객체지향 프로그램의 화이트박스와 블랙박스 재사용성 측정 메트릭스

(Metrics for Measuring of White-box and Black-box Reusability in Object-Oriented Programs)

윤 희 환 † 김 영 집 †† 구 연 설 †††

(Hee Whan Yoon)(Young Jip Kim)(Yeon Seol Koo)

요 약 객체지향 프로그램에서 클래스는 수정한 후 재사용하는 화이트박스 재사용과 수정없이 재사용하는 블랙박스 재사용으로 나눌 수 있다. 컴포넌트 기반 소프트웨어 개발 방법론에서의 컴포넌트는 블랙박스 재사용 형태를 띤다. 클래스와 컴포넌트는 절차적인 특성과 객체지향적인 특성을 모두 가지고 있으므로 이를 고려하여 재사용성을 측정해야 한다.

이 논문에서는 클래스와 컴포넌트의 재사용성 측정 모델과 측정 기준을 제안한다. 제안된 모델을 사용하여 측정된 클래스는 화이트박스 재사용이 유리한지 블랙박스 재사용이 유리한지를 판단할 수 있다. 아울러 총평가점수를 산정하여 비교하므로 어느 클래스가 재사용성이 높은지를 알 수 있다.

Abstract The reuse of a class in object-oriented programs is classfied into white-box reuse to reuse with modification and black-box reuse to reuse without modification. A component in component-based development has the property of black-box reuse. In order to measure resuability of class and component, we must consider all the procedural and object-oriented attribute.

In this paper, we propose a new model for measurement of class and component reusability and the measure criteria. A class that is measured by proposed model can judge whether white-box reuse has the advantage or black-box reuse has the advantage. In addition, we can know which class is high or low in reusability as compared with the sum of estimated scores.

1. 서 론

소프트웨어 개발에 있어서 프로그램의 규모는 커지고, 복잡해지는 추세인데 반해 소프트웨어 개발 기술은 하드웨어의 발전 속도에 비해 느린 형편이다. 이로 인해 소프트웨어 개발 비용과 유지보수 비용이 증가하게 되어 소프트웨어 위기에 봉착하게 되었다. 소프트웨어 위기 문제를 해결하기 위한 대안으로 소프트웨어 재사용 기술이 소개되었다.

소프트웨어 재사용(software reuse)이란 소프트웨어를 개발할 때 모든 것을 새로 만드는 것이 아니라 기존의 개발된 소프트웨어를 이용하여 새로운 소프트웨어를 개발하는 과정이다[1]. 소프트웨어 재사용은 이미 개발된 소프트웨어에서 공통적으로 이용된 부분들을 표준화하고 이들을 새로운 소프트웨어 개발 과정에서 재사용함으로써 소프트웨어 개발 기간을 단축시키고 소프트웨어의 생산성과 품질 향상, 유지보수 비용과 테스트 비용을 절감할 수 있다[1][2][3].

재사용성(reusability)은 새로운 소프트웨어 개발에 재사용 할 객체나 부품들의 재사용하기 쉬운 정도를 말한다[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

† 종신회원 : 원주대학 사무자동화과 교수
yoonhw@sky.wonju.ac.kr
†† 정 회 원 : 숭실대학교 전자계산원 정보처리과 교수
yjkll8@soongsil.or.kr
††† 종신회원 : 충북대학교 컴퓨터과학과 교수
yskoo@cbucc.chungbuk.ac.kr
논문접수 : 2000년 3월 23일
심사완료 : 2000년 12월 11일

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 재사용이 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 메트릭에 의해 측정되어 평가된다. 클래스는 대부분 절차 언어 측면과 객체 지향 언어 측면 모두를 가지고 있으므로 절차 언어(procedural language)의 품질 메트릭과 객체 지향 언어의 품질 메트릭을 모두 고려하여 재사용성을 측정하여야 한다.

