CORBA 기반의 군 정보체계 전환 기법의 설계 및 구현

장창복, 강우석, 조성훈, 김동혁, 이찬섭, 조완수*, 최의인
한남대학교 컴퓨터공학과
*국방과학연구소
e-mail:chbjang@dblab.hannam.ac.kr

Design and Implementation of System Migration Technique of Defense Information Systems based on CORBA

Chang-Bok Jang, Woo-Suck Kang, Sung-Hoon Cho,
Dong-Hyuk Kim, Chan-Seob Lee, Wan-Soo Cho*, Eui-In Choi
Dept of Computer Engineering, Han-nam University
*Agency for Defense Development

요약

현재 군 정보체계는 하부 구조와 개발 플랫폼의 정의 없이 정보 시스템 개발 사업이 진행되고 있다. 이러한 스토브 방식의 기존 정보 시스템을 개방 체계로 전환하기위해서는 분산 객체 기술을 적용한 공통적인 접근 방법이 필요하다. 하지만 현재의 군 정보 시스템은 C4I 시스템 개발 자원의 낭비와 현재 시스템간의 상호 운용성이 매우 낮은 실정이다. 본 논문에서는 이질 분산 환경에서 이러한 문제들을 해결하기 위해 분산 객체 미들웨어인 CORBA를 사용하여 시스템 통합 기술과 체계 전환 기술 및 절차를 확보하고, 상호운용 가능한 시스템의 테스트베드를 구축했다. 본 논문을 통해 확보된 체계 전환 절차는 체계 전환 개발 단계에서 지침서로써 사용될 수 있도록 하였으며, 개발 비용의 절감 효과가 있다.

1. 서론

이질 분산 환경인 군 시스템은 공통적인 정보체계 기반구조 정립 및 개발 플랫폼의 정의 없이 정보체계 개발 사업이 진행됨에 따라 정보체계 개발 사업 간의 기술 공유 및 체계 재사용이 매우 미흡하다. 기존의 스토브파이프(stovepipe) 방식의 정보체계로는 분산 객체 기술을 이용한 개방 체계로 전환하고자하는 공통적 접근 방법이 개발되어 있지 못하다. 이로 인해 국방정보체계 및 C4I 체계 개발에 대한 국방 자원의 부분적 낭비가 발생하고 있고, 체계 간 상호운용성도 매우 낮은 수준에 머무르고 있다. 따라서 이를 해결하기 위해 분산 객체 미들웨어 환경에서 효율적인 정보 교환을 지원할 수 있는 하부구조인 CORBA를 이용한 시스템 통합 기술과 분산 객체 미들웨어에 의한 체계 전환 기술 및 절차 확보에 관한 연구가 필요하다.

본 논문에서는 다양한 정보기술에 의해 개발된 정보체계 및 C4I 체계를 상호연동하고, 국방 자원 투자 보호 및 분산 컴퓨팅 환경에서의 상호 운용성을 보장하기 위해 분산객체 미들웨어인 CORBA를 이용하여 클라이언트 계층과 이 기종 시스템 계층 간의 인터페이스를 구현하였다. 특히, IDL은 클라이

이 논문은 2001년도 국방과학연구소의 연구비 지원에 의하여 수행되었음.

언트 계층이 이 기종 시스템 계층의 다양한 정보체계에 대해 분산된 객체 및 다양한 데이타 자원들의 투명적 접근, 분산된 데이타들의 투명적 교환, 다양한 데이타 형식의 자유로운 변환, 기존 애플리케이션들의 재사용 기능 등의 구체적인 방법을 제공한다[2, 3]. 또한, 기존 시스템에 접근할 때 이용하는 다양한 인터페이스를 IDL 인터페이스로 대응시키는 효과적인 래핑 메커니즘을 제공함으로써 기존 시스템을 보다 쉽고 강력하게 분산 환경으로 전환할 수 있도록 하였다.

