韓國 컴퓨터情報學會 論文誌
第10卷 第4號, 2005. 9.

2005-10-4-4-11

CBD 환경에서 컴포넌트의 재사용성 측정 메트릭스

윤 희 환*

Metrics for Measurement of Component Reusability in Component-Based Development

Hee-Whan Yoon*

요 약

프로그램의 재사용은 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 컴포넌트의 재사용성 측정 모델과 측정 기준을 제안한다. 제안된 모델을 사용하여 측정된 컴포넌트는 재사용 정도를 측정하여 재사용성이 높은 컴포넌트를 선택할 수 있다.

Abstract

The reuse of a programs is classified 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 component that is measured by proposed model can know a degree of reuse and we can select which component is high in resuability.

▸ Keyword : Reusability, Metrics, Component

• 제1저자 : 윤희환
• 접수일 : 2005.06.22, 심사완료일 : 2005.08.02
* 원주대학 컴퓨터정보관리과 부교수

Ⅰ. 서 론

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 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.

요즈음 소프트웨어의 재사용성을 향상시키기 위해 컴포넌트 단위의 재사용 기법이 도입되고 있다. CBD(Component-Based Development)는 미리 구현된 소프트웨어의 단위 모듈인 컴포넌트(component)를 조립하여 소프트웨어를 구축하는 방법이다. 컴포넌트란 미리 구현된 독립된 단위 모듈의 소프트웨어 부품을 말하며, 새로운 소프트웨어 개발 과정에서 컴포넌트를 재사용함으로서 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][6]. 즉 개발자는 블랙박스 형태의 컴포넌트를 사용하여 시스템을 조립한다. 기성품 형태의 컴포넌트는 소프트웨어 개발 시스템이나 프레임워크(framework)에 플러그 앤 플레이(plug-and-play)하여 새로운 시스템을 구축한다. 이처럼 컴포넌트의 명세와 구현의 분리는 완벽한 캡슐화를 제공함으로서 컴포넌트 제공자와 사용자 사이에서 명세의 변경을 제한하고 보호하는 역할을 수행한다.

또한 객체지향개발이 소프트웨어 개발비용이나 품질을 만족할 만큼 향상시켜 주지 못했고, 컴포넌트 기반 개발 방법론(CBD)은 어플리케이션 개발의 산업화에 의해 개발비용의 절감 및 개발기간의 단축으로 신속한 제품화가 가능하기 때문에 어플리케이션을 개발하는 최선의 방법으로 간주되며 부각되었다. 이러한 컴포넌트들은 높은 안정성과 신뢰성을 확보하기 위하여 소프트웨어 메트릭스 시스템을 통하여 품질이 평가되고, 인증되어야 한다.

컴포넌트는 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.

현재 대표적인 컴포넌트 모델로는 마이크로소프트사의 COM/DCOM, OMG의 CORBA, 썬마이크로시스템즈사의 JavaBeans/EJB 등이 있다. 이 논문에서는 블랙박스 재사용의 성질을 가진 마이크로소프트사의 COM 컴포넌트의 재사용성 모델을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 컴포넌트와 컴포넌트 기반 개발에 대해 기술하고, 3장에서는 소프트웨어 품질 메트릭에 대해서 설명하다. 4장에서는 컴포넌트 재사용성 측정 모델과 측정 메트릭을 제안하고, 5장에서 제안한 모델을 실제 적용한 결과를 제시하며, 6장에서 결론을 맺는다.

Ⅱ. 관련 연구

2.1 컴포넌트 특성

컴포넌트 기반의 개발(CBD) 또는 컴포넌트 기반 소프트웨어 공학(CBSE)은 컴포넌트 개발은 물론 컴포넌트를 조립하여 시스템을 구축하는 것을 뜻한다. [5], [6], [7]의 연구를 요약하면 컴포넌트가 지니는 공통적인 특성은 모듈성(modularity), 인터페이스(interface), 아키텍처(architecture), 조립(composition), 동적 바인딩(dynamic binding), 커스터마이즈(customization) 등이다.