소프트웨어 재사용은 독립적인 소프트웨어 부품을 블록과 같이 조립하여 새로운 시스템을 구성하는 방법 즉 부품 안에서 일어나는 세부 내용의 변경 없이 재사용하는 블랙박스 재사용(black-box reuse)과 재사용할 소프트웨어 부품을 적절히 변경하여 재사용하는 방법 즉 부품의 내용을 자세히 알아야 하는 화이트박스 재사용(white-box resue)이 있다. 재사용 부품은 블랙박스 재사용과 화이트박스 재사용의 경우를 모두 고려하여 재사용성을 측정하여야 한다.

그러나 이러한 클래스 단위의 재사용도 복잡성이나 중복성, 제어의 흐름으로 인하여 한계점에 이르렀다. 이와 같은 클래스 재사용의 한계점을 극복하고, 소프트웨어의 재사용성을 향상시키기 위해 컴포넌트 단위의 재사용 기법이 도입되고 있다. 컴포넌트란 미리 구현된 독립된 단위 모듈의 소프트웨어 부품을 말하며, 새로운 소프트웨어 개발 과정에서 컴포넌트를 재사용함으로써 소프트웨어 개발 기간을 단축시키고, 소프트웨어 생산성과 품질 향상, 유지보수 비용과 테스트 비용을 절감할 수 있다. 또한 컴포넌트는 개발자에게 내부의 상세한 부분은 숨기고 인터페이스만 제공하여 쉽고 빠르게 소프트웨어를 구축할 수 있다[5]. 즉 개발자는 블랙 박스 형태의 컴포넌트를 사용하여 시스템을 조립한다[6].

클래스 단위와 컴포넌트 단위의 재사용 방식 사이에는 여러 가지 차이점이 있다. 표 1에서 보는 바와 같이 클래스 단위의 재사용은 소스코드 수준의 재사용이므로 클래스의 내부 소스코드를 수정하여 재사용한다. 그러나 컴포넌트 단위의 재사용은 바이너리 코드를 재사용함으로 내부 코드를 알 필요가 없고, 그 자체가 독립적으로 실행이 가능한 단위이지만 클래스 단위의 재사용은 소스코드를 개발 시스템에 결합하여 컴파일을 해야하며 클래스 그 자체로는 독립적으로 실행될 수 없으므로 모듈성이나 재사용 용이성이 컴포넌트 단위의 재사용이 높다. 또한 컴포넌트 단위의 재사용은 인터페이스를 통해 사용자의 목적이 맞게 컴포넌트의 속성이나 메소드을 변경할 수 있다.

표 1 클래스와 컴포넌트의 비교

단위 / 특징	클래스		컴포넌트
	블랙박스	화이트박스	
재사용 형태	소스코드 (수정없이)	소스코드 (수정한 후)	바이너리 코드
커스터마이즈	×	소스코드수정 (어렵다)	속성 설정 (용이)
모듈성	낮다		높다
인터페이스	적음		많음
재사용용이성	낮다		높다

소프트웨어 개발의 시간 단축과 효율의 극대화를 위하여 컴포넌트 기반 소프트웨어 개발(component-based software development: CBSD)에 대한 투자와 연구가 활발히 이루어지고 있다. 현재 대표적인 컴포넌트 모델로는 마이크로소프트사의 COM/DCOM, OMG의 CORBA, 썬마이크로시스템즈사의 JavaBeans/EJB등이 있다.

이 논문에서는 객체 지향 언어에서 재사용 단위인 클래스의 블랙박스 재사용과 화이트 박스 재사용의 경우를 모두 고려한 재사용성 측정 모델과 블랙박스 재사용의 성질을 가진 마이크로소프트사의 COM 컴포넌트의 재사용성 모델을 제안하고 이를 비교하여 본다.

2. 소프트웨어의 품질 메트릭

2.1 절차적 품질 메트릭

클래스나 컴포넌트는 객체지향언어에 의해 작성되지만 그 내면에는 근본적으로 절차언어의 속성을 동시에 지니고 있으므로 절차적 품질 메트릭을 살펴볼 필요가 있다. 구조적 프로그래밍 언어의 특성을 지니고 있는 절차적 품질 메트릭은 많이 제안되었고 아울러 사용되고 있다.

소프트웨어의 품질은 소프트웨어 생산물의 품질을 표현하고 평가할 수 있는 속성들이다.

