CORBA 기반 시스템 통합 모델

김 남 용[†] · 왕 창 종[††]

요 약

하드웨어와 소프트웨어의 다양화는 날로 증가하고 있으며, 네트워크 컴퓨팅 환경은 다양해져 가고 있다. 소프트웨어 개발은 이 기종 컴퓨터 집합, 다른 장소에서의 여러 가지 데이터 유형 저장, 운영체제의 비호환성에 의한 작업, 그리고 여러 가지 데이터베이스와 프로토콜 때문에 많은 비용이 들어가는 작업이 되어가고 있다.

CORBA는 분산 컴퓨팅환경과 이기종 분산 환경의 시스템통합을 위한 표준이다. CORBA는 효과적인 시스템 통합을 위해 기술적인 이익을 제공하며, 이 기종의 시스템들의 분산 의사소통 환경을 위한 하부구조를 제공한다.

본 논문에서는 분산객체환경, 소프트웨어 재사용 그리고 WWW과의 연결을 위하여 CORBA에 기반을 둔 시스템통합 모델을 제안한다. 제안된 모델은 팩터리서버, 트레이딩 서버, 변환서버 그리고 응용서버들로 구성된다. 제안된 모델은 응용들의 개발과 시스템의 통합을 용이하게 한다. 또한 WWW와의 연결을 위한 게이트웨이를 구현함으로써, WWW으로의 확장이 가능하다. 본 연구에서 제안한 모델를 증명하기 위하여, 원격 교육 시스템을 본 연구에서 제안한 모델에서 제공하는 서비스들을 사용하여 쉽고, 효과적으로 설계하였다.

The System Integration Model based on CORBA

Nam Yong Kim[†] · Chang Jong Wang[††]

ABSTRACT

Diversity in hardware and software is increasing ever and our networked computing environment is becoming more diverse. The development of software becomes expensive works because of a collection of diverse computers, storing various data type in different places, working together by incompatabilities of operating system and various databases and protocols.

CORBA is a standard for distributed computing environment and for system integration of heterogeneous distributed environment. CORBA provides many technical benefits for effective system integration and seamless infrastructure for distributed communication environment of heterogeneous systems.

In this paper, we proposed a system integraton model based on CORBA for distributed object environment, software reuse and the interconnecion of WWW. The model is composed of factory server, trading server, conversion server and applicaton server. The proposed model can easy application development and system integration. And we implemented the gateway for cooperation with WWW. As a proof of the proposed model, we show how the distance learning system designed using the services provided by the proposed model.

† 정 회 원:인하대학교 전자계산공학과 박사과정
†† 정 회 원:인하대학교 전자계산공학과 교수
논문접수:1997년 5월 28일, 심사완료:1997년 11월 6일

1. 서 론

오늘날의 컴퓨팅 환경은 우리들이 사용하고 있는 하드웨어 및 소프트웨어들이 제한된 범위 내에서 유용한 기능을 제공하고 있으나, 네트워크 상에서 서로 효율적으로 상호 운용되지 못하고 있다. 현재 소프트웨어 분야는 다양한 운영체제, GUI, 통신 프로토콜로 인하여 상호연동이 어렵다[1]. 이로 인해 분산 환경에서 시스템의 성능 개선이나 유지보수에 많은 비용이 요구되며, 사용자의 요구에 적절히 대응하지 못하고 있는 실정이다. 분산 환경에서 기존에 개발한 소프트웨어의 효율적인 재사용과 통합이 절실히 요구되며, 특히 분산 환경에서의 시스템 통합은 과거에 많은 비용을 들여 개발한 기존의 응용 소프트웨어와 정보를 재사용할 수 있다는 점에서 매우 중요한 분야로 부각되고 있다[2, 3, 4, 14].

시스템 통합을 위해 OLE/COM, OpenDoc, CORBA 등과 같은 분산 객체기술을 이용한 모델들이 제안되고 있다. 이중 CORBA는 분산객체를 지원하는 OMG의 표준으로서, 분산 응용을 개발하는데 있어서, 일관된 인터페이스를 제공한다. 이를 통해서, 시스템 개발 모델을 표준화함으로써, 시스템의 유지 보수 및 확장을 용이하게 한다. 또한 상속성, 캡슐화등의 객체 지향 기술을 지원한다. 이러한 CORBA의 특징은 시스템 통합의 기반 기술로써 널리 이용될 수 있다[3, 5, 6, 7].

Mowbray는 기존 시스템의 통합방식을 6단계로 분류한 통합 성능 모델(integration capacity model)을 제시하였으며, 시스템 통합 시 소프트웨어 아키텍처의 중요성을 제시하고 있다[4]. 이 중 프레임워크(framework) 기반의 시스템 통합은 응용들간의 중복된 기능들을 하나의 객체로 제공하고, 기존 응용들을 객체화된 서비스로 이용되도록 재공학하여, 조직이나 개인의 요구 변화에 유연하게 대응할 수 있는 적응성을 제공한다[8].

본 논문에서는 기존 시스템들을 통합하고, 분산 환경에서의 서비스에 요구되는 기능을 분석, 정의하여 분산 객체 기술을 이용한 응용 시스템으로 통합하기 위한 프레임워크를 제안한다. 제안된 프레임워크는 ORB, 트레이딩 서버(trading server), 변환 서버(conversion server), 팩토리 서버(factory server), 그리고 WWW와의 통합을 위한 게이트웨이(gateway)로 구성된다. 또한, 기존의 응용 프로그램들은 객체 래퍼(wrapper)를 이용하여 통합되고, 사용자는 WWW 브라우저를 통해 일관된 인터페이스를 제공받을 수 있다.

