산출물 추출 및 분류를 위한 Index/XML 순서관계 시스템 설계

선 수 균†

요 약

소프트웨어 개발은 다양한 산출물(클래스 부품, 클래스 다이어그램, 폼, 객체, 디자인 패턴)을 생성한다. 본 논문은 이런 산출물의 효율적인 추출 및 분류를 위한 Index/XML 순서관계 시스템을 제안한다. 이 시스템에서 산출물 순서 관계 추출은 패턴 관계정보를 메타 모델링 할 수 있으며 데이터베이스 할 수 있어 재사용 및 저장이 용이하다. 이 Index/XML 순서관계 시스템은 산출물의 추출과 분류를 위한 여러 가지 산출물의 관계·정보를 쉽게 변형할 수 있다. 이 시스템은 디자인 패턴을 효율적으로 분류 추출할 수 있도록 설계한다. 기능적인 인덱싱, 표준 패턴을 위한 순서 기준 인덱싱은 인덱스 아이디로 그룹화 할 수 있으며 분류할 수 있어 효과적이다. 이 정보를 이용하여 산출물들을 효과적으로 분류 및 추출을 할 수 있다.

A Design of Index/XML Sequence Relation Information System for Product Abstraction and Classification

Su-Kyun Sun†

ABSTRACT

Software development creates many product that class components, Class Diagram, form, object, and design pattern. So this paper suggests Index/XML Sequence Relation information system for product abstraction and classification, the system of design product Sequence Relation abstraction which can store, reuse design patterns in the meta modeling database with pattern Relation information. This is Index/XML Sequence Relation system which can easily change various relation information of product for product abstraction and classification. This system designed to extract and classify design pattern efficiently and then functional indexing, sequence base indexing for standard pattern, code indexing to change pattern into code and grouping by Index-ID code, and its role information can apply by structural extraction and design pattern indexing process, and it has managed various products, class item, diagram, forms, components and design pattern.

키워드 : JiKU 객체 관리 저장소(Object Management repository), 산출물(Product), UML(Unified Modeling Language), 관계정보(Relation_information), Index/XML 순서관계 시스템(Index/XML Sequence Relation Information System), 추출 및 분류(Abstraction and Classification), 디자인패턴(Design Pattern), 클래스 다이어그램(Class Diagram)

1. 서 론

소프트웨어를 개발하는데 필요한 많은 객체들이 요구되고 있으며, 다양한 산출물이 생성된다. UML은 OMG라는 표준화단체에서 승인된 노테이션의 표준 규격이다[1, 5, 11]. 그 과정 중에서 클래스 다이어그램, 객체, 폼, 원시 코드, 모델링 방법, 디자인패턴, 사용자 인터페이스 객체, 데이터베이스 스키마, 정보저장소등의 다양한 산출물이 생성된다[4, 8]. 뿐만 아니라 재사용을 위한 처리기는 구조적 방법론을 사용한 Teamwork과 마찬가지로 다양한 산출물(Product : 클래스 부품, 다이어그램, 양식, 부품, 디자인패턴)을 모두 관리하지 못하는 단점이 있다[13, 15, 16].

본 논문은 이런 산출물을 효율적으로 관리하는데 초점을 둔다. 현재 미국 내의 설계 패턴 컨퍼런스인 PLoP(Pattern Languages of Programs)과 유럽의 설계 패턴 컨퍼런스인 EuroPLoP를 통해서 공식적으로 발표되어 알려져 있는 디자인패턴(Design Pattern)은 수백 가지에 이르며 계속적으로 증가하고 있다[6]. 이렇게 증가하고 있는 패턴의 재사용성을 향상시키기 위해서 디자인패턴의 효율적인 추출, 검색, 및 분류가 필수적이다. 그래서 디자인패턴 분야에 관한 분류는 가장 잘 분류된 Erich Gamma의 생성패턴, 구조패턴, 행위패턴이 23개 있는데 계속 증가하는 패턴들을 효율적으로 분류하기에는 부족하다. 뿐만 아니라 개발에 필요한 많은 산출물을 객체와 객체간의 구조와 객체간의 내부 모델간의 관련된 정보를 객체 내부 메타 모델링하기가 어렵다. 이에 따라 클래스 추출에 있어 관련성 있는 사항을 함께 추출하지 못하는 단점과 클래스 다이어그램의 계층 정

† 종신회원 : 동원대학 e-business과 교수
논문접수 : 2003년 11월 25일, 심사완료 : 2004년 7월 19일

보와 관계정보를 쉽게 찾을 수 없었다.

따라서 본 논문은 객체지향 프로그램을 개발하는데 발생하는 산출물을 효율적으로 추출 및 분류하기 위한 Index/XML 순서관계 시스템을 설계한다.

Index/XML 순서관계 시스템은 산출물 순서관계 추출 설계, 산출물 인덱싱 설계로 설계한다. 특히 순서관계정보 단계는 형성된 클래스 다이어그램의 관계정보를 저장표기로 저장할 수 있어 설계단계에서 클래스간의 관계정보를 효율적으로 관리 할 수 있다. 이에 따라 "Index/XML 순서관계 시스템"로 클래스(class)들 사이의 관련된 여러 관계정보를 UML 설계방법에 적용할 수 있는 구조로 변형함으로서 개발자가 관계정보를 쉽게 파악할 수 있다. 그 다음으로 산출물 인덱싱 설계를 함으로서 모델링 설계언어인 UML 설계방법에 쉽게 적용할 수 있으며 개발에 필요한 많은 산출물을 객체와 객체간의 구조와 객체간의 내부 모델간의 관련된 정보를 객체 내부 메타 모델링하기가 더욱 쉬워진다. 뿐만 아니라 이것은 디자인패턴을 구성하고 있는 UML 관계 관련성에 대한 순서관계추출인 구조적 추출을 할 수 있다. 그 결과 산출물 인덱싱 설계인 디자인패턴 인덱싱과정으로 패턴이 어느 상황에 적용될 수 있는가에 대한 효율적인 분류와 추출을 할 수 있다. 또한 패턴을 규격화시키는 순서기준 인덱싱로 디자인패턴을 패싯분류 항목으로 코드화시킴으로서 산출물들을 효율적으로 추출 및 분류를 할 수 있다. 소프트웨어 개발의 생산성을 더욱 높일 수 있도록 Index/XML 순서관계 시스템 모델을 제시한 것이 본 논문의 목적이다.

2. 관련 연구

2.1 객체지향 시스템에서의 디자인 패턴 추출

디자인패턴은 순환되는 조직이고, 객체지향 시스템의 객체(object)의 커뮤니케이션에서 발견할 수 있다. 패턴을 추출, 활용하기 위해서는 반복되는 패턴의 정보를 파악해야한다. 또한 각 클래스의 역할에 대한 정보를 제공하는 클래스 다이어그램의 이해와 유지보수, 패턴 구성요소 사이의 확실한 관계를 알아야 한다. 개념 분석(Concept analysis)은 공통적인 속성을 가지는 객체를 그룹화 하는 것을 가능하게 한다[12]. 즉 개념들을 객체로 그룹화 할 수 있다.

