IE Interfaces
Vol. 16, No. 2, pp. 268-279, June 2003.

Visual Basic과 UML을 사용한 애플리케이션 개발시의 핵심적 단계: 주문처리 업무를 중심으로

한 용 호†

부산외국어대학교 정보시스템학과

Critical Steps in Building Applications with Visual Basic and UML: Focusing on Order Processing Application

Yongho Han

Department of Information Systems, Pusan University of Foreign Studies, Busan, 608-738

This paper presents critical steps in building client/server application with UML and Visual Basic, which are derived from the implementation case of a typical order processing system. To begin with, we briefly review the software architecture, the diagrams and the object-oriened building process in the UML. In the inception phase, it is critical to define the project charter, to draw use case diagrams, and to construct a preliminary architecture of the application. In the elaboration phase, it is critical to identify classes to be displayed in the class diagram, to develop user interface prototypes for each use case, to construct sequence diagram for each use case, and finally to design an implementation architecture. Steps to construct implementation architecture are given. In the construction phase, it is critical to design both the database and components. Steps to design these components are described in detail. Additionally the way to create the Internet interface is suggested.

Keywords: application development, UML, visual basic

1. 서 론

인터넷 기술과 객체지향 패러다임이 정보기술의 여러 부분에 적용되면서, 분산 컴퓨팅 환경에서의 클라이언트/서버 기술은 분산 객체 컴퓨팅으로 전환되고 있다. 분산 객체 컴퓨팅은 단순한 기술만의 변화를 넘어서서, 오늘날의 컴퓨팅 환경의 문제를 해결하기 위한 패러다임이며, 소프트웨어 공학 프로세스를 확장 적용시키는 촉진제의 역할을 수행한다. 이러한 소프트웨어 개발기술의 성공요소는 재사용 아키텍처를 기반으로 하는 객체지향 기술로서, 이에 대한 효율적인 활용전략이 주요 이슈로 대두되고 있다. 최근에는 객체지향 기술에 근간을 둔 컴포넌트 기반의 소프트웨어 개발기술 (CBD: Component Based Development)이 정립되어 가고 있으며, 또한 기술적 컴포넌트 프레임워크를 제공하는 업체 간 표준전략은 객체기술과 인터넷 기술이 접목된 마이크로소프트사의 COM, OMG의 CORBA, SUN사의 Enterprise JavaBeans로, 서로 보완적인 관계를 유지하면서 발전하고 있다. CBD는 비즈니스 객체를 통한 업무의 재사용까지도 의미하고 있다. 지금까지 많은 방법론이 사용되어 왔지만, 같은 방법론이라 할지라도 모델과 표현 방법이 서로 달라 어려움을 겪어 오던 중, 1997년 OMG (Object Management Group)에서 UML(Unified Modeling Language)을 객체지향 방법론의 표준으로 제정하기에 이르렀다.

†연락저자 : 한용호 교수, 608-738 부산시 남구 우암동 55-1 부산외국어대학교 정보시스템학과, Fax : (051)640-3375,
e-mail : yhhan@pufs.ac.kr
2002년 11월 접수, 1회 수정 후 2003년 3월 게재 승인

UML을 이용하여 애플리케이션을 성공적으로 개발하기 위해서는, 개발의 각 단계마다 프로젝트의 성공에 큰 영향을 미치는 요소들을 파악하는 것이 도움이 될 것이다. 그러한 요소들을 파악하기 위해서는 애플리케이션 개발 과정을 소개하는 문헌이나 애플리케이션 개발 사례를 소개하는 연구 결과들을 참조할 수 있다. 애플리케이션 개발 과정을 소개하는 일부 문헌(Rumbaugh *et al*. (1999); Gomaa (2000); Ryu *et al*. (2000))에서는 부분적으로는 사례를 들어 구체적으로 설명하고 있지만, 전체적으로는 여러 애플리케이션에 공통적으로 적용될 수 있도록 원론적인 설명에 치중하고 있다. 한편, 다른 종류의 문헌(Reed, 2000)에서는 한 애플리케이션만의 개발 과정을 자세히 소개하고 있는 관계로, 그 애플리케이션에만 한정되고 다른 애플리케이션에는 적용될 수 없는 내용을 많이 포함하고 있다.

따라서 본 연구에서는 이 두 접근방법의 절충적인 접근방법으로서 UML과 Visual Basic을 사용한 고객의 주문처리 애플리케이션의 구체적 구축 사례를 통해서, 일반적인 클라이언트/서버 애플리케이션의 개발에 공통적으로 적용될 수 있는 구체적 요소들을 개발 단계별로 파악하고자 한다. 본 연구를 통해 파악된 요소들은 UML과 연결될 때, C++, Java 등의 다른 언어의 사용 시에도 활용될 수 있다.

본 연구의 2장에서는 소프트웨어 아키텍처, UML 다이어그램 및 객체지향 개발 프로세스 등 UML의 개요를 소개한다. 3장에서는 프로젝트 기술서의 작성, use case의 작성 등 개념 단계를 설명한다. 4장에서는 클래스의 파악, 유저 인터페이스 프로토타입의 개발, sequence 다이어그램의 작성, 기술적 조망 등 정제 단계를 설명한다. 5장에서는 데이터베이스 및 컴포넌트의 설계 및 구축, UML 클래스 다이어그램으로부터 코드의 생성, DCOM를 이용한 분산 실행, 그리고 인터넷 인터페이스 등 구축 단계를 설명한다. 마지막으로 6장에서는 결론을 서술한다.

2. UML의 개요

UML은 애플리케이션을 시각화하고, 기술하고, 구축하며 또한 산출물들을 문서화하는 데 사용되어지는 모델링 언어이다. UML은 다음과 같은 특징들을 가지고 있다.

(1) 표준화, 분산화, 단순화, 부품화 등 소프트웨어 개발상의 쟁점들을 해결하고 있다.
(2) 통합된 표기법을 제공한다.
(3) 사용자의 요구분석에서부터 분석, 설계, 구현, 테스트까지의 작업 결과에 대해 시각적이면서도 투명한 산출물을 제공한다.
(4) 자체적인 개발 프로세스를 가지고 있지는 않지만, 애플리케이션 개발의 전체적 라이프 사이클을 통해 UML으로의 접근이 가능하다.

UML의 애플리케이션 아키텍처, UML 다이어그램 및 객체지향 개발 프로세스에 대한 기본적 내용을 다음과 같이 소개한다.

2.1 애플리케이션 아키텍처

UML은 애플리케이션으로 구축될 대상 시스템에 대하여 "4+1" 뷰 아키텍처를 제공한다. 즉, 대상 시스템을 <그림 1>과 같이 유즈 케이스 뷰(use case view), 논리적 뷰(logical view), 구현 뷰(Implementation view), 프로세스 뷰(process view) 및 배치 뷰(deployment view)의 5개 부분으로 구성하고 구조화시킨다.

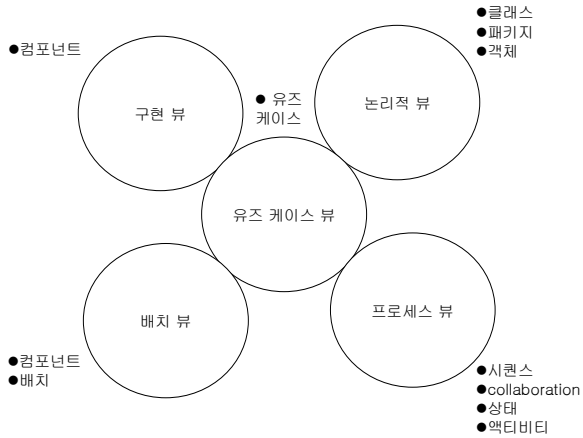

그림 1. 애플리케이션 아키텍처의 "4+1" 뷰.