본 논문의 구성은 2장에서 기존의 시스템 통합 모델과 OMA의 CORBA, 그리고 기존의 시스템 통합에 대한 연구를 고찰하고, 3장에서는 COBRA 기반 시스템 통합 모델 제시와 사용자에게 일관된 인터페이스 제공을 위한 게이트웨이를 설계한다. 4장에서는 제안된 프레임워크를 원격교육시스템에 적용하였으며, 마지막 5장에서는 결론 및 향후 연구 과제를 기술한다.

2. 관련 연구

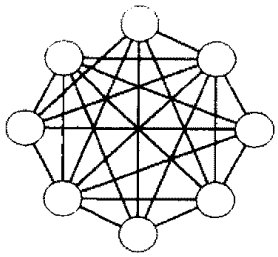
본 장에서는 기존의 시스템 통합 모델에 대해 고찰하고, 시스템 통합 방법으로 제시되고 있는 CORBA 기술과 이를 사용한 기존 시스템 통합에 대한 연구에 대해 기술한다.

2.1 시스템 통합모델

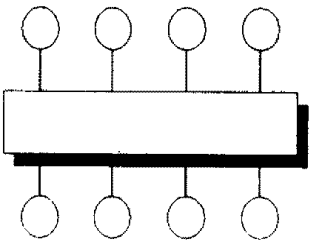
시스템 통합 시 가장 중요한 과정은 적절한 소프트웨어 아키텍처를 결정하는 것이다. 잘 구성된 소프트웨어 아키텍처는 시스템의 적응성과 재사용성을 높여주며, 시스템을 보다 쉽게 이해될 수 있도록 한다[4, 9]. 프레임워크 방식은 과거의 클래스 라이브러리 형태와는 달리 응용의 운영 방식에 대한 내용을 응용 프레임워크가 갖고 있다. 응용 프로그래머는 이제 프레임워크에 의해 호출될 새로운 함수 또는 클래스를 개발하면 된다[14, 15]. (그림 1)과 〈표 1〉은 기존 방식의 인터페이스와 프레임워크 방식의 인터페이스를 비교하고 있다.

Mowbray가 제시한 통합 성능 모델은 어떤 조직이나 기업체에서 기존 시스템의 통합 단계를 평가할 수 있는 모델로서, 다음과 같은 6 단계로 분류한다[4].

제 1단계는 어떤 특별한 시스템 통합 없이 상업적으로 판매되는 소프트웨어를 그대로 사용하는 단계이다. 이러한 경우 통합은 고려되지 않는다. 제 2단계는 필요에 의해 사용 가능한 여러 통합 기술들을 사용하여, 각 응용들 간의 통합을 도모한다. 제 3단계는 OSF DCE와 같은 RPC기술을 이용하여 시스템을 통합하는 단계이다. 제 4단계는 분산 객체 기술을 이용하여 시스템을 통합하는 단계로서, 분산 환경에서의



(a) 일반적인 인터페이스(N*N)
(a) Custom Interface(N*N)



(b)프레임워크 기반 인터페이스(N)
(b) Framework-based Interface (N)

(그림 1) (a) 일반적인 인터페이스(N*N) (b)프레임워크 기반 인터페이스(N)
(Fig. 1) (a) Custom Interface(N*N) (b) Framework-based Interface (N)

〈표 1〉 클래스 라이브러리와 프레임워크의 비교
〈Table 1〉 The comparison of the Class Library and Framework

클래스 라이브러리	프레임워크
· 클라이언트가 사용할 수 있는 클래스 집합 · 클라리언트가 함수를 호출 · 미리 정의된 제어흐름이 없음 · 미리 정의된 상호작용성이 없음 · 미리 정의된 기본 행위가 없음	· 서브클래싱에 의해 수정 방법 제공 · 클라이언트 코드를 호출 · 수행의 제어흐름이 정의 됨 · 객체들간의 상호 연결 관계 정의 · 기본 행위 제공

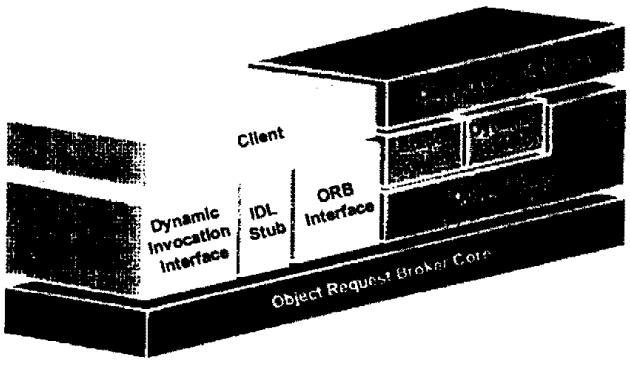
전송을 위해 ORB를 이용한다. 하지만 분산 처리가 필요하지 않는 경우는 기존 메커니즘을 그대로 이용한다. 제 5 단계는 프로젝트에 종속적인 요구에 의해 새로운 프레임워크를 개발하고 이를 이용하여 시스템을 통합한다. 시스템이 요구하는 각종 서비스들을 객체로 제공한다. 하지만 프로젝트간, 시스템간의 완전한 상호 운용성을 보장하지는 못한다. 제 6단계는 여러 프로젝트들간에 사용할 수 있는 재사용 가능한 높은 품질의 프레임워크를 개발하는 단계이다. 객체화된 서비스들은 현 시스템뿐만 아니라 다른 프로젝트나 시스템에서 사용이 가능하며, CORBA 표준을 따른다.

