IE Interfaces
Vol. 12, No. 4, pp. 635-642, December 1999.

이기종 ERP 연동되는 컴포넌트형 SCM 시스템 개발

장경원[1] · 최정원[1] · 여성주[1] · 왕지남[2]

[1]아주대학교 산업공학과 / [2]아주대학교 기계 · 산업공학부

The Development of Component-Based SCM System to be integrated with Heterogeneous ERP

Kyung-Won Chang[1] · Jung-Won Choi[1] · Sung-Joo Yeo[1] · Gi-Nam Wang[2]

Recently, business application vendors have responded to customer demand with several technological and organizational changes for reflecting network-oriented process flow with electronic communication and expanding their market. In organizational changes. vendors are adding new functionality to their core offerings through mergers and acquisitions, or by partnering with front-office operation and SCM vendors. And technological changes are embracing the development of component-based application and middleware technology to be possible with integration of their function modules. We will present the design and development process of component-based SCM system to be integrated with heterogeneous ERP under implementation. Especially, we survey and analyze the technology related with CBD(Component-Based Development) to be adapted this research - component's design and methodology, the interface for component integration, middleware's development methodology of Enterprise Middleware vendo to bundle various types of middleware into a single package product with easy-to-use interface.

1. 서 론

시장의 요구 변화와 격심해져가는 세계 시장 경쟁은 기업의 비즈니스 프로세스(business process)에 대해 끊임없는 변화와 수정을 요구하고 있다. 이러한 변화를 유도하는 핵심 요인은 전 세계적인 시장 경쟁, 짧아져가는 생산주기와 제품 life cycle, 그리고 시장의 세분화와 포화라고 할 수 있다. 신속성(Speed), 유연성(Flexibility), 단순성(Simplicity)이 BPR(Business Process Reengineering)의 성공 핵심 요소이지만, 그 개별 프로세스의 적용범위는 오히려 다양한 관련 기업체, 고객, 공급업체 등 상하위 공급체계까지 확장되어 가는 추세이다. 그러므로 오늘날의 사용자들은 고도로 통합된 비즈니스 응용 시스템(business application system) 뿐만 아니라 경제적이고 용이한 사용성을 포함한 분산된 응용시스템 (Distributed Application System)을 요구하고 있다. 기업내 기능중심의 순차 처리에서 네트워크 지향의 프로세스 흐름으로의 변화는 주요 ERP(Enterprise Resource Planning) 벤더(vendor)들로 하여금 더 쉽고, 더 경제적인 구현과 분산 채널(distribution channel)로 이주하는 다양한 전략을 세우게 하고 있다. 그 전략 중의 하나가 더 작고 더 단순하며 저가의 기업 표준 인터페이스(interface)를 제공할 수 있는 컴포넌트 기반 ERP로의 변화이며, 또 다른 전략으로는 회계, 재무, 인사, 제조 등의 기간계 업무(back-office operation)의 통합기능을 제공하는 ERP 제품에 판매, 고객 서비스, 현장 서비스(field service)와 같은 판매 프로세스 관점에서 관리하는 front-office operation(예, SFA(Sales Force Automation))의 통합 그리고 기업 내외부의 공급체계의 통합(예, SCM(Supply Chain Management))을 통해 기업 전사적인 통합 애플리케이션(application)을 제공하고자 하는 "portfolio assembly" 전략을 사용하고 있다. 이러한 전략은 곧 컴포넌트 기반의 애플리케이션 개발과 기능 모듈들의 통합을 가능케 하는 분산객체를 이용한 미들웨어 기술로 대변할 수 있다. 본 연구에서는 개발 중에 있는 이기종 ERP와 연동되는 컴포넌트형 SCM(Supply Chain Management) 시스템 설계 및 그 구현과정을 제시하고자 한다. 또한 컴포넌트 설계 및 개발 방법론과 이의 연동을 위한 인터페이스 그리고 시스템간 연동을 위한 미들웨어에 대해서 다양한 형태의 미들웨어를 하나의 패키지 상품으로 개발한 기업 미들웨어 벤더를 대상으로 제시하고자 한다.

2. 컴포넌트 개발 방법론

2.1 컴포넌트의 구조

관련된 서비스를 물리적인 블랙 박스 캡슐(black-box encapsulation)로 제공하는 실행 가능한 코드 단위로서 정의할 수 있는 컴포넌트는 2가지 기본 아이디어에서 도출된 개념이다. 첫째로는 만약 소프트웨어 컴포넌트를 부품을 조립하여 완성된 제품을 만드는 것과 같이 조립할 수 있다면, 애플리케이션 개발이 급진전할 것이라는 것과, 둘째로는 기업 고유의 요구에 따라 상호운용성(interoperability)이 있는 소프트웨어 컴포넌트들을 개발자들이 필요한 부품을 찾아 끼우듯이 서로 연동하여 유효하게 이용할 수 있어야 한다는 것이다.

컴포넌트는 <그림 1>과 같이 컴포넌트 기능 정의 부문인 상세 설명 계층(Specification Layer)과 구현을 위한 서비스 정의 계층(Implementation Layer)의 명백한 분리를 통해 완전한 캡슐화의 문제를 해결하였다. 이는 OMG(Object Management Group)의 CORBA와 마이크로소프트사의 DCOM과 같은 상호작용 표준(Interaction Standards)을 통해 이루어질 수 있다. 재사용을 위한 소프트웨어 작성 표준을 통해 컴포넌트의 물리적 구현부분이 변하더라도 제공되는 서비스는 일관되게 유지될 수 있는 특성을 갖게 된다. 컴포넌트가 갖고 있는 이러한 특성들이 legacy 소프트웨어와의 연동에 있어서는 특히 중요하게 작용되고 있다.

