工學博士 學位論文

CORBA 기반 해운전자상거래 에이전트의 설계 및 구현

Design and Implementation of CORBA-Based Shipping Electronic Commerce Agents

指導敎授 金 是 和

2002年 6月

韓國海洋大學校 大學院

海事輸送科學科

張 日 東

목 차

List of Table

List of Figures

Design and Implementation of CORBA-Based Shipping Electronic Commerce Agents

Il-Dong Jang

Department of Maritime Transportation Science
Graduate School, Korea Maritime University

[Abstract]

Owing to the development of information technology, the internet environment has provided an immense cyber space where the whole world can share beyond limits or boundaries on the earth. The e-market and the electronic commerce has been a common practices in every field of industry lately.

The shipping industry has also experienced some practices of the electronic commerce, however, most of the practices so far have been confined to the online services on the web site providing a limited contents concerning its business. In order to realize the more advanced practices of the shipping electronic commerce, normal activities in shipping such as the negotiation and the contract on purchasing/selling or chartering ships can happen interoperably in the existing heterogeneous network environment with various database systems and different platforms.

This paper aims at studying on the design and implementation issue of Shipping Electronic Commerce Agents, which is named SECA, based on CORBA(Common Object Request Broker Architecture) under client/server computing environment. CORBA is a standard middleware supporting heterogeneous networks and designed as a platform-neutral infrastructure for inter-object communication. So, CORBA is known as a powerful middleware system that can integrate non-interoperable computing. That is why CORBA is adopted in this study.

The study, at first, is motivated by the literature survey on the practices of shipping electronic commerce as well as the research on CORBA-based system design and implementation. Then, the systems analysis for the design and implementation of SECA was carried out by reflecting some practices of shipping. Finally, the prototype CORBA-based SECA, which is consisted of client agent, MS-SQL server agent, trader agent, and interbase server agent, was designed and implemented under client/server computing environment.

The paper reports the design and implementation process of SECA in detail and comments some notable aspects of SECA as well. The author concludes the paper stating the needs of further study on web-based SECA and expecting that the proposed design and implementation approach of SECA would be utilized in shipping electronic commerce in the near future.

용어 정리

COM : Component Object Model

CORBA : Common Object Request Broker Architecture

DCOM : Distributed COM

DII : Dynamic Invocation Interface

GUID : Globally Unique IDentifiers

IDL : Interface Definition Language

IIOP : Internet Inter-ORB Protocol

OMA : Object Management Architecture

ORB : Object Request Broker

OMG : Object Management Group

SECA : Shipping Electronic Commerce Agents

RMI : Remote Method Invocation

RPC : Remote Procedure Call

제 1 장 서 론

1.1 연구의 배경

오늘날 컴퓨터 산업에 있어 분산 컴퓨팅(Distributed Computing)은 시스템 통합관리에 있어 중요한 기술 중 하나이다. 지난 수년간 인터넷의 대중화와 컴퓨터 기술의 발전으로 분산 어플리케이션에 대한 관심이 증대되면서, 특히 분산 컴퓨팅과 객체지향 기술(Object Oriented Technology)의 발전은 IT 전반에 영향을 미치고 있으며, 이 두 가지 기술을 통합하려는 시도로 분산객체 관리 시스템(Distributed Object Management System)이 연구되고 있다.

1989년에는 기존의 객체지향 기술을 바탕으로 응용 프로그램의 객체지향 표준을 제정하기 위해 OMG(Object Management Group)[1]라는 비영리 단체가 탄생하게 되었다. 처음에는 11개 관련단체(DEC, HP, SUN, NCR, IONA, NEC, ANC, ETRI, 3COM, PHILIPS, CANNON, Transarc 등)들이 참가하여 표준화된 분산 기법인 CORBA(Common Object Request Broker Architecture)[2][3][4]를 제시하였지만, 지금은 약 850여개 업체가 참가하고 있다. 지난 몇 년간 CORBA는 여러 분야에서 잘 적용되었고, 프로그램 개발자들에게 다양한 언어로 개발된 컴포넌트들을 사용할 수 있도록 지원한다. CORBA의 핵심은 이질적인 컴퓨터 환경[5][6]에 분산되어 있는 소프트웨어 컴포넌트를 객체화하여, 그 인터페이스를 ORB(Object Request Broker)에 등록함으로써, 클라이언트 프로그램이 원격 컴포넌트 객체에 대해 마치 지역 객체처럼 접근하도록 하는 것이다.

CORBA는 현재의 클라이언트/서버 환경에서 운영되는 분산 컴포넌트들이 서로 다른 주소공간에서 네트워크를 통해 이종의 플랫폼 상에서도 동작되도록 효율적으

로 묶어 주는 인프라이며, 또한 이종의 시스템들 간에 프로그램을 분산시켜 부하를 줄여주고 시스템의 성능 저하와 네트워크 병목현상[7]을 해결해 준다. 더욱이 분산 컴퓨팅환경 하에서 객체지향기술의 도입은 필수적이며, 객체지향 기술은 개인과 조직에게 비용절감과 생산성 향상의 기회를 부여함으로써, 이를 통한 정보통신의 경쟁력을 제고하여 준다[8].

분산처리기술과 더불어, 네트워크 연결 및 관리에 대한 투명성, 다양한 시스템 호환성, 언어 독립성, 객체지향기술 등의 장점을 가진 미들웨어[9]로서의 CORBA는 본 연구에서 요구하는 속성을 만족시키고 있으며, 다양한 플랫폼에 존재하는 데이터베이스들에 접근할 수 있게 해준다[10]. 여기서 미들웨어로 CORBA를 선택한 이유는, CORBA가 DCOM(Distributed Component Object Model)[11] 보다 독립성과 다양한 플랫폼의 지원 등 몇 가지 장점을 가지고 있기 때문이다. CORBA는 오늘날 사용되고 있는 모든 주요 운영체제에서 사용이 가능하며, C++, Ada, COBOL, Smalltalk, JAVA, Object Pascal 등 다양한 프로그래밍 언어로 구현된 시스템들간에도 연동이 가능하다[12].

최근 10년간 기하 급수적으로 성장한 인터넷 환경은 가상공간(Cyber Space)을 형성하여, 전 세계를 시간과 공간을 초월한 단일 세계로 묶어주고 있다. 해운시장에 있어서, 전통적 상거래 행위와 구별되는 전자상거래 시스템의 도입은 거래성사기간의 단축, 거래 비용의 절감 등을 위한 새로운 가능성을 제시한다. 그러나 현재의 해운전자상거래에서는 가상공간에서 단지 사진이나, 텍스트 위주의 컨텐츠를 제공할 뿐이다. 또한 현재의 해운전자상거래 시스템들은 다양한 플랫폼 상에서 서로 다른 데이터베이스 시스템들을 가지고 있어서, 데이터베이스의 공유나 효율적인 정보통신에 어려움이 많다. 이 연구는 데이터 공유, 통신 투명성 및 상호운용성을 지원하는 CORBA를 기반으로 선박매매 및 용선 등을 중개하는 해운전자상거래 에이

전트인 SECA(Shipping Electronic Commerce Agents : 이하 SECA)의 설계 및 구현을 다룬다.

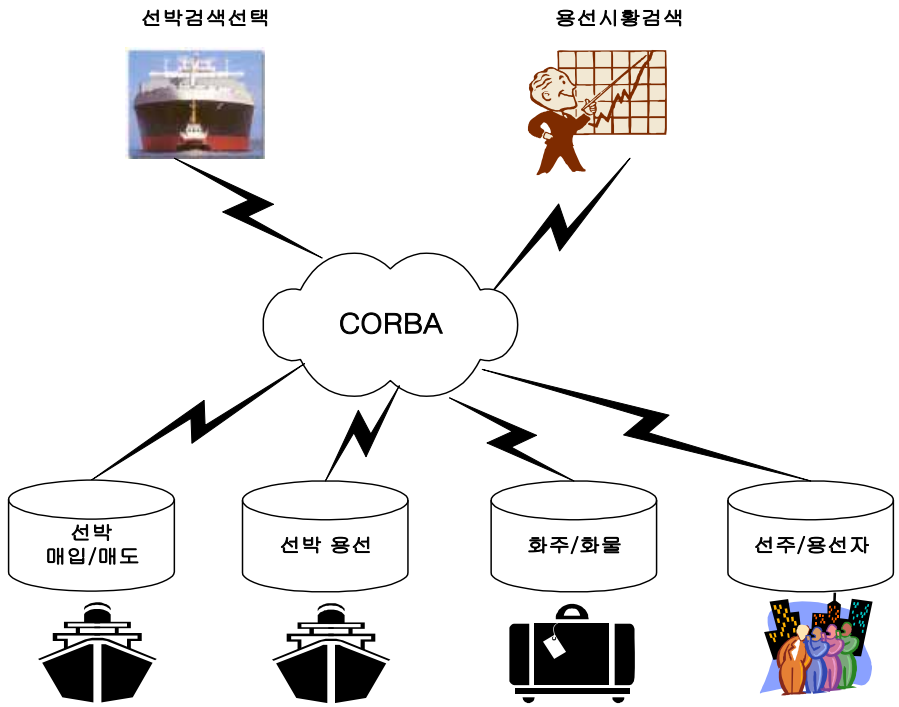

Fig. 1.1 Conceptual Diagram of Shipping Electronic Commerce Agent

Fig. 1.1은 서로 다른 데이터베이스관리시스템에서 관리되는 복수의 데이터베이스가 서로 독립적으로 구축되어 있을 때, 사용자가 CORBA 기반의 SECA를 통하여 필요한 정보를 얻는 과정을 개념적으로 보여준다. Fig. 1.1에서 알 수 있듯이 선박 매매 및 용선 거래의 대상이 되는 선박을 검색하려면 선박 매입/매도 데이터베이스 및 선박용선 데이터베이스에 접근할 수 있어야 하며, 용선시황을 검색하기 위해서는 선박용선 데이터베이스 뿐만 아니라 화주/화물 및 선주/용선자 데이터베이스도 접근할 수 있어야 한다. CORBA 기반의 SECA는 이러한 요구에 맞게 구현된 메소드를 통해, 다양한 플랫폼상의 데이터베이스에 투명하게 접근할 수 있게 해준다.

1.2 연구의 범위와 목적

인터넷의 발달과 더불어 네트워크 기술의 발전은 분산 컴퓨팅환경으로의 변화를 초래하였고 "정보로부터 독립된 섬"에서 벗어나 정보 및 데이터의 공유와 통합을 촉진하였다. 실제로 관리 응용 프로그램에서 인트라넷을 사용하는 기업의 사례는 Fortune지 선정 1000개 기업의 2/3 정도라고 보고되고 있으며[13], 또한 Forrester Research Inc.의 조사결과에 따르면 미국내 주요 50개 기업 중 관리 응용 프로그램에 인트라넷을 사용하는 기업의 사례는 현재 16% 정도가 개발 중이거나 개발할 계획을 갖고 있다[14].

오늘날의 정보화 사회의 기업 환경은 기업의 생존과 경쟁력 강화를 위해 정보 시스템의 효율적인 구축과 함께 기업의 조직구조를 근본적으로 재구성하지 않으면 안되는 이른 바 업무재설계(Business Process Reengineering)를 요구하고 있다. 또한 인터넷 사용자의 증가로 가상공간에서 상품의 정보와 다양한 상거래 서비스를 제공하려는 인터넷 비즈니스 업체가 기하급수적으로 늘어나고 있으며 그 응용 분야가 확대되고 있다[15][16].

최근의 정보 시스템은 CORBA[17], JAVA[18][19], WWW 등을 통합한 오브젝트 웹[20] 기반의 분산객체[21] 컴퓨팅환경 하에서 동작하며 정보의 공유 및 분배는 실시간으로 이루어진다. 분산 컴퓨팅 인프라의 구축으로 방대한 자료의 획득은 손쉬워졌지만, 원하는 정보의 획득은 용이한 문제가 아니다. 이러한 문제 해결을 위해 1990년대 중반에 들어서면서 정보의 홍수 속에서 원하는 정보를 신속하게 얻기 위한 에이전트의 연구가 시작되었다. 해운항만분야에서도 해운전자상거래의 활성화를 위해 거래당사자들의 상거래 비용 및 시간을 절감하고 거래 정보 및 데이터의 공유와 통합을 쉽게 구현할 수 있는 에이전트의 필요성이 증대되었다. 이러한 에이

전트는 초고속 정보통신망에서의 신속한 정보 검색을 가능하게 할 뿐 아니라 온라인 쇼핑 등의 전자상거래나 메시징과 같은 이동 컴퓨팅 분야에서도 활용된다[22].

한편, 세계경제질서의 재편성과정에서 1995년에 세계무역기구(World Trade Organization: WTO)가 출범하였고, 우리나라는 1999년에 경제협력개발기구(Organization for Economic Cooperation and Develement: OECD)의 회원국이 됨에 따라 국내 해운시장도 개방하지 않을 수 없게 되었다. 그 결과 보호주의적인 성격을 가지는 국적선 우선 정책이었던 국적선 보유장려제도나 지정화물제도가 폐지되어 한국 해운업은 더 이상 보호주의 그늘에 안주할 수 없게 되었으며, 그 어느 때 보다도 치열한 국제적인 경쟁 환경에 직면하고 있다. 실제로 세계적인 해운선사, 선박중개업자 및 항만운영관리업자들이 사이버 시대의 경쟁력 제고를 위해 전자상거래 네트워크 구축에 총력을 기울이고 있다. 만약 해운항만 관련기업들이 가까운 장래에 해운전자상거래의 보편화 조류에 대응하지 못한다면 21세기 무한경쟁에서 도태될 수 밖에 없을 것이다.

현재 해운기업의 전자상거래는 대부분 자사의 웹사이트를 기반으로 이루어지고 있는데, 주로 선박운항일정 제공, 화물선적의뢰, 화물부킹, 화물선적, 화물추적, 자사 홍보 등의 컨텐츠를 제공하고 있다. 기존의 해운전자상거래 환경 하에서는 단순한 키워드나 카테고리로 거래 대상을 검색하므로, 계속적인 검색 작업과 반복 확인 작업이 필요하며 이러한 이유로 신속하고 정확한 선박매매 및 용선 정보를 얻기가 어렵다. 따라서 인터넷 비즈니스가 거대한 조류로 밀려오고 있는 현실을 감안할 때 해운상거래에서도 전자상거래 시스템의 도입이 필요하다. 또한 해운항만 및 물류기업간의 정보 공유를 원활히 할 수 있도록 공동 네트워크를 구성하는 방안도 강구할 필요가 있으며, 인터넷 접속 및 거래에 소요되는 시간을 단축시켜 고객의 이용 편의성을 높여야 할 것이다[23].

따라서 이 연구에서는 선박용선 및 매매를 가능하게 하는 CORBA 기반의 해운 전자상거래 에이전트인 SECA를 분석, 설계 및 구현하고자 한다. SECA는 선박중개인을 대신하여 이동 에이전트 프로그램이 네트워크를 스스로 순회하면서 용선 또는 매매를 원하는 선박정보를 검색하여 중개해주므로 사용자가 원하는 자료의 검색을 보다 신속히 처리할 수 있고, 용선 및 매매 계약담당 인력을 대체하는 효과가 있다. 시스템 구현환경은 메인프레임 환경이 아닌 클라이언트/서버 환경으로 하여 중앙 집중식 시스템들이 가지는 서버 부하 문제 및 네트워크 트래픽 문제를 동시에 해결하도록 하였다. 이 연구의 목적은 갈수록 경쟁이 심화되고 있는 국제해운 환경 속에서 우리 나라 해운기업이 환경변화에 능동적으로 대처하면서, 해운전자상거래가 일반화 될 가까운 미래를 대비할 수 있는 해운전자상거래 기반기술 확보에 기여하는 데 있다.

1.3 논문의 내용 및 구성

이 논문의 내용은 CORBA 기반의 해운전자상거래 에이전트를 설계하여 구현하는 것을 다루고 있으며 전부 6장으로 구성되어 있다.

먼저, 제1장 서론에서는 이 연구의 배경, 연구의 범위와 목적을 서술한다.

제2장에서는 해운전자상거래의 개요와 선박의 가격결정 및 용선에 관하여 다루며, 특히 해운전자상거래에 있어 중개인의 역할에 대한 내용을 분석하였다.

제3장에서는 클라이언트/서버 하에서의 CORBA 통신과정을 설명하며 특히 CORBA의 구성요소 및 호출 원리를 중점적으로 설명한다. 또한 클라이언트 인터페이스 정의언어(IDL : Interface Definition Language)를 통해 서비스를 제공받을 수 있음을 보인다.

제4장에서는 SECA를 구현하게 된 배경과, 구현을 위한 요구 사항 및 기능을 분석하고, 구현된 SECA의 특징, 개요, 설계 방법론, 적용 방안 및 사용자 인터페이스를 설명한다. 또한 선박검색과 메시지 전송 방법론을 제안하였으며 SECA에서의 운용 예를 보인다.

제5장에서는 클라이언트 에이전트(Client Agent), MS-SQL 서버 에이전트(MS-SQL Server Agent), 트레이드 에이전트(Trader Agent), 인터베이스 서버 에이전트(InterBase Server Agent)로 구성된 SECA의 동작과정을 설명한다.

마지막으로, 제6장 결론에서는 인터넷을 통한 선박의 매매 및 용선 거래계약 건의 사례가 매우 드문 것이 현실임을 지적하고, 이 연구가 해운전자상거래의 실무에 작으나마 기여할 수 있기를 기대하는 언급으로 논문을 맺는다.

제 2 장 해운전자상거래의 현황

먼저 해운전자상거래의 현황을 이해하기 전에 '전자상거래'를 간단히 정의하고자 한다. 전자상거래(Electronic Commerce)는 1989년 미국 국립 로렌스 리버모아(Lawrence Livermore National Laboratory)에 의해서 미국 국방성의 프로젝트를 수행하는 과정에서 처음 사용된 후, 1990년대 인터넷과 WWW(World Wide Web)의 출현으로 본격적인 출발을 시작하였다.

1996년 유엔국제전자상거래법위원회(UNCITRAL : United Nations Commission on Interational Trader Law)의 전자상거래모델법에서는 전자상거래란 "전자문서교환(EDI), 전자메일, 텔렉스, 팩시밀리를 포함하여 다양한 전자적인 메시지 형태의 정보에 의한 상업적 활동"이라고 정의하고 있다. 즉, 전자상거래란 사람과 사람이 물리적인 매체의 전달을 통해 상품을 사고 파는 전통적인 상거래와는 달리 컴퓨터와 네트워크라는 전자적인 매체를 통해 상품을 사고 파는 행위를 통칭하며, 광의로는 이러한 협의의 의미는 물론 기업내 또는 기업과 기업간 거래관계의 모든 프로세스를 전자적으로 처리하는 것으로 정의할 수 있다[24]. 현재 전자상거래 형태로 기업간 거래(B2B)가 전체 규모의 약 80%를 차지하고[25] 있지만 점차 기업과 소비자(B2C)로 이동하고 있으며 해마다 갈수록 높은 비율로 증가하고 있다[26].

전자상거래 동향으로 미국은 전자거래지원센터(ECRC : Electronic Commerce Resource Center)가 중소기업의 인력에 대한 교육과 기술지원을 통해 전자상거래 마인드를 확산시켰으며, 민간부문에서는 ISG(Industry Steering Group)를 통해 업체간 CALS 추진업무를 조정하였으며, 우리 나라도 ECRC를 통한 전문인력 양성과 기술지도 및 컨설팅을 지원하고 특히 산자부·정통부 등의 정부부문 시범사업추진, 전자상거래 기술모델 개발, 실증사업의 개발 및 추진을 지원하고 있다.

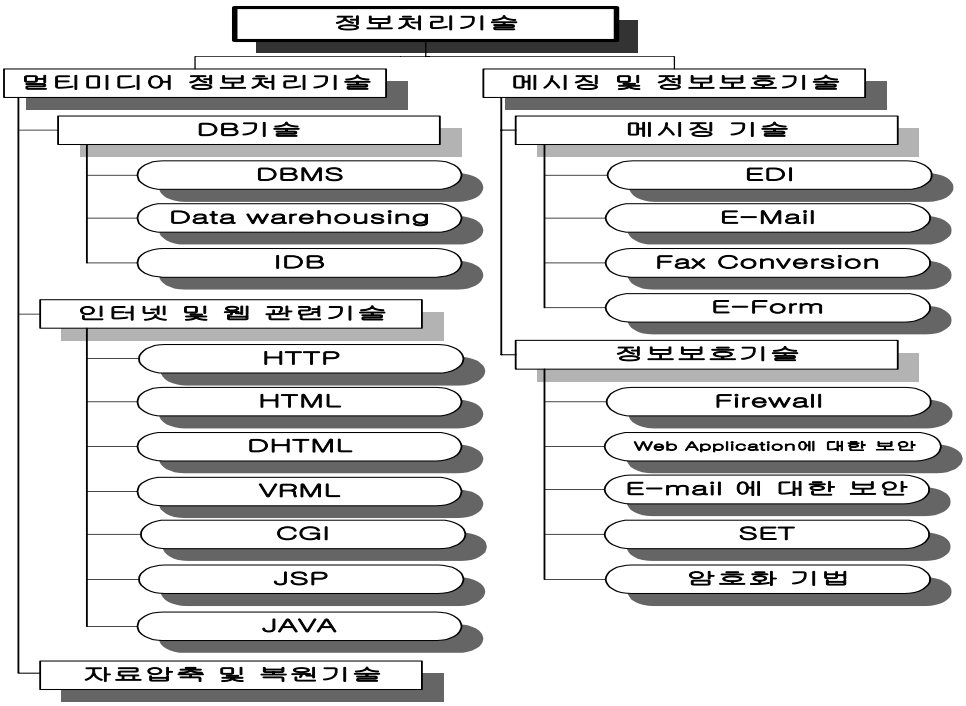

Fig. 2.1 Schematic Diagram of Information Processing Technology on EC

Fig. 2.1에서와 같이 정보처리기술의 발달로 인터넷과 웹 관련, 보안기술의 발달에 힘입어 전자상거래가 더욱 활성화 될 것으로 보이며, 기존에 오프라인에서 기업들도 경쟁적으로 온라인으로 활동범위를 넓히는데 주력함으로써 전자상거래는 새로운 비즈니스 형태로 자리매김을 하고 있다. 세계경제는 지금 정보기술과 새로운 미디어인 인터넷의 발달에 힘입어 전자상거래 거래양식이 증가할 뿐 아니라 국내외에서 가상공간을 기반으로 하는 수많은 웹사이트들이 다양한 경제활동도 다양화됨에 따라 분산 컴퓨팅, 컴포넌트화가 서서히 주류를 이룰 것이다[27].

최근 해운항만분야에 수많은 웹사이트가 개설되고 경쟁적으로 고객유치와 전자상거래에 나서는 것에 대해 업계의 평가는 상반되고 있는데, 하나는 이들 웹사이트의 성공 가능성을 회의적으로 보는 시각이고, 다른 하나는 이러한 사이트가 크게

도움이 된다는 입장이다. 회의적인 시각은 인터넷을 기반으로 하는 선복 경매 및 화물예약 등을 관습이나 기존 거래 형태를 고수하며 전자결제에 대한 불안과 참여 업체가 많지 않다는 이유로 뿐만 아니라 해운전자상거래를 도입한 해운업체들의 상당수는 아직도 제대로 된 수익성 모델을 갖추지 못하기 때문에 성공하기 어려울 것이라는 입장이다. 그러나 전자상거래를 긍정적으로 보는 입장의 견해는 다르다. 전자상거래는 투명성을 제고시키고, 효과적 고객 서비스 기능에 있어 운임, 비용, 운항일정, 화물추적 등 실시간 제공에 따른 인력대체효과 등 새로운 틈새시장의 개척과 각종 비용절감 효과로 향후 더욱 발전하게 될 것이라 본다.

우리 나라 해운·항만 물류업체들의 경우 대부분 인터넷 홈페이지 홍보 기능에 많은 기능을 두고 있다. 따라서 현재의 인터넷 홈페이지의 개념을 홍보 기능보다는 전자상거래의 기능에 초점을 맞추어 운영할 필요가 있으며, 해운 및 물류업체간 정보 공유를 원활히 할 수 있도록 공동 네트워크를 구성하는 방안도 강구할 필요가 있다.

2.1 전자상거래 에이전트

1950년 튜링(Alan Turing)의 논문에서 제기된 "기계는 생각할 수 있는가"라는 질문은 인터넷과 웹을 중심으로 한 컴퓨터 환경의 재편에 따라 '지능형 에이전트(Intelligent Agent)를 어떻게 구현할 것이냐'의 문제로 옮아갔다.

사람마다 에이전트를 보는 시각이 다르고 연구 방향이 다르기 때문에 에이전트에 대한 정의는 매우 다양하지만, 한마디로 종합하여 보면 에이전트란 사용자를 대신하여 주어진 일을 정확하게 수행할 수 있는 하드웨어나 소프트웨어에 기반한 컴퓨터 시스템이라고 정의할 수 있겠다[28]. 즉, 사람을 대신하여 업무를 수행하기 위해서는 사람과 같은 지능을 가지고 있을 필요가 있으므로, 소프트웨어 에이전트는 인공지능 분야의 하나로 연구되고 있으며, 인터넷의 발전과 함께 인터넷에서 활동하는 소프트웨어 에이전트에 관한 연구개발이 활발히 진행되고 있다[29].

또한, 에이전트 시스템은 네트워크를 통한 이동성 유무에 따라 이동형 에이전트와 정적 에이전트로 분류되기도 하지만 협력 인터페이스 에이전트와 협력학습형 에이전트 등으로 분류될 수 있다. 에이전트 시스템을 구조적 관점에서 숙고형 에이전트(Deliberative Agent), 반응형 에이전트(Reactive Agent), 복합형 에이전트(Hybrid Agent)로 분류되기도 한다[30][31][32]. 에이전트는 지능을 갖춘 객체로서 보다 독립적이면서도 상호 협동이 용이한 시스템을 구축하는데 적합한 개념이다.

이 장에서는 에이전트에 관한 개념 수립을 위하여 기본적인 이론을 살펴보면 다음과 같이 구분된다. 소프트웨어 에이전트는 분산인공지능(Distributed Artificial Intelligent)의 한 분야인 멀티에이전트 시스템(Multi-Agent System)으로부터 발전하였다[33][34].

분산인공지능은 세 가지 분야로 나눌 수 있는데 멀티에이전트 시스템, 분산문제

해결(Distributed Problem Solving) 그리고 병렬인공지능(Parallel AI)분야가 이에 속한다. 따라서 소프트웨어 에이전트는 분산 컴퓨팅이 가지는 모듈성과 병렬성으로 인한 속도 향상, 중복에서 오는 신뢰성과 같은 장점 뿐 아니라 인공지능의 특징인 지식을 기반으로 하는 동작을 수행하고 유지보수가 쉬우며, 재사용성, 플랫폼 독립성 등을 포함하고 있다. 에이전트는 독립적이며 특정한 일을 하도록 만들어졌지만 자신의 임무를 수행하면서 동시에 다른 에이전트와의 협력을 위한 통신 기능도 갖추고 있다.