본 논문에서는 통합 성능 모델 중 6단계의 표준 프레임워크 모델을 이용하여 새로운 통합 모델을 제시한다.

2.2 CORBA

CORBA 규격에 의해서 만들어진 응용 객체들은 ORB를 통하여 통신한다. ORB는 CORBA로 표준화되었으며 서로 다른 곳에 존재하는 객체간의 메시지를 전달하고 표준화하는 계층이다[10, 11]. 객체 서비스는 모든 객체를 구현할 때 유용하게 사용할 수 있도록 OMG에서 정의한 공통되는 서비스들로 객체 이름 관리(naming service), 사건 서비스(event service) 등이 있다. 공통 지원 기능은 응용들 간에 객체를 공유하거나 응용을 쉽게 생성하기 위한 프레임워크이다.

CORBA는 객체 사이에 위치와 구현에 관한 투명성을 보장함으로써, 이기종 분산 환경 하에서 정보를 용이하게 교환하게 된다. CORBA는 이기종 객체간의 통신에서 중심 역할을 하며, 각 객체는 CORBA에 등록된 후 서로 다른 언어나 환경에서도 객체 인터페이스를 통하여 통신할 수 있다. CORBA를 기반으로 시스템을 통합함으로써 이기종 상호간의 운용성 보장, 멀티미디어 객체, 복제, 비동기성 메시지 전달, 동시성, TCP/IP외 특정 프로토콜, 브릿징 등을 지원할 수 있으며, 객체 참조 형식 및 전역 식별자 정의 등의

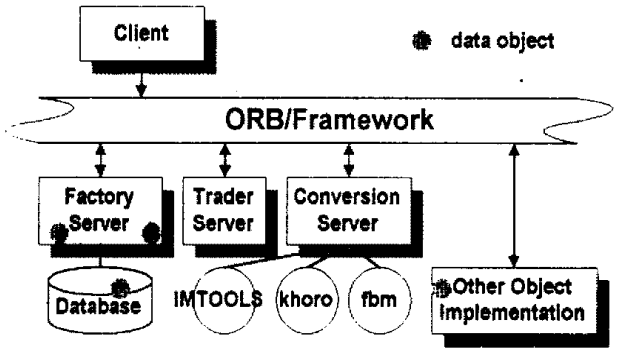


(그림 2) CORBA
(Fig. 2) CORBA

장점을 가질 수 있다. CORBA는 (그림 2)와 같이 구성되며, CORBA중에서 ORB가 핵심을 이루는 요소이다.

2.3 CORBA를 이용한 기존 시스템 통합

DISCUS(Data Interchange and Synergistic Collateral Usage Study)는 미국정부가 추진한 최초의 CORBA 관련 프로젝트로서 분산 객체들 간의 상호 운용성을 보장하고, 시스템 통합에 필요한 설계 및 구현 비용을 절약하기 위한 프레임워크를 개발하는 데 있으며, 개발된 프레임워크는 (그림 3)과 같다[12].



(그림 3) DISCUS의 구조
(Fig. 3) The architecture of DISCUS

DISCUS는 3개의 서버를 두고 클라이언트에게 시스템 통합에 대한 서비스를 제공하고 있다. 각 서버는 데이터 객체들을 생성·제거·관리하는 서버와 서로 다른 데이터 표현을 사용하는 응용들간의 자료 변환 서비스를 제공하는 서버 그리고 객체들간의 위치 투명성을 보장하기 위한 서비스들로 구성된다.

MITRE사의 DOMIS는 미 공군의 CTAPS(Contingency Theater Automated Planning System)의 문제점들을 해결하기 위해 분산 객체기술을 적용하여 하부 시스템을 통합하고, 복잡한 스토브-파이프 시스템을 DOM(Distributed Object Management) 아키텍처로 변환하는 기술과 방법론의 개발을 목적으로 한 프로젝트이다[13].

DOMIS 프로젝트는 기존 시스템을 재사용 가능한 객체 서비스 기반의 소프트웨어 아키텍처로 재구성하는 기술과 방법론의 개발에 중점을 두고 있다. 이는 각각의 응용들에 중복되거나 비슷한 기능을 수행하는 모듈들을 분리하여 CORBA 객체로 변환한 후 기존 응용을 재구성하는 과정을 거치게 된다. 〈표 2〉는 HyperDesk사의 ORB를 이용하여 캡슐화한 결과로서 기존의 인터페이스를 그대로 유지하면서 CORBA 객체로 변화하는 데 필요한 코드는 전체 코드에 비하면 매우 적은 양임을 알 수 있다.

3. 응용들간의 시스템 통합을 위한 프레임워크 설계

분산 환경에서 기존의 응용 프로그램들의 통합은 기존의 소프트웨어를 재사용할 수 있는 장점이 있으며, 프레임워크 기반의 통합은 실제 통합에 필요한

〈표 2〉 APS, CAFMS, JMAPS의 encapsulation cost
〈Table 2〉 The Encapsulation Cost of APS·CAFMS·JMAPS

	Lines in Module	Lines Written	Percent Reused
JMAPS, CAFMS & APS IDL	306	260	15%
JMAPS, CAFMS & APS methods	2,647	455	83%
Generic manager client	1,983	220	89%
DOMIS HMI	2,694	729	73%
Workarounds	3,189	419	87%
Total DOMIS devolopment	10,819	2,083	81%
JMAPS, CAFMS & APS	484,000	0	100%
TOTAL	494,819	2,083	99.6%

설계, 구현 비용을 최소화할 수 있다. 프레임워크 기반 없는 시스템 통합은 각 응용들이 각기 고유의 인터페이스를 가지고 있으며, 다른 프로그램과의 통합 시 각기 다른 인터페이스를 요구하므로 n×n 개의 인터페이스가 필요하다. 만일 여러 응용 프로그램들간의 공통된 인터페이스가 존재하면, 클라이언트는 단일한 인터페이스로 여러 객체 구현들을 호출할 수 있게 된다.