Paolo가 제안한 논문은 일반적인 관계를 공유하는 클래스 그룹을 확인하기 위하여 개념 분석(Concept analysis)을 이용한다[12]. 이 방법은 클래스다이어그램으로부터 디자인 패턴을 추출하기 위해 효과적으로 사용될 수 있다. 개념 분석은 일반적인 속성을 가지고 있는 객체의 그룹을 허용한다. 객체는 클래스들의 그룹이고, 속성은 클래스들 사이의 관계이다. 개념 분석은 이들의 관계 문맥(context)에서 시작한다. 패턴 추출은 객체의 배치, 속성들의 배치, 객체와 속성사이의 이진(binary) 관계, 속성은 각 객체에 의해 소유했던 문맥에서 시작한다. 존재하는 클래스다이어그램을 위해 두개 클래스의 이진관계는 관계(상속과 연관)에 의하여 연결된다. 속성의 배치를 공유하는 모든 객체의 그룹을 (X, Y)로 표현한다.

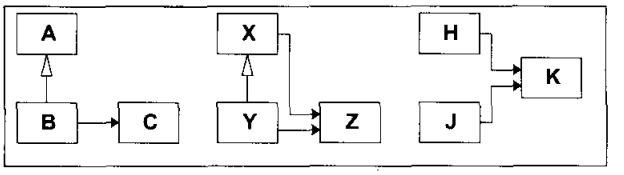
$$X=\{o\in O \mid \forall a\in Y\ ;\ (o,a)\in P\}$$

$$Y=\{o\in A \mid \forall a\in X\ ;\ (o,a)\in P\}$$

위의 식에서 O는 객체의 집합, A는 속성의 집합, P는 객체와 속성 사이의 이진관계이다. X는 Concept의 클래스의 확장을 말하고, Y는 연관관계를 표시한다.

(그림 1)은 클래스다이어그램의 구조를 나타낸 것이다. 즉, 클래스 사이의 관계들은 클래스 관계 R로 표현할 수 있다. 일반적인 관계를 공유하는 클래스 그룹의 배치로 구성된다.

$(Y, X)_e \in R$는 클래스 Y와 X 사이의 확장관계, $(H, K)_a \in R$는 클래스 H와 K 사이의 연관관계로 표현할 수 있다.



(그림 1) 상속과 연관 관계를 포함되는 클래스다이어그램

순서 안에서 집단화된 클래스들은 이름에 의한 것보다 순서 인덱스에 의해서 위치될 수 있다. 연관된 속성은 순서 인덱스의 쌍이 된다. (그림1)에 참조하여 보면, 두 개의 클래스의 순서 (B, A, C)와 (Y, X, Z)는 $(1,2)_e$와 $(1,3)_e$ 같은 일반적인 속성을 갖는다. 첫 번째 클래스 B, Y는 두 번째 클래스 A, X에 확장되는 관계이고, 첫번째 클래스 B, Y는 세번째 클래스 C, Z에 연관관계가 된다. 클래스들의 순서에 대한 관계 정보를 얻을 수 있다.

2.2 McCabe 측정법

McCabe의 순환 복잡도(cyclomatic complexity)는 프로그램의 논리적인 복잡도의 분량상 측정을 제공하는 소프트웨어 척도이다.

〈표 1〉 McCabe 방법

- 흐름도의 영역 수는 순환 복잡도와 일치한다.
- 흐름도 G에 대한 순환 복잡도 V(G)는 다음과 같이 정의한다

$$C(G) = E - N + 2P$$

C : 사이클매트릭스　　E : 관계의 수
N : 클래스의 수　　P : 외부 모듈 수

- 흐름도 G에 대한 순환 복잡도 V(G)는 다음과 같이 정의한다.

$$V(G) = P + 1.$$

기본 경로 시험 방법의 내용으로 사용될 때, 순환 복잡

도에 대해 계산된 값은 프로그램의 기본 집합에 독립적인 경로들의 수를 수정한다. 순환 복잡도는 그래프 이론에 기본을 두고 있으며, 아주 유용한 소프트웨어 척도를 우리에게 제공한다[3].

3. JiKU/XML 객체 관리 저장소

본 장에서는 효율적인 객체 관리와 객체와 객체간, 객체 내부를 모델링하여 UML 클래스다이어그램으로 표현하여 저장표기로 저장함으로서 패턴화를 이룰 수 있는 것이 JiKU/XML 객체 관리 저장소이다[17].

객체 관리 저장소의 구성은 데이터(Data) 단계, 프로세서(Process) 단계, 뷰어(Viewer)단계, 뷰어(Viewer) 단계,버전(Version) 단계로 되어 있다.

이 버전 단계는 객체에 대한 메타 모델링, 객체간의 관계에 대한 메타 모델링, 객체내부에 대한 메타 모델링을 설계하여 클래스다이어그램을 UML로 도식화하기가 편리하다. 이 버전 단계의 마지막 단계에서는 뷰어단계와 프로세서단계에서 형성된 부품의 다이어그램 좌표값, 부품 다이어그램 표현으로 클래스 다이어그램(Class Diagram)을 생성한다. 이 클래스 다이어그램을 객체 관리 저장소에 저장하기 위해서는 UML로 도식화해야하고 다이어그램의 관계를 Index/XML 순서관계 시스템저장표기로 간략화하여 가장 비슷한 디자인 패턴을 추출할 수 있고 분류할 수 있어 설계 재사용에 용이하다.

따라서 Index/XML 순서관계 시스템은 이러한 기능을 바탕으로 해서 클래스 부품을 보다 효율적으로 저장하기 위한 객체를 추출하는 동시에 UML클래스 관계정보 추출을 더 보완하여 연결관계를 형성할 수 있는 기능을 더 추가한 것이다. 이것은 객체와 객체간, 객체 내부간의 메타 모델링한 후 UML로 도식화할 수 있다. 산출물에 대한 메타모델링 결과를 UML로 표현할 수 있다. 객체 내부의 메타모델링에서는 양식이나 다이어그램의 내부 내용 구조를 세부적으로 표현하거나 특정 양식이나 다이어그램간에 존재하는 종속(dependency) 관계를 표현 할 수 있다.

