컴포넌트 메트릭스를 이용한 컴포넌트 설계 재정비
(Improvement of Component Design using Component Metrics)

고 병 선 † 박 재 년 ††
(Byung-Sun Ko) (Jai-Nyun Park)

요 약 컴포넌트 기반 개발 방법론은 클래스보다 더 큰 단위인 컴포넌트를 통해 높은 추상화와 재사용을 목표로 하는 개발 방법론이다. 컴포넌트 기반 시스템과 개별 컴포넌트의 품질 향상을 위해서는 개발 이전에 측정하고, 그 결과를 컴포넌트 개발 과정에 반영할 수 있어야 한다. 그러므로 컴포넌트 분석 및 설계 단계에 적용 가능한 컴포넌트 메트릭에 대한 연구가 필요하다.

따라서 본 논문에서는 컴포넌트 설계 정보에 기반한 컴포넌트 응집도, 결합도, 독립도 메트릭을 제안한다. 제안한 메트릭은 컴포넌트의 서비스를 제공하기 위한 오퍼레이션들의 동작 유형의 유사도에 기반 한다. 또한, 기능적으로 응집도가 높으며 복잡도가 낮고 유지 보수가 용이한 컴포넌트 설계가 되도록 하기 위한 클러스터링 기법을 사용한 컴포넌트 재설계 과정을 제안한다. 그리고 개발 이전에 컴포넌트 측정과 컴포넌트 재설계 과정을 통해 바람직한 컴포넌트 설계가 되도록 할 수 있음을 사례 연구를 통해 확인했다.

키워드 : 컴포넌트 인터페이스, 응집도, 결합도, 독립도, 재설계

Abstract The component-based development methodology aims at the high state of abstraction and the reusability with components larger than classes. It is indispensible to measure the component so as to improve the quality of the component-based system and the individual component. And, the quality of the component should be improved through putting the results into the process of the development. So, it is necessary to study the component metric which can be applied in the stage of the component analysis and design.

Hence, in this paper, we propose component cohesion, coupling, independence metrics reflecting the information extracted in the step of component analysis and design. The proposed component metric bases on the similarity information about behavior patterns of operations to offer the component's service. Also, we propose the redesigning process for the improvement of component design. That process uses the techniques of clustering and is for the thing that makes the component as the independent functional unit having the low complexity and easy maintenance. And, we examine that the component design model can be improved by the component metrics and the component redesigning process.

Key words : component interface, cohesion, coupling, independence, redesign

1. 서 론

소프트웨어를 개발함에 있어서 가장 중요한 목표 중 하나는 생산성이고, 다른 하나는 품질이다. 재사용 기술은 소프트웨어 개발의 가장 중요한 목표인 높은 개발 생산성과 품질을 위해, 소프트웨어도 하드웨어가 공장에서 규격화되어 생산되는 것과 같이 규격화하여 하드웨어 부품처럼 기능의 조각을 조립하여 재사용 한다면, 단기간에 저비용으로 고품질의 소프트웨어를 생산할 수 있다고 생각되었다[1].

정보 시스템 개발의 새로운 대안인 컴포넌트 기반 개발 방법론은 클래스보다 더 큰 단위인 컴포넌트를 통해 높은 추상화(abstraction)와 재사용을 목표로 하는 개발 방법론이다[2,3]. 컴포넌트 기반 개발 방법론은 소프트웨어도 하드웨어 부품처럼 조립이 가능해 단기간에 저비용으로 고품질의 소프트웨어를 생산할 수 있다는 것이다. 컴포넌트는 복잡한 데이타와 동작은 내부에 숨기고 외부에 인터페이스만 분리되어, 사용자는 인터페이스를 통해서만 컴포넌트에 접근하여 사용할 수 있다. 이러한 컴포넌트의 캡슐화 특성은 에러의 영향을 제한시켜 유

† 비 회 원 : 숙명여자대학교 정보과학부
kobs@sookmyung.ac.kr
†† 종신회원 : 숙명여자대학교 정보과학부 교수
jnpark@sookmyung.ac.kr
논문접수 : 2003년 4월 18일
심사완료 : 2004년 6월 17일

지 보수를 용이하게 하는 장점을 갖는다.