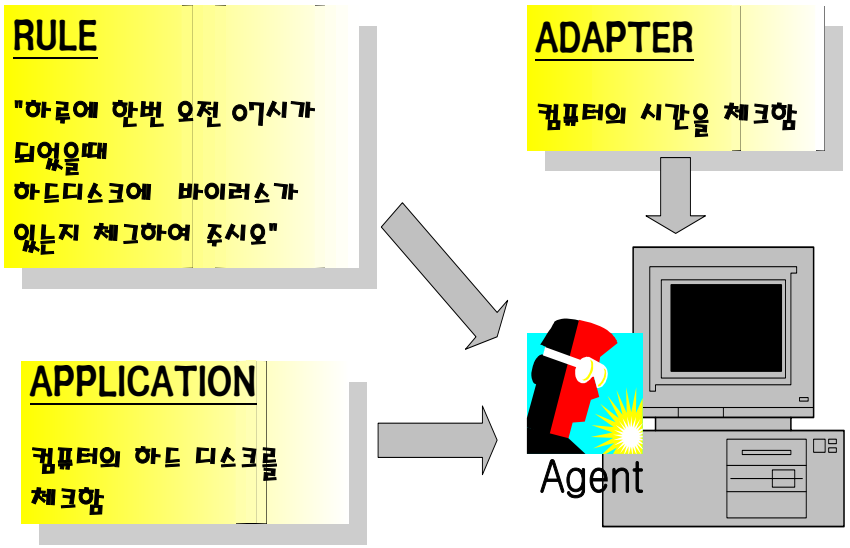

Fig. 2.2 Example of Stand-Alone Agent

예를 들어보면 사용자가 아침 7시에 컴퓨터를 사용하지 않을 경우, 하드디스크에 바이러스 감염여부를 체크하여 주는 프로그램도 일종의 혼자 일을 수행하는 독립형 타입의 에이전트 프로그램으로 볼 수 있을 것이다. (Fig. 2.2)

현재 에이전트에 관한 연구는 크게 두 가지 방향으로 연구되고 있다.

첫째는 1977년 이후의 경향으로 개별적인 에이전트의 행동보다는 에이전트 사이의 통신과 협력과 협상을 통한 이해 충돌 해결, 수행할 작업의 해체와 분산 등에 관한 연구이다. 에이전트의 목표는 다수의 협력 에이전트로 구성된 시스템을 분석하고, 설계하고, 통합하여 독립적인 에이전트에 관한 연구로 에이전트의 이론과 아키텍처, 언어에 대한 연구이다.

둘째는 1990년 이후 에이전트의 유형별 분류에 대한 보다 포괄적인 연구로써 여기서는 에이전트 기술을 이용한 여러 응용 분야에 대한 연구 부분이다. 이는 사람의 지능을 반영하는 행동을 수행하는 응용 수준의 소프트웨어 객체로서 시스템과 사람 사이에서 중간자적인 역할을 수행하면서 복잡하고 다양하며 방대한 양의 서비스를 수행한다.

2.1.1 기존의 연구

현재 전자상점에 적용된 제품추천은 1990년대 초반 미국의 돈 페퍼스와 마사로저스에 의해 처음으로 개념이 확립된 일대일 마케팅으로써 관련 상품의 일괄구매를 유도하는 크로스셀링(Cross selling), 동종이급 상품을 제안하는 업셀링(Up selling), 그리고 타겟셀링(Target selling)등으로 나눌 수 있다[35][36].

・동시크로스셀링 : 먼저 고객이 제품을 장바구니에 넣는 경우 거래분류기에서 해당 제품과 관련된 대표상품명을 검색하여 고객에게 제공.
・순차크로스셀링 : 일단 고객이 상품을 구매한 후 다시 전자상점에 로그인 했을 때 고객의 거래내역을 바탕으로 시간에 순차적으로 관련 상품을 제공.
・업셀링 : 순차크로스셀링과는 달리 업셀링에서는 고객이 구매 후 일정 시간이 지나면 동종이급의 상품을 제안하는 것으로 시간 항목이 고려된다.

이러한 고객의 신상명세와 구매행동을 바탕으로 개별화된 판매기법을 통한 고객맞춤 서비스를 제공하여 고객의 만족을 증진시키며 지속적인 구매를 유도한다.

한편, Amazon과 같은 전자상점은 고객의 구매패턴과 고객 정보를 기반으로 프로파일을 구축한 뒤, 고객이 웹사이트에 접속할 때 적절한 서적을 추적해 주는 에이전트 기법을 적용한 서비스를 제공하고 있다[37].

지능형 에이전트의 역할에 따라 분류하면 다음과 같다[38].

1) 감시자 에이전트

특정 정보나 사건을 자동적으로 찾아주는 에이전트로 현재 Fishwrap[39]은 1993년 가을부터 미국의 MIT공대의 한 연구실에서 Fishwrap이라는 전자신문의 원형(Prototype)을 제작하여 사용하였으며, 그 외 Newshound, Personal Journal,

Personal View[40], Job Center[41] 등이 있다.

2) 학습 에이전트

학습 에이전트는 사용자의 과거 기호를 학습하고 있다가 그 정보를 바탕으로 사용자 개인의 취향에 맞게 일을 처리하는 능력을 가지고 있다. Firefly[42]는 원래 MIT 연구실에서 개발한 HOMR이라고 불리는 가정용 음악서비스였으며 다른 사용자와 대화하는 능력이 있고, 사용자가 좋아할 만한 음악을 추천할 수 있는 개인적인 소프트웨어 에이전트이다. 그 외 Similarities Engine[43], Web Hunter, Eyes, Open Sesame[44], LifeStyle Finder[45] 등이 있다.

3) 정보검색 에이전트

정보검색 에이전트는 지능적인 방법으로 정보를 조사하는 능력이 있다. 이는 탐색기준에 포함되는 키워드가 실제 찾고자 하는 문서에 포함되어 있지 않더라도 문서를 찾을 수 있는 능력을 가지고 있다. 이러한 능력에 비추어 볼 때 이들은 앞에서 에이전트 성질에서 언급한 지능형 에이전트로 분류할 수 있지만[46] 자율성이 두드러지지 않는다. Applesearch, Pagekeeper, Peregrine[47], Homework helper[48] 등이 있다.

4) 쇼핑 에이전트

판매자와 제품가격을 비교하여 제공해 주는 에이전트이다. 전자상거래가 활성화되면서 쇼핑에이전트는 더욱 활발해질 것으로 보인다.

Bargain Finder[49], Good Stuff Cheap[50], FIDO : The Shopping Doggie[51] 등이 있다.

5) 도우미 에이전트

도우미 에이전트는 임무를 자동으로 수행한다. 한 예로 전자우편함이 꽉 차있기 때문에 더 이상 편지를 수신할 수 없다고 사용자에게 알려줄 수도 있으며 진단하고 치료하는 것이다. Track[52]과 네트워크 컴퓨터 회사로부터 Windows NT에 관한 정보를 제공하는 에이전트 등이 있다.

R. H Guttman과 A. G Moukas등은 소비자 구매행동 단계에 따라 지능형 에이전트가 어떻게 적용되는가를 설명하였다.

Fig. 2.3 Process of Consumer Purchasing

Fig. 2.3은 소비자 구매행위단계로써 현재 요구파악 에이전트, 상품탐색 에이전트, 판매자 탐색단계 에이전트까지 개발되었고 협상단계 에이전트는 개발 중에 있음을 나타낸다.

2.1.2 에이전트의 성질

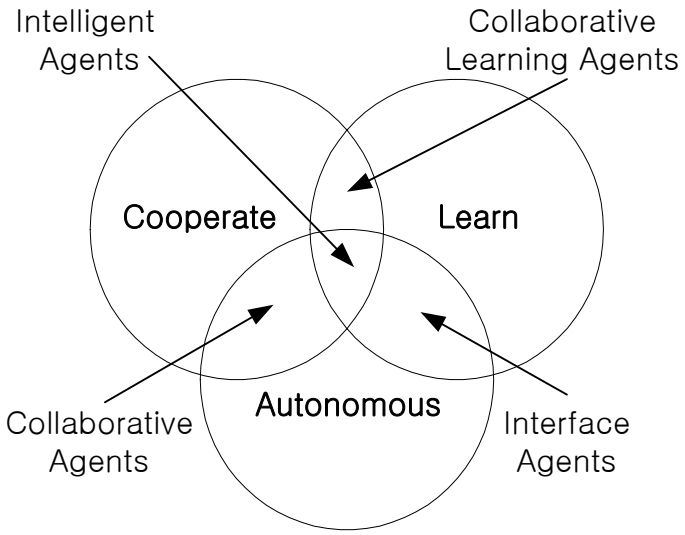

Fig. 2.4 A Typical Example of Agents

일반적인 에이전트의 성질은 다음과 같다[53][54]. (Fig. 2.4)

1) 자율성(Autonomy) : 사람이나 다른 에이전트의 지시가 없어도 스스로 동작할 수 있으며 자신의 행동이나 내부의 상태에 대한 제어능력을 가지고 있다[55]. 자율성은 사용자로 하여금 상위단계 목적에 집중을 하게 하고 그 목적을 달성하기 위한 세부 절차 등은 에이전트가 맡게 된다[56].

2) 사회성(Social ability) : 에이전트 통신 언어를 사용하여 사람이나 다른 에이전트와의 의사소통이 가능하다. 즉, 하나의 에이전트로는 처리하지 못하는 작업의 수행을 위해 다른 에이전트의 도움을 필요로 할 때, 에이전트간의 메시지 교환에 의존하게 되며 통신 언어를 이용한 에이전트간 통신은 메시지 전달이나 공유 메모리방법을 이용할 수 있고, 다른 에이전트 메소드를 불러 수행하기도 한다[57].

3) 지능성(Intelligence) : 지능은 지식베이스와 추론 능력을 갖추고 사용자의 의도를 파악하여 계획을 세우고 학습을 통하여 새로운 지식을 스스로 터득하는 성질로 인공지능에서 많이 연구된 결과를 근거로 하며, 지능은 사실상 자율성과 밀접한 관계를 가지게 되며 지능을 바탕으로 에이전트는 같은 작업이라도 계획과 경험을 통해 더 나은 효과를 기대할 수 있겠다.

4) 이동성(Mobility)[58] : 이동성은 사용자가 요구한 작업을 현재의 호스트로 이동시켜 수행함으로써 수행의 효율을 높이고 네트워크 부하를 줄이는 효과를 가져오며, 이동성은 기존의 클라이언트/서버의 개념과는 다른 개념으로 서버의 내용을 클라이언트를 통해 전송 받아 정보를 얻거나 작업수행을 하는 것이 아니고 클라이언트가 필요로 하는 작업을 위해 에이전트를 서버로 보내어 수행시킨다.

이외에도 부가적인 성질외에도 에이전트가 가지는 성질은 여러 가지가 있는데, 그 중에서 환경 변화에 대하여 반응할 수 있는 반응성(Reactivity), 틀린 정보를 주고받지 않은 정직성(Veracity), 그리고 반드시 목적을 달성하는 방향으로 작업을 수행한다는 이성적 행동(Rationality)등이 있다. 이는 컴퓨터 시스템이 복잡해지면서 보다 추상적인 방법으로 동작을 기술할 필요가 생기면서 등장한 것으로 믿음, 소망, 이성을 가지는 개체로서의 BDI(Beliefs Desire and Intentions) 에이전트는 그 수행 결과를 예측할 수가 있다[59].

에이전트의 응용 분야는 작업흐름 관리, 네트워크 관리, 항공관제시스템, 정보 탐색, 전자상거래, 교육, 전자우편, 전자도서관, 지능형 데이터베이스, 어플리케이션 일정표 관리 등 사용자의 요구가 다양해지고 복잡해짐에 따라 보다 쉬운 시스템 통합과 관리에 있어 사람을 대신할 활용도가 높은 지능형 에이전트를 요구하였다.

2.2 해운전자상거래 개요

오늘날 정보통신 기술이 발달하고 인터넷이 급격히 증가함에 따라 해운산업에 있어 전자상거래로 통한 그 이용이 증가되고 있다.

해운전자상거래(Shipping Electronic Commerce)는 "해운산업과 관련된 모든 사업영역에서 컴퓨터를 매개로 하여 디지털 신호에 의한 정보거래로 이루어지는 상업적인 활동"이라고 정의할 수 있으며 해운산업에는 정기선, 부정기선, 조선, 여객선사업, 선박관리업, 선원관리 및 선용품의 조달, 크루즈, 해운관련 IT 산업, 항만, 복합운송주선업 등 다양한 부문에 적용시킬 수 있다.

현재 운영중인 대표적 해운전자상거래 사이트는 Table 2.1에 나타난 바와 같이 선·화주를 연결하는 화물매칭사이트와 선복거래 사이트, 온라인 운임견적 사이트, 해상보험 및 선용품 등을 판매하는 사이트, 벌크 및 용대선거래 사이트, 해운관련 정보제공 사이트 등이 있다.

Table 2.1 List of Major Shipping Electronic Commerce Sites (출처 : [60])

웹사이트 주소	주요 내용	투 자 사
Asiaship.com	용선, 선박매매, 해운시황	한진, P&O Nedlloyd, RCL 등
e-Janworld.co.jp	용선, 선박매매	NYK, 자스보, 야마미쯔 등
LevelSeas.com	살물선 매매 및 용선	BP Amoco, Clarkson, Cargill 등
Marine-Net.com	용선, 선박매매, 화물중개	MOL, K-Line, 히타치, 후지쯔 등
MarineProvider.com	벙커중개, 선용품 매매	Fearnleys, Euronav, OMI 등
MaritimeDirect.com	선박매매 및 해운종합정보	Santon Capital, Bill Livanos 등
OceanConnect.com	벙커중개, 보험	BP, Shell, Clarkson, Eletson 등
OneSea.com	용선, 기자재, 해운정보	Lief Hoegh, Bergesen 등
Seasupplier.com	유조선 용선	OMI

많은 해운관련 사이트 중에서 누구나 포탈(Portal)이라는 용어를 사용하는데 주저함이 없는 사이트는 Asiaship.com일 것이다. Asiaship.com은 해운 및 항만관련 산업의 광범위한 B2B 인터넷 포탈서비스를 제공하고 있다. 해운전자상거래는 거래당사자들의 대부분이 기업들이므로 B2B라 할 수 있으며 다수의 판매자와 구매자들간의 거래를 중개위주의 구조를 가진 전자상거래라고 볼 수 있다[61]. 해운전자상거래를 사용하고자 하는 목적은 물류정보의 데이터처리, 서류의 단순화, 고객서비스의 향상과 더불어 육상, 해상, 항공 등 개별 화물 정보망과 통관, 무역, 금융 등 유관망의 상호연계로 물류관련 서비스를 일괄적으로 제공하여 물류비 절감과 물류 효율화를 통한 경쟁력 향상을 하고자 함이다.

해운전자상거래의 특징을 요약하면 다음과 같다.

첫째, 기업간 거래이다.

기업간 전자상거래로 구매자와 판매자가 모두 기업인 경우이다. 또한 기업의 활동 중 판매, 금융, 물류, 무역 등의 기업간의 업무를 인터넷 기술을 통해 지원하는 것을 의미하며 해운전자상거래에 있어 전형적인 형태이다. 특히 여기서 다룰 선박은 일반인이 생각하는 단순한 물건이 아닌 크기며, 가격면에서 볼 때 기업을 인수하는 가격에 맞먹는 것이기 때문에 선박 각 부문에 전문가가 필요하다. 따라서 선박을 인수하거나 용선을 의뢰하는 경우는 전체적인 사항을 꼼꼼히 점검해야 한다.

둘째, 수직적 시장이다.

해운전자상거래는 '해운산업'이라는 특정산업과 시장을 대상으로 하고, 다수의 거래자들이 참여할 수 있는 '인터넷'이라는 가상의 공간을 제공하므로, 수직적 가상시장이다.

셋째, 와해성 기술혁신을 가진다.

와해성 기술혁신이란 시장의 경쟁기반을 변화시켜 시장의 판도를 재편시키는 혁신을 통칭하며, 이러한 와해성 기술혁신은 반드시 첨단 기술에 기반하고 있어야 하는 것은 아니다[62]. 오히려 상당수의 와해성 혁신은 기존 기술에 비해 성능이 월등하지도 않고 가격적인 이점이 있는 것도 아니었다. 그러나 와해성 기술혁신이 기존 기술이나 시장을 잠식해 나가는 원인은 와해성 기술이 기존의 지배적 기술과는 매우 다른 속성을 지니고 있기 때문이며 특히, 해운전자상거래는 해운산업에서 와해성 기술혁신이 가능하게 하는 원동력인 것이다. 전통적으로 대부분의 정기선사들은 화주가 의뢰한 컨테이너 선적에 대한 운임율을 결정하기 위해 많은 전화통화, 팩스, 고객정보 갱신, 그리고 다수의 내부적 협상 등과 같은 복잡한 업무프로세스를 가진다. 즉, 선적에 대한 운임율을 인터넷 상에서 경매를 통해서 결정한다면, 이는 기존 정기선 해운산업의 업무흐름을 완전히 와해시키는 것이다[63].

이제 해운전자상거래는 각 부문별 사업에 특화·전문화된 서비스를 제공하는 것에서부터 거의 모든 부문들을 종합적으로 서비스를 제공하는 해운포탈 서비스로 발전하고 있다. 일반적으로 해운전자상거래를 제공하는 서비스의 형태에 따라 정보제공형, 고객유치형, 포탈형으로 나눈다. 정보제공형은 온라인 신규진출업체의 홍보형, 컨설팅업체 기업의 소개, 뉴스 제공업체의 특징을 가지고 있다. 고객유치형은 전자상거래 주력업체, 선복, 운임, 유류, 선용품 등 온라인 판매 등을 하고 있다. 포탈형은 대형업체가 주로 구축하며 원스탭 서비스를 표방하며 기존 협력체제 구축을 하고 있지만 아직까지는 실제거래나 수익은 대체로 미흡한 실정이다[64].

2.2.1 해운시장

"차세기에는 해운중개인(브로커)이 필요 없는 시대가 올 것이다". 더욱이 새로운 인터넷 용선시스템(New Internet Chartering System)의 출현으로 전통적인 용선브로커의 역할이 크게 위축될 것으로 전망되며, 용선시스템을 통해 선박의 가동률을 향상시키고, 선주와 용선자 모두에게 비용절감과 시장효율을 증대시키는 효과를 가져다 줄 것으로 기대된다[65].

차세대 인터넷 기술을 이용하여 슈퍼 컴퓨터, 대형 데이터베이스, 첨단 가시화 장비와 과학장비를 통합하여 네트워크로 연결된 가상의 슈퍼컴퓨터 혹은 메타 컴퓨터를 형성할 것으로 기대하며[66], 인터넷 기술의 발전은 해운시장 환경 변화의 바람을 몰고 왔으며, 해운기업은 시대적 흐름에 대한 변화에 적응하지 못한다면 결국 도태되고 말 것이다. 따라서 해운기업은 해상물동량 및 항로 패턴의 변화, 이에 의한 운임시황의 변화, 선박 기술 혁신의 진전과 이에 의한 시장구조의 변화, 편의치적선(Flags of Convenience), 개발도상국 등 선박의 진출에 의한 경쟁구조의 변화에 신속하게 대처해야 할 것이다.

한편, 250년 역사의 Baltic Exchange는 세계 해운거래시장 중 탱커시장의 50%, 건화물시장의 30~40%를 점하고 있는 전통적인 해운거래소이지만 가상 공간에서 사업을 한다고 발표하였다. 이것은 전통적인 해운거래소가 지니고 있는 비용증가, 고객확보, 시설확충, 영업강화 및 정보수집의 한계점을 극복하기 위한 방안으로 가상공간 선택은 사업 환경이 인터넷 공간으로 옮겨가고 있음을 의미한다. 이에 따라 업계 일각에서는 세계적 해운거래시장인 Baltic Exchange 등을 중심으로 활동하고 있는 브로커들이 일자리를 잃고 대량 실직하는 등 해운거래시장의 판도가 완전히 흔들릴 것이란 전망도 나오고 있다.

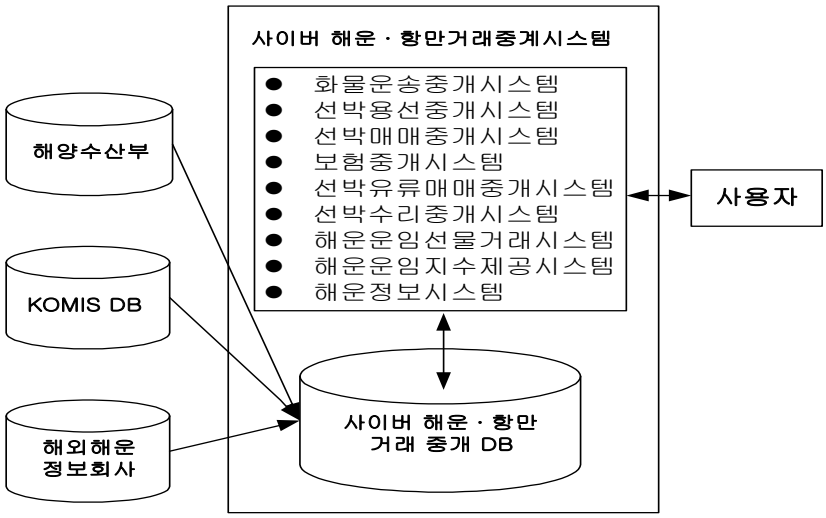

Fig. 2.5 Conceptual Diagram of Data Transaction on Shipping Electronic Commerce (출처: [67])

Fig 2.5에서 알 수 있듯이 사용자는 사이버 해운·항만거래중개 시스템(화물운송중개 시스템, 선박용선중개 시스템, 선박매매중개 시스템, 보험중개 시스템, 선박유류매매중개 시스템, 선박수리중개 시스템, 해운운임선물거래 시스템, 해운운임선물거래 시스템, 해운운임지수제공 시스템, 해운정보 시스템)을 이용함에 있어 사이버 해운·항만거래중개 데이터베이스를 둠으로써 해양수산부 데이터베이스, KOMIS 데이터베이스, 해외해운정보회사 데이터베이스 등과 같은 전문화된 데이터베이스에 접근할 수 있다는 것을 개념적으로 보여준다. 그러나 아직은 이들 전자상거래 업체들의 인터넷 서비스는 아직 본궤도에 올라있지 않지만 온라인 거래, 화물 추적, 업체 정보 및 운항일정을 포함하여 광범위한 서비스를 제공을 목표로 하고 있다. 특히 WTO 체제 아래 세계경제는 상호간에 실질적인 도움을 주고받으며 선의의 경쟁시대에 돌입하였으며 세계경제의 추이로 보아 더 이상 자국의 이익만을 위한 경제제도나 규범만을 고집할 수 없는 시대의 흐름에 부응하는 것은 당연한 변화라

할 것이다.

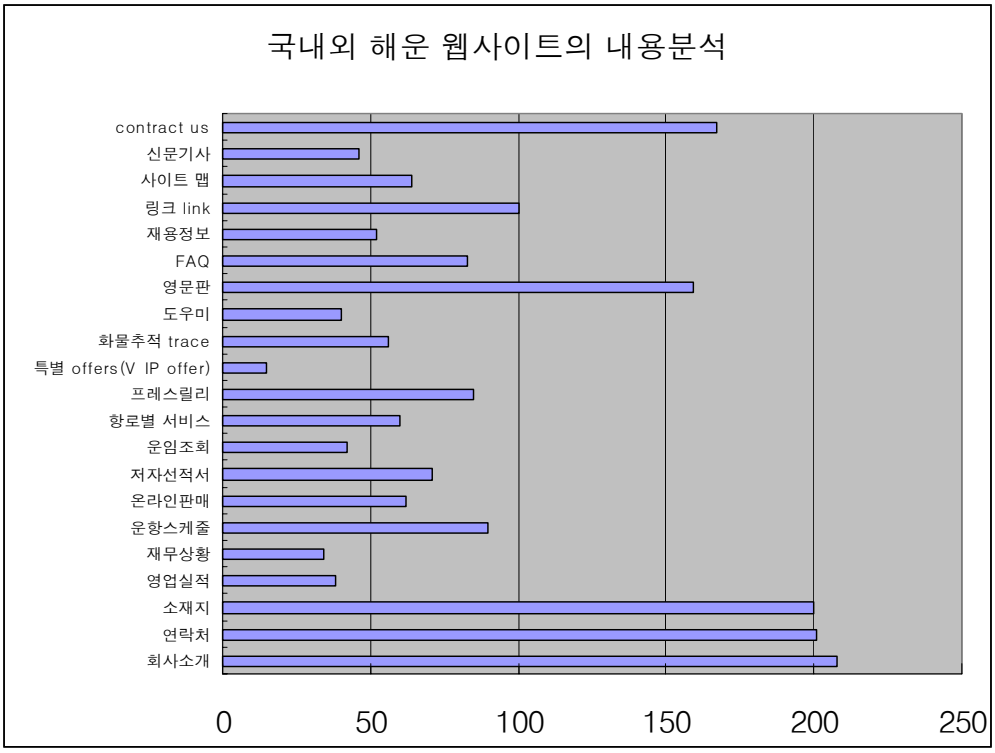

Fig. 2.6 Shipping Web Sites of Contents Analysis(출처 : [68])

Fig. 2.6에서와 같이 아직까지 낮은 웹사이트를 구축을 보여주고 있으며 조사대상 기업 중 2/3이상이 비즈니스에 적극적으로 활용하기보다는 단순히 웹 공간상의 영역을 확보하는 수준에 머물고 있다는 것을 알 수 있다[69].

2.2.2 선박의 가격결정 및 용선시장

선박을 평가하는 목적은 매매, 투자, 융자, 자산재평가, 소송관계의 처리를 원활하게 하는 것이며 평가의 목적에 따라 평가방법이나 표현방법이 달라질 수 있다.

선박의 평가를 위해서 먼저 선박의 특징을 살펴보면 다음과 같다[70].

첫째, 선박은 선체의 구조, 기관, 의장품별로 구분하여 평가되지만 단가가 고가인 것이 대부분이다.

둘째, 금융관계 등에서는 담보취득 시 동산 또는 의제부동산으로써 부동산에 비해 안전성의 결여로 담보취급이 타부동산 보다 낮은 것이 보통이다.

셋째, 선박 가격은 선박의 종류, 용도에 따라서 국내외 경제의 불황 등 경제여건화로 연휴상태가 계속되면 매매가 중단되기도 하고 수·해운업의 정도에 따라 가격이 민감하게 작용하기도 한다.

넷째, 선박의 내용년수가 지나면 선박으로서 효용가치가 없는 것으로 본다.

다섯째, 부동산이 영속성을 갖는 자연적 특성이 있는데 비하여 선박의 수명은 상대적으로 짧은 편이므로 이들 내용년수에 따라 감가 수정에 유의해야 한다.

따라서 선박을 평가할 때에 이상과 같은 선박의 특징을 충분히 감안하여 정확한 평가가 되어야 할 것이다. 선박의 매매는 대체로 선진국에서 매도하려하고, 후진국 또는 개발도상국인 경우 대체로 매입하려는 경향이 강하다. 선박을 매매하는 방법에는 지불 보증 없이 차용 선료를 지불하듯이 선가에 이자를 일정기간 간격으로 선주 혹은 선주가 지정한 은행에 사려는 자가 지불 완료한다. 또는 본선을 매매하거나 아니면 자기자금이나 혹은 차용으로 선가를 일시 지불하여 선박을 매매하기도 한다.

이 장에서는 SECA 구현에 앞서 먼저 실무에서 필요한 매매선박의 가격결정, 선박매매의 절차, 선박매매계약서 내용을 알아야 할 것이다. 국제해운시장이나 국내해운시장에서 많은 선박의 매매가 이루어지고 있다는 것은 여러 가지 리포트에 의해 알려져 있으며 선박가격의 상황은 해운시장의 호불황, 전쟁동란, 기타 요인에 의해서 움직인다. 즉, 수요에 대한 즉응성이라고 하는 커다란 메리트를 주축으로 해서 어떤 때는 크게, 어떤 때는 적게 이루어지고 있다. 최근에는 선박의 기술혁신이 대형화, 특수화, 고속화됨에 따라 선박의 진부화 속도가 옛날에 비해서 빠른 관계로 비교적 선령이 적은 선박도 매매되고 있는 실정이다. 특히 중고선의 가격도 해운의 사정을 달리하는 지역의 선사들에 의해 많이 이루어지고 있다.

