컴포넌트 설계에 대한 응집도와 결합도 메트릭스

고 병 선† · 박 재 년††

요 약

소프트웨어 개발의 독립성과 높은 생산성을 향상시키기 위한 재사용 기술로 컴포넌트 기반 개발 방법론은 널리 사용되게 되었다. 소프트웨어의 품질을 향상시키기 위해서는 측정 가능해야 하므로, 컴포넌트의 특성을 반영한 컴포넌트 메트릭스가 필요하다. 따라서 본 논문에서는 컴포넌트 기반 시스템의 컴포넌트 설계 정보에 기반한 컴포넌트 응집도와 결합도 메트릭스를 제안한다. 오퍼레이션이 컴포넌트의 서비스를 제공하기 위해 공통으로 사용하는 클래스에 대한 정보를 이용해 오퍼레이션 사용도를 구하고, 이를 통해 오퍼레이션 유사도를 구한다. 컴포넌트 응집도와 결합도는 오퍼레이션 유사도에 의해 계산되며, 컴포넌트 분석 단계에 추출 가능한 정보로부터 계산된다. 그리고 사례 연구를 통해 컴포넌트 메트릭스의 필요성을 객체지향 메트릭스와의 비교를 통해 살펴본다.

Cohesion and Coupling Metrics for Component Design Model

Byung-Sun Ko† · Jai-Nyun Park††

ABSTRACT

The component-based development methodology becomes famous as the reuse technology for independence and productivity of software development. It is necessary component metrics for component-based systems, because it should be measurable to improve the quality of the software. Hence, in this paper, we propose component cohesion and coupling metrics which is reflected in characteristics of component. The operation use value is calculated by the information of classes interface commonly uses to offer the component's service. And, the operation similarity value is calculated by the operations use value. Component cohesion and coupling is calculated by the operation similarity and based of the information which is extracted in the analysis phase. And, we examine the necessity of component metrics in comparison with object-oriented metrics.

키워드 : 컴포넌트 설계(Component Design), 인터페이스(Interface), 응집도(Cohesion), 결합도(Coupling)

1. 서 론

재사용 기술은 하드웨어가 공장에서 규격화되어 생산되는 것과 같이 소프트웨어도 규격화하여 하드웨어 부품처럼 기능의 조각을 조립하여 재사용하는 과정 중에 소프트웨어 개발의 가장 중요한 목표인 품질과 높은 개발 생산성을 달성할 수 있다고 생각하였다[1].

재사용 기술의 사용으로 높은 품질과 개발 생산성이라는 소프트웨어 개발의 목표를 달성하기 위해 1990년대부터 컴포넌트 기반 개발 방법론이 나타나기 시작했다. 이 방법론은 이미 존재하는 독립적인 기능의 조각인 컴포넌트를 조립함으로써 시스템을 개발하는 방법으로, 사용자의 변화하는 요구사항을 보다 쉽게 충족시켜 고품질의 시스템을 효율적으로 개발할 수 있는 기술이다. 컴포넌트는 복잡한 데이터와 동작은 내부에 숨기고 인터페이스만 외부에 분리되어 사용자, 조립자 그리고 개발자는 인터페이스를 통해서만 컴포넌트에 접근 및 사용할 수 있다. 이러한 컴포넌트의 캡슐화 특성은 에러의 영향을 제한시켜 유지보수를 용이하게 한다[2].

DeMarco가 "측정할 수 없는 것은 이해할 수 없다."라고 말한 것처럼 이해하기 위해서는 반드시 측정이 필요하다. 이에 소프트웨어에 대한 측정이 필요하고 이를 위해 소프트웨어 메트릭스가 필요하다. 소프트웨어의 특성상 소프트웨어에 대한 유지보수성(maintainability), 재사용성(reusability), 기능성(functionality), 신뢰성(reliability), 효율성(efficiency) 등과 같은 외부적(external) 품질 특성은 측정 가능한 내부(internal) 속성들에 의해 간접적으로 측정된다[3, 4]. 내부적 품질 속성에는 응집도(cohesion), 결합도(coupling), 복잡도(complexity), 크기(size) 등이 해당된다. 그 중 하나의 단위 안에 있는 요소들 간의 관계 정도를 측정하는 응집도와 단위들 간의 의존도를 측정하는 결합도는 신뢰성 있고 유지보수가 쉬운 소프트웨어에 대한 중요한 품질 측정 요소가 된다. 재사용이 가능한 독립된 부품 단위인 컴포

† 준 회 원 : 숙명여자대학교 대학원 컴퓨터과학과
†† 정 회 원 : 숙명여자대학교 정보과학부 교수
논문접수 : 2003년 5월 26일, 심사완료 : 2003년 6월 23일

넌트의 개발을 위해서는 컴포넌트 설계 단계에서 응집도와 결합도가 중요한 의미를 갖는다.

따라서 본 논문에서는 컴포넌트의 서비스가 인터페이스와 클래스들과의 상호 작용 관계를 통해 제공된다는 컴포넌트의 특성을 반영한 컴포넌트 기반 시스템의 개발 초기 단계인 분석 및 설계 단계에 적용 가능한 컴포넌트 응집도와 결합도 메트릭스를 제안한다.

2. 연구 배경

2.1 컴포넌트 기반 시스템 개발

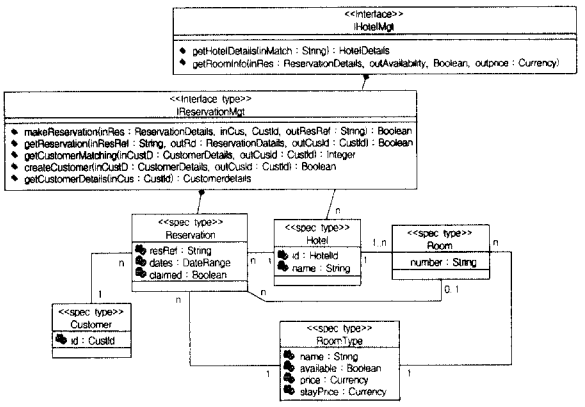
컴포넌트 기반 시스템 개발은 두 단계로 구성된다. 컴포넌트 아키텍처 명세와 컴포넌트 개발로 구성되는 개별 컴포넌트 개발을 위한 단계와 개발된 개별 컴포넌트의 조립으로 어플리케이션을 개발하는 두 단계로 구성된다. 컴포넌트 아키텍처 명세 단계에서는 지원하고자 하는 업무 영역에 대한 표준적인 컴포넌트 구성을 분석하여 아키텍처를 제시하고 각 컴포넌트의 명세를 설계한다. 컴포넌트 개발 단계에서는 주어진 컴포넌트의 명세를 만족하도록 컴포넌트를 설계하고 구현한다. 어플리케이션의 개발 영역에서는 개발된 컴포넌트를 재사용함으로써 요구되는 어플리케이션을 단기간에 개발한다[5-7].

