컴포넌트 재구성을 위한 JFC기반 통합 객체 관리 모델 설계

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

Implementation of AMOSS by Using JDBC - based on the Integration Object manager 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 있고 JFC(Java Foundation Classes : Swing)는 분산 네트웍 환경에서 매우 적합한 환경으로 주목 받고 있다. DB와 application에서 부수적인 문제점들이 발견되고, 서로다른 platform을 기반으로 한 client들과의 연동을 위해서, 각 platform에 따른 application이 개발되어야 했다. 이런 환경에서 개발자가 프로그램 변경이 발생할 때 연결관계에 따라 템플리트에 의해 개발된 부분을 코드로 변경해 주는 도구가 필요하게 되었다. 이를 위해 JFC기반의 통합 Middlware의 선정이 필요하게 되었다. 따라서, 본 논문에서는 JFC기반의 통합 관리하는 통합 객체 관리 모델을 설계한다. 이것은 특히 컴포넌트 재구성을 위한 것이고, 목적은 기존의 시스템을 재사용하고 현존하는 컴포넌트를 재구성하여 최소한의 코드 수정을 통하여 시스템을 구동할 수 있게 함으로써 소프트웨어의 경제성을 높이는 것이 본 논문의 목적이다.

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 이기종간의 시스템을 통합할 수 있는 통합 Middlware의 선정이 필요하다[9,10,2,3,4,5]. 또한 컴포넌트 재구성을 위해서 서로 관련성 있는 정보를 서로 연결관계로 링크하여 관리한다면 정보를 효율적으로 관리할 수 있는 거이다. 따라서 논문에서는 컴포넌트 재구성을 위한 JFC기반으로한 통합 모델을 설계하여 객체지향 프로세스에 의해 생성되는 산출물들을 효과적으로 관리한이다. 특히, 분산 데이터 처리를 위한 표준화 작업을 이루어 현존하는 다양한 플랫폼 및 응용 시스템을 그대로 살리면서 통합환경에서 컴포넌트를 재구성하여 초기 프로그램 코드를 생성하여 JFC기반의 통합 객체 관리 모델(Integration Object Management Model)를 설계한다.

본 논문의 목적은 통합객체 관리 모델을 중심으로 기존 시스템을 통합에 필요한 환경을 만들어 효율적으로 관리함으로서 소프트웨어 재사용성을 향상시켜 생선성을 극대화시키는 것이 목적이다. 또한 객체지향 소프트웨어 공학 프로세스에서 발생하는 다양한 산출물을 데이터 베이스화하여 필요한 산출물을 관리하고, 다양한 CASE 도구들과 서로 호환성 있게 유지, 보수관리할 수 있게 하는 것이다.

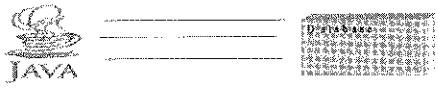
자바 언어의 특성을 구현의 융통성도 높다고 할 수 있는 JDBC(Java Database Connectivity)를 이용한다. 장점은 platform에 구애받지 않는 실행 환경 제공과 코드의 수정 없이 다양한 DBMS와 연동이 가능한 점을 볼 때, 재사용성(Reusability)가 뛰어나다[4]. 또한 JFC(Java Foundation Classes : Swing)을 이용하여 platform에 관계없이 동일한 UI(User Interface)의 제공이 가능하다. 따라서 본 논문에서 설계한 시스템은 JDBC Thin driver를 이용하여 Oracle DB와 연동 할 수 있게

하고, JDBC를 이용하여 DB와 원격지 사용자간의 보다 효과적인 이용이 가능하게 하였다. 또한 기존 AWT가 platform의 peer 객체에 따라서 서로 다른 사용자 환경을 제공하고, 문제점의 해결을 위해, JFC를 사용하였다.

2. 관련 연구

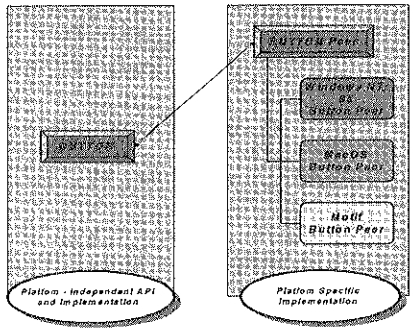
JDBC의 특징은 다음과 같다. JDBC란 SQL을 실행하기 위한 자바 API라고 할 수 있으며 다음과 같은 특징이 있다. 표준 SQL 데이터베이스 접근 인터페이스이다.RDB에 대해 일관된 인터페이스 제공을 한다. 고수준의 도구나 인터페이스를 작성하기 위한 일반적인 표준이 될 수 있다.

