리팩토링 조립을 위한 메타모델

김은지, 김경민, 김태공
인제대학교 전산학과
e-mail: ejkim8511@naver.com, {kmkim, ktg}@cs.inje.ac.kr

The Meta-model for Composition of Refactoring

Eun-Ji Kim, Kyung-Min Kim, Tae-Gong Kim
Dept. of Computer Science, Graduate School of Inje University

요 약

리팩토링은 프로그램의 행위를 변경하지 않으면서 프로그램의 내부 구조를 수정하는 과정이다. 소프트웨어의 유지보수 측면에서 유용하기 때문에 리팩토링이 활발하게 이용되고 있다. 리팩토링에 대한 관심이 높아지면서 요소 리팩토링을 정의해서 이들의 조립을 통해 새로운 복합 리팩토링을 정의하려는 연구들이 많이 진행되고 있다. 그러나 이러한 연구들에서는 복합 리팩토링을 구성하고 있는 조립된 리팩토링의 종류가 요소 리팩토링으로 제한되어 있고, 조립 방법도 대부분 요소 리팩토링들의 순차적인 결합으로 되어 있는 경우에만 처리되고 있다. 이에 본 연구에서는 조립할 수 있는 리팩토링의 구성 요소를 요소 리팩토링뿐만 아니라 이미 정의되어 있는 복합 리팩토링도 가능하게 하고, 조건처리와 반복 등 다양한 방법으로 리팩토링들을 조립할 수 있는 메커니즘을 제안한다. 그리고 이러한 다양한 형태의 조립 메커니즘을 지원해주는 메타모델을 정의한다. 이를 통해 리팩토링들의 재사용성과 확장성을 향상시킬 수 있을 것이다.

1. 서론

리팩토링은 프로그램의 행위를 변경하지 않으면서 프로그램의 내부 구조를 수정하는 과정이다[1]. 이러한 리팩토링의 결과로 코드의 확장성, 모듈화, 재사용성, 유지보수성 같은 품질을 개선하여 개발의 속도를 높이고 코드의 복잡도를 낮출 수 있다[2]. 소프트웨어의 유지보수 측면에서 유용하기 때문에 최근에는 소프트웨어의 개발에서도 리팩토링을 많이 이용하고 있다.

리팩토링에 대한 관심이 많아지면서 리팩토링의 자동화와 리팩토링 재사용을 위한 조립에 대한 연구들[3,4,5,6,7]이 많이 진행되고 있다. 기존 연구들[8,9,10]에서는 작은 단위로 정의되어 있는 요소 리팩토링들의 조립으로 새로운 복합 리팩토링을 정의하고 있다. Roberts의 연구[8]에서는 요소 리팩토링들의 순차적인 결합 'chain'으로 리팩토링을 조립하고 있다. Cinneide의 연구[9]에서의 리팩토링 조립은 요소 리팩토링들이 순차적으로 적용되는 'Chainning'과 반복적으로 적용되는 'Set iteration'을 리팩토링 조립 방법으로 제안하고 있다.

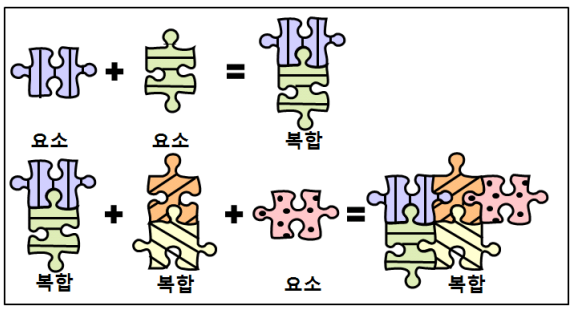
이처럼 기존 연구들에서는 복합 리팩토링을 정의하기 위하여 리팩토링 조립의 대상을 요소 리팩토링으로 제한하고 있고, 조립 방법도 순차적인 방법과 반복적인 방법만 사용하고 있다. 이것은 요소 리팩토링의 개수에 영향을 받아 복합 리팩토링을 정의할 수 있는 경우의 수가 제한된다.

