제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

디자인 패턴을 이용한 리팩토링 사례 연구

유명환, 정인정
고려대학교 전산학과
e-mail:{myong, chung}@korea.ac.kr

A Case Study on Refactoring using Design Patterns

Myong-Hwan Yoo, In-Jeong Chung
Dept of Computer Science, Korea University

요 약

소프트웨어가 대형화되고 복잡해짐에 따라 개발 과정에서 많은 요구사항이 발생되고 변화가 일어난다. 이러한 상황에서 소프트웨어는 유연성을 지니고 요구사항을 충족시켜야 한다. 그러나 추후에 발생할 요구사항을 모두 예견하고 충족시키는 것은 불가능하다. 이를 위해 기존 소프트웨어를 재사용하고 유연성을 높일 수 있도록 소프트웨어의 리팩토링(refactoring)을 해주어야 한다. 리팩토링은 행동(behavior) 변화가 없이 보호하고 소프트웨어를 효율적이고 유지보수가 쉽도록 전환하는 과정이다. 리팩토링에서 추상 슈퍼클래스(abstract superclass)의 생성과 디자인 패턴을 이용함으로써 소프트웨어의 재사용성을 높여주고 유연성을 제공해 줄 수 있다. 본 논문에서는 사례 연구로써 표준프로파일 제공시스템의 통합검색부분에 추상 슈퍼클래스의 추출 및 생성, 디자인 패턴, 기존 프로그램의 합성 등의 방법을 적용하여 리팩토링을 하도록 한다.

1. 서론

소프트웨어는 계속적으로 새로운 요구사항이 추가되고 기존의 요구사항이 변한다. 변화된 요구사항을 충족시키기 위해서 새로 프로그램을 만들지 않고 기존의 유사한 요구사항을 구현한 소스 코드를 그대로 복사·수정하거나 기존의 서브루틴(subroutine)을 반복 호출한다. 때문에 소프트웨어는 방대해지고 복잡하게 된다[6]. 또한 이러한 소프트웨어는 낮은 품질과 높은 가격의 유지보수 비용이 든다[7].

위와 같은 소프트웨어의 비효율적인 구조와 복잡함을 막기 위해 소프트웨어를 재작성을 해야 한다. 리팩토링은 소프트웨어의 재작성을 통해 소프트웨어의 재사용성과 유연성을 고양하고 유지보수의 비용을 절감하는데 그 목적이 있다. 또한 소프트웨어의 구조를 단순화하고 개선함과 동시에 이전의 행동에 변화가 없도록 보장해주어야 한다[2,3,4,5,6].

본 논문에서는 리팩토링의 사례연구로써 구성은 다음과 같다. 2장에서는 리팩토링을 위한 관련 연구로써 추상 슈퍼클래스의 생성, 디자인패턴과 기존 프로그램의 합성에 대해 언급한다. 3장에서 표준프로파일 제공시스템[9] 중 통합검색을 리팩토링하며 4장에서는 향후 연구과제와 결론에 대해 언급하겠다.

2. 관련 연구

2.1. 추상 슈퍼클래스의 추출

추상 클래스는 슈퍼클래스로써 디자인된다. 추상 클래스는 서브클래스들에게 통일된 인터페이스를 제공해고 실제적인 구현은 서브클래스에서 함으로써 클래스의 구조를 단순화시키며 서브클래스의 일반화한다. 서브클래스들의 대표성을 띠고 일반화를 시키는 추상 슈퍼클래스의 추출 및 생성은 다음과 같은 과정을 거친다[6].

첫째, 소프트웨어의 클래스 구조에서 비슷한 레벨과 비슷한 역할을 하는 클래스들을 찾고 이들을 위한 추상 슈퍼클래스의 역할을 하는 유일한 이름을 가진 빈 클래스를 생성한다. 둘째, 비슷한 기능을 수행하는 메소드들을 찾고 슈퍼클래스에 시그내쳐(signature)를 추가하고 서브클래스에 해당하는 메소드를 적합하게 바꾼다. 셋째, 프로그램의 구조나 데이터 타입 등과 같은 호환성을 맞추고 메소드의 몸체(body)를 만든다. 넷째 서브클래스에 있는 공통된 속성(attribute)을 추상 클래스의 속성으로 이동시킨다. 마지막 단계로써 프로그램의 선조건(precondition)과 후조건(postcondition)의 변화가 없도록 하여 공통된 코드부분을 이동시킨다.