4. Index/XML 순서관계시스템 설계

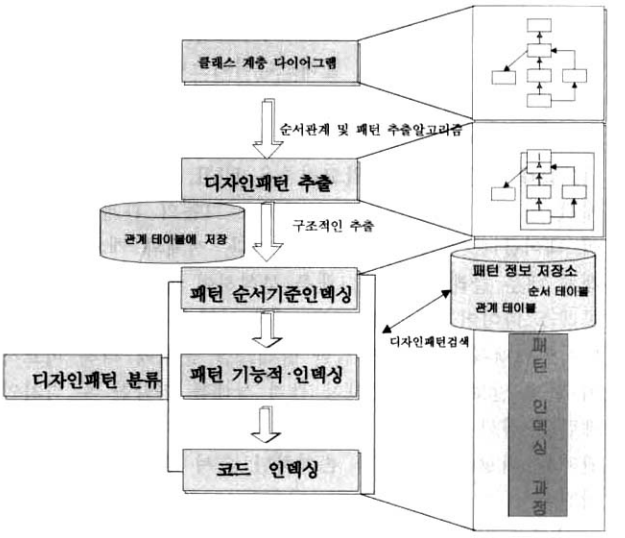
본 논문에서는 많은 산출물을 효율적으로 관리하기 위해 Index/XML 순서관계 시스템을 설계한다. 이 시스템은 두 가지로 나뉠 수 있는데 산출물 관계순서 설계와 산출물 인덱싱 설계이다. 이것은 클래스의 상호관계와 메타모델링(meta-modeling) 기반으로 한 객체와 객체간의 관계, 객체 내부간의 관계를 패턴화함으로써 디자인 패턴을 효율적으로 분류하고 추출할 수 있는 시스템이다. 또한 이 Index/XML 순서 관계 시스템은 객체관리 저장소 설계의 단점을 보완한 UML 관계정보를 효율적으로 추출할 수 있다. 뿐만아니라 관계 저장정보를 쉽게 찾을 수 있어 설계 패턴화 할 수 있기 때문에 소프트웨어 개발하는데 생산성을 높일 수 있다.

이 시스템은 정보의 상세화와 연결관계에서 구현하기 어려운 양방향 관련성도 쉽게 정의할 수 있게 설계한다.

4.1 산출물 순서 관계 추출 설계

산출물 순서 관계 추출 설계는 클래스간 순서에 의한 Index 순서 관계 추출을 설계한다. 이 순서관계 시스템은 (그림 2)에 있는 디자인패턴 추출과 구조적인 추출(물리적)로 클래스 순서에 의한 관계(Relation) 추출 알고리즘을 실행한다. UML 클래스다이어그램을 분석하여 보면, 많은 클래스들이 관계를 가지고 순환되고 있다. 클래스 관계를 추출하기 위해서 각 클래스의 순서(sequence)와 관계에서 시작된다. 전체 클래스다이어그램에서 각 클래스의 순서 정보와 관계정보를 파악하여, 저장소에 저장한다.

객체의 위치를 도형화하여 다이어그램을 형성한다. 이것을 UML로 도식화하기 위한 설계부분에 순서관계 정보를 이용한 디자인패턴을 이용한다. 이 적용은 클래스 관계(Relation) 추출 알고리즘을 적용하여 실행한다.



(그림 2) Index/XML 순서관계 시스템 구성도

객체와 객체간, 또는 객체 내부 클래스다이어그램을 분석하여 보면 많은 클래스들이 관계를 가지고 순환되고 있다. 클래스 관계를 추출하기 위해서 각 클래스의 순서(sequence)와 관계에서 시작된다. 먼저 객체와 객체간의 전체 클래스다이어그램에서 각 클래스의 순서정보와 관계정보를 파악하여 저장소에 저장한다. 디자인 패턴 추출은 (그림 2)에서 클래스 다이어그램에서 클래스간의 순서관계 비교 알고리즘에 의해서 관계 추출을 할 수 있다. 따라서 객체와 객체간, 또는 객체 내부 안에 있는 클래스 사이의 순서는 (α, β)와 같은 순서쌍으로 표현 할 수 있다. 클래스 α에서 β사이로 관계가 존재한다는 의미이다.

이 과정은 클래스다이어그램에서 존재하는 클래스 사이의 관계가 클래스 α와 β사이의 일반화 관계(Generalization : G)가 존재할 때, $(\alpha, \beta)_G$과 같이 표기할 수 있다. 이것을 저

장소에 저장할 때는 $G(\alpha$클래스 위치, β클래스 위치)에 저장된다. UML에서 클래스 사이에 존재하는 여러 가지 관계를 나타낸 것이 <표 2>이다.

〈표 2〉 클래스다이어그램에서의 관계 표시

관 계	기 호	저장표기
연관(association)	——→	S
집합(Aggregation)	——◆	A
복합(Composite)	——◇	C
의존(Dependency)	----→	D
일반화(Generalization)	——▷	G
실체화(Realization)	----▷	R

여기에서 이론 표기는 이론적으로 표기할 때 사용하는 것으로 저장소에 저장할 때는 저장표기로 저장된다. (그림 2)의 상단 부분은 UML 클래스다이어그램에서 클래스간의 관계의 추출 과정을 도식화 한 그림이다.

(리스트 1)은 클래스 순서 비교 알고리즘을 나타낸 것이다. 클래스 순서 비교 알고리즘은 UML 클래스다이어그램에서 인접한 두 클래스를 조사하고 그들의 관계를 파악하여 테이블에 저장하는 알고리즘이다. 객체와 객체간, 또는 객체 내부 클래스다이어그램을 분석하여 형성된 것이 UML 클래스 다이어그램이다. 다음 진행 사항은 이 UML 클래스 다이어그램에서 서로 인접한 클래스를 순서에 의해 비교하여 관계 알고리즘을 통하여 관계 상태를 파악한 후 디자인 패턴을 검색한다. 이것이 끝나면 모든 클래스 사이의 모든 관계를 비교하여 관계가 존재하면 순서 쌍 테이블에 저장시킨다.

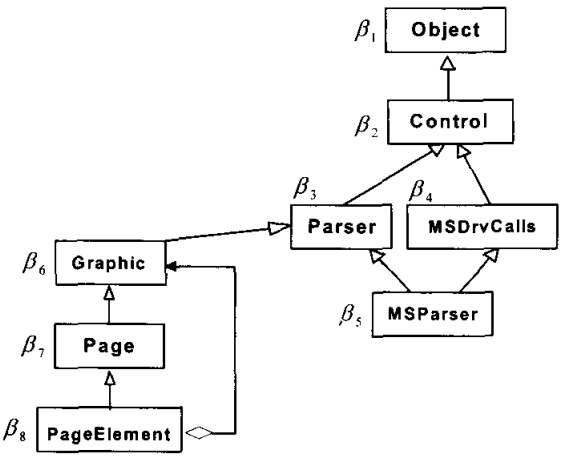
우선 패턴을 추출하기 위해서 디자인패턴 테이블과 UML 클래스의 테이블을 비교한다. 우선순위를 검색하여 클래스 순서를 비교하여 동일한 순서를 가진 클래스들의 정보를 가지고 관계(Relation)를 조사하여 동일한지 비교한다.

(1) DTK 클래스다이어그램에서의 패턴추출

디자인패턴의 적용사례를 들어 디자인패턴의 추출하는 과정을 나타내는 예로 Jagdish Bansiya의 Drawing Tool Kit(DTK)의 클래스 계층 다이어그램을 도입한다. DTK 시스템은 다양한 출력장치에 사용하기 위해 어플리케이션 소프트웨어로부터 그림을 프린트, 저장하기 위해 사용되는 시스템의 일부이다. PostScript, HPGL, TIFF, GIF, JPG 등등의 파일을 볼 수 있고, 사용자가 원하는 종류의 그림 파일로 변환하여 주는 프로그램이다.

