객체 지향 프로그램에서 클래스 재사용성 측정 모델링

윤 희 환† · 구 연 설††

요 약

소프트웨어를 개발하는데 기존의 개발된 시스템의 컴포넌트를 재사용하면, 생산성 향상과 신뢰성 향상, 생산 원가를 절감할 수 있다. 그러기 위해서 기존의 소프트웨어 시스템에서 재사용성이 높은 소프트웨어 부품을 추출하는 것이 매우 중요하다. 클래스의 재사용은 수정을 한 후 재사용하는 화이트박스 재사용과 수정 없이 재사용하는 블랙박스 재사용으로 나눌 수 있다. 또한 클래스는 절차 언어 측면과 객체지향 언어 측면을 가지고 있으므로 이들을 모두 고려하여 재사용성을 측정하여야 한다.

블랙박스 재사용 경우는 클래스의 독립성이 중요한 품질 기준이 된다. 클래스의 독립성은 정보은닉, 클래스 사이의 결합도, 응집도 등으로 정량화 할 수 있다. 화이트박스 재사용은 수정성이 중요한 품질 기준이 된다. 수정성은 클래스의 복잡도, 상속도, 결합도, 응집도 등에 의해 정량화 할 수 있다.

본 논문에서는 객체지향 프로그램에서 클래스 재사용 측정 모델과 측정 기준을 제안한다. 제안된 모델을 사용하여 측정된 클래스는 수정을 통한 재사용이 유리한지 수정없는 재사용이 유리한지를 판단할 수 있다.

Modeling for Measurement of Class Reusability in Object-Oriented Programs

Hee-Whan Yoon† · Yeon-Seol Koo††

ABSTRACT

The reuse of software components from existing software system enhances productivity and reliability, decreases the cost of production in software development. The extraction of software components with high reusability from existing software system is very important in software reuse. The reuse of a class is classified into white-box reuse to reuse with modification and black-box reuse to reuse without modification. A class has the property of procedural language and object-oriented language. Therefore, it must measure reusability in consideration of two properties.

In black-box reuse, independence of class is important quality. It can quantify through information hiding, coupling between objects, cohesion, etc. In white-box reuse, modification is the best important quality. It can quantify through class complexity, coupling, cohesion, documentation, etc.

We propose a new model for measurement of class reusability and the measure criteria in object-oriented program. A class that is measured by proposed model can judge whether the reuse with modification has the advantage or the reuse without modification has the advantage.

1. 서 론

소프트웨어 재사용은 이미 개발된 소프트웨어에서 공통적으로 이용된 부분들을 표준화하고 이들을 새로운 소프트웨어 개발 과정에서 재사용 함으로써 소프트웨어 개발 기간을 단축시키고 소프트웨어의 생산성과 품질 향상, 유지보수 비용과 테스트 비용을 절감할 수 있다[1,2,3]. 재사용 가능한 부품은 기존의 소프트웨어 시

† 정 회 원 : 원주대학 사무자동화과 교수
†† 정 회 원 : 충북대학교 자연과학대 컴퓨터과학과 교수
논문접수 : 1998년 2월 16일, 심사완료 : 1998년 12월 12일

스템에서 추출하는 것이 쉽고 빠른 방법이다[4]. 기존의 소프트웨어 시스템의 부품들은 재사용성이 높은 부품과 그렇지 못한 부품들이 있다. 따라서 소프트웨어 개발자가 재사용 가능한 부품을 효율적으로 재사용하기 위해서 부품의 재사용성을 측정하여 적합 여부를 파악하여야 한다. 왜냐하면 재사용성이 높은 부품은 더욱 쉽게 재사용이 가능하고 그렇지 못한 경우에는 부품의 재사용이 오히려 부품을 새로 개발하는 것보다 비용이 많이 들 수도 있기 때문이다.

재사용 대상은 명세서, 설계, 소스 코드 등이 있는데, 소스 코드의 재사용은 명세서나 설계 정보의 재사용에 비해 제약 조건을 가지고 있다[5]. 즉 프로그래밍 언어와 운영체제, 응용분야 등에서 한계를 가지고 있다. 그러나 현재 실제적으로 가장 많이 재사용이 이루어지고 있고, 명세서나 설계 정보는 표현 방법이 잘 정립되어 있지 않아 재사용성 측정이 어렵기 때문에 본 연구에서는 소스 코드 수준으로 객체 지향 언어의 클래스를 재사용 단위로 하였다. 객체 지향 언어의 클래스의 재사용성 측정은 품질 척도에 의해 측정되어 평가된다. 클래스는 대부분 절차 언어 측면과 객체 지향 언어 측면 모두를 가지고 있으므로 절차 언어(procedural language)의 품질 척도와 객체 지향 언어의 품질 척도를 모두 고려하여 재사용성을 측정하여야 한다. 현재 소프트웨어소프트웨어소프트웨어 시스템 자체의 품질 평가는 많이 연구되고 있지만 재사용 가능한 부품인 클래스의 재사용성 측정에 대한 정량적 품질 평가에 대한 연구는 활발하지 못하다.