본 연구에서는 제안하는 프레임워크는 기존의 응용들의 통합에 사용되는 기본적인 기능을 프레임워크 내에서 제공하여, 단일한 인터페이스로 여러 응용을 통합할 수 있도록 한다. 제공하는 상호 운용성의 형태는 응용간 데이터 교환, 데이터 소스에 대한 액세스, 응용 기능에 대한 액세스이다.

3.1 프레임워크 객체 모델

제안된 프레임워크는 응용 객체와, 데이터 객체의 두 종류의 객체 모델을 이용한다. 응용 객체는 모든 종류의 응용 프로그램들을 포함하며, 프레임워크에서 제공하는 4개의 연산들을 구현함으로써 상호 운용성을 보장받는다. 데이터 객체는 데이터 교환과, 결과 전달을 위한 컨테이너로서, 데이터 명칭과 값의 쌍으로 구성되는 단순 데이터 객체와 구조적 데이터 표현을 위한 태블릿 객체로 구성된다.

3.1.1 응용 객체

응용 객체는 convert, exchange, query, execute의 4가지 연산으로 구성되며, 이 4가지 기본 연산들은 다시 open, close, destroy 연산을 CommonObject에서 상속받아 구현된다.

```
module FRAME {
  interface CommonObj { ... }
  interface ap:FRAME::CommonObj{ ... }
  interface dt:FRAME::CommonObj { ... }
  interface ft:FRAME::ap{ ... }
}
CommonObj {
  exception ALREADY_OPEN(string reason);
  exception NOT_OPEN(string reason);
  void destroy()
     context(Context_attribute);
  void open()
    raises(ALREADY_OPEN);
     context(Context_attribute);
  void close()
    raises(NOT_OPEN);
     context(Context_attribute);
}
```

```
Interface AppObj: FRAME::CommonObj {
  enum Operation { GETDATA, PUTDATA, OPENFRONTEND, GETMETADATA }
  struct Script {
    string language;
    string statements;
};
typedef sequence<dt> Seqobject;
void exchange(in Operation exchangetype, inout dt dataobj);
void convert(inout string format, in string propertyname, inout dt dataobj);
void query(in Script query, out dt responseObj);
void execute(in Script commandlist, in Seqobj in inputdataObject,
             out Seqobj outputdataobject )
```

Convert 연산은 응용들이 사용하고 있는 다양한 데이터 형식을 클라이언트가 요구한 형식으로 변환시켜 주는 연산이다. 각 응용들이 사용하는 데이터 형식은 매우 다양하며, 클라이언트에서는 현재의 형식과, 필요한 형식을 지정하고, 객체 구현에서는 이것을 요구된 형식으로 변환하며, 실패할 경우 예외 처리를 실행한다. 이때 트레이딩 서버는 클라이언트가 요구하는 변환 작업을 수행할 서버에 정보를 제공하여, 클라이언트의 동적 서버 접근을 보장한다.

Exchange 연산은 클라이언트와 서버의 데이터 교환 연산이다. 분산 객체들의 데이터 교환이 네트워크 전송 프로토콜에 투명하게 이루어짐을 보장한다.

Query 연산은 원하는 데이터를 미리 알 수 없을 때 데이터를 검색하기 위한 연산이다. 질의 서비스는 질의로 유도된 탐색의 결과를 새로운 데이터 객체로 생성하고, 클라이언트는 질의 연산을 이용해서 새로운 데이터 소스에 접근할 수 있다. 이 질의 연산 실행이 수행되기 위해서는 트레이더 서버에 질의 스크립트를 전달하고, 질의에 맞는 객체의 위치정보를 알 수 있을 때 새로운 데이터 객체를 반환하고, 그렇지 않으면 예외를 발생시킨다.

Execute 연산은 기존 응용들의 캡슐화를 보장하기 위하여 exchange 연산과 query 연산을 결합한 연산이다. 이 연산은 수행되어야 할 연산의 명령어 스크립트와 일련의 데이터 객체들을 객체 구현에 전달한다. Execute 연산은 클라이언트에게 단지 새로이 생성된 데이터 객체의 객체 참조만을 반환하여 실행에 대한 은폐성 및 분산 객체들에 대한 투명한 접근을 제공한다.

