통합 객체 관리 모델 중점을 둔 JDBC기반의 AMOSS 구현

선수균*, 송영재**
*동원 대학 사무자동화과
**경희대학교 전자계산공학과

Implementation of AMOSS by Using JDBC - based on the Integration Object Management Model

Su-Kyun Sun* , Yong-Jea Song**
*Dept. of Office Automation, DongWon Collage, Korea
**Dept. of Computer Engineering, KyungHee University, Korea

요 약

최근 전산 환경은 통합되는 개방형 시스템으로 변모하고 있다. 서로 다른 platform을 기반으로 한 client들과의 연동을 위해서, 각 platform에 따른 application이 개발되어야 했다. 이러한 문제점을 극복하기 위해 이 기종간의 시스템을 통합할 수 있는 통합 Middlware의 선정이 필요하다. 따라서, 본 논문에서는 객체지향 소프트웨어 공학 프로세스에 의해 생성되는 산출물을 객체 형태로 통합 관리하고 객체들을 효율적으로 관리해 주는 통합 객체 관리 모델을 제안한다. 이 모델로 기존의 시스템을 재사용하고 급변하는 소프트웨어 산업에 능동적으로 대체와 소프트웨어 개발에 시간을 함으로써 현존하는 다양 DB군들을 최소한의 코드 수정을 통하여 구동할 수 있게 함으로써 경제성을 높이는 것이 본 논문의 목적이다. 따라서 이 모델을 중심으로 자동차 관리 서비스 도구(AutoMobile Customer Service Shop: AMOSS)를 구현한다.

1 서론

객체 지향 패러다임의 확산으로 인하여 소프트웨어 개발을 위한 객체 모델의 사용이 일반화되고 있다[1]. 기존의 ODBC를 기반으로 구현된 DB 연동 application의 경우, DB와 application에서 부수적인 문제점들이 발견되고, 서로 다른 platform을 기반으로 한 client들과의 연동을 위해서, 각 platform에 따른 application이 개발되어야 했다. 이러한 문제점을 극복하기 위해 이 기종간의 시스템을 통합할 수 있는 통합 Middlware의 선정이 필요하다[2].

본 논문에서는 객체지향 소프트웨어 공학 프로세스에 의해 생성되는 산출물을 객체 형태로 통합 관리하는 통합 객체 관리 모델을 제안한다. 분산 데이터 처리를 위한 표준화 작업을 이루어 현존하는 다양한 플랫폼 및 응용 시스템을 그대로 살리면서 통합환경에서 제어기능을 추가시켜 성능 문제를 향상시키기 위한 새로운 분산 객체 모델인 통합 객체 관리 모델(Integration Object Management Model)이다. 본 논문에서는 이 모델을 중심으로 AMOSS(AutoMobile Customer Service Shop)를 설계 구현한다.

본 논문의 목적은 통합객체 관리 모델을 중심으로 기존 시스템을 통합에 필요한 환경을 만들어 효율적으로 관리함으로서 소프트웨어 재사용성을 향상시켜 생산성을 극대화시키는 것이 목적이다. 또한 객체지향 소프트웨어 공학 프로세스에서 발생하는 다양한 산출물을 데이터 베이스화하여 필요한 산출물을 관리하고, 다양한 CASE 도구들과 서로 호환성 있게 유지, 보수 관리 할 수 있는 통합 객체 관리모델을 제안하고 이것을 기초로 생성기를 생성하여 개발환경에 적응할 수 있는 새로운 코드를 만들어 궁극적으로 객체지향 소프트웨어 개발의 생산성을 대폭 개선시키는 데 있다.

자바 언어로 분산 애플리케이션을 작성하는 방법도 JDBC, RMI, CORBA 등의 여러 가지 방법이 있다. 이 세가지 방법에는 각기 그 특성과 장점 단점이 있는데 자바 언어의 특성을 구현의 융통성도 높다고 할 수 있는 JDBC(Java Database Connectivity)를 이용하

여 소프트웨어를 개발한다. 장점은 platform에 구애받지 않는 실행 환경 제공과 코드의 수정 없이 다양한 DBMS와 연동 가능한 점을 볼 때, 재사용성(Reusability)가 뛰어나다[4]. 또한 JFC(Swing)을 이용하여 platform에 관계없이 동일한 UI(User Interface)의 제공이 가능하다. 따라서 본 논문에서 구현한 AMOSS는 JDBC Thin driver를 이용하여 Oracle DB와 연동하고, JDBC를 이용하여 DB와 원격지 사용자간의 보다 효과적인 이용이 가능하게 하였다.[3] 또한 기존 AWT가 platform의 peer 객체에 따라서 서로 다른 UI를 제공하는 문제점의 해결을 위해, JFC를 사용하였다.