소프트웨어 부품의 품질을 평가함에 있어서 이를 정량적으로 측정할 수 있는 메트릭이 중요하다. 메트릭이란 소프트웨어가 보유하고 있는 특성, 품질, 속성의 크기나 정도의 계량적인 표현을 말한다. 절차 언어에 대

해 지금까지 여러 정량화된 품질 평가 도구와 메트릭들이 정의되어 왔다. 복잡도를 나타내는 메트릭으로 LOC(Line Of Code), MaCabe의 순환 복잡도(cyclomatic complexity), Halstead의 소프트웨어 과학(software science) 등이 있고, 모듈성을 나타내는 메트릭으로 Yourdon과 Constantine의 Fan-in, Fan-out, 그리고 응집도(cohesion)와 결합도(coupling) 등이 있다. 문서화 정도를 나타내는 메트릭으로 주석이 있으며, 주석의 비율로 측정 가능하며, 프로그램의 각 부분의 정확한 기능, 사용법과 인터페이스를 사용자에게 쉽게 이해하게 해 준다.

2.2 객체지향적 품질 메트릭

객체지향적 품질 메트릭은 절차적 품질 메트릭과는 달리 객체지향언어의 특성인 상속, 정보은닉, 캡슐화, 객체 추상화 등과 같은 특성을 가지고 있어야 한다. 기존에 제안된 객체지향적 품질 메트릭들을 살펴보면 다음과 같다. S. R. Chidamber와 C. F. Kemerer는 클래스당 가중치를 갖는 메소드의 수(WMC), 상속 트리의 깊이(DIT), 자노드의 수(NOC), 객체간의 결합도(CBO), 클래스에 대한 응답도(RFC), 인스턴스 변수를 참조하는 클래스 내의 다른 메소드의 수 즉 메소드들의 응집도(LCOM) 등을 제안하였다7]. 이는 계측이론에 근거를 두고 있으며 클래스 수준에서 품질을 평가 할 수 있게 한다. M. Lorenz는 프로젝트, 클래스, 메소드 단위로 측정하는 메트릭들을 제안하였다[8]. 앞에서 언급된 메트릭들이 많이 적용되고 있으나 각기 평가 기준이 다르고 그에 대한 타당성 여부에 대해서도 논쟁이 계속 되고 있다. Briand, Morasca와 Basili는 객체지향 소프트웨어 메트릭의 모호성을 해결하기 위해 품질 평가 측정 개념을 정의하였다. 즉 크기, 길이, 복잡도, 결합도, 응집도를 기준으로 한 속성 기반 메트릭을 제안하였다[9].

2.3 기존의 재사용 품질 메트릭 시스템

클래스 단위의 소프트웨어의 재사용성을 측정하는 연구가 많이 있어 왔다. Prieto Diaz와 Freeman[4]은 라이브러리 시스템(library system)을 구축하였는데, 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.

CARE 시스템[3]은 C언어를 지원하는 시스템으로 재사용 속성 모형(reusability attribute model)을 제안하였다. 이 방법은 프로그램의 복잡도를 측정하는데 사용되는 메트릭을 위주로 함으로써 재사용성의 측정을 복잡도 측정으로 한정짓는 결과를 가져왔으며[10], 정량적으로 품질의 값을 추출할 수 있지만 절차언어의 속성에만 치중하여 객체 지향 언어의 속성을 반영하지 못하였다[11].

REBOOT(Reuse Based on Object-Oriented Techniques) 시스템[12]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 메트릭을 사용하였다.

3장 1절과 2절에서 각각 절차적 품질 메트릭과 객체지향적 품질 메트릭에 관한 주요한 연구에 대해 살펴보았는데, 절차적 품질 메트릭들은 객체지향언어의 품질 속성들을 반영하지 못하였고, 객체지향적 품질 메트릭들은 이론에 너무 치우쳐 있고 정량적 기준을 제시하는데 미흡하였다. 이 논문에서 재사용 단위인 클래스는 객체지향언어의 속성과 절차언어의 속성을 모두 포함하고 있으므로 양쪽 속성을 모두 고려하여 블랙박스 재사용과 화이트박스 재사용에 대한 재사용성 모델과 아울러 컴포넌트에 대한 재사용성 모델을 제안하고자 한다.