(1) use case 뷰는 대상 시스템 외부에서 인식되는 시스템의 기능을 표현한다. 기술 중립적이며, 시스템 솔루션의 방법(how) 대신, 내용(what)의 표현에 중점을 둔다.
(2) 논리적 뷰는 대상 시스템 내부의 정적인 구조, 즉 클래스, 객체 및 이들 사이의 관계들을 표현한다. 논리적 뷰는 패키지 다이어그램, 클래스 다이어그램 및 객체 다이어그램을 통하여 나타난다.
(3) 구현 뷰는 실행 모듈들과 그들 사이의 의존관계를 나타낸다. 모든 요구사항들이 최종적으로 코드로 구현되는 것을 보장하기 위하여, 이 모듈들은 산출물들을 검증한다. 구현 뷰는 컴포넌트 다이어그램을 통하여 표현된다.
(4) 프로세스 뷰는 대상 시스템을 프로세스(process)와 스레드(thread)들로 분할하여 나타낸다. 이러한 분할로 인해, 효과적인 자원의 사용, 병렬 실행, 및 비동기적 이벤트들의 처리가 가능해진다. 프로세스 뷰는 컴포넌트 다이어그램, deployment 다이어그램, sequence 다이어그램, collaboration 다이어그램, statechart 다이어그램, 및 activity 다이어그램 등으로 나타낼 수 있다.
(5) 배치 뷰는 컴포넌트 다이어그램과 배치 다이어그램을 통하여 시스템의 물리적 배치와 실행시의 컴포넌트의 배치를 나타낸다.

이러한 애플리케이션 아키텍처는 구축되는 시스템에 대한 서로 다른 관점들을 조절하고 관리하는 데 사용되며, 프로젝트에 관련된 모든 사람들 사이에 의사 소통의 수단으로 사용된다.

2.2 UML 다이어그램

UML은 클래스 다이어그램, use case 다이어그램, collaboration 다이어그램, sequence 다이어그램, 컴포넌트 다이어그램, statechart 다이어그램, 배치 다이어그램, activity 다이어그램으로 구성된다(Rational, 2000). 이들 다이어그램은 정적 다이어그램 및 동적 다이어그램으로 구분해 볼 수 있다(Rumbaugh, 2000).

(1) 정적 다이어그램은 시스템의 구조 및 기능성을 나타내는 다이어그램으로 클래스 다이어그램, use case 다이어그램, component 다이어그램, 배치 다이어그램이 여기에 속한다.

(2) 동적 다이어그램은 시간의 흐름에 따라 정적 다이어그램 내 구성 요소들 사이의 상호작용을 나타낸다. statechart 다이어그램, activity 다이어그램, sequence 다이어그램, collaboration 다이어그램이 여기에 속한다.

다른 다이어그램들에 비해 클래스 다이어그램, sequence 다이어그램, 컴포넌트 다이어그램 및 배치 다이어그램 등이 상대적으로 많이 사용되고 있다.

2.3 객체지향 개발 프로세스

UML에 고유한 개발 프로세스는 아직 정의되어 있지 않은 상태이며, 특정한 프로세스의 사용도 지정하고 있지 않다. 오늘날 객체지향 프로젝트에서 RUP (Rational Unified Process)라는 애플리케이션 개발 프로세스의 방법론이 UML과 함께 많이 사용되고 있기 때문에, 본 연구에서는 이 RUP를 사용하였다. RUP에서의 개발 프로세스를 <그림 2>와 같이 나타낼 수 있다(Ryu *et al*., 2000).

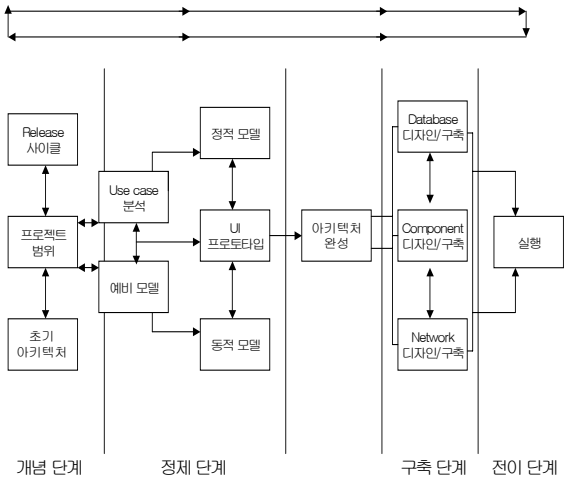

그림 2. 객체지향 개발 프로세스 모델.

3. 개념 단계

개념 단계에서는 사용자의 요구사항을 바탕으로 프로젝트 기술서 및 use case를 작성해야 한다.

3.1 프로젝트 기술서의 작성

본 연구의 대상이 되는 한 가상적인 기업(이후로는 A사로 칭함)은 웹을 포함하여 원격지에서 A사의 제품에 대한 주문을 낼 수 있도록 허용해 주는 주문처리 클라이언트/서버 애플리케이션의 개발을 원하고 있다. 본 연구에서는 Visual Basic과 UML을 사용하여 이 애플리케이션을 개발하고자 한다.

개념 단계는 프로젝트 수행과정의 첫 단계로서, 애플리케이션의 특성들을 파악하고 프로젝트의 범위를 설정하기 위하여 먼저 프로젝트 기술서를 작성해야 한다. 프로젝트 기술서는 프로젝트 안의 여러 이벤트들, use case들, 및 아키텍처의 구성요소들을 포함하며, 개념 단계의 주 산출물에 해당된다.

A사의 프로젝트 기술서의 내용 가운데 애플리케이션의 핵심을 이루게 될 액터(actor)들과 이벤트들을 각각 <표 1>과 <표 2>와 같이 나타낼 수 있다.

표 1. A사에 대한 액터들

액 터	정 의
Customer	제품에 대한 주문을 한다.
Supplier	이 기업에 부품을 공급한다.
Accouting System	시스템으로부터 회계정보를 입수한다.
Billing Clerk	재무적인 관점에서 모든 판매를 조정하고 추적한다.
Shipping Clerk	처리된 제품 주문들의 배송을 조정한다.
Packaging Clerk	발송할 제품을 준비한다.
Order Clerk	주문을 접수하고, 미처리 주문들을 관리한다.
Customer Service Clerk	고객들에게 서비스한다.
Manager	회사에 대한 주문들의 상태에 대한 보고서들을 요청한다.

3.2 use case의 작성

이제 <표 2>의 이벤트 테이블 내 각 이벤트들을 해당 use case에 할당한다. use case는 시스템의 구축 방법(how)이 아니라, 기능성(what)을 나타내는 도구로서, 목표지향적인 성격을 지니고 있다. use case는 기술 중립적이며, 어떠한 애플리케이션 아키텍처나 프로세스에도 적용할 수 있다. 또한 use case는 허브(hub)의 역할을 맡고 있어, 모든 요구사항들이 이로부터 파생되어 나온다. <표 2>의 이벤트 테이블 내 각각의 이벤트는 하나의 use case 안에 표시되어야 한다. 반면에 하나의 use case 안에

는 일반적으로 여러 개의 이벤트들이 표현될 수 있다.