소프트웨어 재사용은 합성 중심 방법(Composition-based)과 패튼에 의한 생성 방법(Generation-based)으로 나눌 수 있다[5]. 합성 중심 방법은 독립적인 소프트웨어 부품을 블록과 같이 조립하여 새로운 시스템을 구성하는 방법이다. 이 방법은 부품 안에서 일어나는 세부 내용의 변경 없이 재사용하는 블랙박스 재사용(Black-box reuse)이 많다. 반면에 패튼에 의한 재사용은 부품을 유연하게 정의하여 파라메터 값을 주어 적절히 변경을 가하여 재사용하는 방법이다. 이 방법은 부품의 내용을 자세히 알아야 하는 화이트박스 재사용(White-box resue)이다. 재사용 부품은 블랙박스 재사용과 화이트박스 재사용의 경우를 모두 고려하여 재사용성을 측정하여야 한다.

본 논문에서는 객체 지향 언어에서 재사용 단위인 클래스의 블랙박스 재사용과 화이트박스 재사용의 경우를 모두 고려한 재사용성 측정 모델을 제안한다.

2. 관련연구

소프트웨어의 재사용성을 측정하는 연구가 많이 있어 왔다. Prieto Diaz와 Freeman[4]은 라이브러리 시스템(library system)을 구축하였고, 그 내부에 있는 평가 시스템이 검색된 각 소프트웨어 부품을 정규화하고, 재사용에 요구되는 노력의 정도에 따라 평가하여 순위를 정하였다. 여기서 재사용 노력의 정도를 나타내기 위해 5개의 프로그램 속성, 즉 프로그램 크기, 프로그램 구조, 문서화 정도, 프로그래밍 언어, 재사용자의 숙련도 등의 품질 속성을 사용하였다. 재사용자의 숙련도에 따라 프로그램 크기, 프로그램 구조와 문서화 정도의 기준이 변하므로 퍼지 이론을 적용하여 평가하였다. 이는 재사용자의 숙련도에 따라 품질 속성의 기준이 바뀌므로 품질의 값들을 정량적으로 측정하지 못하였다.

REBOOT(Reuse Based on Object-Oriented Techniques) 시스템[6]은 유럽의 여러 기업에 의해 4년에 걸쳐 수행된 프로젝트로서, 현재의 재사용 기술을 보급하기 위해 개발된 객체지향 재사용 시스템이다. REBOOT 시스템에서 재사용 부품에 대한 품질 측정은 호환성(portability), 유연성(flexibility), 이해용이성(understandability), 적합성(confidence) 등의 척도를 사용하였다. CARE 시스템[2]은 C언어를 지원하는 시스템으로 재사용 속성 모형(reusability attribute model)을 제안하였다. 여기서 소프트웨어 부품의 재사용성을 측정하기 위해 4개의 척도(metrics) 즉, Halstead의 소프트웨어 과학(software science)에 의한 모듈의 크기(volume), McCabe의 순환 복잡도(cyclomatic complexity), 실제 크기와 추정치의 차이 정도를 구하기 위한 정규성(regularity), 재사용 빈도를 사용하였다. 이 방법은 프로그램의 복잡도를 측정하는데 사용되는 척도를 위주로 함으로써 재사용성의 측정을 복잡도 측정으로 한정짓는 결과를 가져왔으며[7], 정량적으로 품질의 값을 추출할 수 있지만 절차언어의 속성에만 치중하여 객체 지향 언어의 속성을 반영하지 못하였다[8].

3. 소프트웨어의 품질 척도(metrics)

3.1 절차 언어의 품질 척도

소프트웨어의 품질은 소프트웨어 생산물의 품질을

표현하고 평가할 수 있는 속성들이다. 소프트웨어 부품의 품질을 평가함에 있어서 이를 정량적으로 측정할 수 있는 척도가 중요하다. 척도란 소프트웨어가 보유하고 있는 특성, 품질, 속성의 크기나 정도의 계량적인 표현을 말한다. 절차 언어에 대해 지금까지 여러 정량화된 품질 평가 도구와 척도들이 정의되어 왔으며 주요한 연구 성과는 <표 1>과 같다.

〈표 1〉 절차 언어의 주요 품질 척도
〈Table 1〉 Quality metrics in procedural language

항 목	척 도
복잡도	LOC MaCabe의 Cyclomatic Complexity Halstead의 Software Science
모듈성	Yourdon, Constantine의 Fan-in, Fan-out Coupling Cohesion
문서화	주석(comments)

LOC(Line Of Code)는 소프트웨어를 개발하는데 사용된 비용과 노력을 측정하는 크기 중심의 직접적인 측정 방법으로 이용하기는 쉬우나, 프로그래밍 언어에 종속적이고 비절차적 언어들을 쉽게 수용할 수 없다[9].

McCabe의 순환 복잡도(cyclomatic complexity)는 프로그램의 제어 구조를 측정하는 대표적인 척도로서 프로그램 수행경로를 프로그램 라인으로 표시하는 노드와 경로를 표시하는 간선으로 구성된 그래프를 나타낸 후 V(G) = e−n+2 (e = edges의 수, n = nodes의 수) 공식에 의해 값을 구한다. 복잡도가 5~10인 경우에 매우 구조적이며 안정된 프로그램으로 평가된다.