3. 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 재사 용을 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[13].

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 서

로 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 화이트 박스 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 메트릭을 모두 적용하여 재사용성을 측정한다.

독립된 소프트웨어의 단위 모듈로 구현된 컴포넌트는 내부의 상세한 부분은 숨기고 인터페이스만 제공하여 기성품 형태로 조립하여 새로운 시스템을 구축하므로, 클래스의 블랙박스 재사용 형태과 유사하다. 또한 외부 인터페이스를 통해 컴포넌트 내의 자료를 변경하여 사용하므로 인터페이스 복잡도를 고려하여 재사용성을 측정하여야 한다.

3.1 클래스의 화이트박스 재사용성 측정 모델

화이트박스 재사용성(white-box reusability)의 중요한 품질 메트릭은 수정성이다. 화이트박스 재사용성을 측정하는데 필요한 메트릭은 다음과 같다.

WBR 메트릭 : CC , CBO , DIT , DOC

CC 메트릭 : WMC , LCOM , RFC

DOC 메트릭 : 주석(comments)

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 메트릭로 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 깊이도 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.

3.2 클래스의 블랙박스 재사용성 측정 모델

블랙박스 재사용성(black-box reusability)은 클래스의 독립성과 정보은닉이 중요한 품질 요소이다. 이를 측정하는데 필요한 메트릭은 다음과 같다.

BBR 메트릭 : CI , IH , PA

CI 메트릭 : CBO , LCOM

여기서 BBR = 블랙박스 재사용성(Black-Box Reusability)

CI = 클래스 독립성

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 의

미한다. 모듈내에서 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 메트릭로 공통적으로 사용하고 있다. 결합도가 크면 수정에 따른 파급효과가 크므로 수정성에 영향을 미치고, 응집력 결여도의 값이 크면 이해가 어려워 수정에 영향을 미치므로 화이트박스 재사용성의 중요한 메트릭이 된다. 응집도와 결합도는 모듈성을 측정하는 메트릭로 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 메트릭은 블랙박스 재사용성에서도 중요한 메트릭이 된다.

3.3 컴포넌트 재사용성 측정 모델

컴포넌트는 미리 구현된 실행 단위로 사용자에게 내부 상세한 부분을 숨기고 인터페이스만 제공하여 쉽게 블랙박스 형태로 사용된다. 컴포넌트 재사용성(component reusability)은 독립성과 정보은닉, 컴포넌트의 인터페이스가 중요한 품질 요소이다. 이를 측정하는데 필요한 메트릭은 다음과 같다.

CR 메트릭 : COI , IH , IC
COI 메트릭 : CBO , LCOM
IC 메트릭 : CP, CM, CE

여기서 CR = 컴포넌트 재사용성(Component Reusability)
COI = 컴포넌트 독립성
IC = 인터 페이스 복잡도
CBO = 객체 사이의 결합도
LCOM = 응집력의 결여도
CP = 컴포넌트 속성
CE = 컴포넌트 이벤트

인터페이스는 컴포넌트에 의해 제공되는 서비스를 정의하며, 컴포넌트의 인터페이스로는 컴포넌트 내부 상태에 대한 변화를 줄 수 있는 속성 인터페이스와 컴포넌트에 의해 제공되는 서비스를 호출하는 함수 인터페이스, 이벤트가 발생했을 때 이를 처리는 이벤트 인터페이스로 구성된다. 즉 인터페이스를 통하여 컴포넌트를 커스터마이즈하여 사용하므로 인터페이스의 복잡도가 재사용에 영향을 미친다.

4. 재사용성 측정 메트릭

Ⅳ장에서 재사용성 측정을 위한 모델이 제안되었고 그에 대한 구체적인 메트릭에 대해 살펴보았다. 여기서는 실제 재사용성 측정을 위한 메트릭들의 계산 방법에 대해 알아본다.

