레거시 Java Code로부터 디자인 패턴 추출을 위한 AOL 기반 프로세스

이창목[0], 이정열*, 김정옥**, 유철중**, 장옥배**
*정인대학 사무정보계열 **전북대학교 컴퓨터과학과
{cmlee[0], jokim, jylee}@cs.chonbuk.ac.kr, {cjyoo, okjang}@moak.chonbuk.ac.kr

AOL-Based Process for Design Patterns Extraction from Legacy Java Code

Chang-Mog Lee[0], Jeong-Yeal Lee*, Jeong-Ok Kim**, Cheol-Jung Yoo**, Ok-Bae Chang**
Dept. of Computer Science, Chonbuk National University

요 약

객체지향 디자인 패턴은 아주 작은 재사용 구조로서 객체지향 방법론의 가장 큰 장점인 재사용성과 모듈성을 극대화시켜 실제 구현 과정에서 해결 방안으로 제시 가능한 것으로 이를 적용하면 시스템 개발은 물론 유지 보수에도 큰 효과가 있다. 순공학에서는 디자인 패턴을 이용하는 이점이 명확하지만, 소프트웨어 인공물들 즉, 디자인이나 코드 등에서 디자인 패턴의 사례를 발견하기 위해 사용하는 역공학 기술은 프로그램의 이해, 디자인을 코드로 변환하는 변환성, 코드의 질적 평가 등의 핵심 영역에서 유용하다. 본 논문은 Java 소스 코드를 AOL(Abstract Object Language)이라는 추상객체언어를 이용하여 클래스 특성 추출기 및 패턴 식별기라는 단계를 통해 구조적 디자인 패턴을 추출하는 프로세스에 대하여 기술한다.

1. 서 론

객체지향 디자인 패턴이 주목받는 이유는 기존의 환경에서 일반적인 디자인 문제에 대하여 잘 알려진 해결책을 제시하기 때문이다[1]. 가장 잘 알려진 객체지향 디자인 패턴 관련 문서는 Gamma 등이 저술한 문헌[2]이다. 디자인 패턴 추출은 판에 박힌 매칭문제로 여겨졌었고, Wills에 의해 확인된 다양한 소스가 논의되었다.[3]. 순공학은 디자인 패턴을 이용하는데 따르는 이점이 명확하다. 그러나 프로그램을 이해하는 것과 유지보수할 수 있는 관점에서 패턴은 패턴내의 각 클래스에 대한 지식, 패턴 구성요소들과 또는 시스템의 나머지 부분간의 관계에 대한 중요한 정보를 제공한다. 그러므로 유지보수에서 디자인 패턴 사례의 식별은 소프트웨어 인공물의 통찰을 제공하고, 변경될 곳, 재사용될 곳, 또는 확장될 곳 등이 예상되는 지점을 나타낸다. 따라서, 디자인 패턴을 이용하여 디자인된 시스템은 모듈성, 관계의 분리, 재사용성과 확장의 용이성 등과 같은 특성을 구분하기가 용이하다.

본 논문에서는 여러 곳에서 표현되고있는 구조적 디자인 패턴을 추출하기 위한 추출 프로세스를 제시한다. 추출 프로세스는 소프트웨어 산물중의 하나인 소스코드로부터 디자인 패턴 후보들을 추출하기 위해 추상객체언어(이하 AOL : Abstract Object Language) [4]와 클래스 특성추출기, 그리고 디자인 패턴 식별기 등의 프로세스를 두었다. 따라서 본 논문에서 제시하는 객체지향 디자인 패턴 추출 프로세스는 Java 소스 코드 및 Java IDE 툴로부터 생성된 소스 코드는 AOL이라고 하는 중간적 언어로 표현된다. AOL로의 변환 이유는 특정 프로그래밍 언어에 독립성을 보장하기 위해서이다. 이렇게 AOL로 모델된 정보를 이용하여 패턴 인식 단계에서 AOL을 통한 클래스 정보를 분석하고 클래스 특성 및 디자인패턴 식별기를 통해 디자인 패턴을 분류해 낸다. 본 논문의 구성은 다음과 같다. 먼저 2장 관련연구에서는 디자인 패턴 추출을 위한 기존의 연구에 대해 알아보고 3장에서는 디자인 패턴 추출 프로세스를 마지막 4장에서는 결론 및 향후 연구과제를 기술한다.