본 논문에서는 컴포넌트 기반 시스템의 개별 컴포넌트의 개발을 위한 설계 모델에 관심을 가지므로, 전통적인 소프트웨어 개발 단계의 요구사항 분석, 분석, 설계의 전반부로 구성되는 컴포넌트 아키텍처 명세 단계에 대해서만 관심을 갖고, 이 부분에 대해서만 살펴본다.

컴포넌트 아키텍처 명세 단계에서는 요구 사항의 분석 결과를 토대로 업무 영역의 클래스들을 중요한 클래스를 중심으로 서로 연관되고 의존하는 것끼리 묶어, 클래스들 사이에 높은 응집력과 낮은 결합력을 갖도록 컴포넌트로 분할한다. 그리고 각 컴포넌트가 제공할 서비스를 나타내는 인터페이스를 정의한다. 인터페이스 상호작용 모델링은 컴포넌트가 제공하는 서비스를 분석하여 구체적인 오퍼레이션을 도출한다. 따라서 분석 단계에서는 시스템이 어떤 구조와 행위를 제공하는지를 파악하여 컴포넌트들의 역할과 관계를 정의한다[6].

(그림 1)은 본 논문의 사례 연구인 호텔 예약 시스템의 인터페이스 명세 다이어그램으로, 컴포넌트 아키텍처에 추가적으로 컴포넌트와 컴포넌트가 제공하는 서비스인 인터페이스의 구체적인 기능에 대한 인터페이스 명세를 나타낸다.

컴포넌트는 독립된 기능 단위로 자신만의 서비스를 제공하는 소프트웨어 단위이다. 이러한 컴포넌트의 서비스는 인터페이스에 접근하여 사용 가능하며, 인터페이스는 컴포넌트의 서비스를 표현하므로 매우 중요하다[6]. 컴포넌트 인터페이스 명세는 컴포넌트 개발자뿐만 아니라 시스템 분석자 그리고 설계자, 컴포넌트 조립자에게 유용하다. 컴포넌트가 제공해야 하는 기능과 인터페이스 사용자가 기대하는 기능을 정의하기 위한 인터페이스 명세는 컴포넌트 사용자와의 계약으로, 컴포넌트의 서비스를 제공하기 위한 인터페이스의 오퍼레이션에 대한 상세한 사항을 기술한다. 인터페이스 명세는 오퍼레이션 시그너처(signature)가 무엇인지와 오퍼레이션의 사전/사후 조건과 오퍼레이션이 호출되었을 때 컴포넌트의 상태나 오퍼레이션의 매개 변수에 미치는 영향이 무엇인지 등에 관한 상세한 계약 내용을 기술한다. 이 인터페이스 명세에 따라 컴포넌트가 제공하는 서비스에 대한 결정이 이루어지므로 중요하다.



(그림 1) 인터페이스 명세 다이어그램

2.2 객체지향 메트릭스

객체지향 시스템은 속성과 메소드가 캡슐화된 클래스를 기반으로 하며, 메세지를 통한 클래스들 간의 상호작용을 통해 서비스가 제공된다. 대부분의 객체지향 메트릭스는 객체지향 시스템의 기본 단위인 클래스에 기반을 두고, 속성, 메소드, 메시지 교환, 상속성, 캡슐화 등을 고려한 내부 품질 속성들의 측정을 통해, 객체지향 특성인 상속성, 재사용성, 캡슐화 등을 평가하고자 했다[3, 8]. 본 절에서는 본 논문과 관련이 있는 응집도와 결합도에 대한 대표적인 객체지향 메트릭스에 대해 살펴본다.

Chidamber와 Kemerer[9]가 제안한 메트릭스는 대표적인 클래스 중심 메트릭스이다. LCOM(Lack of COhesion in Methods)은 메소드의 유사성 정도에 의한 클래스 응집도로, 메소드가 사용하는 인스턴스 변수(instance variable)들의 교집합에 의해 이루어진 서로소 집합으로 계산된다. 클래스가 $M_1, M_2, \cdots, M_n$의 n개의 메소드를 갖고, 그 중 한 메소드 M_j가 사용하는 인스턴스 변수의 집합을 I_j라 할 때, $LCOM$은 다음과 같이 정의된다.

$$LCOM = |P| - |Q|, \text{ if } |P| > |Q|$$
$$= 0, \textit{ otherwise}$$

이때, $P = \{(I_i, I_j) \mid I_i \cap I_j = \emptyset\}$이고,
$Q = \{(I_i, I_j) \mid I_i \cap I_j \neq \emptyset\}$이다.

응집 결여도는 모든 메소드 조합 쌍 중에서 같은 속성을 공유하지 않는 메소드 쌍의 수와 공유하는 메소드 쌍의 수의 차로 계산되는데, 메소드들이 같은 속성을 사용하다면 클래스의 유사도가 크고 매우 응집력이 높음을 알 수 있다.

CBO(Coupling Between Object Classes)와 RFC(Response For a Class)는 메소드의 호출에 의한 클래스 결합도 메트릭이다. 객체 간 결합도 CBO는 상속성이 없는 클래스들 사이에서 하나의 클래스가 다른 클래스의 메소드나 속성을 사용하여 결합한 수를 측정한다. 이 값이 크면 클래스 설계의 모듈화와 재사용을 어렵게 만들며 변경에 대한 파급 효과가 커 유지보수를 어렵게 한다. 클래스에 대한 반응도 RFC는 클래스가 호출하는 모든 메소드의 수이다. RS가 한 클래스의 응답 집합(response set)이라 할 때, RFC는 다음과 같이 정의된다.

$$RFC = |RS|$$

이때, $RS = \{M\} \cup_{all\ i} \{R_i\}$이다.