Halstead의 소프트웨어 과학은 프로그램을 연산자와 피연산자라고 하는 독립된 원소들의 집합으로 정의한다. 이는 프로그래밍 언어와는 무관하게 동사격인 연산자와 명사격인 피연산자의 숫자를 계산하여 프로그램 내에 내재된 논리의 규모를 측정하는 것이다. 소프트웨어 과학은 프로그램의 전체 길이, 프로그램의 크기 등의 수식을 개발하였으며, 여기에 사용된 4가지 계수는 다음과 같다.

n_1 - 프로그램에 나타나는 서로 다른 연산자의 수
n_2 - 프로그램에 나타나는 서로 다른 오퍼랜드의 수
N_1 - 프로그램에 나타나는 연산자의 전체 수
N_2 - 프로그램에 나타나는 오퍼랜드의 전체 수

Halstead는 4가지 계수를 사용하여 프로그램의 전체 길이(N), 프로그램의 크기(V) 등의 수식을 정의하였다.

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$
$$V = N \log_2 (n_1 + n_2)$$

프로그램의 크기 V는 프로그램을 기술하는데 요구되는 정보의 양을 나타낸다. Halstead의 연구는 소프트웨어 과학을 조사하기 위해 행해졌던 실험적인 검증으로 높이 평가되고 있다. 응집도(cohesion)는 모듈에 있는 기능적 응집력을 측정하는 것으로 프로그램의 다른 부분에서 실행되고 있는 절차들과 거의 상호작용을 요구하지 않으면서 소프트웨어 모듈 내에서 단일 작업 수행하는 것이 이상적이다. 제어폭(fan-out)은 어떤 모듈에 직접 종속되어 있는 모듈의 개수를 의미하며, 제어폭이 과다하면 복잡도를 증가시키는 요인이 된다. 적정한 제어폭은 7±2개이다. 공유도(fan-in)는 특정 모듈이 상위 모듈로부터 호출될 때, 그 특정 모듈을 호출하는 상위 모듈의 개수를 의미한다. 일반적으로 공유도가 높으면 모듈 분해가 잘된 것으로 유지보수를 용이하게 한다.

결합도(Coupling)는 모듈간의 상호 의존성을 측정하는 방법으로 가능한 최하의 결합도를 갖도록 해야 한다. 그렇게 함으로써 파급효과를 줄일 수 있다. Fan-in, Fan-out, 응집도와 결합도는 모듈의 독립성을 측정하는 중요한 척도이다. 독립적인 모듈은 설계와 코드 속성으로 인한 부작용의 파급효과를 제한하고, 재사용 모듈이 가능하기 때문에 유지보수와 시험도 더욱 쉽게한다. 문서화 정도는 주석의 비율로 측정 가능하며, 프로그램의 각 부분의 정확한 기능, 사용법과 인터페이스를 사용자에게 쉽게 이해하게 해 준다. 원시코드 대 주석의 비율은 50 : 50 이상이 좋다[10].

3.2 객체 지향 언어의 품질 척도

객체 지향 기술을 사용한 소프트웨어 시스템은 전통적인 방법과 많은 차이가 있다. 첫째로 재사용과 품질이 강조된 반복적 생명 주기를 갖고, 소프트웨어 구조도 매우 다르다. 둘째로 추상화 수준이 높고, 상속관계가 소프트웨어 구조에 중요한 영향을 미친다. 따라서 절차 언어에 대한 품질 척도를 객체지향 언어에 적용하는 것이 무리이다. 객체지향 언어의 기본 척도에 대한 연

구는 <표 2>에 나타나 있으며, 각 연구는 근본 목표에 따라 제안된 척도도 다르다.

3.1절과 3.2절에서 각각 절차 언어와 객체 지향 언어 품질 척도에 관한 주요한 연구에 대해 살펴보면 절차 언어의 품질들은 객체지향 언어의 품질 속성들을 반영하지 못하였고, 객체지향 언어의 품질 척도들은 이론에 너무 치우쳐 있고 정량적 기준을 제시하는데 미흡하였다. 본 논문에서 재사용단위인 클래스는 객체지향 언어의 속성과 절차 언어의 속성을 모두 포함하고 있으므로 양쪽 속성을 모두 고려하여 블랙박스 재사용과 화이트박스 재사용에 대한 재사용성 모델을 제안하고자 한다.

