산업기술연구소 논문집, 3, 183-192, (1997), 서경대학교
Journal of Institute of Industrial Technology, 3, 183-192, (1997),
SeoKyeong University

COM/OLE를 기반으로한 전문가시스템 추론기관의 컴포넌트화

민 미 경*

Abstract

In this paper an inference engine of expert systems is proposed, which is based on the Component Object Model(COM) and has the feature of OLE(Object Linking and Embedding) automation server. It supports compatibility with other systems based on the Component Object Model. Also, as it has the feature of OLE automation server, it provides OLE controller with extensible inference interface. With making the inference engine as a component of expert systems based on COM/OLE architecture, the efficiency of services and reusability can be enhanced.

1. 서 론

전문가 시스템은 어떤 특정 영역의 문제를 해결함에 있어 전문가를 모방한 컴퓨터 모델을 이용한다. 문제를 해결함에 있어 전문가의 지식으로 지식베이스를 구축하고 이러한 지식을 이용하여 전문가가 결론에 도달하는 과정대로 추론을 행하는 컴퓨터 시스템이다[3]. 기존의 전문가 시스템은 현재의 기술적인 면에서 보면 많은 문제점을 가지고 있다. 첫째, 많은 시간과 노력을 투자하여 만듦에도 불구하고 시스템의 사용범위가 제한되어 있다. 전문가 시스템과 다른 시스템간에 서로 서비스를 제공받기 위한 공통된 인터페이스가 존재하지 않으므로 서비스를 서로 주고받을 수가 없다. 둘째, 전문가 시스템이 DLL(Dynamic Linked Library) 형태로 추론 서비스를 제공할 경우 많은 문제점을 나타낸다. 그 예로 버전(Version) 문제를 들 수 있는데 이를 해결하기 위해 여러 버전의 DLL들이 공존할 수밖에 없다. 셋째, 구축된 시스템의 지식베이스가 특정 영역에 제한되어 있으므로 여러 응용 분야에 적용하기에는 부적합하다[6].

* 서경대학교 컴퓨터과학과 조교수

본 논문에서는 분산 지능형 시스템 구축 기반 환경[4]을 만들기 위해, 컴포넌트 개념을 전문가 시스템의 핵심 모듈인 추론기관에 적용시켜 컴포넌트화된 추론기관을 설계하고자 한다. 제안된 시스템은 첫째, COM을 기반으로 한 구조를 따르고 있으므로 시스템간의 인터페이스가 존재하여 서비스의 제공에 신뢰성을 줄 수 있다. 인터페이스란 시스템간의 커뮤니케이션 메커니즘을 담당하는 부분이다[1]. 이러한 인터페이스가 COM 구조의 바탕을 이루고 있으므로 잘 정의된 인터페이스의 구현이 핵심이다. 둘째, 인프로세스 서버(In-Process Server), 지역 서버(Local Server), 원격 서버(Remote Server) 등의 여러 가지 모델을 위치 투명성(Location Transparency)을 통하여 단 하나의 모델만으로 모든 종류의 클라이언트와 서버간 커뮤니케이션을 구축한다[4]. 위치 투명성이란 서버가 어떤 곳에 존재하든지 간에 클라이언트는 상관없이 서버의 객체를 사용할 수 있다는 것이다. 셋째, 전문가 시스템과 DBMS와의 접속을 통하여 특정 영역에 국한되지 않는 시스템 구축이 가능하다. 여러 분야의 지식베이스를 DBMS에 저장하고 이러한 지식을 ODBC를 통하여 응용 영역에 맞게 적용한다. ODBC(Open DataBase Connectivity)란 DBMS 와 응용 프로그램간의 추상적인 인터페이스 계층을 의미한다[5]. 이 계층을 통하여 응용 프로그램이 쉽게 DBMS 에 접속할 수 있다.