2.2. 디자인 패턴과 디자인 패턴의 발견 및 적용

디자인 패턴은 소프트웨어의 구조를 명백하게 하고 객체들의 상호 작용의 경향을 제공하고 구성요소들 간에 약한 결합을 함으로써 소프트웨어의 문서화나 유지보수의 비용절감을 꾀할 수 있다. 리팩토링 과정에서 디자인 패턴의 이용은 소프트웨어의 유연성과 재사용성을 증가시키고 개발자에게 있어서 공통된 생각을 제공해줌으로써 개발자 상호간의 의사소통을 원활히 하도록 도와 준다[1,2,3,4,7].

디자인 패턴의 발견과 적용은 그림 1)과 같은 과정을 거친다.

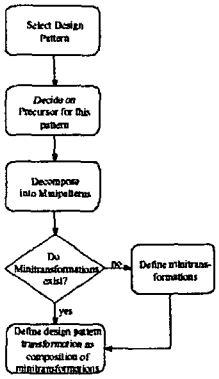

그림 1) The Design Pattern Methodology[3]

2.3. 기존 프로그램들의 합성

리팩토링은 필요에 따라 기존의 프로그램들을 합성해야 하며 합성 후에 적합여부를 검사하기 위해 선·후조건을 판단해야 한다. 기존 소프트웨어들의 합성을 위한 리팩토링은 연결방법(chaining)과 집합반복방법(set iteration)이 있다[5].

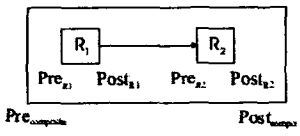

그림 2) 리팩토링 연결방법[5]

$$\bigwedge_{i=1}^{i=n} pre_{someRefactoring(x_i,\ldots)}$$

$$\bigwedge_{i=1}^{i=n} post_{someRefactoring(x_i,\ldots)}$$

식 1) 집합반복방법: 선·후조건[5]

연결방법은 소프트웨어의 일련의 순서에 따른 합성 방법으로써 그림 2)와 같은 연결 구조를 가지고 선·후조건를 합성하게 된다. 일련의 AND에 의해서 후조건이 결정된다. 이는 R1의 후조건이 R2의 선조건으로 연결되고 이러한 연결과정은 $Post_{R1} \wedge_{seq} Post_{R2}$를 통해 후조건이 결정된다.

집합반복방법은 연결방법의 확장된 형태로써 인자로 각 구성요소들을 가지며 리팩토링 과정을 거치는 방법이다. 이 방법은 식 1)과 같은 선·후조건의 도출과정을 거치고 이는 위에서 언급한 연결방법과 유사한 형태이다.

3. 사례 연구

3.1. 표준프로파일 제공시스템의 통합검색의 구조

표준프로파일 제공시스템은 표준프로파일 검색, 관리를 수행할 수 있고 정보전달의 목적으로 공지사항, 토론 게시판, 표준초안, 내부 자료실 등과 같은 여러 공개게시판과 연구과제를 수행과 의사교환을 할 수 있도록 연구 위원회별로 접근 제한이 있는 자료실, 게시판을 두었다[9].

표준프로파일 제공시스템의 통합검색은 표준프로파일 검색과 공개와 권한제한이 있는 여러 게시판들을 한 곳에서 동일한 화면 인터페이스를 통해 검색을 한다. 표준프로파일 검색은 표준제목, 키워드, 설명, 선정이유, 효과, 기타와 같은 표준프로파일의 여러 필드를 사용자의 질의 조건에 따라 검색한다. 게시판의 검색은 공개 게시판과 자료실뿐만 아니라 사용자 접근권한이 있는 각 위원회 게시판과 자료실을 모두 검색한다.

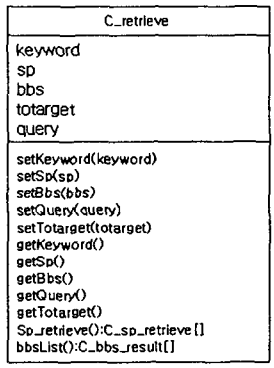

그림 3) 통합검색을 위한 클래스

현재 통합검색은 하나의 클래스로 모든 기능을 수행한다. 통합검색에 해당하는 클래스는 사용자의 질의를 생성하여 표준프로파일 또는 각 게시판과 자료실을 검색한다. 속성인 keyword, sp, bbs, totarget, query는 set 메소드를 사용하여 사용자의 입력을 받아 데이터베이스에 질의 할 수 있도록 적합한 시스템 질의를 만들고 Sp_retrieve()를 사용하여 표준프로파일을 검색하고 bbsList()를 사용하여 각 게시판과 자료실을 검색한다.