매매선박의 가격결정의 요인으로는 먼저 선박에 대한 감정평가는 일반 부동산보다 더욱 고도의 공정성, 신뢰성, 공익성, 사회성이 요구되며 선박의 평가방법으로는 원가방식·비교방식·수익방식으로 분류될 수 있다. 선박의 수익력은 그 선박의 속도, 선형, 하역조건, 선가와 운임에 의해서 결정되고 있으므로 선박을 취득하는 입장에 서게되면 선가는 가능한 한 저렴해야 한다. 그러나 매매선가는 일반적으로 운임시황을 기반으로 팔려는 사람과 사려는 사람의 수급관계에 의해서 결정되며 매매선가에 대해서는 운임지수와 같은 매매선가수수가 없기 때문에 선가의 추세를 알기 위해서는 일정한 기간에 이루어진 매매선약선 중 선형, 선령, 성능이 비슷한 선박의 실무매매 가격을 참고로 비교 검토가 필요할 것이다.

우리 나라에서는 아직 선박매매업이 그렇게 활성화되어 있지 않기 때문에 선박과 화물을 중심으로 영업하는 용선중개인들이 핵심적인 역할을 담당하고 있다고 말할 수 있다. 2001년 2월말 현재 224개 업체(순수한 용선중개업체는 약 170여개)가 활동하고 있는 것으로 나타났다[71]. 기존 브로커를 이용할 경우 선주와 용선주가 각각 커미션을 브로커에게 지급하고 있는데, 인터넷 용선을 이용할 경우 비용을

절감할 수 있는 장점이 있다[72]. SeaLogistics사는 최근 Seasupplier.com으로 서비스를 하고 있으며 OneSeaDirect 사와 합병을 통해 인터넷을 이용한 용선 및 전자상거래의 경쟁력을 강화하였다. SeaLogistics사는 석유 메이저 회사들이 Texaco, Chevron, Koch Industries, LG Caltex 등의 강력한 지원을 받고있는데 SeaLogistics사는 전 지구적 용선시장을 겨냥하는 반면 OneSea는 해운기업의 선수품 조달, 선원 크루즈 부문을 담당하고 있다. 용선시장의 형성은 주로 정기선 해운(Liner shipping)시장 보다는 액체화물 및 대량벌크화물시장을 주요 대상으로 하고 있는데 선박 관리까지를 통합하여 제공하는 등 그 영역을 확산시키고 있다.

2.2.3 해운전자상거래와 중개인

전통적으로 해운업은 해상운송(Shipping, Sea Transport, Ocean Transportation) 서비스에 관련된 기업군이며 해상에서 여객이나 화물을 이동시키는 운송 서비스의 생산을 말한다[73]. 해운업이 경제적으로 존립하기 위해서는 부대비용을 부담할 수 있고, 계속 발전하기 위해 필요한 자본을 축적할 수 있어야 하며, 독립체로서 해운업은 다음 조건으로 특징지어진다[74].

첫째, 해운업은 어떤 행태든지 선박과 관련되어 있다.

둘째, 해운업은 생산과 판매가 동시에 이루어지는 서비스 산업이다.

셋째, 해운업 종사자에 대한 자본비율은 일반 제조업에 비해 수배가 높으며 최근 더욱 증가하는 추세를 보이고 있다. 특히 이는 선박매입비용, 항구부대설비 등에서 두드러진다.

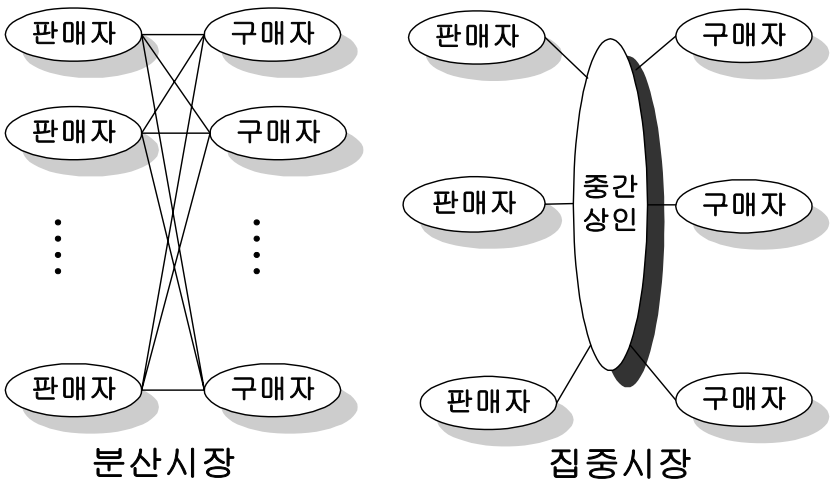

Fig. 2.7 Basic Frame of Market

최초로 해운서비스의 서비스 품질에 관한 중요성은 피어슨(Pearson, 1980)이 강

조하였다[75]. 거래란 구매자와 판매자가 제품과 서비스를 교환하는 것을 의미하며 구매자와 판매자가 정보를 교환하는 단계, 거래조건을 협상하는 단계, 정산과 계약 이행을 모니터링 하는 단계로 진행된다.

Fig. 2.7에서 알 수 있듯이 분산시장(Decentralized Market)에서, 구매자와 판매자가 중간상인(Intermediary, 브로커, 소매상, 경매상)을 통해 연결되는 집중시장(Centralized Market)의 형태가 변모되었다[76]. 즉, 분산시장에서 모든 판매자(n명)과 구매자(n명)의 접촉 횟수는 n^2이고, 집중시장에서 접촉횟수는 $2n$이다. 따라서 중간상인은 판매자와 구매자의 접촉횟수를 감소시켜 준다[77].

선박매매중개인(Sale and Purchase Broker)을 간단히 정의하면, '타인간의 선박매매의 중개를 영업으로 하는 자'라고 할 수 있으며 이를 좀 더 구체적으로 설명하면 '해운중개업에 종사하는 중개인으로서 선박을 팔고자하는 자와 선박을 사고자 하는 사람 사이를 연결시켜 주고 매매계약이 성약될 수 있도록 중개하는 자'라 할 수 있겠다. 즉, 선박을 팔고자하는 선주와 선박을 사고자 하는 구매자 사이에서 그 매매를 중개하는 자가 선박매매중개인이다. 선박매매중개인 중에는 선박을 팔고자하는 선주측에서 일하는 선주측 중개인(Ower's broker)과 구매자측에서 일하는 구매자측 중개인(Buyer's Broker)이 있으나, 이것이 고정되어 있는 것은 아니며 상황에 따라 그 처지가 뒤바뀔 수 있으며 , 선박매매중개인이 선주측의 선박 매도를 의뢰받아 구매자를 중개시켜 주는 역할을 하게 되며 선주측 중개인으로 활동하는 것이고, 구매자측의 의뢰를 받아 선박을 중개시켜 주는 역할을 하게 되면 구매자측 중개인으로 활동하게 되는 것이다.

Buxmann과 Gebauer(1998)은 전자시장의 구조가 지리적 인접성 보다는 핵심역량을 지닌 사업별 중간상인 집단으로 구성될 것이라고 보고 있다. 구매자측의 구매의사 공개(Order)→선박매매중개인의 중고선 시장 탐색(Search)→적정 후보 선박

선정→제의(Proposal)→각종 명세 확인→검선(Survey)→협상(Nego)→계약확정(Confirm)→성약(Fix)→인도(Delivery) 순으로 이루어진다.

인터넷 기반으로 화주의 요구를 파악하여 이를 충족시킬 수 있는 선사를 화주와 연결시켜주는 해상운송중개인(Ocean Transportation Intermediarty)의 출현에 따라 해상운송주선인(Freight Forwarder)과 무선박운송업체(NVOCC : Non-Vessel Operating Common Carrier)와의 차이가 없어지면서 소형 화주들은 해운기업과 직접 운송계약 체결이 많아질 것으로 보인다.

우리 나라는 세계 1, 2위의 조선 능력을 보유하고 있으며 세계 10대 선복량 보유국가인 만큼 앞으로는 조선업계 및 해운업체와 국내해운중개업이 상호 연계되어 국내 고객뿐만 아니라 외국 고객들에게도 양질의 서비스를 제공할 발판을 마련해야 할 것이다. 물론 인터넷의 영향으로 판매자와 구매자가 직접적인 관계를 맺을 가능성이 높아지고 이에 기존의 중개업이 소멸하는 것으로 의미하기도 하지만 Saker, Butler, Steinfield(1995)는 탈중개화 가설[78]에 논리적 결함이 있음을 지적하고, 전자상거래의 등장은 판매자와 구매자간의 직접거래비용 뿐만 아니라 중개업자의 중개수수료 또한 감소시킬 수 있음을 주장하였다.

한편 Bailey와 Bakos(1997)는 사례연구를 통하여 전자상거래에 참여하는 모든 기업이 "탈중개매매조직(Disintermediary)" 현상을 겪는 것이 아니라, 오히려 전자상거래를 이용함으로써 새로운 형태의 역할을 수행할 수 있다는 증거를 제시하였다[79]. 그러나 해운전자상거래가 활성화되더라도 용선거래 시장에서는 탈중개화 현상은 나타나지 않을 것으로 보이며, 또한 이미 용선시장에 존재하고 있는 온라인 용선 중개업자들의 경우 기존의 오프라인 용선 중개업자들의 역할을 대체하기보다는 거래비용의 감소를 통해 이들의 업무 효율성을 증진시켜 상호 보완 관계를 형성할 것이다.

제 3 장 클라이언트/서버 구조와 CORBA

3.1 클라이언트/서버 시스템

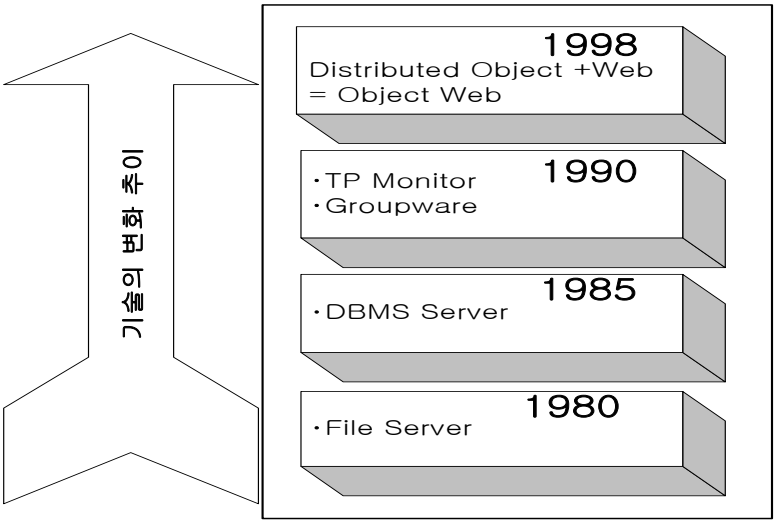

Fig. 3.1 Development Process of Technology Client/Server Systems

'클라이언트/서버[79]'라는 개념은 메인 프레임/단말기 구조의 문제점을 해결하기 위한 구세주 같은 존재로 컴퓨터 업계에 등장했다. 클라이언트/서버 환경은 1980년대 초반 이후 기업내의 컴퓨터의 사용 형태의 변화에 발맞추어 발전하여 현재는 모든 기업이 선호하는 어플리케이션 아키텍처가 되었다. 최근 정보는 기하급수적으로 증가됨에 따라 정보의 공유측면에서 더욱 분산 컴퓨팅환경의 요구가 절실하게 되었으며 보다 효과적인 모델로 부각되고 있다[80][81]. 클라이언트 시스템은 서버 프로세스가 있는 특성이나 장소에 상관없이 네트워크의 서버 시스템에게 서비스를 요청할 수 있다.

클라이언트/서버 구조는 다음과 같은 장점을 제공한다[82].

- 데이터에 대한 부서별 접근을 가능하게 한다.
- 보다 효율적인 형태로 데이터를 제공할 수 있다.
- 최소한의 비용과 노력만으로도 MIS의 데이터 무결성을 중앙 집중적으로 관리할 수 있다. 여기서 데이터 무결성을 보장한다는 것은 서버의 데이터가 정확하며 완전한 상태임을 유지한다는 것을 뜻한다.
- 전체 데이터베이스에 데이터 무결성을 강제할 수 있다.
- 클라이언트와 서버 사이의 처리 부담을 적절히 분산할 수 있다.
- 대부분의 데이터베이스 서버들이 제공하는 개선된 데이터 무결성 기능을 사용할 수 있다.
- 테이블 전체를 필요로 하는 데스크탑 데이터베이스 어플리케이션에 비해, 꼭 필요한 데이터의 부분 집합만을 클라이언트에게 제공하여 네트워크 트래픽을 줄일 수 있다.

그리고 클라이언트/서버 모델에서 서버는 정해진 제한된 서비스만을 제공하고 이외의 서비스는 클라이언트 측에서 이루어져야 하며, 클라이언트가 요청하는 작업이 서버에 구현되어 있지 않은 경우에는 프로그램 작성자가 서버에 새로운 기능을 첨가하거나 클라이언트가 일련의 서비스 호출을 수행하면서 서비스를 제공해야 했다.

Fig. 3.1에서는 클라이언트/서버 시스템의 발전과정을 보여주고 있는데, 1998년 객체 웹(Object Web) 기술의 발전을 가능하게 한 것은 급속한 네트워크의 발전과, 소프트웨어의 복잡도가 커지면서 요구되는 프로그래밍 언어에 대한 독립성, 네트워크 투명성, 이식성, 상호운용성을 가능케하는 CORBA의 역할이 컸다. 이것의 좋은 예로 ORB가 있으며 이는 네트워크상에 흩어져 있는 객체에 대해 네트워크 장비나 운영체제, 혹은 그 객체가 존재하는 응용 프로그램에 관계없이 객체의 동작을 수행

시킬 수 있게 한다. 이것은 명백히 상호운용성을 제공한다고 할 수 있으며 여러 단계의 추상화를 통해 얻어진다. 여기서 객체란, 잘 정의된 명령을 수행하고 이해되는 추상적인 엔티티(Abstract Entity)이다. 이 명령들은 객체의 '메소드'라 불리며 분산객체관리 분야에 부응하는 해법이 CORBA와 JAVA와 웹기술을 접목하는 형태의 객체 웹이다. 웹과 객체지향 기술을 통합하려는 시도에서 사용한 언어가 JAVA다. 그러므로 객체 웹이란 JAVA, 웹 기술을 통합하는 시스템 구조를 이룬다고 할 수 있다. 그러나 JAVA는 객체 웹 기술에서 필요한 도구지만 충분한 도구는 되지 못한다.

이것을 보안해주는 분산객체 기술이 CORBA 있다. 클라이언트/서버 환경에서 서버는 단순한 데이터 제공 작업 이외에도 훨씬 더 많은 일을 하게 된다. 객체지향 프로그램만을 사용하여 프로그램을 개발해도 서로 다른 기계와 운영체제에서 최종기계 코드를 연결하는 것은 어렵다. 왜냐하면 서로 다른 시스템간 또한 다른 언어 객체를 사용하는 것은 곤란하기 때문에 상속받을 때도 슈퍼클래스까지 접근해야 한다. 따라서 프로그램 내용이 변화되면 다시 컴파일하고 링크해야 한다. 이러한 문제 해결을 위해 컴포넌트라는 방식을 나오게 했던 배경이 된다.

3.1.1 클라이언트/서버 에이전트의 객체 호출

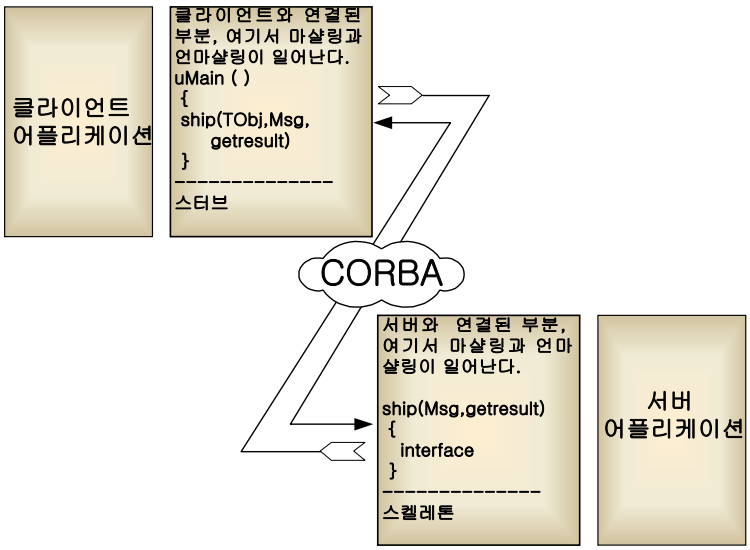

Fig. 3.2 Static Calling Application

Fig. 3.2는 클라이언트/서버 환경하에서 어플리케이션의 기능 모델을 보여준다. 서버 어플리케이션은 컴파일러에 의해 생성된 서버 측 코드와 연결되어 있다. 서버 측 코드에는 역시 마샬링과 언마샬링 코드가 존재한다. 화살표의 끝을 가지 선들은 데이터의 스트림을 보여줄 뿐 다른 의미는 없다. 구체적으로는 클라이언트 어플리케이션 안에 서버 어플리케이션의 함수를 직접 코딩하고, IDL 컴파일러가 컴파일하여 자동생성시켜 링크된다.

클라이언트 어플리케이션에서 서버 어플리케이션을 호출하는 방법은 서버 어플리케이션의 함수를 직접 클라이언트 어플리케이션 속에서 호출하는 방법과 ORB가 준비하는 서버 어플리케이션을 호출하기 위한 함수를 사용해서 호출하는 방법에는 두 종류가 있다.

이러한 호출방법의 차이점은, 첫 번째 서버 어플리케이션의 인터페이스를 클라이언트 어플리케이션에서 어떻게 지정할 것인가 하는 것이다. 정적호출 인터페이스는 서버 어플리케이션의 인터페이스를 클라이언트에 고정하기 때문에 성능면에서는 동적호출 인터페이스 보다 우수하나, 서버 어플리케이션에 변경이 있을 경우 그때마다 클라이언트 어플리케이션도 변경해 주어야 한다. 그러나 동적호출 인터페이스는 클라이언트 어플리케이션의 내부에서 호출한 서버 어플리케이션의 함수를 직접 코딩하지 않기 때문에 서버 어플리케이션이 변경되어도 클라이언트 어플리케이션을 수정할 필요가 없다.

두 번째는 클라이언트 어플리케이션에 서버 어플리케이션으로 오퍼레이션을 보내는 방법의 차이이다. 오퍼레이션을 보내는 방법에는 호출한 함수의 처리 결과가 클라이언트 어플리케이션으로 돌아올 때까지 다른 처리를 행하지 않는 동기와 호출한 함수의 처리결과가 돌아올 때까지 기다리지 않고 다른 처리를 하는 비동기 두 종류가 있다. 정적 호출 인터페이스의 경우 오퍼레이션은 모두 동기로 보낸다.

3.1.2 클라이언트/서버 개발

인터넷이 보편화되고 컴퓨터 및 네트워크의 성능이 향상됨에 따라 컴퓨터를 이용하는 양상은 변하고 있다. 분산처리 분야에서 네트워크의 기술적인 변화는 통신망을 통해 서로 연결된 컴퓨터로 어떤 문제를 해결하는데 사용할 수 있도록 해주고 있다.

원래 웹은 클라이언트와 서버로 이루어져 있는데, 클라이언트는 브라우저를 가지고 있으며, 서버는 클라이언트의 브라우저로 볼 수 있는 데이터를 저장하고 있는 형태의 2계층 시스템으로 생각되었다. 그러나 클라이언트 작용이 점점 더 중요해지고, 웹에 담긴 내용이 동적으로 변함에 따라서 중간계층에 대한 필요가 분명해지게 되었다. 이 논문에서는 CORBA를 통신 미들웨어로 사용하였는데 CORBA는 객체지향 언어의 특징보다 유연한 통신 메커니즘을 지원하기 때문에 기존의 절차적 RPC(Remote-Process Calling) 미들웨어보다 진보된 것이다. 원격프로시저 호출은 동기적 호출이므로 호출이 실제로 돌아올 때까지 어플리케이션은 블록킹된다. 이것은 병렬 설계 구조의 목적에 위배된다. 따라서 분산된 환경에서는 이러한 블록킹 테크닉은 긍정적으로 보지 않는다.

여기서는 인터페이스 정의 언어를 선언할 필요가 없고, 타입 라이브러리를 작성할 필요가 있으며 대부분 CORBA 객체 초기화와 팩토리 설정 사항은 자동적으로 코드가 만들진다. 다만 타입 라이브러리에서 선언한 인터페이스에 대한 메소드 부분만 논리적으로 처리할 수 있게 하였다. 이종의 데이터베이스에서 데이터베이스를 가져오고 처리하는 부분은 서버 어플리케이션에서 구현하고, 클라이언트는 서버에서 데이터를 가져오는 메소드만을 정의하면 된다. 클라이언트/서버 데이터베이스 시스템에서 클라이언트 시스템은 데이터베이스 서버에 데이터를 요구하면 데이터베이스 서버는 조건에 맞는 선박 레코드를 추출하여 네트워크를 통하여 클라이언

트 시스템에게 전송한다. 데이터 처리 작업의 분리는 클라이언트/서버 시스템이 메인 프레임을 기반으로 하는 중앙 집중된 데이터베이스 시스템과 다른 점이다.

이 논문에서는 클라이언트/서버 모델로 개발하였다. 여기서는 BDE(Boland Database Engine)를 통한 데이터베이스로 연결된다. 즉, SQL Links를 통한 오라클(Oracle), 사이베이스(Sybase), 인포믹스(InforMix), 마이크로소프트 SQL 서버, 데 DB2, 인터베이스(Interbase) 같은 서버들에 대한 고유한 드라이버들을 제공하며, 그 외의 데이터베이스들을 위한 ODBC 연결도 제공한다.

여기서 SQL Server는 ODBC DSN으로 연결하였으며 알리아스(alias)는 sqlShip이고 InterBase의 Ship은 203.255.212.90:c:\shipagent\data\ship.gdb에 올려져 있다. BDE 수준의 데이터베이스 알리아스는 TDatabase.AliasName속성으로 설정한다. TDatabase 컴포넌트를 사용하면 사용자가 서버 데이터에 로그인하는 과정을 직접 제어할 수 있다. 사용자는 로그인 과정에서 유효한 사용자 이름과 패스워드를 입력해야만 원하는 데이터에 접근할 수 있게 된다. 기본적으로, 서버 데이터베이스에 연결을 시도하면 표준적인 로그인 대화상자가 나타난다. 에이전트 시스템은 분산 데이터 처리를 위해 여러 클라이언트의 질의 요청을 받게되면 서버 시스템으로 접근할 수 있는 원격지의 시스템 정보를 획득하여 연결되고 클라이언트의 질의에 해당하는 검색자료를 SECA가 찾게 해준다.

3.2 CORBA

1980년대 이후로 사람들은 객체지향 기법을 이용하는 것에 대해 많은 연구로 프로그래밍 언어, GUI 그리고 데이터베이스 등 여러 분야에서 객체지향 기법을 이용하여 시스템을 구성하였다. 그러나 지금은 분산 환경에서 서로 다른 시스템에 상관없이 투명하게 응용프로그램 운용할 수 있는 방법을 바탕으로 응용 프로그램들을 결합하기 위한 객체지향 표준을 제정하기 위해 OMG는 분산처리 환경에서 객체간의 상호운용이 가능하도록 표준 스펙인 CORBA를 개발하였다[83][84][85]. 이외 현재 어플리케이션 보다 쉽게 개발하기 위한 미들웨어로 DCOM, 그리고 JAVA RMI(Remote Mothod Invocation)[86][87]가 널리 쓰인다.

CORBA는 OMG의 OMA(Object Management Architecture)[88][89][90] 핵심 요소로써 객체 모델을 기본 모델로 이용되며, 개방 분산 환경에서 응용프로그램들간의 상호운용성 및 이종의 객체들에 대한 요청 및 응답에 대한 투명성을 제공하고, 주요 구성요소들의 재사용을 보장하는 기능을 제공한다. 이 외에도 OMA에는 응용프로그램인 응용객체와 응용프로그램으로 제공되는 공통의 객체 또는 클래스들의 모임인 공통설비들이 있다. CORBA는 언어, 플랫폼, 위치 투명성이 보장되며 초기 CORBA에서는 인터페이스 정의언어 정의와 어플리케이션 프로그래밍 인터페이스를 제정함으로써 특정 ORB의 구현 내에서 클라이언트와 서버와 상호 소통할 수 있도록 하였고, 이 논문에서는 사용된 ORB는 Visigenic의 Visibroker 4.0이다.

CORBA는 특히 개방형 구조(Open Architecture)이기 때문이며 XML의 상호연동을 위해서 DOM(Document Object Model)[91][92] API를 사용하여서 컴포넌트 기반 기술과 웹과의 호환이 가능하다. ActiveX와 DCOM 구조는 인텔 CPU와 윈도우 기반의 제품군에 한해서만 제대로 작동되므로 사용자로 하여금 선택의 기회를 그

만큼 좁게 만들며 지나치게 마이크로소프트사 위주로 진행되지만, 이에 비해 JAVA나 CORBA는 표준화[93][94][95] 방안에 대해 비교적 개방적이기 때문이다.

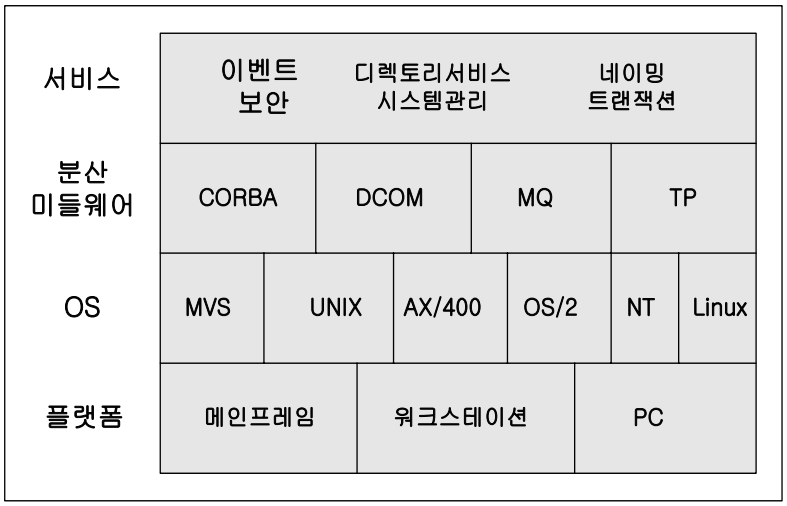

Fig. 3.3 Logic Architecture of Middleware

Fig. 3.3 은 미들웨어 구조를 설명한 것으로 여기서 특히 CORBA가 주목받는 이유는 CORBA가 기존에 존재하는 모든 다른 형태의 클라이언트/서버 미들웨어를 포함할 수 있는 잠재력을 미들웨어를 정의하기 때문이다. CORBA는 기존의 어플리케이션을 버스에 가져가기 위한 단일화된 메타포어(Metaphor)로서 객체를 사용과 더불어 서비스에 대한 명세는 항상 구현과 독립된다. 이것이 기존 시스템들을 버스 내에 통합할 수 있게 한다. CORBA의 객체는 캡슐화가 되어 인터페이스만을 접근할 수 있으며 내부로는 접근을 할 수 없다. 또한 CORBA로 위치의 투명성을 제공하며 이는 어떤 객체들이 같은 기계의 프로세스인지 다른 프로세스에 있는 것인지를 구분하지 않고 위치에 상관없이 호출된 객체에 대해 똑같은 방법으로 메소드가 참조됨을 의미한다.