$\{M\}$은 클래스 안에 선언된 메소드의 집합이고, $\{R_i\}$는 메소드 i에 의해 호출되는 메소드들의 집합이다. 이는 클래스의 반응 정도를 나타내며 호출되는 메소드가 많아질수록, 클래스의 테스트와 디버깅이 복잡하며 유지보수를 어렵게 한다.

2.3 컴포넌트 메트릭스의 필요성

컴포넌트 기반 시스템은 하나 이상의 컴포넌트로 구성되며, 컴포넌트는 하나 이상의 클래스와 인터페이스로 구성되며, 또한 인터페이스는 오퍼레이션의 집합이라는 클래스와는 다른 구조적 특성을 갖는다. 그러므로 클래스보다 큰 재사용 단위로서의 컴포넌트에 대한 측정은 컴포넌트가 갖는 구조적인 특성과 클래스와는 다른 측정 인자를 반영한 새로운 컴포넌트 메트릭스가 필요하다. 컴포넌트의 품질을 측정하기 위한 컴포넌트 메트릭스에 대한 연구는 아직까지 미흡한 실정이다. 컴포넌트 기술은 새로운 기술이 아닌 객체지향 기술을 기반으로 재사용이라는 목적을 달성하기 위해 발전된 기술로, 많은 부분에 있어 객체지향 기술의 영향을 받고 있는 게 사실이다. 그래서 아직까지 컴포넌트의 품질을 측정하기 위한 컴포넌트 메트릭스에 대한 연구가 미흡한 실정이다.

그러나 컴포넌트 기술은 객체지향 기술과는 다른 특성을 가지므로, 객체지향 메트릭스를 컴포넌트 기반 시스템의 컴포넌트 개발에 적용하여 품질을 측정하는 것은 부적합하므로, 컴포넌트 메트릭스가 필요하다[4].

Cho[10]가 제안한 컴포넌트 메트릭스는 컴포넌트의 구조적인 특성을 반영한 컴포넌트 메트릭스이기는 하나, 컴포넌트 고유의 특성보다는 클래스의 특성을 많이 반영한 객체지향적 메트릭스라는 인상을 준다. 또한, 측정을 위해 수집해야할 정보가 많아, 사용이 용이하지 않으며, 컴포넌트 메트릭스의 측정 결과와 컴포넌트의 품질간의 관계에 대한 설명이 미흡하다. 그리고, Kim[11]과 Choi[12]는 컴포넌트의 품질 측정을 위한 고유한 메트릭스라기 보다는 클래스들 간의 관계에 의한 컴포넌트 식별의 수단으로서의 메트릭스라 할 수 있다.

따라서 컴포넌트의 내부 품질 속성에 대한 측정을 통해 컴포넌트의 외부 품질을 향상시키기 위한 컴포넌트 메트릭스에 대한 연구가 필요하다. 이러한 메트릭스는 컴포넌트 기반 시스템의 내부 품질 속성에 의해 컴포넌트의 기능적 독립성을 정량화함으로써, 더욱 신뢰성 있고 유지보수가 쉬운 컴포넌트를 만들기 위함이다.

3. 컴포넌트의 특성을 반영한 컴포넌트 메트릭스

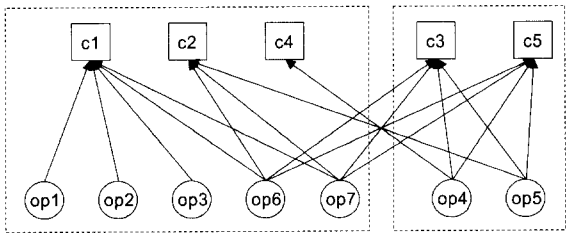
3.1 개 요

컴포넌트는 개별 특성을 지닌 소프트웨어 조각으로, 재사용성과 대치성을 목적으로 인터페이스를 통해 서비스를 제공하는 소프트웨어 패키지이다[7, 13]. 컴포넌트를 사용하는 클라이언트는 컴포넌트가 지원하는 인터페이스만 알면 컴포넌트가 제공하는 모든 서비스를 사용할 수 있다. 또한, 인터페이스가 유지되는 한 컴포넌트 내부의 설계는 얼마든지 변할 수 있으며, 같은 서비스를 제공하는 다른 컴포넌트로 대치될 수도 있다. 인터페이스는 컴포넌트가 제공하는 서비스에 대한 정의로, 컴포넌트를 조립, 사용, 식별하기 위한 표준을 제시한다[6]. 컴포넌트의 형태나 구현 방식은 다를 수 있지만 반드시 인터페이스를 기반으로 컴포넌트가 서로 조립되고 상호 작용을 한다는 점은 공통이다. 컴포넌트는 인터페이스를 통해서 접근할 수 있으며 내부 구성은 하나 이상의 클래스로 구성된다. 인터페이스는 컴포넌트의 서비스를 추상화시켜 소프트웨어 단위인 컴포넌트를 외부로 보이게 하는 역할과 컴포넌트 내부와 외부를 연결하는 매개자 역할을 한다. 인터페이스의 기능은 구체적으로 함수의 역할을 하는 오퍼레이션의 집합으로 제공된다[6, 7]. 외부로부터 컴포넌트의 서비스에 접근할 수 있도록 하기 위한 메소드의 일종인 오퍼레이션은 이름과 매개변수 목록을 가지며, 컴포넌트 내부나 외부의 클래스와의 상호 작용을 통해 기능을 제공한다[4, 6].

[정의 1] 컴포넌트 기반 시스템의 컴포넌트 분석 모델에 대한 컴포넌트의 구성 요소와 그들 사이의 상호 작용을 보여주는 컴포넌트 구성 그래프 COG(Component Organization Graph)는 다음과 같이 정의한다.

$COG = (V, E)$는 컴포넌트 기반 시스템의 컴포넌트들이 기능을 수행하기 위한 컴포넌트 구성 요소들 사이의 상호 작용을 나타내는 방향성 있는 그래프(directed graph)이다. 여기서, $V = \{v_1, v_2, \cdots, v_n\}$는 정점의 유한 집합이고, $E = \{e_1, e_2, \cdots, e_n\}$는 간선의 유한 집합이다. 정점의 유한 집합은 $V = V_c \cup V_o$로, V_c는 컴포넌트 설계에 나타난 클래스를

표현하는 정점들의 집합으로 사각형으로 그래프에 표현한다. V_o는 컴포넌트 설계에 나타난 인터페이스를 구성하는 오퍼레이션의 집합으로, 원으로 그래프에 표현한다. 간선의 유한 집합은 오퍼레이션이 사용하는 클래스들에 대한 방향이 있는 실선으로, $E=\{(x,y)\mid x\in V_o, y\in V_c, x\ uses\ y\}$이다. 그리고 시스템의 개별 컴포넌트는 정점과 간선을 둘러싸는 외곽의 점선 사각형으로 그래프에 표현한다.