이에 본 연구에서는 복합 리팩토링을 정의하는데 사용하는 리팩토링의 구성 요소를 요소 리팩토링과 정의되어 있는 복합 리팩토링으로 확장하고, 리팩토링의 조립 방법도 조건처리와 반복 등 다양한 방법이 가능하도록 조립 메커니즘을 제안한다. 이러한 다양한 형태의 조립 메커니즘을 지원해주는 메타모델을 정의한다. 이를 통해 리팩토링들의 재사용성과 확장성을 향상시킬 수 있을 것이다.

2. 리팩토링 조립 메커니즘

2.1 리팩토링 조립을 위한 구성요소

본 연구에서는 복합 리팩토링을 구성하는 리팩토링의 종류를 요소 리팩토링과 복합 리팩토링으로 확장한다. 요소 리팩토링뿐만 아니라 정의되어 있는 복합 리팩토링도 새로운 복합 리팩토링을 정의하는데 조립될 수 있다면 좀 더 다양하고 큰 복합 리팩토링을 정의할 수 있다. 그림 1은 복합 리팩토링의 구성요소를 나타낸 것이다.

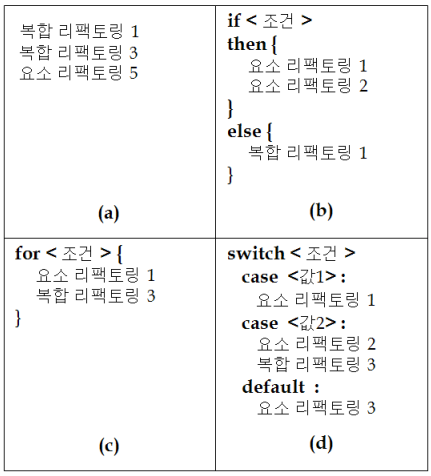


(그림 1) 복합 리팩토링의 구성 요소

그림 1에서와 같이 두 개의 요소 리팩토링으로 새로운 복합 리팩토링을 정의할 수 있고, 두 개의 정의되어 있는 복합 리팩토링과 하나의 요소 리팩토링으로 커다란 복합 리팩토링을 정의할 수 있다.

2.2 리팩토링 조립 메커니즘

2.1의 구성요소를 이용하여 다양한 형태의 복합 리팩토링을 정의할 수 있도록 리팩토링의 조립 메커니즘을 제안한다. 리팩토링의 조립 메커니즘으로는 순차적인 방법, 조건문, 반복문, 분기문이 있다. 그림 2는 리팩토링의 조립 메커니즘을 종류별로 나타낸 것이다.



(그림 2) 리팩토링 조립 메커니즘

그림 2의 (a)는 순차적인 방법이다. 대부분의 기존 연구에서 리팩토링의 조립 방법으로 사용하는 방법이다. 순차적인 방법은 조립된 리팩토링을 순차적으로 수행하는 것이다. 정의되어 있는 복합 리팩토링 1을 수행하고, 다음으로 복합 리팩토링 3과 요소 리팩토링 5를 순서대로 수행하는 것이다. 만약 3개의 리팩토링을 순차적인 방법만으로 조립한다면 정의할 수 있는 복합 리팩토링의 경우의 수는 6가지이다. 이처럼 순차적인 방법만으로 복합 리팩토링을 정의한다는 것은 매우 제한적이다. 그리하여 순차적인 방법 외에 조건문, 반복문, 분기문과 같은 방법을 제안하고자 한다.

그림 2의 (b)는 조건문을 사용하는 방법이다. 제시된 조건이 참일 경우에는 then 다음에 조립된 요소 리팩토링 1과 2를 순차적으로 수행하고, 조건이 거짓일 경우 else 다음에 조립된 복합 리팩토링 1만 수행하게 되는 방법이다. 이렇게 조건문을 사용하게 되면 조건에 따라 수행되는 리팩토링의 종류도 개수도 다르게 정의할 수가 있다.

그림 2의 (c)는 반복문을 사용하는 방법이다. 반복문에도 제시되는 조건은 어떤 조건에 따라 결정되는 요소들의 집합이다. 이러한 집합에 포함되어 있는 요소들의 개수만큼 요소 리팩토링 1과 복합 리팩토링 3을 수행하는 일이 반복 될 것이다. 이러한 반복문은 요소의 개수에 따라 조립된 리팩토링의 수행을 유동적이게 할 수 있다.