3.2.1 CORBA의 호출 원리

CORBA를 통하여 개발자들은 서로 다른 플랫폼 상에서 작성된 분산 응용프로그램을 하나의 클라이언트/서버 시스템으로 통합할 수 있게 되었으며 객체지향 개념을 바탕으로 네트워크상에 분산된 다양한 객체들을 손쉽게 호출할 수 있게 되었다[96]. 일반적인 객체지향 코드에서는 메소드 호출자와 이에 대한 호출처리자가 하나의 프로세스에 존재한다. 그러나 CORBA와 DCE(Distributed Computing Environment)[97]같은 분산객체 환경에서는 호출자와 호출처리자가 네트워크를 사이에 두고 분산되어 존재한다. 이는 호출자인 클라이언트 쪽에서는 네트워크를 통해 서비스를 호출하고 그 결과를 전달받는 코드가 필요한데 이를 '스터브(Stub)'라 하며, 호출처리자인 구현 객체 쪽에서는 네트워크를 통해 전달받은 서비스 호출을 처리하고 그 결과를 다시 클라이언트에게 전달해 주는 코드가 필요하며 이를 '스켈레톤(Skeleton)'이라 한다.

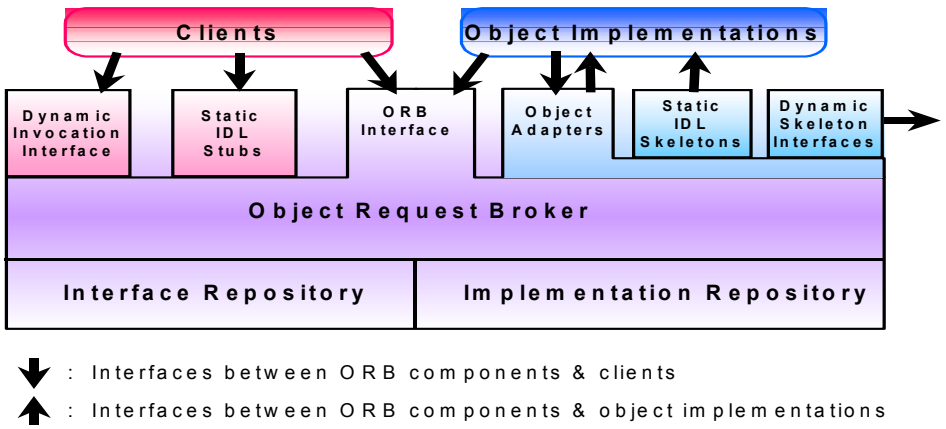

Fig. 3.4 Components of CORBA Distributed Object Computing Model

파생 방법에서는 ORB의 IDL 컴파일러는 IDL 인터페이스와 데이터 타입에 기반하여 적당한 스켈레톤 클래스를 생성되며 실행에 필요한 모든 기능의 코드는

CORBA에 의해 자동으로 제공되며 서비스가 호출되는 원리는 Fig. 3.4와 같다.

① 클라이언트 IDL 스터브

클라이언트 IDL 스터브는 객체 서비스에 대한 정적 인터페이스를 제공한다. 이 프리 컴파일된 스터브는 클라이언트가 서버상의 해당 서비스를 호출하는 방법을 정의한다. 즉, 클라이언트는 이 IDL 스터브을 통해 ORB와 통신하게 되며, 스터브 객체를 통해서 클라이언트와 서버는 서로 통신하는 상대방이 마치 자신의 로컬 객체인 것처럼 프로그래밍할 수 있다[98].

② 동적호출 인터페이스(DII : Dynamic Invocation Interface)

서버에 있는 객체의 메소드를 실시간에 찾아내도록 해준다. 이는 인터페이스 저장소라는 데이터베이스에서 해당 클래스의 모든 인터페이스를 저장해두고 동적 호출이 수행될 때 이를 참조하여 해당 메소드를 수행한다. 서버 측에서 DII에 해당하는 것은 동적 스켈레톤 인터페이스(DSI : Dynamic Skeleton Interface)이다.

③ 인터페이스 저장소(Interface Repository) API

인터페이스 저장소 API는 모든 등록된 컴포넌트 인터페이스, 그들이 지원하는 메소드와 요청하는 파라미터들에 대한 명세서를 확보하고 수정을 할 수 있도록 한다. IDL로 정의된 인터페이스를 기계가 읽을 수 있는 형태로 담고 있는 실시간에 생기는 일종의 데이터베이스이다.

④ ORB 인터페이스

어플리케이션에서 필요한 로컬 서비스의 API를 담고 있다. 예를 들면 객체의 레퍼런스를 문자열로 바꾸어 주는 것 등이다. 이는 객체의 레퍼런스를 저장 등의 작업을 할 때 유용하다.

⑤ 서버 IDL 스켈레톤

서버가 제공하는 모든 서비스의 정적 인터페이스를 제공한다. 즉, 클라이언트 서비스 요청에 대한 구현된 객체를 직접 호출하며, 처리된 결과를 클라이언트에게 전달한다.

⑥ 동적 스켈레톤 인터페이스(DSI : Dynamic Skeleton Interface)

동적 스켈레톤은 목표 객체와 메소드를 식별하기 위하여 유입 메시지 내의 파라미터 값을 살핀다. 반면, 일반 컴파일된 스켈레톤은 특정 객체 클래스에 대해 정의되고 각 IDL 정의 메소드에 대한 메소드 구현을 기다린다.

⑦ 객체 어탭터(Object Adapter)

객체 어댑터는 ORB의 핵심 서비스상에 존재하며 서버의 객체를 대신하여 서버스에 대한 요청을 수용한다. 예를 들어, 클라이언트가 서버 객체의 메소드를 요청하기 위해서는 서버 객체에 대한 참조를 가지고 있어야 하는데, 이 객체 참조를 얻기 위해 ORB의 서비스를 요청하게 되고, 이후 이를 사용하여 특정 객체에 대한 메소드를 호출하는 데 사용한다.

⑧ 구현 저장소 (Implementation Repository)

클라이언트의 호출이 발생하면 통신을 담당하는 ORB기능을 통해 구현객체에 전달되고, 구현 객체 쪽의 객체 어탭터는 기본적으로 해당 객체를 호출 가능하게 생성하고 원하는 호출에 해당하는 메소드를 실행시킨다. 실행된 메소드에 의해 처리된 결과는 다시 클라이언트에 전달된다. CORBA는 클라이언트에게 정적호출과 동적 호출의 두 가지 호출 방식을 제공되며 정적인 호출방법은 ORB에게 제공하는 기본 API기능을 이용해 서비스를 요청하는 것이다.

3.2.2 분산 환경

오늘날 컴퓨터 산업에 있어 분산 컴퓨팅은 중요한 추진 중 하나이다. 지난 수년간 인터넷의 대중화와 컴퓨터 기술의 발전으로 분산 어플리케이션에 대한 관심이 증대되면서 분산객체기술은 복잡한 분산 어플리케이션 개발에 성공적인 패러다임으로 인식되었다. 현재 많은 프로그램의 개발환경이 객체지향 프로그래밍의 방법에 의해 행해지고 있다. 즉, 프로그래머가 구현한 객체들을 소프트웨어적으로 재사용을 가지도록 하는 것이다. 그러나 객체지향 프로그래밍은 그 객체를 사용함에 있어 그 객체를 구현하는데 이용한 컴파일러등 같은 환경에서만 사용이 가능하며 복잡한 상속 때문에 사용하기가 많이 번거롭다.

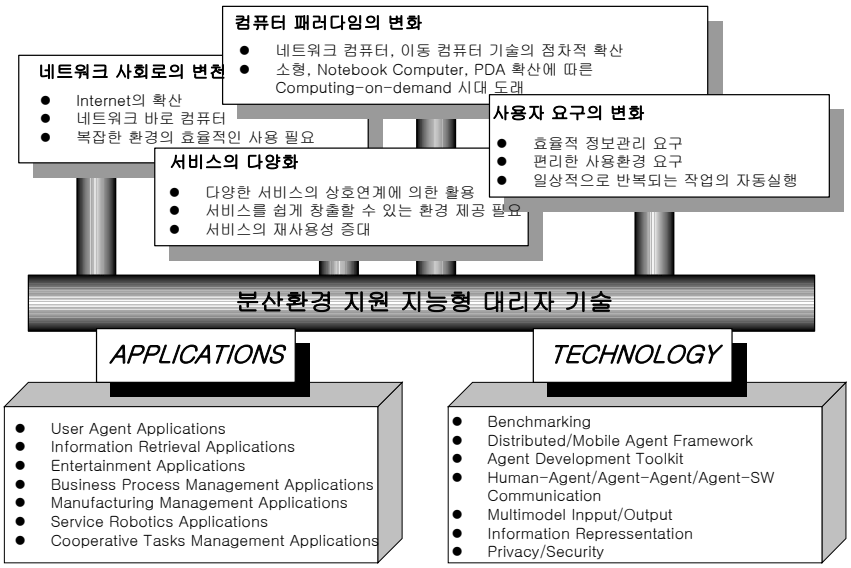

Fig. 3.5 Transformation of Computing Environment

하지만 이러한 단점을 극복하기 위한 것으로 컴포넌트 사용이다. 즉, 여러 객체지향 프로그래밍은 인터페이스를 통해 구현된 프로그램을 그대로 사용할 수 있으며, 객체지향 프로그래밍의 상속은 컴포넌트 내부에 한정시켜 운용하게 된다. 따라

서 이 컴포넌트의 개념을 이용하면 외부의 인터페이스를 통해서만 작동하며 외부와 철저히 단절되어 시스템의 단순성을 높일 수 있기 때문이다.

Fig. 3.5는 분산 환경 기반으로 하는 에이전트 기술 변화에 설명이다[99]. 분산 환경에서 수행되는 에이전트는 초고속 통신망과 일반 유무선 망을 기반으로 하는 분산 컴퓨팅환경에서 사용자의 요구 사항을 능동적으로 처리할 수 있는 지능형 소프트웨어를 지칭한다.

원래 웹은 클라이언트와 서버로 이루어져 있는데 클라이언트는 브라우저를 가지고 있으며 서버는 클라이언트의 브라우저로 볼 수 있는 데이터를 저장하고 있는 형태의 2계층 시스템으로 생각되었다. 분산 이종의 통합 검색 방법은 클라이언트/서버 구조를 기반으로 한다. 클라이언트/서버 구조는 2계층 클라이언트/서버 구조와 3계층 클라이언트/서버 구조로 분류한다[100].

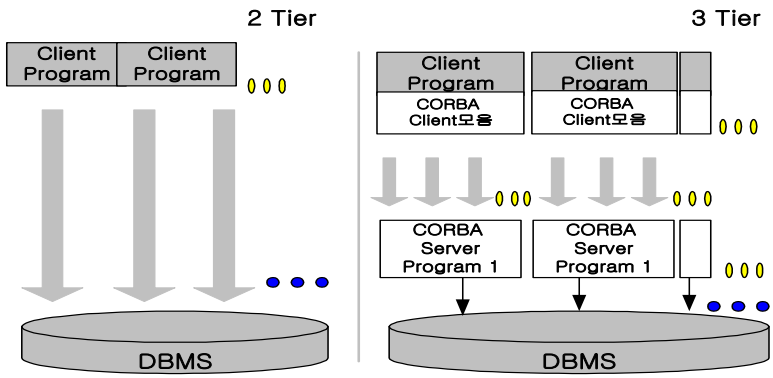

Fig. 3.6 Efficiency of Three-Tier Architecture System

2계층 클라이언트/서버 구조는 클라이언트에서 사용자와 인터페이스 하기 위한 화면 운용(표현 규칙의 운용 : Presentation Rule), 업무 규칙(Business Rule)의 구

현, 그리고 자료의 접근 규칙(Data Access Rule)의 운용에 대한 작업을 맡고, 서버는 주로 데이터를 관리하는 작업을 맡는 방식이다. (Fig. 3.6) 이 방식은 클라이언트에서 서버 접근 정보 및 검색 질의를 위한 모든 정보를 관리하여야 한다. 3계층 클라이언트/서버 구조는 사용자와 인터페이스 하기 위한 화면 운용을 클라이언트 시스템이 담당하고, 업무 규칙의 구현과 자료의 접근 규칙의 운용을 서버 시스템에서 담당하는 방법으로 서버와 클라이언트가 작업을 분담하는 형태로 시스템을 운용하는 방식이며 3계층 구조의 장점은 다음과 같다.

첫째, 대규모 인터넷과 인트라넷 클라이언트/서버 어플리케이션의 요구에 적합하다.

둘째, 대부분의 코드가 서버 상에서 실행된다.

셋째, 어플리케이션은 서비스의 수준을 만들어 네트웍 트래픽을 최소화한다.

넷째, 클라이언트 SQL에 대한 처리없이 서버상의 비즈니스 로직만을 호출하면 된다.

다섯째, 데이터베이스 스키마를 클라이언트에 노출하지 않음으로 우수한 보안을 제공한다.

분산시스템의 궁극적 목표는 사용자들이 서로 다른 장소에 위치한 컴퓨터들을 하나의 통합된 컴퓨팅 장비로 인식하여 시스템 자원의 위치에 관계없이 사용할 수 있는 환경을 제공해 준다[101]. 더욱이 지역간, 업종간, 기업간 다양한 자원을 효율적인 공유로 분산시스템의 규모가 점점 커지고 이종의 환경이 확대됨에 따른 CORBA에 대한 인식이 날로 높아지고 있으며, 현재 분산객체기술은 컴포넌트 개념을 포함한 EJB/CORBA 와 COM+/DCOM의 각축전 양상으로 진행되고 있다.

3.2.3 인터페이스 정의 언어

CORBA의 경우 분산된 네트워크상에 따로 위치한 객체간의 투명한 접근을 제공하는 서로간의 약속이 필요하다. 이런 약속을 정의할 때 특정언어(C, C++, JAVA, Object Pascal 등)에 의존하지 않는 인터페이스 정의 언어가 필요하게 되었는데, 그것이 바로 IDL이다. CORBA의 핵심은 각 컴포넌트들을 위한 명확하고 구체화된 인터페이스인 IDL를 제공하는 것이다. IDL은 실행되는 것이 아니고 단지 객체의 인터페이스를 언어에 독립적인 방법으로 기술하는 것으로 프로그래밍 언어가 아니다. 여기서 IDL은 프로그래밍 언어가 아니라 특정구현 언어로의 매핑을 가능하게 하는 선언적이고, 언어 중립적 언어라 할 수 있다[102][103].

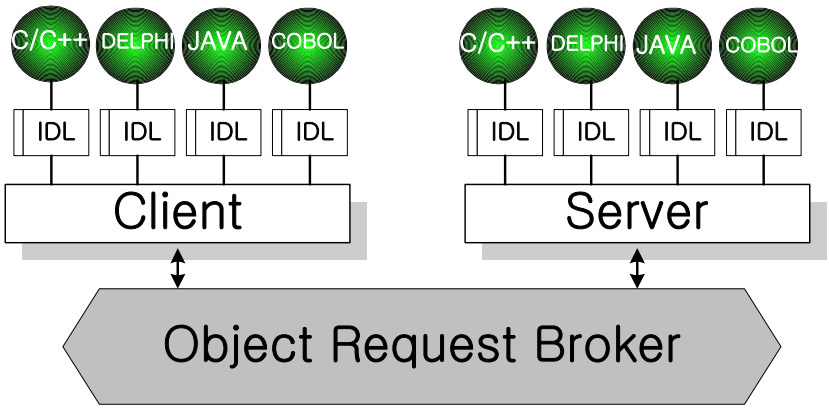

Fig. 3.7 CORBA IDL Language Binding

IDL의 목적은 분산 서비스와 상호작용을 하기 위하여 일반적인 데이터 타입으로 분산 서비스를 정의하는 것이다. 개발자가 IDL을 이용하여 어떤 인터페이스를 정의하면 IDL 컴파일러는 작성된 IDL 파일을 각각의 언어로 매핑 시켜준다. 따라서 인터페이스는 클라이언트와 구현 객체간에 맺어진 통신 규약이라 할 수 있다.

IDL은 Fig. 3.7에서 보듯이 ORB와 다른 프로그래밍 언어 사이의 표준적이고 공통인 인터페이스 역할을 담당한다고 할 수 있으며, 이런 공통의 인터페이스는 프로그래밍 언어와 플랫폼에 독립적인 구현 방법을 제공함으로써 시스템 구축에 있어 개발자가 많은 유연성을 가질 수 있다.

```
server agent idl
[   uuid(EEF5596F-DFE5-4B49-ACE3-316C6F016F01),
   version(1.0),
   helpString("Server Library")]
library Server
{   importlib("STDOLE2.TLB");
   importlib("STDVCL40.DLL");
   [ uuid(F5974473-7915-4774-A530-6630D9EED8BF),
      version(1.0),
      helpString("Dispatch interface for MSServer Object"),
      dual,
      oleautomation ]
    interface IMSServer: IDispatch
   { [id(0x00000001)]
      HRESULT _stdcall GetMsg([in] BSTR msg, [out, retval] BSTR
* Result );
      [id(0x00000002)]
      HRESULT _stdcall GetResult([in] BSTR msg, [out, retval] BSTR
* Result );
   };
   [ uuid(D3A5A9A1-9DB8-4018-86AF-4F1ED25CC098),
      version(1.0),
      helpString("MSServer Object") ]
   coclass MSServer
   {   [default] interface IMSServer;
   };
};
```

Fig. 3.8 Implementation of MS-SQL Server IDL

Fig. 3.8에서 보는 바와 같이 구현된 MS-SQL Server Agent IDL의 예이다. CORBA에서 ORB는 복수 벤더들이 ORB와 저장소들간에 컴포넌트와 이들의 인터

페이스를 고유하게하며 전역적으로 식별하기 위한 저장소 ID라는 전역 식별자를 제공한다. 저장소 ID는 IDL 내의 프라그마를 통해 생성되며 이 프라그마는 저장소 ID를 DCE의 UUID(Universal Unique IDentifiers)를 통해 생성된 것을 보여준다.

이 연구에서 구현된 타입 라이브러리인 인터페이스는 IMSServer이며, 이 인터페이스의 메소드는 GetMsg와 GetResult이며 타입라이브러리의 BSTR 타입의 경우 델파이 언어에서는 문자열로 변환되게 된다. CORBA 객체 작성시 주의해야 할 것은 타입 라이브러리이다. 이전 버전까지는 타입에 대해 신경을 많이 써야 했다. 그러나 이 논문에서는 타입 라이브러리 자체에 Native C++ 타입으로 표현되고, 실제로 구현되는 코딩에서는 자동적으로 오브젝트 파스칼 타입으로 변환되기 때문에 크게 신경쓰지 않았다. 클라이언트와 서버는 ORB와 CORBA의 객체 버스에 연결되어 각각 프로그래밍 언어에 상관없이 IDL이라는 언어 바인딩을 통해 통신한다. 클라이언트/서버 각각은 특정 프로그래밍 언어로 구현되어 있다고 하더라도 클라이언트 IDL 바인딩을 통해 투명하게 서버가 제공하는 서비스를 제공받을 수 있다.

제 4 장 CORBA 기반 해운전자상거래 에이전트 설계 및 적용

4.1 CORBA 기반 SECA

정보사회가 고도화됨에 따라 필요한 컴퓨팅 자원의 필요성은 급격히 증가하고 있으나, 한정된 자원으로 인하여 다수의 사용자가 필요한 자원을 충분히 사용하는 것이 매우 제한이다. 이 연구에서는 원하는 선박을 보다 쉽게 검색할 수 있는 에이전트를 둠으로써 각 데이터베이스 서버들 간의 통신을 통해 데이터를 공유함으로 보다 많은 데이터를 대상으로 검색을 할 수 있는 에이전트를 설계하였다.

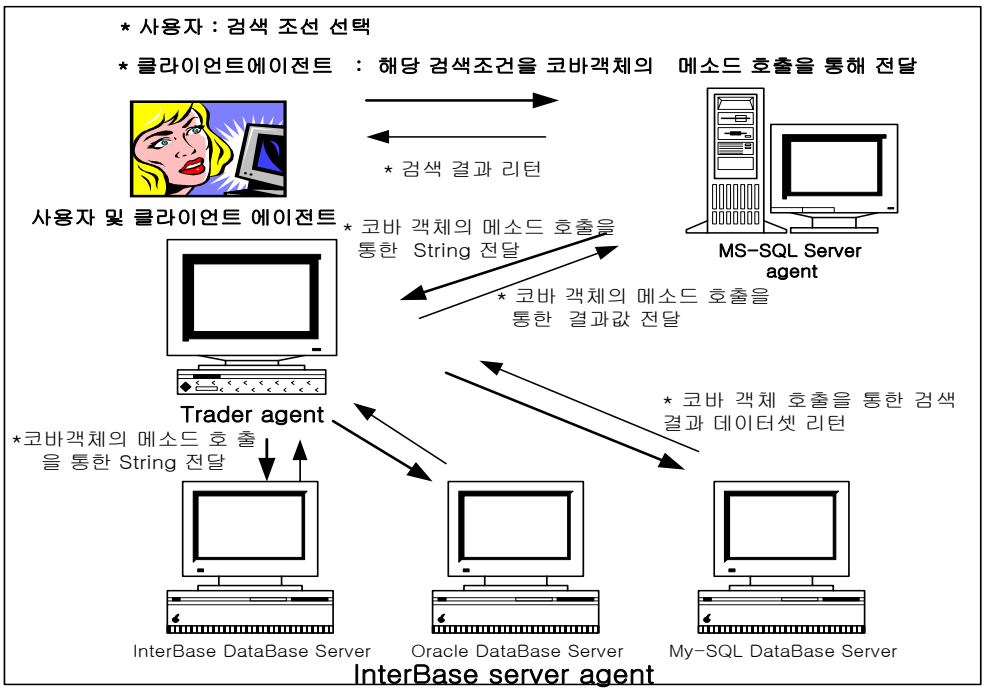

Fig. 4.1 Schematic Diagram of SECA

전체적인 SECA 구성은 Fig. 4.1과 같으며 클라이언트 에이전트에서는 MS-SQL 데이터베이스 서버로의 사용자등록, 선박등록이 이루어지며, 선박검색은 MS-SQL 서버 에이전트의 CORBA를 사용하였다. 클라이언트가 MS-SQL 데이터베이스 서버에 선박검색 질의를 했을 때 원하는 선박이 서버에 있을 경우에는 질의에 해당하는 레코드를 데이터셋으로 돌려주고, 검색 선박이 서버에 없을 경우에는 트레이드 에이전트에게 질의어를 프로토콜에 맞게 문자열로 변환하여 CORBA 객체의 메소드를 호출하며 VCL(Visual Component Libary)에서 데이터셋을 대표하는 것은 추상 클래스인 TDataSet이다.

트레이드 에이전트는 해당 질의어를 각 데이터베이스 서버에 보내고, 각 데이터베이스 서버는 문자열을 다시 질의어로 변환하여 지역 데이터베이스 서버에게 질의를 한 후 결과 데이터셋을 프로토콜에 맞게 문자열로 변환하여 CORBA 객체의 메소드를 호출하여 데이터를 전달한다. 여기서 트레이드 에이전트의 역할은 받은 메시지를 CORBA 객체에서 메소드를 호출하여 데이터를 전달한다. MS-SQL 서버 에이전트는 이미 클라이언트 에이전트에서 연락방법으로 선택한 전자우편 또는 화면 쪽지선택 방법에 따라 최종 사용자에게 검색 결과를 전달한다. 만약 검색 결과가 인터베이스 서버에도 없을 시 일정시간 간격으로 트레이드 에이전트는 선박검색을 수행한다. 자료구조 측면에서 볼 때, 내부의 테이블 스키마는 선종별로 각 서버에 따로 저장되어 있고, 고객이 검색할 전체 선박 정보로 선종명(Ship Type), 용선료(Charter hire) 등의 정보를 저장하고 있는 선박정보 저장 테이블을 가진다.

4.1.1 SECA 출현 배경

1) 정보기술의 발전

1980년대 이후 정보기술의 급속한 발달에 근거하여 많은 양의 데이터가 나타났으며 데이터웨어하우스(Datawarehouse)의 구축으로 원하는 정보를 얻기가 손쉬워졌다[104]. 학습 능력을 가진 프로그램의 등장으로 가공할 만한 정보처리 능력뿐만 아니라 정보통신 기술의 정보 유통기능과 연계하여 다량의 데이터를 실시간에 분석하고 이를 통해 경영전략을 수립할 수 있는 길이 열렸다.

2) 환경 변화

재래시장에서 인터넷 전자시장, 또는 가상사회가 출현하면서 새로운 패러다임에 기초한 가상기업(Virtual Corporation)으로 형성되고 있으며 이는 곧 '오프라인 기업'에서 '온라인 기업' 전이되고 있음을 의미한다.

3) 자금의 흐름의 가속화

정보화는 곧 세계를 하나의 지구촌으로 묶는 결과를 가져왔으며 급격한 사회 변화는 자금 흐름 역시 가속성을 가져왔다.

4) 분산 컴퓨팅 시스템의 발달

특히 네트워크 기술의 발달은 분산 컴퓨팅환경으로의 변화를 초래하였고, 이를 지원하기 위해 다양한 기술의 발달을 가져왔다. 인공지능 분야에서도 이러한 영향으로 인해 분산 인공지능분야가 1970년대 이후 발달하기 시작하여 최신 정보 시스템 구조는 분산객체 컴퓨팅으로 대변되고 있는 CORBA, JAVA, WWW등을 통합한 오브젝트 웹 기술이 기반이 된다. 결국 인터넷 기반으로 하여 가장 저렴한 비용

으로 최신 정보의 공유 및 분배를 실시간으로 함으로써 업무 능률을 극대화하는 것이다.

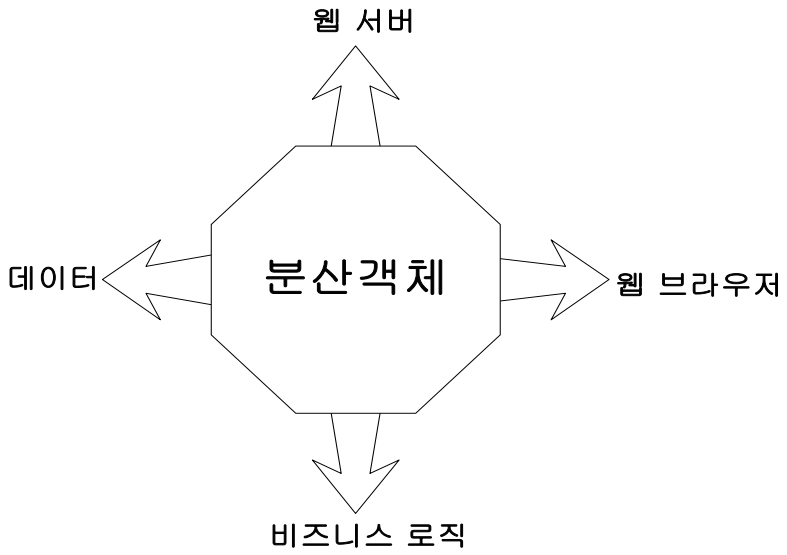

Fig. 4.2 Modular Configuration of Web and Distributed Objects

5) 인터넷의 발달

인터넷은 현재 우리 생활에 가장 많은 영향을 미치고 있는 분야 중 하나이며, 많은 프로그램이 웹을 중심으로 구축되고 있고, 기업들도 인터넷을 활용한 정보공유와 경제활동에 초점을 맞추고 있다. 해운기업에 있어서도 예외가 될 수 없으며 유선 및 무선통신(인공위성)을 통한 선박매매/용선중개에 대한 선박의 정보 공유를 가능하게 하였다.