(그림 2) 컴포넌트 구성 그래프

(그림 2)는 본 논문에서 제안한 컴포넌트 메트릭스를 적용하여 컴포넌트 설계의 응집도와 결합도를 측정하기 위한 사례 연구의 호텔 예약 시스템에 대한 컴포넌트 구성 그래프이다. 호텔 예약 시스템은 5개의 클래스로 구성되며, 2가지 인터페이스를 통해 기능을 제공하는 두개의 컴포넌트로 구성되는 시스템이다. 그래프에서 사각형은 클래스를 표현하고, 원은 오퍼레이션을 표현하며, 실선은 오퍼레이션이 클래스를 사용하는 것을 나타내며, 외곽의 점선 사각형은 컴포넌트를 나타낸다. 컴포넌트의 응집도와 결합도를 측정하기 위해 시스템 분석 모델로부터 측정에 필요한 요소만으로 간단히 표현한 그래프를 사용함으로써, 메트릭스의 적용 과정을 좀더 객관적이며 쉽게 보일 수 있다.

[정의 2] 컴포넌트의 서비스를 제공하기 위한 컴포넌트 인터페이스와 클래스들 사이의 상호 작용은 오퍼레이션 사용도 행렬 OUM(Operation Use Matrix)으로 표현된다.

$OUM(op_i,\ op_j)$
$= op_i$와 op_j가 공통으로 사용하는 클래스 수, if $i\neq j$
$= op_i$가 사용하는 클래스 수, if $i=j$

OUM은 컴포넌트 인터페이스가 컴포넌트의 서비스를 제공하기 위해, 서로 다른 오퍼레이션들이 공통으로 사용하는 클래스가 어느 정도인지를 나타내는 행렬이다. 행렬의 행과 열은 인터페이스를 구성하는 오퍼레이션의 집합으로, 정방 행렬(square matrix)이며 대칭 행렬(symmetry matrix)이다. 행렬의 대각 원소(diagonal elements) 부분은 오퍼레이션 자신이 사용하는 클래스 수를 나타내며, 비대각 원소(off-diagonal elements) 부분은 서로 다른 오퍼레이션 쌍이 컴포넌트의 서비스를 수행하기 위해 공통으로 사용하는 클래스의 수를 나타낸다. 행렬의 한 행은 임의의 오퍼레이션이 컴포넌트의 서비스를 제공하기 위해 다른 오퍼레이션들과 공통으로 사용하는 클래스의 양을 나타내는 행벡터로 이해할 수 있다. 행렬의 행벡터는 특정 오퍼레이션이 다른 오퍼레이션들과 공통으로 사용하는 클래스의 많고 적음을 직관적으로 분별할 수 있어, 오퍼레이션의 동작 유형을 파악할 수 있다. (그림 3)은 사례 연구인 호텔 예약 시스템의 OUM이다.

	1	2	3	4	5	6	7
1	1	1	1	0	0	1	1
2	1	1	1	0	0	1	1
3	1	1	1	0	0	1	1
4	0	0	0	3	2	2	2
5	0	0	0	2	3	3	3
6	1	1	1	2	3	4	4
7	1	1	1	2	3	4	4

(그림 3) 오퍼레이션 사용도 행렬

[정의 3] 오퍼레이션의 동작 유형을 나타내는 오퍼레이션 행벡터의 코사인(cosine) 값은 두 오퍼레이션의 동작 유형의 유사도(similarity)를 나타내므로, 이로써 오퍼레이션 유사도 행렬 OSM(Operation Similarity Matrix)을 나타낸다.

$$OSM(op_i, op_j)=\frac{\sum_{k=1}^{t} v_{ik}\times v_{jk}}{\sqrt{\sum_{k=1}^{t} v_{ik}^2}\times\sqrt{\sum_{k=1}^{t} v_{jk}^2}}$$

오퍼레이션 사용도 행렬의 i 번째 오퍼레이션인 op_i에 대한 행벡터는 $OUM[op_i]=(v_{i1}, v_{i2}, \cdots, v_{it})$이며, v_{it}는 i 번째 행벡터의 t번째 값을 의미한다. 오퍼레이션 사용도 행렬의 각 행은 오퍼레이션의 클래스에 대한 상호작용인 동작 유형을 나타내고, 서로 다른 오퍼레이션의 벡터 값에 대한 코사인 값은 두 오퍼레이션의 동작 유형의 유사도를 나타내므로, 그 값으로 오퍼레이션 유사도 행렬을 나타낸다. OSM의 행과 열은 오퍼레이션 수로 정방 행렬이고, 행렬의 값은 두 오퍼레이션의 유사도인 코사인 값으로 0에서 1사이의 값을 갖는다. 행렬의 대각 원소 부분은 동일한 오퍼레이션 사이의 유사도 값으로 1이므로, OSM은 대칭 행렬이다. (그림 4)는 (그림 3)의 OUM으로부터의 사례 연구에 대한 OSM이다.

	1	2	3	4	5	6	7
1	1.000	1.000	1.000	0.390	0.482	0.710	0.710
2	1.000	1.000	1.000	0.390	0.482	0.710	0.710
3	1.000	1.000	1.000	0.390	0.482	0.710	0.710
4	0.390	0.390	0.390	1.000	0.941	0.882	0.882
5	0.482	0.482	0.482	0.941	1.000	0.959	0.959
6	0.710	0.710	0.710	0.882	0.959	1.000	1.000
7	0.710	0.710	0.710	0.882	0.959	1.000	1.000

(그림 4) 오퍼레이션 유사도 행렬

3.2 컴포넌트 응집도

외부에서 컴포넌트의 서비스에 접근하기 위한 매개자 역할을 하는 인터페이스는 컴포넌트 내부의 클래스들과의 상호작용을 통해 서비스를 제공한다. 컴포넌트의 기능 독립성은 인터페이스가 컴포넌트 내부의 클래스들과의 상호작용을 통한 기능 수행을 통해 제공된다. 만일, 인터페이스의 오퍼레이션이 컴포넌트 내부의 클래스들을 많이 사용한다면, 컴포넌트 내부 구성 요소들의 친화력인 오퍼레이션과 클래스 사이의 연관 정도가 높아져 응집성 높은 컴포넌트가 된다. 좋은 컴포넌트는 인터페이스와 내부 클래스들간에 높은 응집력(highly cohesive)을 가져야 하며, 다른 컴포넌트와는 낮은 결합력(loosely coupled)을 가져야 한다[4, 6, 7].