따라서 본 논문에서는 컴포넌트의 구조적 특성을 반영한 컴포넌트 메트릭을 제안한다. 제안된 메트릭은 컴포넌트 기반 시스템의 개발 초기 단계인 분석 및 설계 단계에서 추출될 수 있는 컴포넌트에 대한 정보인 인터페이스와 클래스 사이의 상호작용에 대한 정보만을 이용하여 컴포넌트 응집도, 결합도 그리고 독립도를 측정하도록 정의하였다. 제안된 컴포넌트 메트릭은 개발 이후의 측정이 아닌 개발 이전의 측정으로, 상대적으로 빨리 컴포넌트에 대한 품질을 평가하여 이후 컴포넌트의 개발에 반영할 수 있다는 장점을 갖는다. 그리고 보다 기능적으로 응집도가 높으며 독립된 기능 단위로 복잡도가 낮고 유지 보수가 용이하며 재사용성이 높은 컴포넌트 설계가 되기 위해, 클러스터링 기법을 이용한 컴포넌트 재설계 과정도 제안한다. 재설계 과정을 통해 컴포넌트의 설계 품질을 향상시켜, 유지 보수성, 재사용성, 기능 독립성이 향상된 컴포넌트 설계가 되고자 한다.

2. 연구 배경

2.1 컴포넌트 기반 소프트웨어 개발 절차

컴포넌트 기반 시스템 개발은 두 단계의 개발 과정으로 구성된다. 첫째는 컴포넌트 자체를 개발하는 것이고, 둘째는 개발된 컴포넌트를 이용하여 컴포넌트 기반의 소프트웨어를 개발하는 것이다. 그림 1은 컴포넌트 자체의 개발과 개발된 컴포넌트를 조합하여 시스템을 구축하는 컴포넌트 기반 소프트웨어 개발을 나타낸다. 컴포넌트 자체의 개발을 위한 단계는 컴포넌트 기반 시스템의 구성 요소가 될 독립적인 기능 단위인 개별 컴포넌트를 개발하는 단계이다. 주어진 업무 영역을 분석하여 개념 모델링을 한 후, 컴포넌트 모델을 설계한다. 그리고 컴포넌트 모델에 기초하여 컴포넌트를 구현, 테스트하여 개발된 컴포넌트를 인증하는 단계로 구성된다. 이

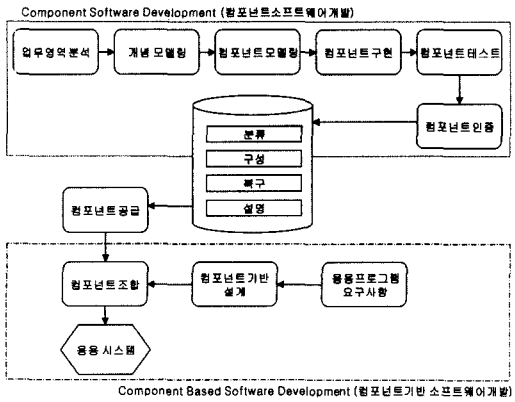

그림 1 컴포넌트 기반 소프트웨어 개발

러한 과정을 컴포넌트 소프트웨어 개발 CSD(Component Software Development)이라 한다. 그리고 개발하고자 하는 응용 시스템의 요구사항을 만족시키기 위해 이미 개발되어 배포된 컴포넌트를 조립하여 시스템을 개발하는 과정을 컴포넌트 기반 소프트웨어 개발 CBSD (Component Based Software Development)이라 한다. 본 논문에서는 품질 좋은 컴포넌트 기반 소프트웨어를 개발하기 위한 초석이 되는 개별 컴포넌트 개발에 초점을 맞춘다.