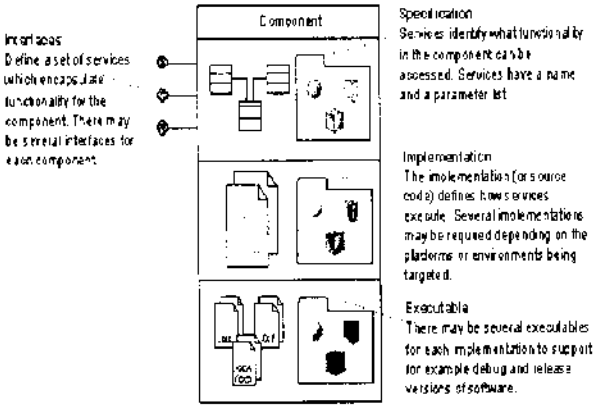

그림 1. 컴포넌트 구조.

2.2 컴포넌트 기술

최근에 많은 소프트웨어 개발 업체들은 미리 조립되거나 존재하는 컴포넌트들의 조립과 통합을 위해서 컴포넌트 개발 방법과 그들간의 인터페이스에 대하여 많은 관심을 가지고 있다. CBD (Component-Based Development)는 오늘날의 분산 시스템을 지원할 수 있는 접근법으로서, 애플리케이션 개발의 뷰(view)들을 시스템 기능이 잘 정의된 다이어그램(diagram)의 조립 프로세스로서 제공하고 있다. 이 접근법의 실제적인 형태는 컴포넌트 기반 기술로서 나타났으며, 그 중 하나가 컴포넌트형 애플리케이션을 여러 관점에서 묘사할 수 있도록 설계된 UML(Unified Modeling Language)이다. UML은 초기에는 객체 지향 모델링 표기법으로 개발되었으나 OMT(Object Modeling Technique)와 Booch의 광범위한 컨소시움을 통해 컴포넌트 설계 능력으로까지 확장되었으며, 이는 시스템의 분석 및 설계 단계에 적합한 모델링 그룹과 실제로 제어 소프트웨어를 구현하는 단계에 적합한 모델링 그룹으로 나뉘어 구성되어 있다. 그 중 전자에 속하는 그룹으로는 class diagram, use case diagram 등이 있으며, 후자 그룹으로는 component diagram, deployment diagram 등이 있다.

컴포넌트형 애플리케이션을 개발하기 위해서는 모델링 언어와 더불어 방법론 개발이 중요시되고 있다. 이러한 방법론은 컴포넌트 개발자들의 관련 기술사용의 효율 증대를 위해 요구되는데, 최근 의 경향은 분산 웹 기반 하에서 legacy 시스템과 패키지 애플리케이션의 재사용을 지원할 수 있는 방향으로 나아가고 있다. 그리하여 많은 관련 벤더들은 컴포넌트로부터 기업 규모의 시스템을 개발하기 위한 방법론을 제시하고 있는데, 그 중 대표적인 개발 방법론은 다음과 같다.

- Rational사의 Unified Process
  : 소프트웨어 라이프사이클 전체를 수렴할 수 있는 프로세스 프레임워크(Framework) 제공
  : UML 표기법 사용
- Select사의 Select Perspective Method
  : 일반적인 컴포넌트 설계 접근법 묘사
  : UML 표기법 사용
- Sterling Software사의 Enterprise-CBD approach
  : Catalysis 접근법에 의해 영향을 받음
  : UML을 사용하여 컴포넌트 specification과 조립(assembly) 부문 강화

본 연구에서는 많은 방법론 중 가장 표준화되어있으며 Sterling Software사가 CBD 방법론 개발을 위해 Icon Computing사와의 연합을 통해 개발한 Catalysis 접근법을 제시하고자 한다.

2.3 컴포넌트 개발 방법론

- Catalysis 접근법

컴포넌트 개발을 위한 기본적인 접근은 분석을 통하여 지원되어질 수 있는 요구사항과 함께 시작한다. 요구정의는 Use case Diagram과 애플리케이션과 컴포넌트를 정의하기 위한 Business type Modeling을 이용한다. 도메인에 대한 정확한 분석을 위하여 분석 시에는 도메인 모델링을 이용하며 요구사항의 정의와 도메인에 대한 분석이 끝나면 실제 구조적인 접근을 위하여 Architecture Modeling과 컴포넌트 상세 설명을 위한 Context Model, Interface Model, Component specification을 이용한다.

(1) Use Case Diagram을 통한 분석

Use Case Diagram은 actor가 어떤 행동(use case)을 하는지에 대한 정의를 표현할 수 있다. 컴포넌트의 정확한 목적을 정확히 정의할 수 있으며소프트웨어 설계에서 actor와 소프트웨어, 소프트웨어의 작은 모듈과 큰 모듈사이에의 상호 운영을 표현할 수 있다. Use Case Diagram에서 snapshot은 Use case 발생전후에 생성될 수 있는 상태들을 보여주는 인스턴스(Instance)로서 형식적인 제약조건이나 목적 등을 가시적으로 보여주고 표현하기 위하여 사용되어진다.