〈표 2〉 객체지향 언어 관련 척도들
〈Table 2〉 Quality metrics related with object-oriented languages

연구자	근본 목표	제안된 척도
Chidamber, Kemmerer[11]	설계 규모와 복잡도	클래스당 가중치를 가진 메소드 수 상속 트리의 깊이 자식 노드의 수 객체간의 결합도 클래스에 대한 반응도 응집력 결여도
Jenson, Bartley[12]	소프트웨어 개발 노력	객체의 수 연산의 수 인터페이스의 수 Man Hours = f(Objects, Operations, Interfaces)
Dumke, Neumann, Stoeffler[13]	시스템, 클래스, 메소드 수준별 복잡도	1. 시스템 수준 클래스의 수, 계층 구조의 깊이 및 너비, 평균 서브 클래스의 수, 평균 메소드의 수, 평균 변수의 수 2. 클래스 수준 클래스 메소드의 수, 인스턴스 메소드의 수, 클래스 변수의 수 3. 메소드 수준 클래스의 라인 수, MaCabe의 Cyclomatic Number 4. 객체지향적 요인 상속 복잡도, 통신 척도, 클래스 계층 구조의 깊이, 재사용 정도, 포괄성 척도, 다형성 척도
Moreau, Dominick[14]	복잡도	객체가 보낼 수 있는 메시지 수 객체가 응답하는 메시지 수 상속 복잡도
Bieman, Karunanithi[15]	재사용성	Client와 Server 입장에서 각각 Verbatim, Generic과 Leveraged reuse 척도 제안

4. 재사용성 측정 모델

앞에서 언급한 바와 같이 재사용 방법에는 블랙박스 재사용과 화이트박스 재사용이 있다. 화이트박스 재사용은 소프트웨어 부품을 재사용 하고자 하는 요구사항에 맞게 수정한 후 재사용 하므로 수정성이 재사용성 측정의 중요한 품질 요소가 된다. 왜냐하면 수정에 드는 노력과 비용이 새로운 부품을 만드는 것보다 더 많이 든다면 재사용할 필요가 없기 때문이다. 따라서 후보자 부품들의 수정성을 측정하여 수정성이 높은 부품을 선택하는 것이 재사용을 쉽고 빠르게 한다.

블랙박스 재사용은 수정없이 부품을 재사용하는 경우로 클래스의 독립성과 정보은닉이 재사용성 측정의 중요한 품질 요소이다. 독립성은 정보은닉과 결합도, 응집도 등이 중요하다. 정보은닉이 잘되어 있을수록 추상화가 잘되어 있어 해당 부품의 세부사항을 알지 않고도 그 부품을 쉽게 재사용 할 수 있다[16]. Rising[17]에 의하면 모듈 단계의 정보은닉을 측정하는 2가지 요소 즉, 첫째는 모듈이 하나의 설계결정을 가지고 있는지에 대한 것이고, 둘째는 모듈의 인터페이스 내에 설계결정에서 필요없는 개체들이 최소화 되어 있는지에 대한 것이다. 이것으로 객체 기반 언어(object-based language)에 적용할 수 있는 모듈단계의 정보은닉 척도를 다음과 같이 제안하였다.

Information Hiding = 2 * [0 | 1] + Number of visible variables

첫째 항에서 하나의 설계결정이 모듈 내에서 캡슐화(encapsulation)가 되어 있으면 0이고 그렇지 않으면 1이 된다. 둘째 항은 전역 변수의 개수를 의미한다. 이 식의 첫 번째 항은 객관적이지 못하며, 너무 이론에 치우쳐 있다.

정보은닉을 높이기 위해서는 부품의 높은 응집도, 낮은 결합도, 적은 매개 변수가 필요하다. 만약 소프트웨어 부품이 매우 복잡하고 어려워 수정성이 나쁘더라도 독립성이 좋으면 그 부품은 요구사항만 만족하면 블랙박스 재사용은 용이하다. 이상과 같이 블랙박스 재사용과 화이트박스 재사용에서 재사용성의 품질 기준이 서로 다르므로 소프트웨어 부품의 재사용성을 측정할 때 두 가지 경우를 구별하여 측정해야 한다. 즉 소프트웨어 부품이 재사용에 필요한 요구 조건을 만족하는 경우에는 블랙박스 재사용이 되고, 요구조건을 만족하는 부품이 없는 경우에는 가장 유사한 부품을 선택하여

변경한 후 사용하는 화이트박스 재사용이 된다. 그러므로 하나의 부품에 대해서 두 가지 방법 중 하나만 만족하더라도 재사용이 가능하므로 재사용성 측정에서도 두 가지 재사용성을 함께 각각 측정해야 한다. 또한 하나의 클래스는 객체들의 속성과 메소드(멤버함수)들을 정의한 것이기 때문에 절차 언어와 객체 지향 언어의 특성을 모두 가지고 있으므로 각각의 품질 척도를 모두 적용하여 재사용성을 측정한다.

4.1 화이트박스 재사용성(white-box reusability) 측정 모델

앞에서 논의된 바와 같이 화이트박스 재사용성의 중요한 품질 척도는 수정성이다. 화이트박스 재사용성을 측정하는데 필요한 척도는 다음과 같다.