다음은 개별 컴포넌트 소프트웨어 개발에 대해 살펴보겠다. 개별 컴포넌트 소프트웨어 개발을 위한 업무 영역별 작업 흐름은 전통적인 소프트웨어 개발 단계의 요구사항 분석, 분석, 설계로 구성되는 아키텍쳐 명세 영역인 전반부와 설계의 후반부, 구현, 테스트로 구성되는 컴포넌트 개발 영역인 후반부의 두 부분으로 구성된다 [4]. 요구사항 분석 단계는 개발될 시스템의 요구사항을 도출하고 이해하기 위해 기능 중심의 유즈케이스(use case) 모델을 통해 시스템과 엑터(actor)가 제공하는 기능을 파악한다. 분석 단계에서는 시스템을 구성하는 컴포넌트들의 역할과 그들 간의 관계가 컴포넌트 아키텍쳐로 정의된다. 설계 단계에서는 정의된 시스템의 구조와 행위를 어떻게 효율적으로 실현할 것인지 결정하며, 이는 컴포넌트 구조 설계와 컴포넌트 인터페이스 설계의 두 부분으로 구성된다. 컴포넌트 구조 설계는 요구사항 분석의 결과를 토대로 업무 영역의 클래스들을 서로 연관되고 의존하는 것끼리 중요한 클래스를 중심으로 높은 응집력을 갖도록 컴포넌트로 분할하고, 각 컴포넌트들이 갖게 될 인터페이스를 정의한다. 컴포넌트 인터페이스 설계는 도출된 컴포넌트의 역할과 관계를 기반으로 인터페이스를 구성하는 오퍼레이션들을 상세히 정의하며, 각 오퍼레이션에 대해 입력 인자와 출력 값의 형(type) 뿐만 아니라 오퍼레이션의 결과를 정의하는 선행/후행 조건을 기술하는 컴포넌트 명세를 한다. 구현 및 테스트 단계는 구현 환경에 대한 고려를 통해 컴포넌트를 구현하고, 구현된 컴포넌트가 요구사항을 만족시키는지 테스트하게 된다.

2.2 컴포넌트 메트릭

컴포넌트 기술은 새로운 기술이 아닌 객체지향 기술을 기반으로 재사용이라는 목적을 달성하기 위해 발전된 기술이기에, 많은 부분에 있어 객체지향 기술의 영향을 받고 있는 게 사실이다. 그래서 아직까지 컴포넌트 메트릭에 대한 연구가 미흡하고, 기존의 객체지향 메트릭을 그대로 컴포넌트에 적용하거나 약간의 수정 후 사용하는 경우가 대부분이다.

그러나 컴포넌트는 클래스와는 다른 구조적 특성을 갖는다. 컴포넌트 기반 시스템은 하나 이상의 컴포넌트

로 구성되며, 컴포넌트는 서비스 제공을 위해 밀접한 연관성이 있는 클래스들과 인터페이스로 구성된다. 컴포넌트가 제공하는 서비스인 인터페이스는 구체적으로는 기능 수행을 위한 오퍼레이션으로 구성되며, 인터페이스의 오퍼레이션들은 컴포넌트 내부나 외부의 클래스들을 생성, 수정, 삭제, 참조하는 상호 작용을 통해 컴포넌트의 서비스를 수행하게 된다. 그러므로 객체지향 메트릭을 컴포넌트 기반 시스템의 컴포넌트에 적용하여 품질을 측정하는 것은 부적합하므로, 컴포넌트가 갖는 구조적인 특성과 클래스와는 다른 측정 인자를 반영한 새로운 컴포넌트 메트릭이 필요하다[2].

컴포넌트 메트릭에 대한 기존 연구는 다음과 같다.