2. 관련 연구

JDBC의 특징은 다음과 같다. JDBC란 SQL을 실행하기 위한 자바 API라고 할 수 있으며 다음과 같은 특징이 있다.

- 표준 SQL 데이터베이스 접근 인터페이스이다.
- RDB에 대해 일관된 인터페이스 제공
- 고수준의 도구나 인터페이스를 작성하기 위한 일반적인 표준이 될 수 있다.

AMOSS에서는 windows NT용 Oracle DB를 사용하고 JDBC를 이용하여 3-tier의 형태로 구성한다.

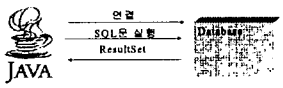
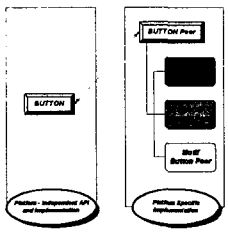


그림 1. JDBC의 연결

JFC(Swing의 특징)은 다음과 같다. AWT를 기반으로 Java상에서 보다 진보적인 UI를 제공하기 위한 library를 제공하며 JFC(Java Foundation Classes)라고 한다. Swing은 AWT와 비교하여 다중 look and feel을 동적으로 지원한다. Swing은 MS windows에서는 MS windows와 동일하게 보여진다. AWT와 Swing의 차이점은 (그림 2)와 같다.[3]



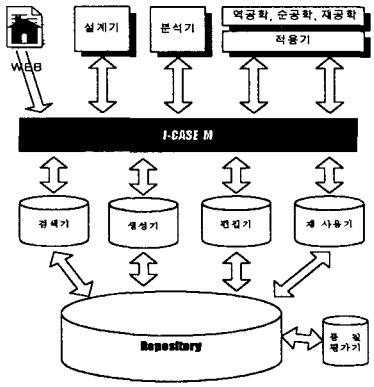
(그림 2). AWT와 JFC(Swing)

3. 통합 객체 관리 모델

[9] 단점인 이 기종간의 통합불능과 서비스 품질 향상을 위한 제어 기능 부재로 효과적인 관리를 할 수가 없었고 웹과 연동 할 수 있는 환경이 매우 미약했다. 따라서 본 논문에서는 이 기종간의 시스템을 통합할 수 있는 Middleware의 선정이 필요하게 되었고 보다 효율적으로 관리해 주고 급변하는 소프트웨어 산업에 능동적으로 대처하고 소프트웨어 개발에 시간을 단축함으로서 통합 환경에서 시스템 성능 저하 문제점을 해결하고 ObjectWeb을 이용한 분산객체 환경에서 프레임워크 객체 모델로 현존하는 다양한 플랫폼 및 응용 시스템을 그대로 살리고 제어 메카니즘을 추가하여 효과적인 관리를 할 수 있는 모델을 제안한다. 또한 객체저장소의 효율적인 관리와 검색을 위한 연결관계 관련성을 부여함으로써 객체 모델을 바탕으로 라이브러리에 효과적인 저장이 이루어진다.

통합 객체 관리 모델(Integrated object Management Model)은 세 계층으로 나누어지는데 분류계층과 통합 관리기 계층, 저장계층이다. 분류를 View 객체라 하는데 사용자에게 정보를 디스플레이 하는 객체이다. 통합 관리기를 제어객체라 하는데 사용자가 입력을 처리하는 객체이다. 저장을 모델 객체라 하는데 데이터를 갖는 모델 객체이다.

또한 이 객체들은 소프트웨어 라이프 싸이클의 효율적인 관리를 하며, 분산객체 환경에 적합한 구조를 만드는데 중점을 두고 소프트웨어 개발에서 생성된 산출물들을 관리 할 수 있는 것이 객체저장소까지 포함하고 있다.