1) A 클래스다이어그램 1

클래스다이어그램으로부터 패턴을 분석하기 위해 순서에 의한 관계비교 알고리즘으로부터 클래스 순서와 관계에 대한 정보를 분석한다. 클래스 β_2에서 β_1 관계를 살펴보면 일반화 관계, $(\beta_2, \beta_1)_G$ 관계, ***G(2, 1)***로 저장된다.



(그림 3) DTK 클래스다이어그램 1

β_3, β_4에서 β_2로의 일반화 관계를 비교하여 (β_2, $\beta_3)_G$ 은 ***G(2, 3)***, (β_4, $\beta_2)_G$ 관계는 ***G(4,2)***으로 구성된다. β_5에서 β_3, β_4의 관계를 비교하여 (β_5, $\beta_3)_G$, (β_5, $\beta_4)_G$, ***G(5, 3), G(5, 4)*** 관계임을 분석해 낼 수 있다. 각각을 비교하면 (β_6, $\beta_3)_G$, (β_7, $\beta_6)_G$, (β_8, $\beta_7)_G$, (β_6, $\beta_8)_G$, (β_8, $\beta_6)_G$의 관계가 있다. 분석이 끝난 후 패턴 정보 저장소에 있는 Gamma의 기본 패턴을 먼저 체크한다.

(리스트 1)는 (그림 2)의 상단에서 제시한 순서관계 알고리즘이다. 이것을 이용하여 클래스 순서관계 및 패턴 추출 알고리즘에 적용하여 비교한다.

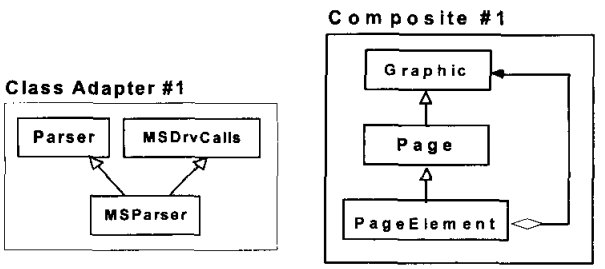
```
1 : Repeat
2 : set D_a  : /* 디자인패턴 테이블의 마지막 레코드 */
3 : set T_b  : / * 테스트 패턴 테이블의 마지막 레코드 */
4 : set ( α , α ) : /* 디자인패턴 클래스의 관계  */
5 : set ( β , β ) : /* : 테스트 패턴 클래스의 관계  */
6 : set  n, m  : /*  T 테이블 모든 클래스와 비교
7 : Until  total_count
8 :  If design pattern of index Not exist
9 : while ( D_1 ← D_a  ){ //  D 테이블 모든클래스와 비교
10 :   while ( T_1 ← T_b ){ // 클래스 순서 비교
11 :  If (α_n , α_n+1)와 (β_n , β_n+1) 순서를 비교 동일  then
12 :                    // 클래스 관계 비교
13 :    if  ...  와  ...  관계비교동일
14 :              then
15 :          동일 패턴 → 새로운 테이블 저장
16 :                    else
17 :                  동일 패턴 아님
18 :     endif
19 :                    else
20 :                  동일 패턴 아님
21 :                endif
22 :                    }
23 :              }
24 :      Endif
25 :     Read  index from repository
```

```
26 :        Else  /* index exist */
27 :     Repeat
28 :       retrieval  index-id
29 :     Until
30 : Endif\
```

(리스트 1) 클래스 순서비교 및 디자인패턴 추출 알고리즘

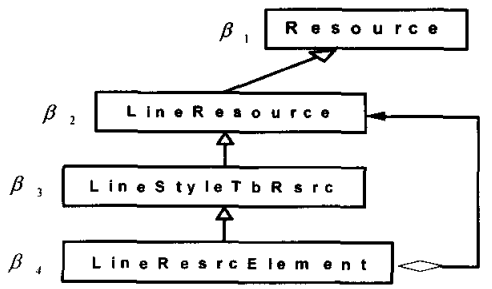
패턴 정보 저장소에 저장된 클래스 어댑터 패턴의 클래스 간의 관계 $(\alpha_2, \alpha_1)_G$, $(\alpha_2, \alpha_3)_G$와 $(\beta_5, \beta_3)_G$, $(\beta_5, \beta_4)_G$의 순서 관계가 동일함을 알 수 있는 것은 순서기준 패턴인 G(*,0) G(*,0)가 같기 때문이다. 즉, $(\alpha_2, \alpha_1)_G$, $(\alpha_2, \alpha_3)_G$은 패턴정보 저장소에 있는 정보이다. 따라서 이 정보와 $(\beta_5, \beta_3)_G$, $(\beta_5, \beta_4)_G$이 같기 때문에 관련된 패턴이라고 할 수 있다. β_3, β_4, β_5은 클래스 어댑터 패턴의 참여자(participate)이다. Parser(β_3)클래스는 adaptee 클래스, MSDrvCalls(β_4)는 GoF에 의해 기술된 어댑터 패턴 솔루션의 target 클래스이다.

또한 컴포지트 패턴의 클래스간의 관계 $(\alpha_1, \alpha_3)_G$, $(\alpha_2, \alpha_1)_G$, $(\alpha_3, \alpha_1)_{AS}$, $(\alpha_3, \alpha_1)_C$는 $(\beta_7, \beta_6)_G$, $(\beta_8, \beta_7)_G$, $(\beta_8, \beta_6)_{AS}$, $(\beta_8, \beta_{46})_C$ 순서 관계가 동일하다. Page(β_7)가 Graphic(β_6) 클래스에서 상속 분기의 중간 클래스일 동안, PageElement(β_7)는 Graphic(β_5) 클래스를 위해 되풀이 되는 역할을 한다. 이러한 순서 관계의 비교에 대해 클래스다이어그램 1에서는 (그림 4)와 같이 클래스 어댑터 패턴과 컴포지트 패턴을 추출할 수 있다.



(그림 4) A클래스다이어그램 1에서 추출된 패턴

2) 클래스다이어그램 2



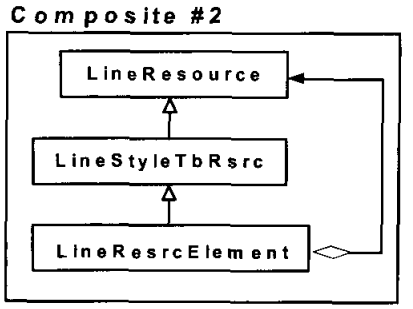
(그림 5) DTK 클래스다이어그램 3

같은 방법으로 (리스트 1) 클래스 순서 비교 알고리즘을 이용하여 관계를 비교하면 아래의 관계테이블을 만들 수 있다.