그러므로 더욱 신뢰성 있고 유지보수 가능한 품질 높은 시스템을 위해, 시스템에 대한 설계의 품질을 측정하기 위한 대표적인 척도인 응집도를 이용해 컴포넌트의 기능 독립성을 파악할 수 있다. 컴포넌트의 인터페이스는 서비스를 제공하기 위해 오퍼레이션이 내부의 클래스들과 상호작용을 하는데, 동작 유형이 서로 유사한 오퍼레이션들과 상호작용을 많이 하는 클래스들로만 컴포넌트가 구성된다면 컴포넌트의 응집도는 높을 것이다.

따라서 오퍼레이션이 컴포넌트 내부의 클래스들을 공통으로 사용하는 정도를 나타내는 동작 유형에 의한 오퍼레이션 유사도에 의해 컴포넌트 응집도를 측정한다. 인터페이스에 의한 컴포넌트 응집도 CCH(Component CoHesion by interface)는 다음과 같다.

[정의 10] $$CCH(Co_i) = \frac{\sum_{i=1}^{n_i}\sum_{j=1}^{n_i} OSM(op_i, op_j)}{n_i \times n_i}$$

이때, 임의의 컴포넌트 Co_i의 오퍼레이션의 전체 수는 n_i이다. i와 j는 임의의 컴포넌트의 오퍼레이션을 나타내는 것으로, $1 \le i, j \le n_i$로 일련번호이다. 따라서 임의의 컴포넌트에 대한 응집도 측정은 컴포넌트 내부에 속한 오퍼레이션들로부터 가능한 모든 오퍼레이션의 쌍에 대한 오퍼레이션 유사도 값의 합을 가능한 오퍼레이션 쌍의 수로 나눈다. OSM의 값은 오퍼레이션 쌍에 대한 코사인 값으로, 0~1사이를 만족하므로, 가능한 오퍼레이션 쌍의 수에 의한 비율(ratio)로 응집도를 구하면, 값은 0~1사이의 범위로 정규화된다. 컴포넌트 내부의 오퍼레이션들이 서로 유사도가 높다면, 동작 유형이 비슷한 오퍼레이션이 한 컴포넌트에 속하게 되는 것이므로 컴포넌트 응집도는 높을 것이다.

3.2.1 응집도 속성에 대한 평가

Briand[14]는 특정 소프트웨어 산출물에만 적용되지 않도록 일반적(general)이며, 정확한 수학적 개념에 근거하여 엄격한(rigorous) 수학적 프레임워크(mathematical framework)를 제안했다. 이 프레임워크는 메트릭에 대해 크기(size), 길이(length), 복잡도(complexity), 응집도(cohesion), 결합도(coupling) 등의 분야에 대한 개념(concept)과 성질(property)을 정의하였다. 이 프레임워크는 새로운 소프트웨어 메트릭을 정의할 때 만족해야할 기준으로 사용가능하며, 소프트웨어 측정(measurement)에 대한 이론적인 기반을 마련했다는 의의가 있다.

따라서 본 논문에서는 Briand가 정의한 프레임워크를 컴포넌트 기반 시스템의 컴포넌트에 적용하여 컴포넌트 응집도와 결합도 메트릭의 이론적 타당성을 검증한다. 먼저 컴포넌트 응집도와 결합도 메트릭스의 이론적 검증을 하기 위해 필요한 용어에 대해 정의한다.

요소인 외부의 클래스와의 상호작용을 나타낸다.

- Co는 컴포넌트 기반 시스템의 임의의 컴포넌트이다.
- E_{Co}는 컴포넌트 Co의 내부 구성 요소로, 클래스 집합 $CSet_i$와 오퍼레이션 op로 구성되는 인터페이스 집합 $ISet_i$로 구성된다.
- R_{Co}는 컴포넌트가 서비스를 수행하기 위한 상호작용 관계를 나타낸다.
- $IntraR_{Co}$는 컴포넌트가 서비스를 수행하기 위한 상호작용 R_{Co}중 자기 자신의 컴포넌트 내부의 구성요소인 클래스와의 상호작용을 나타낸다.
- $InterR_{Co}$는 컴포넌트가 서비스를 수행하기 위한 상호작용 R_{Co}중 다른 컴포넌트의 구성
- $Cohesion(Co)$는 컴포넌트 Co에 대한 응집도 측정값이다.
- $Coupling(Co)$는 컴포넌트 Co에 대한 결합도 측정값이다.
- $InputR(Co_i)$는 다른 모든 컴포넌트로부터 컴포넌트 Co_i로 향하는 수입관계(incoming relationship)를 나타내며, $OutputR(Co_i)$는 컴포넌트 Co_i로부터 다른 모든 컴포넌트로 향하는 수출관계(outgoing relationship)를 나타낸다.
- $\max(value_1, value_2, \cdots, value_n)$는 n개의 값 중 최대값을 의미한다.
- $\Sigma(value_1, value_2, \cdots, value_n)$는 n개의 값들의 총합을 의미한다.

응집도 속성에 대한 이론적 검증은 다음과 같다.

성질 1-1 : Non-negativity and Normalization

$$(\forall Co)\ (0 \le Cohesion(Co) \le 1)$$

응집도는 음수의 값을 가져서는 안되며, 항상 일정한 구간 내에 존재하도록 정규화 되어야 한다는 것을 의미한다. 응집도는 컴포넌트 내부 구성 요소의 관련성을 나타내므로, 컴포넌트의 크기에 비례하는 최대값을 갖는다. 그러나 응집도를 비교하기 위해 동일한 구간에 속하도록 정규화(nor-

malization)를 시키면 컴포넌트의 크기에 독립적이 되어 서로 다른 컴포넌트에 대해 비교 가능하다. 본 논문의 컴포넌트 응집도는 컴포넌트 내 오퍼레이션 쌍에 대한 0에서 1사이의 코사인 값인 오퍼레이션 유사도 값의 합을 가능한 오퍼레이션 쌍의 수로 나누기 때문에, 0에서 1사이의 값으로 정규화 되며 음수의 값을 가질 수 없으므로, 성질 1-1은 만족한다.