마지막으로 그림 2의 (d)는 분기문을 사용하는 방법이다. 분기문에서는 조건의 결과에 따라 다양한 경우로 분기할 수 있다. 조건의 결과가 값1이라면 요소 리팩토링 1을 수행하고, 결과가 값2라면 요소 리팩토링 2와 복합 리팩토링 3을 수행하고, 결과에 만족하는 경우가 없다면 default로 요소 리팩토링 3을 수행하는 것이다. 조건문을 사용하는 방법과 같이 조건이 참과 거짓으로만 분리되는 것이 아니라 조건의 결과에 따라 여러 가지 경우에 대해 처리할 수 있다.

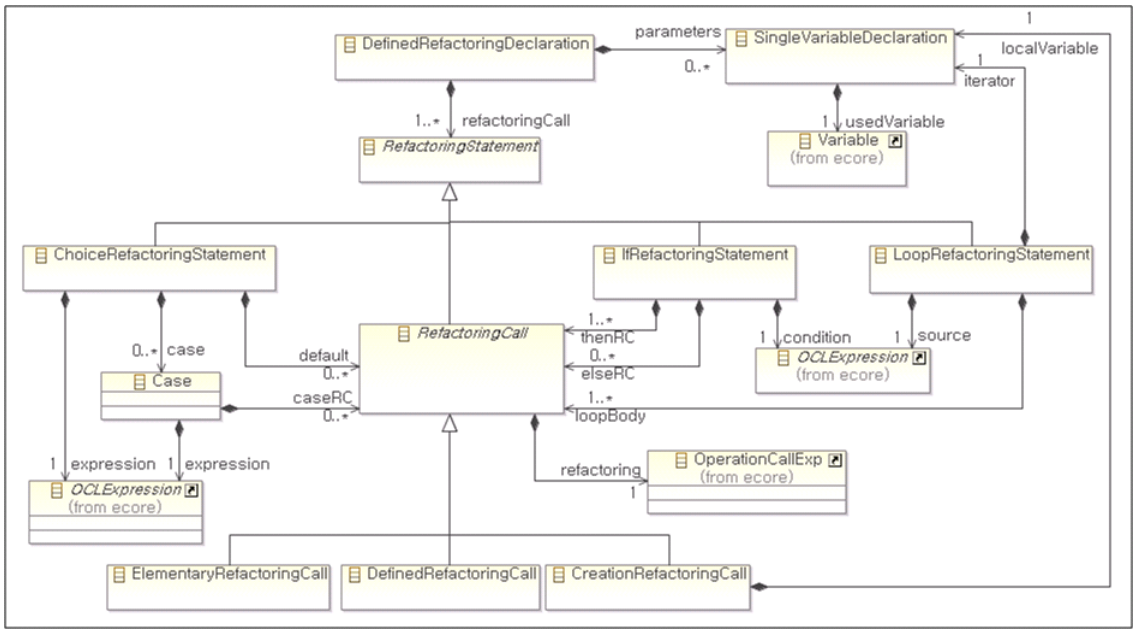
본 연구에서는 순차적인 방법까지 포함하여 리팩토링 조립 방법으로 4가지의 방법을 제안하였다. 이러한 조립 메커니즘을 사용하여 리팩토링들을 조립한다면 크고 다양한 복합 리팩토링을 정의 할 수 있을 것이다.

3. 리팩토링 조립을 위한 메타모델

본 연구에서 제안하는 조립 메커니즘을 지원해주는 메타모델을 정의한다. 이 메타모델은 EMF(Eclipse Modeling Framework)[11]를 기반으로 정의되었으며, 복합 리팩토링에 필요한 인자 값들과 리팩토링 조립 메커니즘에 관한 정보를 관리한다. 그림 3은 메타모델의 구조이며, 표 1에서 메타모델의 모델 엘리먼트들을, 표 2에서는 모델 프로퍼티들을 간단히 설명한다.