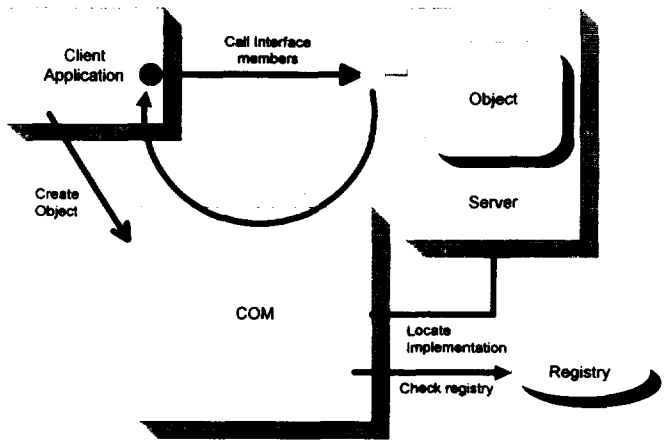
본 논문의 구성은 다음과 같다. 제 2장에서는 본 시스템의 기반인 COM에 대하여 기술하고 제 3장에서는 서비스의 확장을 가능하게 한 OLE 오토메이션에 대해 기술한다. 제 4장에서 추론엔진의 컴포넌트화 기법을 기술한 후 마지막으로 제 5장에서 결론을 내린다.

2. COM(Component Object Model)

COM 은 OLE의 기반이 되는 객체 모델에 사용되는 객체 시스템의 기반 프레임워크이다[1]. COM의 구성요소에는 V-Table, Interface, GUID(Globally Unique IDentifier), IID(Interface IDentifier), Registry가 있다. 클라이언트는 자신이 서비스를 얻고자 하는 서버의 GUID를 COM에게 제공한다. COM은 이것을 레지스트리를 통하여 해당 서버와 일대일 대응시킨다. COM은 서버 내부의 객체에 대한 초기 인터페이스 포인터를 클라이언트에 제공한다. 초기 인터페이스의 QueryInterface 함수를 통하여 나머지 인터페이스의 IID를 얻는다. 서비스를 제공하는 인터페이스의 멤버 함수는 V-Table을 통하여 접근할 수 있다. 클라이언트가 레지스트리를 통하여 서버 객체의 인터페이스를 <그림 1>과 같이 얻을 수 있다

GUID란 DCE(Distributed Computing Environment)내에서 각 객체 식별자들이 충돌을 일으키지 않도록 16비트 범용 고유 식별자 표준을 둔 것이다. 각 컴퓨터의 48비트 네트워크 어댑터 ID와 현재 날짜 및 시간을 사용하여 만들어 진다. 이 GUID는 시

간과 공간상에서 유일무이한 값을 갖는다. COM은 GUID를 사용하여 모든 고유 식별자 문제를 해결한다. DCE 표준에 따르면 GUID는 { 01234567 - 1234 -1234 -1234 - 012345678AB } 형식의 16진수로 표시된다. 이러한 GUID는 클라이언트가 서버 객체의 서비스를 요구할 때 레지스트리 내에서 사용된다.



그 림 1. 객체 위치 지정과 활성화 클라이언트

인터페이스란 클라이언트와 서버 객체 사이에 있는 하나의 협약이다. 이와 같은 인터페이스를 통해서만 클라이언트와 서버 객체 사이의 커뮤니케이션이 이루어진다. 인터페이스는 COM 확장 기능을 통해 표준 DCE 인터페이스 정의 언어(Interface Definition Language) 형식에 맞게 서술한다.

```
[ uuid ( 188a06c0 - 325a - 213f - 1531 - 23bc1167523 ), object ]
Interface IRete : IUnknown
{
  HRESULT Load ( BSTR fb, BSTR rb ) ;
  HRESULT Run ( ) ;
  HRESULT Visible ( SHORT flag ) ;
}
```

위의 예는 IRete 인터페이스 형태를 보여준다. COM에서는 하나의 객체가 여러 개의 인터페이스를 가질 수 있다[1]. 이러한 다중 인터페이스의 개념이 버전 문제를 해결한다. 버전이 상승될 때마다 새로운 인터페이스가 추가되고 서비스의 확장이 이루어진다. 각 인터페이스는 서로의 간섭 없이 서비스를 제공해 준다. IUnknown이란 인

터페이스는 다른 모든 인터페이스의 근간이 되고 그들과 다형적인 관계를 갖고 있다. 모든 인터페이스의 첫 부분에는 세 개의 IUnknown 멤버 함수가 있다. AddRef, Release 그리고 QueryInterface가 그것이다. AddRef는 객체의 레퍼런스 카운터를 증가시킨다. Release는 객체의 레퍼런스 카운터를 감소시킨다. 이 카운터가 0이 되었을 때 그 객체를 삭제한다. QueryInterface는 객체에 대한 다른 인터페이스의 존재 여부와 포인터를 관리한다. 따라서 IRete 인터페이스는 QueryInterface, AddRef, Release, Load, Run, Visible 함수를 가지는 V-Table의 포인터가 된다.