4.1.2 사용자 인터페이스 설계

1) Log On 화면

Fig. 4.3 Execute SECA

Fig. 4.4 Log On

Fig. 4.4는 Log On 화면에서 회원이면 ID와 패스워드를 입력한 후 로그인 하고, 비회원인 경우 회원등록해야 하며 회원등록에 필요한 사항은 아래 Fig. 4.5와 같다.

회원으로 등록되면 등록버튼을 누르면 MS-SQL 데이터베이스 서버에 회원으로 등록되게 된다. Log On은 ID에 의해 등록된 선박에 대한 검색(전체 서버내의 등록된 데이터) 수정, 삭제, 등록 등이 모두 가능하다.

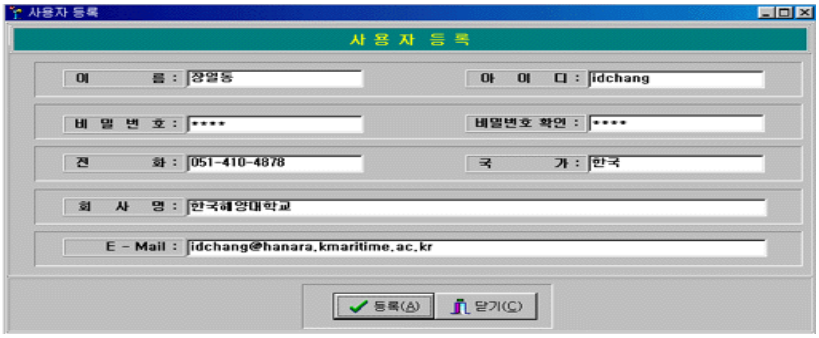

Fig. 4.5 Screen of Specification Contact Member's Registration

2) 메인화면과 선박용선등록

SECA 시스템의 사용자 인터페이스는 로그인에서 부터 검색조건은 선종(sType), 건조년도(MakeCountry), 발전기수(GeneratorNum), 연료형태(FuelType), 건조국가(MakeCountry), 최대속도(MaxSpeed), KW(KwInt), 용선상태, 건조회사(VesselCorp), 용선비용(Rental), 총톤수(GrossTon), 순톤수(NetTon), 용선기간(RentalGiFrom), 용선구분(GuBun), 나용선(BareboatCharter), 정기용선(TimeCharter), 항해용선(VoyageCharter)등에 모두 사용할 수 있는 통합환경을 지원하는 클라이언트 사용자 인터페이스를 구현한 화면이다.

SECA는 용선선박 등록정보 화면에서 사용자등록, 선박관리, 선박검색, 종료로 구분되며, 선박검색 에이전트에서는 입력, 수정, 삭제가 모두 가능하며, 용선선박 정보에 대한 검색조건은 AND(그리고) 개념으로 선택되어진다. (Fig. 4.6)

Fig. 4.6 Main Screen of SECA System

4.1.3 선박검색과 메시지 전송

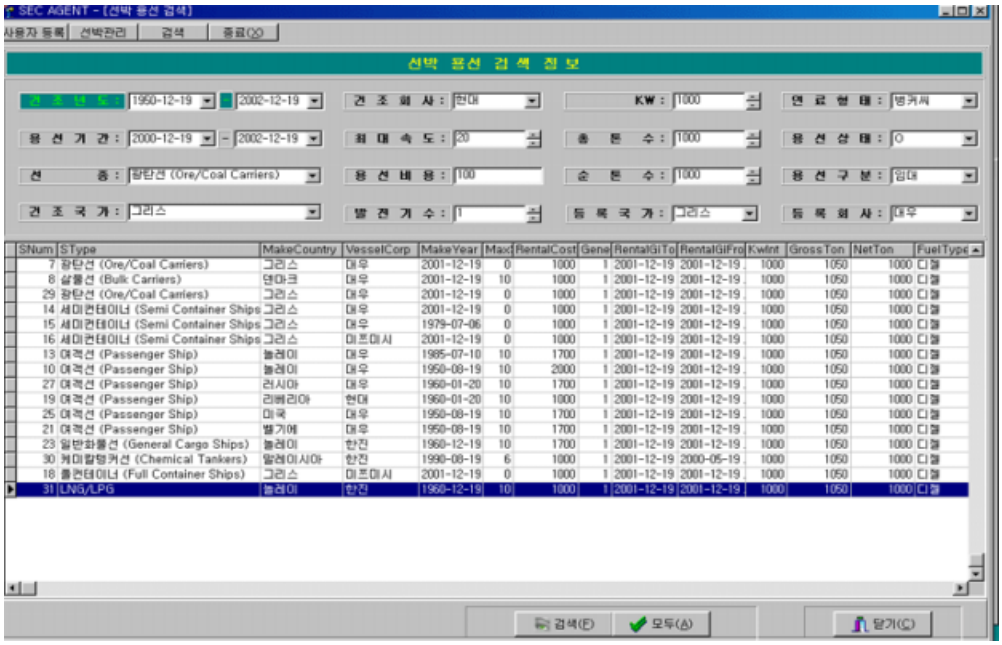

Fig. 4.7 The Visual Screen Displaying Result of Vessel Search

인터베이스에 질의를 위한 Makemessage 함수호출은 각 필드를 프로토콜에 의해 문자열로 변환하여 생성된 메시지를 서버 CORBA 객체의 GetMsg함수를 호출하여 전송한다. 검색된 결과는 CORBA 객체의 GetResult 메소드 호출을 통해 트레이드 에이전트로부터 전송된 메시지를 필드별로 DivideMsg가 분할하여 Tblsearch에 저장된다. 메시지 전송에 있어 검색된 결과과정은 인터베이스 서버 에이전트→트레이드 에이전트→MS-SQL 서버 에이전트→클라이언트 에이전트 순으로 전달된다.

```
unit uAgentReg;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, ExtCtrls, Buttons, IniFiles;
type
  TfAgentReg = class(TForm)
    Label1: TLabel;
    BOk: TBitBtn;
    BCancel: TBitBtn;
    RG1: TRadioGroup;
    Label2: TLabel;
    ...................
  var
  fAgentReg: TfAgentReg;
  BClikType : integer;
  IniFile   : TIniFile;
implementation
uses uMain, uFunction, uSearch, Server_TLB;
{$R *.DFM}
.....................
//클라이언트 에이전트에서 MS-SQL Server 에이전트와의 통신을 위한
MS-SQL Server 에이전트의 CORBA객체(IMSServer)생성
//및 MS-SQL Server 에이전트(IMSServer)의 GetMsg method 호출에 의한
데이터 전송
procedure TfAgentReg.BOkClick(Sender: TObject);
var
  F       : IMSServer;
begin
  if RG1.ItemIndex = -1 then
  begin
    ShowMessage('연락방법을 선택하지 않았습니다.');
    Exit;
  end;

F:=TMSServerCorbaFactory.CreateInstance('iIMSServerCorbaFactory');
  //서버한테 검색 data 전송
  F.GetMsg(OneRecordSet);
  MessageReceiveManner          := RG1.ItemIndex;
  fMain.tmrCheckSearch.Enabled  := True;
  Close;
end;
//클라이언트 에이전트와 MS-SQL Server 에이전트 CORBA객체와의 연결
procedure TfAgentReg.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
    Action        := caFree;
end;
end.
```

Fig. 4.8 Implementation Connecting Registration

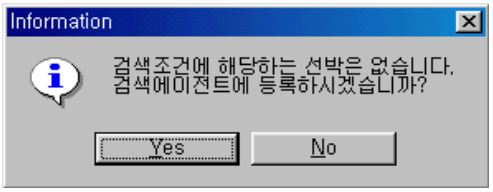

Fig. 4.9 Searching Agent Registration

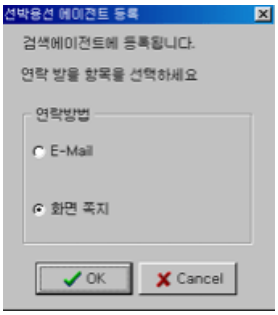

Fig. 4.10 Screen of Contacting

Fig. 4.10은 에이전트 등록방법에 있어 연락방법 중 화면쪽지로 선택하였을 경우이고, Fig. 4.11는 인터베이스 서버 에이전트로부터 받은 검색결과를 클라이언트 에이전트를 통해 사용자에게 화면쪽지로 전송된 결과이다. 화면쪽지로 전송시에는 제목과 날짜 시간정보가 나타나게 하였다.

Fig. 4.11 Screen of Memo

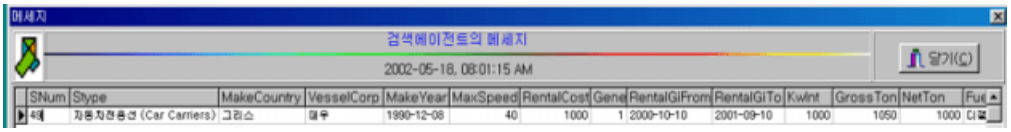

Fig. 4.12 Message Result Received by Opened Memo

Fig. 4.12는 화면쪽지로 배달된 정보는 그리스에서 건조된 대우회사가 자동차전용선을 검색할 수 있었다. 여기서 사용된 필드의 구분은 sNum, sType, MakeCountry, VesselCorp, MakeYear, MaxSpeed, RentalCost, RentalgiForm, RentalGito,

KwInt, GrossTon, NetTon, FuelType, Rental, CuNum, GuBun, CuName, CuTel, CuEMail, CuContry, CuCompany 등으로 구분되며, 고객 번호(CuNum)는 용선 등록당시 사용자의 고유번호로 인식하여 항상 갖고 다녀 등록자에게 전자우편이나 화면쪽지로 그 분리된 메시지를 사용자에게 전달된다.

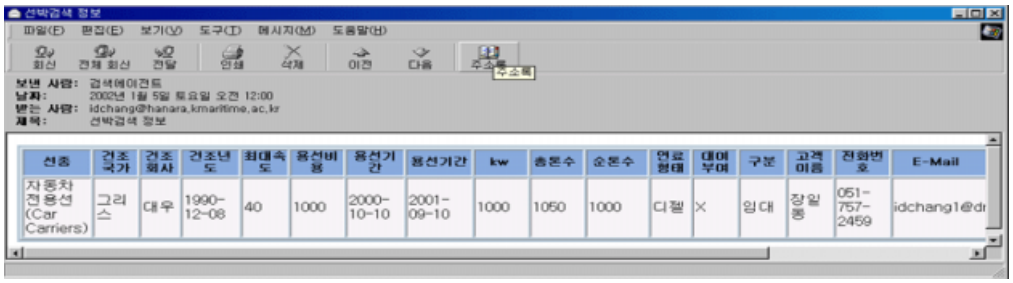

Fig. 4.13 Message Result Received by Opened E-mail

Fig. 4.13에서 연락방법을 전자우편을 선택하였을 경우, MS-SQL 서버 에이전트가 트레이드 에이전트로부터 받은 검색결과를 회원등록 시 등록된 전자우편주소로 전송된 결과를 열어본 결과이다.

4.2 SECA 분석 및 설계

먼저 자원 중개자(Resource Broker)의 역할을 살펴보면, 자원 중개자는 응용 프로그램으로부터 받은 추상적인 요청을 구체적인 요청으로 변환하는 일을 한다. 즉, 자원 중개자는 자원에 대한 정보를 요약하는 작업과 자원을 선택하는 작업등을 수행한다. 여기서는 자원의 선택업무만을 담당하는 각 에이전트를 둠으로써 각 서버에 올라와 있는 선박을 검색한다. 또한 3계층 구조의 클라이언트/서버 시스템 구현으로 기존 클라이언트/서버 시스템의 클라이언트의 기능을 축소하여 상대적으로 얇은 클라이언트층을 구성 하였다. 분산객체 구조가 인터페이스로 호출한 미리 정의된 루틴을 사용하여 상호 작용하도록 만든 컴포넌트 객체 모델을 말한다.

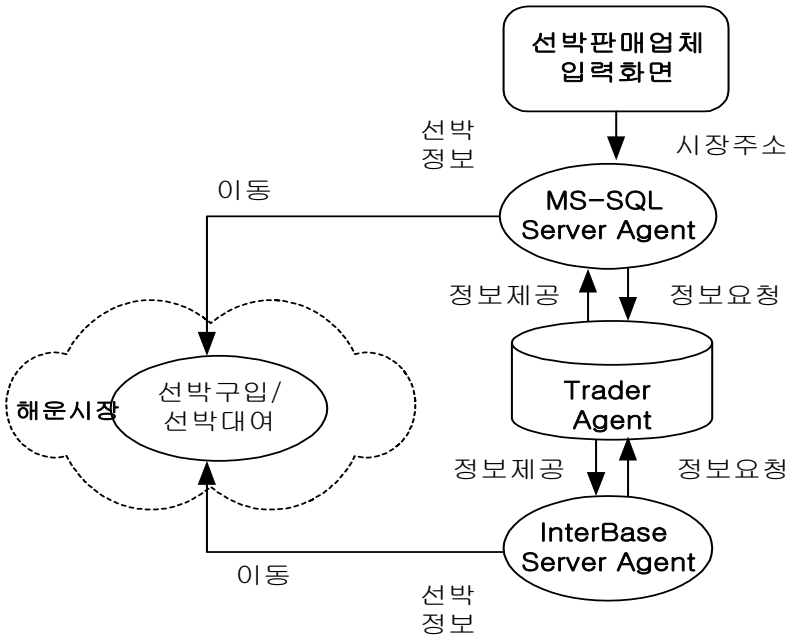

Fig. 4.14 The Process and Phase in SECA

대부분이 윈도우 기반인 클라이언트에 비해 다양한 운영체제로 구성된 서버쪽이 CORBA와 더불어 JAVA가 강점을 가질 수 있을 것이다. CORBA를 사용하는 경우에 분산 컴퓨팅을 위한 응용 프로그램 개발이 쉬워질 뿐만 아니라 시스템와 운용체계, 프로그래밍 언어와 무관하게 분산객체들간에 커뮤니케이션이 이루어진다. 클라이언트 응용 프로그램은 이동성 에이전트를 호출하여 각각 서버를 이동하면서 서비스를 요청하여 결과를 종합한 뒤 클라이언트의 플랫폼으로 가져와 결과를 전달한다[105]. Fig 4.14에서는 서버와 의미적 수준의 통신을 수행한다는 점과 서버 측으로 이동하므로 데이터가 서버에 상주할 수 있어 데이터의 무결성 보장이 더욱 용이하고, 네트워크 지연으로 인한 처리 속도의 지연도 방지할 수 있으므로 트랜잭션 처리 시스템을 보여주고 있다[106].

이 논문에서 제안하는 컴포넌트의 동적인 재구성 구조는 클라이언트와 어플리케이션 서버 사이에 투명한 네트워크를 통한 분산 3계층 구조로 이루어졌기 때문에 타 시스템보다 훨씬 성능이 뛰어나다고 할 수 있다[107].

4.2.1 시스템의 설계

일반적으로 인터넷/인트라넷 환경에서 다수의 컴퓨팅 자원을 공유하고 연동하기 위해서는 TCP/IP에 기반한 HTTP/CGI와 소켓(Socket) 프로그래밍, JAVA의 애플릿(Applet) 뿐만 아니라 ODBC, JDBC(JAVA Database Connectivity)[108], CORBA나 DCOM 등의 기술을 사용할 수 있다. 이 연구에서는 시스템과 연동하기 위해 클라이언트/서버 시스템에서 CORBA 기반 프로그래밍 기술과 ODBC 기술을 활용하여 시스템을 구성한다.

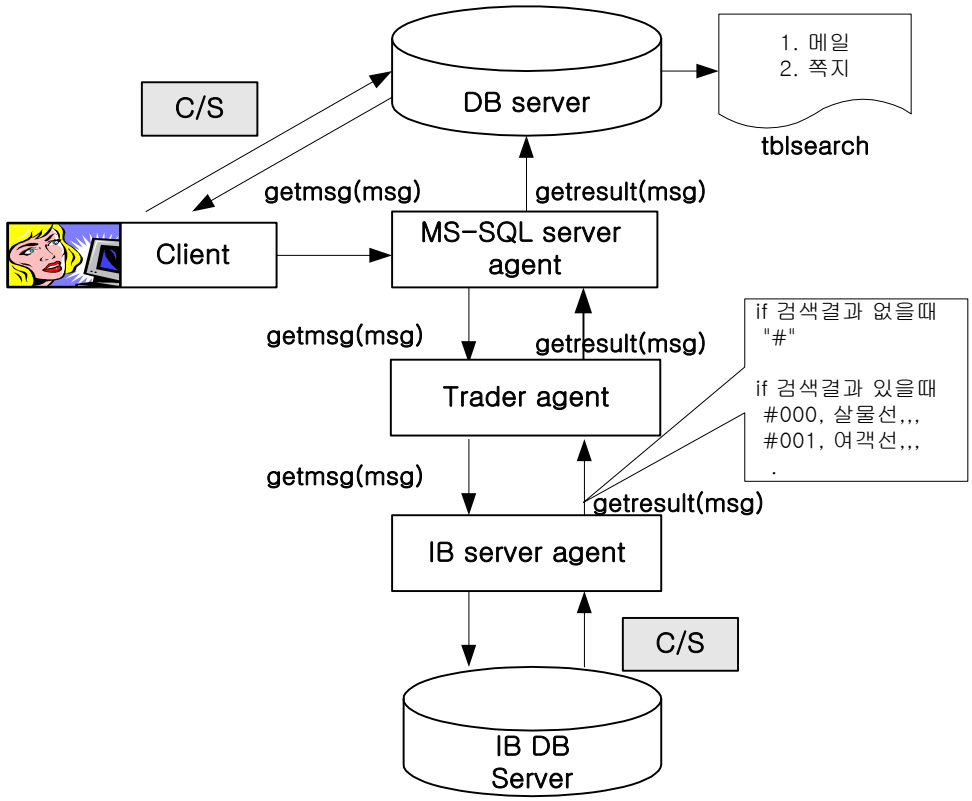

Fig. 4.15 Flow Chart for Implementation of the SECA System

Fig. 4.15에서 보는 바와 같이 SECA 시스템의 전체 흐름도를 보여주고 있다. 그 흐름과정을 살펴보면 먼저, 클라이언트가 데이터베이스 서버에 자료가 있는지를 검색한다. 만약 검색된 자료가 없으면 GetMsg로 MS-SQL 서버 에이전트에 자료검색을 요청한다. 따라서 이 논문에서는 분산 환경에서 클라이언트는 처리 내용이 커지면 전체 성능의 저하를 가져올 수 있기 때문에 경량(thin) 클라이언트로 개발되었다. SECA는 프론트 엔드를 델파이 언어로, 중간계층의 컴포넌트를 SECA 어플리케이션 서버로, 백 엔드를 데이터베이스로 하는 3계층으로 설계되었다. CORBA 서버 어플리케이션을 만들기 위한 과정은 아래 Fig. 4.16과 같다.

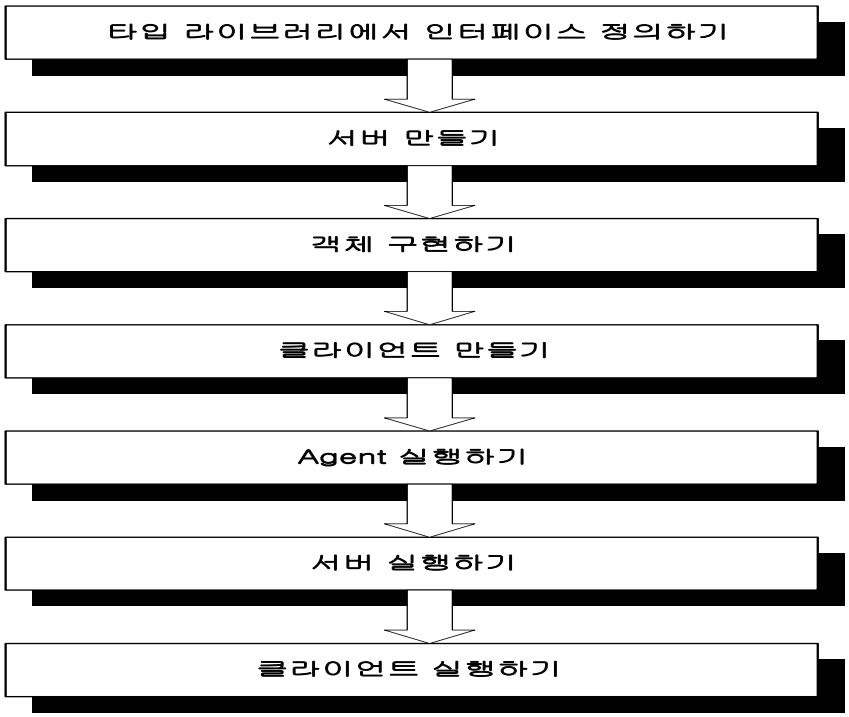

Fig. 4.16 CORBA Server Application Sequence Process

4.2.2 시스템의 개요

이 연구에서는 선박용선 및 매매를 가능하게 하는 CORBA 기반의 해운전자상거래 에이전트인 SECA를 분석, 설계 및 구현하였다. SECA 개발함에 있어 델파이5.0으로 개발된 이유는 SQL links를 통해서 오라클, 사이베이스, 인포믹스, 마이크로소프트 SQL 서버, DB2, 인터베이스 등과 같은 서버들에 대한 고유한 드라이버들을 제공하며, 그 외의 데이터베이스들을 위한 ODBC 연결도 제공되기 때문이다.

구현된 SECA 시스템의 다음과 같은 통신과정을 거쳐 수행하며 그 기능은 다음과 같다.

첫째, 클라이언트 시스템이 사용하는 하드웨어, 소프트웨어, 그리고 네트워크 플랫폼에 상관없이 다중 이질성 클라이언트로부터 데이터접근을 투명하게 한다.

둘째, 네트워크을 통한 클라이언트의 데이터베이스 서버에 대한 요구를 가능하게 한다.

셋째, 지역 서버에서 클라이언트 데이터 요구를 처리한다.

넷째, 네트워크를 통해 클라이언트에서 SQL 결과만을 보낸다.

SECA의 구현으로 사용자는 분산 환경에서 실시간으로 신속한 검색처리가 가능하며, 그 검색과정을 살펴보면 다음과 같다. 먼저 사용자가 검색조건을 선택하면, 클라이언트 에이전트는 해당 검색 조건을 선택하여 CORBA객체의 메소드 호출을 통해 MS-SQL 서버 에이전트에 전달하고, 전달된 메소드는 다시 CORBA 객체의 메소드 호출을 통해 트레이드 에이전트에 전달되고, 이는 다시 인터베이스 서버 에이전트로 전달된다. 인터베이스 서버 에이전트는 인터베이스 서버에 올려진 자료를 검색하여 검색조건에 맞는 자료를 CORBA 객체 호출을 통한 검색결과 데이터셋을 역순으로 사용자에게 검색된 결과를 E-mail 이나 쪽지 형태로 전달된다.

여기서 구현된 객체로 'Ship'이라는 클래스를 사용하였고, 그 Ship이라는 클래스에서 인터페이스 메소드는 GetMsg와 GetResult이며, 클라이언트 측 GetMsg()는 스터브를 통해 원격지의 스켈레톤으로 전달된다. 검색결과는 GetResult()라는 메시지로 검색된 결과 최종사용자에게 전달된다. 이는 SECA는 선박중개인을 대신하여 에이전트가 용선 또는 매매에 있어 원하는 선박정보를 검색하여 중개해주므로 사용자가 원하는 자료의 검색을 보다 신속히 처리가 가능하였다. 더욱이 클라이언트/서버 환경에서 개발되어 시스템의 성능 저하와 네트워크 병목현상을 동시에 해결하였다. 이제 전자상거래를 도입하려는 해운기업은 이러한 시스템의 도입으로 거래 당사자간 최적의 상대와 직접적 협상이 가능할 뿐 아니라 화물거래, 운임경매, 선용품구매, 선박용선, 선박매매, 선원공급, 컨테이너 관리 등의 업무도 효율적으로 수행할 수 있을 것이다.

4.2.3 시스템의 특징

인터넷은 21세기 산업발전의 원동력이며, 인류역사상 가장 빠른 속도로 확산된 미디어로써 현재 인터넷 비즈니스는 빠르게 확산되고 있으며, 특히 전자상거래는 더 이상 새로운 기술이 아닌 우리 환경의 필수적인 요소로 자리잡고 있다. 이러한 변화에 대응하기 위해 실시간으로 데이터 검색처리가 가능한 클라이언트/서버 환경에서 CORBA 기반 SECA 시스템에 관한 연구를 수행하게 되었다.

SECA 시스템의 주요 기술 분야는 최종 사용자 인터페이스의 편의성을 들 수 있으며, 데이터베이스 서버로는 인터베이스 관계형 데이터베이스로 개발되었다.

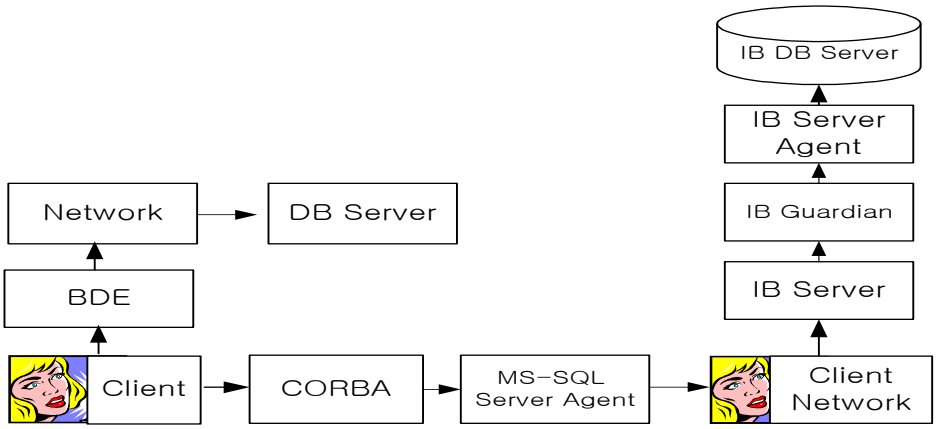

Fig. 4.17 Communication Process of Proposed System using CORBA

Fig. 4.17은 이 논문에서 구현되고 설계된 3계층 방식으로 데이터베이스에 접근하는 방법을 나타낸다[109]. 클라이언트는 BDE를 통해 데이터베이스 서버에 접근한다. 만약 원하는 정보를 얻지 못했으면 MS-SQL 서버 에이전트에게 메시지를 보내고, 인터베이스 서버를 실행시키기 위해 IB Guardian을 실행시키고 인터베이스 에이전트에게 인터베이스 데이터베이스를 검색을 요청한다.

이 연구에서는 InterBase 5.5로 네트워크 환경에서 사용할 수 있는 인프라이즈사에서 제공하는 관계형 데이터베이스로 BDE를 통한 관계형 데이터베이스 어플리케이션을 개발하였다. 물론 BDE를 거치지 않고 직접 인터베이스 서버에 접속해서 데이터를 처리할 수 있는 인터베이스 전용 컴포넌트를 포함하기도 한다. 구현 방법으로는 CORBA객체의 메소드 형식을 사용하였다. 내부언어로는 질의를 일정한 형식의 문자열로 변환하도록 구성하였으며 MS-SQL 서버 에이전트, 트레이드 에이전트, 인터베이스 서버 에이전트 구현으로 델파이 언어로 구현하여, 지역 데이터베이스는 MS-SQL과 원격 데이터베이스로는 인터베이스로 구현되었고, ORB는 Visibroker를 사용하였다. SECA 시스템의 특징은 객체지향 컴퓨팅 방식도입으로 다양한 플랫폼상의 데이터베이스에 투명하게 접근을 보장하며, 실시간 데이터 처리가 가능하며, 이종의 시스템에 프로그램을 분산시켜 시스템의 성능저하와 네트워크 병목현상을 동시에 해결하였고, 해운기업의 업무환경에 맞게 구현하였고, 시각적 대화식 방법으로 최종 사용자 인터페이스의 편의성과 신속성을 가졌다는 것이다.