2. 관련 연구

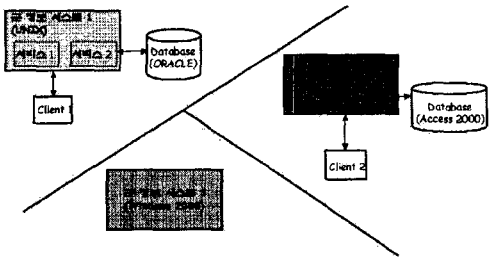
현재 분산 객체 미들웨어 기술 중 대표적인 것은 OMG의 CORBA, Microsoft의 DCOM, SUN의 RMI 등이 있으며, CORBA(Common Object Request Broker Architecture)가 가장 광범위하게 사용되고 있으며 사실상의 표준으로서 많이 연구되고 있다[7]. 이러한 분산 객체 미들웨어 기술을 사용하여 체계전환한 대표적인 프로젝트로는 MITRE의 DOMIS, METU의 MIND, Dublin City 대학의 OASIS 프로젝트가 있다.

DOMIS(Distributed Object Management Integration System) 프로젝트는 MITRE에서 미 공군의 CTAPS(Contingency Theater Automated Planning System)에 분산 객체기술을 적용하여 DOM(Distributed Object Management) 구조로 변환하는 기술과 방법론을 개발한 연구이다[1, 4].

METU(Middle East Technical University)의 MIND(METU Interoperable Database System)는 멀티 데이타베이스 시스템에서 이질적이고, 연방화된 DBMS들 사이에서의 상호 운용성을 구현한 연구이고[5], Dublin city 대학에서 추진한 OASIS(ODMG Architectures for Specification of Interoperable System)은 의료 환경에서 사용하기 위한 멀티데이터베이스 프로토타입의 구성에 초점을 맞춘 연구이다[6].

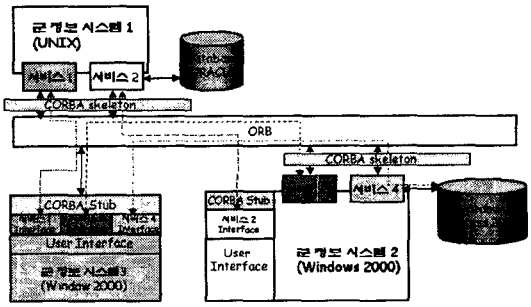
3. 시스템 구조

현재의 군 정보 시스템은 [그림 1]처럼 이질적인 환경으로 구성되어있다. 따라서 시스템 간의 상호 운용이 불가능한 상태여서 시스템에 중복되게 제공되는 서비스들이 존재한다.



[그림 1] 군 정보 시스템 구조

이러한 군 정보 시스템을 상호 운용 가능한 시스템으로 전환하기 위해서 분산 객체 미들웨어인 CORBA의 ORB와 IIOP를 사용하였고, IDL(Interface Definition Language)를 이용하여 서버와 클라이언트간의 인터페이스를 정의하였다.



[그림 2] CORBA를 이용한 정보체계 전환 구조

CORBA 기반 정보체계로 전환하였을 경우 새로운 시스템이 도입되어도 기존의 각 정보 시스템에서 제공하는 서비스들을 새로 개발할 필요 없이 해당 서비스의 인터페이스만을 호출함으로써 서비스를 이용할 수 있게 된다. CORBA를 이용한 정보체계 전환 구조는 [그림2]와 같다.

4. 체계 전환 절차

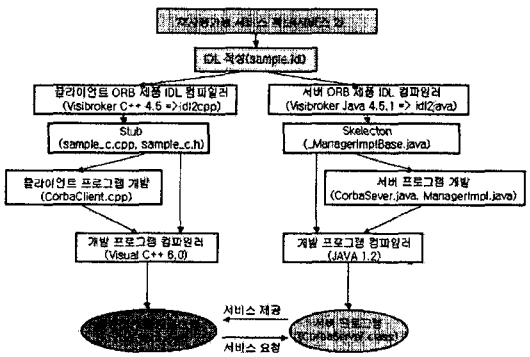
기존 시스템을 상호 운용 가능한 형태로 변환하기 위한 과정을 간략히 설명하면 다음과 같다.