IID는 객체가 다중 인터페이스를 가질 경우 각 인터페이스를 구분해주는 인터페이스 ID를 의미한다.

Registry란 WIN32 환경 하에서 각각의 객체를 중앙 집중식으로 관리하기 위한 시스템이 정의한 일종의 데이터베이스이다. 레지스트리 구조는 다음과 같다.

HKEY_CLASSES_ROOT

- CLSID

- { 01234567 - 1234 - 1234 - 1234 - 012345678AB }

  - InprocServer32 = < path to server >

위의 경우는 인프로세스 서버나 지역 서버에 관한 레지스트리이고 원격 서버(Remote Server)로 확장될 경우에는 기계 이름까지 등록해 주어야 한다. COM은 CLSID와 서버를 연결시키기 위해 레지스트리를 검색한다.

V-Table은 인터페이스 포인터가 가리키고 있는 테이블로서 이 테이블 안에 객체 메소드에 대한 포인터를 저장하고 있다. 따라서 인터페이스 포인터를 사용하여 객체 멤버 함수를 호출할 수 있다. 그 형식은 다음과 같다.

```
IRete -> Load ( fbase, rbase ) ;
IRete -> Run ( ) ;
IRete -> Visible ( TRUE ) ;
```

인터페이스 정의부에는 모든 인터페이스 함수에 대한 인수 타입이 서술되어 있으므로 컴파일러가 사용자 대신 이들과 관련된 타입을 검사해줄 수 있다.

3. OLE 오토메이션(Automation)

OLE(Object Linking and Embedding) 오토메이션은 객체 시스템이 제공하는 다른 프로그램의 기능 중 일부를 실행 시간에 이용하는 것이다[1]. OLE 문서 서버

(Document Server)와 마찬가지로 오토메이션에도 서버와 컨트롤러(혹은 클라이언트) 두 종류의 프로그램이 존재한다. 즉, OLE 오토메이션 서버로 동작하는 프로그램이 있고 그 프로그램의 기능 중에서 이용하고자 하는 것이 존재할 경우 처음부터 그런 기능을 만들어내는 대신 자신의 프로그램을 오토메이션 컨트롤러로 작성하면 다른 프로그램의 그 기능을 이용할 수 있다. 오토메이션의 가장 큰 특징은 단 방향으로만 제어가 가능하다는 것이다. 즉, 컨트롤러에서 서버로의 호출만 가능하다. 서버는 자신에게 발생한 이벤트 등을 컨트롤러에게 알릴 수 없다. OLE 오토메이션 서버가 자신의 기능을 다른 프로그램들에게 노출시키기 위해 프로퍼티(Property)와 메소드(Method)를 이용한다. 프로퍼티란 대개 윈도우나 컨트롤 등의 객체 속성을 설정하는데 사용되며, 메소드란 그 객체를 제어하는데 이용된다. 이는 IDispatch 라는 인터페이스의 구현을 통해 이루어'진다. 오토메이션 서버의 작성 시에는 그것의 프로퍼티와 메소드를 ODL(Object Description Language)로 기술한 텍스트 화일이 필요하다. 이 파일을 ODL 컴파일러로 컴파일하면 타입 라이브러리(Type Library)를 생성한다. 오토메이션 컨트롤러는 이 파일을 마치 DLL 을 사용할 때 임포트 라이브러리(Import Library)를 사용하듯이 링크하여 사용한다. 그러면 오토메이션 컨트롤러에서 오토메이션 서버의 기능을 사용할 수 있다. ODL로 작성된 메소드와 프로퍼티의 예는 다음과 같다.

```
[ uuid ( 127458cf - 3411 - 4673 - 4476 - bbcf7648cba ) ]
DispInterface  IRete
{
   Properties:
      SHORT state ;
   Methods:
      void Load ( BSTR fb, BSTR rb ) ;
      void Run ( ) ;
      void Visible ( SHORT winflag ) ;
}
```

OLE 오토메이션 서버는 DLL 객체로 존재하는 인프로세스 서버(In-Process Server)와 EXE 객체로 존재하는 지역 서버(Local Server)로 나누어진다. 지역 서버는 32 비트와 16 비트간의 연동이 쉽고 서버내에서 메시지 루프를 직접 제어할 수 있으며 마샬링(Marshaling)의 오버헤드를 지닌다. 인프로세스 서버는 크기가 작고 속도가 빠르며 마샬링이 필요 없다. 마샬링이란 프로세스간 상호 커뮤니케이션을 담당하는 메커니즘이다. 인터페이스 포인터를 마샬링 한다는 것은 포인터와 객체 프로세스간의 연결에 필요한 정보를 담은 마샬링 패킷을 만든다는 것을 의미한다. 이렇게 생성된