4.3 SECA의 적용

이 절에서는 에이전트 전송 프로토콜 및 구현 객체들을 실행시키기 위한 MS-SQL 서버 측 모듈들로 스터브와 스켈레톤을 통한 SECA와 CORBA 객체와의 연결을 위한 클라이언트/서버 기반 에이전트 구현 방법을 제안한다. 연동하기 위해 HTTP[110], SMTP, CORBA 기술과 ODBC 기술을 활용하여 시스템을 구성될 수 있지만, 이 논문에서는 클라이언트/서버 환경하에서 미들웨어로 CORBA를 사용하여 구현하였다.

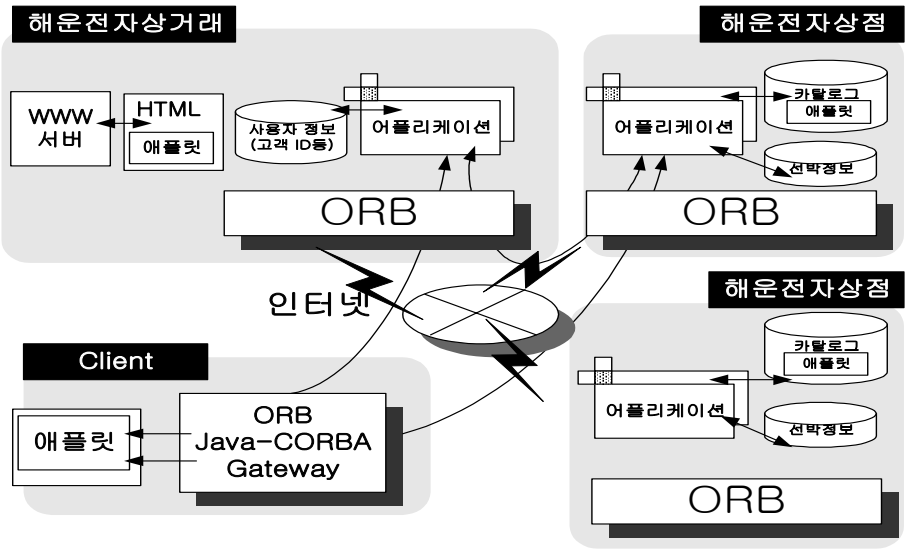

Fig. 4.18 Example of Building Conceptual Diagram for Shipping Electronic Commerce

Fig. 4.18에서 ORB를 사용해서 해운전자상점가 전체의 시스템을 구축하여 해운전자상점마다 독립된 시스템으로 구축할 수 있다. 각 해운전자상점의 내용변경은 자주 일어나지만 머신의 기종도 묻지 않는다. 또한, 각 해운전자상점의 내용변경은

자주 일어나지만, 해운저자상점 전체의 시스템을 멈추지 않아도 변경할 수 있기 때문에 해운전자상점가마다 언제라도 상품의 변경과 디스플레이를 바꿀 수 있다. 또한 해운전자상점은 백엔드로 기존의 시스템으로 연결될 수도 있고, 전자상점의 교체와 해운전자상점의 교체와 증설도 간단히 할 수 있다. 만약 JAVA로 어플리게이션을 작성하는 경우에는 ODBC는 클라이언트 서버형 데이터베이스 접근할 수 없고 JDBC-ODBC 브릿지를 이용해야 한다[111][112]. 따라서 CORBA 객체에 대한 초기화와 인터페이스에 대한 구현부를 실현시켜 주는 부분이며, 대부분의 객체 초기화와 팩토리 설정 사항은 자동적으로 코드가 만들어지며, 다만 타입 라이브러리에서 선언한 인터페이스에 대한 메소드 부분만 논리적으로 처리해 주면 된다. 다시 말해 CORBA를 사용하면 분산된 컴포넌트와 교류하기 위해 필요한 네트워크 코드를 프로그래머가 코딩할 필요는 없다.

구현된 SECA는 트레이드 에이전트와의 통신에서 CORBA객체의 메소드를 이용하며 프로그래밍 API를 통해서 정보를 전달하기 때문에 객체간의 결합도가 높다. 또한 원격지에서 메소드를 호출하지만 서버 에이전트의 입장에서는 로칼의 메소드를 호출하는 것처럼 사용할 수 있다. 앞에서 언급한 바와 같이 이러한 기능을 제공할 수 있는 기술은 CORBA, DCOM(COM)등이 있으나 이 논문에서는 어떤 플랫폼에서도 실행이 가능한 CORBA 기술을 이용하여 SECA를 구현하였기 때문에 분산된 데이터베이스에 저장된 선박 정보를 쉽게 검색할 수 있는 SECA 시스템의 구조 적용하였다.

4.3.1 SECA 시스템의 전체 구성

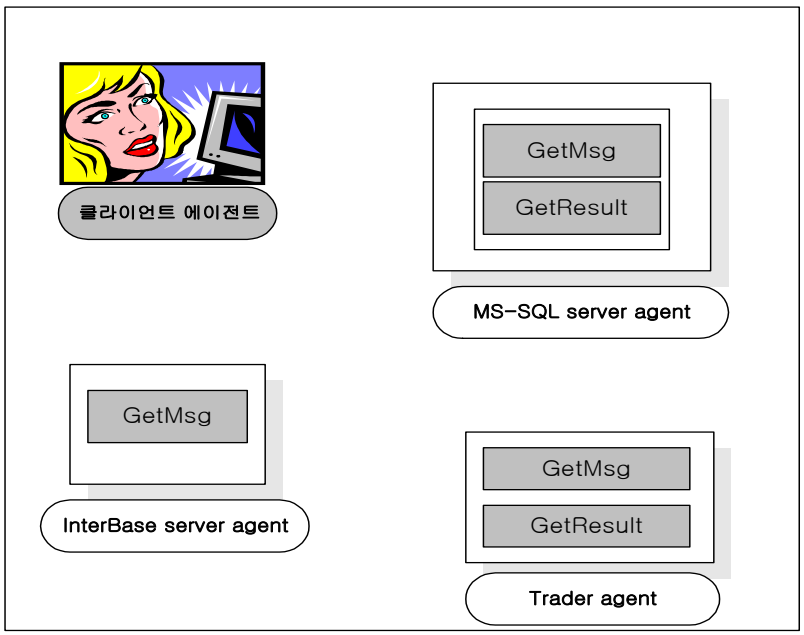

Fig. 4.19 Conceptual Diagram of SECA

Fig. 4.19는 시스템 구현을 위한 SECA의 메소드는 2개가 구성됨을 보이며, 첫째가 'GetMsg'이고 두번째가 'GetResult'이다. 이 두 메소드가 Trader Agent가 MS-SQL 서버 에이전트와 인터베이스 서버 에이전트를 사이를 오가면서 클라이언트 사용자에게 정보를 검색하여 준다.

이 논문에서 SECA를 실행시키기 위해서는 먼저 인터베이스 서버와 InterBase Guardian를 시작시키고, Smart Agent의 도움을 받아 VisiBroker 활성화 데몬(Activation Daemon)을 활성화로 모든 객체 구현을 추적이 가능하게 한 다음, 이 연구에서 구현된 MS-SQL 서버 에이전트, 트레이드 에이전트, 인터베이스 서버 에이전트를 실행시키고 마지막으로 Shipagent.exe를 실행시킨다.

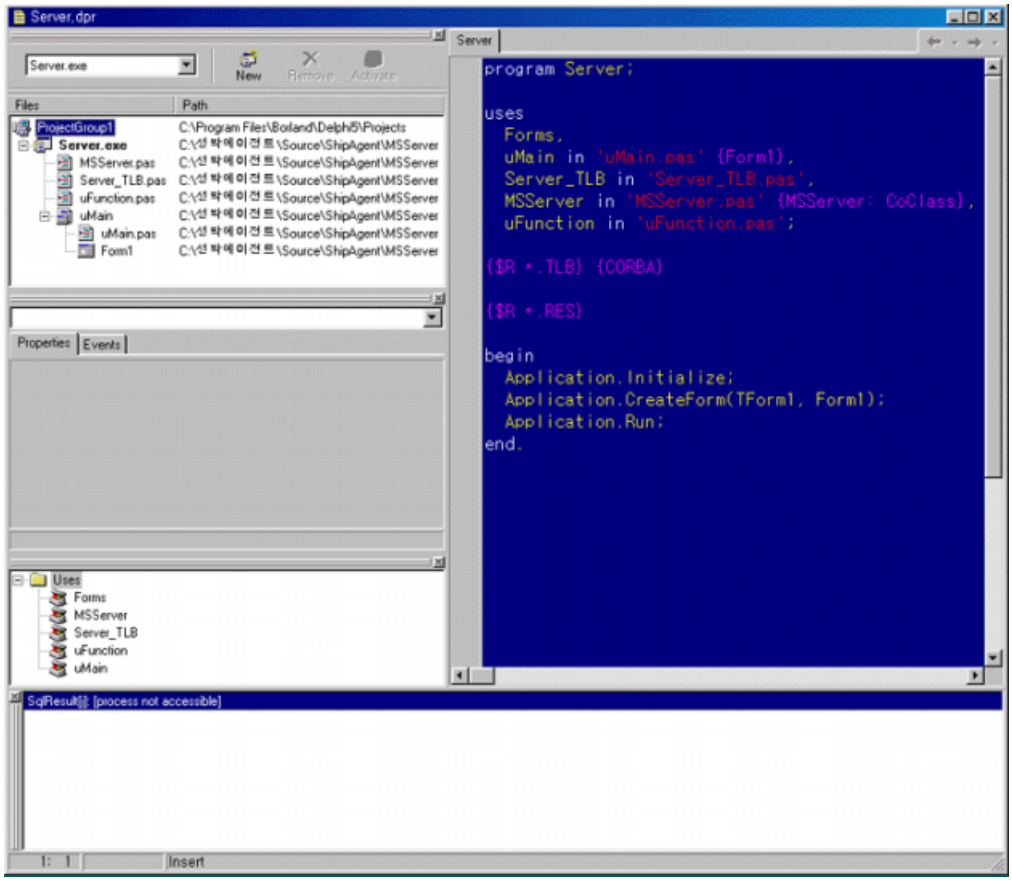

Fig. 4.20 Implementation of Server Programming

Fig. 4.20은 ProjectGroup1 프로젝트 파일인 타입라이브러리를 완성한 모습이다. 구현된 서버 프로그래밍으로 C:\선박에이전트\Source\ShipAgent\MSServer에 올려진 Server.dqr의 소스 코드이고, ProjectGroup1에 Server.exe\MSSever.pas, Server_TLB.pas, uFunction.pas, uMain\uMain.pas, Form1 타입라이브러리 편집기에 나타난 ProjectGroup1 프로젝트 타입라이브러리 정보이다.

```
unit uMain;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, ToolWin, ExtCtrls, StdCtrls, Psock, NMsmtp;
type
  TfMain = class(TForm)
    ........
  private      { Private declarations }
  public       { Public declarations }
  end;
var  fMain: TfMain;
implementation
uses  URegUser, uLogin, uFunction, uSell, uSearch, uDM, uMessage,
Server_TLB;
{$R *.DFM}
procedure TfMain.FormCreate(Sender: TObject);
begin
  Top          := 0;
  Left         := 0; end;
procedure TfMain.ToolButton3Click(Sender: TObject);
begin   Close; end;
.........
procedure TfMain.ToolButton4Click(Sender: TObject);
begin
  //열려있는 모든 자식 창을 닫는 함수
  AllChildClose;
  Application.CreateForm(TfSearch, fSearch);
  fSearch.Show; end;
procedure TfMain.tmrCheckSearchTimer(Sender: TObject);
var
  F        : IMSServer;
begin
  with fDM.qryCheck do
  begin
    Active          := False;
    SQL.Clear;
    SQL.Add('Select * from tblSearch');
    SQL.Add('WHERE (CuCustSearchNum = :pa1)');
    ..........
procedure TfMain.FormClose(Sender: TObject; var Action: TCloseAction);
begin
{   try
    if (F <> Nil) then  F._Release;
  ...........
procedure TfMain.Button2Click(Sender: TObject);
begin
  MessageReceiveManner       := 0;
  SendMessageToUser;
{     with NMSMTP1 do
    begin
      ..........
    with NMSMTP1.PostMessage.Body do
    begin        Clear;
      LoadFromFile('검색내용.htm');
    end;            //end with
    NMSMTP1.Connect;
    NMSMTP1.SendMail;
} end; end.
```

Fig. 4.21 Implementation uMain.pas

4.3.2 SECA의 전송 프로토콜

Fig. 4.22 Protocol of Record Transmission

• 선종(Ship Type) - 살물선(bulk carriers), 광탄선(ore/coal carriers), 자동차전용선(car carriers), 원목선(log carriers), 풀컨테이너(full container ships), 세미컨네이너(semi container ships), 일반화물선(general cargo ships), 핫 코일선(hot coil ships), 유조선(oil tankers), 케미칼탱커선(chemical tankers), LNG/LPG, 여객선(passenger ship) 등중 하나를 입력.

• 선박국적(Ship's nationality) - 한국, 미국, 일본, 영국, 프랑스, 독일, 러시아, 중국, 스페인, 브라질, 포루투칼, 그리스, 노르웨이, 덴마크, 벨기에, 스웨덴, 말레이시아, 싱가폴, 태국, 리베리아, 사이프러스, 파나마, 쿠웨이트, 이집트, 이탈리아,… 등중 하나를 입력.

• 조선소(Shipbuiling) - 현대, 삼성, 대우, 한진, 미쯔미시,… 등.

• 건조년도(Year) - 년 4자리 월 2자리로(1960/03) 범위검색을 위해서 몇년 몇월부터 몇년 몇월까지를 입력.

• 최대속도(Knots) - 최대속도(30) 입력.

• 용선료(Charter hire) - 1천만원일 경우에는 1을 입력.

• 발전기 갯수(Generater) - 2인 경우 2 입력.

• 용선기간(Charter period) - 30일인 경우 입력변수 30입력.

• kw - 3000일 경우 300입력.

• 총톤수(Gross ton) - 3000톤일 경우 3000입력.

• 순톤수(Net ton) - 2000톤일 경우 2000입력.

• 연료형태(Fuel) - 디젤, bunckerC 중 하나를 입력.

메세지를 보내기 위해 프로토콜로 SQL을 변환한 데이터이다.

Bulk carrier, korea, 14, 30, 2002-03, 3, 300, 100000, 50000, bunkerC, ...

4.3.3 에이전트와 CORBA 객체와의 연결

1) 트레이드 에이전트에서 인터베이스 CORBA 객체와의 연결

```
procedure TForm1.Timer2Timer(Sender: TObject);
var
    F : IIBCorba;
begin
  Form1.Timer2.Enabled        := False;
  F:=TIBCorbaCorbaFactory.CreateInstance('IIBCorbaCorbaFactory');
  Form1.Label1.Caption        := MsgStr;
  F.GetMsg(MsgStr);
end;
```

Fig. 4.23 Connecting InterBase Server CORBA Object from Trader Agent

Fig. 4.23는 보는 바와 같이 트레이드 에이전트와 CORBA 객체와의 연결의 구현으로 먼저 IIBCorba를 객체로서 선언하고, TIBCorbaCorbaFactory.CreateInstance

부분으로 생성을 하며 GetMsg(MsgStr)부분으로 연결을 한다.

여기서 CreateInstance로 전달되는 인스턴스는 'IIBCorbaCorbaFactory'와 같이 정해주었다. 트레이드 에이전트에서 인터베이스 서버 에이전트와의 통신을 위해 CORBA 객체(IIBCorba)생성 및 CORBA 객체(IIBCorba)의 GetMsg 메시지 호출에 의해 연결됨을 보여준다.

2) 트레이드 에이전트에서 MS-SQL 서버 에이전트 CORBA 객체와 연결

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
F : IMSServer;
begin
  Timer1.Enabled:= False;
F:=TMSServerCorbaFactory.CreateInstance('IMSServerCorbaFactory');
  Form1.Label2.Caption:= MsgStr;
  F.GetResult(MsgStr);
end;
```

Fig. 4.24 Connecting MS-SQL Server CORBA Object from Trader Agent

위 객체 구현을 위해 크게 선언 부분으로 IMSServer를 객체로서 선언을 하고, TMSServerCorbaFactory.CreateInstance 부분으로 생성을 하며 GetResult (MsgStr)부분으로 연결을 한다. 여기서 CreateInstance로 전달되는 파라미터의 값을 가져와야 하며, 인스턴스는 'IMSServerCorbaFactory'와 같이 정해주면 된다. MS-SQL 서버 에이전트는 먼저 F를 선언하고 IMSServer에서 Data를 전달하기 위한 메소드인 GetResult를 호출한다. (Fig. 4.24)

제 5 장 SECA 시스템 구현

이 논문에서는 분산객체시스템의 표준인 CORBA에 초점을 맞추어 SECA를 구현하였다. SECA는 다른 인터베이스 서버와의 원활한 통신을 위하여 에이전트를 둠으로써, 각 서버들은 서로의 위치나 운영체제, 데이터베이스의 종류에 관계없이 통신에만 신경을 쓰면 된다.

에이전트 통신 언어는 전달하고자 하는 지식을 담는 내부언어와 메시지 통신을 위한 외부언어로 구성되며, 외부언어는 CORBA의 객체 메소드 호출을 사용하고, 내부언어는 SQL의 파라미터들을 문자열로 변환하여 사용하였다. 한편, 사용자로 하여금 시간을 절약하도록 선박이 없을 경우에 서버에 선박 정보를 등록하게되면 서버가 다른 인터베이스 서버에 접속하여 검색을 실시하거나 지역 서버에 해당 선박이 등록되면 사용자의 전자우편이나 클라이언트 화면이 쪽지 메시지로 데이터를 전송하게 하였다. 만약 그러한 에이전트의 역할이 아니라면 선박중개에서 다수의 분산된 서버가 존재하며 지역적인 서버의 데이터베이스 내에서만 검색하고 검색한 선박이 없을 경우에 사용자가 임의의 시간에 재 접속하여 검색한다면 매우 번거러운 일이 될 것이다.

이제 SECA의 등장으로 사용자 검색 편의성, 시간의 효과성을 기대할 수 있으며, 각 데이터베이스 서버들 간의 통신을 통해 데이터를 공유함으로써 보다 많은 데이터를 대상으로 검색을 할 수 있는 시스템 통합의 유효성을 보였다. 이제 단순히 그림이나 텍스트 정보가 아닌 메타정보로 변화가 일어나고 있으며 그러한 변화는 결국 SECA의 역할이 증대되고 있음을 의미한다.

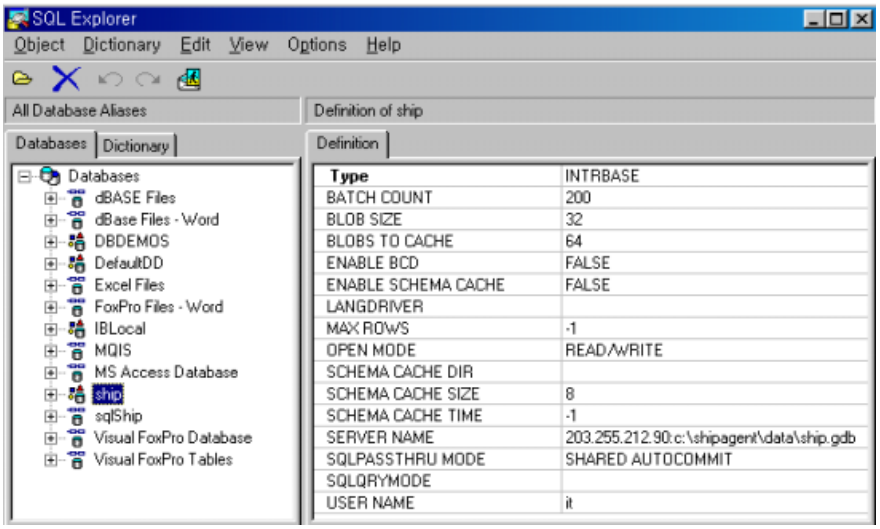

Fig. 5.1 Database Information

SQL Explorer에서의 구성은 Fig. 5.1과 같으며 인터베이스 서버 위치는 203.255.212.90에 올려져 있다.

먼저 데이터베이스에 접근하기 위해서는 로그인과 패스워드를 입력해야 접근이 가능하며 구성으로는 첫째, 두 대의 데이터베이스 서버 중 어느 곳으로 정보가 보내어져야할 것인가를 결정하는 루틴과 둘째, IDL에서 정의하여 구현된 여러 가지 구현개체 호출들을 정의하는 부분 등으로 구성된다. 구현객체를 호출하는 부분을 다시 순차적으로 살펴보면 먼저 연결하고자 하는 데이터베이스 서버에서 접속하기 위한 객체 호출, 선박을 선택하여 데이터베이스 서버에 저장시키기 위한 객체 호출, 검색결과를 보기 위해 화면상에 나타나야할 개인 ID와 선박정보를 되돌려 받기 위한 객체 호출, 연결된 데이터베이스 서버와의 접속을 중단하기 위한 객체 호출 순으로 구성된다.

```
program ShipAgent;
uses
  Forms,
  uMain in 'uMain.pas' {fMain},
  uLogin in 'uLogin.pas' {fLogin},
  uFunction in 'uFunction.pas',
  URegUser in 'URegUser.pas' {RegUser},
  uDM in 'uDM.pas' {fDM: TDataModule},
  Splash_frm in 'Splash_frm.pas' {Splash},
  uSell in 'uSell.pas' {fSell},
  uSearch in 'uSearch.pas' {fSearch},
  uAgentReg in 'uAgentReg.pas' {fAgentReg},
  uMessage in 'uMessage.pas' {fMessage};
{$R *.RES}
begin
  Application.Initialize;
  Splash := Tsplash.Create(Application);
  splash.Show;
  splash.Update;
  while Splash.Timer1.Enabled do
    application.ProcessMessages;
  splash.Hide;
  splash.Free;
  Application.CreateForm(TfDM, fDM);
  Application.CreateForm(TfMain, fMain);
  Application.Run;
end.
```

Fig. 5.2 ShipAgent.exe

Fig. 5.2는 클라이언트 에이전트에서 구현된 ShipAgent.dpr 소스 부분이다. uMain은 클라이언트 에이전트 메인 폼이며, uLogin은 ID와 패스워드 입력, uFunction은 함수 모음, uRegUser은 사용자 등록, fDM은 데이터 모듈과 sql관련, Splash화면, fSell은 선박용역등록, fSearch 용선검색, fAgentReg는 에이전트 화면, fMessage는 e-mail와 Memo 관련 폼이다.

5.1 클라이언트 에이전트의 구현

이 논문에서 제안한 CORBA 기반의 분산 환경 지원이 가능한 SECA 개발을 위하여 플랫폼에 독립적인 CORBA 개발도구로 Visigenic사의 Visibroker[113]를 사용하였다. 이 Visibroker은 라이브러리가 포함되어 있고 Borland의 JAVA 서비스를 제공하는 프로그램으로 구성되어 있다.

인터넷상에 설치되어 있는 각종 방화벽(Firewall)로 인하여 CORBA 객체에 접근할 수 없는 경우도 터너링 기능을 제공하여 접근이 가능하도록 해준다[114]. 이러한 이유 때문에 분산 환경을 구축하는데 있어 미들웨어의 도입과 활용은 가장 중요한 결정사항 중의 하나라고 할 수 있다. 해운전자상거래 시장에 적용하여 분산된 환경에서 에이전트 통신 언어와 분산객체인 CORBA를 이용하여 분산된 시스템에서의 정보의 전달과 공유를 지원할 수 있는 에이전트 기반 시스템을 제시하고, 서로간의 정보를 공유하고 교환하는 해운시장 에이전트 시스템을 구현하였다. ID와 패스워드를 입력하고 난 후, 그 다음 단계로 넘어가지 않고 모형의 간소화를 위해 바로 화면상에 제한적으로 나열되어 있는 각 서버에 올라와 있는 선박을 선택한다.

클라이언트 객체가 서버 객체에게 서비스를 요구하기 위해서는 먼저 서버 객체에 대한 식별자를 얻어야 한다. 즉, 클라이언트 어플리케이션은 사용자가 서버 쪽의 데이터를 조작하는 데 필요한 인터페이스에 제공한다. 사용자가 인터페이스를 통해 명령한 서비스를 서버쪽에 요청하는 것이 클라이언트의 첫 번째 임무이며 클라이언트는 사용자의 지시에 따라 요청을 만들어서 서버에게 보내고 서버가 제공한 서비스를 적절히 사용자에게 보여준다.

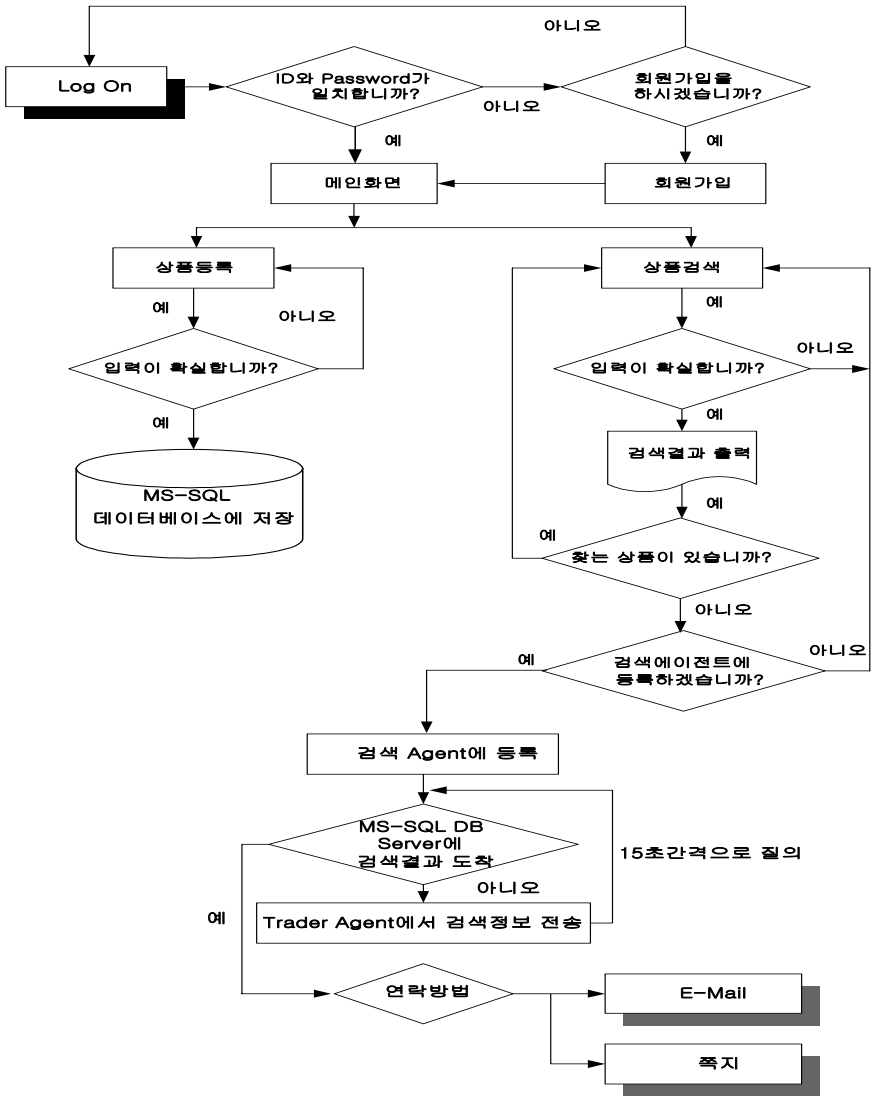

Fig. 5.3 Flow Chart of Client Agent

사용자 확인을 위해 사용될 ID와 패스워드 입력 시 등록된 고객의 ID는 해당 고객이 정확하게 입력한다고 가정하고 패스워드만 올바른지 확인한다. (Fig. 5.3)

그리고 마지막으로 선박 선택은 제한된 품목별 선박에서 콤보박스(ComboBox)를 이용하여 임의로 선택할 수 있다. 네트워크를 통해 분산되어 있는 모든 서버 객체는 고유한 식별자를 갖고 있는데 이를 해당 객체의 객체참조라 하며 객체참조는 서버 객체가 동작하는 호스트의 네트워크 주소, 서버 이름, 객체 이름과 인터페이스 이름 등으로 구성되는 구조체이다.

5.2 MS-SQL 서버 에이전트의 구현

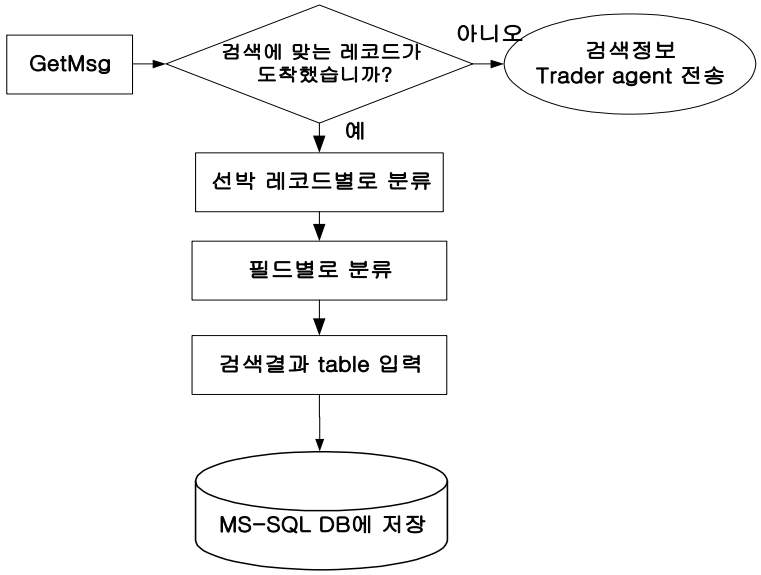

Fig. 5.4 Data Flow of MS-SQL Server Agent

Fig. 5.4는 MS-SQL 서버 에이전트 구현을 위한 GetMsg 메시지의 호출이 분류되어 MS-SQL 데이터베이스에 저장되는 과정은 다음과 같다.

(1) GetMsg 메소드

선종	,	건조국가	,	최대속대	,	임대비용	,	임대기간	,	발전기수	,	KW	,	총톤수	,	순톤수	,	연료형태	,	대여구분	,	임대구분	,	등록국가	,	등록회사	,

Fig. 5.5 Data Fields of Registration Ship

선박 데이블에 추가한 레코드를 SECA와의 프로토콜에 맞도록 문자열을 생성하여 SECA의 MS-SQL 서버 에이전트 CORBA 객체의 GetMsg 메소드를 호출하여

전송하며 MS-SQL 서버 에이전트와의 프로토콜은 다음과 같다. (Fig. 5.5)

등록된 선박이 검색되었으면 검색결과 변수에 저장되어 있는 데이터를 이용하여 사용자가 연락방법으로 선택한 매체(전자우편 또는 쪽지)로 검색 결과를 전송하고, 검색된 결과가 없으면 MS-SQL 서버 에이전트에 매 15초 간격으로 질의한다. 만약 사용자의 연락방법이 쪽지일 경우 새로운 창이 활성화된다.

(2) GetResult 메소드

SECA에서 검색된 선박 레코드를 MS-SQL 서버 에이전트로 전송하기 위한 메소드이다. 원하는 선박이 검색되면, 선박레코드별로 분류하여 MS-SQL 데이터베이스의 Tblsearch 표에 저장된다.

혹 등록된 선박을 검색하지 못하면, 다시 일정시간 후 검색정보를 클라이언트 에이전트로 다시 보내고 그 메시지를 트레이드 에이전트로 보낸다.