3.1.2 데이터 객체

데이터 객체 모델은 데이터 명칭과 값 쌍의 private 영역, CommonObj 객체로부터 상속받게 되는 open, close, destroy 연산, 데이터 객체의 추가 및 제거를 위한 add, delete 연산, 데이터 객체 값을 설정하는 set,

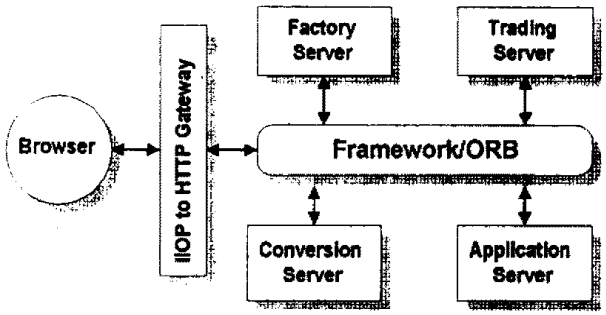
get 연산, 메타 데이터 정보를 제공하는 meta_describe 연산으로 구성된다. 데이터 객체내의 데이터 명칭과 값의 자료형은 동적 연산 활용을 위하여 IDL 자료형에서 "any"형으로 정의된다. 데이터 객체의 기본 자료형은 데이터 명칭과 값의 쌍으로 쉽게 정의될 수 있지만, 배열 또는 구조체로 표현되어야 할 객체들은 명칭과 값 객체 리스트에 추가로 구조적 데이터를 생성하여 테블릿 객체에 링크시킨다. 즉, 구조적 데이터 객체는 테블릿 형식으로 표현되는데, 명칭과 값 쌍으로 표현된 데이터 객체에 의해서 참조되도록 한다.

```
interface dt:FRAME::CommonObj {
  readonly attibute string objectType;
  struct FormattedDataRep {
    string format;
    any value;
  };
  exception ALREADY_EXITS (string reason);
  void add (in string newpropertyname )
    raises(NOT_OPEN, ALREADY_EXITS )
      context(Context_attribute);
  exception NOT_FOUND (string reason);
  void delete(in string newpropertyname )
    raises( NOT_OPEN, NOT_FOUND )
      context(Context_attribute);
  typedef sequence<string> Namelist;
  typedef sequence<TypeCode> Typelist;
  void describe(out unsigned long numberofproperties, out NameList names,
                out TypeList types )
    raises (NOT_OPEN)
      context(Context_attribute);
  exception TYPE_NOT_FOUND (string reason);
  exception PROPERTY_NOT_FOUND (string reason);
  void set (in string propertyname, in any value )
    raises (NOT_OPEN, NOT_FOUND, PROPERTY_NOT_FOUND )
      context(Context_attribute);
  void get(in string propertyname, out any value )
    raises (NOT_OPEN, NOT_FOUND, PROPERTY_NOT_FOUND)
      context(Context_attribute);
};
```

3.2 CORBA 기반 프레임워크 모델

제안하는 프레임워크는 프레임워크/ORB, 객체들의 위치 정보를 제공하는 트레이딩 서버(trading server), 데이터들의 변환을 담당하는 변환 서버(conversion server), 각종 데이터 객체들의 생성·복사·삭제를 담당하는 팩토리 서버(factory server), WWW와의 통합을 위한 게이트웨이(gateway)로 구성된다.

프레임워크/ORB는 클라이언트의 요구 및 객체 구현 결과를 전달하는 객체 버스로서 응용 객체 및 데이터 객체들에게 공통으로 필요한 인터페이스들로 구성된다. 변환 서버는 서로 다른 데이터 표현을 사용하는 응용들간의 자료 변환에 필요한 정보 및 변환 서비스를 지원하며, 트레이더 서버는 객체들간의 위치 투명성을 보장하기 위해 적절한 서비스를 동적으로 선택할 수 있게 하며, 팩토리 서버는 데이터 객체들에 대한 생명 주기 서비스를 제공하여 데이터 객체들을 생성·제거·관리하며, 기존 응용들을 객체 래퍼를 이용하여 재사용하기 위한 응용 서버로 구성된다. (그림 4)는 제안하는 프레임워크의 모델을 나타낸다.



(그림 4) CORBA 기반 프레임워크 모델
(Fig. 4) The architecture of Framework

3.2.1 팩토리 서버

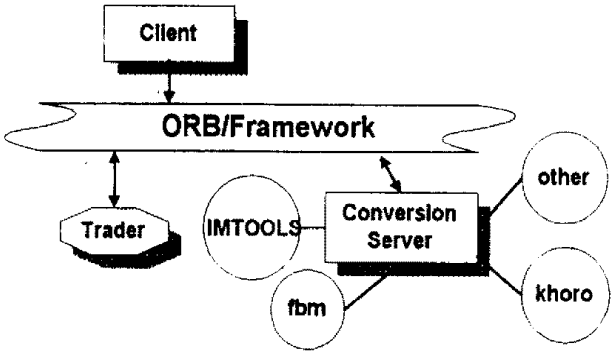
팩토리 객체는 단순 데이터 객체 및 구조적 데이터 객체들의 생명주기 서비스를 제공한다. 팩토리는 데이터 객체들을 생성, 제거, 관리하고 분산된 데이터 객체를 액세스할 때 애플리케이션 객체에 의해서 개방되고 닫혀진다. 팩토리 객체의 private 영역은 참조할 데이터들의 리스트를 포함하고 있고, open, close, destroy, create_dat, create_tab, open_dat, copy_tab 연산을 이용하여 데이터 객체들을 생성하거나 복사한다. 클라이언트는 query 연산을 이용하여 어떤 데이터 객체의 상태나 목록을 얻게 되며, 팩토리는 객 데이터 객체에 대한 내부 상태나 초기화 등의 연산에 대한 정보를 가지고 있게 된다. 팩토리 객체의 IDL 정의는 다음과 같다.

```
interface ft: FRAME::ap {
void create_dat(in string type, out dt dataobjecthandle)
    context(Context_attributes );
typedef sequence<string> ColList;
typedef sequence<TypeCode> TypeList;
void create_tab(in long maxrows, in long maxcols, in ColList columnnames,
                in TypeList types, out tb handle)
    context(Context_attributes );
void copy_dat(in dt dataObject, out dt dataObjectCopy)
    context(Context_attributes);
void copy_tab(in tb tableObject, out tb tableCopy)
    context(Context_attributes );
};
```