<그림 3>은 A사의 use case 다이어그램을 나타낸다. 여기서 각 액터는 하나의 관계를 통해 다른 액터들과 연결된다. Accounting System 액터는 외부 시스템으로서, (인터페이스 스테레오타입을 나타내는) 박스 형태로 표시되었다. Customer Service Clerk를 Order Clerk으로 연결하는 화살표는 일반화(generalization) 관계를 나타낸다. 즉, Customer Service Clerk는 Order Clerk의 한 사람이라는 것을 의미한다. 마지막으로, Order Clerk에서 Customer까지 점선으로 된 화살표는 (Order Clerk이 Customer에게 의존한다는) 의존(dependency) 관계를 나타낸다. 각 이벤트가 use case 안에서 위치를 찾았는지의 여부를 파악함으로써 일관성(consistency)을 유지할 수 있다.

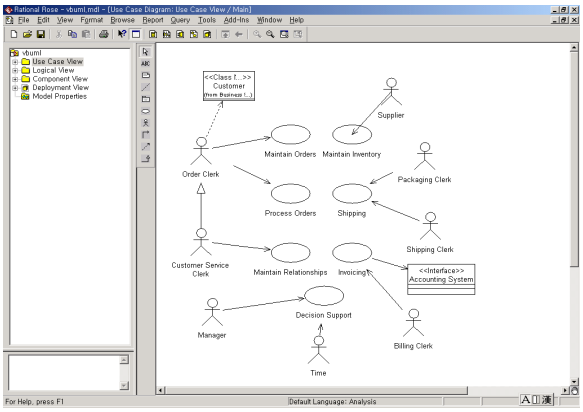

그림 3. use case 다이어그램.

다음은 정상(happy), 대체(alternate) 및 예외(exception)라는 세 가지 유형의 경로를 파악해야 한다. 정상 경로는 모든 것이 정상적으로 진행되었을 때 거치는 경로를 가리키며, 대체 경로는 최상은 아니지만, 양호한 것으로 간주되는 경로를 가리킨다. 예외 경로는 오류를 처리하기 위한 경로를 의미한다.

<표 3>, <표 4> 및 <표 5>는 각각 A사의 정상 경로, 대체 경로 및 예외 경로들 중의 일부를 나타낸 것이다. 각 use case 안에는 여러 개의 경로가 표시될 수 있다.

표 3. 정상 경로들

use case	정상 경로
Maintain Orders	고객이 주문상태에 관하여 질의한다.
Maintain Inventory	제품들이 구매명령서가 부착된 채, 창고에 도착한다.
Process Orders	고객이 제품을 주문하고, 신용카드로 지불한다.
Shipping	전체 주문이 현재의 재고로부터 고객에게 발송된다.
Invoicing	주문이 신용카드로 결재되었다는 내용을 고객에게 발송한다.
Maintain Relationships	고객이 배달 주소의 변경을 요청다.
Decision Support	관리자는 backorder 상태보고서를 요청한다.

표 2. 이벤트 테이블

주어	동사	목적어	빈도	도착 패턴	반 응
Customer	Places	Order	100/일	불규칙적	주문사항이 편집되고 시스템에 저장된다.
Shipping Clerk	Sends	Order	700/일	불규칙적	주문사항이 발송조건에 따라서 포장되고, 선적된다.
Customer	Buys	Warranty	60/일	불규칙적	주문사항이 계약조건과의 부합 여부를 검토하고, 기록된다.
Customer	Changes	Order	5/일	불규칙적	주문사항을 변경하고 기록한다.
Supplier	Sends	Inventory	5-10/일	불규칙적	새로운 재고가 체크인된다.
Customer	Cancels	Order	1/주	불규칙적	주문사항이 시스템에서 제거된다.
Time	Produces	Backorder Report	3/주	불규칙적	Report가 생성된다.
Time	Produces	Accounting Interfaces	1/주	불규칙적	시스템에 인터페이스한다.
Customer	Changes	Address	5/주	불규칙적	주소가 변경된다.
ServiceClerk	Prepares	Order	100/일	불규칙적	선적을 위해 패키지를 읽는다.
Manager	Inquires	Orders	5/일	불규칙적	요청을 처리한다.
Billing Clerk	Inquires Past Due	Invoice	10/일	불규칙적	청구서를 생성한다.

표 4. 대체 경로들

use case	대체 경로
Maintain Orders	고객이 이전에 주문한 제품의 수량 변경을 요청한다. 고객이 주문의 취소를 요청한다. 고객이 주문에 새로운 항목의 추가를 요청한다. 고객이 대금결재방법의 변경을 요청한다.
Maintain Inventory	제품이 구매명령서를 부착한 채 창고에 도착하였으나, 주문받은 제품으로서는 아직 불충분하다. 재고품을 보충하기 위하여 제품을 주문한다. backorder를 채우기 위하여 제품을 주문한다.

표 5. 예외 경로들

use case	예외 경로들
Maintain Orders	고객이 시스템에서 존재하지 않는 주문의 취소를 요청한다. 고객이 주문의 변경을 요청하였으나, 추가될 제품이 시스템에 존재하지 않는다.
Maintain Inventory	제품이 구매명령서 또는 선적 명세가 붙지 않은 채 도착한다.

이상 살펴본 바와 같이 use case는 일반적으로 비즈니스의 관점에서 파악된다. 한편, 대부분의 애플리케이션에서 use case로서의 모든 요건들을 만족하면서도, 특히 IT 전문가들에게 큰 의미를 지닌 특수한 use case들도 존재한다. 이러한 use case를 shadow use case라 부르며, 그 예로서는 보안, 감사, 기록보존 및 애플리케이션 인프라 구조 등을 들 수 있다.

표 6. use case "Process Orders"에 대한 정상 경로의 구체적 단계

1. 고객이 고객 번호를 제시한다.
2. 고객이 인식된다.
3. 고객이 원하는 각 제품에 대하여
   3.1 제품 ID를 요청한다.
   3.2 필요하다면 ID에 서술을 붙인다.
   3.3 양을 요청한다.
   3.4 항목 가격을 계산한다.
4. 총 주문합계를 계산한다.
5. 세금을 적용한다.
6. 운송비를 적용한다.
7. 총 합계를 고객에게 제시한다.
8. 고객이 신용카드 번호를 제시한다.
9. 고객의 신용카드를 확인한다.
10. 재고량을 감소시킨다.
11. 판매를 종료한다.

이상과 같이 모든 use case들을 정의한 후, 각 경로의 기능을 실행하는 데 필요한 단계들을 더욱 구체적으로 표현한다. 이 과정은 how의 관점이 아니라, what의 관점에서 수행된다.

예를 들어, use case "Process Orders"에 대한 정상 경로의 구체적인 단계를 <표 6>과 같이 나타낼 수 있다. use case "Process Orders"에 대해 표본적으로 수행한 이상의 과정을 <표 7>에 표시한 바와 같이 각 단계 안의 모든 use case들에 대하여 동일한 방법으로 수행한다.

개념 단계의 마지막으로 애플리케이션의 예비적인 아키텍처를 구성한다. <표 8>은 A사에 대한 주문처리 애플리케이션의 예비적인 아키텍처를 요약한 것이다.