(그림 3) 통합 객체 관리 모델 전체 구성도

따라서 [9]에서 제안한 모델을 더 확장시켜 통합 객체 관리 모델(Integrated object Management Model)을 제시한 것이다. 개발기간을 단축시켜 소프트웨어 산출물들을 효율적으로 유지보수 하고 저장 관리하여 생산성을 극대화시키는 것이 본 논문에서 제안한 목적이다. (그림 3)은 통합 객체 관리 모델 전체 구성도를 나타낸 것으로 세 가지 계층으로 구성되는데 프로세싱 관리계층, 통합 관리기 계층(I-CASE Manager), 마지막으로 객체 저장소 관리 계층이다.

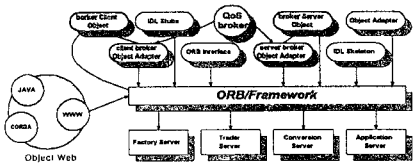
3.1 프로세싱 관리 계층

소프트웨어 라이프싸이클 관리를 하기 위한 생명주기 서비스를 제공하고, 객체지향 소프트웨어 개발과정 중에는 문서, 다이아그램, 원시코드, 방법론 정보, 설계 패턴등의 프레임

워크, 사용자 인터페이스 객체, 데이터 베이스 스키마 객체 등의 다양한 산출물이 있는데 이런 산출물들을 분류하는 방법을 제시하는 것이 프로세싱 관리 계층이라 하고 분류객체라 한다. 분류를 View 객체라 하는데 사용자에게 정보를 디스플레이하는 객체이다. 즉 여러 산출물들을 어디에 저장시킬 것인가를 결정시키고, 재사용할 수 있도록 분류하고 저장하는 단계를 결정하는 것이 프로세싱 관리 계층이다.

3.2 통합 관리기 계층

[9]에서는 제어 메카니즘이 없고, 분산 객체 시스템에서 클라이언트와 서버에 각각 프로세스와 데이터의분형과 불균형으로 통합에 필요한 적절한 아키택쳐가 절실하게 필요하게 되었다. 통합 관리기는 이러한 단점을 보완하고 프레임워크 기반의 통합은 실제 통합에 필요한 설계 구현 비용을 최소화 한다. 통합 관리기(I-CASE Manager)의 구조를 나타낸 것이 (그림 4)과 같다.



(그림 4) 통합 객체 관리기 구성도

구성은 프레임워크/ORB, 객체들의 위치 정보를 제공하는 트레이딩 서버(trading server), 데이터들의 변환을 담당하는 변환 서버(conversion server), 각종 데이터 객체들의 생성·복사·삭제를 담당하는 팩토리 서버(factory server)[18], 소프트웨어 QoS서버, ObjectWeb로 구성된다.

통합 객체 관리기(I-CASE M)는 ObjectWeb이란 소프트웨어 아키텍쳐를 사용하고 WWW와 CORBA로 연결하여 통합 관리하는데 필요한 구조를 만들어 통합 관리하는데 용이하게 했다. CORBA는 효과적인 시스템통합을 위해 기술적인 이익을 제공하며 이 기종의 시스템들의 분산 의사소통 환경을 위한 하부구조를 제공한다.

Broker객체는 서비스 품질의 향상을 위한 제어 메카니즘으로 클라이언트 응용 프로그램과 클라이언트가 요구하는 서비스를 제공하는 서버들을 중재하는 제어기법이다.

Broker 객체는 서비스 품질의 향상 제어 기능을 수행하기 위해서는 기존의 ORB에 BOA (Broker Object Adapter)를 추가하게 된다. BOA는 클라이언트 BOA와 서버 BOA로 구성된다. 클라이언트 BOA와 서버 BOA가 존재하는 이유는 BOA가 제공하는 QoS 제어 메커니즘은 분할 레벨에 따라 다음과 같이 네가지로 나눌 수 있는데 첫째는 Process Data Server QoS 제어 메커니즘와 둘째는 Process Balanced QoS 제어 메커니즘, 세째는 Data Balancd QoS 제어 메커니즘와 끝으로는 Process/data Balanced QoS 제어 메커니즘로 이루어 지는데 본 논문에서는 Process/data Balanced QoS 제어 메커니즘으로 적용을 했다.