WBR 척도 : CC, CBO, DIT, DOC
CC 측정에 필요한 척도 : WMC, LCOM, RFC
여기서 WBR = 화이트박스 재사용성(White-Box Reusability)
CC = 클래스의 복잡도
WMC = 클래스 당 가중치를 가진 메소드 수
LCOM = 응집력의 결여도
RFC = 클래스에 대한 반응도
CBO = 객체간의 결합도
DIT = 상속 트리의 깊이
DOC = 내부 문서화 정도(주석)

재사용 소프트웨어 부품의 복잡도가 높으면 이해성이 낮아지므로 수정하는데 그만큼 어려움이 따른다. 클래스의 복잡도는 클래스 자체의 복잡도와 각 클래스의 맴버 함수의 복잡도가 있다. 클래스에 선언된 메소드의 수(WMC)가 많을수록 하위 상속 클래스에 영향을 많이 주므로 재사용을 어렵게 한다. RFC는 클래스의 반응정도를 나타내는 것으로 클래스 안에 선언된 메소드의 수와 이를 메소드에서 직접 호출된 다른 메소드 개수의 합이다. 이것은 클래스 안에 선언된 메소드 수와 다른 클래스와의 메시지 교환을 합하여 클래스의 복잡도를 나타낸다. 따라서 RFC의 값이 큰 클래스는 메시지에 반응하여야 하는 의무가 크기 때문에 클래스를 이해하기 위해서는 많은 노력이 필요하게 된다. LCOM은 메소드들이 사용하는 클래스의 속성들을 분석하여 그 사이의 유사도를 측정하는 것으로 유사도가 크며 응집력이 높은 것이다. 따라서 LCOM의 값이 크면 이해가 어렵고 오류의 가능성이 많으므로 클래스를 분리하는 것이 좋다. 또한 클래스 안의 메소드의 응집도가 크면 해당 모듈의 캡슐화가 좋다는 의미이고, 수정을 할 경우 부품 내부에서의 파급효과도 적어진다. 객체 사이의 결합도는 클래스 사이의 상호작용을 나타내는 척도로 상속관계가 없는 다른 클래스의 메소드나 속성을 사용하여 결합하는 정도이다. 결합도가 높으면 설계의 모듈화가 나쁘고, 수정을 할 경우 부작용의 파급효과가 크기 때문에 재사용을 어렵게 한다. 클래스의 상속도는 클래스에서 상속관계를 측정한다. DIT는 상속 트리의 깊이로 값이 클수록 상속되는 속성과 메소드가 많아지므로 복잡하다는 것을 의미한다.

DOC는 내부 문서화 정도를 나타내며, 주석으로서 측정한다. 코드의 의미를 설명하는 주석이 많을수록 코드의 가독성을 높이고, 아울러 이해성도 높아지므로 부품의 수정성을 좋게 한다.

4.2 블랙박스 재사용성(black-box reusability) 측정 모델

앞에 논의한 바와 같이 블랙박스 재사용성은 클래스의 독립성과 정보은닉이 중요한 품질 요소이다. 블랙박스 재사용성을 측정하는데 필요한 척도는 다음과 같다.

BBR 척도 : CI, IH, PA
CI 측정에 필요한 척도 : CBO, LCOM
여기서 BBR = 블랙박스 재사용성(Black-Box Reusability)
IH = 정보은닉 정도
PA = 매개변수(parameter)
CBO = 객체 사이의 결합도
LCOM = 응집력의 결여도

독립성은 다른 모듈과 과다한 상호작용을 피하면서 동시에 하나의 기능만을 갖게 하는 특성을 나타내고 있다. 독립성이 높은 모듈은 기능이 구분되고 인터페이스가 간단해지기 때문에 재사용성을 높인다. 독립성은 두 개의 품질 평가 기준인 객체 사이의 결합도와 응집력의 결여도를 사용하여 측정한다. 정보은닉은 한 모듈내의 설계 결정을 다른 모듈들이 알지 못하게 하는 것을 의미한다. 모듈 내에서 구현된 복잡한 알고리즘, 외부와의 인터페이스에 대한 세부사항의 구현 등이 설계 결정의 실례이다.

매개변수는 소프트웨어 부품을 사용하기 위해 꼭 알아야 하는 인터페이스로 이를 최소화 해야 인터페이스 정보를 줄일 수 있고 동시에 재사용을 쉽게 할 수 있다.

응집력 결여도와 객체 사이의 결합도는 화이트박스와 블랙박스 재사용성 척도로 공통적으로 사용하고 있다. 결합도가 크면 수정에 따른 파급효과가 크므로 수정성에 영향을 미치고, 응집력 결여도의 값이 크면 이해가 어려워 수정에 영향을 미치므로 화이트박스 재사용성의 중요한 척도가 된다. 응집도와 결합도는 모듈성을 측정하는 척도로 추상화와 정보은닉 개념과 밀접한 관련이 있고, 이는 모듈의 독립성에도 영향을 미친다. 즉, 재사용하고자 하는 부품의 세부사항을 몰라도 그 부품을 쉽게 재사용 가능하게 한다. 그러므로 이 두 가지 척도는 블랙박스 재사용성에서도 중요한 척도가 된다.

5. 재사용성 측정 척도

4장에서 재사용성 측정을 위한 모델이 제안되었고 그에 대한 구체적인 척도에 대해 살펴보았다. 여기서는 실제 재사용성 측정을 위한 척도들의 계산방법에 대해 알아본다.