Kim[5]의 메트릭은 컴포넌트 품질을 측정하기 위한 메트릭이라기 보다는 클래스들 간의 관계에 의해 컴포넌트를 식별하기 위한 수단으로서의 메트릭이라 할 수 있다. Cho[6]의 메트릭은 컴포넌트의 구조적인 특성을 반영하기는 했으나 컴포넌트 고유의 특성보다는 클래스의 특성을 많이 반영한 객체지향적 메트릭이라는 성격이 강하다. 또한, 측정을 위해 수집해야할 정보가 많아 사용이 용이하지 않으며, 컴포넌트 메트릭의 측정 결과와 컴포넌트의 품질간의 관계에 대한 설명이 미흡하다는 아쉬움이 남는다. 그리고 Choi[7]의 메트릭은 도메인 영역의 분석 클래스를 기반으로 분석 클래스에 인터페이스의 사용 의미에 따른 가중치를 준 컴포넌트 응집도와 결합도를 이용해 독립적인 컴포넌트 식별을 위한 메트릭이다. 이는 본 논문의 사례 연구에서 비교 연구로 사용하고자 하므로, 컴포넌트 응집도 메트릭과 결합도 메트릭에 대해 자세히 살펴본다.

시스템의 임의의 컴포넌트 BC_i에 대한 응집도 메트릭 CHC는 다음과 같다.

$$CHC(BC_i) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l} W(BM_i(C_j))}{k \times l \times BW}$$

여기서, k는 컴포넌트 인터페이스 메소드 수이며, l은 컴포넌트 내 클래스의 전체 수이다. $W(BM_i(C_j))$는 i 번째 인터페이스 메소드가 j 번째 클래스를 사용하는 유형에 대한 가중치이며, BW는 이러한 가중치 중 최대 가중치이다. 가중치는 인터페이스 메소드의 클래스 사용에 대한 유형에 따라 다른 값을 사용하는데, 생성 및 삭제의 경우 20을, 수정의 경우 3을, 참조의 경우 1의 값을 사용한다. 응집도는 인터페이스 메소드가 클래스를 사용하는 경우의 유형에 따른 가중치의 합을, 인터페이스 메소드가 클래스를 사용하는 가능한 모든 경우의 수에 최대 가중치를 곱한 값으로 나누어 계산한다.

시스템의 두 컴포넌트 BC_i와 BC_j사이의 결합도 CBC는 다음과 같다.

$$CBC(BC_i, BC_j) = \frac{\sum_{k=1}^{E} W_k(BC_i, BC_j)}{O \times Q \times BW}$$

여기서, O와 Q는 두 컴포넌트의 인터페이스 메소드의 수이며, E는 두 컴포넌트 인터페이스 메소드 수의 합이다. $W_k(BC_i, BC_j)$는 두 컴포넌트의 인터페이스 메소드가 다른 컴포넌트의 클래스 사용하는 경우에 대한 가중치이다. 결합도는 두 컴포넌트의 인터페이스 메소드들의 가능한 호출에 대한 가중치의 합을 두 컴포넌트 간에 가능한 인터페이스 메소드의 모든 호출 수에 최대 가중치를 곱한 값으로 나누어 계산한다.

따라서 컴포넌트의 내부 품질 속성에 대한 측정을 통해 컴포넌트의 외부 품질을 향상시키기 위한 컴포넌트 메트릭에 대한 연구가 필요하다. 이러한 컴포넌트 메트릭은 컴포넌트 기반 시스템의 내부 품질 속성에 의해 컴포넌트의 기능적 독립성을 정량화함으로써, 유지보수와 재사용이 쉬운 컴포넌트를 만들기 위함이다.

2.3 클러스터링

클러스터링(clustering)은 여러 개체들을 대상으로 개체가 가지고 있는 수치적 자료를 토대로 개체의 특성을 측정한 후에, 그 특성에 기초하여 개체들을 몇 개의 그룹으로 분류하는 기법을 말한다[8]. 다시 말하면, 각 개체의 특성을 측정할 수 있는 변수를 정하여 거리(distance) 또는 유사성(similarity)으로 개체들 사이의 유사성을 파악한 후, 측정된 거리나 유사성을 통해 관련성이 많은 개체들을 동일한 그룹으로 집단화하는 다변량 기법이다[9].