(2) Sequence Diagram(chart)을 이용한 접근

이와 같은 설계가 다 완성되어진 후 각각의 Use case들 사이의 상호작용을 표현하기 위하여 Sequence chart 또는 Diagram을 사용한다. 이는 분산되어져 있거나 크고 작은 서로 다른 객체들 사이에서의 메시지 전달을 보여주기 위한 표기법이다. 이와 같이 객체 지향 컴포넌트 내에서 서로다른 객체들간의 메시지 전달과 통신은 기존의 객체지향 모델링 방법과 매우 유사하다.

(3) 인터페이스 설계

객체들을 하나의 컴포넌트로 구현하기 위해서는 적절한 모듈단위로의 재정의와 재정의된 컴포넌트들과의 인터페이스가 필요하다. Context modeling의 한 부분인 인터페이스는 Interface Modeling을 이용하여 설계되어질 수 있으며, 입력과 출력을 정의하고 Component Specification을 통해 기능을 상세화시킨다. 여기서 컴포넌트들은 <그림 2>의 절차를 통해 생성된 컴포넌트일 수도 있고 미리 만들어져 제공된 컴포넌트일 수도 있다. CBD 개발시에 인터페이스에서 발생되어질 수 있는 문제가 비동기 문제이다. 이렇게 수많은 컴포넌트들간의 이벤트 전달 과정에서 비동기의 문제점은 얼마만큼 인터페이스 설계를 정확히 하느냐에 따라서 해결되어 질 수 있다. 또한 이기종의 컴포넌트일 경우는 미들웨어를 이용한 분산객체기술로 컴포넌트들간의 인터페이스를 설계할 수 있다.

이와 같은 기업 애플리케이션의 컴포넌트화는 또 다른 기술, 즉 분산 객체 지향의 미들웨어를 출현시키는 계기가 되기도

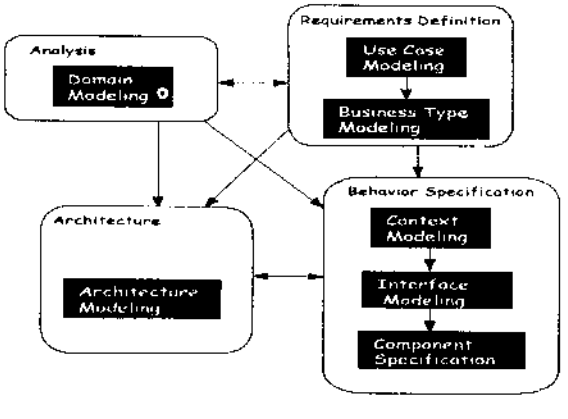

그림 2. Catalysis 방법론의 모델링 절차.

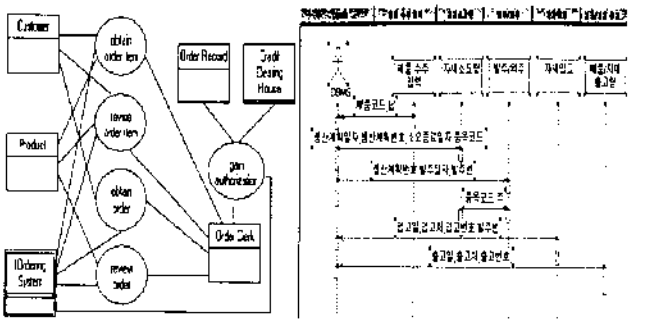

그림 3. Use case/Sequence Diagram.

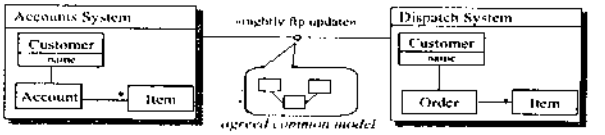

그림 4. 인터페이스 모델링.

하였으며, 기술적으로는 인터페이스를 이용하여 동일회사 제품뿐만 아니라 다른 벤더 제품과의 통합을 시도하고 있다.

3. 미들웨어 개발 방법론

3.1 미들웨어 진화

미들웨어라는 용어는 애플리케이션에서 애플리케이션으로의 데이터 추출 및 이동을 용이하게 하는 일종의 소프트웨어를 묘사하기 위해 사용되었다. 미들웨어는 전통적으로 TP 모니터(Transaction-processing monitors), 데이터베이스 액세스(database-access), 메시징(messaging), 객체 브로커링(object brokering)의 4가지 분야 중 하나로 귀착된다(Alan, 1998). 통신 게이트웨이(Communication gateway)은 TP 모니터로서의 미들웨어이고 Open Database Connectivity와 Java Database Connec- tivity는 데이터 액세스의 예이다. 더 최근에 CORBA, Enter- prise JavaBeans, ActiveX/COM과 같은 기술은 객체 기반 애플리케이션을 위한 미들웨이로서 사용되고 있다. 그리고 웹 클라이언트(Web client)를 서버 기반의 애플리케이션과 연결하는 Web and Java application server도 있다.

미들웨어 아키텍처(architecture)의 발전은 분산 컴포넌트 환경의 요구로 인해 유도되고 있다. Multi-tier 환경에서, 객체와 컴포넌트는 다른 객체와 컴포넌트로부터 서비스를 요구하게 된다. Object Broker 또는 ORBs(Object-component Re-quest Brokers)는 사용하는 언어나 파일 포맷(file format)과 상관없이 다양한 부문들이 통신될 수 있도록 애플리케이션의 요구에 적절한 객체와 컴포넌트를 찾아서 정보를 라우팅(routing)하고 요구되는 변형을 수행한다. 이러한 발전과정을 통해 오늘날 미들웨어는 다음과 같은 새로운 카테고리로 재분류되고 있다.