[단계 1] 요구 사항 분석 및 CORBA 지원 제품 사용법 습득
[단계 2] 환경 구축
[단계 3] 소스 코드 분석
[단계 4] 기존 코드의 재모듈화
[단계 5] 사용 가능한 모듈의 호출 인터페이스 설계
[단계 6] 통합 테스팅

5. 클라이언트/서버 어플리케이션 개발

본 논문에서 제시한 체계 전환 절차를 통해 실제 시스템을 통합하기 위해서 [그림 3]과 같은 어플리케이션 개발 절차가 필요하다.

본 논문에서는 개발 환경으로 윈도우 NT와 유닉스 플랫폼을 사용하였고, 프로그램 개발 툴로 C++와 자바를 사용하였다. CORBA 지원 어플리케이션으로는 Inprise사의 Visibroker를 사용하였다[8, 9, 10].



[그림 3] 서비스 2 이용을 위한 개발 절차

개발 절차는 다음과 같다.

가. 재사용 가능한 서비스 확보

재사용 가능 서비스 추출 단계를 통해 서비스를 확보한다. 본 연구에서는 [그림 2]의 유닉스 운영체제 환경인 군 정보 시스템 1에서 제공되는 서비스 2를 구현 객체로써 정의하고, 이를 윈도우에서 호출하여 사용하였다. [그림 4]는 오라클 데이터베이스내의 PERSON 테이블의 내용을 검색하고, 갱신하는 서비스 중에서 검색 부분이다.

```
import java.sql.*;
import java.lang.*;
public static void main(String[] args)
{
 try {
     String url = "jdbc:oracle:oci8:@";
     DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
     Connection conn =
                  DriverManager.getConnection (url, "scott", "tiger");
     stmt = conn.createStatement ();
     Statement stmt = db.createStatement();
     ResultSet rs = stmt.executeQuery("SELECT * FROM PERSON");
     }
catch (SQLException e) {
     System.err.println("SQL Error : " + e.getMessage());
     }
}
```

[그림 4] PERSON 테이블 검색

나. IDL을 이용한 클라이언트와 서버의 인터페이스 설계

IDL을 통해서 클라이언트와 서버의 인터페이스를 설계한다. 인터페이스 설계 시 [그림 5]와 같이 데이터베이스(ORACLE 8i)에 존재하는 데이타를 검색하기 위해 접근하고자 하는 테이블과 같은 구조로 데이타를 정의하였다. 본 논문에서는 PERSON 테이블의 데이타를 검색하는 서비스를 이용하므로 PERSON 테이블과 같은 구조의 Person 구조체를 정의하였다. 또한 테이블 검색을 위한 연산으로 select() 연산을 정의하였다.

작성된 IDL 파일을 개발 언어에 맞는 ORB 제품 컴파일러로 컴파일하여 스터브(Stub)와 스켈렉톤(Skeleton) 파일을 생성한다. 본 논문에서는 클라이언트 측에서는 Visibroker C++ 4.5를, 서버 측에서는 Visibroker Java 4.5.1을 사용하였다.

```
module sample
{
 exception DBException
  {
   string reason;
  };
 struct Person
 {
      string snumber;
      string name;
      string addr;
      string email;
      string phone;
 };
 typedef sequence<Person> PersonSeq;
 interface Manager
 {
      PersonSeq select() raises (DBException);
      void insert (in Person p) raises (DBException);
      void remove (in string snum) raises (DBException);
 };
};
```

[그림 5] IDL 작성

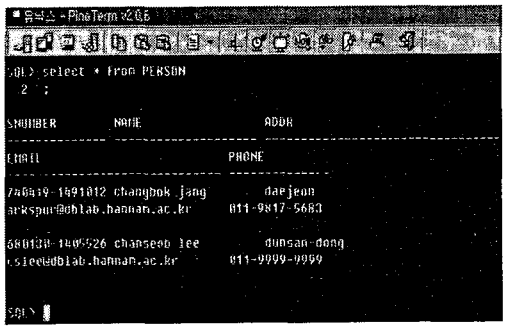
다. 프로그램 개발

생성된 스터브와 스켈렉톤의 클래스를 이용하여 클라이언트 측 프로그램과 서버 측 프로그램을 작성하였다. 스터브와 스켈렉톤은 통신을 위한 인터페이스를 정의하는 클래스들로 구성되어 있다. 실제 프로그램에서는 이런 클래스를 상속받아 이용한다. 개발 툴로는 비주얼 C++과 자바를 이용하였다.