<표 1> 메타모델의 모델 엘리먼트 설명

요소 이름	설 명
DRC	DefinedRefactoringDeclaration 메타모델의 최상위 엘리먼트
CRS	ChoiceRefactoringStatement 리팩토링 조립 메커니즘의 분기문 엘리먼트
Case	분기문에서의 분기를 나타내는 엘리먼트
IRS	IfRefactoringStatement 리팩토링 조립 메커니즘의 조건문 엘리먼트
LRS	LoopRefactoringStatement 리팩토링 조립 메커니즘의 반복문 엘리먼트
ERC	ElementaryRefactoringCall 작은 단위로 정의되어 있는 요소 리팩토링을 나타내는 엘리먼트
DRC	DefinedRefactoringCall 정의되어 있는 복합 리팩토링을 나타내는 엘리먼트
CRC	CreationRefactoringCall 생성하는 리팩토링을 나타내는 엘리먼트
RC	RefactoringCall 복합 리팩토링의 구성요소들(ERC, DRC, CRC)의 상위 엘리먼트
RS	RefactoringStatement 리팩토링의 조립 메커니즘(CRS, IRS, LRS, RC)의 상위 엘리먼트
SVD	SingleVariableDeclaration 복합 리팩토링의 인자 값을 나타내는 엘리먼트



(그림 3) 리팩토링 조립을 위한 메타모델

<표 2> 메타모델의 모델 프로퍼티 설명

프로퍼티 이름	설 명
DFD의 parameters	인자 값을 나타내는 SVD를 가지는 프로퍼티
DRD의 refactoringCall	RS를 가지는 프로퍼티
SVD의 usedVariable	인자 값을 EMF의 Ecore Metamodel의 Variable로 나타내는 프로퍼티
CRS의 expression	조건을 EMF의 Ecore Metamodel의 OCLExpression으로 나타내는 프로퍼티
CRS의 case	다양한 경우를 나타내는 Case를 가지는 프로퍼티
CRS의 default	조건의 결과와 일치하는 경우가 없을 때 실행할 RC들을 가지는 프로퍼티
Case의 expression	조건의 결과 값을 OCLExpression으로 나타내는 프로퍼티
Case의 caseRC	조건의 결과 값이 같을 때 실행할 RC들을 가지는 프로퍼티
RC의 refactoring	리팩토링의 호출을 EMF의 Ecore Metamodel의 OperationCallExp로 나타내는 프로퍼티
CRC의 localVariable	생성된 요소를 SVD로 가지는 프로퍼티
IRS의 condition	조건을 OCLExpression으로 나타내는 프로퍼티
IRS의 thenRC	조건이 참일 때 실행할 RC들을 가지는 프로퍼티
IRS의 elseRC	조건이 거짓일 때 실행할 RC들을 가지는 프로퍼티
LRS의 source	조건의 집합을 OCLExpression으로 나타내는 프로퍼티
LRS의 iterator	집합의 요소를 SVD로 가지는 프로퍼티
LRS의 loopBody	반복적으로 실행할 RC들을 가지는 프로퍼티

4. 적용사례

4개의 리팩토링 'pullUpFieldHavingOnlyOneVDF', 'createFDforPullUpVDF', 'addField', 'separateVDF'를 이용하여 'PullUpVDF' 복합 리팩토링을 정의한다. 표 3은 복합 리팩토링을 구성하는 4개의 리팩토링에 대해 설명한다. 그림 4는 4개의 리팩토링들을 조립하여 'PullUpVDF' 복합 리팩토링을 정의한 것이다.