3.2. 리팩토링 사례

검색부분은 기존의 표준프로파일 검색 프로그램과 각 게시판 별로 하나의 게시판을 검색할 수 있는 프로그램이 존재한다. 통합검색은 기존의 두 프로그램을 통합검색에 적합하도록 소스코드를 복사·수정한 것이다. 따라서 기존의 프로그램을 복사·수정이 아닌 유지보수가 편이하도록 기존의 두 프로그램과 이를 합성하여 리팩토링을 해야한다. 자바 클래스와 JSP의 연계를 위해 사용되는 set, get 메소드는 실제 기능 수행에는 영향을 미치지 못하기 때문에 앞으로 나오는 클래스 다이어그램에서는 생략하고 주요 메소드만을 보여준다.

3.2.1. 통합검색 프로그램의 기능별 분석 및 분해

통합검색은 사용자의 질의를 입력받아 검색 대상을 선택한다. 이후 표준프로파일 검색과 게시판 검색을 위해 JDBC를 통해 데이터베이스에 연결하기 위해 연결 객체를 생성하고 이를 리턴받아 데이터베이스에서 통합검색을 수행한다. 위와 같이 하나의 클래스에서 시스템 질의 생성, JDBC를 이용한 데이터베이스 연결객체 생성, 검색을 모두 수행한다. 위

와 같은 기능을 하는 통합검색 클래스(그림 3)를 기능별로 분해하고 리팩토링해야 할 요소를 찾으면 다음과 같다.

사용자의 질의를 화면 인터페이스를 통해서 입력받아 프로그램 내에서 수행할 수 있도록 시스템 질의로 전환하는 부분과 Sp_retrieve()와 bbsList()에 공통적으로 있는 데이터베이스 접속을 위한 연결객체 생성부분과 마지막으로 프로파일과 게시판을 검색할 수 있는 부분으로 나눌 수 있다.

3.2.2. 기능별 프로그램의 합성

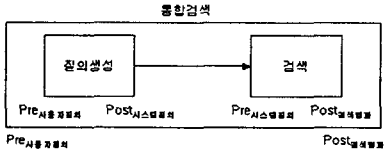

그림 4) 통합검색의 리팩토링 연결방법

앞서 언급한 바와 같이 통합검색은 화면 GUI를 담당하면서 사용자의 질의를 입력받아 시스템 질의를 생성해주는 부분과 검색 부분은 서로 합성되어야 한다. 통합검색 클래스인 C_retrieve 클래스를 기능별로 질의 생성부분과 검색부분으로 나누고 선·후조건의 변화 없이 그림 4)와 같이 연결방법으로 합성하여 리팩토링한다. 검색부분은 표준프로파일 검색과 게시판 검색하는 각각의 클래스로 나누는 형태로 만든다. 이때 검색을 수행하는 기존의 클래스를 이용한다. 그림 5)는 리팩토링 초기단계의 클래스 다이어그램이다. 표준프로파일 제공시스템 내의 모든 게시판 검색을 위해 클래스의 전환은 뒤에서 설명한다.

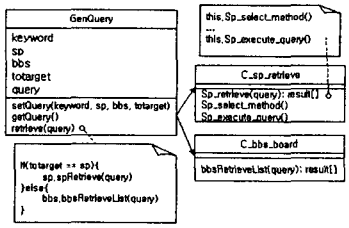

그림 5) 수정된 통합검색의 클래스 다이어그램 (Ver1): 기존의 개발된 클래스 이용

3.2.3. 추상 슈퍼클래스의 추출과 디자인 패턴을 이용한 리팩토링

그림 5)에 나타나있는 C_bbs_board 클래스는 현재 하나의 게시판만을 대상으로 검색을 수행하기 위한 것으로 모든 게시판을 권한유무를 판단하여 통합검색을 수행하기 위한 추가적인 기능이 필요하다. 어떤 게시판에 접근하고자 할 때 게시판 접근 전에 사용자의 접근 권한이 있는지를 판별하는 한 후에 C_bbs_board의 객체를 사용하여 검색이 가능하다. 그러므로 C_bbs_board 클래스에 추가적인 기능을 할 수 있도록 각 게시판의 사용자 권한 유무를 판별하고 다음으로 각 게시판들의 리스트를 만들어 검색을 할 수 있도록 한다. 이러한 기능을 추가해 주고 기존의 C_bbs_board 클래스를 이용할 수 있도록 하기 위해 Decorator 패턴[1](그림 6)을 이용한다.