3.3 객체 저장소 관리 계층

JDBC를 이용하여 현존하는 시스템을 재사용하고 시스템 성능을 향상시키는 것이다. 앞 계층에서 분류가 결정되면 효율적인 검색과 재사용 할 수 있도록 객체저장소에 저장해야한다. 이런 객체 저장소의 주요기능으로는 많은 산출물을 관리해야 하는데 저장소 관리 계층은 이런 관리를 하는 계층이다. 또한 다양한 도구와 사용자의 동시 공유를 위한 병행 처리, 주요 산출물 객체에 대한 등급별 접근 처리와 보완처리, 시스템 장애 발생에 대한 회복, 객체의 올바른 상태를 유지하기 위한 무결성 처리 기능이 제공되어야 한다. 이러한 기능은 객체 저장소를 파일 시스템을 기반으로 하지 않고, DBMS를 기반으로 개발함으로써 제공받을 수 있다. 또한 도메인에서 발견되는 영구적인 정보나 데이터를 저장 및 관리하는 것인데, 객체지향의 저장소 관리 계층은 객체들로 구성되어 있다. 또한 정보 저장소는 객체저장소(Object Repository)라 하고 부품을 클래스별로 저장한다. 이런 객체 저장소를 관리할 수 있는 기능이 객체 저장소 관리 기능이다.

4 AMOSS의 설계

AMOSS의 설계는 통합 객체 관리 모델 중 객체 저장소 관리 계층을 적용해서 설계했다. 어느 특정 주제를 선정하여, JDBC와 JFC를 이용한 실제 프로그래밍을 통해서 작성한다. 객체 저장소의 DB schema는 customer_management 으로 고객의 삽입, 삭제, 갱신을 위한 관리를 위해서 작성된 테이블로서 차주이름, 주민등록번호, 주소, 전화번호, 면허번호등에 관한 정보를 가지고 있다. member는 실제 자동차 정비소에서의 인력 관리를 위해서 작성된 테이블이다. car_problem는 자동차의 각 부위별 파손정도를 integer값으로 나타내며, 0의 경우는 이상 무, 1의 경우는 수리, 3의 경우는 교환으로 설정하였다. 이 설계환경은 하나의 DBMS를 원격지에서 보다 효과적으로 사용하고, 다양한 client의 platform상에서 동일하게 실행되는 사용자 환경을 제공하고, 구조적 사항은 RDBMS를 사용하여 고객정보를 관리하고 JDBC Thin Driver를 사용하여 객체 저장소인 DB와 연동한다. 또한 연결된 DB를 JDBC를 이용하여 DBMS를 원격지에서 효과적으로 사용한다. GUI로 요구되는 사항으로는 사원관리를 위한 사원정보와 고객관리를 위한 고객정보, 정비소 관리를 위한 정비현황정보와 부속현황 정보등로 나누어 볼 수 있다. GUI에서 DB로 제공이 요구되는 data로는 새로운 고객과 자동차 정보와 고객관리의 갱신, 삭제, 삽입을 위한 data가 요구된다. 객체 저장소의 클래스에는 자동차 다이아그램, 양식, 컴포넌트에 공통적인 정보들, 즉 설명 속성들이 포함되어 있다. 자동차의 번호(id), 종류(name), 설명(description), 작성자(creator), 작성시간 날짜(create_time) 등의 검색을 위한 속성들이 포함된다.

JDBC thin driver와 JFC를 이용한 UI 설계부분은

다음과 같다.

```
static final String driver_class = "oracle.jdbc.driver.Oracle
Driver";
static final String connect_string = "jdbc:oracle:thin:system
/manager@aegis.kyunghee.ac.kr:1526:ORCL";
public Office(String setTitle)
{
        super(setTitle);
        setLayout(new BorderLayout());
        try
        {
      Class.forName(driver_class);
        }
        catch(java.lang.ClassNotFoundException e)
        {
      System.err.println("class not found");
      return;
        }
        try
        {
myConnection=DriverManager.getConnection(connect_string);
    System.out.println("connected !!");
        }
        catch(SQLException e)
        {
System.err.println("sql exception occurs" + e.getMessage());
    return;
        }
```