<표 3> 복합 리팩토링을 구성하는 리팩토링

리팩토링 종류	설 명
pullUpFieldHavingOnlyOneVDF	Field(FD)에 하나의 VDF만 있으면 부모 클래스로 Field를 올리는 요소 리팩토링
createFDforPullUpVDF	주어진 VDF의 타입과 접근지정자가 같은 새로운 FD를 만들어주는 생성 리팩토링
addField	생성된 FD를 실제 인스턴스 모델에 추가시키는 요소 리팩토링
separateVDF	주어진 VDF를 포함하고 있는 FD에서는 VDF를 분리하고, 생성된 새로운 FD에 VDF를 추가시키는 요소 리팩토링

```
if <vdf.container.fragments->size() = 1 >
 then {
    pullUpFieldHavingOnlyOneVDF(vdf.container) ;
 }
 else {
   FieldDeclaration newFD = createFDforPullUpVDF(vdf.container) ;
   addField(newFD, vdf.container.container);
   separateVDF(vdf, newFD);
   pullUpFieldHavingOnlyOneVDF(newFD);
 }
```

(그림 4) 'PullUpVDF(vdf:VDF)' 복합 리팩토링

그림 4에서 'PullUpVDF' 복합 리팩토링은 하나의 인자값(vdf)을 사용하고, 조립 방법으로는 조건처리 방법을 사용하고 있다. 꺽세 기호 '<' 안에 명세되어 있는 조건문의

조건이 참일 경우 'pullUpFieldHavingOnlyOneVDF' 요소 리팩토링을 수행하고, 조건이 거짓일 경우 'createFDfor-PullUpVDF', 'addField', 'separateVDF', 'pullUpField-HavingOnlyOneVDF'를 순차적으로 수행하면 된다. 그림 5는 메타모델을 기반으로 'PullUpVDF' 복합 리팩토링을 직렬화한 XMI 문서이다.

```
1 <?xml version="1.0" encoding="ASCII"?>
2 <DefinedRefactoringDeclaration xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" x
3   <refactoringCall xsi:type="IfRefactoringStatement">
4       <condition xsi:type="ocl.expr:OperationCallExp">
5           ...
6       </condition>
7       <thenRC xsi:type="ElementaryRefactoringCall">
8           <refactoring xsi:type="ocl.expr:OperationCallExp">
9               <source xsi:type="ocl.ecore:OperationCallExp">
10                  ...
11                  <referredOperation xsi:type="ecore:EOperation" href="ocl:///oclenv.ecore#/4/Project_Class/pullUpFieldHavingOnlyOneVDF"/>
12              </source>
13          </refactoring>
14      </thenRC>
15      <elseRC xsi:type="ElementaryRefactoringCall">
16          <refactoring xsi:type="ocl.expr:OperationCallExp">
17              <source xsi:type="ocl.ecore:OperationCallExp">
18                  ...
19                  <referredOperation xsi:type="ecore:EOperation" href="ocl:///oclenv.ecore#/1/Project_Class/createFDforPullUpVDF"/>
20              </source>
21          </refactoring>
22      </elseRC>
23      <elseRC xsi:type="ElementaryRefactoringCall">
24          <refactoring xsi:type="ocl.expr:OperationCallExp">
25              <source xsi:type="ocl.ecore:OperationCallExp">
26                  ...
27                  <referredOperation xsi:type="ecore:EOperation" href="ocl:///oclenv.ecore#/2/Project_Class/addField"/>
28              </source>
29          </refactoring>
30      </elseRC>
31      <elseRC xsi:type="ElementaryRefactoringCall">
32          <refactoring xsi:type="ocl.expr:OperationCallExp">
33              <source xsi:type="ocl.ecore:OperationCallExp">
34                  ...
35                  <referredOperation xsi:type="ecore:EOperation" href="ocl:///oclenv.ecore#/3/Project_Class/separateVDF"/>
36              </source>
37          </refactoring>
38      </elseRC>
39      <elseRC xsi:type="ElementaryRefactoringCall">
40          <refactoring xsi:type="ocl.expr:OperationCallExp">
41              <source xsi:type="ocl.ecore:OperationCallExp">
42                  ...
43                  <referredOperation xsi:type="ecore:EOperation" href="ocl:///oclenv.ecore#/4/Project_Class/pullUpFieldHavingOnlyOneVDF"/>
44              </source>
45          </refactoring>
46      </elseRC>
47  </refactoringCall>
48  <parameters>
49      <usedVariable xsi:type="ocl.ecore:Variable" name="vdf">
50          <eType xsi:type="ecore:EClass" href="JavaEAbstractSyntax#//VariableDeclarationFragment"/>
51      </usedVariable>
52  </parameters>
53 </DefinedRefactoringDeclaration>
```