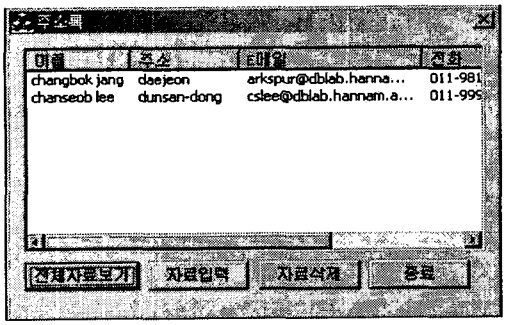
라. 컴파일 후 프로그램 실행

클라이언트와 서버 프로그램을 모두 작성한 후에 각각을 컴파일하고 링크하여 실행 파일을 만든다. [그림 6]는 유닉스 환경에서 직접 데이터베이스에 접근하여 질의를 하였을 때 그 결과를 보여주는 화면이고, [그림 7]는 서버 측에 구현되어 있는 서비스

를 클라이언트(윈도우 환경)가 이용하여 질의를 하였을 때 그 결과를 보여주는 화면이다. 두 경우의 결과가 같음을 알 수 있다.



[그림 6] 서버에서 데이터베이스 검색 결과



[그림 7] 클라이언트에서 데이터베이스 검색 결과

6. 결론

본 연구에서는 분산 객체 미들웨어 기술을 이용하여 이질적인 군 정보 시스템을 상호 운용 가능한 형태로 전환할 수 있는 방안을 연구하였다.

구현된 CORBA 객체는 객체에 접근할 수 있는 인터페이스만 알고 있으면 인터페이스 호출함으로써 서비스를 이용할 수 있다. 그러므로 이질적인 군 정보 시스템에서 상호 운용 가능하도록 체계 전환을 하기 위해서는 상호 운용하려는 서비스들을 확보하고, 다른 시스템에서 사용 가능하도록 하기 위해 구현객체로 정의하며, 구현객체에 접근하기 위해 IDL을 이용한 구현객체와 클라이언트사이의 인터페이스를 정의한다. 본 연구에서는 상호 운용 가능한 서비스를 확보하기 위한 방안과 체계전환을 위한 기법을 제안하였다.

향후 연구되어야 할 과제로는 소프트웨어 개발 비용 절감 및 효율적 개발 방법을 제시할 수 있는 컴포넌트 기반 개발 기법을 이용하여, 상호 운용하고자 하는 CORBA 객체 정의 시에 적용할 수 있는 컴포넌트 개발 방안이 연구되어야 할 것이다.

참고문헌

[1] Distributed Object Management Integration System (DOMIS) FY 94 Final Report, MITRE

[2] M. T. Roth, P. Schwarz, "A Wrapper Architecture for Legacy Data Source", IBM Almaden Research Center

[3] S. Vinoski, "CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments", IONA Technologies. Inc

[4] Distributed Object Management Integration System(DOMIS) FY96 Final Report, MITRE

[5] Dogac, A., et. al., "METU Object-Oriented Database system", SIGMOD Record, Vol. 24, NO. 3, Sept. 1995

[6] M. Roantree, J. Murphy, W. Hasselbring, " The OASIS Multidatabase Prototype ", ACM SIGMOD Record, 28:1, March 1999

[7] Orfali, R., Harkey, D., and Edwards, J., The Essential CORBA: System Integration Using Distributed Objects, Wiley, 1996

[8] VisiBroker for Java Programmer's Guide, Inprise Corporation, 100 Enterprise Way, Scotts Valley, CA 95066-3249

[9] VisiBroker for C+ Programmer's Guide, Borland Software Corporation, 100 Enterprise, Way Scotts Valley, CA 95066-3249

[10] Oracle8 I JDBC Developer's Guide and Reference Release 3 (8.1.7), July 2000, Part No. A83724-0, Oracle