클러스터링 알고리즘을 이용한 효율적인 디자인 패턴 분류에 관한 연구

최영건ᵠ, 김귀정, 송영재
경희대학교 전자계산공학과
e-mail : ykchoi@case.kyunghee.ac.kr

A Study on Efficient Design Pattern Classification Using Clustering Algorithms

Young-Keon Choiᵠ, Gui-Joung Kim , Young-Jae Song
Dept. of Computer Engineering, Kyung-Hee University

요 약

디자인 패턴은 시스템 설계시의 일반적인 문제들을 해결하기 위해 클래스를 조직화한 것이다. 본 연구는 디자인 패턴을 클러스터링 하기 위하여 클래스의 관계를 나타내는 구조를 이용한 패턴 클러스터링 알고리즘을 제안하였다. 제안한 디자인 패턴의 클러스터링은 패턴 저장 시 패턴 클러스터링에 의해 분류하고 링크정보를 이용하여 저장하므로 저장소를 효율적으로 관리 할 수 있으며 또한 재공학에 의해 추출된 클래스로 표현된 전체 시스템 구성도로부터 패턴의 사용정보를 추출하므로 시스템의 재설계시에 도움을 줄 수 있도록 하였다.

1. 서론

객체지향 개발 방법론을 활용하기 위해서는 실제적인 개념과 규격화된 패턴, 상황에 따른 올바른 객체의 선택과 적용이 필요한데 이러한 객체지향 설계 개념을 이론적으로 제시하는 것으로 그치지 않고 프로그래밍에 적용 가능하도록 구체적으로 규격화 시킨 것이 디자인 패턴이다.[1]. 디자인 패턴은 객체지향 방법론의 가장 큰 장점인 재사용성과 모듈성을 극대화 시킨 실제 구현 과정에서의 해결책이며 현재 미국내의 설계 패턴 컨퍼런스인 PLoP(Pattern Languages of Programs)와 유럽의 설계 패턴 컨퍼런스인 EuroPLoP를 통해서 공식적으로 발표되어 알려져 있는 디자인 패턴은 수 백 가지에 이른다. 이렇게 증가하고 있는 패턴의 재사용성을 향상시키기 위해서 부품의 효율적인 관리가 필수적이다.

기존의 클러스터링은 클래스간의 클러스터링이나 클래스내의 클러스터링이 주를 이루었다. 따라서 기존의 클러스터링 방법은 클래스나 모듈의 응집도와 결합도를 이용하여 클러스터링 하였다.[2] 그러나 클래스 간의 관계와 구조에 중점을 준 디자인 패턴을 이러한 응집도나 결합도를 이용한 기존의 클러스터링 방법으로 클러스터링 하는 것이 효과적일 수 없다. 따라서 본 연구는 클래스 간의 관계로 이루어진 디자인 패턴을 클러스터링하기 위하여 패턴 클러스터링 알고리즘을 제안하였다.

패턴 클러스터링 과정은 패턴들이 가지고 있는 특성을 위해 2 단계의 분류과정을 거치는데 1 단계 클러스터링 과정에서는 패턴의 기능에 따른 분류를 하며 2 단계 클러스터링 과정에서는 패턴의 구조를 가지고 분류를 하게 된다. 패턴 구조에 의한 분류 시 패턴 알고리즘을 사용하여 분류한다.

제안한 디자인 패턴의 클러스터링은 패턴 저장 시 패턴 클러스터링에 의해 분류되어 패턴의 링크정보를 이용하여 저장하므로 저장소(Repository)를 효율적으로 관리 할 수 있으며 또한 시스템의 리엔지니어링시 기존의 역공학 기술을 사용하여 소스코드로부터 추출한 클래스로 표현된 전체 시스템 구성도로부터 패턴의 사용정보를 추출하므로 설계정보를 이용하여 시스템의 재설계시에 도움을 줄 수 있도록 한다.