· Development middleware(개발 미들웨어)
: 객체 또는 컴포넌트 기술을 사용하는 분산된 응용 시스템을 구축하고자 할 때 그 연결자로서 사용
· Integration middleware(통합 미들웨어)
: 분산된 그리고 이기종의 애플리케이션사이에서 비즈니스 프로세스를 통합 하고자 할 때 그 연결자로서 사용

이 두 분야의 차이는 가치 부여에 있다고 할 수 있다. 개발 미들웨어는 애플리케이션 구축을 위한 것이고 반면에 통합 미들웨어는 기존의 애플리케이션과 시스템 통합을 위한 것이다.

본 논문에서는 주요 ERP 벤더 중 15.6%(1997년 기준, source: Gartner Group)의 시장점유율을 갖는 SAP R/3의 컴포넌트 인터페이스 설계와 시스템간 연동을 목적으로 하는 기업 미들웨어 전문 벤더 중의 하나인 Active Software의 미들웨어의 개발방법론을 제시하고자 한다.

3.2 SAP R/3

(1) BAPIs

SAP은 R/3 프로세스와 데이터를 SAP Business Objects(이하 BO로 칭함)의 형태로 구현함으로서 객체 지향 기술을 사용하고 있다. 외부 다른 애플리케이션은, 표준화된 플랫폼(platform)이며 독립적인 인터페이스인 BAPIs(Business Application Programming Interfaces)를 사용하여 BO에 접속할 수 있다. BO는 R/3의 데이터와 비즈니스 프로세스를 캡슐화한 일종의 "black box"라고 할 수 있으며 그 구조는 4단계의 계층(layer)으로서 구성되어 있다.

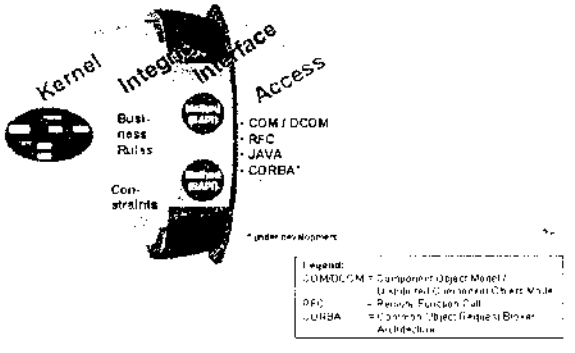

그림 5. SAP Business Object의 Layers.

· Kernel layer : 객체의 데이터와 프로세스
· Integrity layer : BO에 적용되는 룰과 제약조건으로 구성된 비즈니스 논리(business logic)
· Interface layer : 외부에 대해 해당 객체의 인터페이스 정의
· Access layer : 객체의 데이터에 대한 외부의 접속을 얻기 위해 사용되는 기술 정의

BAPIs는 BOR(Business Object Repository)에서 BO의 메소드(method)의 집합으로 정의되어 있으며, 함수 모듈(function module)로서 구현된다. 그 정의와 실제 구현의 구분을 통해 BAPIs는 2가지 방법 - BOR에서 BAPIs에 대한 객체 지향 접속방법과 C/C++ RFC class libraries call을 통한 접속방법 - 으로 접속할 수 있다. 그 중 전자는 클라이언트가 윈도우즈 플랫폼인 경우에 해당되며, BAPIs Active X Control은 OLE Automation server로서의 기능을 하게 된다. 즉, BOR에 있는 BO에 대해 OLE 객체 인스턴스(object instance)를 생성시켜 클라이언트에 포함시킴으로서 BAPIs를 호출하는 방법이다.

비윈도우즈 플랫폼에서는 BAPIs Active X Control과 같은 객체 지향 접속은 아직 가능하지 않다. 이러한 경우에는 BAPIs에 연결된 함수 모듈에 대해 RFCs(Remote Function Calls)를 호출함으로서 BAPIs에 접속할 수 있다.

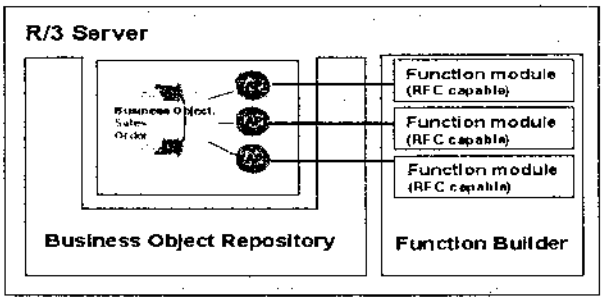

그림 6. BO, BAPIs, function module의 관계도.

(2) ALE

ALE(Application Link Enabling)은 분산된 응용 시스템의 설계 및 운영을 위한 툴 뿐만 아니라 기술적으로 독립적인 애플리케이션을 연결할 수 있는 분산 시나리오와 비즈니스 프로세스 중심의 통신기술을 포함한 미들웨어이다. 분산시나리오가 저장되어 있는 ALE 분산 참조 모델(ALE Distribution reference model)은 "재고관리"와 같은 "distributable function type", 메시지 및 데이터의 할당기준을 제공하는 "filter object type", "message type"으로 구성되어 있으며 이러한 기능함수들을 drag and drop GUI 툴을 사용하여 고객의 시스템에 대해 적절한 통합기능을 제공한다.