표 7. 단계별 use case 리스트

단계 1	Process Orders
	Maintain Orders
	Maintain Relationships
	Application Infrastructure
단계 2	Maintain Inventory
	Shipping
	Invoicing
	Maintain Relationships
단계 3	의사결정 지원
	보안
	감사
	기록보존

표 8. Process Orders의 예비 아키텍처

컴포넌트	실 행
하드웨어: 클라이언트	166-MHz Pentium (128MB RAM, 8GB Hard Disk)
하드웨어: 서버	Dual CPU 400-MHz Pentium 기반 서버 (256MB RAM 및 60GB 저장능력을 지원하는 RAID-5 I/O subsystem)
소프트웨어: OS (서버)	Windows 2000 server
소프트웨어: OS (클라이언트)	Windows 98
소프트웨어: 애플리케이션 (클라이언트)	Visual Basic 6.0 Enterprise Edition
소프트웨어: 데이터베이스 (서버)	Microsoft SQL Server 2000
소프트웨어: 웹 (서버)	Microsoft IIS 5.0
소프트웨어: 웹 인터페이스 (서버)	Active Server Pages
소프트웨어: 비주얼 모델링	Rational Rose Enterprise Edition
프로토콜: 네트워크	TCP/IP
프로토콜: 데이터베이스	ADO

4. 정제 단계

정제 단계에서는 모든 클래스를 파악하고, 유저 인터페이스 프로토타입을 개발하고, sequence 다이어그램을 작성한 후, 마지막으로 실행 아키텍처를 구성해야 한다.

4.1 클래스의 파악

먼저 각 use case들 안의 모든 정상 경로, 대체 경로 및 예외 경로들의 내용을 더욱 구체화시킨다. 이를 위해 3절의 개념 단계에서 드러난 business rule들을 정의한다. 일반적으로 business rule들은 그 성격에 따라 다음과 같이 몇 개의 유형으로 나뉘어 진다.

(1) 정형적 사실: 어떤 사실(또는 조건)들이 충족되기를 요한다. 예를 들어, 각 주문은 주문접수 날짜가 있어야 한다. 각 고객은 신용상태가 승인되어야 한다.

(2) 행동 제약: 어떤 조건에 기초하여, 하나 이상의 행동을 금지시킨다. 예를 들어, 한 고객이 C급 평가를 받았으면, 그 고객을 비즈니스 고객 대상에서 제외시킨다.

(3) 행동 유발: 어떤 조건이 충족될 때, 하나 이상의 행동을 유발시킨다. 예를 들어, 어떤 invoice가 예정일까지 지불되지 않을 때, 1%의 벌과금이 부과된다.

(4) 추론: 어떤 사실이 충족되면, 하나의 결론이 유추되거나 새로운 지식이 알려진다. 예를 들어, 한 명의 고객이 가입되면, 그 사람은 비즈니스 고객으로 변하게 된다.

(5) 유도: 계산을 수행한다. 예를 들어, 총 가격은 지난 모든 주문에 현 주문으로 생긴 추가금액을 더한 것이 된다.

이러한 분류 체계를 이용하면 비즈니스 룰들을 쉽고 빠르게 파악할 수 있다. 파악된 각각의 비즈니스 룰은 연관된 use case의 템플릿 안에 하나의 항목으로 정의되어야 한다. 정의된 비즈니스 룰은 나중에 클래스의 속성이나 연산, 또는 클래스 사이의 연관관계 등으로 표현된다.

표 9. A사의 클래스들

엔티티	· Address	· Customer	· Order
	· Invoice	· Order Header	· Order Line
	· Order Summary	· Payment	· Product
	· Shipment	· Supplies	· Sheet Music
인터페이스	· Maintain Order Panel	· Inquire Order Panel	· Process Order Panel
	· Maintain Relationships Panel	· Credit Card Interface	· Accounting System Interface
컨트롤	· Process Orders Controller	· Maintain Orders Controller	· Maintain Relationships Controller
	· Infrastructure Controller		

이제까지 정의된 use case들을 기초로 하여 클래스들을 정의하고, 이것들을 클래스 다이어그램에 표현해야 한다. 클래스 다이어그램은 UML에서 가장 중요한 다이어그램에 해당하며, 이후에 작성될 컴포넌트 다이어그램과 배치 다이어그램에까지 궁극적인 영향을 미친다. 전체 클래스들을 엔티티, 인터페이스 및 컨트롤의 세 가지 유형(스테레오 타입)으로 분류할 수 있다(Jacobson, 1994). 이 기준에 따라 A 기업의 클래스들은 <표 9>와 같이 분류될 수 있다.

다음으로 연관관계(association), 일반화관계(generalization), 및 종속관계(dependency) 등 클래스들 간의 관계들을 정의한다. <그림 4>는 <표 7> 내 단계 1의 use case들에 대응하는 비즈니스 서비스를 Rational Rose의 클래스 다이어그램으로 나타낸 것이다.

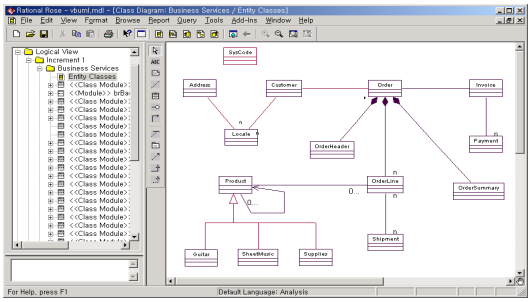

그림 4. 클래스 다이어그램.

4.2 유저 인터페이스 프로토타입의 개발

유저 인터페이스 프로토타입은 액터와 user case 사이의 경계를 분명히 하고, 잘못된 개념이나 누락된 사항 등이 드러나게 해준다. 그리고 애플리케이션이 제공할 인터페이스의 특징을 조기에 피드백시켜 준다. 또한 use case들을 더욱 자세히 표현해 준다.

A 기업의 경우 <그림 5>는 주문 처리 애플리케이션에 대한 전체적인 화면 구조도를 나타내며, <그림 6>은 "Process Orders" use case에 대한 화면 구조도를 나타낸다. 이 구조도의 내용에 따라 Visual Basic을 사용하여 Process Orders 폼 등을 디자인하게 된다.

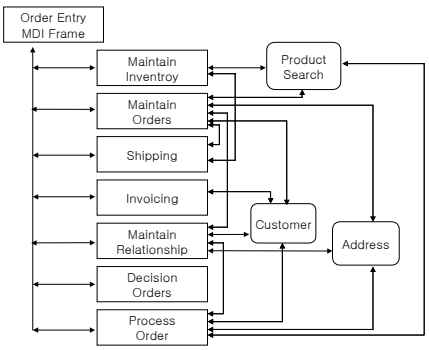

그림 5. 전체 화면 구조도.

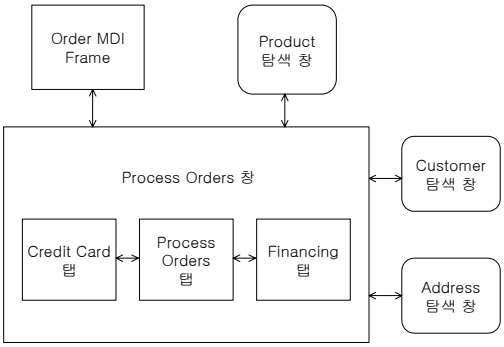

그림 6. Process Orders 화면 구조도.

4.3 sequence 다이어그램의 작성

이제 애플리케이션 도메인의 동적 뷰를 모델링한다. 이를 위하여 UML로부터 sequence 다이어그램, collaboration 다이어그램, statechart 다이어그램, activity 다이어그램의 4가지 다이어그램을 이용할 수 있다. 일반적으로 sequence 다이어그램이 많이 사용되고 있다. <그림 7>은 A 기업의 use case "Process Orders"의 정상 경로를 Rational Rose의 sequence 다이어그램으로 나타낸 것이다.