2. 관련연구

디자인에서 패턴의 존재는 그에 상응하는 소스 코드로 반영될 수 있다[5]. 디자인 패턴은 객체지향 패러다임에서는 새로운 분야이고, 클래스 명세화 방법과 역공학에서도 부분적으로 연구되고 있다. Kramer와 Prechelt는 Pat이라 불리는 시스템 접근법을 제안하고 개발하였다[6]. Pat은 구조적 정보로부터 구조적 디자인 패턴의 사례를 유추하는 것이다. 디자인 정보를 추출하기 위해 CASE 도구 저장소의 역공학적 능력에 의존적이고, 이를 표현하기 위해 Prolog fact를 사용하였다. 산업체와 공개영역(public domain) 소프트웨어 상에서의 실험적 결과를 제공되나 다른 접근법과 직접적인 비교는 불가능하다. 더욱이 Pat은 사례 지향적이므로, 코드상에서 실제 존재하지 않는 디자인 패턴도 검색한다.

Keller는 시스템의 역공학적 소스코드가 주어진다면, 일반적 혹은 특별한 디자인 패턴 모두를 추출하고 시각화하기 위한 접근법과 프로토타입을 개발하였다[7]. 이 도구는 자동, 수동 반자동 디자인 패턴 추출을 지원한다.

Shulli은 기존의 객체지향 소프트웨어 시스템에서 관례와 도메인 종속적 디자인 패턴을 발견하는 것을 돕는 귀납적 방법을 개발하였다. 그러나 이 방법은 수동적으로 수행된다.

Antoniol and Tonella는 코드나 디자인으로부터 직접적으로 반복적인 디자인 패턴의 유추에 대한 접근법을 제안하였다[8]. 패턴 라이브러리 어떠한 것이라도 이용

가능성에 대해 어떠한 가정도 없었고 개념 분석 알고리즘만이 패턴을 유추하는데 적용되었다. 이 외에도 기존의 여러 사람들이 시도한 디자인 패턴 추출을 위한 도구 및 방법들이 있으나 본 논문에서는 클래스 객체를 명세하기 용이한 AOL을 이용하여 구조적 디자인 패턴을 추출하는데 주안을 두었다.

3. 디자인 패턴 추출 프로세스

이번 장에서는 객체지향 디자인 패턴의 추출을 위한 추출 프로세스에 대하여 기술한다.

객체지향 디자인 패턴 추출 프로세스중 Java 소스 코드 또는 Java IDE 툴로부터 생성된 소스 코드를 AOL이라는 추상객체언어로 변환을 하는 단계가 있다. 이는 프로그래밍 언어와 특정 IDE 툴로부터 생성된 코드로부터 중간적 표현으로 나타내기 위해서이다. 이렇게되면 클래스들간의 구조적 관계를 표현하는데 독립성을 보장하기 때문이다. AOL은 기본적으로 클래스 다이어그램에 초점이 맞춰있다. 이는 클래스들간의 관계뿐만 아니라 클래스, 메소드, 애트리뷰트 등을 모델하는데 용이하기 때문이다. BNF 문법을 확장한 AOL 문법은 그림 1과 같다.

```
Conventions: the metacharacters follow the extended BNF notation
   {}   means zero or more times
   []   means an optional element, so 0 or 1 time
   ""   means a terminal symbol
   |    means the boolean symbol OR
   ()   means a block of elements, useful to group them

AOL_design_description ::= list_AOL_declarations
list_AOL_declarations ::= {AOL_decl ";"}
AOL_decl ::= class | association | generalization | aggregation

/* CLASSES */
class ::= CLASS class_name scope
                [ATTRIBUTES attribute_list]
                [OPERATIONS operation_list]
                                ;;;;;
                                ;;;;;
/* RELATIONS */
association ::= RELATION [relation_name] ROLES roles
                [ATTRIBUTES attribute_list]
                [IS_A_CLASS assoc_class_id_ref]
                                ;;;;;
                                ;;;;;
```

그림 1 BNF 문법을 확장한 AOL 문법의 예

그림 2는 본 논문에서 제시하는 디자인 패턴 추출 프로세스를 보여준다. 각 단계별 프로세스의 세부사항은 다음과 같다.

3.1 Java 소스코드 or Java IDE 생성 코드

본 논문에서 제시하는 객체지향 디자인 패턴 추출 프로세스는 Java 소스 코드나 Java 언어를 지원하는 통합 개발도구 즉, IDE 툴로 생성된 소스 코드로부터 디자인 패턴을 추출하는 것이다.