3.2.2 변환 서버

오늘날 대부분의 응용들은 각기 고유의 데이터 형

식을 이용하고 있다. 따라서 응용간 자료 교환이 어려우며, 필요시 특정 형식을 어떤 공통 형식으로 변환하는 기능을 이용하거나, 특정 자료 변환 유틸리티를 이용하여 데이터를 변환하게 된다. 변환 서버는 이러한 과정을 사용자에게 숨기고, 프로그램 수행 시 동적인 데이터 변환이 가능하게 하는 서버이다. 객체 구현에서 클라이언트에서 필요한 데이터 형식을 지원하지 않으면, 트레이더 서버를 이용하여 적당한 변환 작업을 제공하는 서버를 찾고, 이를 호출하여 적절한 변환이 이루어지게 하는 것이 변환 서버의 기능이다. (그림 5)는 변환 서버의 구조를 보여준다.

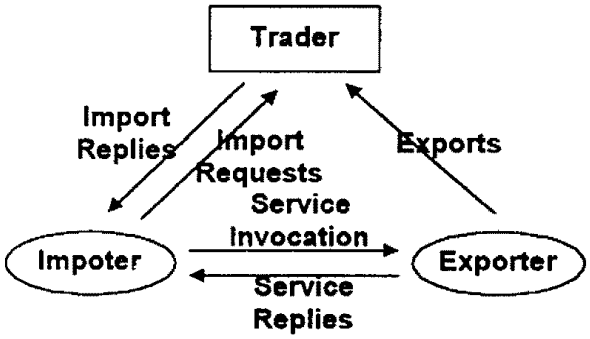


(그림 5) 변환서버의 구조
(Fig. 5) The architecture of Conversion Server

3.2.3 트레이딩 서버

트레이딩 서버는 클라이언트/서버 응용의 클라이언트(importer)가 실시간 동안에 서버(exporter)의 적절한 서비스를 동적으로 선택할 수 있도록 한다. 시스템의 상태, 서비스, 서비스 구성 요소가 동적으로 변화되어도 적절한 서비스 제공자를 사용할 수 있도록 하는 서버이다. 트레이더와 서버와의 상호 작용은 (그림 6)과 같다.

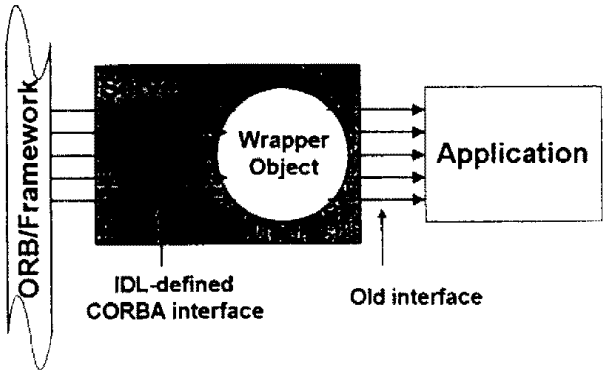
그림에서와 같이 트레이더는 분산 시스템에서 클라이언트가 적절한 서버를 찾도록 하는 제 3의 객체이다. 트레이더는 서버가 서비스 제공자를 알리기를 원할 때 서버로부터 서비스 제공자를 받는다. 서비스 제공자는 제공하기를 원하는 서비스의 속성을 갖고 있다. 트레이더는 클라이언트가 서비스를 원할 때 적절한 서비스 제공자를 찾아준다. 트레이더는 클라이언트의 서비스 요구를 만족하는 가장 적절한 서비스 제공자를 선택하여 반환한다. 성공적으로 수행된 후에 서비스를 요구하는 클라이언트는 서비스 제공자와 바인딩할 수 있다. 트레이더에 의해 실시간 동안에 적절한 서비스를 선택하는 것은 클라이언트가 그들의 요구를 만족시킬 수 있는 서버에 대한 사전 정보 없이 이루어질 수 있다.



(그림 6) 트레이더 서버의 구조
(Fig. 6) The architecture of Trader Server

3.2.4 응용 서버

응용 서버는 기존 응용들을 객체 래퍼(wrapper)를 통하여 통합하기 위한 서버이다. 객체 래퍼는 기존 시스템간의 통합을 위해 응용과 프레임워크와의 연결 수단으로서 응용과 CORBA와의 연결 수단으로 사용되며, (그림 7)과 같은 구조를 갖는다. 객체 래핑(wrapping)은 캡슐화 계층(encapsulation layer)을 통해 기존 프로그램에 대한 액세스를 제공한다. 객체 래퍼의 한 면은 기존 프로그램과 통신하며, 다른 면은 프레임워크와 통신할 수 있는 인터페이스로 구성된다. 이렇게 객체 래퍼를 사용하게 되면, 클라이언트 측에서는 기존 프로그램과 새로운 CORBA 객체 구현 사이에 특별한 구별이 필요 없게 된다.



(그림 7) 객체 래퍼
(Fig. 7) The Object Wrapper

객체 래퍼가 덧붙여진 시스템은 기존 다른 프로그램과의 인터페이스는 그대로 유지하면서 ORB를 통해 다른 CORBA 객체와 통신할 수 있게 된다. 이때 IDL과 API사이의 변환이 직접적인 경우 작은 크기의 래퍼를 이용해도 충분하지만, 파라미터의 변환 등으로 직접적으로 변환될 수 없을 경우에는 큰 래퍼를 필요로 하게 된다.