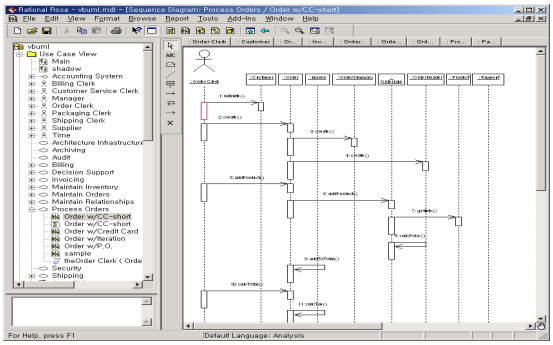

그림 7. sequece 다이어그램.

4.4 실행 아키텍처의 구성

이제 <표 8>의 예비 아키텍처를 기초로 하여 실행 아키텍처를 구성해야 한다. 실행 아키텍처는 세 가지 종류의 기술로 구성된다.

(1) 개발 기술: 애플리케이션을 구성하는 데 필요한 여러 툴들을 의미한다. 이 툴들의 기능으로는 데이터베이스 기술, 애플리케이션 구축 도구, 소스 제어, configuration 관리 및 소프트웨어 배분 등을 들 수 있다.

(2) Data Access 기술: 애플리케이션에서 데이터에 접근하는 방법을 가리킨다. 데이터베이스 복제 기술 및 Data Access 인프라 (데이터베이스 API 및 ODBC)가 이에 포함된다.

(3) 애플리케이션 기술: 애플리케이션의 여러 계층들을 분리하는 층화(layering) 전략을 포함하여, 애플리케이션의 분할방법과 계층들의 관리방법을 가리킨다.

A사에 대한 실행 아키텍처는 다음 단계들을 거쳐 구성한다.

[단계 1] 애플리케이션의 논리적 서비스를 <표 10>과 같이 세 개의 계층으로 분할한다.

표 10. 논리적 계층들의 분할

계 층	범 위	목 적
Presentation 서비스	· 데이터 표현 · 데이터 접수 · GUI	· 사용의 편이성 · 자연적, 직관적 유저 인터페이스 · 빠른 반응시간
Business 서비스	· 핵심 비즈니스 룰 · 애플리케이션/대화 흐름 제어 · 데이터 무결성 확인	· business rule의 엄격한 집행 · 경제적인 코드 · 유지 비용의 감소
Data 서비스	· 내구성있는 데이터 저장/검색 · API들을 통한 DBMS 접근 · 동시성 확보	· 일관성 및 신뢰성 있고, 안전한 DB · 정보 공유 · 빠른 반응시간

[단계 2] 세 개의 논리적 계층으로부터 <그림 8>과 같이 두 개의 물리적 계층을 구성한다.

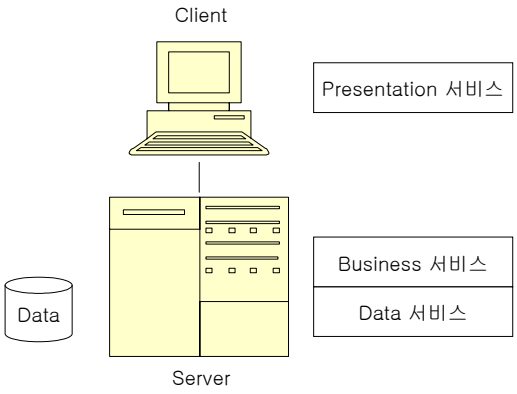

그림 8. 논리적 계층과 물리적 계층.

[단계 3] 계층을 세분화시킨다. Business 서비스 계층을 Business Context 및 Business Rule의 두 가지 유형의 서비스로 세분한다. 그리고, Data 서비스 계층을 Data Translation, Data Access 및 Database의 세 유형의 서비스로 세분한다. 그 결과 전체 애플리케이션의 논리적 기능은 <표 11>과 같이 6 계층의 모형으로 나타난다. 이 모형은 다른 애플리케이션에도 그대로 사용될 수 있다.

[단계 4] 이 6 개 계층 사이에서 사용할 프로세스 간 통신방식을 정한다. RPC와 CORBA를 포함하여 몇 가지 대안들을 생

각해 볼 수 있으나, A사의 경우 COM/DCOM을 사용하기로 한다. COM/DCOM을 실행하는 컴포넌트를 만들기 위하여, Visual Basic에서 ActiveX DLL 또는 ActiveX EXE 형태의 프로젝트 유형을 사용한다. Presentation 서비스 계층은 표준 EXE (VB 폼) 및 웹 front-end의 두 방법을 모두 사용할 것이며, 나머지 각 계층들은 ActiveX DLL 또는 ActiveX EXE 형태의 컴포넌트를 취한다.

표 11. 논리적 6 계층 모형

Presentation 서비스	· 유저 인터페이스 기술(Microsoft Windows)을 제공한다.
Business Context 서비스	· syntax 및 context 편집 작업을 수행한다.
Business Rule 서비스	· business rule을 실행한다.
Data Translation 서비스	· 비즈니스 계층의 요청을 적절한 언어(예:SQL)로 전환시킨다.
Data Access 서비스	· 액세스 API (예: ADO)를 통해 적절한 언어(예: SQL)를 전달한다.
Database 서비스	· 물리적 데이터베이스 기술을 나타낸다. (예: SQL Server)

[단계 5] 인터넷을 포함시킨다. "Process Order" use case의 정상 경로 중의 주문조회 기능을 인터넷을 통하여 수행할 수 있도록 만든다. 이 기능을 구현하기 위하여, 웹 페이지와 Business 서비스 계층 사이에 일종의 인터페이스가 필요하다. 이 인터페이스는 IIS 상에서 실행되는 Active Server Pages (ASP)로 구성키로 한다. 이 ASP는 인터페이스로서의 역할과 프로세스의 컨트롤러로서의 역할을 모두 수행하게 된다. 즉, 브라우저로부터의 요청을 웹 서버에 전달하고, 반응 HTML 페이지를 만드는 데 필요한 객체를 생성하고 이 객체를 통해 브라우저로 메시지를 전달한다.

이상의 과정을 거쳐 구성되는 A사의 실행 아키텍처를 <표 12>와 같이 나타낼 수 있다.

표 12. 실행 아키텍처

Presentation 서비스	· VB 표준 EXE · HTML 및 script 폼
Business Context 서비스	· VB 표준 EXE · Active Server Pages (인터넷)
Business Rule 서비스	· VB ActiveX DLL 또는 EXE
Data Translation 서비스	· VB ActiveX DLL 또는 EXE · SQL
Data Access 서비스	· VB ActiveX DLL 또는 EXE · ADO 및 OLE/DB
Database 서비스	· SQL Server

5. 구축 단계

구축 단계에서는 데이터베이스 및 컴포넌트의 설계 및 구축, 클래스 다이어그램으로부터 코드의 생성, COM/DCOM을 이용한 분산 실행, 그리고 인터넷 인터페이스의 구축 등을 수행한다.

5.1 데이터베이스의 설계 및 구축

이 단계에서는 객체지향 개념을 지닌 UML 모델로부터, 객체지향 개념을 가지지 않는 관계형 DB를 생성한다. 이를 위해 각 클래스의 모든 속성에 적절한 데이터형을 지정해야 하며, 다음과 같은 과정을 거쳐야 한다.

[단계 1] 클래스로부터 테이블을 생성한다. 클래스 다이어그램으로부터 관계형 DB를 생성하기 위한 가장 간단한 방법은 각 클래스에 대응하는 테이블을 1:1로 생성하는 것이다. 그러나, 이 방법을 사용하면, 너무 많은 수의 테이블이 생성되고, join 연산이 너무 많이 사용되고, 데이터의 비정규화 등의 문제 등이 야기된다. 따라서 이보다 더 좋은 방법으로서, use case들의 경로를 sequence 다이어그램 상에서 관찰하여(예를 들어, Order 객체, OrderHeader 객체, OrderSummary 객체 순으로 Customer가 많은 수의 메시지를 전달하고 있는 사실을 발견함으로써) 이러한 객체 간 상호작용의 빈도를 효율적으로 지원할 수 있는 방향으로 관계형 DB의 물리적 구조를 구성할 수 있다.

