제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

패턴정보저장소를 이용한 웹기반의 인덱스 순서관계정보모델 설계

선수균
*동원 대학 e-비즈니스과

Design of Web-based Index Sequence Relation Information Model Using Pattern-In Repository

Su-Kyun Sun
Dept. of e-besiness, DongWon Collage, Korea

요 약

최근에는 웹 환경에 적합한 개방형과 컴포넌트들을 효율적으로 분류하고 추출하는 방법이 진행되고 있다. 본 논문에서는 인터넷 환경하에서 생성되는 산출물을 컴포넌트 객체 형태로 통합 관리하고, 데이터베이스와 연결하여 객체들을 효율적으로 관리해 주는 웹기반 인덱스순서관계정보모델을 설계한다. 이 제안의 장점은 "인덱스 순서관계정보"로 클래스(class)들 사이의 관련된 여러 관계정보를 UML 설계방법에 적응할 수 있는 구조로 변형함으로서 개발자가 관계정보를 쉽게 파악하여 모델링 설계언어인 UML 설계방법에 쉽게 적응할 수 있게 한다. 이 모델로 기존의 시스템을 재사용하고 급변하는 소프트웨어 산업에 능동적으로 대체와 소프트웨어 개발에 시간을 단축함으로써 현존하는 다양한 데이터베이스군들을 최소한의 코드 수정을 통하여 구동할 수 있게 함으로써 소프트웨어 개발 경제성을 높이는 데 있다.

1 서론

최근에는 웹 환경에 적합한 개방형과 컴포넌트들을 효율적으로 분류하고 추출하는 방법이 진행되고 있다[5.6.7]. 본 논문에서는 인터넷 환경하에서 생성되는 산출물을 컴포넌트 객체 형태로 통합 관리하고, JDBC를 이용하여 데이터베이스와 연결하고 패턴정보와 객체들을 효율적으로 관리하기 위한 패턴 정보저장소를 중심으로 웹기반 인덱스순서관계정보모델을 설계한다. 이 모델로 기존시스템을 재사용하고 급변하는 소프트웨어 산업에 능동적으로 대체와 소프트웨어 개발에 시간을 단축함으로써 현존하는 다양한 데이터베이스군들을 최소한의 코드 수정을 통하여 구동할 수 있게 함으로써 소프트웨어 개발 경제성을 높이는데 있다. 본 논문에서는 기존 분류방법과 추출방법의 단점을 보완하고, 패턴을 인덱싱과정으로 패턴정보의 효율적인 추출 및 분류가 될 수 있도록 "인덱스 순서관계 정보 모델"를 제안한다. 또한 인덱스 순서관계정보모델을 제안함으로서 패턴을 효율적으로 추출, 검색하여 패싯항목을 코드화하고 데이터베이스화한 객체관리 모델을 제시함으로서 패턴정보 인덱싱과정으로 시스템 성능 향상과 분류의 효율성을 극대화시키는데 목적을 둔다.

"인덱스 순서관계정보모델"을 설계로 효율적인 패턴정보를 추출 및 분류를 위해 패턴 인덱싱과정을 설계하였다. 이것은 패턴정보를 패턴 정보 저장소에 저장함으로서 구조적인 추출과 패턴이 어느 상황에 적용될 수 있는가에 대한 개발자간의 경험적 상황을 고려한 기능적 인덱싱 자동부여로 설계한다. 이 과정은 정확하게 분류하는 방법으로 정확도를 향상시켜 소프트웨어 생산성을 향상 시키는게 목적이다.

따라서 본 논문에서는 웹환경에서 통합 관리하고, 데이터를 수집하고 검색하여 최적의 데이터로 저장할 수 있는 패턴 정보와 JDBC를 이용한 웹기반의 인덱스순서관계정보모델을 설계하고 컴포넌트를 추출하고 효율적으로 분류를 함이다. 또한 객체지향 소프트웨어 공학 프로세스에서 발생하는 다양한 산출물을 데이터베이스화하여 필요한 산출물을 관리하고, 정보 저장소를 오라클 데이터 베이스로 관리함으로써 클라이언트와 서버를 연결하는 기능을 JDBC를 이용하여 웹 환경에 쉽게 적응할 수 있도록 설계했다. 이것을 기초로 생성기를 생성하여 개발환경에 적응할 수 있는 새로운 코드를 만들어 궁극적으로 객체지향 소프트웨어 개발의 생산성을 대폭 개선시키는 데 있다.