	$(\beta_2, \beta_1)_G$	$(\beta_3, \beta_2)_G$	$(\beta_4, \beta_3)_G$	$(\beta_4, \beta_1)_{AS}$	$(\beta_1, \beta_4)_C$
C	√	√	√	√	√

(그림 6) 클래스다이어그램 3의 관계테이블

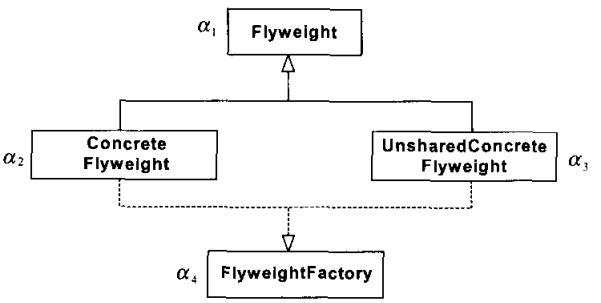
디자인패턴의 클래스 관계 테이블과 (그림 6)의 클래스다이어그램 2의 관계 테이블을 패턴 추출 알고리즘을 적용하여 비교한다. 하나의 컴포지트 패턴이 추출된다. 이런 방법을 적용하여 (그림 14)과 같이 클래스 어댑터 1개, 컴포지트 1개, 플라이웨이트 2개의 패턴이 DTK 클래스다이어그램이 추출되었다. (그림 14)은 발견된 6개의 패턴과 참여하고 있는 클래스를 나타내고, 캡슐화 위에 패턴의 이름을 표시하였다.



(그림 7) 클래스다이어그램 3에서 추출된 패턴

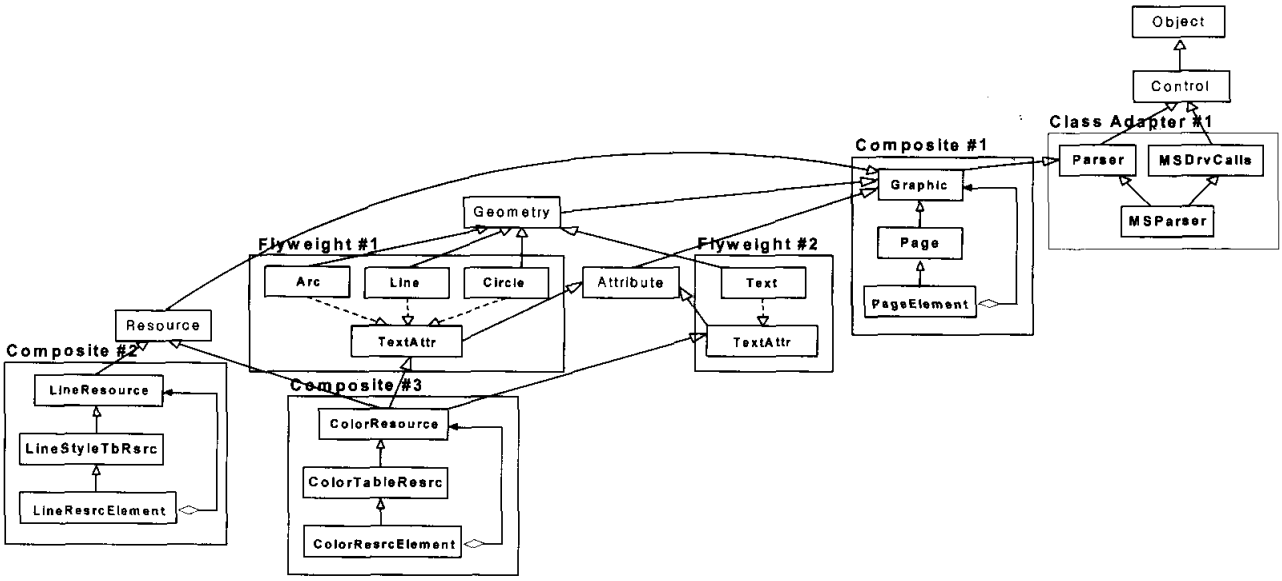
4.2 산출물 인덱싱과정 설계

디자인패턴 인덱싱과정은 패턴의 역할정보와 패턴이 어느 상황에 적용될 수 있는가에 대한 순서기준을 정하는 순서기준 인덱싱과 UML 클래스 다이어그램을 XML 문서로 변환하는 기능적 인덱싱, 패턴을 규격화시키는 순서기준 인덱싱, 패턴을 코드화시키는 코드 인덱싱, 그리고 Index-ID생성 과정으로 설계한다. 순서기준 인덱싱 설계는 다음과 같다.



(그림 8) 플라이웨이트 패턴구조

(그림 8)은 플라이웨이트(Flyweight) 패턴을 나타낸 것인데, 이 클래스 순서에 의한 관계를 살펴보면, α_2에서 α_1은 일반화 관계, 즉 *G(2, 1)*, α_3에서 α_1은 일반화 관계, *G(3, 1)*, α_2에서 α_4는 연관 관계, *S(2, 4)* α_3에서 α_4는 연관 관계, *S(3, 4)* α_4에서 α_1은 일반화 관계, *G(4, 1)* α_1에서 α_4는 복합 관계 *C(1, 4)*를 이루고 있다.



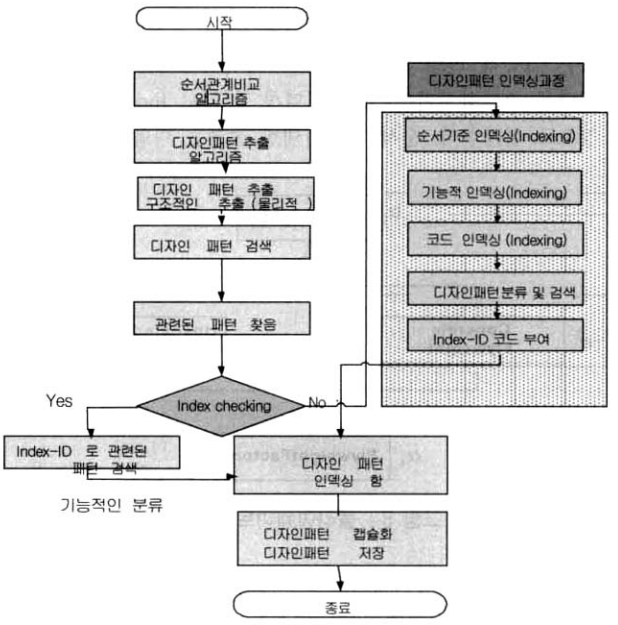
(그림 9) DTK 클래스다이어그램에서 추출된 패턴

G(2,1) G(3,1) S(2,4) G(4,1) C(1, 4)