[단계 2] 클래스들의 종속 관계, 연관 관계, 포함/혼합 관계, 및 상속 관계를 관계형 DB에 대응시킨다.

[단계 3] 키(key)의 구조를 결정하고 정규화를 실시한다.

[단계 4] 비주얼 모델링 툴을 사용하여 DDL(Data Definition Language)을 생성한다. 비주얼 모델링 툴은 UML과 프로그래밍 언어를 통합하는 데 필수적이다. A사의 경우 SQL DDL을 사용하여 테이블 구조를 생성하기 위한 툴로서 Rational Rose를 사용하였다.

[단계 5] Data Translation 서비스 계층을 설계한다. 이를 위해 두 가지 방법을 고려할 수 있다. 첫번째 방법은 각각의 논리적 서비스 요청(예를 들어, 지정일 이후에 들어온 모든 주문들의 검색)에 대응하여 하나의 public 연산을 가지는 클래스를 만드는 것이다. 두 번째 방법은 각 비즈니스 클래스(예: Customer)에 대하여 데이터 변환 클래스(예: CustomerDT)를 하나씩 만드는 것이다. 첫번째 방법은 시스템 내에 존재하는 클래스의 수가 적게 되는 반면, 각 클래스가 매우 무거워지는 단점이 있다. 두 번째 방법은 높은 응집력을 지니며 상대적으로 가벼운 클래스들이 생성되지만, 클래스의 수가 많아지는 단점이 있다. A사에 대하여 융통성과 향후의 확장성을 고려하여, 두 번째 방법을 선택한다. 이에 따라 Data Translation 서비스 컴포넌트로서의 역할을 수행하기 위하여, 여러 가지 비즈니스 클래스들을 그 내부에 포함하는 하나의 ActiveX DLL을 만든다.

[단계 6] Data Access 서비스 계층을 설계한다. Data Access 서

비스 계층의 ActiveX 컴포넌트 DASVC.DLL를 만들어 DARetrieve (select) 및 DAQuery(update, delete, insert)의 작업을 담당토록 한다.

이렇게 설계함으로써 전체 시스템에서 단지 하나의 클래스만이 데이터베이스 API의 호출을 전담하게 되어, 향후 기술적인 내용이 변화하더라도 이 두 작업이 위치하는 곳의 API 액세스 로직만 재작성함으로써 쉽게 대처할 수 있다. 두 계층의 논리적 서비스는 <표 13>과 같이 요약될 수 있다.

표 13. Data Translation 서비스 계층

Data Translation 서비스 (DTSVC.DLL)	· 비즈니스 클래스와 1:1로 대응하는 translation 클래스(CustomerDT, OrderDT 등)를 가지고 있다. · SQL문을 만들어 Data Access 서비스 계층에 전달한다.
Data Access 서비스 (DASVC.DLL)	· ADO를 이용하는 SQL 액세스 작업(DARetrieve 및 DAQuery)을 수행한다. · DARetrieve를 통한 요청에 대한 질의 결과를 문자열 배열 형태로 반환한다.

5.2 컴포넌트의 설계

여기서는 첫째, Visual Basic 폼과 같은 사용자 인터페이스와, 둘째, 폼 위의 입력내용을 비즈니스 계층에 전달하는 기능을 분리시킨 애플리케이션 아키텍처를 제시한다. 이 아키텍처는 강력한 실행을 보장하며, 동시에 비즈니스의 확장에 따라 이 아키텍처도 확장이 가능하다는 장점을 지니고 있다.

표 14. Presentation, Business Context 및 Business Rule 서비스 계층

Presentation 서비스 EXE(VB 폼), 브라우저	· 애플리케이션의 시각적 요소들을 지닌다. · VB 폼 또는 인터넷 브라우저를 많이 사용한다. · 입력 사항을 다른 계층에 전달하기 전에, 그것의 syntax 및 context 편집 기능을 지닌다.
Business Context 서비스 EXE(VB 폼), 브라우저	
Business Rule 서비스 (BRSVC.EXE)	· (Customer, Order 등의) 비즈니스 클래스들을 포함한다. · workflow의 상당부분을 다룬다. 그리고 사용하기 쉬운 객체 인터페이스를 유저 인터페이스에게 제공한다. · Data Translation 서비스 계층에 있는 대응되는 클래스들과 통신한다.

<표 12>의 각 계층들 중 적어도 presentation 서비스, business context 서비스, 그리고 일부 business rule 서비스의 3개 서비스는 클라이언트 상에 존재할 수 있다. 단, business context 서비스는 VBScript 또는 JavaScript의 형태로 클라이언트(브라우저) 상에서 실행될 수도 있으며, 또한 웹 브라우저라는 presentation 서비스 기술을 사용할 경우, ASP와 JavaScript의 형태로 웹 서버에 존재할 수도 있다. <표 14>는 본 절에서 살펴볼 계층들과 그 역할을 나타낸 것이다.

(1) Presentation 서비스 계층: presentation 서비스 계층은 새로운 기술의 등장과 사용자의 요구로 인하여 모든 계층 가운데 가장 변화가 많은 계층에 해당한다. 지속적으로 변화하는 presentation 방법에 대한 요구사항을 수용하고, 확장성이 더 좋은 디자인을 제공하기 위하여, model-view-controller (MVC) 프레임워크를 이용할 수 있다.

(ⅰ) model: MVC 프레임워크 안에서 model은 Customer, Order와 같은 객체 클래스를 나타낸다. model을 사용함으로써 프레임워크를 사용자 인터페이스 정보가 전혀 없는 상태로 유지할 수 있다. model은 ActiveX 컴포넌트 BRSVC.EXE 안에 들어있는 VB 클래스 모듈의 형태로 애플리케이션 안에서 표현된다.

(ⅱ) view: MVC 프레임워크 안에서 view는 클라이언트에 전달되어 화면에 나타나는 인터페이스를 나타낸다. view는 표준 실행 파일 UISVC.EXE 안에 들어있는 VB 폼들을 통하여 애플리케이션 안에서 표현된다.

(ⅲ) controller: model과 view 사이에서 작동하는 서비스이다. view와 controller 사이에는 1:1의 관계가 있다. 그래서 생성되는 각 VB 폼마다 이에 대응하는 controller로서 하나씩의 VB 클래스 모듈이 존재하게 된다. controller가 view(폼)와 model(객체 클래스) 사이에서 요구사항들을 중개하는 기능을 수행함으로써, 사용자 인터페이스가 가벼워지는 장점이 있다.

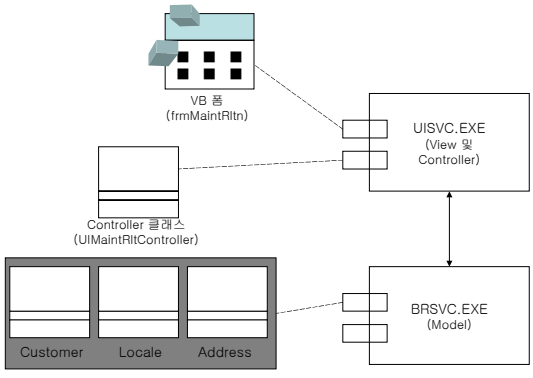

그림 9. MVC 프레임워크.