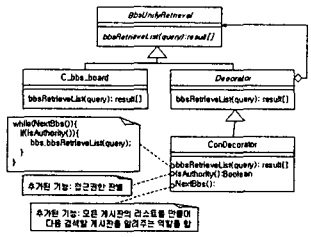

그림 6) 통합검색을 위해 C_bbs_board에 Decorator 패턴 적용

Decorator 패턴이 적용된 BbsUnifyRetrieval 클래스와 C_sp_retrieve 클래스는 검색을 위한 것으로 목적과 형태가 유사하고 단지 검색 대상과 알고리즘만 다르다. GenQuery 클래스에서는 통합검색을 수행하며 추후에 표준프로파일이나 게시판의 검색 알고리즘의 변경과 요구사항이 변경되어 첨부파일과 같은 기타 다른 데이터의 검색 방법을 제공을 고려해야 한다. 이와 같은 문제를 해결하고 동일한 인터페이스를 제공해야 한다. 이를 위해 Strategy 패턴[1](그림 7)을 적용하여 동일한 인터페이스로 검색을 수행하고 추후에 발생할 요구사항을 고려한다.

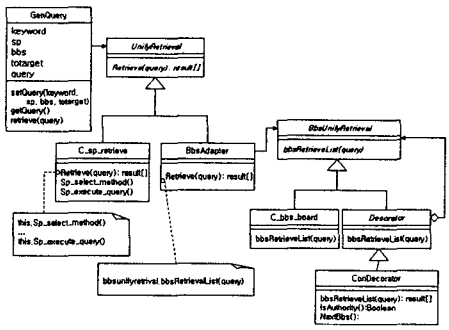

그림 7) 수정된 통합검색의 클래스다이어그램 (Ver2): Strategy 패턴과 Adapter 패턴 적용

Strategy 패턴을 사용하기 위해서 먼저 C_sp_retrieve 클래스와 BbsUnfiyRetrieve 클래스의 추상 슈퍼클래스를 찾고 각 클래스의 메소드의 이름과 시그내쳐를 수정해야 한다. 이때 Decorator 패턴이 적용된 BbsUnfiyRetrieve 클래스는 서브클래스들에게 영향이 없도록 Adapter 패턴[1]을 사용한다.

C_sp_retrieve의 객체와 C_bbs_board의 객체는 데이터베이스 접속을 위한 연결 객체를 생성하는 구현부분이 공통적으로 있다. 만약 데이터베이스의 환경설정 변경이나 다른 데이터베이스로의 변경이 발생할 경우 C_sp_retrieve 클래스와 C_bbs_board 클래스 내에 있는 데이터베이스 연결 객체 생성부분을 모두 수정해야 하는 단점이 있다. 이를 해결하기 위해 C_sp_retrieve 클래스와 C_bbs_board 클래스 내에 있는 데이터베이스 연결 객체 생성부분의 소스코

드를 삭제해 주고 Template Method 패턴[1](그림 8)을 사용한다. 이렇게 Template Method 패턴을 사용하여 데이터베이스 연결객체 생성부분을 클래스로 만들어 줌으로써 데이터베이스의 환경설정 변경이나 다른 데이터베이스의 변경에 따라 적합한 연결객체를 생성해줄 수 있다.