(그림 10) 디자인패턴 클래스 관계 테이블

Flyweight 순서 기준 패턴
G(0,) G(0,*) S(0,*) G(*,0) C(0,*)*

(그림 11) 플라이웨이트구조와 순서기준패턴

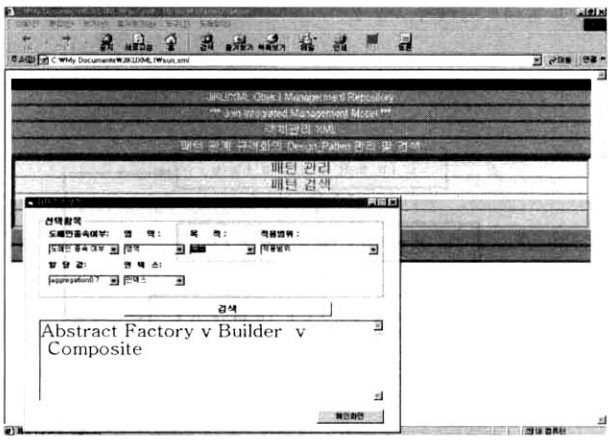


(그림 12) 산출물 인덱싱과정 설계

이러한 관계를 저장소에 저장시키는 디자인패턴의 클래스다이어그램을 분석하여 (그림 11)과 같은 순서 기준 패턴을 설계할 수 있고 테이블을 만들 수 있다. 새롭게 추가되는 패턴은 구조를 일반화하여 클래스간의 관계를 클래스 순서 비교 알고리즘을 사용, 비교하고 새로운 관계를 저장할 수 있다.

디자인패턴 분석은 UML 클래스다이어그램으로부터 클래스 순서에 대한 관계 정보를 파악하는 것에서 시작된다. 먼저 Index-ID를 체크한다. 이 과정은 사용자 추가 패턴이 Index-ID를 가지고 있는지를 먼저 체크 후 기본 패턴의 순서 관계 테이블을 검색한다. 예를 들어 기본 패턴인 프록시 패턴을 패턴 정보 저장소에 저장 할 때는 ***G(2, 1) S(3, 2)*** 로 저장소에 저장하고 순서기준 패턴이 자동으로 정해진다. 또한 기본 패턴인 프록시 패턴을 검색하려면 "순서기준 패턴"인 ***G(*, 0) S(0, *)***으로 검색하면 된다.

(그림 12)은 산출물 인덱싱과정을 생선한 결과로 생성된 화면으로 여기서 패턴 수와 질의어 수를 나타나는데 질의어 형태는 디자인패턴 검색을 위한 질의어에 의하여 디자인패턴 검색 질의어 형태를 나타낸 것이다.



(그림 13) 산출물 인덱싱과정 결과화면

기능적 인덱싱 설계는 UML 클래스 다이어그램을 인터넷 문서의 표준으로 자리잡은 XML 문서 형태로 변환시킬 수 있도록 한다. <리스트>는 Iterator 패턴을 추출하여 XML 문서로 변환한 결과를 나타낸 것이다.

```
<?xml version = '1.0' ? encoding = "EUC-KR"?>
<?xml : stylesheet type="text/xsl" href="Iterator.xsl"?>
<LIBRARY>
   <pattern name='Iterator'>
     <structure>
        <relations>
        <inheritance origin='ConcreateIterator'
                        target='Iterator'></inheritance>
        </relations>
        <roles>
        <role syslabel='Aggregate' abstract='abstract'>
        <operations>
        <operation constuctor='constructor' override='do' access
          ='public' return='nonreturn' syslabel='CreteleIterator'>
                        </operations>
        </role>
        <role syslabel='Client' abstract='concrete'>
        </role>
        <role syslabel='Iterator' abstract='abstract'>
        <operations>
        <operation constuctor='constructor' override='do' access
          ='public'return='nonreturn' syslabel='First'></operation>
        <operation constuctor='constructor' override='do' access
          ='public'return='nonreturn' syslabel='Next'></operation>
        <operation constuctor='constructor' override='do' access
          ='public'
        </operation>
        </role>
        <role syslabel='ConcreteIterator' abstract='abstract'>
        <layout rows='2' columns='3'>
        <box name='Aggregate'row='1'column='1'></box>
        </layout>
        </structure>
        </pattern>
</LIBRARY>
```

(리스트 3) Iterator 패턴의 XML 변환 결과

또한 <리스트>는 Iterator 패턴을 순서기준 인덱싱한 것을 XSL 문서로 변환한 결과를 나타낸 것이다.

```
<?xml version="1.0" encoding="euc-kr"?>
<xsl : stylesheet xmlns : xsl="http : //www.w3.org/TR/WD-xsl">
<xsl : template match="/">
<html>
    <head><title> iterator </title>
      <style>
       .com{color : blue;}
      </style>
    </head>
    <body>
<xsl : if test="parant[.$eq$ 'S']">
    <font><xsl : value-of select="parat"/>S(0,*)</font></xsl : if>
<xsl : if test="parant[.$eq$ 'A']">
    <font><xsl : value-of select="parat"/>A(0,*)</font></xsl : if>
<xsl : if test="parant[.$eq$ 'C']">
    <font><xsl : value-of select="parat"/>C(0,*)</font></xsl : if>
<xsl : if test="parant[.$eq$ 'D']">
    <font><xsl : value-of select="parat"/>D(0,*)</font></xsl : if>
<xsl : if test="parant[.$eq$ 'G']">
    <font><xsl : value-of select="parat"/>G(0,*)</font></xsl : if>
<xsl : if test="parant[.$eq$ 'R']">
    <font><xsl : value-of select="parat"/>R(0,*)</font></xsl : if>
.....
</body>
  </html>
</xsl : template>
</xsl : stylesheet>
```

(리스트 4) Iterator 패턴의 XSL로 변환 결과

5. 적용 사례 및 측정

본 장에서는 Index/XML 순서관계 시스템을 제안한 모델을 가지고 여러 가지방법으로 적용사례를 들어 측정한다. XML 사용으로 분산환경에 적응이 쉽게 할 수 있으며 클라이언트와 서버를 이용하여 Index/XML 순서관계 시스템 설계로 형성된 인덱싱된 디자인패턴과 인덱싱 안된 디자인패턴을 비교분석 한 후 검색하는데 응답시간을 측정한다. 추출된 패턴을 효율적으로 활용하기 위하여 패턴별로 그룹화하고, 블랙박스화 함으로서 시스템의 이해와 재사용성을 높인다.

5.1 추출 시스템 비교분석

본 연구의 성능을 평가하기 위해 Jagdish Bansiya가 제안한 "Automating Design-Pattern Identification" 방법, Paolo Tonella와 Giulio Antiniol가 제안한 "Object-oriented Design Pattern Inference"의 추출 방법과 비교한다[7,12]. Identification 방법은 클래스들의 책임과 역할에 기반을 두고 패턴을 추출한다[7]. Inferenc 방법은 클래스 다이어그램에서 클래스들 사이의 구조적인 관RP를 고려하여 동일한 관계를 공유하는 클래스들의 그룹을 추출하는 시스템이다[12].

〈표 3〉 기존 팬턴 추출 방법과의 비교

추출 방법 / 비교항목	Identification[7]	Inference[12]	본 논문의 제안방법
클래스간의 정보추출 방식	상속/포함 관계	순서관계 비교	순서관계 비교
Gamma 패턴 적용	제한적	제한적	가능
새로운 패턴 추출	불가능	불가능	가능
UML 설계법 적용	불가능	불가능	가능
추출된 패턴 캡슐화	불가능	불가능	가능

Identification, Inference와 본 논문 제한 방법을 비교해 보면 <표 3>과 같으며, 제시한 기준을 살펴보면 다음과 같다.