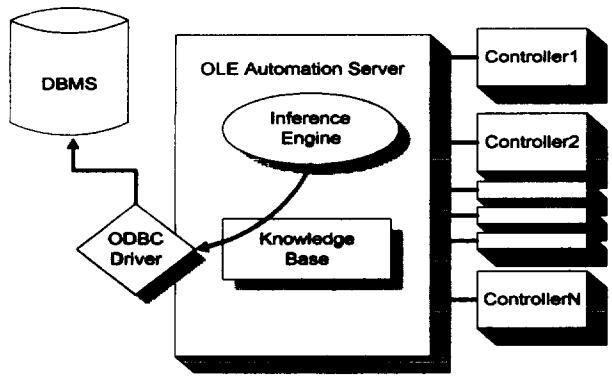
패킷은 클라이언트 프로세스로 전달되고 클라이언트 프로세스의 인터페이스 포인터로 바뀐다. 그 다음 클라이언트가 이 인터페이스 포인터를 이용해서 원하는 호출을 하게 된다. Win32 에서 마샬링은 RPC (Remote Procedure Call)로 이루어지는데 이는 네트워크 상에서도 동작이 가능하다. 따라서 분산 COM 환경 하에서도 서비스가 가능하다. 네트워크에서 서로 다른 시스템에 있는 프로세스간의 RPC는 패킷 송수신의 형태로 이루어지며 하나의 지역 시스템의 프로세스 사이에 RPC는 메시지를 주고 받는 방법으로 이루어진다.

4. 추론 서버와 컨트롤러

4.1 추론기관의 컴포넌트화

전문가 시스템은 사실과 규칙으로 표현되는 지식베이스와 이를 이용하여 추론을 행하는 추론기관, 사용자 인터페이스의 세 부분으로 구성되어 있다. 추론기관은 지식베이스를 이용하여 실질적인 추론을 행하는 역할을 하며, 작업 메모리 상에 존재하면서 추론을 진행할 수 있도록 도와준다[3]. 또한 추론기관은 사용자와의 대화를 통해 결론을 알리거나 부족한 정보를 받아들이는 일을 행한다. 본 논문에서는 이러한 추론기관을 컴포넌트화 하였다. 그 이유는 첫째, 타 시스템에서, 특히 지능형 시스템일 경우, 추론기관의 서비스를 반드시 필요로 한다. 둘째, 추론기관의 복잡성과 대형성으로 인하여 구현에 있어 상당한 어려움이 있다. 따라서 이미 구현되어 있는 추론기관의 서비스를 개방함으로써 중복될 수 있는 수고와 노력을 줄인다. 셋째, 프로그래밍 모델이 점차 단일 기능만을 가지고, 쉽게 이식될 수 있는 개방형 컴포넌트 구조로 변화하고 있다. 여기에 추론기관의 컴포넌트화는 반드시 필요한 것이다.

본 논문에서 컴포넌트화한 추론기관은 W/S용 전문가 시스템 구축 언어인 HEXPERT[7]의 추론기관이다. HEXPERT의 추론엔진은 Rete 알고리즘[2]을 사용하여 규칙 지식 표현 방법과 객체지향 사실 표현 방법을 사용하는 지식베이스를 추론할 수 있도록 변형한 것이다. HEXPERT 추론기관을 Windows-NT에서도 운용할 수 있도록 플랫폼 변경 작업을 한 다음, COM과 OLE 오토메이션 기술을 이용하여 추론기관의 분산객체 구조에 적합한 인터페이스를 설계하였다. COM/OLE를 기반으로 한 전문가 시스템의 전체구조는 <그림2>와 같다. 제안되는 시스템의 전체 구조는 COM을 기반으로 하고 OLE 오토메이션 서버의 기능을 갖춘다. 지식베이스를 저장할 DBMS는 WIN32 환경에서 클라이언트/서버 구조를 갖는 SQL Server를 채택한다. 본 시스템과 SQL Server와의 인터페이스는 ODBC(Open DataBase Connectivity)라는 추상계층을 이용한다. ODBC는 특정 DBMS에 종속되지 않는 특징을 가져 범용적인 DBMS 접근에 적합하다[5].