메시지 기반 애플리케이션의 그 연결원리를 보면, 애플리케이션 부문은 비즈니스 메시지 생성 및 입력 메시지(input message) 처리를 위한 통합된 작업흐름(workflow)을 지원한다. 또한 비즈니스 데이터 생성 및 수신 애플리케이션을 위한 프로세스를 정의한다. 분산(Distribution) 부문은 비즈니스 단계를 기술 단계와 연결하는 역할을 한다. 즉, 메시지 수신자에 대한 점검 및 메시지 필터링(filtering)과 변환을 담당한다. 이러한 연결은 R/3사이에서, R/2 및 third-party system과의 연결도 포함한다. 통신 부문은 기술적인 단계에서의 데이터 전송을 담당하며, 이는 e-mail에 대한 표준 라우팅인 X.400과 같은 기술 표준을

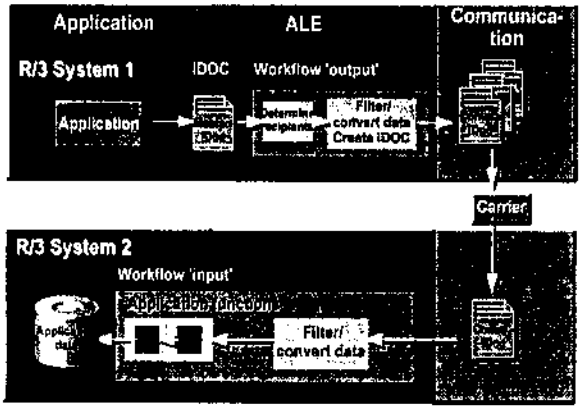

그림 7. Message-based coupling의 원리.

지원한다. EDI 인터페이스로 알려진 Intermediate documents (IDOCs)는 데이터의 교환 역할을 담당한다.

3.3 Active Software(Active Works)

3.1에서 제시된 바와 같이 다양한 형태의 미들웨어들이 하나의 패키지, 즉 기업 미들웨어라는 제품으로 출시되었다. 이 제품은 패키지 애플리케이션, 데이터베이스, legacy 시스템, 인터넷 애플리케이션과 같은 컴퓨팅 자원 통합의 용이성을 제공해 주는 역할을 수행하며 그 주 벤더로는 Active Soft- ware, Cross Worlds Software, Extricity Software 등이 있다. 그 중 Active Software의 ActiveWorks은 고객의 비즈니스 프로세스를 반영하는 이벤트 메시지(event-message)에 기반을 두고 있으며, 이벤트의 저장, 대기(queue), 라우팅 기능을 갖는 정보 브로커(information broker)와 비즈니스 프로세스를 시스템에 반영하는 에이전트(Agents), 개발자에게는 애플리케이션사이의 이벤트와 상호작용을 정의할 수 있게 하고 관리자에게는 시스템을 관리하고 모니터링할 수 있게 하는 GUI 툴을 포함하고 있다. 특히 ActiveWorks 의 어댑터는 애플리케이션 함수와 APIs을 통해, 애플리케이션과 또 다른 정보자원에 대한 동적인 접속기능을 제공하며, 어댑터 파트너 프로그램(Adapter Partner Program)을 통해 다양한 하드웨어, 소프트웨어 벤더뿐만 아니라 시스템 통합 벤더와의 포괄적인 비즈니스 연합을 이루고 있다. 이러한 연합으로 Active Software와 파트너들은 애플리케이션, 데이터베이스, 미들웨어, 프로그래밍 언어, 메인프레임 데이터에 대해 38개의 패키지 어댑터와 고객 맞춤 어댑터(custom-developed adapter)를 제공하고 있다. 특히 미들웨어 측면으로는 4종류의 어댑터 - CORBA, Javelin/Workflow, MQ Series, Telephony API -가 구축되어 있다. CORBA 미들웨어 어댑터는 CORBA 기반의 클라이언트와 Active- Works의 접속기능을 제공한다. 쓰여진 프로그래밍 언어와는 상관없이 CORBA 기반의 객체는 API 요청(request)를 할 수 있는 인스턴스를 생성하기 위해 IDL(Inter- face Definition Lan- guage)를 사용할 수 있다. CORBA 객체 서버와 ActiveWorks 클라이언트가 속한 중간 프로세스가 인스톨되고,

그림 8. CORBA-middleware adapter.

이 프로세스는 CORBA 클라이언트 애플리케이션과 정보 브로커 사이 그리고 정보 브로커에 연결된 다른 시스템 사이의 정보 전송을 위한 게이트웨이(gateway)로서 작용한다. CORBA 클라이언트 객체와 CORBA 게이트웨이 프로세스사이의 통신은 IIOP 표준 프로토콜을 사용한다.

4. 컴포넌트형 SCM시스템 개발

4.1 대상 시스템 분석

본 연구의 대상이 되는 기업은 가공된 식료품 생산을 위주로 하는 회사로 유통기한이 비교적 길다는 특징이 있다. 현재 이 기업은 결품의 발생으로 인해 소비자의 수요에 못 미치는 것과 과잉재고로 인한 재고 비용의 누적 등이 있으며 이는 합리적인 물류시스템 및 재고자동보충시스템이 도입되지 못해 발생하는 결과라고 볼 수 있다. 즉 근본적으로 정확한 수요예측 시스템의 부재로 인한 정확한 이고(공장에서 사업장으로의 물류)계획을 설정할 수 없어 합리적인 접근을 할 수 없다는 것이다. 현재 시스템의 문제점은 다음과 같다.

