.

```
POST /rpc/object1.asp HTTP/1.1
Host: rpcserver.domain.com
Content-Type: text/xml
Content-Length: 152
Component-Name:
Interface-Name:
```

그림7 HTTP의 Header

이처럼 본 연구에서 제안하는 모델은 HTTP와 XML이라는 기술을 인프라로 사용함으로써 운영체제, 플랫폼, 소프트웨어 벤더를 초월하는 상호운용성을 확보할 수 있게 된다.

3. 결론 및 향후 연구과제

소프트웨어의 대형화, 통합화가 요구되어지면서 재사용성이 뛰어난 컴포넌트 기반의 소프트웨어 개발 방법론이 대두되고 있지만, 모든 컴포넌트들은 자신의 개발 컴포넌트 미들웨어 프레임워크에 의존적이며 이종의 모델에서 개발된 컴포넌트와는 유기적인 결합을 이루지 못하고 있다.

따라서 본 논문에서는 이종의 컴포넌트 모델들과 관련된 컴포넌트들 사이에서 각각의 컴포넌트를 호출하고 사용하는 방법으로 범용적인 프로토콜 HTTP와 시스템 독립적인 문서 XML을 사용하여 이종의 컴포넌트 미들웨어 프레임워크 간의 상호운용성을 위한 모델 설계를 해보았다.

이 모델의 특징은 기존의 범용적인 프로토콜 HTTP를 사용함으로써 방화벽에 의한 원거리 컴포넌트 호출시 발생하는 문제점을 제거하고, XML 문서에 컴포넌트의 호출과 처리 결과에 대한 정보를 나타냄으로서 다양한 컴포넌트 미들웨어 프레임워크 간에 상호작용이 이루어 질 수 있다. 또한 서비스의 요청을 실제로 받아 처리하는 각각의 데몬들이 컴포넌트와의 정규적인 체크를 통해 컴포넌트의 사용여부를 사용자에게 보다 정확하게 알려줄 수 있다. 새로운 컴포넌트 모델의 등장에도 이 모델은 해당 데몬만을 추가함으로서 사용자가 시스템의 코드수정 없이 새로운 컴포넌트의 인터페이스 메소드를 호출할 수 있다는 장점을 가진다.

현재 본 논문에서 제안하는 모델의 부분적인 구현이 진행 중에 있으며, 나아가 이종의 미들웨어 프레임워크 간의 효율적인 상호운용성을 위해 트랜잭션 처리와 지속성 유지, 보안 등과 같은 비기능적인 요구 사항들을 추가하는 작업이 필요하다.

[참고문헌]

[1] OMG, "The Component Object Request Broker: Architecture and Specification", 1999

[2] Sun Microsystems, Inc., "Java 2 Platform Enterprise Edition Specification", ver1.2, 1999

[3] 마이크로 소프트웨어, "개발 환경의 패러다임 시프트 MS 비주얼스튜디오 닷넷", 2001.1

[4] 최성운 홍선주 장진호 전인걸, "MDA기반 S/W 컴포넌트", 제80호 TTA저널, 2002.4

[5] W3C, "XML Specification", http://www.w3c.org/TR/WD-xml.html, 1997

[6] RFC1945, Irvine H. Frystyk MIT/LCS, "HyperText Transfer Protocol(HTTP)", 1996

[7] W. Richard Stevens, "TCP/IP Illustrated, Vol.3", Wesley, 1993

[8] R. Khare and A. Rifkin, "X Marks the Spot", http://www.cs.caltech.edu/~adam/papers/xml/x-marks-the-spot.html, 1997

[9] G., Booch, J., Rumbaugh, and I., Jacobson, "The Unified Modeling Language User Guide", Addision-Wesley, 1999

[10] 정지훈, "웹 서비스", 한빛미디어, 2002

[11] W3C Recommendation, "Extensible Markup Language(XML)1.0", http://www.w3c.org/TR/1998/REC-xml-19980210, 1998.2.10