첫째, 클래스정보 추출 방식이다. 기본의 Identification 방법은 클래스 사이의 상속/포함 관계정보만을 가지고 있기 때문에 Gamma의 구조적 팬턴(structural pattern)에 한정되어 적용이 가능하다. Inference 방법은 클래스 사이의 관계에 따른 순서비교 방법을 따르고 있으며, 본 논문에서 제안한 방법도 동일한 방법을 사용하여 패턴을 추출한다.

둘째, UML 설계 적용 여부이다. 기존의 시스템에서는 클래스 사이의 관계 파악에 연관(association)과 확장(extend) 관계를 사용하고 있고, 본 논문에서 방법에서는 UML 클래스 다이어그램의 관계, 연관(association), 집합(aggregation), 복합(composition), 의존(dependency), 일반화(generalization), 실체화(realization)를 적용하여 정보를 추출한다. 클래스 사이의 다양한 관계를 적용함으로써 기존의 추출 방법인 Inference 방법이 갖고 있는 제약적인 단점을 보안하였다. 본 논문에서 제시한 방법은 Gamma의 23개 기본 패턴을 추출할 수 있고, UML 클래스 다이어그램에 적용하여 추출 할 수 있다. 또한 Gamma가 정의한 기본 패턴 외의 정의되지 않은 패턴을 추출하여 활용할 수 있다. 따라서 추출된 패턴을 캡슐화 함으로서 복잡도를 감소시킬 수 있고, 시스템의 분석을 용이하게 하고 개발자 사이의 커뮤니케이션을 용이하게 한다. 시스템의 모듈성을 극대화하여 개발비용, 유지보수 성능을 향상시킬 수 있다.

5.2 복잡도 측정

본 장에서는 산출물 추출에 대한 복잡도와 검색시간을 측정한다. 산출물(디자인패턴) 추출 전과 후의 클래스다이어그램의 복잡도를 McCabe 방법으로 추출한다. McCabe 방법을 이용하여 DTK와 WP의 복잡도를 측정한다.

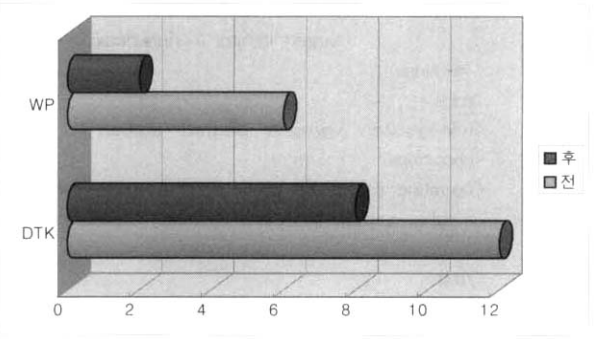
클래스 정보에서 복잡도를 측정하기 위해 McCabe가 정의한 방법을 사용한다. DTK 클래스 계층다이어그램 (그림 7)을 분석해 보면, E(관계의 수)는 33개, N(클래스의 수)는 23개이다. 계층다이어그램으로부터 패턴을 추출하여 가시화한 다이어그램 (그림 12)에서는 E(관계의 수)는 17개, N(클래스의 수)는 11개, 단일 모듈에서의 P는 1이므로 <표 4>와 같은 복잡도를 얻을 수 있다.

<표 4> 다이어그램에서의 복잡도 측정 비교

클래스다이어그램		관계(E)	클래스(N)	복잡도(C)
DTK	전	33	23	12
	후	17	11	8
WP	전	21	17	6
	후	9	9	2

두가지 UML 클래스다이어그램의 패턴 추출 전, 후의 복잡도를 계산, 비교하면 <표 4>과 같고 그림으로 비교하면 (그림 14)과 같다.

모든 다이어그램에서 디자인패턴 추출 후의 복잡도가 추출 전의 복잡도 보다 감소했음을 알 수 있다. 디자인패턴을 추출하고, 가시화함으로서 대형 시스템을 분석을 돕고, 재사용을 높이며, 개발자간의 커뮤니케이션을 용이하게 하여 생산성 향상, 소프트웨어의 유지보수 비용의 절감, 시스템에 대한 새로운 요구를 수용을 할 수 있다.



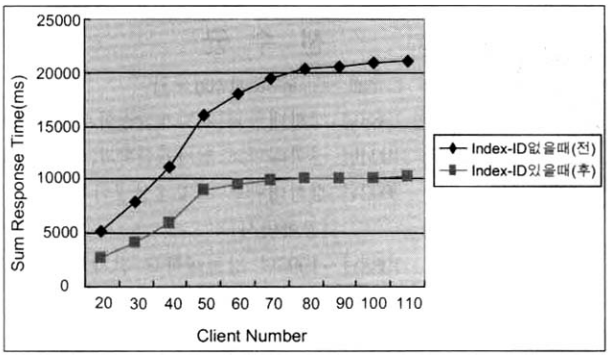
(그림 14) 복잡도 측정 비교

다른 방법으로 인덱싱된 산출물 검색 측정을 한다. 제안한 Index/XML 순서관계시스템을 이용한 디자인패턴과 질의어를 증가시키면서 인덱싱된 패턴과 인덱싱 안된 패턴과 서로 비교하여 검색함으로서 인덱싱된 패턴의 검색 응답시간으로 성능평가를 한다. 설계 패턴 컨퍼런스인 PLoP의 30개 패턴과 Erich Gamma의 23개 패턴을 사용하여 클래스의 구조와 순서 관계를 파악한 후 "클래스 순서관계비교 알고리즘"과 "패턴 추출 알고리즘"을 적용하여 구조적으로 추출한 후 디자인패턴을 구조적 추출과 디자인패턴 인덱싱과정을 한다[3, 6, 7]. 성능 평가는 인덱싱된 패턴과 인덱싱 안된 패턴들을 각각 평가대상으로 한다. 산출물이 Index ID가 있는 것과 없는 디자인패턴을 서로 측정한다.

<표 5> 인덱싱된 디자인패턴 수 증가 대비 클라이언트의 성능

디자인패턴의 수과 질의어	인덱싱 안된 클라이언트 누적응답 시간(ms)	누적응답 시간대비 개선된 성능비율(%)
100	8430	1.5%
200	8707	3%
300	9121	3.5%
400	9674	4.5%
500	10396	7%
600	12090	8.5%
700	15010	13%
800	21027	15%

따라서 <표5>는 클라이언트의 수가 110개일 때 디자인 패턴의 수와 질의어가 증가함에 따라 이전 디자인패턴의 수와 질의어에 비해 향상된 처리시간의 누적된 비율이다. 100개의 디자인패턴의 수와 질의어에 비해 800개의 디자인 패턴의 수와 질의어가 있는 경우는 15%로 응답 속도가 향상되었음을 알 수 있었다. 마지막 성능평가로 패턴이 순수한 Index-ID가 있는 것과 없는 디자인패턴을 서로 성능평가를 하였다.