- 미출고의 발생
- 출고거점과 배송지역의 부적절함
- 물류/영업의 서비스 개념의 차이 발생
- 배정 제품의 미출고 잦은 발생
- 사업장 창고에서 제품 부족이나 결품 현상의 잦은 발생
- 사업장 창고에 과잉재고 발생

4.2 시스템 설계

(1) 2일 근거리 수송

이는 전체시스템에서 수송부분의 핵심으로 역수송(공장에서 수배송센터에 제품을 수송한 뒤 다음날 생산량의 부족과 수요의 증가로 해당 제품을 다른 수배송센터로 수송하는 것을 의미)을 방지하는 데 있다. 자주 발생하는 이 역수송은 단일 수송 만을 고려한데에서 발생한 것으로 역수송으로 인한 물류비용은 생각보다 많이 차지한다. 이를 해결하는 방법으로 그 다음날까지를 고려하는 2일 근거리 수송으로 이러한 역수송을 줄일 수 있다.

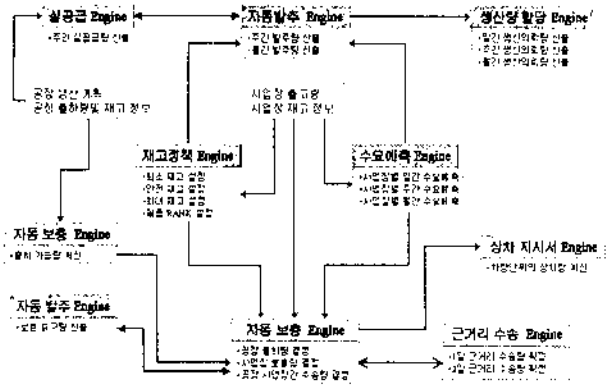

그림 9. 재고 자동 보충 시스템 정보 흐름도.

(2) 생산 · 판매 · 물류 통합

본 시스템은 생산과 판매, 물류 각 단계의 통합을 통해 시간적 동기화뿐 아니라 각 단계에서 발생하는 데이터의 실시간 공유 및 적정 재고 수준의 설정으로, 재고 유지비용을 절감할 수 있는 수요예측기반의 재고자동보충시스템을 <그림 9>와 같이 설계하고자 한다. 즉, 재고 정책에서 각 제품에 대한 재고 속성(안전재고량, 최소/최대 재고량, 제품의 RANK)을, 수요예측에서는 일 · 주 · 월별 데이터를 통한 예측치를 DB에 저장한다. 공장의 생산계획엔진에서의 생산가능량과 수요량을 함께 계산해서 공장의 생산량을 결정한다.

이는 생산 공장에 생산량을 할당하게 해주며 수배송 센터에 보낼 제품량을 2일 근거리 수송을 기반으로 한 자동보충엔진을 수행함으로서, 각 수배송센터에 배송될 제품의 양이 결정한다. 또한 배차엔진에서 최적의 경로로 제품의 수배송이 이루어지도록 결정된다.

4.3 컴포넌트 모델링 및 설계

사용자의 요구분석 그리고 구현 가능한 모델을 제시하기 위해 본 연구에서는 Catalysis 컴포넌트 개발 방법론에 UML의 모델링 기술을 이용하여 재고자동보충시스템을 대상으로 한 모델링 과정과 자동 보충 및 근거리 수송 엔진(Engine)에 대한 컴포넌트 설계도 일부를 제시하고자 한다.

4.4 시스템의 구현

(1) 시스템 개발환경

- 플랫폼 : 한글 Windows NT 4.0 버전
- 데이터베이스 : Oracle Workgroup Server 7.3 for NT
- 프로그래밍 언어 : Visual C++ 6.0
- 컴포넌트 구성 규격 : ATL/COM
- DB 트랜잭션 컴포넌트 : OLE DB