2.1.1 Modularity

컴포넌트는 시스템의 대치할 수 있는 부분으로 여러 시스템에 적용가능한 부품이며 시스템에 매우 독립적이다. 물

론 컴포넌트가 사용되기 위해서는 적절한 아키텍쳐에 종속적이어야 한다. 컴포넌트가 독립성이 강하다는 것은 여러 도메인에서 컴포넌트의 속성을 변경하지 않고 하드웨어처럼 플러그 앤 플레이(plug-and-play)하여 소프트웨어를 구성할 수 있음을 의미한다.

2.1.2 Interface

컴포넌트는 블랙박스 형태의 재사용 단위로 소프트웨어 개발 시스템에서 컴포넌트를 사용하기 위해 컴포넌트 외부 인터페이스만 알면 되며 내부의 구현 모듈에 대해서는 알 필요가 없다. 인터페이스는 컴포넌트에 의해 제공되는 서비스를 정의한다.

2.1.3 Architecture

컴포넌트는 다른 컴포넌트나 프레임워크와 상호작용할 수 있도록 하기 위해 미리 정의된 아키텍쳐에 맞게 설계되고 구현되어야 한다. 컴포넌트 기반 아키텍쳐는 컴포넌트를 첨가하거나 대치하여 기능적인 향상을 이룰 수 있도록 프레임워크 형태로 제공된다. CORBA나 DCOM과 같은 컴포넌트 아키텍쳐는 프레임워크 형태로 컴포넌트가 운영될 수 있도록 컴포넌트를 조정하거나 대신 작업을 처리해 준다.

2.1.4 Composition

플러그 앤 플레이할 수 있는 컴포넌트는 인터페이스를 통해 컴포넌트를 조립한다. 이러한 조립은 다중의 컴포넌트들 간에 이루어지는 행위를 설계하는 것이 필요하며, 실행 시에 조립될 수 있어야 한다.

2.1.5 Dynamic Binding

동적 바인딩은 컴포넌트 기반 소프트웨어를 개발하기 위한 중요한 특징 중에 하나로서 어플리케이션과 컴포넌트 사이에서 양방향으로 이루어진다. 실행 시에는 인터페이스를 통해 동적으로 어플리케이션에 연결될 수 있으며, 다른 컴포넌트와도 동적으로 연결된다. 또한 개발 시스템의 재컴파일 없이 동적으로 바인딩되어 사용될 수 있다.

2.1.6 Customization

시스템 개발자는 컴포넌트를 사용하여 어플리케이션을 개발할 때, 컴포넌트의 속성을 변경하여 개발하려는 시스템에 컴포넌트를 적절하게 끼워 넣을 수 있다. 이를 커스터마이즈라 하며 일반적으로 컴포넌트 내의 데이터를 접근하기 위해 인터페이스를 이용한다.

2.2 Component-Based Development 환경 모델

컴포넌트 기반 개발 방법론(Component-Based Development: CBD)은 컴포넌트 생산, 선택, 평가 및 통합으로 구성되는 소프트웨어 개발 방법의 새로운 패러다임이다. 이 방법론이 성공적이기 위해서 이와 관련된 많은 기술들이 컴포넌트 기반 개발 방법론 내에서 지원되어야 한다. 즉, 컴포넌트 기반 개발 방법론은 많은 관련 기술의 통합을 필요로 하므로 이러한 기술들이 통합 구현되어야 그 효과를 극대화할 수 있다. (그림 1)은 컴포넌트 기반 개발 방법론과 관련된 기술들로 구성된 컴포넌트 기반 개발 방법론의 모델을 제시한다. 컴포넌트 기반 개발 방법론의 모델은 컴포넌트 생성 부분과 컴포넌트 구축 부분으로 나눌 수 있다. 컴포넌트 생성과 관련된 기술은 객체지향 프로그래밍(Object-Oriented Programming)에 기반한 재사용을 위한 설계/개발(Design/Development for Reuse), 디자인 패턴 및 프레임워크(Design Patterns and Frameworks), 컴포넌트 기술/명세(Component Description/ Specification), 영역 공학(Domain Engineering) 및 컴포넌트 인증(Component Certification) 등이 있다. 컴포넌트 구축과 관련된 기술은 재사용 저장소(Reuse Repository)를 포함하는 재사용에 의한 설계/개발(Design/Development with Reuse), 영역 공학(Domain Engineering) 및 재사용 메트릭스(Reuse Metrics) 등이 있다. (그림 1)은 OOP 패러다임과 새로운 CBD 패러다임을 통합함으로서 OOP의 한계를 극복하고 재사용 효과를 극대화시키는 접근 방법으로 제시되었다.