개체들 사이의 유사성을 측정하는 방법은 다양하게 존재하는데, 측정 방법 및 의미에 따라 거리와 유사성으로 구분할 수 있다. 개체들 사이의 거리를 측정하는 방법으로는 유클리드(euclidean) 거리, 유클리드 제곱(squared euclidean) 거리, 체비셰프(chebychev) 거리, 맨하튼(manhattan) 거리, 민코우스키(minkowski) 거리 등이 있다. 유사성 측정 방법으로는 상관 계수(correlation coefficient)와 코사인(cosine) 값이 있다[8].

유사성은 값이 클수록 두 개체가 서로 유사하다는 것을 의미하나, 거리는 그 값이 작을수록 거리가 가까워 유사함을 의미한다. 유사성이 클수록 두 개체가 유사하다는 의미가 거리에 비해 직관적이므로, 본 논문에서는 유사성으로 개체간의 관련성을 측정하는 방법을 사용한다. 또한, 오퍼레이션들 사이의 유사성을 측정하는 방법으로는 계산 과정이 상대적으로 간단하며, 0에서 1 사이의 값으로 일정한 범위를 갖도록 정규화가 되는 코사인 값을 사용한다. 그리고 서로 유사도가 높은 개체들을 찾아 결합시키는 클러스터링 방법으로는 군 평균법(group average method)을 사용한다. 군 평균법이란 각 군집에

있는 모든 개체들의 유사성 평균값을 구해, 유사성 평균값이 가장 근사한 군집들을 하나의 군집으로 합치는 방법이다.

3. 컴포넌트 독립성 측정을 위한 컴포넌트 메트릭

3.1 개요 및 추상화 모델 정의

시스템 설계의 품질을 측정하기 위한 대표적인 척도인 응집도와 결합도를 이용해 단위 모듈 내부나 단위간의 의존도를 측정할 수 있다[10,11]. 응집도는 모듈 내부 구성 요소들이 서로 얼마나 관련되어 있는지의 정도를 나타내는 소프트웨어 속성이며, 결합도는 모듈 간의 상호 의존도에 대한 척도이다. 바람직한 시스템 설계가 되기 위해서는 가능한 모듈을 독립적으로 생성함으로써 높은 응집도와 낮은 결합도를 가져야 하며, 이의 이점은 모듈간의 상호작용이 적어 한 모듈에서 발생한 오류가 다른 모듈에 영향을 미치는 파급 효과를 줄일 수 있어 유지보수 작업을 국부적으로 수행할 수 있게 된다는 것이다[2,5]. 컴포넌트의 설계가 바람직하다면, 컴포넌트 구성 요소인 클래스 사이에는 밀접한 관련성으로 높은 응집도를 가지며, 다른 컴포넌트의 클래스와의 상호 작용은 적어 낮은 결합도를 가지는 컴포넌트가 될 것이다.

다음은 컴포넌트 메트릭을 정의하기 위해 필요한 컴포넌트에 대한 용어와 추상화 모델을 정의한다.

정의 1. 컴포넌트 기반 시스템을 S라 하고, 분석된 컴포넌트들을 Co_i라 하면, 시스템은 다음과 같이 정의된다.

$$S = \{Co_1, Co_2, \ldots, Co_c\}$$

이때, 컴포넌트 기반 시스템의 전체 컴포넌트 수는 c개이다. □

정의 2. 시스템 내 임의의 컴포넌트 Co_i의 구성요소는 E_{Co_i}이다. E_{Co_i}는 클래스 C로 구성되는 클래스 집합 $CSet_i$와 오퍼레이션 op로 구성되는 인터페이스 집합 $ISet_i$로 구성된다.