4.1 클래스의 복잡도

객체 지향 메트릭 연구중에서 가장 대표적인 것이 M.I.T의 Chidamber와 Kemerer의 연구[7]인데 계측이론에 근거를 두고 메트릭 집합을 정의하였기 때문에 신뢰성이 높다. 따라서 이 논문에서도 재사용성에 영향을 미치는 주요한 메트릭을 사용하여 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 순환수(cyclomatic number)로 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} Uall $_i$ {R_i}
{R_i} = 메소드 $_i$에 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.

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

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

LCOM = | P | - | Q | 단, | P | > | Q | 경우
= 0 그렇지 않은 경우

4.2 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.

4.3 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.

4.4 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[14].

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

4.5 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 퍼블릭 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.

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

4.6 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.

$$PA = \left(\sum_{i=1}^{n} M_i\right) / n$$

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.

4.7 인터페이스 복잡도

인터페이스 복잡도는 컴포넌트의 속성, 메소드, 이벤트와 관련이 있다. 설계나 실행시에 변경할 속성값, 수행할 메소드, 발생 가능한 이벤트 등이 많으면 상대적으로 알아야하는 정보가 많아지므로 재사용에 불리하다.

CP = 컴포넌트의 속성값 개수

CM = 컴포넌트 메소드들의 매개변수의 합

$$CM = \sum_{i=1}^{n}(1 + M_i * 0.1)$$

CE = 컴포넌트의 발생 가능한 이벤트 개수

여기서 M_i는 i번째 메소드의 매개변수 개수이고 해당 메소드가 매개변수가 없거나 혹은 개수가 다르므로 0.1의 가중치를 부여하여 차이를 두었고, n은 컴포넌트의 메소드의 개수이다.

5. 적용 예 및 평가

기존의 소프트웨어 시스템에서 재사용성을 측정한 후 이 결과가 재사용성이 높은지 낮은지를 판단할 수 있는 기준이 필요하다. 표 2는 메트릭의 선택 기준을 나타낸다.

McCabe는 경험적 측정을 통해 cyclomatic number의 상한값을 10으로 제시하였다[15]. 그리고 한 클래스의 멤버 함수의 수는 7±2가 좋으므로[11] WMC는 70(10×7)으로 정하였다. RFC, CBO와 LCOM의 기준값은 [7]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.

DOC는 주석 대 코드비율을 계산할 때, 주석 라인 수를 주석이 아닌 라인 수와 나누어서 구하는데 C프로그램의 경우 보통은 1.0이상이면 좋고 최소한 0.8이상 되어야 한다. 이것은 이 논문의 계산 방법에 따라 약 0.5가 되므로 0.5 이상을 최소 기준으로 정하였다.

표 2 각 메트릭의 권장 선택 기준

메트릭	WMC	RFC	LCOM	CBO	DIT	DOC	PA
기준치	70이하	6이하	0	0	3이하	0.5이상	3이하

매개 변수는 [10]에서 측정한 결과 평균 2.44개이므로 이를 올림하여 3이하로 정하였다.

재사용성 측정을 위한 각 메트릭들의 계산 방법과 결과 값들의 의미가 다르므로 비교를 위해 모든 메트릭들에게 100점을 만점으로 하여 각각의 특성에 따라 등급

을 나누어 점수를 주어 평가하도록 하며, 각 메트릭의 평가 점수를 합산하여 전체 재사용성을 평가한다. 다음은 각 메트릭의 평가점수 계산 방법이다.

WMC 평가점수 = 100, WMC ≤ 70 일때
100 - (WMC - 70)
0, 계산 결과가 음수 일때
RFC 평가점수 = 100, RFC ≤ 6 일때
100 - (RFC - 6)
0, 계산 결과가 음수 일때
LCOM 평가점수 = 100, LCOM = 0 일때
100 - LCOM
0, 계산 결과가 음수 일때
CBO 평가점수 = 100, CBO = 0 일때
100 - CBO × 10
0, 계산 결과가 음수 일때
DIT 평가점수 = 100, DIT ≤ 3 일때
100 - (DIT - 3) × 10
0, 계산 결과가 음수 일 때
IH 평가점수 = IH × 100
DOC 평가점수 = 100, DOC ≥ 0.5 일때
DOC × 100 × 2
PA 평가점수 = 100, PA ≤ 3 일때
100 - (PA - 3) × 10
0, 계산 결과가 음수 일 때

화이트박스 재사용성은 6개의 메트릭 평가점수의 합으로 총점수가 600점이 만점으로, 블랙박스 재사용성은 4개의 메트릭 평가점수의 합으로 총점수가 400점이 만점이다.