```
procedure TForm1.Timer2Timer(Sender: TObject);
var
    F       : IMed1;
    Msg     : String;
begin
  Timer2.Enabled       := False;
  F    := TMed1CorbaFactory.CreateInstance('iMed1CorbaFactory');

  F.GetResult(SearchResultStr);
end;
procedure TfMain.FormCreate(Sender:TObject);
begin
   SqlStr:=TStringList.Create;
end;
```

Fig. 5.6 Information of Calling Trader Agent by InterBase Server agent

첫 번째 procedure 부분은 인터베이스 서버 에이전트에서 트레이드 에이전트와의 통신을 위한 트레이드 에이전트의 CORBA객체(IMED1)생성 및 트레이드 에이전트(IMED1)의 GetResult 메소드 호출에 의한 데이터 전송이고, 두 번째에 나타나

는 procedure는 인터베이스 서버 에이전트와 트레이드 에이전트 객체와의 연결 구현부분이다. (Fig. 5.6)

```
unit MSServer;
interface
uses
  Windows, Messages, SysUtils, Classes, Grahpics, Controls, ComObj, StdVcl,
CorbaObj, Server_TLB;
type
  TMSServer = class(TCorbaImplementation, IMSServer)
  private
    { Private declarations }
  public
    { Public declarations }
  protected
//클라이언트 에이전트에서 MS-server 에이전트 메시지 호출
    function GetMsg(const msg; WideString): WideString; safecall;
//IBase 서버 에이전트로부터 온 메시지 결과
    function getresult(const msg; WideString): WideString; safecall;
  end;
implementation
uses CorbInit, uFunction, uMain;
function TMSServer.GetMsg(const msg; WideString): WideString;
begin
  MsgStr                    := Msg;
  Form1.Timer2.Enabled      :=True;
  Result                    :=msg;
end;
function TMSServer.getresult(const msg; WideString): WideString;
begin
  Form1.Timer3.Enabled      :=True;
  MsgStr                    :=Msg;
end;
initialization
TCorbaObjectFactory.Create('MSServerFactory','MSServer','IDL:Server/
MSServerFactory:1.0',IMSServer,TMSServer,iMultiInstance,tmSingleThread);
end.
```

Fig. 5.7 Implementation of MS-SQL Server Agent

Fig. 5.7은 구현된 MS-SQL 서버 에이전트에 대한 소스부분이다. 여기서 특정한 서버와 관련된 CORBA 객체를 인스턴싱화해 주기 위해서는 TCorbarObjectFactory라는 클래스를 생성해 주어야 한다.

TCorbaObjectFactory.Create에서 MSServerFactory는 인터페이스명이고, MSServer는 인턴스 이름이고, 'IDL:Server/MSServerFactory:1.0'은 저장소 ID이고, IMSServer

는 구현 GUID이고, TMSServer는 구현 클래스이고, iMultiInstance 인스턴싱이고, tmSingleThread는 스레드 모델이다.

MSServerFactory라는 인터페이스 파라미터는 팩토리의 인터페이스명을 알려준다. 그러나 여기서 객체를 생성해 주는 팩토리에 대한 인터페이스를 위치시켜 주기 위해 특정한 팩토리 인스턴스 이름(InstanceName)파라미터는 클라이언트 어플리케이션에 의해 사용되는 인스턴스 명칭을 명시준다.

문자열은 'IDL:Server/MSServerFactory:1.0'과 같은 형태로 표현되며, ImplGUID를 구현해 주는 클래스를 명시해 주는 것으로 실제 구현 클래스인 반면, 대응되는 스켈레톤 클래스는 아니다. 인스턴싱 파라미터는 팩토리가 모든 CORBA 클라이언트가 공유하는 하나의 객체 인스턴스를 생성해 주는지 알려줄뿐만 아니라 각 CORBA 클라이언트가 분리된 인스턴스를 생성할 것인지 결정해 주는 역할을 수행한다.

```
procedure TForm1.Timer3Timer(Sender: TObject);
begin
  Timer3.Enabled      := False;
  Label2.Caption      := MsgStr;
  DivideMsg;
InsertRecordTo;
end;
```

Fig. 5.8 Implementation of Connecting MS-SQL Server Agent and Trader Agent

검색된 결과는 SQL 서버 CORBA 객체의 GetResult 메소드 호출을 통해 트레이드 에이전트로부터 전송된 메시지를 필드별로 DivideMsg가 분할하여 Tblsearch에 저장됨을 보여준다. (Fig. 5.8)

```
function TMSServerStub.GetMsg(const msg: WideString): WideString;
  var
    OutBuf: IMarshalOutBuffer;
    InBuf: IMarshalInBuffer;
  begin
    FStub.CreateRequest('GetMsg', True, OutBuf);
    OutBuf.PutWideText(PWideChar(Pointer(msg)));
    FStub.Invoke(OutBuf, InBuf);
    Result := UnmarshalWideText(InBuf);
  end;
  function TMSServerStub.GetResult(const msg: WideString): WideString;
  var
    OutBuf: IMarshalOutBuffer;
    InBuf: IMarshalInBuffer;
  begin
    FStub.CreateRequest('GetResult', True, OutBuf);
    OutBuf.PutWideText(PWideChar(Pointer(msg)));
    FStub.Invoke(OutBuf, InBuf);
    Result := UnmarshalWideText(InBuf);
  end;
```

Fig. 5.9 Implementation of MS-SQL Server Stub Module

Fig. 5.9는 구현된 MS-SQL 서버 스터브 모듈로서 GetMsg()의 스터브에 의해서 호출됨을 보여준다.

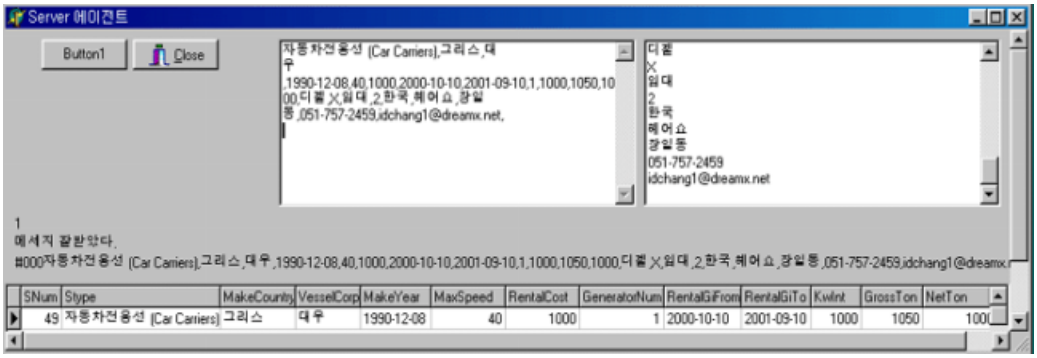

Fig. 5.10 MS-SQL Server Agent Specification

Fig. 5.10은 인터베이스 서버 에이전트로부터 받아온 GetResult의 결과는 다시 트레이드 에이전트를 거쳐 MS-SQL 서버 에이전트로 전달된 결과를 보여주고 있으며, 여기서 MS-SQL 서버 에이전트는 DivideMsg에 의해 분할되어 전달된 필드 명세를 보여주고 있다.

```
  procedure TMSServerSkeleton.GetMsg(const InBuf: IMarshalInBuffer; Cookie:
Pointer);
  var
    OutBuf: IMarshalOutBuffer;
    Retval: Wide스트링;
    msg: Wide스트링;
  begin
    msg := UnmarshalWideText(InBuf);
    Retval := FIntf.GetMsg(msg);
    FSkeleton.GetReplyBuffer(Cookie, OutBuf);
    OutBuf.PutWideText(PWideChar(Pointer(Retval)));
  end;
  procedure TMSServerSkeleton.GetResult(const InBuf: IMarshalInBuffer;
Cookie: Pointer);
  var
    OutBuf: IMarshalOutBuffer;
    Retval: Wide스트링;
    msg: Wide스트링;
  begin
    msg := UnmarshalWideText(InBuf);
    Retval := FIntf.GetResult(msg);
    FSkeleton.GetReplyBuffer(Cookie, OutBuf);
    OutBuf.PutWideText(PWideChar(Pointer(Retval)));
  end;
```

Fig. 5.11 Implementation of MS-SQL Server Agent Skeleton Module

Fig. 5.11은 구현된 MS-SQL 서버 스켈레톤의 모듈로서 포장된 정보는 GetMsg()의 스켈레톤에 의해 포장이 풀려지고(Unmarshall)호출된다. 처리된 GetMsg()의 결과는 다시 네트워크를 통해 클라이언트 스터브에게 포장되어 전달된다. 클라이언트 쪽의 GetMsg() 스터브는 스켈레톤을 통해 넘겨받은 결과의 포장을 풀고 해당 결과값을 처리한다.

5.3 트레이드 에이전트의 구현

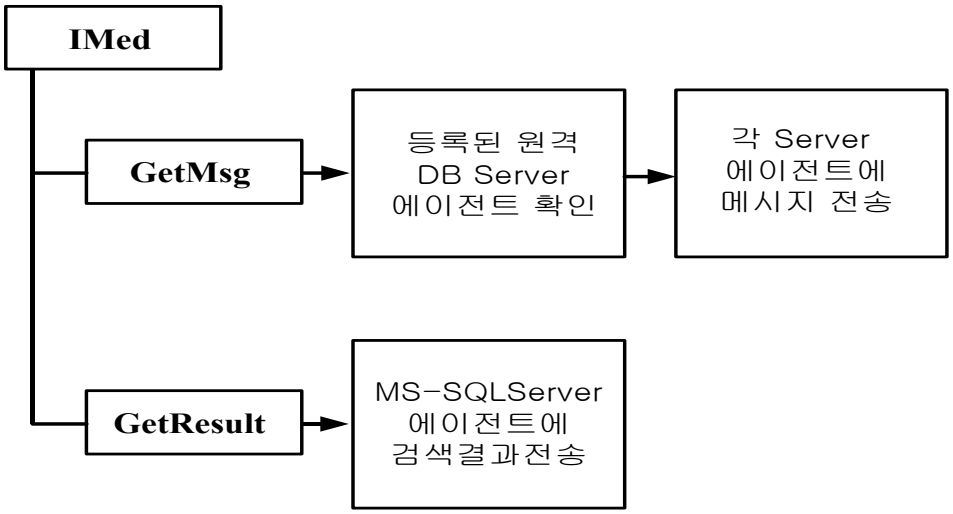

Fig. 5.12 Schematic Diagram of Trader Agent

```
{ TMed1Stub }
function TMed1Stub.GetMsg(const msg: WideString): WideString;
var
  OutBuf: IMarshalOutBuffer;
  InBuf: IMarshalInBuffer;
begin
  FStub.CreateRequest('GetMsg', True, OutBuf);
  OutBuf.PutWideText(PWideChar(Pointer(msg)));
  FStub.Invoke(OutBuf, InBuf);
  Result := UnmarshalWideText(InBuf);
end;
function TMed1Stub.GetResult(const msg: WideString): WideString;
var
  OutBuf: IMarshalOutBuffer;
  InBuf: IMarshalInBuffer;
begin
  FStub.CreateRequest('GetResult', True, OutBuf);
  OutBuf.PutWideText(PWideChar(Pointer(msg)));
  FStub.Invoke(OutBuf, InBuf);
  Result := UnmarshalWideText(InBuf);
end;
```

Fig. 5.13 Implementation of Trader Stub Module

Fig. 5.12는 전체 구성으로는 CORBA 객체는 IMed이고, 메소드는 GetMsg, GetResult가 있음을 보여주며 Fig. 5.13은 구현객체를 실현시키기 위한 트레이드 스터브 모듈부분이다. 즉, 객체 서비스에 대한 정적인 인터페이스를 제공하는 부분이며 이 클라이언트 쪽의 스터브는 보통 IDL 전위 처리기를 거치면 생성되는데 이 스터브를 통하면 클라이언트 쪽에서 원격 메소드 호출을 마치 지역 메소드를 호출하는 것처럼 프로그램밍 하였다.

```
{ TMed1Skeleton }
constructor TMed1Skeleton.Create(const InstanceName: String;
const Impl: IUnknown);
begin
  inherited;
  inherited InitSkeleton('Med1', InstanceName, 'IDL:Med/IMed1:1.0',
tmMultiThreaded, True);
  FIntf := Impl as IMed1;
end;
procedure TMed1Skeleton.GetImplementation(out Impl: IUnknown);
begin
  Impl := FIntf;
end;
procedure TMed1Skeleton.GetMsg(const InBuf: IMarshalInBuffer; Cookie:
Pointer);
.....
end;
procedure TMed1Skeleton.GetResult(const InBuf: IMarshalInBuffer;
Cookie: Pointer);
var
  OutBuf: IMarshalOutBuffer;
  Retval: WideString;
  msg: WideString;
begin
  msg := UnmarshalWideText(InBuf);
  Retval := FIntf.GetResult(msg);
  FSkeleton.GetReplyBuffer(Cookie, OutBuf);
  OutBuf.PutWideText(PWideChar(Pointer(Retval)));
end;
```

Fig. 5.14 Implementation of Trader Agent Skeleton Module

Fig. 5.14는 트레이드 스켈레톤 모듈의 구현이며, 이 논문에서 구현된 SECA 과정은 MS-SQL 서버 에이전트는 검색 선박이 지역 데이터베이스 서버에 없을 경우

트레이드 에이전트의 GetMsg 메소드 호출을 통해서 검색선박에 대한 조건을 프로토콜에 맞게 문자열로 변환해서 트레이드 에이전트에게 전달하게 되며, 전송된 메시지를 트레이드 에이전트는 등록된 인터베이스 서버 에이전트로 전달한다.

```
[   uuid(EF70757D-66BC-4AE6-AA1C-A7F1941BDBEF),
    version(1.0),
    helpString("Med Library")]
library Med
{
    importlib("STDOLE2.TLB");
    importlib("STDVCL40.DLL");
    [       uuid(D919261B-4E5E-46DE-A2D2-27347545F08D),
        version(1.0),
        helpString("Dispatch interface for Med1 Object"),
        dual,
        oleautomation   ]
     interface IMed1: IDispatch
    {       [id(0x00000001)]
        HRESULT _stdcall GetMsg([in] BSTR msg, [out, retval] BSTR *
Result );
        [id(0x00000002)]
        HRESULT _stdcall GetResult([in] BSTR msg, [out, retval] BSTR *
Result );
    };
    [   uuid(B8F5DD49-9110-4E73-85FD-33856CDB993F),
        version(1.0),
        helpString("Med1 Object") ]
```

Fig. 5.15 Implementation of Trader Agent IDL

Fig. 5.15는 구현된 트레이드 IDL의 예이며 타입 라이브러리인 인터페이스는 IMed1이며, 이 인터페이스의 메소드는 GetMsg와 GetResult이며 타입 라이브러리의 BSTR 타입의 경우 문자열 변환되게 된다. 즉, CORBA 객체에 대한 초기화와 인터페이스에 대한 구현부를 실현시켜는 부분이며, 대부분의 객체 초기화와 팩토리 설정 사항은 자동적으로 코드가 만들어진다.

5.4 인터베이스 서버의 구성 및 구현

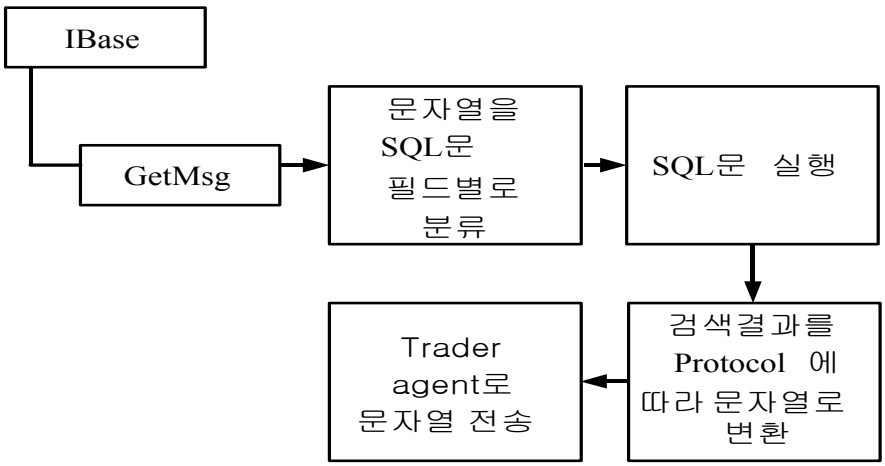

Fig. 5.16 Schematic Diagram of InterBase Server Agent

인터베이스 서버 전체 구성도로 인터베이스 서버 에이전트에 사용되고 있는 CORBA 객체는 IBase이고, 메소드는 GetMsg가 있다. (Fig. 5.16)

인터베이스 서버 에이전트에 검색된 선박들은 트레이드 에이전트의 GetResult 메소드를 호출로써 CORBA객체의 메소드를 호출하여 전송 받은 선박 검색 정보를 프로토콜에 맞게 문자열로 변환해서 MS-SQL 서버 에이전트에게 전달한다.

인터베이스 서버에서 트레이드 에이전트로부터 전송 받은 메시지를 디코딩하여 데이터베이스에 질의하여 해당되는 데이터가 있을 경우에 같은 프로토콜로 CORBA객체 호출을 통하여 검색된 레코드들을 검색한다.

구현된 인터베이스 서버 에이전트는 Fig 5.17과 구현되며 프로젝트 파일이름과 같은 이름의 리소스파일(.RES)을 프로젝트에 링크함을 보여준다.