3.3 IIOP TO HTTP 게이트웨이

HTTP 기반의 WWW을 이용하는 클라이언트들이 분산된 환경에서 CORBA 객체로 된 서비스들을 사용하기 위해서는 CORBA 객체들의 통신을 담당하는 IIOP 기반의 ORB를 통하여 통신을 해야 하기 때문에 클라이언트 측에서는 IIOP 기반의 브라우저(browser)를 사용하여야 한다. 그러나 기존의 WWW를 사용하고 있는 클라이언트들의 수는 매우 방대하고 클라이언트 측의 브라우저를 모두 IIOP 기반 브라우저로 바꾼다는 것은 매우 많은 비용이 된다[16].

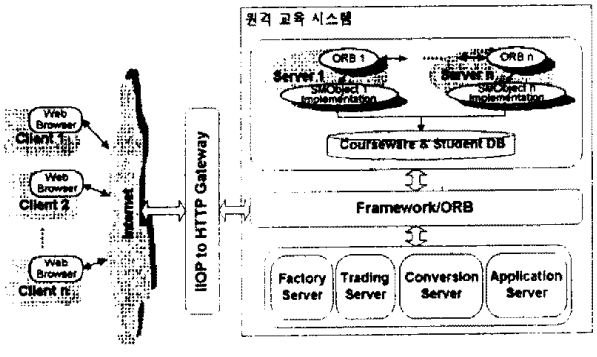
```
interface http
        struct header
        string name;
        string value;
        struct request
                string URL;
                sequence < header > headers;
        struct request_body
                string URL;
                sequence < header > headers;
                sequence < octet > body;
        struct response
                short respcode;
                string resptext;
                sequence < header > headers;
                sequence < octet > body;
/* HTTP 표준 전달 방식 */
        response GET(in request request);
response HEAD(in request request);
response POST(in request_body request);
response PUT(in request_body request);
response DELETE(in request request);
response LINK(in request request);
response UNLINK(in request request);
```

따라서 본 연구에서는 HTTP 기반의 WWW 사용자들이 클라이언트 측의 아무런 변화를 주지 않고 CORBA 객체 서비스를 사용할 수 있도록 하기 위하여 IIOP 프로토콜을 HTTP 프로토콜로 매핑을 수행하는 게이트웨이를 둠으로써 효율을 향상 시켰다. HTTP의 IDL 매핑은 다음과 같은 프로토타입을 갖는다.

HTTP 헤더는 명칭과 값 쌍으로 나타내어지고, 문자열로 인코딩된다. 요구는 문자열로 인코딩된 URL과 헤더들로 이루어져 있다. 이것은 PUT과 POST를 제외한 모든 메소드들을 만족시키기에 충분하다. PUT과 POST는 body를 필요로 한다. 따라서 요구는 또 다른 필드를 가진다. 옥탯(octet)은 8비트의 크기를 가지며, 통신 시스템에서 전달되어져 올 때, 어떠한 변환도 이루어지지 않는다. 모든 요구에 대한 응답은 상태 코드와 그것의 대응하는 스트링, 헤더들, 그리고 옥탯으로 나타나는 body들로 이루어져 있다. 이 IDL은 스터브(stub) 컴파일러를 통하여 헤더 파일과 클라이언트 스터브, 그리고 서버 스켈레톤(skeleton)을 만드는데, 이 때 스터브 컴파일러는 IDL 컴파일러를 사용한다.

4. 제안된 모델을 이용한 교육 시스템 설계

본 연구에서 제안한 프레임워크를 이용하여 실시간 원격교육시스템을 설계하였다[17]. 본 교육 시스템은 학습자에게 개별 학습을 지원하는 학습자 모듈과 코스웨어, 학생 정보를 가지는 데이타베이스를 프레임워크에 추가함으로써 설계된다. 설계된 교육 시스템의 학습자 모듈은 CORBA 객체들로 설계한다. 이를 통해서, 전체 시스템을 분산 시스템으로 개발함으로써, 기존의 웹 기반 교육 시스템에서 문제가 되는 서버의 지나친 부하문제의 해결이 가능하며, 또한 프레임워크와의 인터페이스를 용이하게 한다. 학습자는 웹 브라우저를 이용해서 교육 시스템에 접근하며, 시스템은 사용자의 정보와 상태에 따라서 적절한 코스웨어를 추출하여 사용자에게 전달한다. 설계된 원격교육시스템은 프레임워크를 통하여 여러 가지 서비스들을 제공받음으로써, 개발자가 보다 용이하게



(그림 8) 원격 교육 시스템
(Fig. 8) The Distance Learning System

응용을 개발할 수 있다. 또한, 객체사이의 호환성을 유지함은 물론 게이트웨이를 통하여 웹 브라우저를 이용할 수 있게 됨으로써 기존의 다른 시스템보다 많은 장점을 가지고 있다.

5. 결론 및 향후 연구과제

오늘날 이종간의 하드웨어, 소프트웨어의 통합은 중요한 논점으로 부각되고 있으며, 이를 실현하기 위한 여러 가지 제안들이 제시되고 있다. 시스템 통합을 위해 OLE/COM, OpenDoc, CORBA 등과 같은 분산객체기술을 이용한 모델들이 제시되고 있으며, 이중에서 CORBA는 600여개의 회사들이 참여해 객체 지향 기술을 기반으로 이기종 분산 환경에서 응용들의 표준을 제공하는 기술이다.

