의존관계 그래프에 기반한 재사용 컴포넌트 후보 추출

강민수*, 이기열*, 이병정*, 홍의석**, 이종석***, 이숙희****, 우치수*
* 서울대학교 전산과학과 ** 안양대학교 영상처리학과 *** 우석대학교 정보통신 컴퓨터 공학부
**** 서경대학교 전자계산학과

Identifying candidate of reusable component based on dependency relation graph

Minsoo Kang*, Keeyoull Lee*, Byungjeong Lee*, Euysuk Hong**, Jongseok Lee***, Sukhee Lee****, Chisu Wu*
* Dept. of Computer Science, Seoul Nat'l University
** Dept. of Image Processing, Anyang Unviversity
*** Dept. of Computer Engineering and Infomation Communication Engineering, Woosuk University
**** Dept. of Computer Science, Seokyeong University

요 약

소프트웨어 품질 향상과 생산성 향상이라는 측면에서 소프트웨어 재사용의 중요성이 널리 인식되어지고 그에 따른 재사용 컴포넌트에 대한 관심이 나날이 커져가고 있다. 이에 따라 재사용성을 측정하는 방법에 대한 연구의 중요성이 커지고 있다 본 논문에서는 기존의 결합도를 측정하는 방법을 변형하여 두 클래스간의 의존 정도를 측정하고 클래스를 노드로, 측정한 의존 정도를 에지 값으로 하는 방향 그래프를 그린다. 그리고 그 그래프를 클러스터링을 하여 재사용 컴포넌트의 후보를 추출하는 방법을 제시한다.

1. 서론

소프트웨어 품질 향상과 생산성 향상이라는 측면에서 소프트웨어 재사용의 중요성이 널리 인식되어지고 그에 따른 재사용 컴포넌트에 대한 관심이 나날이 커져가고 있다. 소프트웨어 재사용을 통하여 제품이 출시되는 시간을 줄여주고 향상된 소프트웨어 품질과 개발과 유지 측면에서의 비용 절감을 얻을 수 있다. Henderson-Sellers 의 연구에 따르면 컴포넌트를 새로 개발하는 것보다 기존의 컴포넌트를 수정하여 재사용함으로써 얻어지는 비용절감이 약 55%에 달한다[4] 이에 따라 재사용성을 측정하는 방법에 대한 연구의 중요성이 커지고 있다[1][5].
그림 1은 재사용성을 측정하고 재사용 컴포넌트를 추출해내는 단계를 포함하는 객체지향 설계/개발 프로세스이다.

시스템을 설계하는 단계로 프로세스를 시작하고 중요한 시스템 컴포넌트들이 결정이 된 후 본 연구에서 사용할 방법을 적용하여 재사용 컴포넌트의 후보들을 추출해내고 재사용성을 측정한다 그리고 소프트웨어 설계자가 그 결과를 보고 수정한후 재사용성을 다시 측정한다 이러한 과정을 반복한 후 설계가 완전해지면 설계와 문서를 저장할 수 있다 그리고 시스템을 구현하고 그 구현된 결과물과 구현 문서를 저장한다 새로운 프로젝트가 수행이 될 때 저장해 두었던 기존의 설계와 구현 결과물을 재사용할 수 있다 이렇게 하면 모든 것을 새로 설계하거나 구현할 필요 없이 기존의 것을 사용하여 비용을 줄일 수 있다[5]

본 논문의 구성은 다음과 같다 2장에서는 기존의 재사용 컴포넌트 추출 방법에 대하여 알아보고 3장에서는 클래스 간의 결합도 측정을 통한 재사용 컴포넌트 추출 방법에 대하여 기술하고 4장에서는 결론을 맺는다.