2.2.1 컴포넌트 기술/명세

블랙박스 재사용(blackbox reuse)의 경우, 사용자의 의도나 요구사항을 만족시키기 위해 사용될 수 있는 컴포넌트는 실행코드 자체와 컴포넌트 명세로 구성된다. 컴포넌트의 명세는 인터페이스 정의 언어(IDL: Interface Definition Language)나 정형 명세 언어(formal description language)에 의해 작성된다.

어플리케이션 개발자가 컴포넌트 기반 개발 방법론을 사용해 소프트웨어를 구현할 때, 구현 내역에 대해서는 알 필요가 없다. 그는 구현으로부터 분리된 명세 내의 인터페이스에 정의된 오퍼레이션을 호출함으로서 컴포넌트를 쉽게 사용할 수 있다. 인터페이스 정의 언어는 인터페이스와 오퍼레이션을 정의하기 위한 CORBA 명세의 일부분이다. 재사용을 위한 설계/개발 과정에서 컴포넌트의 명세가 정형 명세 언어를 사용해서 기술된다면 컴포넌트의 명세는 자동적으로 소스코드로 변환될 수 있다. 아키텍쳐 명세 언어(ASL: Architecture Specification Language)는 CORBA

IDL[8]의 확장으로서 이것은 컴포넌트의 기능을 제공 인터페이스(provided interface)와 필요 인터페이스(required interface)로 구분하여 정의한다. UML[9] 및 Catalysis[10]가 컴포넌트 기반 개발 방법론 영역에서 대표적인 명세 기술이다.