3.2 Code -> AOL 번역기

이번 단계에서는 Java 소스 코드나 Java IDE 툴로 생

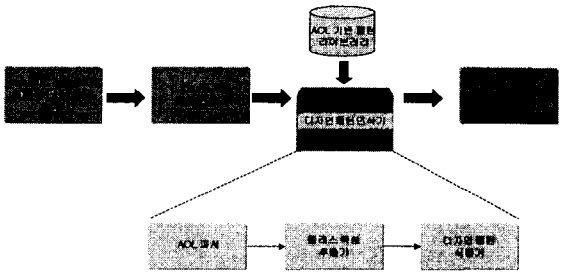

그림 2 디자인 패턴 추출 프로세스

성된 소스 코드를 AOL이라는 중간적 언어로 변환한다. IDE 툴로부터 생성된 소스 코드는 비록 같은 Java 라는 언어를 사용한다 할지라도 또 다른 Java IDE 툴을 사용하여 생성한 소스코드와 호환이 되지 않는데 이는 각 IDE 툴마다의 특성이 다르기 때문이다. 따라서 Java 원시 코드를 공통적으로 표현할 수 있는 중간적 언어가 필요한데 본 논문에서는 AOL이라는 언어를 사용하였다. AOL은 기본적으로 클래스 다이어그램에 초점이 맞춰있다.

즉, 클래스, 메소드, 애트리뷰트 등이 모델된다. 현재 AOL 버전은 클래스 다이어그램과 연관된 UML을 지원한다. 따라서 AOL 디자인 표현 언어는 어떠한 특정 프로그래밍이나 또 다른 디자인 표현 언어로부터 독립성을 보장한다. 그림 3은 구조적 패턴중 어댑터 패턴과 일치하는 AOL 객체 모델 기술을 보여준다.

```
CLASS    TargetIF
          OPERATIONS
                    PUBLIC interfaceMethod();
CLASS    Adapter
          OPERATIONS
                    PUBLIC interfaceMethod();
CLASS    Adaptee
          OPERATIONS
                    PUBLIC otherMethod();
GENERALIZATION    TargetIF
         SUBCLASS Adapter;
RELATION    Refers
         ROLES
                  CLASS    Adapter     MULT One,
                  CLASS    Adaptee     MULT One,
```

그림 3 AOL 기술된 Adapter 패턴의 표현 예

3.3 AOL기반 패턴 라이브러리

본 논문에서 제시하는 디자인 패턴 추출 프로세스는 디자인 패턴 라이브러리와 밀접한 관련이 있다. 본 논문에서는 구조적 디자인 패턴의 범위로만 제약하여 이러한 정보가 AOL로 이미 라이브러리화 되어있다. 기존의 일반적인 매칭방법에서는 라이브러리의 역할은 디자인 패턴에 상응하는 사본의 정보를 가지고 있었다. 따라서 소스 코드들로부터 추출된 패턴 정보가 라이브러리에 저장되어있는 정보와 정확히 일치할 경우에만 패턴정보를 인식하는 방식이었으므로 인식률이 현저히 낮아진다. 그러나 AOL 기반 패턴 라이브러리를 이용하여 소스 코드로

부터 AOL로 변환한 정보와 매칭시키는 방법은 구조적인 정보를 표현하기에 적합한 AOL의 특성상 AOL 파서에 의해 구문을 분석하기 용이하기 때문에 패턴 매칭이 간단하다.

3.4 디자인 패턴 인식기

디자인 패턴 인식기는 그림 2에서도 나타나있듯이 AOL 파서, 클래스 특성 추출기, 디자인 패턴 식별기 등의 세 부분으로 나뉘어져 있다.

- AOL 파서 : 3.2절에서 소스 코드 정보를 AOL로 변환하였던 정보를 객체지향 디자인 패턴을 추출하기 위한 첫 번째 작업으로 AOL 파서를 이용하여 구문 분석을 하는 단계이다. AOL 파서를 통해 식별자에 대한 참조를 인식하고 일부 간단한 동일성 체크를 수행하는 단계이다.

- 클래스 특성 추출기 : 이 단계는 AOL 파서에 의해 파싱된 정보로부터 클래스의 특성을 추출하는 단계이다. 즉, 검색 공간의 부피를 줄이기 위해서 필수적인 수단이다. 클래스 특성 추출기를 통해서 복잡한 관계들 즉, 집합(aggregation), 연관(association), 상속(inheritance), 그리고 메소드들에 관한 정보가 추출된다. 이러한 정보를 추출해냄으로써 불필요하고 복잡한 관계를 줄이기 위함이다. 그림 4는 각 클래스에 대한 좀 더 상세한 측정기준을 보여주고 있다.