성질 1-2 : Null Value

$(\forall Co)\ (\ IntraR_{Co} = \varnothing \Rightarrow Cohesion(Co) = 0\ )$

컴포넌트 내부 구성요소들 사이에 상호작용이 없다면, 응집도는 0 이라는 것을 의미한다. 응집도는 내부 구성요소들 사이의 연관성이 얼마나 견고한지(tight)의 정도를 나타내는 값이므로, 컴포넌트의 구성요소들 사이에 전혀 관련성이 없는 최악의 경우라면 응집도는 0이므로, 성질 1-2는 만족된다.

성질 1-3 : Monotonicity

$(\forall Co)\ (\ E_{Co} = E_{Co'}$ and $IntraR_{Co} \subseteq IntraR_{Co'}$
$\Rightarrow Cohesion(Co) \leq Cohesion(Co')\ )$

컴포넌트 내부 구성요소의 양적 변화 없이 상호작용만 증가한다면, 하나의 모듈로 통합(encapsulation)되기 위한 관계의 증가를 의미하므로, 응집도는 감소하지 않는다는 것을 의미한다. 컴포넌트의 설계에서 인터페이스와 클래스 사이의 관계가 추가되면, 하나의 컴포넌트로 캡슐화 되기 위한 추가 정보이므로 응집도는 감소하지 않는 것을 나타내므로, 성질 1-3은 만족한다.

성질 1-4 : Cohesive Components

$(\forall Co, \forall Co_1, \forall Co_2)$

$(\ Co = Co_1 \cup Co_2$ and $InputR(Co_1) \cap OutputR(Co_2) = \varnothing$
and $InputR(Co_2) \cap OutputR(Co_1) = \varnothing \Rightarrow$
$\max\{Cohesion(Co_1), Cohesion(Co_2)\} \geq Cohesion(Co)\ )$

컴포넌트 사이에 상호 작용이 전혀 없는 두 컴포넌트를 합친다면 응집도는 원래 두 컴포넌트의 최대 응집도보다 크지 않다는 것을 의미한다. 서로 관련이 없는 두 컴포넌트를 하나로 통합할 경우, 관련 없는 요소들이 함께 통합되어 컴포넌트의 구성 요소들 사이에 분리 집합이 생기게 된다. 이는 구성 요소들이 하나의 컴포넌트로 통합되기 위한 관계의 감소를 의미한다. 컴포넌트가 하나로 통합되면 컴포넌트의 크기는 커지나 하나로 통합된 컴포넌트에서 인터페이스가 내부 클래스와의 상호 작용은 증가하지 않으므로, 응집도는 이전의 최대 응집도보다 증가하지 않는다는 것을 의미하므로, 따라서 성질 1-4는 만족한다.

3.3 컴포넌트 결합도

컴포넌트 기반 시스템은 독립된 기능 단위인 컴포넌트를 조합하여 좀 더 큰 서비스를 제공하는 시스템을 구성하게 된다. 조합되는 컴포넌트들은 인터페이스를 통한 컴포넌트들 사이의 상호작용에 의해 시스템의 서비스를 제공한다.

바람직한 컴포넌트 설계는 인터페이스와 내부 클래스들 사이에는 높은 응집력을, 다른 컴포넌트와는 낮은 결합력을 가져야 한다[4, 6, 7]. 그러므로 더욱 신뢰성 있고 유지보수가 용이한 품질 높은 시스템을 위해, 시스템에 대한 설계의 품질을 측정하기 위한 대표적인 척도인 결합도를 이용해 컴포넌트의 기능 독립성을 파악할 수 있다. 독립된 기능 단위인 컴포넌트가 되기 위해서 다른 컴포넌트와의 상호 작용은 적어야 하므로, 인터페이스가 다른 컴포넌트와 상호 작용을 많이 할수록 컴포넌트의 결합도는 높다할 수 있으며, 이는 바람직하지 못한 컴포넌트이다.

따라서 컴포넌트 인터페이스의 오퍼레이션과 다른 컴포넌트 인터페이스의 오퍼레이션들과의 상호 작용을 나타내는 동작 유형을 통한 오퍼레이션 유사도에 의해 컴포넌트의 결합도를 측정한다. 인터페이스에 의한 컴포넌트 결합도 CCP (Component CouPling by interface)는 다음과 같다.

[정의 11]

$$CCP(Co_i) = \left(\sum_{i=N1+1}^{N2} \sum_{j=1}^{l} OSM(op_i, op_j) - \sum_{i=N1+1}^{N2} \sum_{j=N1+1}^{N2} OSM(op_i, op_j) \right) \times \frac{1}{n_i(l-n_i)}$$

임의의 컴포넌트 Co_i에 n_i개의 오퍼레이션이 포함되고, 시스템 전체의 오퍼레이션 수 l은 각 컴포넌트의 오퍼레이션 수의 합으로 $l = n_1 + n_2 + \cdots + n_m$이다. $N1 = \sum_{r=1}^{i} n_{r-1}$로 순서상 현재 컴포넌트 이전까지의 오퍼레이션 수의 합이고, $N2 = \sum_{r=1}^{i} n_r$로 순서상 현재 컴포넌트까지의 오퍼레이션 수의 합이다. 따라서 컴포넌트의 결합도 측정은 현재 컴포넌트의 오퍼레이션으로부터 다른 컴포넌트에 속한 모든 오퍼레이션의 쌍에 대한 오퍼레이션 유사도 값의 합을 가능한 쌍의 수로 나눈다. 현재 오퍼레이션으로부터 다른 모든 오퍼레이션에 대한 유사도 값의 합은 두 오퍼레이션 사이의 직접적인 상호작용과 직접 상호작용을 하는 오퍼레이션을 통한 간적적인 상호작용을 모두 포함하므로, 결합도 측정 공식의 분자 부분에 포함한다. OSM의 값은 오퍼레이션 쌍에 대한 코사인 값으로 0 ~1사이를 만족하므로, 결합도 측정값도 0~1사이의 범위로 정규화 된다. 그러므로 서로 다른 컴포넌트 사이의 결합도를 비교할 수 있으며, 그 값이 작을수록 기능적으로 독립적인 컴포넌트가 되어 바람직한 설계가 된다. 한 컴포넌트의 오퍼레이션과 다른 컴포넌트들의 오퍼레이션 사이의 동작 유형인 오퍼레이션의 유사도가 높을수록, 오퍼레이션의 상호작용인 동작 유형이 유

사하므로 컴포넌트의 결합도는 높다 할 수 있다. 그러나 컴포넌트간의 오퍼레이션의 동작 유형이 서로 상이하다면, 컴포넌트가 서로 관련이 없이 독립적으로 동작을 하게 되는 것을 의미하므로, 컴포넌트 결합도는 낮다고 할 수 있다.