5.1 클래스의 복잡도

객체 지향 척도 연구 중에서 가장 대표적인 것이 M.I.T의 Chidamber와 Kemerer의 연구[11]인데, 이 연구는 측정 이론에 근거를 두고 척도 집합을 정의하였기 때문에 신뢰성이 높다. 따라서 본 논문에서도 재사용성에 영향을 미치는 주요한 Chidamber와 Kemerer의 WMC, RFC, LCOM, CBO, DIT 등의 척도를 사용하여 측정하고, 그 외의 척도들은 별도로 정의하여 측정한다.

(1) WMC(Weighted Methods per Class)

클래스 C에 메소드 $M_1, M_2, \cdots, M_n$이 존재한다. 이때 $c_1, c_2, \cdots, c_n$을 메소드의 복잡도라 하면

$$WMC = \sum_{i=1}^{n} c_i$$

이다. 여기서 메소드의 복잡도는 순환 복잡도로 구한다. 메소드의 수와 각각의 메소드가 가지는 복잡도는 클래스의 수정에 소요되는 노력과 시간을 알려주는 지시자 역할을 한다.

(2) RFC(Response For a Class)

클래스가 호출하는 메소드 수이다.

$RFC = |RS|$

$RS = \{M\} \cup all_i \{R_i\}$

$\{R_i\}$ = 메소드 i에 의해 호출되는 메소드 집합

$\{M\}$ = 클래스의 메소드 집합

호출되는 메소드가 많을수록 클래스의 복잡도는 증가하며, 이것은 수정을 어렵게 한다.

(3) LCOM(Lack of COhesion in Method)

하나의 클래스에서 임의의 메소드 M_i가 사용하는 인스턴스 변수들의 집합을 (I_i)라하면 이 클래스에는 $M_1, \cdots, M_i, \cdots, M_n$에 대해 n개의 $(I_1), \cdots, (I_i), \cdots, (I_n)$이 존재한다. 여기서 LCOM은 n개 집합들의 교집합으로 구성된 집합의 개수이다.

$P = \{(I_i, I_j) \mid I_i \cap I_j = \varnothing\}$

$Q = \{(I_i, I_j) \mid I_i \cap I_j \neq \varnothing\}$

$LCOM = |P| - |Q|$ 단, $|P| > |Q|$ 경우

$= 0$ 그렇지 않은 경우

5.2 CBO(Coupling Between Objects)

하나의 클래스가 다른 클래스의 메소드나 속성을 사용하여 결합한 수이다. 클래스 사이의 과도한 결합도는 모듈성을 해치고, 프로그램의 이해를 어렵게 하므로 재사용성이 낮아진다.

5.3 DIT(Depth of Inheritance Tree)

클래스에 있어서 상속의 깊이를 클래스의 DIT라 한다. 즉, DIT는 상속트리에서 루트 노드로부터의 최대 길이이다.

상속이 깊은 트리는 많은 메소드들과 클래스들을 포함하기 때문에 복잡도가 높아진다.

5.4 문서화 정도

재사용 가능한 소프트웨어 부품의 정확한 기능, 사용법과 인터페이스를 사용자가 정확하게 아는 것이 중요하다. 이것은 내부 문서화 정도로 코드에 대한 주석의 비율로 구할 수 있으며, 비율이 높을수록 소프트웨어 부품에 대한 정보를 많이 알 수 있으므로 이해하는데 도움을 주고 아울러 재사용을 쉽게 한다. 주석 비율은 50 : 50이상이면 좋다[10].

$$DOC = \frac{\text{클래스의 주석 라인 수}}{\text{클래스의 총 라인 수}}$$

5.5 정보은닉 정도

캡슐화란 연관된 여러 항목을 하나로 묶는 것으로 캡슐 속의 여러 항목에 관한 정보가 외부에 은닉되었다고는 보지 않는다. 정보은닉은 캡슐 속에 쌓여진 여러 항목들에 대한 정보를 외부에 감추는 것을 의미한다. 그러므로 캡슐화되고 정보은닉된 객체는 하나의 블랙박스가 된다. 클래스에서 외부에 공개하고자 하는 정보들은 공개 인터페이스(public interface)로 정의해 외부의 객체들이 이 인터페이스를 통해 정보를 교환한다. 즉 공개된 정보가 많을수록 정보은닉 정도는 낮아진다.

한 클래스의 정보은닉 정도는 클래스 내의 전체 인스턴스 변수 및 메소드들의 수에 대한 정보은닉된 인스턴스 변수와 메소드의 비율로 구하였다.

$$IH = \frac{\text{정보은닉된 인스턴스 변수및 메소드의 총 액세스된 수}}{\text{인스턴스 변수및 메소드의 총 액세스된 수}}$$

5.6 매개변수

클래스 내의 메소드들이 매개변수를 많이 가지고 있으면 알아야 할 정보가 그만큼 많아진다. 따라서 각 메소드의 매개변수의 수가 적을수록 재사용을 쉽게 한다. 매개변수를 계산하는 방법은 다음과 같다.

$$PA = (\sum_{i=0}^{n} M_i)/n$$

여기서 Mi는 i번째 메소드의 매개변수 개수이고, n은 메소드의 개수이다.