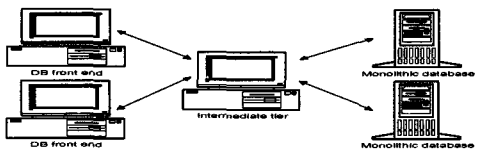
본 논문의 목적은 다음과 같다. 첫째 "인덱스 순서관계정보"로 클래스(class)들 사이의 관련된 여러 관계정보를 UML 설계방법에 적응할 수 있는 구조로

변형함으로서 개발자가 관계정보를 쉽게 파악하여 모델링 설계언어인 UML 설계방법에 쉽게 적응할 수 있게 한다. 둘째 패턴정보를 구성하고 있는 UML 관계 관련성에 대한 구조적 추출과 패턴정보 인덱싱과정으로 패턴이 어느 상황에 적용될 수 있는가에 대한 기능적 인덱싱과 패턴을 규격화시키는 순서기준 인덱싱로 패턴을 패싯분류 항목으로 코드화시킨다. 셋째 객체관리모델을 이용하여 패턴을 데이터베이스화하고 패턴 정보저장소를 구축함으로서 메타정보를 등록하여 패턴 구조를 모델링할 수 있으며 디자인패턴을 용이하게 추출, 검색, 분류함으로써, 궁극적으로 객체 지향 소프트웨어 개발의 생산성을 대폭 개선시키는 데 있다.

2. 관련 연구

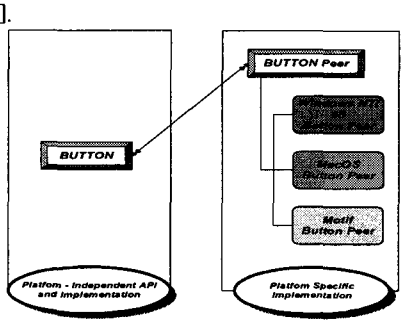
2.1 DBMS 모델

DBMS 모델은 일반적으로 데이터베이스(DB)와 사용자(Client)간의 관계로 정의할 수 있으며, Two-Tier 모델의 경우, DB front end와 DB engine으로 분리되고 다른 방식으로는 지역데이터와 원격 데이터 분리할 수 있다. Two-Tier 모델의 경우 UI를 분리할 수 있으므로 개발이 용이하나, mirrong, caching, proxy services, secure transactions 등의 문제가 있다[1,2,3,4].



(그림1) 3 - Tier Database

Three-Tier 모델은 Two-Tier 모델에서 발생되는 mirroring, caching, proxy services, secure transactions등이 해결되었고, 실제적 구조는 그림 3과 같다[1.7].

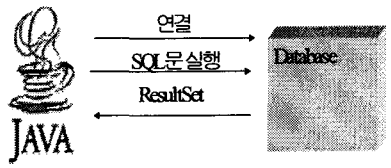


(그림 2) AWT와 JFC(Swing)

JFC(Swing의 특징)은 다음과 같다. AWT를 기반으로 Java상에서 보다 진보적인 UI를 제공하기 위한 library를 제공하며JFC(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.[1,2,4]

2.2 JDBC

JDBC의 특징은 다음과 같다. JDBC란 SQL을 실행하기 위한 자바 API라고 할 수 있으며 다음과 같은 특징이 있다. 표준 SQL 데이터베이스 접근 인터페이스이다[4].



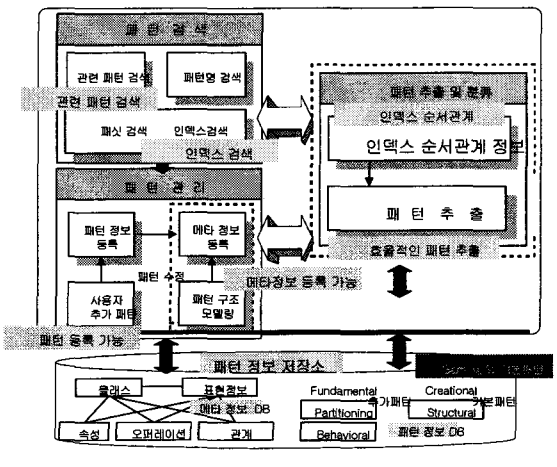
(그림 3) JDBC의 연결

RDB에 대해 일관된 인터페이스 제공을 한다. 고수준의 도구나 인터페이스를 작성하기 위한 일반적인 표준이 될 수 있다. 본 논문에서는 windows NT용 Oracle 데이터베이스(DB)를 사용하고 JDBC를 이용하여 3-tier의 형태로 구성하여 설계한다.