3.3.1 결합도 속성에 대한 평가

결합도 속성에 대한 이론적 검증은 다음과 같으며, 결합도 메트릭스의 이론적 평가를 위해 필요한 용어에 대한 정의는 앞 절에서 설명했으므로 생략한다.

성질 2-1 : Non-negativity

$(\forall Co)\ (\ Coupling(Co) \geq 0\ )$

결합도 측정값은 음수가 될 수 없다는 것을 의미한다. 결합도는 컴포넌트와 컴포넌트 사이의 관련성을 나타내는 값으로, 하나의 컴포넌트는 다른 컴포넌트와 상호작용을 하거나 하지 않을 수 있으므로, 결합도는 음수의 값을 가질 수 없다. 따라서 성질 2-1은 만족한다.

성질 2-2 : Null Value

$(\forall Co)\ (\ InterR_{Co} = \emptyset \Rightarrow Coupling(Co) = 0\ )$

컴포넌트들 사이에 상호작용이 없다면 결합도는 0 이라는 것을 의미한다. 결합도는 컴포넌트간의 상호작용의 정도를 나타내는 값으로, 컴포넌트가 다른 컴포넌트와 상호작용을 전혀 하지 않을 수도 있으므로, 성질 2-2는 만족한다.

성질 2-3 : Monotonicity

$(\forall Co)\ (\ E_{Co} = E_{Co'}$ and $InterR_{Co} \subseteq InterR_{Co'}$
$\Rightarrow Coupling(Co) \leq Coupling(Co')\ )$

컴포넌트 내부 구성 요소에는 변화 없이, 컴포넌트들 사이의 상호작용만 증가한다면, 결합도는 감소하지 않는다는 것을 의미한다. 컴포넌트의 외부 구성 요소에 대한 상호작용인 컴포넌트 사이의 상호 작용만 증가한다면, 컴포넌트 사이의 상호작용의 정도를 나타내는 결합도는 감소하지 않으므로, 성질 2-3은 만족한다.

성질 2-4 : Merging of Components

$(\forall Co, \forall Co_1, \forall Co_2)$

$(\ Co = Co_1 \cup Co_2$ and $E_{Co} = E_{Co_1} \cup E_{Co_2}$
and $InterR_{Co} = InterR_{Co_1} \cup InterR_{Co_2} \Rightarrow$
$\Sigma\{Coupling(Co_1), Coupling(Co_2)\} \geq Coupling(Co)\ )$

두 컴포넌트를 결합하여 새로운 컴포넌트를 만든다면, 새로운 컴포넌트의 결합도는 두 컴포넌트의 결합도의 합보다 증가하지는 않는다는 것을 의미한다. 두 컴포넌트가 하나로 합쳐진다면 컴포넌트간의 상호작용 중 일부는 컴포넌트 내부의 상호작용으로 포함될 수 있으므로, 컴포넌트 사이의 상호작용의 정도를 나타내는 결합도는 두 컴포넌트의 결합도의 합보다 증가하지는 않는다. 따라서 성질 2-4는 만족한다.

성질 2-5 : Disjoint Component Additivity

$(\forall Co, \forall Co_1, \forall Co_2)$

$(\ Co = Co_1 \cup Co_2$ and $InputR(Co_1) \cap OutputR(Co_2) = \emptyset$
and $InputR(Co_2) \cap OutputR(Co_1) = \emptyset \Rightarrow$
$\Sigma\{Coupling(Co_1), Coupling(Co_2)\} = Coupling(Co)\ )$

컴포넌트간 상호작용이 전혀 없는 두 컴포넌트를 결합하여 새로운 컴포넌트를 만든다면, 결합도는 원래 두 컴포넌트의 결합도 합과 같다는 것을 의미한다. 두 컴포넌트 사이에 상호작용이 전혀 없기 때문에 두 컴포넌트를 하나로 결합할 경우 컴포넌트 사이의 상호작용은 그대로 유지되고 변화가 없으므로, 기존 컴포넌트의 결합도의 합과 동일하다. 따라서 성질 2-5는 만족한다.

4. 사례 연구 및 비교 연구

호텔 예약 시스템은 고객이 호텔에 대한 정보를 검색하고, 객실 예약 가능 여부의 확인을 통해 호텔 객실을 예약하기 위한 시스템이다. 호텔 예약 시스템의 컴포넌트 설계 모델에 대해 응집도와 결합도를 측정해 보겠다. 또한, 컴포넌트 메트릭스의 필요성 및 유효성을 확인하기 위해 객체지향 메트릭스와의 비교 연구를 하겠다.

컴포넌트 설계 모델인 (그림 1)의 인터페이스 명세 다이어 그램으로부터 추출 가능한 정보로부터 컴포넌트 구성 그래프를 그리면 (그림 2)이고, 오퍼레이션 사용도 행렬은 (그림 3)이며, 오퍼레이션 유사도 행렬을 구하면 (그림 4)이다. 오퍼레이션 유사도 행렬의 각 오퍼레이션에 대한 유사도 값에 기반 하여 컴포넌트 응집도와 결합도를 측정한 결과는 <표 1>과 같다. 설명의 편의를 위해, (그림 2)의 컴포넌트 구성 그래프의 왼쪽부터 컴포넌트 A, B라 명명하겠다. 측정 결과 컴포넌트 B가 응집도가 높고 결합도가 낮은 이상적인 컴포넌트로, 컴포넌트의 서비스를 나타내는 인터페이스가 유사도가 높은 오퍼레이션들로 구성되어 있음을 알 수 있다.