2. 관련 연구

2.1 Gamma 의 패턴 분류

디자인 패턴을 Gamma 는 우선적으로 패턴이 하는 역할에 따라 생성패턴, 구조패턴, 행위패턴으로 구분하였다. 생성패턴은 객체의 생성방식을 결정하는 포괄적인 솔루션을 제공하는 패턴으로 클래스 정의와 객체 생성 방식을 구조화, 캡슐화 하는 방법을 제시한다. 구조패턴은 클래스와 객체가 보다 대규모 구조로 구성되는 방법에 대한 해결책을 제시하는 패턴으로 다른 기능을 가진 객체가 협력을 통해 어떤 역할을 수행할 때 객체를 조직화 시키는 일반적인 방식을 제시한다. 마지막으로 행위 패턴은 객체의 행위를 조직화, 관리, 연합하는데 사용되는 패턴으로 객체간의 기능을 배분하는 일과 같은 알고리즘 수행에 주로 이용된다.

또한 <표 1>에서 보는 바와 같이 Gamma 는 패턴의 사용목적에 따라 24 개의 기본적인 패턴을 정의 내렸다.[3]

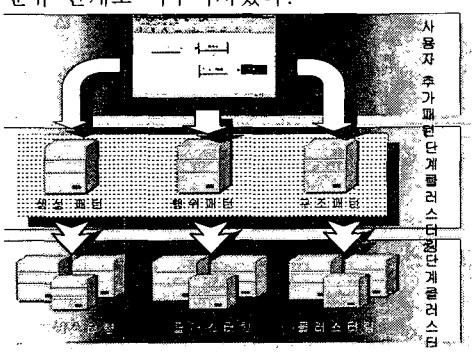
<표 1> Gamma의 디자인 패턴 분류

Purpose		
Creational	Structural	Behavioral
Abstract Factory	Adapter	Chain of Responsibility
Builder	Bridge	Command
Factory Method	Composite	Interpreter
Prototype	Decorator	Iterator
Singleton	Façade	Mediator
	Flyweight	Menento
	Froxy	Obsever
		State
		Strategy
		Template Method
		Visitor

3. 패턴 클러스터링

3.1 패턴 클러스터링 과정

패턴 클러스터링 과정은 (그림 1)에서와 같이 크게 2 단계로 나누어진다. 첫 번째 단계에서는 패턴이 어떠한 역할을 수행하는가의 역할에 따른 기능적 분류를 하고 두 번째 단계에서는 패턴들이 가지고 있는 클래스들 간의 관계를 이용하여 클러스터링 하는 구조적 분류 단계로 이루어져있다.



(그림 1) 패턴 클러스터링 과정

사용자는 UML 의 클래스 다이어그램을 사용하여 사용자 패턴을 형성하면 클러스터링 과정을 통하여 패턴 라이브러리에 저장하게 된다. 그러나 패턴 구조를 이용한 클러스터링 방법은 단순히 패턴의 구조가 일치하는 부분이 있다고 해서 관련이 있는 패턴이라고 정의 내리기 어렵다. 따라서 패턴구조를 이용한 클러스터링 하기 전에 전 단계로 패턴을 기능적으로 분류를 하는 과정을 거친다. 기능적 분류 단계에서는 패턴이 하는 역할에 따라 패턴을 3 가지로 분류한다. 객체의 생성 방식을 결정하는 포괄적인 방법을 제공하는 패턴이면 생성 패턴으로 분류하고, 객체를 조직화 시키는 일반적인 방식을 제시하는 패턴이면 구조 패턴으로 분류하며, 그리고 객체간의 기능을 분할하는 역할을 수행하는 패턴이면 행위패턴으로 분류하여 인위적으로 선택을 한다.