3. 인덱스 순서 관계 정보 모델 설계

분산객체 환경에서 프레임워크 객체 모델로 현존하는 다양한 플랫폼 및 응용 시스템을 그대로 살리고 웹기반 인덱스순서관계정보 모델을 설계한다. 이 모델은 패턴검색, 패턴 추출 및 분류, 패턴관리, 패턴정보 저장소로 나뉜다.

이 모델의 제안은 개발기간을 단축시켜 소프트웨어 산출물들을 효율적으로 유지보수 하고 저장 관리하여 생산성을 극대화시키는 것이 본 논문에서 제안한 목적이다. (그림 4)은 인덱스순서관계정보 모델 전체 구성도를 나타낸 것이다.



(그림 4) 인덱스순서관계정보 모델 구성도

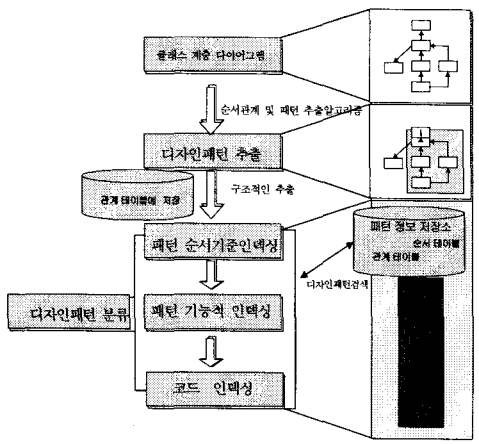
3.1 패턴정보 추출 및 분류

패턴 추출 및 분류구조의 전체 구성도는 (그림5)과 같으며 패턴추출 및 분류, 패턴검색, 패턴관리, 패턴정보저장소등으로 구성된다. 디자인패턴 추출 및 분류는 패턴 정보 저장소와 밀접한 관계가 있으며 구조적인

추출과 효율적인 분류를 위하여 디자인패턴 인덱싱과정을 함으로서 코드화된 디자인패턴으로 효율적인 추출 및 분류를 할 수 있다. 소프트웨어의 분류방법은 소프트웨어의 재사용성(reusability)을 실용화시키는 핵심 요소들 중에 가장 중요한 역할을 한다. 패턴을 한마디로 정의한다면 "소프트웨어 엔지니어의 경험"이라고 말할 수 있다. 따라서 본 장에서는 "소프트웨어 엔지니어의 경험"과 "패턴의 코드화"로 패턴을 효율적으로 추출 및 분류하기 위한 인덱스 순서관계정보를 제안한다. 이것은 패턴 인덱싱 과정과 패턴을 추출하는데 구조적인 추출로 나눠 실시함으로서 개발자의 경험적인 면을 고려하여 패싯 분류방법으로 적용했고, 패턴 코드화 과정인 인덱싱함으로서 패턴을 효율적으로 분류하는 방법을 제시한다.

3.2 인덱스순서관계정보 제안

패턴을 보다 쉽게 이해하고 적용하기 위해서는 UML(Unified Modeling Language)의 관계(Realtion) 정보 사용이 필수적이다. 그러나 패턴의 체계적인 분류, 공유가 이루어지지 않아 적당한 패턴을 찾지 못하거나, 참조되지 못하는 경우가 많다. 기존 추출 방법[9]은 패턴을 추출하는데 구조적인 추출만 하기 때문에 UML로 모델링하여 패턴 메타 정보를 나타낼 수 없어 효율적인 패턴 정보를 추출할 수 없었다. 이런 단점을 해결하기 위하여 패턴 메타 정보를 구축할 수 있는 패턴 정보저장소를 구축함으로서 패턴을 메타모델링하여 설계 재사용을 극대화할 수 있는 시스템을 구축하였다.