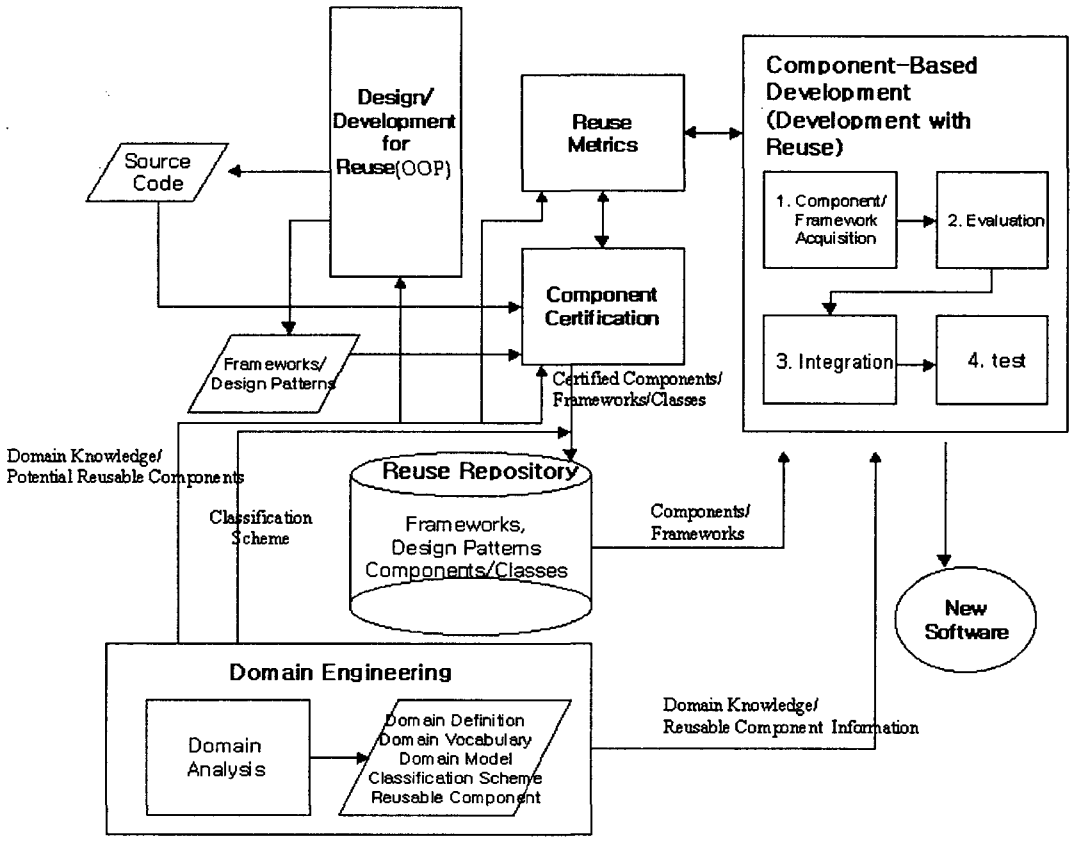

그림 1. 컴포넌트 기반 개발 방법의 환경 모델
Fig. 1 Environment Model of Component-Based Development

2.2.2 컴포넌트 생성

컴포넌트 기반 개발 방법론의 주요 효과는 컴포넌트에 내재된 재사용성에서 비롯된다. 그러므로 OOP 기반의 재사용을 위한 설계/개발 기술은 품질 좋고 재사용성이 높은 컴포넌트가 미래의 사용을 위하여 생성, 분류, 저장 및 관리될 수 있도록 구현되어야 한다. 그러나 재사용 컴포넌트를 설계/생성하는 것은 어렵고 이를 지원하는 도구가 없기 때문에 컴포넌트 생성 작업은 숙련된 설계자나 프로그래머만이 할 수 있으며, 컴포넌트 기반의 소프트웨어 개발 도구가 필요하다. OOP에 의해 생성된 객체/클래스는 사용하기 쉽지 않으므로 OOP에 근거를 두고 개발된 컴포넌트가 항상 사용하기 쉬운 곳은 아니다. 그러므로 OOP 기반의 프레임워크는 물론, OOP에 의해 생성된 컴포넌트는 사용의 용이성을 위해 인터페이스를 포함한 명세와 이진코드 형태의 구현 부분을 분리시켜 제공하여, 사용자가 복잡한 내부 구조를 몰라도 쉽게 사용할 수 있게 해야 한다.

2.2.3 영역 공학

영역 공학(domain engineering)은 영역 분석과 영역 모델링으로 구성된다. 영역은 전문화된 지식/정보의 집합체로서 특정 비즈니스와 연관이 있다. 영역 분석은 연관된 집

합 내의 객체와 오퍼레이션을 식별하고 공통적인 성질을 분석하여 현존 혹은 미래 시스템으로부터 공통적인 객체를 발견하는 과정으로 정의된다. 영역 분석은 현존 시스템으로부터 재사용 가능 컴포넌트를 발견함은 물론, 재사용을 위한 설계/개발 과정 중에서도 어느 재사용 컴포넌트가 신규 개발 시스템을 위하여 생성되어야 하는가를 결정하기 때문에 재사용 성공의 필수적인 요소이다. 영역 모델링 활동은 컴포넌트의 요구사항을 이해하고 그들을 컴포넌트 명세의 인터페이스 정의로 변환하는 것을 지원한다. (그림 2)에서처럼 영역 분석과 영역 모델링을 수행한 뒤에 영역 어휘(domain vocabulary), 영역 모델, 예상 재사용 컴포넌트 및 분류 체계(classification scheme)가 결정될 수 있다.

2.2.4 컴포넌트 인증

컴포넌트가 신규로 생성되면, 이들 컴포넌트들은 조직 내에 확립된 소프트웨어 메트릭스(metrics) 시스템을 사용하여 평가되고 인증되어야 한다. 컴포넌트 인증은 컴포넌트를 신뢰할 수 있게 하여 재사용을 촉진하는데 도움을 준다.