<그림 9>는 A사의 use case "Maintain Relationships"에 대한 MVC 프레임워크를 나타낸 것이다. 폼 frmMainRltn이 UISVC.EXE 안에 존재하며, MVC 프레임워크의 view 기능을 담당한다. 클래스 모듈 UIMaintRltnController도 이 UISVC.EXE 안에 존재

하면서, MVC 프레임워크의 controller 기능을 수행한다. 모든 비즈니스 정보의 원천이 되는 객체 클래스들은 ActiveX 컴포넌트 BRSVC.EXE 안에서 클래스 모듈(Customer, Address, Locale)로서 실행된다.

(2) Business Context 서비스 계층: 이 서비스 계층은 사용되는 presentation 기술과 매우 밀접하게 연동되어 작동한다. 사용자가 인터페이스를 사용할 때, 이 서비스 계층은 편집기능의 상당 부분을 담당한다. 이 편집기능들은 UISVC.EXE 컴포넌트와 business rule 컴포넌트 (BRSVC.EXE)의 두 군데로 분산되어 구현된다. 이 서비스 계층은 시스템과 애플리케이션이 접촉하는 동안, 다음과 같은 편집기능을 수행한다.

(ⅰ) syntax 편집: 입력 데이터가 사용자 인터페이스를 떠나기 전, 그 정보를 formatting 및 cleansing 하는 기능을 수행한다. 이러한 편집기능의 예로서 수치 범위의 체크 및 날짜 형태의 확인 등을 들 수 있다.

(ⅱ) context 편집: 정보의 비즈니스 측면에서 편집기능을 수행한다. 예를 들어, 만약 어떤 고객이 신용 지불수단을 선택한 경우, 50 미만의 신용 점수를 가지고 있다면, 판매를 위해 재무 담당자의 승인이 필요하다. 이러한 유형의 룰은 business rule 컴포넌트 BRSVC.EXE 안에서 해당 클래스 안에서 하나의 연산으로 정의될 수 있다.

표 15. entity 클래스들의 공통된 연산

연산명	기 능
getInfo	객체의 인스턴스에 관한 정보를 검색한다.
update	객체의 인스턴스에 관한 정보를 수정한다.
delete	객체의 인스턴스에 관한 정보를 삭제한다.
Insert	객체의 인스턴스에 관한 정보를 삽입한다.
unPack	data translation 서비스 계층의 select 호출로부터 검색된 결과를 사용하여, 객체 멤버 변수를 만든다.

(3) Business Rule 서비스 계층: 이 서비스 계층 안에는 여러 entity 클래스들이 존재한다. 따라서 먼저, 이들 entity 클래스를 담고 있는 실행 컴포넌트는 Visual Basic의 Get 및 Let 연산들을 통해 클래스들의 각각의 속성들을 정의해야 한다. 그 다음, 각 entity 클래스가 제공할 서비스들을 정의해야 한다. <표 15>는 A사의 모든 entity 클래스에 공통적으로 요구되는 연산들을 나타낸 것이다.

business rule 서비스 계층 안의 business rule들은 서비스의 내용에 따라 두 가지로 나누어진다. 첫째는 ActiveX 컴포넌트 BRSVC.EXE 안에서 실행되는 룰로서, 워크플로 지향적인 성격이 강하다. 둘째는 data translation 서비스 계층에서 실행되는 룰로서, 데이터 관리의 기능을 수행한다.

이제 A사의 애플리케이션에 대한 인프라 구조를 <그림 10>과 같이 컴포넌트 다이어그램으로 나타낼 수 있다.

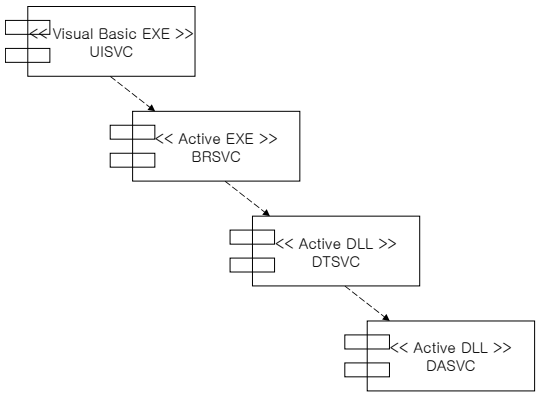

그림 10. 컴포넌트 다이어그램.

(ⅰ) 첫 번째 컴포넌트 UISVC.EXE는 Visual Basic의 표준 EXE 형태로서, 논리적 presentation 서비스 계층 및 business context 서비스 계층에 대응된다. 이 컴포넌트는 인접 컴포넌트인 BRSVC.EXE에 연관된다.

(ⅱ) 두 번째 컴포넌트 BRSVC.EXE는 business rule 서비스 계층에 대응되며, ActiveX EXE 형태이다. 이 컴포넌트는 서버 상에 배치되며, out-of-process 형태로 실행된다. 이 컴포넌트는 인접 컴포넌트인 DTSVC.DLL에 연관된다.

(ⅲ) 세 번째 컴포넌트 DTSVC.DLL은 ActiveX DLL 형태로 data translation 서비스 계층뿐만 아니라, business rule 서비스 계층의 일부 요소들과 대응된다. 각 entity 클래스에 대하여 이에 대응하는 하나씩의 translation 클래스를 포함한다. 이 컴포넌트는 SQL 문의 작성을 주기능으로 하며, 서버상에서 in-process로 실행되며, 다음 컴포넌트인 DASVC.DLL에 연관된다.

(ⅳ) 네 번째 컴포넌트 DASVC.DLL은 ActiveX DLL 형태로 Data Access 서비스를 수행하며, 서버 상에서 in-process로 실행된다.

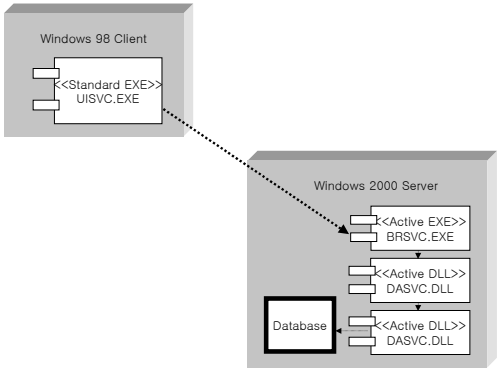

그림 11. DCOM을 사용한 컴포넌트의 배치

클라이언트/서버 시스템을 위한 배치전략으로서, 컴포넌트

UISVC.EXE는 각 클라이언트에, 그리고 나머지 컴포넌트들 BRSVC.EXE, DTSVC.DLL 및 DASVC.DLL는 서버 상에 배치하여 실행시킨다. DCOM을 사용한 컴포넌트의 배치전략을 <그림 11>과 같이 나타낼 수 있다. 여기서 데이터베이스 서버는 자체적인 프로세스로 실행된다.

5.3 클래스 다이어그램으로부터 코드의 생성

이제 Rational Rose에서 구축된 클래스 다이어그램으로부터 코드를 자동으로 생성해야 한다. 이를 위해 필요한 클래스들이 모두 존재하는지를 확인하고, 각 클래스의 연산, 속성들 및 각 연산에 대한 signature(이름, 파라미터들 및 반환 파라미터들)를 확인한다. 앞으로 reverse engineering을 수행할 기회가 남아 있기 때문에, 이 단계에서 모든 것이 완벽해야 할 필요는 없다. 여기서 reverse engineering이란 Rational Rose가 VB 프로젝트에서 생성된 코드를 기초로 하여, 클래스 다이어그램을 다시 수정하는 작업을 의미한다.