(그림 5) 패턴 추출 및 분류과정

따라서 본 논문에서는 패턴 효율적으로 추출 및 분류하기 위해서 "인덱스 순서관계정보"를 제안하여 클래스들 사이의 관련된 여러 관계(Relation)정보를 UML 설계 방법에 쉽게 적응할 수 있는 구조로 변형함으로서 구조적인 추출과 패턴 인덱싱과정이 가능하게 하였으며 패턴의 역할정보와 패턴이 어느 상황에 적용될 수 있는가에 대한 개발자간의 경험적 상황을 고려한 기능적 정보와 순서기준 인덱싱, 기능적 인덱싱, 코드 인덱싱, 자동생성 분류방법을 설계함으로서 검색하는데 걸리는 응답시간을 단축하기 위함이다. 패턴을 코드화함으로서 효율적인 분류, 관리 및 검색이 용이해졌으며 재사용 극대화할 수 있었다. 아래 리스트는 추출 알고리즘을 나타낸 것으로 다음과 같다.

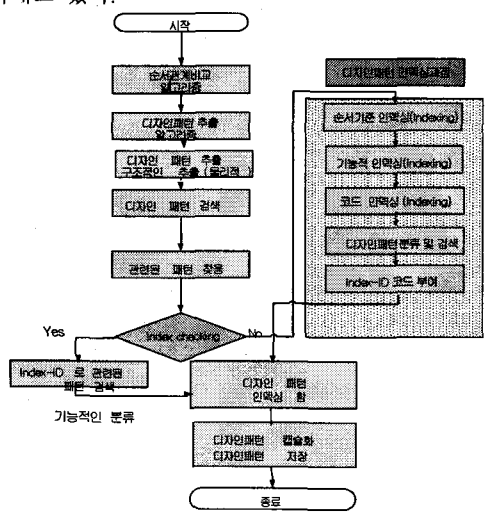
```
1:  Repeat
2:     set ( α, β)  /* α에서 β 방향의 관계  */
3:     set  α : / * 클래스 사이의 첫 번째 클래스  */
4:     set  β : /* 클래스 사이의 두 번째 클래스  */
5:     set  n : /* n개 클래스  */
6:     set  α←1, β←2
7:           // n 개 클래스 사이의 모든 관계 비교
8:  Until   total_count
9:   If design pattern of index Not exist
10:    while ( α = 1 to n-1 )
11:     {
12:       while ( β =  α+1 to n) {
13:                 // 클래스간의 관계 비교
14:          If ( α,  β) or ( β,  α) relation exist then
15:                    // 관계 존재
16:           store  table ( α,  β) ∈R or ( β,  α) ∈R
17:                 //   순서쌍 테이블 저장
18:          Endif
19:           }
20:     }
21:    Read  index from repository
22:   Else  /* index exist */
23:    Repeat
24:      retrieval  index-id
25:    Until
26:   Endif
```

인덱스 순서관계 정보를 제안하게된 목적은 다음과 같다. 첫째 인덱스 순서관계 정보로 클래스(class)들 사이의 관련된 여러 관계(Relation)정보를 UML 설계방법에 적응할 수 있는 구조로 변형함으로서 관계정보를 쉽게 파악하여 모델링 설계언어인 UML 설계방법에 쉽게 적응할 수 있게 한다. 둘째 인덱스 순서관계 정보는 원시 코드 중심의 재사용에서 탈피하고, 시스템 이해도를 높이기 위해 UML방법의 클래스 다이어그램의 구성 요소 중 클래스 상호 간의 관련성을 파악한 후 구조적인 추출을 한다. 셋째 디자인패턴을 구성하고 있는 UML 관계(Relation) 관련성에 대한 구조적 정보뿐만 아니라 패턴이 어느 상황에 적용될 수 있는가에 대한 경험적 정보를 추가하여 기존 단점을 보완했다. 넷째 패턴 인덱싱과정을 실시하여 복잡한 시스템의 복잡도를 감소시킬 수 있으며 패턴을 코드화할 때 패턴이 어떠한 역할을 수행하는가에 따른 기능적 분류와 패턴이 사용될 수 있는 여러 경험적 상황을 패턴 정보로 관리하여 가장 적합한 코드를 부여함으로서 증가하고 있는 수많은 패턴들을 효율적으로 분류하기 위한 것이다. 다섯째 패턴을 인덱싱함으로서 패턴의 효율적인 유지보수와 패턴 재사용을 극대화시키고, 검색분류 추출하는데 향상된 환경을 제공하기 위한 것이다. 논문의 접근은 클래스다이어그램으로부터 순환되는 패턴의 추출에서 시작되며, 각 패턴이 사용될 수 있는 여러 경험적 상황을 패턴 정보로 관리하고 이 정보를 이용하여 패싯 항목으로 설정하고 관련된 패턴 검색, 분류, 인덱싱으로 이어진다. 패턴은 클래스들의 관계에 대한 조직이고 시스템을 여러 가지 패턴으로 분류하여 활용할 수 있다. 패턴을

분류하기 위해 UML 클래스다이어그램으로 패턴을 추출하고, 추출된 패턴을 기준으로 검색하여, 분류와 인덱싱하는 과정을 다루는 것이 "인덱스 순서관계정보"이다.