모듈에서 매개변수의 수가 몇 개가 적당한지 그 기준이 없으므로 메소드들의 매개변수의 평균을 구하여 적을수록 재사용에 유리하다고 할 수 있다.

6. 측정 예

기존의 소프트웨어 시스템에서 재사용성을 측정한 후 이 결과가 재사용성이 높은지 낮은지를 판단할 수 있는 기준이 필요하다. <표 3>은 척도의 선택 기준을 나타낸다.

McCabe는 경험적 측정을 통해 순환 복잡도의 상한값을 10으로 제시하였다[9]. 그리고 한 클래스의 멤버 함수의 수는 7±2가 좋으므로[8] WMC는 70(10×7)으로 정하였다. RFC, CBO와 LCOM의 기준값은 [11]에 의해 측정된 결과 C++의 경우 각각 6, 0과 0이 중간값으로 측정되어 이것을 기준으로 정하였다. DIT는 6이하가 바람직하나 재사용성 입장에서 볼 때 3보다 크면 이해하기가 어렵고 복잡해지므로 재사용성을 해치기 쉽다. 그래서 기준값을 3이하로 정하였다.

〈표 3〉 각 척도의 권장 선택 기준
〈Table 3〉 Recommended criteria of metrics

척도	WMC	RFC	LCOM	CBO	DIT	DOC	PA
기준치	70이하	6이하	0	0	3이하	0.4이상	3이하

DOC는 주석 대 코드비율을 계산할 때, 주석 라인 수를 주석이 아닌 라인 수와 나누어서 구하는데 C 프로그램의 경우 보통은 1.0이상이면 좋고 최소한 0.8이상 되어야 한다. 이것은 본 논문의 계산 방법에 따라 약 0.4가 되므로 0.4 이상을 최소 기준으로 정하였다. 매개 변수는 [7]에서 측정한 결과 평균 2.44개이므로 이를 올림하여 3이하로 정하였다.

〈표 4〉 클래스의 재사용성 측정
〈Table 4〉 Measurement of class reusability

재사용성 / 척 도	CLASS A		CLASS B	
	WBR	BBR	WBR	BBR
WMC	29		10	
RFC	24		8	
LCOM	46	46	8	8
CBO	1	1	1	1
DIT	1		2	
DOC	0.4		0.4	
IH		0.95		0.8
PA		0.86		0.75

[18]에서 C++를 사용하여 CASE 도구를 구현하였는데 여기서 작성된 클래스 "Entry"와 "FunctionEntry"를 추출하여 화이트박스 재사용성과 블랙박스 재사용성을 측정하였다. <표 4>는 5장에서 제안한 척도를 사용하여 화이트박스 재사용성과 블랙박스 재사용성을 측정한 결과이다. <표 4>에서 클래스 A는 "Entry" 클래스, 클래스 B는 "FunctionEntry" 클래스를 의미한다. 클래

스 A는 11개의 인스턴스 변수와 23개의 메소드를 가지고 있고, 클래스 B는 1개의 인스턴스 변수와 8개의 메소드로 되어있다. 클래스 A와 B 모두 WMC의 수치가 낮게 나온 것은 메소드의 논리 구조가 지극히 단순하기 때문이다. RFC가 기준치보다 높은 것은 메소드의 수가 많은 결과이고 클래스 A의 LCOM이 높은 것으로 보아 클래스를 2개 이상으로 나누어 설계하는 것이 응집도를 높이고 아울러 재사용성을 높일 수 있을 것이다. 문서화 정도는 파일 단위로 계산한 결과를 각 클래스에 적용하였으므로 두 개의 클래스가 같은 수치인 0.4로 기준을 만족한다.

정보은닉 정도는 두개 클래스 모두 높게 나왔으며 인터페이스의 정보를 알려주는 PA는 평균 1 미만으로서 각 메소드의 매개변수가 1개정도 사용된 것을 의미한다. 이상으로 보아 클래스 A의 경우 화이트박스 재사용성은 낮으나 블랙박스 재사용은 크게 문제가 없다고 볼 수 있다. 클래스 B는 일부 척도에서 기준치를 초과하였으나 재사용성에는 크게 어려움이 없다고 볼 수 있다.

7. 결 론

본 논문에서 객체지향 프로그램으로 작성된 소프트웨어 시스템에서 재사용 가능한 부품인 클래스에 대해 재사용성을 측정하기 위한 모델을 제안하였다. 부품을 재사용하는 방법에는 화이트박스 재사용과 블랙박스 재사용이 있는데 이 두 가지 방법은 서로 다른 성질을 갖는다. 임의의 클래스가 화이트박스 재사용에는 부적합하나 블랙박스 재사용에는 적합할 수 있으며, 그 반대의 경우도 있기 때문이다. 따라서 제안된 모델에서는 두 가지를 모두 측정하여 상대적인 비교가 가능하며, 측정 예를 통하여 정량화가 가능함을 보여 주었다. 즉 재사용 가능한 부품인 클래스들을 품질 척도에 따라 측정하여 높은 품질의 클래스를 선택할 수 있게 함으로써 재사용 비용과 시간을 줄일 수 있다. 제안된 모델은 소프트웨어 품질을 나타내는 여러 가지 척도들로 이루어져 있으며 각각의 척도들은 서로 다른 가중치를 가질 수 있을 것이다. 이는 재사용성을 측정하는 척도 중에서 재사용성에 미치는 영향이 서로 다르기 때문에 그에 알맞은 가중치를 가질 수 있으며, 풍부한 사례연구와 축적된 자료를 토대로 앞으로 연구되어야 할 부분이다.