(그림 5) PullUpVDF 복합 리팩토링의 XMI 문서

그림 5의 XMI문서에서는 복합 리팩토링의 인자 값(vdf)을 48번 줄에서 표현하고 있다. 리팩토링 조립 방법으로 사용한 조건처리 방법 'IfRefactoringStatement'를 3번 줄부터 47번 줄에 표현하고 있다. 조건처리 방법의 조건은 'condition' 프로퍼티로 4번 줄에, 조건이 참일 때 수행할 리팩토링들을 'thenRC' 프로퍼티로 7번 줄에, 조건이 거짓일 때 수행할 리팩토링을 15번 줄부터 46벌 줄까지 'elseRC' 프로퍼티로 표현하고 있다.

5. 결론

본 연구에서는 리팩토링의 재사용을 향상시키기 위해 리팩토링의 조립 메커니즘을 제안하고 이에 대한 메타모델을 정의했다.

조립할 수 있는 리팩토링의 구성요소를 요소 리팩토링과 정의되어 있는 복합 리팩토링으로 확장하고, 순차적, 조건처리, 반복 등 다양한 방법으로 조립이 가능하도록 조립 메커니즘을 제안했다. 이것은 리팩토링의 재사용성과 확장성을 강화시켜 리팩토링의 활용성이 더욱 향상될 수 있을 것이다.

이러한 조립 메커니즘을 기반으로 새로운 복합 리팩토링을 정의하는데 필요한 리팩토링들의 조립 방법과 선택된 리팩토링의 호출에 필요한 정보들을 관리하는 리팩토링 조립을 위한 메타모델을 정의했다. 이것은 리팩토링의 조립 방법과 복합 리팩토링의 명세를 정형화한 것이다. 적용사례를 통해 메타모델의 적합성을 확인해 보았으며, 앞으로 더 많은 사례들을 통해 메타모델의 정제가 필요하다.

현재 본 연구에서 제안한 조립 메커니즘과 메타모델을 활용하여 리팩토링 조립을 위한 도구의 개발이 진행 중에 있다.

참고문헌

[1] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, "Refactoring: Improving the Design of Existing Code," Addison Wesley, 1999.

[2] T. Mens, T. Tourwe, "A Survey of Software Refactoring," IEEE Transactions on Software Engineering, Vol. XX, No. Y, Month 2004.

[3] William F. Opdyke, "Refactoring Object-Oriented Frameworks," Ph.D thesis, University of Illinois at Urbana-Champaign, 1992.

[4] Martin Kuhlemann, Don Batory, Christian Kastner, "Safe Composition of Refactoring Feature Modules," Technical Report, School of Computer Science, University of Magdeburg, 2009.

[5] Ethan Hadar, Irit Hadar, "The Composition Refactoring Triangle(CRT) Practical Toolkit: From Spaghetti to Lasagna," ACM OOPSLA'06, 2006

[6] 김태웅, 김태공, "OCL을 이용한 자동화된 코드스멜 탐지와 리팩토링," 한국정보처리학회 논문지 D, Vol. 15-D, No.06, pp.0825-0840, 2008. 12.

[7] Javier Perez, Yania Crespo, Berthold Hoffmann, Tom Mens, "A case study to evaluate the suitability of graph transformation tools for program refactoring," International Journal on Software Tools for Technology Transfer(STTT), Vol. 12, No. 3-4, pp.183-199, 2010. 7.

[8] Donald Bradley Roberts, "Practical Analysis for Refactoring," Ph.D thesis, University of Illinois at Urbana-Champaign, 1999.

[9] Mel O Cinneide, "Automated Application of Design Patterns : A Refactoring Approach," Ph.D thesis, University of Dublin, Trinity College, October 2000.

[10] G. Kniesel, H. Koch, "Static Composition of Refactorings," In R. Lammel, editor, Science in Computer Programming; Special issue on program transformation. Elsevier Science, 2004.

[11] Eclipse, "Eclipse Modeling Framework Project," http://www.eclipse.org/modeling/emf/?project=emf