- public, private, protected 속성
- public, private, protected 오퍼레이션
- 직속 상위 또는 하위 클래스
- 클래스에 포함되어 있는 연관과 집합

그림 4 각 클래스의 정보 추출 기준

- 디자인 패턴 식별기 : 디자인 패턴 사례를 발견한다는 것은 정확한 패턴 특성 즉, 관계, 행위, 의도 등을 보이는 클래스 집합을 식별해야만 한다. 때문에 전 단계에서 클래스들의 특징을 추출한 것이다. 이 단계에서는 이러한 특징을 가지고 좀 더 정확한 디자인 패턴을 감별하기 위한 제약사항을 두어서 추출된 클래스 정보에 적용하는 단계이다. 그림 5는 어댑터 패턴을 추출하기 위하여 추출범위를 좁혀 가는 과정을 보여준다.

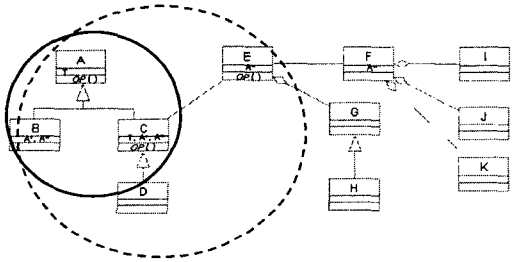

그림 5 어댑터 패턴의 추출 예

만약 그림 5와 같은 정보를 가진 정보가 클래스 특성 추출기를 통해 검출되었다면 B클래스 같은 경우에는 어댑터의 역할을 할 수 없다. 그 이유는 상위 클래스의 오퍼레이션을 가지고 있지 않기 때문이다. 따라서 어댑터 패턴을 만족시키는 후보 클래스 집합은 <A,C,B>, <A,C,E>가 된다. 그러나 여기서 <A,C,B> 클래스 집합은 B클래스에 의해 C 클래스의 오퍼레이션이 호출이 되어야 어댑터패턴의 기능을 만족하는데 그림에서는 호출을 하고있지 않다. 그러므로 최종적인 어댑터 패턴을 나타내는 클래스는 <A,C,E> 클래스 집합이다.

4. 결론 및 향후 연구과제

본 논문에서는 객체지향 디자인 패턴을 추출하기 위한 각 단계별 프로세스에 대하여 알아보았다. 객체지향 디자인 패턴을 추출하기 위하여 언어 독립적인 AOL이라는 중간적 언어를 이용하였다. 패턴을 추출하는 예로써 어댑터 패턴의 추출 과정을 기술하였다. 그 이유는, 어댑터 패턴은 구조적 패턴의 분류에 속하며 구조적 패턴은 다른 패턴부류와는 달리 새로운 기능을 구현하기 위해 객체를 구성하는 방식 자체에 초점이 맞춰져있기 때문이다. 즉, 정적인 면이 강하다. 따라서 AOL이라는 추상객체언어로 변환하기가 용이하다. 이러한 정보를 바탕으로 클래스의 특성과 디자인 패턴 식별기 단계를 적용하여 디자인 패턴을 추출한다.

향후 연구과제로는 추출된 디자인 패턴 정보를 한 눈에 보여줄 수 있는 사용자 인터페이스 프로토타입 개발과 본 논문에서 기술한 디자인 패턴 추출 프로세스를 이용하여 실제 Java 코드로 개발된 비즈니스 도메인의 패턴 추출 사례를 평가하는 것이다. 이는 경험적인 실제 연구가 필요하기 때문에 향후 연구과제로 남겨 놓는다.

참고문헌

[1] Mark Grand, "Patterns in Java: a Catalog of Reusable Design Patterns Illustrated with UML", John Wiley & Sons, 1998.

[2] Gamma, E., Helm, R., Vlissides, J., "Design Patterns : Elements of Reusable Object-Oriented Software", Addison Wesley, 1995..

[3] Wills, L., "Automated Program Recognition by Graph Parsing", Ph.D. Dissertation, MIT, 1992.

[4] G. Antoniol, G. et al., "Object-Oriented Design Patterns Recovery", Journal of Systems and Software, Volume 59, Issue 2, pp. 181-196, 15 November 2001.

[5] Patterns Catalog from http://hillside.net/patterns

[6] Kramer, C., Prechelt, L., "Design Recovery by Automated Search for Structural Design Patterns in Object-Oriented Software", International Workshop on Program Comprehension, pp. 208 ~ 215, 1996.

[7] Keller, R, Schauer, R, Robitaille, S, Page, P., "Pattern-Based Reverse-Engineering of Designs Components", Proceedings of the International Conference on Software Engineering, Los Angeles, pp. 226 ~ 235, 1999.

[8] Antoniol, G., Tonella, P., "Object-Oriented Design Pattern Inference", Proceedings of the International Conference on Software Maintenance, pp. 230 ~ 238, 1999.