설계 시스템은 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 다이나믹하게 지원한다. Swing은 MS windows에서는 MS windows와 동일하게 보여진다. AWT와 Swing의 차이점은 (그림 2)와 같다.[3]

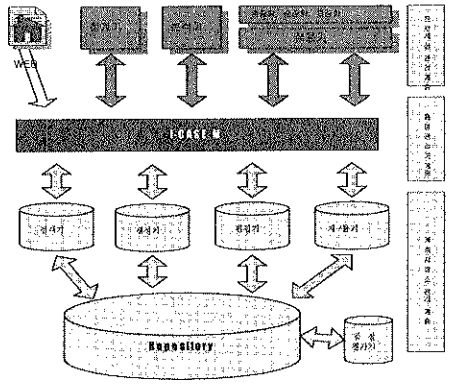


(그림 2). AWT와 JFC(Swing)

3. JFC 기반의 통합 객체 관리 모델 설계

객체저장소의 효율적인 관리와 검색을 위한 연결관계 관련성을 부여함으로써 객체 모델을 바탕으로 라이브러리에 효과적인 저장이 이루어진다. 통합 객체 관리 모델 제안은 소프트웨어 라이프싸이클의 효율적인 관리를 하며, 분산객체 환경에 적합한 구조를 만드는데 중점을 두고 소프트웨어 개발에서 생성된 산출물들을 관리 할 수 있는 것이 객체저장소까지 포함하고 있다.

따라서 [17]에서 제안한 모델을 더 확장시켜 통합 객체 관리 모델((Integrated object Management Model)을 설계한다. 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 관리 계층, 객체 저장소 관리 계층, 통합 관리기 계층 (I-CASE Manager)이다.



(그림 3) 통합 객체 관리 모델 전체 구성도

3.1 프로세싱 관리 설계

소프트웨어 라이프싸이클 관리를 하기 위한 생명주기 써비스를 제공하고 객체지향 소프트웨어 개발과정 중에는 문서, 다이아그램, 원시코드, 방법론 정보, 설계 패턴등의 프레임워크, 사용자 인터페이스 객체, 데이터베이스 스키마 객체 등의 다양한 산출물이 있는데 이런 산출물들을 분류하는 방법을 제시하는 것이 프로세싱 관리 계층이다.

3.2 통합 관리기 설계

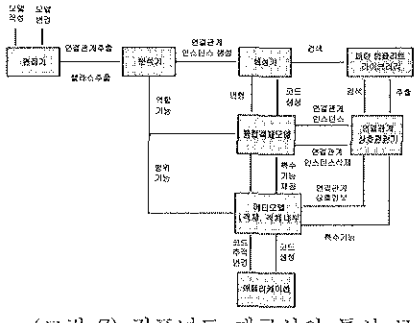
분산 객체 시스템에서 클라이언트와 서버에 각각 프로세스와 데이터의분형과 불균형으로 통합에 필요한 적절한 아키텍처가 절실하게 필요하게 되었다. 통합 관리기는 단점을 보완하고 프레임워크 기반의 통합은 실제 통합에 필요한 설계 구현 비용을 최소화 할 수 있다.