그 림 2. COM/OLE를 기반으로 한 전문가 시스템의 전체 구조

4.2 DBMS 접속

지식베이스를 적재함에 있어 데이터베이스에 저장된 지식베이스를 ODBC(Open DataBase Connectivity) 미들웨어를 이용하여 접근한다. ODBC는 특정 DBMS 에 종속적인 문제점을 해결하기 위해 드라이버와 라이브러리를 분리하여 각 데이터베이스마다 서로 다른 드라이버를 제공하고 응용 프로그램 개발자는 라이브러리에 정의된 API(Application Programming Interface)를 이용하여 개발을 할 수 있도록 한다. ODBC의 구성요소는 응용 프로그램(Application), 드라이버 매니저(Driver Manager), 드라이버(Driver), 데이터 소스(Data source) 이다.

드라이버 매니저의 역할은 ODBC 드라이버를 적재하는 것이다. 그 외에 ODBC 초기화를 담당하고, 에러와 상태를 확인하고, 응용 프로그램 함수 호출을 드라이버에 전달한다. 드라이버의 역할은 데이터 소스와의 접속을 담당하고, 함수 호출에 대한 결과를 응용 프로그램에 리턴하며 커서를 생성하고 관리한다. 데이터 소스의 역할은 드라이버로부터 제공받은 SQL문을 처리한다. 또한, 트랜잭션, 동시성 제어 등의 일반적인 DBMS와 관련된 작업을 처리한다. ODBC 인터페이스는 환경 핸들(Environment Handle), 연결 핸들(Connection Handle), 문장 핸들(Statement Handle)과 같은 3가지 타입의 핸들을 정의하고 있다. ODBC를 이용하여 원하는 정보를 가져오기 위한 과정은 다음과 같다.

1) ODBC 인터페이스를 초기화하기 위해서 환경 핸들을 할당한다. 환경 핸들은 응용 프로그램에 의해서 반드시 한번만 수행되어야 한다.
2) 응용 프로그램이 각각의 드라이버와 접속하기 위해서 연결 핸들을 할당한다.

3) 데이터 소스 이름과 사용자 ID , 암호를 확인하여 원하는 데이터 소스에 접속한다.

4) SQL 문을 위한 문장 핸들을 할당한다.

5) 필요에 따라 C++ 변수와 데이터 소스내의 필드를 바인드 시킨다.

6) SQL 문을 수행하고 결과 값을 받아온다.

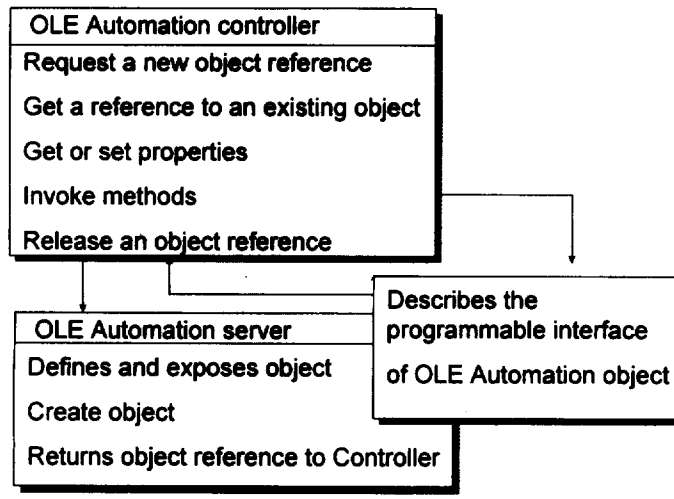
서버의 기능은 크게 코어 단계, 확장 단계1, 확장 단계2와 같이 세 단계로 나누어져 있는데 코어 단계는 가장 기본적이고 널리 지원되는 기능만을 모아놓았다. 따라서 개발자의 응용 프로그램이 여러 데이터베이스를 지원하려면 코어 단계 API만을, 반면 선택된 특정 데이터베이스가 있다면 확장 단계 API까지 사용한다. SQL Server정보는 Data Source Name, Server, Network address 등으로 이루어져 있다. 지식베이스를 저장하기 위해 사용되는 SQL Server 필드는 다음과 같다.

```
<데이터 타입>   <필드명>
  TEXT        FACT_BASE    /* 지식베이스의 Fact 부분을 저장하고 있음 */
  TEXT        RULE_BASE    /* 지식베이스의 Rule 부분을 저장하고 있음 */
  CHAR        KB_DOMAIN    /* 지식베이스의 응용 도메인을 나타냄 */
  INT         KB_ID        /* 지식베이스의 Identity를 나타냄 */
  ...         ...          /* 기타 정보를 저장함 */
```