(2) 컴포넌트 인터페이스 설계

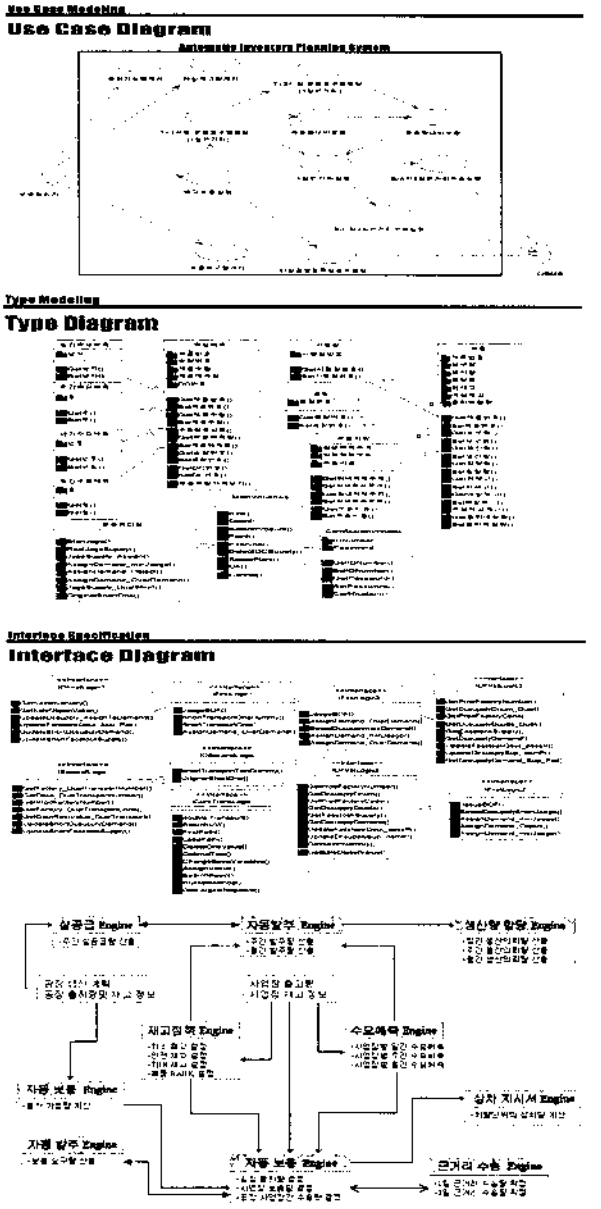

그림 10. 컴포넌트 모델링 및 설계과정.

본 연구에서는 재고자동보충시스템의 설계된 로직에 대해 GUI와 일반 비즈니스 로직을 대상으로 한 비즈니스 로직 컴포넌트(Business Logic Component), 기존 정보시스템 혹은 각 ERP 벤더들이 사용하는 이기종 DBMS와의 인터페이스 문제를 해결하기 위한 DB 트랜잭션 컴포넌트로 분류 · 구축하였다.

비즈니스 로직 컴포넌트는 코드의 재사용과 기능에 대해서 컴포넌트를 설계 · 구현하였으며, 특히 이기종 시스템에 구애받지 않은 데이터 접근을 가능하게 하기 위해 DB 트랜잭션 컴포넌트 구축은 COM을 기반으로 한 OLE DB를 사용하였다.

<그림 11>은 재고정책 컴포넌트로 안전재고량을 계산하는 인터페이스를 보이고 있다.

재고 정책 컴포넌트에는 4개의 인터페이스가 있으며 각각의 인터페이스에는 그에 따른 함수가 있다. 다음은 이들 중 안전재고량을 구하는 함수이다.

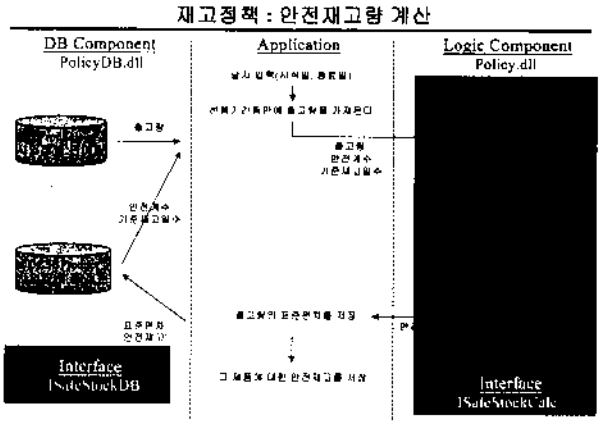

그림 11. 안전재고량 계산 인터페이스.

- 컴포넌트 : Policy.dll
- 인터페이스 : ISafeStock
- 안전재고량을 계산하는 함수

SafeStockCale([in]int iSize,

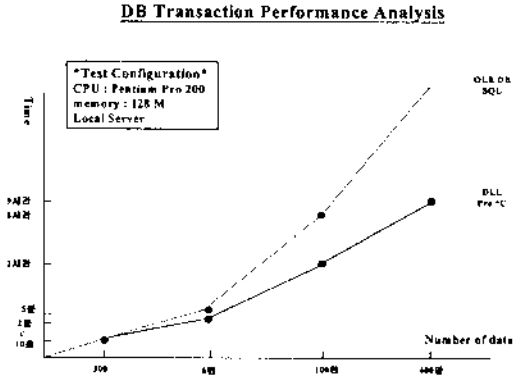

그림 12. Pro C와 OLE DB 성능비교.

[in] float *fOutCapacity,
[in] float fSafeAccount,
[in] float fStandStockDay,
[out, retval] float *retval)

iSize : 출고량의 개수
fOutCapacity : 출고량(배열)
fSafeAccount : 안전계수
retval : 반환값으로 여기서는 안전재고량을 반환해준다.

4.5 구현결과

아래 그림은 Oracle에서 제공하는 Pro C를 사용하여 작성한 프로그램과 OLE DB로 만든 컴포넌트를 데이터 수에 따라 쿼리(quarry)되는 시간을 측정한 것이다.

Pro C의 성능이 많이 앞서는 것을 볼 수 있다. 각 컴포넌트로 구현하는 것은 설계나 구현에서 기존의 방법보다 약간 복잡하고 특히 DB 트랜잭션시 성능이 좀 떨어지는 것을 볼 수 있으나 한 번의 구현으로 프로그래밍 언어에 구애받지 않고 재사용이 가능하다는 장점이 있다. 데이터베이스와의 트랜잭션속도나 성능이 많이 떨어지는데 이를 해결하기 위한 DB 테이블 및 DB 쿼리에 대한 연구, 분산환경에서 COM을 기반으로 DCOM에서 사용할 수 있도록 컴포넌트에 대한 연구가 더 수반되어져야 할 것으로 본다.