이제 Rational Rose 상의 클래스 다이어그램을 토대로 하여 VB 코드를 자동으로 생성한다. 프로토타입 안에 생성된 대부분의 코드는 버튼들이 클릭되었을 때, 화면이 바뀌고 제대로 작동할 수 있도록 하는 등의 기본적인 내용들을 담고 있다. 이후 이 코드로부터 시작하여 본래의 의도에 맞추어 코드를 채워나간다. 그 과정에서 필요시 reverse engineering을 적용하여 새로운 클래스 및 그 속성이나 연산들을 추가할 수도 있다.

5.4 DCOM를 이용한 분산 실행

UISVC.EXE 컴포넌트를 제외한 나머지 컴포넌트들은 서버(윈도우즈 2000 서버)에 배치한다. A사의 애플리케이션에 DCOM을 이용하자면, 클라이언트 측 소프트웨어에서 이에 대응하는 애플리케이션의 원격(remote) 부분이 어느 곳에 존재하는지를 알아야 한다. 이를 위해 Visual Studio Enterprise Tools의 원격 자동화 연결 관리자(RemAuto Connection Manager) 기능을 사용하여, 클라이언트에서 서버로의 접근을 허용하는 패키지를 만든 후, 이를 각 클라이언트에 설치하는 방법을 사용한다.

5.5 인터넷 인터페이스

인터넷을 통한 주문질의가 가능하도록 만들기 위하여, 웹 front-end를 구축하는 방법을 모색해야 한다. 인터넷 사용자가 해당 웹 사이트를 방문하여 주문 번호를 입력함으로써, 주문자의 이름, 고객 정보, 주문 품목, 주문 합계 등의 정보를 얻을 수 있게 만들 수 있다. 이를 구현하기 위해 다음과 같이 기존의 컴포넌트를 재구성하는 방법을 사용할 수 있다. 클라이언트의 UISVC.EXE 안에는 UIMaintRltnController 및 UIProcessOrderController와 같은 사용자 인터페이스 controller가 들어 있다. VB 폼들의 경우, 이러한 controller들이 클라이언트에 위치하여 필요한 정보에 빠르게 액세스할 수 있는 메커니즘을 제공한다. 그러나, 인터넷의 경우 브라우저가 클라이언트의 한 형태이다. 애플리케이션의 컴포넌트의 입장에서 볼 때, 클라이언트는 ASP(Active Server Page)가 된다. ASP는 브라우저 그룹과 애플리케이션의 컴포넌트들 사이에서 중개 역할을 수행한다.

애플리케이션의 컴포넌트가 UISVC.EXE 안에 존재하는 유저 인터페이스 controller를 사용할 수 있으면 좋을 것이다. 그러나, UISVC.EXE는 그 안에 폼들도 포함하고 있는 문제점을 지니고 있다. 따라서 컴포넌트 UISVC.EXE 안에 존재하는 클래스들만을 선택하여, 별도의 ActiveX DLL인 INET.DLL을 구성하는 방법을 사용키로 한다. <그림 12>는 ASP가 이 새로운 컴포넌트 INET.DLL 및 기존의 나머지 컴포넌트들과 어떻게 상호작용하는지를 보여주고 있다.

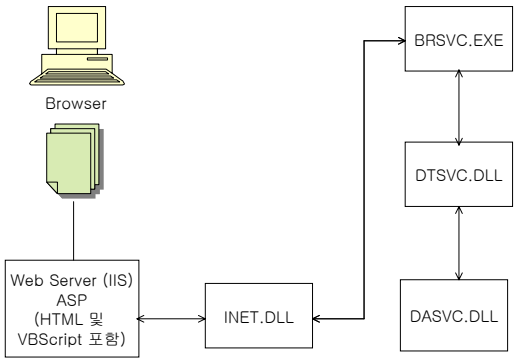

그림 12. ASP와 컴포넌트들.

INET.DLL을 생성하기 위하여 먼저 Visual Basic에서 새로운 프로젝트 inet.vbp를 생성하고, uisvc.vbp로부터 필요한 클래스들(Dictionary.cls, UIMaintRltnController.cls, UIProcessOrderController.cls, Utility.cls)을 이 프로젝트 안에 복사해 넣고 컴파일시킨다. 그리고 나서, 이 컴포넌트 INET.DLL과 상호작용할 HTML 폼과 ASP 파일들을 구성한다.

6. 결 론

UML과 Visual Basic 언어를 사용하여 다양한 영역에서의 애플리케이션들이 구축되고 있지만, 그 구축 과정 면에서 많은 공통점을 지니고 있다. 본 연구에서는 UML과 Visual Basic을 사용한 주문 처리 클라이언트/서버 애플리케이션의 구현 사례를 통하여, 다른 애플리케이션의 구축 과정에서도 충분히 적용될 수 있는 중점 사항들에 초점을 맞추어 개발 단계별로 소개하였다.

먼저 본 연구의 기반을 이루고 있는 UML 소프트웨어 아키텍처, UML 다이어그램, 및 객체지향 개발 프로세스 등에 관하여 간단히 소개하였다. 개념 단계에서는 사용자의 요구 사항을 바탕으로 한 프로젝트 기술서의 작성, use case 템플릿의 작성 그리고 예비 아키텍처의 구성을 강조하였다. 정제 단계에서는

클래스들의 정의, 유저 인터페이스 프로토타입의 개발, sequence 다이어그램의 작성 및 실행 아키텍처의 구성 과정을 예를 통하여 구체적으로 설명하였다. 구축 단계에서는 데이터베이스 및 각종 컴포넌트의 설계 및 구축 과정, 클래스 다이어그램으로부터 코드의 생성, DCOM을 이용한 분산 실행, 그리고 인터넷 인터페이스의 구축방법의 중요성을 파악하였다. 주문처리 애플리케이션의 개발 과정을 통해 파악되어 본 연구에서 제시된 단계별 사항들은 다른 애플리케이션의 개발시에도 대부분 그대로 적용될 수 있을 것이다.

본 연구에서는 Visual Basic을 사용하여 애플리케이션을 구축할 때의 필수불가결한 특징들을 찾아내는 데 초점을 맞추려고 시도하였다. 하지만 찾아낸 특징들을 표시할 수 있는 정형화된 툴이 존재하지 않는 관계로, 그 특징들을 서술, 그림 또는 표의 형태를 통해 나름대로 표현하였다. 이런 문제를 해결하기 위해 애플리케이션 개발상의 주요 특징을 체계적으로 나타낼 수 있는 방법론에 대한 연구가 향후 필요할 것으로 생각된다.

참고문헌

Rumbaugh, J., Jacobson, I., and Booch, G. (1999), *The Unified Modeling Language Reference Manual*, Addison Wesley.

Gomaa, Hassan (2000), *Designing Concurrent, Distributed, and Real-time Applications with UML*, Addison Wesley.

Reed, Paul R., Jr. (2000), *Developing Applications with Visual Basic and UML,* Addison Wesley.

Rational Software Corporation (2000), *Using Rose,* Rational Software Corporation.

Ryu, H. G., Lee S. C., Ryu, S. W., Shin, S. H. (2000), *UML based Object-oriented Client/Server Building*, Hongnung Science Publishing, Korea.

Quantrani, Terry (2000), *Visual Modeling with Rational Rose 2000 and UML,* Addison Wesley.

한 용 호

서울대학교 산업공학과 학사
한국과학기술원 산업공학과 석사
한국과학기술원 산업공학과 박사
현재: 부산외국어대학교 정보시스템학과 교수
관심분야: 비즈니스 프로세스, 워크플로, e-비즈니스