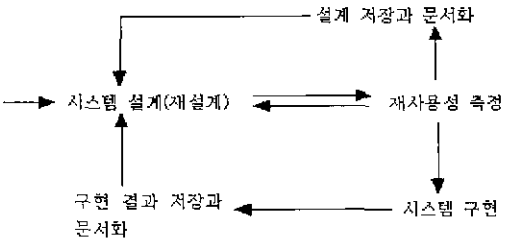

그림 1 설계/구현 프로세스

2. 관련 연구

2.1 상속 트리간의 결합도 측정을 통한 재사용성 측정[5]

작은 모듈일수록 재사용하기 쉽다는 관점이 아니라 재사용의 단위가 커질수록 재사용으로 얻는 비용절감 효과가 크다는 관점으로 접근한다

이들의 연구에서는 클래스를 두 가지 종류로 나눈다 하나는 일반 클래스(general class)로 이것은 다른 어플리케이션에서 다시

사용되어질 것이라고 기대되어지는 클래스이다. 다른 하나는 한정 클래스(specific class)로 단지 그 어플리케이션에서만 사용되어질 것이라고 예상되는 클래스이다. 상속 트리에서 하부에 있는 클래스만 재사용하는 것은 불가능하기 때문에 일반 클래스는 상속 트리의 상부에, 한정 클래스는 하부에 위치해야 한다.

그리고 상속 트리간의 관계를 연관된(related) 상속 트리와 연관되지 않은(unrelated) 상속 트리로 나누는데 다른 어플리케이션에서 같이 재사용되어질 것이라고 기대되어지는 상속 트리가 연관된 상속 트리이다.

이 연구에서 제안된 설계 목표는 두 가지로 정리된다.

첫째는 연관된 일반 클래스들 사이에는 많은 결합이 있다.

둘째는 연관되지 않은 상속 트리 사이에는 오직 한정 클래스들 사이에만 결합이 있어야 한다.

이같은 설계 목표를 가지고 상속 관계 사이의 결합도를 가지고 재사용성을 측정한다.

2.2 호출 서열 트리(call dominance tree)를 이용한 재사용 컴포넌트 추출[3]

이 연구의 대상은 전통적인 언어를 이용한 설계/개발이다. 기존 시스템의 호출 그래프를 서열 그래프(dominance graph)로 변환한 것을 바탕으로 몇 개의 기준들을 제시하고 이 그래프의 서열 관계를 함수적 의존 관계(functional dependency relationship) 로 변역한다. 제안된 기준들은 4개의 룰(rule)로 정리되어 후보 모듈의 프로시져들의 집합체, 그리고 그러한 모듈들 사이의 uses 관계와 is_component_of 관계의 정의에 직용이 된다. 결국 함수적 추상화(functional abstraction)을 구현한 기존의 소프트웨어 컴포넌트를 클리스티링하여 재사용 가능한 모듈을 찾아내어 재사용 제공학에 적용할 수 있다.

3. 클래스 결합도를 사용한 재사용 컴포넌트 추출

본 연구에서는 기존의 결합도를 측정하는 방법을 변형하여 두 클래스간의 의존 정도를 측정하고 클래스를 노드(node)로, 측정한 의존 정도를 에지(edge)값으로 하는 방향 그래프(directed graph)를 그린다. 그리고 그 그래프를 클러스터링을 하여 재사용 컴포넌트(reusable component)의 후보(candidate)를 얻는다.

3.1 클래스 결합도를 이용한 의존관계 그래프

클래스와 클래스 사이의 의존 관계를 이루는 연결 방식은 다음과 같이 7가지로 나누어 생각할 수 있다[2].

(클래스 c, c의 속성 a, c의 메쏘드 m | 클래스 d(≠c), d의 속성 a′, d의 메쏘드 m′)

1 c가 d의 하위클래스
2 d가 a의 타입 (2번부터 7번까지는 c와 d는 상속관계를 이루지 않는 경우이다.)
3 d가 m의 매개변수 타입 또는 반환형
4 d가 m의 지역변수 타입
5 d가 m에서 호출된 메쏘드의 매개변수 타입
6 m이 a′을 참조
7 m이 m′을 호출

위의 경우 c는 d를 의존한다.

좀더 자세히 살펴보면 1번의 경우 상속을 받는다는 것은 상위 클래스의 모든 정보를 사용한다는 것이므로 가장 강력한 의존관계가 되고 상속 트리 내의 임의의 클래스가 재사용된다고 한다면 그 클래스의 선조 클래스도 함께 재사용되어져야만 한다.

2번의 경우 클래스 c는 d의 정보를 모두 가지고 있어야 하고 a를 사용하는 모든 메쏘드가 d의 정보를 사용하는 것이므로 c의 d에 대한 의존도는 아주 높다.