기능적으로 분류된 것을 기준으로 구조적 분류단계에서는 패턴의 구조적인 형태를 이용하여 클러스터링 하도록 하였다. 구조적 분류 단계에서의 클러스터링 방법으로는 3 가지로 분류된 Gamma 의 기본 패턴중의 하나와 사용자가 추가한 패턴이 구조가 일치하는 패턴을 같은 분류로 클러스터링 하도록 클러스터링 알고리즘을 사용하였다

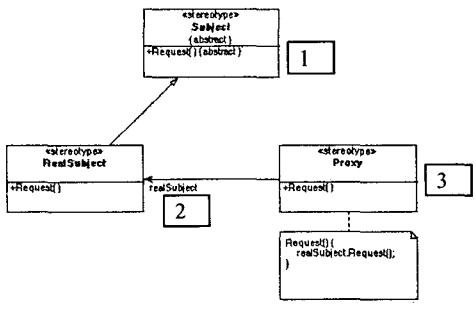
3.2 패턴 클러스터링 알고리즘

패턴 클러스터링 알고리즘은 기준으로 정한 패턴중에서 사용자 추가 패턴의 구조와 일치하는 구조가 있는 경우에 같은 그룹으로 클러스터링 되도록 하는 알고리즘이다.

패턴의 구조는 UML 의 클래스 다이어그램에서 클래스와 클래스의 관계로 나타내어진다. 패턴의 구조를 비교하기 위해 클래스와 클래스간의 관계를 하나의 순서쌍으로 하여 하나의 패턴을 순서쌍의 그룹으로 변환한다.

$$P=\{R_k(i,j)\mid(i,j)\in R,\ i\in C,\ j\in C,\ 1\le k\le n\} \qquad (1)$$

P 은 패턴의 순서쌍을 나타낸다. R 은 클래스의 관계를 나타내고 C는 클래스를 나타낸다.



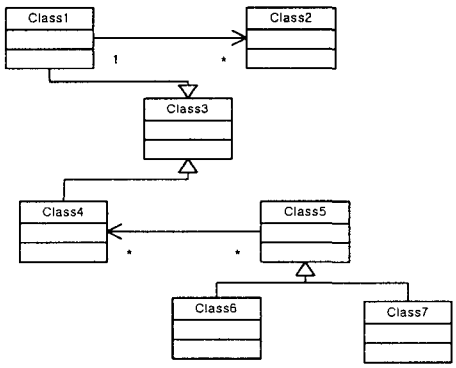
(그림 2) 프록시 패턴

(그림 2)은 프록시 패턴을 나타낸다. 이 프록시 패턴을 순서쌍으로 변환하면 $P=\{G(2,1),S(3,2)\}$으로 나타내어진다. 각 하나의 순서쌍에서 앞부분의 영어는 클래스 다이어그램의 관계를 나타낸 것이고 숫자는 클래스를 나타낸다.

<표 2> 클래스다이어그램에서의 관계

기호	관계	표시
——→	Association	S
——▷	Generalization	G
——◇	Aggregation	E
- - - - - →	Dependency	D
——◆	Composit	C
- - - - - ▷	Realization	R

<표 2>은 클래스다이어그램에서의 관계를 나타내는 기호와 순서쌍에서의 약자 표시를 표로 나타내었다. 순서쌍의 뒤 부분은 관계를 가지는 두 클래스를 나타내었다. 또한 사용자 추가 패턴이 (그림 3)와 같다면 사용자 패턴을 순서쌍으로 변환하면 *P=S(1,2), G(1,3), G(4,3), S(5,4), G(6,5), G(7,5)*으로 변환할 수 있다. 이 사용자 추가 패턴은 프록시 패턴이 확장되어진 상태를 나타내고 있다. 이 패턴 안에서 프록시 패턴을 찾는 알고리즘은 두 패턴을 나타내는 순서쌍의 비교로써 이루어진다



(그림 3) 사용자 추가 패턴

두 패턴을 비교하기 위해 먼저 순서쌍의 숫자로 명명된 클래스를 변환해 주는 과정을 거친다. 변환을 하는 이유는 클래스의 숫자는 임의적인 숫자를 사용했기 때문에 클래스가 명명되는 클래스의 순서와 상관없이 순서쌍을 비교하기 위한 과정이다

Initial Setp $P_1 = \{R_1(i_1, j_1) \mid a \to i_1, b \to j_1\}$

Setp $2 \le k \le n$

$$P_k = \{R_k(i_k, j_k) \mid i_k = i_1 \text{이면}\ ,\ i_k \to a, j_i \to x$$
$$j_k = i_1 \text{이면}\ ,\ i_k \to x, j_i \to a$$
$$j_k = j_1 \text{이면}\ ,\ i_k \to x, j_i \to b$$
$$i_k = j_1 \text{이면}\ ,\ i_k \to b, j_i \to x\}$$