2.2.5 재사용 메트릭스

재사용 메트릭스는 재사용성이 높은 컴포넌트를 식별하며, 재사용을 통해 조직에 높은 이익을 가져 올 수 있는 잠재력이 높은 비즈니스 영역/시스템을 식별하기 위해서 사용될 수 있다. 이와 같이 재사용 메트릭스는 영역 분석을 통하여 시스템들 간에 가장 흔하게 반복되는 재사용 가능 컴포넌트들을 발견할 수 있도록 한다.

McClure[11]는 재사용 메트릭스를 위한 10종의 요소로서 컴포넌트의 공통성(commonality), 컴포넌트의 재사용 임계값(컴포넌트 개발 비용의 회수를 위한 최소 재사용 횟수), 특정 컴포넌트의 타 영역/시스템 내에 사용 대비 재사용의 상대적 장점, 재사용 컴포넌트의 생성 비용, 재사용 컴포넌트의 사용 비용(수정 비용 포함), 컴포넌트 유지보수 비용, 시스템의 공통성 정도, 시스템의 재사용성 정도, 시스템의 재사용 목표 수준, 시스템의 타 영역 내에 시스템 대비 재사용의 상대적 장점 등을 제시하였다. 컴포넌트 인증과 함께 재사용으로부터의 이익을 향상시키기 위한 필수적인 기술이다.

2.2.6 컴포넌트 저장소

컴포넌트들이 개발된 이후 데이터베이스를 이용한 저장소에 효율적으로 저장되어야 한다. 컴포넌트 저장소의 스키마 정보는 컴포넌트 도메인 분류, 컴포넌트의 기능, 특징, 인터페이스, 커스터마이징 스팟(customizing spot), 버전 정보 등을 가지고 있으며 이러한 정보를 저장한다.

컴포넌트의 저장소는 단일 서버에서 관리하는 것이 아니라 분산하여 관리하기 위해서는 CORBA나 DCOM과 같은 분산 컴포넌트 모델을 활용하여 여러 서버에 분산 관리할 수 있다. 마이크로소프트는 윈도우 환경에서 운영되는 COM 저장소를 연구하고 있으며, OMG도 CORBA 지향 저장소에 대한 표준을 발표하였다.

III. 소프트웨어의 품질 메트릭

3.1 절차적 품질 메트릭

컴포넌트나 클래스는 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 대해 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.

3.2 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 제안하였다[12]. 이는 계측이론에 근거를 두고 있으며 클래스 수준에서 품질을 평가할 수 있게 한다. M. Lorenz는 프로젝트, 클래스, 메소드 단위로 측정하는 메트릭들을 제안하였다[13]. 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[14].

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 컴포넌트는 객체지향언어의 속성과 절차언어의 속성을 모두 포함하고 있으며, 아울러 블랙박스 재사용의 형태를 나타내므로 이를 모두 고려하여 컴포넌트에 대한 재사용성 모델을 제안하고자 한다.

Ⅳ. 재사용성 측정 모델

재사용 방법에는 블랙박스 재사용과 화이트박스 재사용이 있다. 화이트박스 재사용은 소프트웨어 부품을 재사용하고자 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[15].

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.

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

4.1 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
- COI = 컴포넌트 독립성
- IH = 정보은닉 정도
- IC = 인터페이스 복잡도
- CBO = 객체 사이의 결합도
- LCOM = 응집력의 결여도
- CP = 컴포넌트 속성
- CM = 컴포넌트 메소드
- CE = 컴포넌트 이벤트

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.

응집력 결여도와 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 척도가 된다.

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 처리하는 이벤트 인터페이스로 구성된다. 즉 인터페이스를 통하여 컴포넌트를 커스터마이즈하여 사용하므로 인터페이스의 복잡도가 재사용에 영향을 미친다.