```
program IBServer;
uses
  Forms,
  uFunction in 'uFunction.pas',
  IBServer_TLB in 'IBServer_TLB.pas',
  uibCorba in 'uibCorba.pas' {IBCorba: CoClass},
  uMain in 'uMain.pas' {fMain};
{$R *.TLB} {CORBA}
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TfMain, fMain);
  Application.Run;
end.
```

Fig. 5.17 Implementation of InterBase Server Agent

Fig. 5.17은 인터베이스 서버 에이전트의 구현 소스이다.

Fig. 5.18 The Visual Screen Displaying Result of Vessel Search

Fig. 5.18에서는 선박검색결과 MS-SQL 데이터베이스 서버에 등록되어 있지 않지만 인터베이스 서버에는 자동차전용선 등록되어있음을 알 수 있다.

인터베이스 서버 에이전트는 인터베이스 서버에 올라와 있는 입력된 메시지를 필드별로 분류하여 검색된 결과를 다시 트레이드 에이전트로 전송하는 역할을 담당한다.

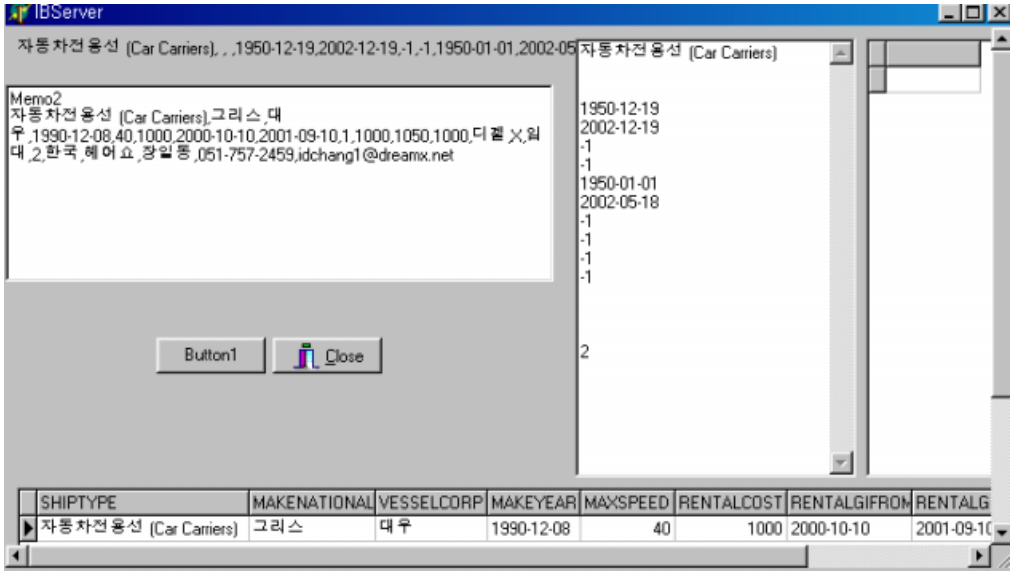

Fig. 5.19 The Visual Screen Displaying Detailed Information

Fig. 5.19는 인터베이스 서버에 올라와 있는 자동차전용선을 찾아옴을 알 수 있다. 검색된 데이터 표시에 있어 레코드 번호는 1부터 검색 선박의 일련번호를 나타내며 #000~#999까지 검색이 가능하고, 한 레코드가 끝나면 다시 #999의 선박일련번호와 한 레코드의 데이터가 연결된다. 만약 GetResult(Msg)가 없을 경우에는 #을 붙여서 데이터의 끝임을 알린다.

5.5 시스템의 전체 동작과정 구현

이 연구에서 구현된 SECA는 분산된 많은 데이터베이스 서버에 접근과정과 선박의 용선 및 매매에 있어서 사용자가 원하는 신속한 정보가 어떤 과정을 거쳐 얻어지는지를 보였다. 이러한 과정이 이루어지기 위해서는 먼저 시스템의 통합과 협동적인 업무의 수행에 있어 각 전산 시스템간의 통신기능, 협의기능, 그리고 정보를 교환 공유할 수 있는 기능까지 포함한다.

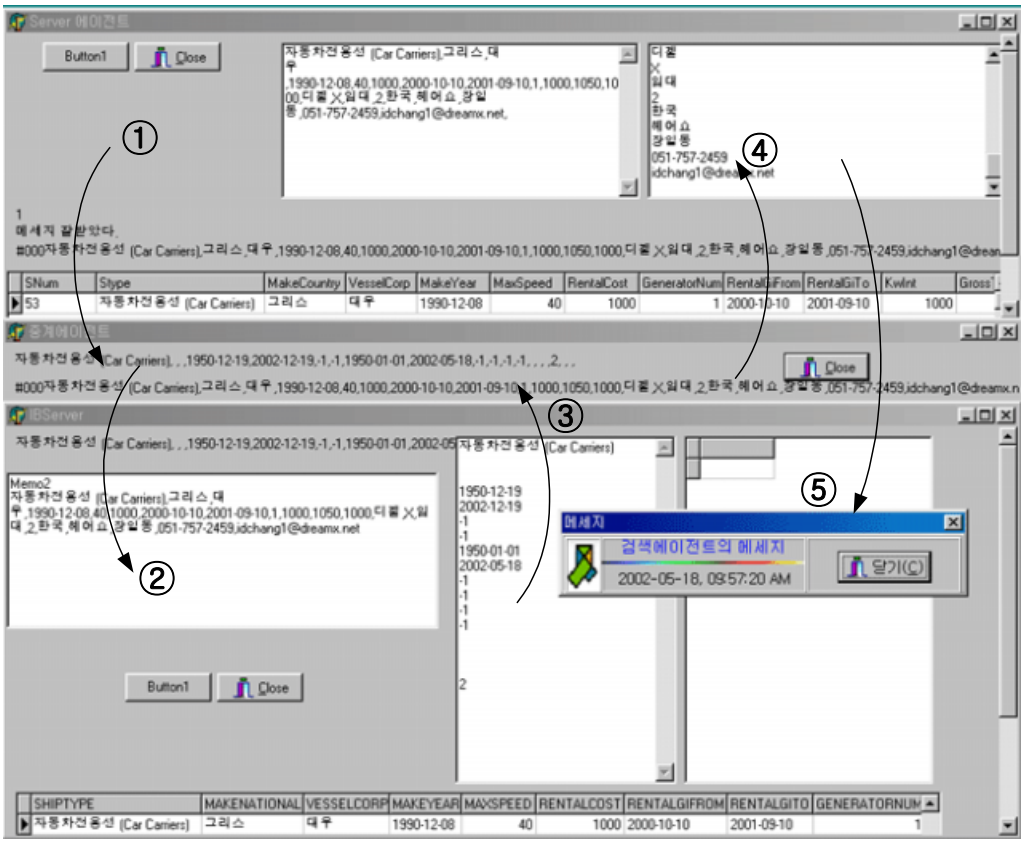

Fig. 5.20 Operation Procedure of SECA System

Fig. 5.20는 구현된 전체시스템의 동작이며, 클라이언트가 필요로 하는 작업을 위해 각 에이전트 서버에 보내어 수행한다. 먼저 클라이언트 에이전트가 선박검색을 시작하고, MS-SQL 서버 에이전트에 검색을 요청한다.

① MS-SQL 서버 에이전트에서 트레이드 에이전트로의 데이터 전송

MS-SQL 서버 에이전트는 자체 RDBMS에 질의한 후 검색 선박이 있을 경우에는 검색결과를 클라이언트 에이전트에게로 전달해준다. 만약 검색 선박이 지역 데이터베이스 서버에 없을 경우에 트레이드 에이전트에게 CORBA 객체의 메소드 호출을 통해서 검색 선박에 대한 조건을 프로토콜에 맞게 문자열로 변환해서 트레이드 에이전트에게 전달한다. 클라이언트가 요청하는 선박정보는 클라이언트의 선박정보 시스템을 통해 브라우징(Browsing)할 수 있다.

② 트레이드 에이전트에서 인터베이스 서버 에이전트로 데이터 전송

Fig. 5.20에서 가운데 그림이 트레이드 에이전트로 그림상단에 표시된 검색을 자동차 전용선을 인터베이스 서버에 올려진 선박을 찾는다. 반환 전에 인터베이스 서버에 올라온 자료를 검색된 결과는 CORBA 객체의 GetResult 메소드 호출을 통해 트레이드 에이전트로부터 전송된 메시지를 필드별로 DivideMsg가 분할하여 Tblsearch에 저장된다.

③ 인터베이스 서버 에이전트에서 트레이드 에이전트로 검색결과 반환

인터베이스 서버에 올라온 자동차전용선을 찾아 트레이드 에이전트로 그 결과값을 반환한다. 검색 결과로는 자동차 전용선(Car Carriers), 조선소로 그리스, 건조회사 대우, 건조년도 1990.12.8, 최대속도 40 Knot, 용선료 1000, 용선기간 2000/10/10이후 등으로 검색되었다.

④ 트레이드 에이전트에서 MS-SQL 서버 에이전트로의 데이터 전송

트레이드 에이전트로부터 전송된 메시지를 MS-SQL 서버 에이전트로 Fig. 5.20과 같은 프로토콜로 CORBA객체 호출을 통하여 검색된 레코드들을 반환한다. 트레이드 에이전트는 웹서버와 응용 프로그램 서버로 구성된다. 트레이드 에이전트의 클라이언트는 선박이 되며 클라이언트가 요청하는 선박정보를 가공하여 제공하는 역할을 한다. 클라이언트 어플리케이션에는 우선 클라이언트가 사용할 데이터 모듈이 필요하며 또한 어플리케이션 서버의 인터베이스 데이터 모듈에 연결할 수 있는 수단이 있어야 있어야 하며 서버 GUID는 자동적으로 {EEF5596F-DFE5-4B49-ACE3-316C6F016F01}가 선택되어 나타났다. 즉, 레지스트리에 등록된 모든 객체 클래스는 128비트의 GUID에 의해 각기 고유하게 구별된다.

⑤ 검색된 결과를 전송

SECA 시스템은 검색된 GetResult(Msg) 메시지 결과를 등록 시 전달방법선택에 있어 선택한 전자우편 또는 쪽지 형식으로 최종 사용자에게 검색 결과를 전송한다.

5.6 시스템 구현의 고찰

오늘날 인터넷을 이용한 전자상거래가 기존의 기업활동 방식과 거래수단, 대상 및 범위를 변화시키고 있으며, 이러한 변화는 상당한 상거래 비용의 절감과 가상공간에서의 새로운 사업 기회를 제공하게 될 것이다. 최근 해운산업에서도 해운전자상거래 서비스업체의 수가 증가하며 기존의 오프라인 해운거래가 온라인에서 이루어질 수 있도록 다양한 서비스를 제공하고 있다. 특히 해운산업은 국제적인 성향이 강한 산업이면서도 대단히 보수적인 특징을 가지고 있다. 그러나 경제환경의 세계화와 정보통신의 급속한 발달, 그리고 인터넷 사용의 폭발적인 증가로 인하여 앞으로 해운기업 분야에서도 해운전자상거래가 보편화될 것으로 예상되며, 이러한 환경의 변화에 적응하지 못하는 해운기업은 결국 경쟁우위를 상실하게 될 것이다[115]. 이 연구에서는 선박용선 및 매매 등의 중개를 가능하게 하는 CORBA 기반의 해운전자상거래 에이전트인 SECA를 설계하여 구현하였다. SECA는 클라이언트 에이전트, MS-SQL 서버 에이전트, 트레이드 에이전트, 그리고 인터베이스 서버 에이전트로 구성된다. SECA는 CORBA기반으로 구현되었기 때문에 다양한 플랫폼 상의 해운전자상거래 관련 데이터베이스에 대한 데이터의 공유가 가능하고, 통신 투명성 및 상호운용성이 보장된다. 클라이언트/서버 환경하에서 SECA를 구성하여 이종 시스템들 간에 프로그램을 분산시켜 부하를 줄임으로써 시스템의 성능 저하와 네트워크 병목현상을 동시에 해결할 수 있었다. 또한 사용자의 편의성을 고려하여 시각적 질의형식으로 구현된 SECA는 사용자가 원하는 정보를 편리하고 신속하게 검색할 수 있게 해 준다.

이러한 SECA가 해운전자상거래의 실무에 적용될 경우, 거래의사결정자는 효율적인 정보교환과 의사소통을 통하여 다양한 대안의 검토와 문제해결에 창의적인

아이디어를 발휘할 수 있는 기회를 갖게 될 것이다. 더욱이 SECA는 사이버 공간에서 정보네트워크 기반으로 양질의 해운전자상거래 서비스를 제공하고 관리할 수 있게 해줌으로 기업간 거래 및 업무의 효율화를 가져다 준다. 또한 중간 수수료 없이 거래 당사자간 최적의 상대와 직접적 협상이 가능할 뿐 아니라 화물거래, 운임경매, 선용품구매, 선박용선, 선박매매, 선원공급, 컨테이너 관리 등의 업무도 효율적으로 수행할 수 있게 해준다.

향후 해운산업 분야에서도 해운 전자상거래에 관한 법적·제도적 여건이 성숙되고 해운기업의 보수적인 특성을 극복하여 해운 전자상거래가 일반화된다면, 이 연구에서 다룬 해운전자상거래 에이전트 구현 기반 기술은 우리 나라 해운산업의 국제 경쟁력 제고에 다소간 기여할 수 있으리라 생각된다.

제 6 장 결 론

최근 UN이 '전자(電子) 정부' 구현 수준을 평가한 결과 우리 나라가 '우수 국가군'으로 평가받아 행정 전산화 수준이 가장 높은 국가군으로 분류되었다[116]. 이는 우리 나라가 공공 부문의 정보이용지표 면에서 상당히 높은 수준의 정보화를 이루었다는 것을 보여 준다. 이 외에 정보 하부구조의 확산 정도를 나타내는 정보설비지표나 정보화 추진을 위한 능력을 나타내는 정보화지원지표 등의 관점에서도 우리 나라의 정보화 수준은 비교적 높은 것으로 나타난다[117].

한편, 국제적인 해운기업 경영환경은 국제물류서비스 시장에서의 전략적 제휴와 미국 신해운법(1999) 발효 등으로 해운기업간의 경쟁이 심화되고 있다. 우리 나라의 해운력은 세계 7위권이며 해양수산부는 국가 해운력 세계 5위권 진입이라는 목표를 설정하고 있다. 국가적으로 비교적 높은 정보화 수준과 해운력을 보유하고 있음에도 불구하고 우리나라 해운기업 분야의 전자상거래의 이용정도는 매우 낮은 실정이며, 현재의 해운전자상거래는 대부분 자사의 웹사이트를 중심으로 주로 선박운항일정 제공, 화물선적의뢰, 화물부킹, 화물선적, 화물추적, 자사 홍보 등의 컨텐츠를 제공하는데 머물러 있다[118]. 이러한 현실의 가장 큰 원인으로는 해운기업의 보수적 특성을 들 수 있을 것이다[119]. 그러나 무한 가상 공간에서 이루어지는 전자상거래의 거대한 조류를 감안할 때, 해운기업 분야에서도 기업 경쟁력의 제고를 위해 전자상거래 시스템의 도입은 불가피할 것으로 예견된다.

이 논문은 전술한 해운전자상거래의 현실이 연구배경이 되어 선박매매 및 용선 등을 중개하는 해운전자상거래 에이전트인 SECA를 설계·구현하는 주제를 다루었다. SECA는 클라이언트/서버 환경하에서, 정보기술 분야의 분산처리기술과 더불어, 네트워크 연결 및 관리에 대한 투명성, 다양한 시스템 호환성, 언어 독립성, 객체지향

기술 등의 장점을 가진 미들웨어인 CORBA를 기반으로 구현되었다. 이 연구에서 구현한 SECA는 클라이언트 에이전트, MS-SQL 서버 에이전트, 트레이드 에이전트, 그리고 인터베이스 서버 에이전트로 구성되어 있다. 그리고 클라이언트/서버 환경하에서 SECA를 구성하는 에이전트의 역할에 따라 구현된 프로그램들을 각각의 에이전트에 분산시킴으로써 시스템의 성능 저하와 네트워크 병목현상을 동시에 해결할 수 있었다.

마지막으로 이 연구의 성과를 정리하면 다음과 같다.

첫째, 대부분 자사의 웹사이트를 중심으로 선박운항일정 제공, 화물선적, 화물추적, 자사홍보 등의 컨텐츠를 제공하는데 그치고 있는 해운전자상거래의 현실에서, 국내 처음으로 선박매매 및 용선 등을 중개하는 CORBA 기반의 해운전자상거래 에이전트 SECA의 설계 및 구현을 다루었다.

둘째, SECA는 클라이언트/서버 환경 하에서 구성 에이전트의 역할에 따라 구현된 프로그램들을 각각의 에이전트에 분산시킴으로써 시스템의 성능 저하와 네트워크 병목현상을 동시에 해결할 수 있었다.

셋째, 사용자의 편의성을 고려하여 시각적 질의형식으로 구현된 SECA는 사용자가 원하는 정보를 편리하고 신속하게 검색할 수 있게 하였다.

이러한 해운전자상거래 에이전트의 설계 및 구현에 관한 연구 결과는 앞으로 해운산업 분야에서도 해운 전자상거래에 관한 법적·제도적 여건이 성숙되고 해운기업의 보수적인 특성을 극복하여 해운 전자상거래가 일반화되면, 해운전자상거래의 실현을 위한 기반 기술에 유용하게 활용되리라 사료되며, 향후에는 웹상의 SECA의 운용에 관한 연구가 필요할 것이다.

참고문헌

[1] Vinoski, S., "CORBA:Intergration Diverse Applications Withn Distributed Heterogeneous Environments", IEEE Communications Magazine, vol. 14, Feb 1997.

[2] George Couloursi, Jean Dollimore and Tim Kindberg, Distributed Systems: Concepts and Design. 2nd Edition, Addison Wesley, 1994.

[3] OMG, "The Common Object Request Broker Architecture and Specification : Revision 2.0", OMG Document 1995.

[4] Zhonghua Yang, Keith Duddy, "Distributed Object Computing with CORBA", DSTC Technical Report 23, 1995.

[5] Mani Chandy, K., Adam Rikfin, Systematic Composition of Objects : Processes and Sessions, Oxford University Press Computer Journal, Volume 40, Number 8, pp. 465-478, October 1997.

[6] Mani Chandy, K., Joseph Kiniry, Adam Rifkin, and Daniel Zimmerman, A Framework for Structured Distributed Object Computing, Parallel Computing, Volume 24, Number 12-13, pp. 1901-1922, November 1998.

[7] David Perkins, Evan McGinnis, "SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 : William Stalling's, 3th, Addison Wesley, 1999.

[8] Mary, E. S. and Loomis, "Object Database : The Essentials", Addison-Wesley Publishing Company, 1995.

[9] Lopez. et. al., Using CORBA and Globus to Coordinate Multidisciplinary Aersocience Applications., Proceedings of the NASA HPCC/CAS Workshop, Feb. 15-17, 2000.

[10] Evaggelia Pitoura, Omran Bukhres, Ahmed Elmagarmid, "Object

Orientation in MultiDatabase System", ACM Computing Surveys, June 1995.

[11] Brown, N. and Kindel. C., Distributed Component Object Model Protocal-DCOM/1.0 Internet Draft, 1996.

[12] Vogel, Duddy, "JAVA programming with CORBA", John Wiley & Sons, INC,. 1998.

[13] Suresh Scridhar, Decision support using the Intranet, *Decision Support System*, 23, p. 19-28, 1998.

[14] Cortese, A., Here comes the Intranet, Business Week, February 26, 76-84, 1996.

[15] Lim, E. P. and Ng, W. K., "An Overview of the Agent-Based Electronic Commerce System(ABECOS) Project", Bulletin of the Technical Committee on Data Engineering, Vol. 23, No. 1, Mar 2000.

[16] Yan, G. H., Ng, W. K. and Lim, E. P., "Toolkits for a Distributed, Agent-Based Web Commerce System", In International IFIP Working Conference on Trends in Distributed System for Electronic Commerce(TrEC '98), Hamburg, Germany, June 1998.

[17] Robert Orfali, Dan Harkey, Jeri Edwards, "Instant CORBA", John Wiley & Sons, INC, 1997.

[18] Kiniry, J. and Zimmerman, D., "A Hands-On Look at JAVA Mobile Agent," IEEE Internet Computing, July/Aug., 1997.

[19] Roure, D. D. et. al., "Agents for Distributed Multimedia Information Management," Proc. of 1st Int'l Conf. on the Practical Applications of Intelligent Agents and Multi-Agent Technology, April 1996.

[20] Robert Orfali, Dan Harkey, "Client/Sever Programming with JAVA and

CORBA", Wiley, 1997.

[21] Orfali, Harkey, Edwards, "The Essential Distributed Objects Survival Guide", John Wiley & Sons INC, 1996.

[22] Jennings, N. R. and Sycara, K.P., Wooldridge, M., "A Roadmap of Agent Research and Development In Journal of Autonomous Agents and Multi-Agent Sysetm". 1(1), pp. 7-36. July 1998.

[23] 길광수, "정기선사, 차별화된 인터넷 부가가치 서비스를 제공해야", KMI 주간해사정보」, 제2000-35호, 2000. 3.

[24] 장일동, 최정만, 황태균. "전자상거래 관리사", 21세기 출판사, 2000. 9.

[25] U.S. Department of Commerce, Retail 전자상거래, Noverber 2000.

[26] KMI 홈페이지, 「주간해사정보」, 2000-30호.

[27] John Desborough, Intranet Web Development, Infobook. 1996

[28] Wooldridge. M. J. and Jennings, N.R.. "Intelligent Agents : Theory and Pratice", The Knowledge Engineering Review 10 (2) pp. 115-152, 1995.

[29] 최중민, "에이전트의 개요의 연구방향", http://cse.hangyang.ac.kr/~kmchoi /papers/agent-intro/kiss/nodel.html

[30] Milojicic, D. S., Condict, M., Reynolds, F., Boilnger, D. and Date, P., "Mobile Objects and Agents", Second USENEX Conference on Objects Oriented Technologies and Systems(COOTs), 1996.

[31] Straber, M., Baumann, J. and Hohl, F., "MOLE: A Jana Based Mobile Agent System", European Conference on Object Oriented Programming, pp. 301-308, 1997.

[32 Aoun, B., "Agent Technology in Electronic Commerce and information Retrieval on the internet" Proc. Of Aus 1996.

[33] Durfee, E. H., Lesser V. R. and Corkill, D., "Gohernent Cooperation among

Communicating Problem Solves", IEEE Transactions on Computers C-33(11), pp. 1275-1291, 1987.

[34] Gasser. L. and Bond, A.H., "Reading in Distributed Artificial Intellingence", San Mateo, CA:Morgan Kaufmann, 1988.

[35] BroadVision and Personalization White Paper, Apr 1997.

[36] 전자상거래에서 패키지 상품판매를 지원하는 트랜잭션 관리 기법, 최희영, 황부현, 허기태, 한국정보처리학회 논문지 제7권 제9호. 2000.

[37]http://www.amazon.com/exec/obidos/subst/home/home.html/103-6853115-4160613

[38] 이건창, "지능형 에이전트의 적용현황과 전망", '분야별 EC 정책연구', 성균관대학교, 1998.

[39] http://fishwrap-docs.www.media.mit.edu/docs

[40] http://www.zdnet.com

[41] http://www.jobcenter.com

[42] http://agents-inc.com

[43] http://www.whiteis.com/se

[44] http://www.openseame.com

[45] http://bf.cstar.ac.com/lifestyle

[46] 장일동, 이희용, "CORBA 기반 지능형 에이전트에 관한 연구", 한국 OA학회 논문집, 2002. 3.

[47] http://www.peregrine.com

[48] http://www.carl.org:80/news/homework.html

[49] Joseph Williams, Bots and other, CA-October p. 15-19, 1996.

[50] http://emporium.turnpike.net/G/gsc

[51] http://www.shopfido.com/cgi-bin/Welcome

[52] http://www.ftp.com

[53] Wooldridge, M. and Jnnings. N. R., Intelligent Agents. Lecture Notes in Artificial Intelligence#890. Springer-Verag, 1995.

[54] Nissen, M., "A Technology & Business Application Analysis, http://www.berkeley.edu/heilmann/agents, 1995.

[55] Lenny, F., "What's an Agent, Anyway? A Sociological Case stuey", http://foner.www.media.mit.edu/people/foner/Julia.html, 1994.

[56] Castelfranchi, C., "Guarantees for autonomy in cognitive agent architecture", in Woolridge M. and Jennings N., eds., *Intelligent Agent : ECAI Workshop on Agent Theories, Architecture, and Languages*, Springer-Verlag, 1995.

[57] Wooldrige, M. and R.Jennings, N., "Intelligent Agents : Theory and Practice", The Knowledge Review, Vol. 10.2,1995, pp.115-152.

[58] Magedanz, T. and Popescu-Zeletin, R., "Towards 'Intelligence on Demand'-On the Impacts of Intelligent Agent on IN", 4th International conference on Intelligence in Networks, November 1996.

[59] Rao, A. S. and Georgeff, M. P., "Bdi agents : From theory to practice", Tech. Rep. 56, Australian Artificial Intelligence Institute, Melbourne, Australia, Apr 1995.

[60] 해양한국, pp. 95, 2000. 10.

[61] 장일동, 위승민, 김시화, "분산 환경에서 해운전자상거래 에이전트 설계 및 구현", 한국항해학회 논문집, 2002. 3.

[62] 윤윤중, "와해성 기술혁신", 주간경제, 제586호 LG경제연구원. 2000. 8.

[63] "E-business for shipping", Compuship, June/July, p. 4, 2000.

[64] 월간 해양수산 통권 제192호, "해운, 항만분야의 전자상거래 도입 동향" pp.

61, 2000. 9.

[65] 강종희, "인터넷 용선으로 브로커 역할 축소 예상", 「KMI 주간해사정보」, 제 2000-34호, 2000. 3.

[66] Allcock, W., Chervenak, A., Foster, I., Kesselman, C., salisbury, C. and Tuecke, S., The Data Grid : Towards an Architecture for the Distributed Managment and Analysis of Large Scientific Datasets. *Journal of Network and computer Application,* 23 : pp. 187-200, 2001.

[67] 해양한국 제 325호, pp. 80, 2000. 10.

[68] 해양한국 제 325호, pp. 95, 2000. 10.

[69] 해운연구 : 이론과 실천, 해운 B2B 전자상거래의 현황과 운영 전략 pp. 50, 2001.

[70] 임위시, "선박의 감정평가에 관한 연구", 석사학위논문, 건국대학교 행정학과, 1984.

[71] 해양한국, 2001. 3.

[72] 강종희 "인터넷 용선으로 브로커의 역할 축소 예상" "KMI 주간해사정보", 제 2000-34호, 2000. 3.

[73] 양시권, 김순갑, "선박적화", 한국해양대학교, pp. 1, 1980.

[74] 한국경영학론 Ⅰ, 한국해운기술부, 연구자료-040, pp. 7, 1987.

[75] Roy P., Containerline Performance and Service Quality, University of Liverpool, Marine Transport Center, pp. 17-26, 1980.

[76] Buxmann, P. and Gebauer, J., "Internet-based Intermediaries-The Case of the Real Estate Market," Proceedings of the 6th European on Information Systems(ECIS'98) June 4-6, 1998.

[77] 주재훈, @비즈니스: 전자상거래, 비봉출판사, 2000.

[78] 류동근, 김상열 "해운전자상거래 시대의 해운산업 거래구조 변화에 관한 이

론적 연구", 제2회 광양항국제포럼 및 한국해운학회 창립20주년 기념 국제학술대회, 2002. 4.

[79] Bailey, J. P., Bakos, Y., "An Exploratory Study of the Emerging Role of Electronic Intermediaries", *International Journal of Electronic Commerce*, Volume 1, No. 3, pp. 7-20, Spring 1997.

[80] Alex Berson and Jay Ranade, "Client/Server Architecutre", Series Advisor McGraw-HILL INTERNATIONAL EDITIONS Computer Science Series. 1992.

[81] Alan, W., Brown and Keith Jaeger, The Future of Enterprise Application Development with Component and Patterns, STERLING SOFTWARE, Aug 1998.

[82] Xavier Pacheco and Steve Teixeira저, 류광, 정우철 편저, Delphi 5.0 프로그래밍 Bible, 정보문화사, 2000.

[83] ITU-T. Basic Reference Model of Open Distributed Processing Part 4 : Architecture Semantics ITU-Rec X.904. ISO/IEC 10756-4, August 1992.

[84] ITU-T. Basic Reference Model of Open Distributed Processing Part 1 : *Overview and Guide to the Use of the Reference Model*. ITU- Rec X.901. ISO/IEC 10746-1, July 1992.

[85] Marcus, J. S., Icaros, Alice and the OSF DME. *Proc. of the Fourth International Symposium on Integrated Network Management*, 83-92, Santa Barbara CA, May 1995.

[86] Wollrath, A., Riggs, R. and Waldo, J., "A Distributed Object Model for the JAVA System", In Proceedings of the 2nd Conference on Object-Oriented Technologies and System (COOTS), Toronto, Canada, June 1996.

[87] Sun Microsystems, JAVA™ Remote Method Invocation Specification, Rev.

1.7. Sun Microsystems, Inc., December 1999.

[88] Jon Siegal. et. al., CORBA Fundamentals and Programming, OMG. 1996.

[89] Robert Orfail, Dan Harkey, Client/Server Programming with JAVA and CORBA Second Edition, John Wiley & Sons, Inc. 1998.

[90] Thomas, J., Mowbray, William, A. and Ruh, Inside CORBA, Addison Wesley Longman, Inc. 1997.

[91] XML specification, Version 2.3.1, OMG, "http://www,w3.org/TR/1998/REC-xml-19980210

[92] CORBA specification Version 2.3.1, OMG, "http://www,omg.org"

[93] Case, J., McCloghrie, K., Rose, M. and Waldbusser, S., "Introduction to version 2 of the Internet-standard Network Management Framework", *Internet Request for Comments 1441*, April 1993.

[94] Case, J. D., Fedor, M. S., Schoffstall, M. L. and Davin, J. R., "A Simple Network Management Protocol", *Internet Request Comments 1157*, 1990.

[95] ISO. *Information Processing Systems-Open Systems Interconnection-Basic Reference Model-Part 4 : Management framework.* International Organization for Standardization, International Standard 7498-4, 1991.

[96] Ofali, R. and Harkey, D., "Client/Server Programming with JAVA and CORBA", JOHN WILEY & SONS, INC., 1998.

[97] Millikin, M. D., DCE : Building the Distributed Future, *Byte*, 125-134, June 1994.

[98] http://www.omg.org/technology/documents/formal/corbaiiop.htm

[99] 민병의, 박상규, 장명욱, 이광로, 황승구, "분산 환경 기반 개방형 에이전트 구조", 정보과학회지 제 15권 1997.

[100] Mani Chandy, K., Adam Rifkin, Paolo, A. G., Sivilotti, Jacob Mandelson,

Matthew Richardson, Wesley Tanaka and Luke Weisman, A World-Wide Distributed System Using JAVA and the Internet. Fifth International Symposium on High Performance Distributed Computing, Syracuse, pp. 11-18. Recipient of the Best Paper Award, August 1996.

[101] Coulouris, G., Dollimore, J. and Kindberg, T., Distributed Systems : Concepts and Design, 2nd Ed., Addison-Wesley, 1994.

[102] Jon Siegel., CORBA Fundamentals and Programming, USA, John Wiley & Sons, Inc., 1996.

[103] Thomas, J. and Mowbrayd., The Essential CORBA, USA, John Wiley & Sons Inc, 1995.

[104] 장동인 "데이터웨어하우스", 대청, 2000.

[105] David M. C., Colin, G. H. and Aaron Kershenbaum., "Mobil Agents : Are they a good idea?", IBM Research Report, 1995.

[106] David, M. C., Colin, G. H. and Benjamin G., "Internet Agent form Mobile Computing", IBM Research Report, 1995.

[107] Jeri Edwards, 3-Tier Client/Server At Work ,Wiley, 1998.

[108] Patil, R., Files, R., Patel-Schneider, P., McKay, D., Finnin, T., Gruber, T. and Neches, R., The DARPA Knowledge Sharing Effor : Progress Report. In Pricipals of Knowledge Representation and Reasonning : Proceeding of the Third International Conference, November 1992.

[109] 왕창종, 이세훈, CORBA 프로그래밍, 1st, 대림, pp. 8-22, 1998.

[110] Berners-Lee, T., Fielding, R. and Frystyk, H., "Hypertext Transfer Protocol HTTP//1.0", Internet Draft draft-ietf-http-v10-spec-04 html, HTTP Working Group, Work in progress, 1995.

[111] Gary Cornell, Cay, S. and Horstmann, 강동헌, 박재현 공저 , core JAVA,

2nd, 영한 출판사, pp. 689-733, 1997.

[112] Ashton Hobbs, 신상호 譯, 데이터베이스 프로그램과 JDBC, 인포북, pp. 13-148, 1998.

[113] http://www.inprise.com/visibroker, Visibroker

[114] Silvano Maffeis. System Support for Distributed Computing, Department of Science University of Zurich, CH-8057 Zurich, 1990.

[115] 류동근·문성혁, "e-commerce의 전망과 해운기업의 대응방안", 2000년도 한국해운학회 봄철학술발표논문집, Vol. 29, 2000. 4.

[116] http://www.chosun.com/

[117] 한재민, 경영정보 시스템, 학현사, pp. 17-18. 1999.

[118] 오현순, "전자상거래에 의한 지역수출 활성화 방안에 관한 연구", p. 47, 1999.

[119] 이희용, 유조선 운항일정계획 의사결정지원 시스템의 개발에 관한 연구, 한국해양대학교 석사논문, p. 4. 1996. 2.