각각의 응용 도메인에 따라 서로 다르게 구축된 지식베이스를 RULE_BASE와 FACT_BASE라는 TEXT 필드에 저장한다. 이 지식베이스는 각각의 도메인에 대하여 유일하므로 KB_ID라는 필드를 정의하여 RULE_BASE, FACT_BASE를 응용 프로그램과 매치시킨다. 각각의 도메인에 대한 설명은 KB_DOMAIN 필드에 나타난다. 만약 서비스를 제공받고자 하는 컨트롤러가 이미 자신의 도메인에 적합한 지식베이스를 가지고 있다면 OLE 오토메이션 서버에게 지식베이스를 화일 형태로 제공함으로써 가능하다.

4.3 인터페이스

OLE 오토메이션은 서버와 컨트롤러의 두 가지 타입으로 분류된다. 일반적으로 서버라 함은 단위 기능을 가진 특정 서비스를 제공할 수 있는 프로그램을 의미하고, 컨트롤러는 이러한 서비스를 제공받을 수 있는 프로그램을 의미한다. 서버와 컨트롤러 사이에는 인터페이스를 통하여 통신한다. 서버와 컨트롤러 사이의 커뮤니케이션 과정을 도식화하면 <그림 3>과 같다.



그 림 3. OLE 오토메이션 서버와 컨트롤러

5. 결 론

전문가 시스템에 대한 연구는 활발히 진행되어 왔다. 그러나 기존의 전문가 시스템은 최근에 발전된 컴포넌트 기술을 적극 활용하지 못하고, 지식베이스의 저장 매개체로써 DBMS를 충분히 활용하지 못하고 있다. 또한 지능형 시스템에서 필요로 하는 추론 기능을 제공하지 못하고, 이질 DBMS와 호환성 있는 접근을 허용하지 못한다.

본 논문에서는 COM을 따르는 시스템 구조를 갖고 OLE 오토메이션 서버의 기능을 가지는 추론기관을 제시하였다. 제시된 추론기관은 기존 전문가시스템의 약점을 다음과 같이 보완하였다. 첫째, COM을 기반으로 한 구조를 가지므로 이 모델에 따라 개발된 다른 시스템들과의 호환성을 가진다. 둘째, OLE 오토메이션 서버의 기능을 가지므로 OLE 컨트롤러에게 확장 가능한 서비스를 제공한다. 셋째, ODBC 인터페이스를 가지므로 이질적인 DBMS 와 호환성 있는 접근을 허용한다. 넷째, 여러 분야의 지식베이스를 DBMS 에 저장해 둠으로써 특정 도메인에 종속되지 않는다.

본 논문의 의의는 기존의 전문가 시스템에서 사용되던 추론기관을 COM을 기반으로 한 시스템 구조에서 컴포넌트로 구성하고 서비스와 재사용성을 확장한데 있다. 즉, 추론기관에 OLE 오토메이션 서버의 기능을 첨가하고, 이러한 서비스를 제공받을 수 있는 OLE 컨트롤러를 제시하였다.

참 고 문 헌

[1] K. Brockschmidt, *Inside OLE*, 2nd Edition, Microsoft Press, 1995.

[2] C. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Journal of Artificial Intelligence*, Vol. 19, No. 1, 1982, pp. 17-37.

[3] L. Giarratano, *Expert Systems*, PWS Publishing Company, 1994.

[4] I. K. Kim and C. H. Lee, "MIRage: Integration of Heterogeneous Distributed Image Database Systems by Object Request Broker", *Proceedings of SPIE Medical Imaging*, SPIE, 1997.

[5] Microsoft Corporation, *Microsoft ODBC 2.0 Programmers Reference*, Vol. 1, Microsoft Press, 1995.

[6] M. Stonebraker, The Integration of Rule Systems and Database Systems, *IEEE Transaction on Knowledge and Data Engineering*, Vol.4, No.5, October 1992.

[7] Suk I. Yoo et. al, "HEXPERT: An Expert System Building Tool", *Proceedings of the International Tools for Artificial Intelligence '91*, IEEE Computer Society, California, November 10-13, 1991, pp. 510-511.