3.3 패턴인덱싱 과정

패턴 인덱싱 과정은 세 가지로 구분하는데 ,패턴이 어떠한 역할을 수행하는가에 따른 역할정보와 패턴이 어느 상황에 적용될 수 있는가에 대한 개발자간의 경험적 상황을 고려해서 분류하는 기능인 기능적 인덱싱, 패턴을 규격화시키는 순서기준 인덱싱이다. 이것은 패턴간의 관련성을 쉽게 찾을 수 있도록 패턴을 규격화하여 상속성(inheritance)을 가짐으로 시스템의 복잡성을 줄일 수 있었다. 패턴검색은 패턴정보저장소에 저장되어 있는 패턴을 효율적으로 검색하여 최상의 관련된 패턴을 찾아내 재사용 할 수 있도록 하고 관련 패턴검색 패싯 검색, 인덱스 검색등 여러 항목을 고려해서 검색함으로서 패턴을 효과적으로 찾아 구조적인 추출의 단점을 보완할 수 있다. 패턴관리는 패턴을 관리하고, 패턴정보를 등록하는 역할을 한다. 패턴정보저장소는 패턴을 저장시키는 곳으로 추가패턴을 등록시킬 수 있으며 기본패턴은 따로 저장되어 있다. 이런 정보를 데이터베이스화한 것이 패턴 정보 저장소이고 데이터베이스와 JDBC과 연동된다. 또한 메타정보 데이터베이스를 두어 각 클래스와 속성, 오퍼레이션, 관계 표현정보를 데이터베이스화한다. 디자인패턴을 UML로 모델링하고 패턴 구조를 메타 데이터로 저장하여 패턴 메타 데이터베이스로 구축하였다. (그림6)은 패턴의 인덱싱과정 알고리즘 전체 구성도를 나타내고 있다.



(그림 6) 패턴 인덱싱과정 알고리즘 구성도

JDBC thin driver을 이용한 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;
        }
```

4. 결론 및 향후 연구과제

본 논문는 웹기반 인덱스순서관계정보 모델을 설계 제시한다. 저장된 텍스트 파일 중에서 필요한 파일만을 선별하여 Oracle Database에 저장한다. 이 저장된 곳은 패턴정보 저장소로 이 데이터베이스와 JDBC를 연결하여 필요한 정보를 파싱하여 JFC(Swing)을 이용하여 RDBMS와의 연동을 기반으로 한 application 개발 설계에 초점을 두었으며 빠르고 안정적으로 원격지 상에서의 사용자 사용환경에 중점을 두었다.

인덱스순서관계정보 모델을 제시하고 패턴 메타 정보를 구축할 수 있는 패턴 정보저장소를 구축하였다. 패턴을 메타 모델링한 데이터베이스에 저장하여 설계하는데 재사용이 가능한 시스템을 제시하였다. 이 과정의 장점은 패싯항목과 패턴간의 관련성을 쉽게 파악 할 수 있도록 패턴을 코드화함으로써 효율적인 분류, 추출, 관리를 할 수 있어 설계 경험을 공유하고, 설계정보 검색이 용이해졌으며, 정확하게 분류하여 정확도를 높여 설계정보를 재사용 할 수 있도록 하였다.

향후 연구과제로서 이 모델을 이용하여 시스템을 구현함으로서 정확도와 응답시간을 측정하여 효율성을 입증하는 것이고 제안의 모델을 다른 시스템과 비교분석하여 검증을 받는 것이다.

참 고 문 헌

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

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

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

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

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

[6] Eric Gamma,Richard Helm, Ralph Johnson, John Vlissides,"Design Patterns : Element of Reusable Object-Oriented Software", Addison-Wesley, 1995

[7] Grady Booch, Ivar Jacobson, and James Rumbaugh."Unified Modeling Language". Rational Software Corporation. January 1997. Version 1.0.

[8] Jagdish Bansiya, "Automating Design-Pattern Identification", Dr Dobb's Journal June, 1995