(그림 4) 패턴 순서쌍 변환 알고리즘

(그림 5)에서 비교 패턴 알고리즘에서 P_f 은 변환후의 기본패턴을 나타내며 P_a 은 변환후의 추가패턴을 나타낸다. 사용자가 추가한 패턴이 기본패턴과 비교하기 위해서 우선 기본패턴을 변환 알고리즘을 사용하여 변환한 다음 추가 패턴 또한 변환 과정을 거쳐 두 패턴이 같은 구조를 가지고 있는지 비교를 한다. 이 때 추가 패턴의 관계를 나타내는 여러 개의 순서쌍을 중에 기본패턴과 관련이 있는 패턴을 찾아내기 위해 추가 패턴의 관계를 나타내는 순서쌍을 차례대로 변환과정을 거친다. 사용자 추가 패턴의 관계를 나타내는 순서쌍을 하나씩 변환 과정을 거쳐서 아래 조건 식(2)에 만족하면 추가패턴은 P_a 은 기본 패턴 P_f 에 클러스터링 된다. 모든 순서쌍을 변환하였는데도 조건 식(2)에 만족하지 않으면 다른 기본패턴과 비교하는 과정을 거친다.

$$P_f \subset P_a \qquad (2)$$

$P_f = \{(P_{f1}, P_{f2}, P_{f3}, \ldots P_{fk}) \mid$ Transforma tion algorithm of Founddatio n Pattern$\}$

for($i = 0,\ i < n,\ i++$)

{

$P_a = \{(P_{a1}, P_{a2}, P_{a3}, \ldots P_{ak}) \mid$ Transforma tion algorithm of Addition Pattern$\}$

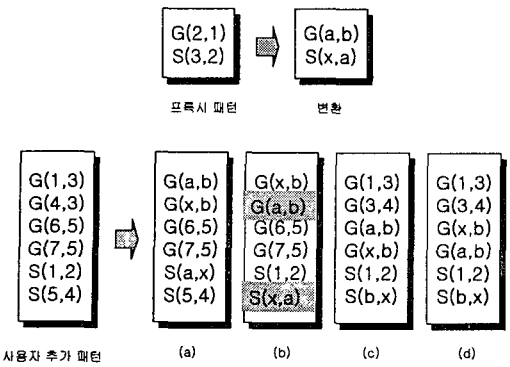
$P_f \subset P_a$: Break Addtion()

$P_{a1} \to P_{an}, P_{ak} \to P_{a(k-1)}$

}

(그림 5) 패턴 비교 알고리즘

(그림 6)은 사용자 추가 패턴과 프록시 패턴과의 비교 과정을 구체적으로 도식화 한 것이다. 프록시 패턴은 하나의 연관관계와 상속관계로 이루어져있다. 사용자 패턴 중에 같은 구조를 가지는 연관관계와 상속관계를 찾아내기 위해서 프록시 패턴을 나타내는 G(1,3)을 G(a,b)로 변환하여 준다. 그리고 다른 순서쌍에도 1 은 a 로 3 은 b 로 변환을 모든 해준다. 변환 후의 프록시 패턴의 순서쌍은 G(a,b), S(x,b)로 나타내어 질 수 있다. 사용자 패턴도 변환 과정을 거쳐 비교를 하는데 사용자 패턴은 4 개의 상속관계를 가진다. 이것 중에 실제적으로 프록시 패턴의 구조를 나타내는 상속관계를 찾기 위해 4 개의 상속관계 중에 하나를 변화과정을 거쳐 프록시 패턴의 변환된 순서쌍과 비교를 한다. 이 변환과정을 거쳐 G(a,b),S(x,b)구조를 가지는 순서쌍을 포함하고 있으면 사용자 패턴은 프록시 패턴의 구조를 가지는 그룹으로 클러스터링 되어진다. 4 개의 상속관계를 나타내는 순서쌍을 비교하여 G(a,b),S(x,b)구조를 가지는 패턴이 나타나지 않으면 프록시 패턴과 사용자 추가 패턴은 같은 구조를 가지는 패턴이라고 할 수 없고 다른 패턴과의 비교를 하는 과정을 반복한다.