표3은 [16]의 C++로 작성된 CASE 파일에서 Entry (클래스A), FunctionEntry(클래스B), ClassEntry(클래스C), ClassMemberDefinition(클래스D) 등 4개의 클래스를 추출하여 화이트박스 재사용성과 블랙박스 재사용성을 5장에서 제안한 메트릭을 사용하여 측정한 결과이다.

표 3 클래스의 재사용성 측정

재사용성 / 척 도	클래스A		클래스B		클래스C		클래스D	
	WBR	BBR	WBR	BBR	WBR	BBR	WBR	BBR
WMC	29		10		10		25	
RFC	24		8		6		23	
LCOM	46	46	8	8	0	0	140	140
CBO	1	1	1	1	2	2	0	0
DIT	1		2		1		0	
DOC	0.4		0.4		0.4		0.49	
IH		0.95		0.8		0		1
PA		0.86		0.75		0.5		0.77
총평가점수	506	339	560	362	560	280	481	300

총평가점수는 각 메트릭의 평가 점수 계산 방법에 따라 계산된 점수들을 합산한 것으로 상대적으로 점수가 높을수록 다른 클래스에 비해 재사용성이 높다. 예를 들어 클래스A의 화이트박스 재사용성의 총평가점수를 산정하여 보면, 각 메트릭의 평가점수 계산 방법에 따라 WMC의 값이 29이므로 'WMC ≤ 70' 조건을 만족하므로 평가점수는 100점이 되고, RFC의 값은 24이므로 '100 - (RFC - 6)' 식에 따라 계산된 평가점수는 82점이 되고, LCOM의 값은 46이므로 '100 - LCOM' 식에 따라 계산된 평가점수는 54점이 되고, CBO의 값은 1이므로 '100 - CBO × 10' 식에 따라 계산된 평가점수는 90점이 되고, DIT의 값은 1이므로 'DIT ≤ 3 일 때' 조건을 만족하므로 평가점수는 100점이 되고, DOC의 값은 0.4이므로 'DOC × 100 × 2' 식에 따라 계산된 평가점수는 80점이 된다. 이들의 각 메트릭의 평가점수들의 합이 총평가점수로서 506점이 된다.

클래스 A, B, C와 D 모두 WMC의 수치가 낮게 나온 것은 메소드의 논리 구조가 지극히 단순하기 때문이다. RFC가 기준치보다 높은 것은 메소드의 수가 많은 결과이고, 클래스 D의 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 적용하였으면, 네 개의 클래스가 모두 기준치에 근접하였다.

클래스 C의 정보은닉된 인스턴스 변수와 메소드가 없으므로 0으로 나왔다. 인터페이스의 정보를 알려주는 PA는 평균 1 미만으로서 각 메소드의 매개변수가 1개 정도 사용된 것을 의미한다. 이상으로 보아 평가 결과는 화이트박스 재사용성은 다른 클래스에 비해 상대적으로 클래스 B와 C가 높아 화이트박스 재사용이 유리하고, 블랙박스 재사용성은 클래스 B가 가장 높게나와서 블랙박스 재사용의 경우 다른 클래스에 비해 유리하다는 것을 알 수 있다. 또한 클래스 B는 화이트 박스 재사용성과 블랙박스 재사용성 모두 높게 나왔으며, 클래스 C의 경우 화이트 박스 재사용성은 높으나 블랙박스 재사용성은 낮다고 볼 수 있으므로, 화이트 박스 재사용성이 유리하다. 반대로 클래스 D는 화이트 박스 재사용성은 낮으나, 블랙박스 재사용성은 높다고 볼 수 있다.