(그림 15) Index-ID로 비교한 누적 응답시간의 결과

따라서 본 논문에서 제안한 디자인패턴 인덱싱과정은 패턴의 역할정보와 패턴이 어느 상황에 적용될 수 있는가에 대한 개발자간의 경험적 상황을 고려한 기능적인덱싱, 패턴을 규격화시키는 순서기준 인덱싱, 패턴을 코드화시키는 코드 인덱싱, 그리고 Index ID코드부여인 분류방법을 제시한 것으로 기존 방식의 단점을 보완했다. 뿐만 아니라 이렇게 디자인패턴을 코드화함으로 효율적으로 분류, 추출, 관리를 할 수 있어 설계 경험을 공유하고, 설계정보 검색이 용이해졌으며 설계정보 재사용 할 수 있도록 도와주는 역할을 극대화하였다.

추출된 패턴은 유지보수, 재사용, 역공학에 사용하는 장점을 갖는다. 또한 추출된 패턴을 캡슐화 함으로서 복잡도를 감소시킬 수 있고, 시스템의 분석을 용이하게 하고 개발자 사이의 커뮤니케이션을 용이하게 한다. 시스템의 모듈성을 극대화하여 개발비용, 유지보수 성능을 향상시킬 수 있다.

따라서 본 논문에서 제한 방법은 UML 모델링이 가능하고, Index 순서관계정보에 의한 분류로 자동화된 분류 방법을 사용했으며 검색시에도 스트링매칭에 의한 검색과 같은 카테고리 내에서의 유사패턴 검색 같이 병행할 수 있다.

6. 결 론

본 논문에서는 Index/XML 순서관계 시스템을 설계한다. 이 제안은 클래스들 사이의 관련된 여러 관계정보를 UML 설계 방법에 쉽게 적응할 수 있는 구조로 변형함으로서 구조적인 추출과 디자인패턴 인덱싱과정으로 나뉜다.

구조적 추출은 클래스다이어그램에서 순서관계 정보를 추출하는 알고리즘을 사용하여 데이터베이스에 저장된 패턴과 추출하려는 클래스다이어그램의 순서 관계정보를 패턴 추출 알고리즘을 사용하여 비교한다. "Index 순서관계정보"의 디자인패턴 인덱싱과정으로 패싯항목과 패턴간의 관련성을 쉽게 파악 할 수 있도록 패턴을 코드화시켜 디자인패턴을 효율적으로 분류하고, 디자인패턴의 재사용성(*reusability*)을 극대화시킬 수 있는 방법을 제시한 것이다.

"Index 순서관계정보"의 효율성을 입증하기 위해 설계패턴 컨퍼런스인 PLoP의 30개 패턴과 Erich Gamma의 23개 패턴을 사용하여 클래스의 구조와 순서 관계를 파악한 후 "클래스 순서관계비교 알고리즘"과 "패턴 추출 알고리즘"을 적용하여 구조적으로 추출한 후 디자인패턴을 구조적 추출과 디자인패턴 인덱싱과정을 측정하였다.

복잡도를 측정한 결과 DTK는 복잡도가 12에서 8로 복잡도가 감소하여 30%정도 줄었다. Index가 없는 디자인패턴보다 Index가 있는 디자인패턴이 약15% 향상되었다. 따라서, "Index 순서관계정보"는 디자인패턴을 재사용할 때 관련된 패턴을 빠르게 찾고, 정확하게 분류하여 정확도를 높였으며 기능적 인덱싱으로 패턴 추출도 가능하기 때문에 설계정보를 쉽게 재사용할 수 있어 소프트웨어 생산성을 극대화 할 수 있었다. 또한 패턴을 코드화로 재사용이 용이하며 웹 환경하에서 적용함으로써 여러 사용자가 공유하여 설계정보를 재사용할 수 있다는 장점이 있다.

앞으로 연구 과제는 캡슐화, 추상화, 상속성, 다형성과 같은 개념을 적용한 UML 클래스다이어그램에 적용 할 수 있는 소프트웨어 측정 척도가 필요하다.

참 고 문 헌

[1] Booch, G., Rumbaugh, J., and Jacobson, I. 1999, "The Unified Modeling Language User Guide", Addison Wesley Publication Company.

[2] E. J. Weyuker. "Evaluating software complexity measures." IEEE Tran. on SE, Vol.14, No.9, pp.1357-1365 1988.

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

[4] F. J. Budinsky, M. A. Finnie, J. M. Vissides, P. S. Yu, "Automated code generation from design patterns," Object technology, IBM Systems Journal Vol.35, No.2, 1996.

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

[6] http : //jerry.cs.uiuc.edu/~plop/plop2000/international Conference Pattern", May, 2000.

[7] Jagdisk Bansiya, "Automating Design-Pattern Identification," Dr Dobb's Journal June, 1995.

[8] J. M. Sagawa, "Repository Manager Technology," IBMSystem Journal, Vol.29, No.2, 1990, pp..209-227.

[9] Martin Fowler, "UML Distilled," Addison-Wesley, 1997.

[10] Nicolas Anquetil and timothy c. Lethbridge, "Experiments with Clustering as a software Re -modularization Method," Proceedings of the 6th Working Conference on Reverse Engineering, pp.235-255, 1999.

[11] OMG, "Common Object Request Broker Architecture : Core Specification," v3.0.2, Dec., 2002.

[12] Paolo Tonella and Giulio Antoniol, "Object-oriented Design Pattern Inference," Proceedings of the IEEE International Conference on Software Maintenance, pp.230-238, 1999.

[13] Udo Hahn, Mattthias Jarke, Thomas Rose, "Teamwork Support in a Knowledge-Based Information Systems Environment," IEEE Transactions on Software Engineering, pp.467-481, 1991.

[14] UML1.3 Specification. OMG Documents ad990608 -ad990609.

[15] 이승형, 송영재, "UML클래스의 효율적인 디자인패턴 추출 시스템에 관한 연구", 한국정보처리학회, 추계학술 논문집, 2000.

[16] 선수균, "효율적인 디자인 패턴 추출 및 분류를 위한 인덱스 관계정보에 관한 연구", 경희대학교 대학원 박사논문, 2002, 경희대학교.

[17] 선수균, 송영재, "통합 객체 관리 모델을 이용한 효율적인 객체관리저장소설계", 정보처리학회논문지D, 제8-D권 제2호, pp.166-174, 2001.

선 수 균

e-mail : sksun@tongwon.ac.kr

1988년 경희대학교 전자계산공학과(공학사)

1994년 경희대학교 전자계산학과(공학석사)

2002년 경희대학교 전자계산공학과 (공학박사)

1988년~1996년 경희대학교 전자계산소 프로그래머 근무

1996년~1997년 세경대학 전자계산과 교수

1997년~현재 동원대학 e-business과 교수

관심분야 : 소프트웨어 공학, 전자상거래, e-비즈니스, S/W 재사용