3번과 4번은 c 전체에서 d에 대한 모든 정보를 알 필요 없이 m에서만 알고 있으면 된다. 따라서 2번에 비해 연결의 강도가 낮고 3번, 1번의 연결 강도는 같다.

5번의 경우 m에서 다른 클래스(c′)의 메쏘드를 호출할 때 새로 객체를 만들고 메쏘드를 호출하게 된다. 이 경우 c는 c′에 대해 4번의 의존관계를 가지고 c′은 d에 대해 3번의 의존관계를 가진다. 따라서 5번은 4번과 3번이 연결된 것이므로 c와 c′의 의존정도, c′과 d의 의존정도를 측정하여 알 수 있어 직접 c와 d의 의존정도를 측정할 필요가 없다.

6번의 경우 m이 a′을 직접 참조한다는 것은 객체지향 방법론의 중요한 특징인 캡슐화에 어긋나는 것이다. 일반적인 경우 다른 클래스의 속성은 직접 사용하지 않고 접근 메쏘드를 사용하는데 이것은 7번과 같은 것이다. 따라서 6번과 7번의 연결 강도는 같다. 그리고 한 메쏘드 안에서 타입이 d인 객체를 여러 개 생성하는 것이 객체 하나를 생성하여 그것의 메쏘드나 속성을 여러번 사용하는 것보다 강하게 연결된 것이므로 4번은 6번과 7번에 비해 연결 강도가 강하다.

본 연구에서 재사용 컴포넌트의 후보를 추출하는 것은 설계 단계 다음의 작업이므로 구현 단계 이후의 작업으로 결정이 되는 4번, 6번, 7번은 고려하지 않는다.

그러고 50개의 속성을 가지고 있는 클래스 c에서 다른 클래스 d를 5번 속성의 타입으로 사용하는 것보다는 4개의 속성을 가지고 있는 클래스 c에서 다른 클래스 d를 1번 속성의 타입으로 사용하는 것이 의존 관계가 더 높다고 할 수 있다. 그것은 메쏘드의 매개변수 타입과 반환형으로 쓰이는 회수에도 마찬가지로 적용된다.

위의 결론을 바탕으로 클래스 c_i가 클래스 c_j에 의존하는 정도 d_{ij}를 나타내면 다음과 같다.

$$d_{ij} = \frac{|\bigcup a_i \in C_j| + \sum_{k=1}^{|M_i|}(|\bigcup p_{ik} \in C_j| + |\bigcup r_{ik} \in C_j|)}{|A_i| + \sum_{k=1}^{|M_i|}(|P_{ik}| + |R_{ik}|)}$$

a_i C_i에 속한 속성 타입
A_i C_i에 속한 속성들의 집합
M_i C_i에 속한 메쏘드들의 집합
p_{ik} C_i에 속한 k번째 메쏘드의 매개변수 타입
P_{ik} C_i에 속한 k번째 메쏘드의 매개변수들의 집합
r_{ik} C_i에 속한 k번째 메쏘드의 반환형 타입
R_{ik} C_i에 속한 k번째 메쏘드의 반환형의 집합

두 클래스 c_i, c_j 간의 의존관계의 정도는 클래스 c_i의 속성, 메쏘드의 매개변수, 반환형에서 타입이 c_j에 속하는 것들의 합을 그것들의 총수로 나눈 것이 된다. 그리고 이 값의 범위는 0에서 1사이가 된다.

이제 클래스를 노드로 의존 관계를 에지로 하는 방향 그래프를 그릴 수 있다 다음과 같은 인터페이스를 가지는 클래스들의 관계를 그래프로 그려보자.

```
class C1 {
private'
  C2 X, C3 Y,

public
  C2 method1(C2 P1, C3 P2);
    .
},
class C2 {
private
  C3 A1;
   ..
public
  void M1(C3 P1, int P2),
   ....
},
class C3 {
   ....
};
```

예제 1

그 결과는 그림 2와 같이 된다

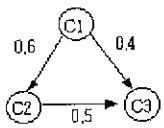

그림 2 예제 1의
의존관계 그래프

3.2 재사용 컴포넌트 후보 추출

위와 같은 의존관계 그래프로 재사용 컴포넌트의 후보(candidate)를 찾아낼 수 있다 재사용생산 컴포넌트의 후보는 다음과 같은 규칙들을 만족해야 한다