5. AMOSS 구현

하나의 DBMS를 원격지에서 보다 효과적으로 사용하고, 다양한 client의 platform상에서 동일하게 실행되는 사용자 환경을 제공하고, 크게 다음과 같이 세 가지 관점에서의 접근이 가능하다[5][6][7].. 첫째 구조적 사항은 RDBMS를 사용하여 고객정보를 관리하고 JDBC Thin Driver를 사용하여 DB와 연동한다. 또한 연결된 DB를 JDBC를 이용하여 DBMS를 원격지에서 효과적으로 사용한다. 둘째는 GUI로 요구되는 사항으로 사원관리를 위한 사원정보와 고객관리를 위한 고객정보, 정비소 관리를 위한 정비현황정보와 부속현황정보등으로 나누어 볼 수 있다. 셋째는 GUI에서 DB로 제공이 요구되는 data이다. 또한 새로운 고객과 자동차 정보와 고객관리의 갱신, 삭제, 삽입을 위한 data가 요구된다.

```
import com.sun.java.swing.*;   import java.awt.*;
import java.awt.event.*;
public class CustomerPanel_1 extends JPanel implements ActionListener
{
        public Connection con;      public String data[];
        public JTextField text[];   public JLabel field[];
        public boolean check_bit[]; public Statement myStatement;
        public ResultSet results;   public final int elem = 7;
        public Hashtable hash;      public CarPanel car;
        public CustomerPanel_1(Connection con)
        {
        con = con;                  hash = new Hashtable();
        hash.put("0", "social_security_num");     hash.put("1", "owner_name");
        hash.put("2", "address");   hash.put("3", "telephone");
        hash.put("4", "driver_license");          hash.put("5", "car_num");
        hash.put("6", "car_model"); setLayout(new GridLayout(0,2));
        data = new String [elem];   text = new JTextField[elem];
        field  = new JLabel[elem];  check_bit = new boolean[elem];
        JLabel titleLabel = new JLabel("고객관리");
        Font titleFont = new Font("Serif", Font.BOLD, 30);
            titleLabel.setFont(titleFont);
                add(titleLabel);
```

위 리스트는 AMOSS 구현 화면이다. 개발 도구로는 JDK ver1.2 으로 JAVA언어의 기본 개발툴, JFC로 Swing을 사용하기 위한 package 정의하였으며 JDBC thin driver를 사용하였다. 이 사원관리는 설명 속성들이 포함되어 있다. 산출물의 번호(id), 종류(name), 설명(description), 작성자(creator), 작성시간 날짜(create_time) 등의 검색을 위한 속성들이 포함된다. 개발 애플리케이션은 고객관리, 사원관리, 부품관리가 있는데 실행화면은 다음과 같다.

(그림 5) 실행 화면

6. 결론 및 향후 연구과제

본 논문에서는 통합 관리 모델을 제시한다.

JDBC와 JFC(Swing)을 이용하여 RDBMS와의 연동을 기반으로 한 application 개발에 초점을 두었으며 이를 통해서 보다 빠르고 안정적으로 원격지 상에서의 사용자 사용환경에 중점을 두었다. 향후 연구과제로서 프로세서 계층에서 분류하는 알고리즘을 제시해야하고, 산출물 객체를 효율적으로 관리하는 객체 저장소 관한 연구를 할 것이고 이를 기반으로 하여 Java servlet을 이용한 WEB상에서의 가상 견적시스템등의 개발이 가능하며, 이 기종 DBMS을 사용할 경우, CORBA와 Java를 연동하여, 이기종 DBMS를 기반으로 다중 platform에서 작동하는 application의 구현에 중점을 둘 것이다.

참 고 문 헌

[1] Jon Meyer & Troy Downing "JAVA Virtual Machine", O'REILLY, 1997

[2] Mary Campione, Kathy Walrath, "The Java Tutorial", Addison-Wesily, 1996

[3] Elliotte Rusty Harold, "Java Networking Programming", O'Reilly & Associates, Inc., 1996

[4] Prashant Sridharan, "Advanced Java Networking", Prentice-Hall, Inc., 1997

[5] Orfali, Harkey and Edwards, "The Essential Distributed Objects Survival Guide", Wiley Press, 1996

[6] Jim Waldo, Geoff Wyant, Ann Wolrath and Sam Kenedall, "A Note on Distributed Computing", Sun Microsystems, 1994

[7] "JDBC specification", http://splash.javasoft.com/jdbc

[8] Emerson, Darnovsky, and Bowman, "The Practical SQL Handbook", Addison-Wesley, 1989

[9] 선수균, 송영재, "통합객체지향 관리기 중점을 둔 F77/J++ 생성기 설계", 99년 추계 학술발표집 한국정보과학회, 1999.