현재 본 연구는 특정 기업을 대상으로 클라이언트/서버 기반의 세부 모듈들을 컴포넌트화 하는 과정에 있다. 이에 특정 기업 대상에서 좀 더 나아가 분산 웹 기반 하에서 일반 기업에도 적용 가능한 컴포넌트 설계와 구현 기술로의 확장에 대한 연구가 더 이루어져야 하며, 더불어 기업의 전사적인 업무구조를 바탕으로 정보시스템 구축시 표준 모델이 되는 패턴들에 대한 분석 및 패턴 라이브러리를 구축에 대한 연구가 더 수반되어져야 한다.

5. 결 론

본 논문에서는 특정 기업을 대상으로 한 클라이언트/서버 기반의 SCM 시스템을 컴포넌트형 시스템 전환을 위한 컴포넌트 설계 과정을 제시하고 있다. 이에 따라 컴포넌트 설계 및 개발 방법론과 컴포넌트 연동을 위한 인터페이스 그리고 시스템간 연동을 위한 미들웨어에 대해서 다양한 형태의 미들웨어를 하나의 패키지 상품으로 개발한 기업 미들웨어 벤더를 대상으로 제시하였다.

오늘날의 분산시스템을 지원할 수 있는 접근법인 CBD에 대해 본 연구에서 적용한 기술과 방법론에 대해 다음과 같이 제시하였다. 객체 지향 모델링 표기법이면서 컴포넌트 설계 능력을 갖고 있는 UML과 컴포넌트로부터 기업규모의 시스템을 개발하기 위한 방법론인 Catalysis 방법론을 기술하였다. 또한 블랙 박스 캡슐이라고도 할 수 있는 컴포넌트에 대해 사용자가 요구하는 서비스의 제공을 위한 컴포넌트간 인터페이스 개발 기술과 그 경향에 대하여 ERP 벤더인 SAP의 R/3와 기업 미들웨어 전문 벤더인 Active Software의 ActiveWorks의 미들웨어 개발 방법론을 기술하였다.

본 연구에서 구현한 컴포넌트형 재고자동보충시스템은 기술적으로 복잡하고 DB 트랜잭션 시에 성능이 떨어진다는 단점이 도출되었다. 이를 해결하기 위해 DB 테이블 및 DB 쿼리에 대한 연구가 수반되어야 하며, 또한 전 시스템에 대해 하나의 컴포넌트만이 구현되어 컴포넌트형 시스템 개발을 위한 프레임워크(Framework)을 제시하지 못하였으며, 이기종 ERP 시스템과의 연동을 위한 효율적인 인터페이스 및 미들웨어 구축 방안을 제시하지는 못하였다. 그러나 특정 기업의 범위에서

벗어나 일반 기업에도 적용할 수 있는 컴포넌트와 패턴 라이브러리 구축을 위해 그리고 컴포넌트형 애플리케이션의 특성인 재사용성과 개발 이점이 뛰어난 컴포넌트 설계를 위한 CBD 및 PBD(Pattern-Based Development)기술 연구가 결합이 된다면 분산환경에서 보다 효율적인 컴포넌트 및 연동기능을 갖는 시스템을 구현하게 될 것이다.

참고문헌

Dave, L. (1998), Message Brokers Rising, DBMS,

Alan, R. (1998), Middleware Evolution, Information Week

Shanker, L. (1999), Enterprise Integration, Distributed Computing.

Active Software Resource Center (1999), EAI Market Overview, White Paper, *Active Software*.

Active Software Resource Center (1999), ActiveWork Solution : Front Office Application Integration, White Paper, *Active Software*.

Graf, P., Tolkmit, G. (1997), *The Business Framework*, White Paper, SAP AG

Technology Marketing (1997), *BAPI Introduction and Overview*, White Paper, SAP AG

Technology Marketing (1996), *Application Link Enabling(ALE) : Integration of Distributed Business Process*, White Paper, SAP AG

Guy, E., Henry, E. (1998), Inside Distributed COM, Microsoft Press.

Gimnes, Stockton, Reilly & Templeman (1999), *Beginning ATL Pro- gramming*, WROX

Wendy, S. (1999), *Visual C++ 6 Database Programming Tutorial*, WROX

Radding, A. (1999), Technology Forcast 1999 : Corporate Application, Survey Reports, *Pricewaterhouse Coopers*, 403 - 425

Sandra, M., Laura, H. (1998), Active Software Introduces Adapter Partner Program, *Active Software*

Sandra, M., Laura, H. (1998), Middleware Adapter : CORBA Edition, *Active Software*

Hans, E. E., Magnus P. (1998), UML Toolkit, John Wiley & Sons, New York

Keith, S. (1997), Component Based Development and Object Modeling, White Paper, *Sterling Software*

Alan, W. B. (1999), Moving from Component to CBD, *Sterling Softwarv*.

장경원

1996년 전남대 산업공학과 학사/석사

1998년 고등기술연구원 기술경영연구실 기술경영팀 연구원

현재: 아주대 산업공학과 박사과정

관심 분야: CPM, ERP/APS/SCM EAI(Enterprise Application Intergration

여성주

1999년 아주대 산업공학과 학사

현재: 아주대 산업공학과 석사

관심 분야: SCM, 분산객체, JAVA&CORBA

최정원

1999년 아주대 산업공학과 학사

현재: 아주대 산업공학과 석사

관심 분야: SCM, DB응용, Component Based Application

왕지남

1983년 아주대 산업공학과 학사

1985년 한국과학기술원 산업공학과 석사

1985년 미 Texas A&M Univ 산업공학과 박사

현재: 아주대 기계 및 산업공학부 부교수

관심 분야: Neural Network, Component Based S/W개발