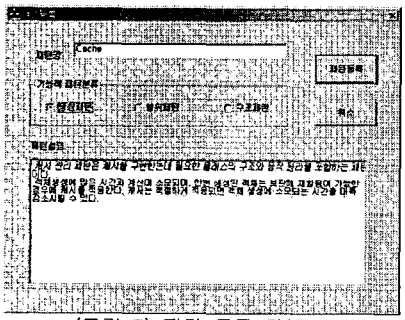


(그림 6) 패턴 비교 과정

4. 패턴 등록 및 관리

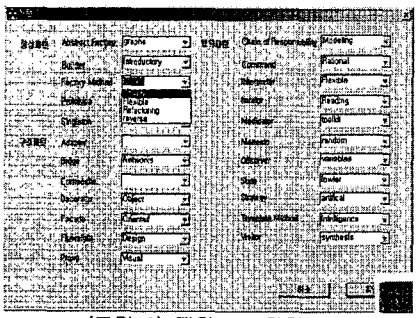
사용자는 패턴을 작성하여 패턴 라이브러리에 등록할 수 있으며 또한 패턴 DB 로부터 패턴들을 검색할 수 있다.

(그림 7)의 패턴 등록은 사용자가 패턴을 등록하는 곳으로써 패턴을 등록하면 클러스터링에 의해 분류되어 저장된다.



(그림 7) 패턴 등록 화면

(그림 8)의 패턴 DB 은 패턴을 Gamma 의 24 개의 패턴을 기준으로 분류되어 있다. 패턴들 중에 하나를 선택하면 패턴보기 대화상자에서 패턴의 구조와 패턴에 대한 간단한 설명을 볼 수 있다.



(그림 8) 패턴 DB 화면

5. 결론

본 논문은 패턴의 효율적인 관리를 위해서 패턴의 구조를 이용한 패턴 클러스터링 알고리즘을 제안하였다. 패턴 클러스터링 알고리즘은 패턴 분류를 위해 패턴이 가지고 있는 클래스의 구조를 이용하여 분류하는 방법으로서 분류를 위한 기본패턴을 정하고 사용자가 새로운 패턴을 추가하려고 할 때 패턴의 클래스와 클래스의 관계를 하나의 순서쌍으로 나타내어 기본패턴과 비교하여 일치하는 부분이 있을 경우 한 그룹으로 클러스터링 되도록 한 것이다.

패턴 클러스터링에 의한 분류는 패턴 저장시 링크 정보를 이용하기 때문에 정보저장소를 효율적으로 관리할 수 있으며 또한 시스템 리엔지리어링 시 클래스로 표현된 전체 시스템 구성도로부터 디자인 패턴의 사용정보를 쉽게 추출하므로 시스템의 재 설계에 도움을 줄 수 있다.

향후 연구 방향으로 사용자 추가 패턴중에 2 개이상의 기본패턴을 포함하고 있을 때의 클러스터링 방법과 패턴 분류시 기존의 기본패턴인 Gamma 의 기본패턴에 분류되지 않았을 경우 기본패턴을 확장하여 사용자가 기본패턴을 추가할수 있도록 하는 방법을 연구하고자 한다.

참고문헌

[1] http://www.omg.org

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

[3] E.Gamma, R.Helm, R.Johnson, and J.Vlissides, "Design Pattern : Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995.

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

[5] R.Prieto-Diaz and P.Freeman, .Classifying Software for Reusability, IEEE Software, Vol.4, No.1, pp.6-16, Jan. 1987.

[6] William B.Frakes and Ricardo Baeza-Yates, "Information Retrieval," Prentice Hall, pp.419-442, 1992.

[7] R.Schauer and R. Keller. Pattern visualization for software comprehension, pages 4.12, 1998.