규칙 1. 상속 트리 내의 하부에 위치하는 클래스는 선조 클래스와 떨어져서는 재사용 컴포넌트에 포함될 수 없다

규칙 2. 재사용 컴포넌트 내의 모든 클래스들은 컴포넌트 밖의 클래스에 의존해서는 안된다

위의 두 조건을 고려하여 그림 3의 그래프를 살펴보자

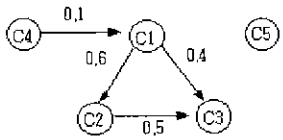

그림 3 의존관계 그래프의 예

그림 3과 같은 경우 {C1, C2, C3}, {C2, C3}, {C1, C2, C3, C4}, {C3} 모두 재사용 컴포넌트의 후보가 될 수 있다. 이 때 살펴볼 수 있는 것이 의존 정도를 나타내는 에지의 값이다.

재사용되는 컴포넌트는 개념적인 하나의 단위이다. 그 내부는 단일한 하나의 영역을 나타내고 그 내부의 응집도는 높아야 할 것이다. 그래프의 에지 값은 클래스 간의 결합도를 나타내지만 관점을 클래스의 집합인 컴포넌트로 바꾸면 그 값은 컴포넌트 내의 응집도를 나타내는 값이 된다

컴포넌트 내의 응집도를 나타내는 값은 컴포넌트 내에 포함된 에지 값의 합을 컴포넌트 내의 클래스 개수로 나눈 것으로 정의할 수 있다. 그 값을 그림 3의 경우에 적용시켜 보면 {C1, C2, C3}의 경우 0.5, {C2, C3}의 경우 0.25, {C1, C2, C3, C4}의 경우 0.4, {C3}의 경우 0으로 {C1, C2, C3}로 구성되는 컴포넌트가 재사용 컴포넌트의 후보로 가장 적합하다는 것을 알 수 있다. 그리고 {C3}의 경우 0이 나온 것은 클래스 하나로 이루어진 컴포넌트는 의미가 없음을 나타낸다고 볼 수 있다

그리고 에지 값은 재사용 컴포넌트를 만들기 위해 수정할 부분을 찾는데도 도움을 준다 작은 값의 에지는 의존도가 낮은 것을 의미하므로 소프트웨어 개발자는 그 부분을 수정하여 의존을 없애 재사용 컴포넌트를 만들 수 있다

4. 결론 및 연구 과제

기존의 결합도를 측정하는 방법을 변형하여 두 클래스간의 의존 정도를 측정하고 클래스를 노드로, 측정한 의존 정도를 에지 값으로 하는 방향 그래프를 그리고 그 그래프를 클러스터링을 하여 재사용 컴포넌트의 후보를 얻을 수 있다 이 방법을 재공학에 적용하여 새로운 소프트웨어 개발 시 많은 비용을 절감할 수 있다.그러나 아직까지 세밀한 부분까지 연구가 진행되지 않아 연결 방식에 따른 의존도의 세기와 클래스 사이의 결합도와 컴포넌트의 응집도 사이의 연구가 좀 더 수행되어야 한다. 앞으로 이 부분을 보완하여 이 방법을 실제로 적용한 실험을 수행하여 검증을 할 것이다.

참고문헌

[1] Judith Barnard, A new reusability metric for object-oriented software, *Software Quality Journal*. Vol. 7, No 1, March 1998, pp 35- 49

[2] Lionel C Briand, John W Daly, and Jurgen K Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Trans. Software Eng.*, vol. 25, no. 1, January 1999, pp 91-121

[3] A. Cimitile and G Visaggio, "Software salvaging and the call dominance tree," *Journal of Systems and Software*, Febrary 1995

[4] Brian Henderson-Sellers and J. M Verner, "Generalization of object-oriented components for reuse Measurement of effort and size change," *Journal of object-oriented programming*, May 1996

[5] Margaretha W. Price and Steven A Demurjian, "Analyzing and Measuring Reusability in Object-Oriented Designs," *Proceedings of the OOPSLA*, 1997, pp 22- 33.