또한 소프트웨어 시스템을 개발하는데 제안된 모델을 이용하여 재사용이 높은 부품을 이용하여 개발해 봄으로써 이 모델의 문제점과 장단점을 분석하여 모델의 신뢰성을 높이는 작업을 계속해야 할 것이며, 재사용 가능한 클래스의 품질을 정량화하고 관련 정보를 자동적으로 분석하고 평가하여 재사용을 지원하는 자동화 도구의 개발이 필요하다.

참 고 문 헌

[1] C. McClure, The Three R's of Software Automation, Prentice Hall, Inc., 1992.

[2] G. Caldiera & R. Basili, "Identifying and Qualifying reusable Software components," IEEE Computer, pp.61-70, Feb. 1991.

[3] C. W. Krueger, "Software Reuse," ACM Computing Surveys, Vol.24, No.2, pp.131-184, June 1992.

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

[5] J. T. Biggerstaff & C. Richter, "Reusability Framework, Assessment and Directions," IEEE Software, pp.41-49, Mar. 1987.

[6] G. Sindre, R. Conradi, "The REBOOT Approach to Software Reuse," The Journal of Systems and Software, Vol.20, pp.201-212, 1995.

[7] 김형섭, 배두환, "수정 및 무수정을 통한 코드 재사용성 측정 모델링", 정보과학회 논문지(B), 24권, 5호, pp.561-575, 1997년 5월.

[8] 김재생, 송영재, "재사용가능한 클래스 후보자들의 품질 메트릭들에 관한 연구", 정보처리 학회 논문지, 4권, 1호, pp.137-117, 1997년 1월.

[9] R. S. Pressman, Software Engineering : A Practitioner's Approach, 4th Ed., McGraw-Hill Companies, Inc., 1997.

[10] R. S. Arnold & W. B. Frakes, "Software Reuse and Reengineering," IEEE Software, pp.476-483, 1991.

[11] S. R. Chidamber & C. F. Kemerer, "A Metrics suite for Object Oriented Design," IEEE Trans. on Software Engineering, Vol.20, No.6, pp.476-493, 1994.

[12] R. L. Jenson & J. W. Bartley, "Parametric Estimation of Programming Effort : An Object Oriented

Model," The Journal of Systems and Software, Vol.15, No.2, pp.107-114, May 1991.

[13] R. Dumke, K. Neumann & K. Stoeffler, "The Metric Based Compiler - A Concurrent Requirement," ACM SIGPLAN Notes, Vol.27, No.12, pp. 29-38, Dec. 1992.

[14] D. R. Moreau & W. D. Dominick, "Object Oriented Graphical Information Systems : Research Plan and Evaluation Metrics," The Journal of Systems and Software, Vol.10, No.1, pp.23-28, 1989.

[15] J. M. Bieman & Santhi Karunanithi, "Measurement of Language-Supported Reuse in Object Oriented and Object-based Software," The Journal of Systems and Software, Vol.30, pp.271-293, 1995.

[16] D. L. Parnas, P. C. Clements and D. M. Weiss, "Enhancing Reusability with Information Hiding," Proceedings of ITT Workshop on Reusability in Programming, Sep. 1983.

[17] L. S. Rising, "An Information Hiding Metric," The Journal of Systems and Software, Vol.26, pp.211-220, 1994.

[18] D. E. Brumbaugh, Object-Oriented Development : Building CASE Tools with C++, John Wiley & Sons, Inc., 1994.

윤 회 환

e-mail : yoonhw@sky.wonju.ac.kr

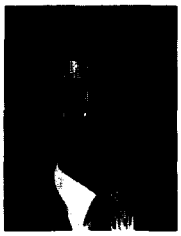
1982년 계명대학교 전자계산학과 (학사)

1984년 중앙대학교 대학원 전자계산학과(이학석사)

1994년~현재 충북대학교 대학원 전자계산학과 박사과정 재학중

1995년~현재 원주대학 사무자동화과 조교수

관심분야 : 소프트웨어공학(재사용, S/W 품질 메트릭스)



구 연 설

e-mail : yskoo@cbucc.chungbuk.ac.kr

1964년 청주대 상학과

1975년 성균관대 대학원 전자자료처리학과

1981년 동국대 대학원 통계학과(이학석사)

1988년 광운대학교 대학원 전자계산학과(이학박사)

1979년 이후 충북대전자계산소장, 한국정보과학회 부회장 역임

현재 충북대 컴퓨터과학과 교수

관심분야 : 소프트웨어공학, 정보통신 등