3.3 객체 저장소 관리 설계

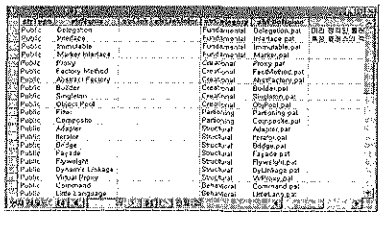
JDBC를 이용하여 현존하는 시스템을 재사용하고 시스템 성능을 향상시키는 것이다. 앞 계층에서 분류가 결정되면 효율적인 검색과 재사용 할 수 있도록 객체저장소에 저장해야한다. 이런 객체 저장소의 주요기능으로는 많은 산출물을 관리해야 하는데 저장소 관리 계층은 이런 관리를 하는 계층이다. 또한 컴포넌트를 재구성하는 설계단계도 포함하고 있다. 프로그래머가 직접 수많은 클래스 연결관계의 템플리트를 이해하고 모델에 필요한 코드를 연결관계를 직접 구현해야 한다면 이는 항상 패턴을 이해하고 이를 자신의 문제에 항상 활용하는 만큼의 노력이 요구될 수도 있다는 단점이 있다. 또한 연결관계는 지속적으로 변화가 일어날 수 있는 부분이고 항상 변화를 요구하고 있다는 문제를 안고있다. 따라서, 프로그래머가 템플리트를 활용하는 방법을 습득하기 위한 노력을 줄여 주고, 어플리케이션에 관련된 부분만을 개발할 수 있도록 지원하고, 변경이 발생하는 연결관계에 따라 템플리트에 의해 개발된 코드의 변경을 책임지는 연결 관계 관

련성 도구가 필요하게 되었다. 이런 단점을 해결하기 위한 것이 컴포넌트 재구성이다.

컴포넌트 재구성은 각 연결관계를 기반으로 객체지향 프로그래밍에 필요한 코드를 생성하는 템플리트를 참조한다. 모델 생성기에 의해 생성된 코드는 코드 편집기에서 사용자가 작성한 코드와 분리하여 관리한다. 즉, 모델에 작성된 클래스 본질에 관련된 코드와 연결관계 관리에 필요한 코드는 별도로 관리한다. 이는 연결관계의 변경이 이루어졌을 때 이를 지금까지 개발한 코드로 직접 반영이 되도록 지원하기 위해서 이다. 이 모델은 첫째는 편집기이다. 편집기에 의해 작성된 모델은 분석기에 의해 분석되고, 저장되며 이렇게 저장된 형태는 다시 생성기에 의해 수정이 필요할 때 편집기에 의해 모델로 재구성된다. 편집기는 개발자가 객체 모델을 구축할 수 있도록 지원하는 서브 시스템으로 모델을 작성 및 수정할 수 있도록 한다. 둘째는 모델 구조를 입력받아서 필요한 템플리트를 선택하여 템플리트의 인스턴스화에 필요한 정보를 구성하고 해당하는 코드를 분석하여 생성하는 서브 시스템이다. 컴포넌트 재구성은 프로그래머가 템플리트를 활용하는 방법을 습득하기 위한 노력을 줄여 주고 어플리케이션에 관련된 부분만을 개발할 수 있도록 지원한다.



(그림 7) 컴포넌트 재구성의 통신 모델



(그림 5) 설계한 데이터베이스의 투플 구조

JDBC thin driver와 JFC를 이용한 UI 설계부분은 다음과 같다.

```
import com.sun.java.swing.*;    import java.awt.*;
import java.awt.event.*;
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;         }
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)
                        Font titleFont = new Font("Serif", Font.BOLD, 30);
                titleLabel.setFont(titleFont);
                        add(titleLabel); }
```

4. 결론 및 향후 연구과제

본 논문에서는 프로그래머가 템플리트를 활용하는 방법을 습득하기 위한 노력을 줄여 주고, 어플리케이션에 관련된 부분만을 개발할 수 있도록 지원하고, 변경이 발생하는 연결관계에 따라 템플리트에 의해 개발된 코드의 변경을 책임지는 연결 관계 관련성 도구가 컴포넌트 재구성이다. 이것을 위한 JFC기반의 통합 객체 관리 모델을 설계한다.

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 컴포넌트 재구성하기 위해서 컴포넌트를 분류하는 알고리즘을 제시해야하고, 객체를 효율적으로 관리하는 객체 저장소에 관한 연구를 할 것이고 인터넷상에서 표준화가 이루어지도록 하는 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] 선수관, 송영재, "통합객체지향 관리기 중점을 둔 F77/J++ 생성기 설계", 99년 추계 학술발표집 한국정보과학회, 1999.

[10] 선수관, 송영재, "통합 객체 관리 모델을 위한 F77/J++ 생성기에 관한 연구", 정보처리 학회 논문지 제 7권 10호, 한국정보처리학회, pp. 3064-3074 2000.11