표 4는 컴포넌트의 인터페이스 복잡도를 나타낸다. COM 컴포넌트인 VisualBasic의 20개의 기본 컨트롤들에 대해서 인터페이스 복잡도를 구하였다. 윈도우 프로그램을 구현하기 위한 도구로서 속성, 메소드, 이벤트가 상당히 복잡하다는 것을 알 수 있다.

표 4 컴포넌트의 인터페이스 복잡도

인터페이스 복잡도	Mean	Median	Max	Min
CP	33	42	58	6
CM	7	7.6	27.2	0
CE	15	0,10,13,16,18,19	26	0

7. 결 론

이 논문에서 객체지향 프로그램으로 작성된 소프트웨어 시스템에서 재사용 가능한 부품인 클래스에 대해 재사용성을 측정하기 위한 모델과 CBD에서 컴포넌트의 재사용성 측정 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 가능하여 화이트박스 재사용이 유리한지 혹은 블랙박스 재사용이 유리한지를 판단할 수 있으며, 적용 예를 통하여 정량화가 가능함을 보여 주었다. 즉 재사용 가능한 부품인 클래스들을 품질 메트릭에 따라 측정하여 높은 품질의 클래스를 선택할 수 있게 함으로써 재사용 비용과 시간을 줄일 수 있다. 컴포넌트의 경우는 블랙박스의 재사용 형태를 나타내며, 인터페이스의 복잡도가 중요한 품질 메트릭임을 알았다. 아울러 많은 실증적 연구를 통하여 컴포넌트의 재사용성을 평가할 수 있는 측정 기준값이 필요하다. 제안된 모델은 소프트웨어 품질을 나타내는 여러 가지 메트릭들로 이루어져 있으며 각각의 메트릭들은 서로 다른 가중치를 가질 수 있을 것이다. 이는 재사용성을 측정하는 메트릭 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. W. Krueger, "Software Reuse," ACM Computing Surveys, Vol. 24, No. 2, pp. 131-184, Jun. 1992.

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

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

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

[5] Brown A. W. & Wallnau K. C., "The Current State of CBSE," IEEE Software, pp. 37-42, Sept./Dct. 1998.

[6] Desmond Francis D'Souza, Alan Cameron Wills, Objects components and frameworks with UML : the Catalysis approach, Addison Wesley Longman, Inc., 1998.

[7] 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.

[8] M. Lorenz, Object-Oriented Software Metrics: A Practice Guide, Prentice Hall, 1994.

[9] L. C. Briand, Sandro Morasca, Victor R. Basili, "Property-Based Software Engineering Measurement," IEEE Tr. on S.E., Vol. 22, No. 2, Jan. 1996.

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

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

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

[13] 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.

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

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

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

윤 희 환

1982년 계명대학교 이공대학 전자계산학과 졸업(이학사). 1984년 중앙대학교 대학원 전자계산학과 졸업(이학석사). 1994년 ~ 현재 충북대학교 대학원 전자계산학과 박사과정 수료. 1995년 ~ 현재 원주대학 사무자동화과. 관심분야는 소프트웨어 공학(재사용, 품질 메트릭스)임.

김 영 집

1976년 2월 동국대학교 경영대학원 E.D.P.S 전공(경영학석사). 1990년 3월 충북대학교 대학원 전산통계학과 전산학 전공(이학석사). 1994년 2월 충북대학교 대학원 전자계산학과 박사과정 수료. 1979년 3월 ~ 현재 숭실대학교 전자계산원 정보처리과 교수. 관심분야는 객체지향 테스팅.

구 연 설

1964년 청주대학교 졸업. 1975년 성균관대학교 경영행정대학원 전자자료처리 전공(경영학석사). 1981년 동국대학교 대학원 통계학 전공(이학석사). 1988년 광운대학교 대학원 전자계산학 전공(이학박사). 1994년 ~ 1995년 한국정보과학회 부회장. 1979년 ~ 현재 충북대학교 컴퓨터과학과 교수. 관심분야는 객체지향 테스팅, 품질관리, 정보 검색, 전자 도서관.