〈표 1〉 컴포넌트 응집도 결합도 측정 결과

	컴포넌트 A	컴포넌트 B	평 균
응집도	0.884	0.980	0.932
결합도	0.630	0.630	0.630

본 논문에서 제안한 컴포넌트 메트릭스의 유용성을 증명하기 위해 컴포넌트 설계 모델에 객체지향 메트릭스를 그대로 적용하여 응집도와 결합도를 계산해 본다. 컴포넌트 메트릭스의 계산 결과와 객체지향 메트릭스의 계산 결과를 비교함으로써, 컴포넌트의 특성을 반영한 컴포넌트 메트릭스의 필요성 및 효용성을 다시 확인하고자 한다. 객체지향 메트릭스 중 가장 대표적인 Chidamber와 Kermerer[9]가 제안한 메트릭스 중 응집도 메트릭과 결합도 메트릭으로 LCOM (Lack of Cohesion in Methods)과 CBO(Coupling Between

Object classes)를 택해, 컴포넌트 설계에 적용하여 응집도와 결합도를 측정하였으며, 측정 결과는 <표 2>와 같다. 객체지향 메트릭스를 컴포넌트에 적용하여 측정해 본 결과, 각 컴포넌트에 대한 응집도와 결합도의 구별이 드러나지 않으며, 정규화가 되지 않아 다른 크기의 컴포넌트에 대해 비교하기가 불가능함을 알 수 있었다.

〈표 2〉 객체지향 응집도 결합도 측정 결과

	컴포넌트 A	컴포넌트 B	평 균
응집결여도	0	0	0
결 합 도	0	2	2

5. 결 론

본 논문에서는 컴포넌트 기반 시스템의 컴포넌트 설계 정보에 기반한 컴포넌트 응집도와 결합도 메트릭스를 제안했다. 모듈 내부 구성요소들의 관련성 정도를 측정하는 응집도와 두 모듈간의 상호 의존도를 측정하는 결합도를 컴포넌트에 적용하여, 컴포넌트가 재사용 단위로서 얼마나 독립적으로 컴포넌트의 기능을 수행하는 지를 측정한다.

컴포넌트는 클래스와 인터페이스의 묶음으로, 인터페이스는 컴포넌트의 서비스를 제공하기 위해 컴포넌트 내부나 외부의 클래스들과 상호작용을 하게 되므로, 오퍼레이션이 서비스를 제공하기 위해 공통으로 사용하는 클래스들의 정보를 오퍼레이션 사용도 행렬로 표현한다. 오퍼레이션 사용도 행렬의 각 행은 오퍼레이션의 동작 유형을 나타내므로, 모든 오퍼레이션들의 쌍에 대해 오퍼레이션의 동작 유형이 어느 정도 유사한지에 의해 오퍼레이션 유사도 행렬을 표현한다. 컴포넌트 내부의 오퍼레이션에 대해 오퍼레이션 유사도 행렬 값에 의해 컴포넌트 응집도를 구하며, 직접적 또는 간접적으로 상호작용을 하는 다른 컴포넌트의 오퍼레이션에 대한 오퍼레이션 유사도 행렬 값에 의해 컴포넌트 결합도를 구한다. 본 논문에서 제안한 컴포넌트 메트릭스는 컴포넌트 기반 시스템 개발 초기 단계에서의 분석 정보를 사용하여 컴포넌트의 응집성과 결합성을 측정하여, 바람직한 컴포넌트 설계인지를 판단한다.

참 고 문 헌

[1] Charles W. Krueger, "Software Reuse," ACM Computing Surveys, ACM Press, Vol.24, No.2, pp.131-183, June, 1992.

[2] 한·카네기멜론대학 기술교류협회, 최신 소프트웨어 공학 기법, V. I. LAND, 2002.

[3] Roger S. Pressman, Software Engineering : A Practitioner's Approach, 5th Edition, McGraw-Hill, June, 2000.

[4] Colin Atkinson, Joachim Bayer, Christian Bunse, Erik Kamsties, Oliver Laitenberger, Roland Laqua, Dirk Muthig, Barbara Peach, Jurgen Wust, Jorg Zettel, Component-Based Product Line Engineering with UML, Addison-Wesley, 2002.

[5] George T. Heineman, William T. Councill, Component-Based Software Engineering : Putting the Pieces Together, Addison-Wesley, 2001.

[6] John Cheesman, John Daniels, UML Components : A Simple Process for Specifying Component-Based Software, Addison-Wesley, 2001.

[7] Clemens Szyperski, Dominik Gruntz, Stephan Murer, Component Software : Beyond Object-Oriented Programming, 2nd Edition, Addison-Wesley, 2002.

[8] Mark Lorenz, Jeff Kidd, Object-Oriented Software Metrics : A Practical Guide, Prentice Hall, 1994.

[9] Shyam R. Chidamber, Chris F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Transactions on Software Engineering, Vol.20, No.6, pp.476-493, June, 1994.

[10] E. S. Cho, M. S. Kim, S. D. Kim, "Component Metrics to Measure Component Quality," pp.419-426, Eighth Asia-Pacific Software Engineering Conference (APSEC'01), 2001.

[11] H. H. Kim and D. H. Bae, "Component Identification via Concept Analysis," Journal of Object Oriented Programming, 2001.

[12] 최미숙, 분석 클래스 기반의 컴포넌트 식별 메트릭스와 컴포넌트 식별 방법에 관한 연구, 숙명여자대학교 박사학위논문, 2002.

[13] Peter Herzum, Oliver Sims, Business Component Factory : A Comprehensive Overview of Component-Based Development for the Enterprise, John Wiley & Sons, 1999.

[14] Lionel Briand, Sandro Morasca, Victor Basili, "Property-based Software Engineering Measurement," IEEE Transactions on Software Engineering, Vol.22, No.1, pp.68-86, January, 1996.

고 병 선

e-mail : kobs@sookmyung.ac.kr

1995년 숙명여자대학교 컴퓨터과학과 졸업(학사)

1998년 숙명여자대학교 일반대학원 컴퓨터과학과(석사)

2003년 숙명여자대학교 일반대학원 컴퓨터과학과 박사학위 취득(박사 졸업예정)

관심분야 : 재사용, 메트릭스, 객체지향 프로그래밍

박 재 년

e-mail : jnpark@sookmyung.ac.kr

1966년 고려대학교 졸업(학사)

1969년 고려대학교 이학석사

1981년 고려대학교 이학박사

1979년~1983년 전남대학교 전산통계학과 교수

1983년~현재 숙명여자대학교 정보과학부 교수

관심분야 : 시스템 개발 방법론, 모델링, 시뮬레이션, 메타 DB