4.2 재사용성 측정 메트릭

4.1절에서 재사용성 측정을 위한 모델이 제안되었고 그에 대한 구체적인 메트릭에 대해 살펴보았다. 여기서는 실제 재사용성 측정을 위한 메트릭들의 계산 방법에 대해 알아본다.

4.2.1 컴포넌트의 독립성

4.2.2.1 LCOM(Lack of COhesion in Methods)

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.

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

LCOM = $|P| - |Q|$ 단, $|P| > |Q|$ 경우

= 0 그렇지 않은 경우

4.2.2.2 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.

4.2.2 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.

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

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

V. 적용 예 및 평가

기존의 소프트웨어 시스템에서 재사용성을 측정한 후 이 결과가 재사용성이 높은지 낮은지를 판단할 수 있는 기준이 필요하다. CBO와 LCOM의 기준값은 [13]에 의해 측정된 결과 C++의 경우 각각 0이 중간값으로 측정되어 이것을 기준으로 정하였다.

소프트웨어에서 사용되는 컴포넌트의 재사용성 측정을 위한 각 메트릭들의 계산 방법과 결과 값들의 의미가 다르므로 비교를 위해 모든 메트릭들에게 100점을 만점으로 하여 각각의 특성에 따라 등급을 나누어 점수를 주어 평가하도록 하며, 각 메트릭의 평가 점수를 합산하여 전체 재사용성을 평가한다. 다음은 각 메트릭의 평가점수 계산 방법이다.

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
IH 평가점수 = IH × 100
PA 평가점수 = 100, PA ≤ 3 일때
100 - (PA - 3) × 10
0, 계산 결과가 음수 일때

표 1. 컴포넌트의 인터페이스 복잡도
Table 1. Interface Complexity of Component

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

〈표 1〉은 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.

VI. 결론

이 논문에서 컴포넌트 기반 소프트웨어 개발 방법론으로 작성된 소프트웨어 시스템에서 재사용 가능한 부품인 컴포넌트의 재사용성 측정 모델을 제안하였다. 소프트웨어 부품을 재사용하는 방법에는 화이트박스 재사용과 블랙박스 재사용이 있는데 이 두 가지 방법은 서로 다른 성질을 갖는다. 컴포넌트의 재사용은 블랙박스의 재사용 형태를 나타내며, 재사용 가능한 부품인 컴포넌트들을 품질 메트릭에 따라 측정하여 높은 품질의 컴포넌트를 선택할 수 있게 함으로서 재사용 비용과 시간을 줄일 수 있다. 또한 컴포넌트의 경우 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.

참고문헌

[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] Desmond Francis D'Souza, Alan Cameron Wills, Objects components and frameworks with UML : the Catalysis approach, Addison Wesley Longman, Inc., 1999.

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

[7] Philippe Krutchen, "Modeling Component System with the Unified Modeling Language", International Workshop on Component-Based Software Engineering, 1998.

[8] Object Management Group, Inc., CORBA, The Common Object Request Broker.

[9] UML Group, Unified Modeling Language 1.1, Rational, 1997.

[10] Desmond Francis D'Souza, Alan Cameron Wills, Composing Modeling Frameworks in Catalysis, Technical Report, Icon Computing Inc., 1997.

[11] C. McClure, Model-Driven Software Reuse, Extended Intelligence, Inc., 1995.

[12] 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.

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

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

[15] 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.

[16] 황석형, 양해술, 클래스 계층구조의 재구성과 품질평가 방법, 정보과학회지 제23권 제3호 2005년 3월.

[17] 정규장, 웹 기반 공간데이터 공통 컴포넌트 설계 기법, 한국컴퓨터정보학회논문지 제9권 제1호 2004년 3월.

[18] 이정렬, 김정욱, 추상화 객체의 클러스터링에 의한 가시적 응집도 향상기법, 한국컴퓨터정보과학회논문지 제9권 제4호 2004년 12월.

저 자 소 개

윤 희 환

2001년 2월 충북대학교 대학원 전자계산학과 박사

1995년~현재 원주대학 컴퓨터정보관리과 부교수

〈관심분야〉 소프트웨어공학, 소프트웨어 품질평가, 컴포넌트공학