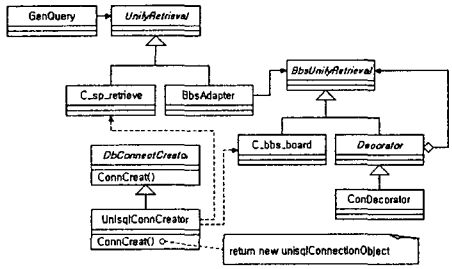

그림 8) 수정된 통합검색의 클래스다이어그램 (Ver3): Template Method 패턴 적용

4. 결론 및 향후 연구 과제

소프트웨어를 개발하는데 있어서 재사용성과 유연성을 높여주는 것은 어려운 일이다. 리펙토링은 이러한 문제를 해결하고자 하며 기존의 개발된 소프트웨어를 재사용성과 유연성이 높은 소프트웨어로 전환해나가는 과정이다. 리펙토링을 하는데 있어서 디자인 패턴은 중요한 역할을 담당한다. 디자인 패턴을 사용함으로써 요구사항의 변화에 소스코드의 큰 변화 없이 문제를 해결해 줄 수 있다. 또한 소프트웨어 개발자간의 의사소통을 원활히 할 뿐만 아니라 전체적인 소프트웨어 구조를 빠르고 정확하게 이해할 수 있다. 이때 기존 소프트웨어 소스 코드에서 일반화하기 위한 추상 슈퍼클래스의 생성과 적합한 디자인 패턴의 발견 및 적용이 중요하다. 본 논문에서는 이러한 방법을 이용하여 표준프로파일 제공시스템의 통합검색 부분을 리펙토링 과정을 보였으며 이를 통해 통합검색의 추후 요구사항을 고려하고 유연성을 높이고자 하였다.

현재 표준프로파일 제공시스템은 MVC 모델[1]을 사용하여 모델(Model), 뷰(View), 컨트롤러(Controller)를 분리시킴으로써 유지보수의 비용을 줄이는데 노력하여 리펙토링을 훨씬 쉽게 할 수 있다. 현재는 통합검색 부분의 리펙토링을 하였으나 표준프로파일 제공시스템의 전반에 대한 리펙토링을 해야할 것이며 리펙토링에 따른 전체적인 시스템 구조변화에 대해 고려해야 한다.

또한 데이터베이스의 스키마(schema)에 대한 변경을 고려해야 한다[6,8]. 데이터베이스의 스키마의 변경은 소프트웨어의 리펙토링과 유사하다. 데이터베이스 스키마에 대한 변경은 데이터에 초점을 두었고 소프트웨어의 리펙토링은 데이터와 프로그램에 초점을 두었다는 것에 차이가 있다[6]. 현재 관계형 스키마를 사용하고 있으나 Unisql[9]은 ORDBMS (Object-Relation Database Management System)로 객체지향 방식의 스키마 설계가 가능하다. 그러므로 객체지향 언어인 자바의 특성을 최대한 활용하기 위해서는 객체지향 방식의 스키마로의 변경이 필요하며 이와 함께 데이터의 변경에 유의해야 한다.

마지막으로 본 논문에서의 리펙토링은 주로 휴리스틱(heuristic) 방법[6]에 의해서 하였으나 도구를 이용한 리펙토링과 좀더 기계적이고 자동적인 리펙토링에 관한 연구가 필요하다.

참고문헌

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley 1995

[2] Mel O Cinneide. *Automated Refactoring to Introduce Design Patterns.* Proceedings of the 22nd International Conference on Software Engineering, 2000, 722-724

[3] Mel O Cinneide, Paddy Nixon. *Automated Applicatin of Design Patterns to Legacy Code.* Proceedings of the Workshop on Experiences in Object-Oriented Re-Engineering, European Conference on Object-Oriented Programming, Lisbon, June 1999.

[4] Mel O Cinneide, Paddy Nixon. *Program Restructuring to Introduce Design Patterns.* Proceedings of the Workshop on Experiences in Object-Oriented Re-Engineering, European Conference on Object-Oriented Programming, Brussels, July 1998.

[5] Mel O Cinneide, Paddy Nixon. *Composite Refactorings for Java Programs.* Proceedings of the Workshop on Formal Techniques for Java Programs, European Conference on Object-Oriented Programming. 2000.

[6] W. F. Opdyke and R. E. Johnson. *Creating Abstract Superclasses by Refactoring,* Proceedings of CSC'93: the ACM 1993 Computer Science Conference, 1993

[7] Hyoseob Kim, Cornelia Boldyreff. *A Case Study on Design Pattern Discovery in Ada.* ACM SIGADA Ada Letters 1997

[8] Lance Tokuda and Don Batory. *Evolving Object-Oriented Designs with Refactorings.* In Proceedings of ASE-99: The 14th IEEE Conference on Automated Software Engineering. IEEE CS Press, October 1999.

[9] 양진혁, 김영도, 정희준, 양진영, 유명환, 공유근, 정인정 김정엽, 정희창. 정보기술 아키텍처를 위한 기술참조모델을 지원하는 표준프로파일 관리 시스템 개발에 관한 연구. 정보처리학회논문지D 제8-D권 제 6호 2001.12.