본 연구에서는 응용 프로그램들을 통합하기 위한 요구 사항을 만족하는 기반 서비스를 구현하기 위해, 각각의 기능을 수행하는 서버 및 게이트웨이로 구성되는 프레임워크를 설계하였다. 제안된 프레임워크는 프레임워크/ORB, 객체들의 위치 정보를 제공하는 트레이딩 서버, 데이터들의 변환을 담당하는 변환 서버, 각종 데이터 객체들의 생성·복사·삭제를 담당하는 팩토리 서버, WWW와의 통합을 통한 인터페이스를 제공하기 위한 게이트웨이로 구성되어 있다. 또한 다양한 응용에서 이용될 수 있는 서비스들을 객체화한 후 제공하여, 소프트웨어 재사용 및 시스템 통합을 통하여 향후 시스템 개선에 유연하게 대응할 수 있는 적응성을 제공한다. 시스템 통합 후 클라이언트는 브라우저를 통한 일관된 인터페이스를 제공받게 되는데, 이를 위해 ORB간 통신을 위한 프로토콜인 IIOP와 HTTP사이를 매핑해주는 게이트웨이(gateway)를 프레임워크에 포함시켰다. 또한 본 시스템의 신뢰성 및 문제점을 보완하기 위하여 설계 된 프레임워크를 멀티미디어 응용과 관련 된 원격교육시스템의 설계에 적용하였다.

향후 연구 분야로는 본 논문에서 제안한 프레임워크를 기반으로 다양한 분야의 업무에 적용하여 문제점을 파악하고, 이를 토대로 좀 더 신뢰성 있는 시스템 통합 프레임워크를 구축하는 것이다. 또한 분산 멀티미디어 응용과 관련된 원격 교육 등의 환경에 광범위하게 응용될 수 있도록 보완하는 것이다.

참 고 문 헌

[1] A. Leinwand and K. F. Conroy, *Network Management*, Addison-Wesley Publishing Company, Inc., pp. 17-36, 1996.

[2] I. Jacobson, *Object-Oriented Software Engineering*, Addison-Wesley Publishing Company, Inc., 1992.

[3] R. Orfali, D. Harkey and J. Edwards, *The Essential Distributed Objects Survival Guide*, John & Sons, Inc., 1996.

[4] T. J. Mowbray and R. Zahavi, *The Essential CORBA: System Integration using Distributeted Objects*, Objects Management Group, John Wiley & Sons, Inc., 1995.

[5] E. Wallace and K. C. Walln, "A Situated Evaluation of the Object Management Group's(OMG) Object Management Architecture(OMA)," 11th Annual Conference Proceedings of OOPSLA '96, San Jose, California, pp. 168-178, 1996.

[6] J. Siegel, *CORBA Fundamentals and Programming*, John Wiley & Sons, Inc., 1996.

[7] T. J. Mowbray, "Choosing between OLE/COM and CORBA," Object Magazine pp. 39-46, Nov. -Dec., 1994.

[8] G. Wilkie, *Object-Oriented Software Engineering*, Addison-Wesley Publishing Company, Inc., 1993.

[9] B. Meyer, *Reusable Software*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1994.

[10] OMG, "CORBA 2.0 Specification," http://www.omg.org/corba/iiop.html, 1995.

[11] OMG, "Control and Management of A/V Streams Request For Proposal," http://www.omg.org/library/schedule/, 1997.

[12] T. J. Brando, MITE *Document*, http://www.mitre.org/research/domis/reports/Orbix.html

[13] T. J. Brando, *DOMIS Implementation of CTAPS Functionality Using Orbix*, MITRE Co., December 1994.

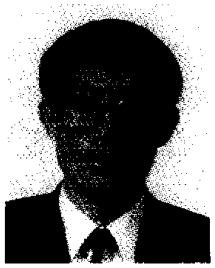
[14] 이종훈, 고희창, 김남용, 이세훈, 왕창종, "분산 객체 기술을 이용한 시스템 통합," 한국정보처리

학회 춘계학술발표논문집, 제4권 1호, pp. 577-581, 1997. 4.

[15] 홍찬기, 김정아, "프레임워크에 의한 재사용기법," 소프트웨어공학회지 제10권 제1호, pp. 46-55, 1997. 3.

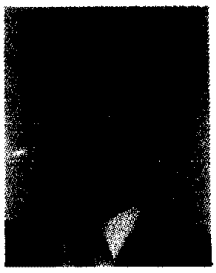
[16] 김찬권, 안치돈, 탁진현, 이세훈, 왕창종, "멀티미디어 서비스를 위한 WWW와 CORBA의 통합," 한국정보과학회 춘계학술발표논문집, 제24권 1호, 1997. 4.

[17] 이승근, 이윤수, 이세훈, 윤경섭, 왕창종, "WWW 상에서의 CORBA기반 교육시스템의 설계," 한국정보과학회 춘계학술발표논문집, 제24권 1호, 1997. 4.



김 남 용

1985년 인하대학교 전자계산학과(학사)
1989년 인하대학교 대학원 전자계산학과(이학석사)
1995년 인하대학교 대학원 전자계산학과 박사과정 수료
1990년~현재 신흥전문대학교 전산정보처리과 조교수
관심분야: Software Engineering, 분산 객체 컴퓨팅, 원격 교육



왕 창 종

1964년 고려대학교 물리학과(학사)
1975년 성균관대학교(경영학 석사)
1981년~1990년 인하대학교 전자계산소장
1992년~1993년 정보과학회 부회장, 전산교육연구회 위원장
1979년~현재 인하대학교 전자계산공학과 교수
관심분야: Software Engineering, 멀티미디어, 분산 객체 컴퓨팅, 전산교육