$$E_{Co_i} = CSet_i \cup ISet_i$$

$$where,\ CSet_i = \{C_1, C_2, \ldots, C_r\}$$
$$ISet_i = \{op_1, op_2, \ldots, op_s\}$$

이때, 클래스 집합 $CSet_i$의 전체 클래스 수는 r개 이고, 인터페이스 집합 $ISet_i$의 전체 오퍼레이션 수는 s개이다. □

컴포넌트는 특정 서비스를 제공하는 소프트웨어 단위로, 객체지향 시스템의 클래스들의 상호작용을 기반으로 클래스보다 큰 재사용을 위한 독립적인 소프트웨어 단의이다. 컴포넌트가 제공하는 서비스는 인터페이스에 의해 표현된다. 인터페이스는 컴포넌트가 제공하는 서비스가 무엇인지만을 나타낼 뿐, 컴포넌트가 어떤 방법으로 서비스를 제공하는지의 컴포넌트 내부를 보이지는 않는데, 이는 객체지향 시스템의 특성 중 캡슐화(encapsulation) 개념으로 대표되는 추상화에 대응된다. 인터페이스는 컴포넌트의 서비스를 표현하는 개념적인 부분이며, 구체적으로는 인터페이스의 구성 요소인 오퍼레이션에 의한 클래스들 사이의 상호작용을 통해 컴포넌트의 서비스가 제공된다. 다시 말하면, 독립적 기능 단위인 컴포넌트는 서비스 제공을 위해 필요한 데이타와 동작의 분리로서 클래스와 인터페이스를 가지며, 인터페이스는 구체적인 동작의 형태인 여러 오퍼레이션으로 구성된다.

정의 3. 임의의 컴포넌트 Co_i의 서비스를 제공하기 위한 인터페이스의 상호작용은 R_{Co_i}이다. R_{Co_i}는 자기 자신인 컴포넌트 내부 구성 요소인 클래스와의 상호작용 집합 $IntraR_{Co_i}$와 다른 컴포넌트의 구성 요소인 외부 클래스와의 상호작용 집합 $InterR_{Co_i}$로 구성된다. 또한, $InterR_{Co_i}$는 다른 컴포넌트로부터 컴포넌트 Co_i로 향하는 수입(incoming) 관계의 내부적 상호작용 집합 $InputR(Co_i)$와 컴포넌트 Co_i로부터 다른 모든 컴포넌트로 향하는 수출(outgoing) 관계의 외부적 상호작용 집합 $OutputR(Co_i)$로 구성된다.

$$InterR_{Co_i} = InputR(Co_i) \cup OutputR(Co_i)$$

$$R_{Co_i} = IntraR_{Co_i} \cup InterR_{Co_i}$$ □

컴포넌트의 서비스는 오퍼레이션을 통해 컴포넌트 내부의 클래스나 다른 컴포넌트인 외부 클래스들 사이에 메시지 전달을 통해 데이타를 교환하며 상호작용을 함으로써 제공된다[12,13]. 인터페이스는 컴포넌트가 제공하는 서비스가 무엇인지만을 나타낼 뿐 어떤 방법으로 제공하는지는 숨기는 추상화 개념으로 사용자에게 제공되어 사용되나, 컴포넌트 설계자나 개발자 입장에서는 인터페이스의 상호작용에 대해 알아야 한다.

정의 4. 임의의 컴포넌트 Co_i의 인터페이스가 서비스를 제공하기 위한, 인터페이스의 컴포넌트 내부적 상호작용 집합 $IntraR_{Co_i}$의 원소에 대한 수치적 계산 결과를 응집도 $Cohesion(Co_i)$라 한다.

$$Cohesion(Co_i) = cohesion\_calculation(IntraR_{Co_i})$$ □

정의 5. 임의의 컴포넌트 Co_i의 인터페이스가 서비스를 제공하기 위한, 인터페이스의 컴포넌트 외부적 상호작용 집합 $InterR_{Co_i}$의 원소에 대한 수치적 계산 결과를 결합도 $Coupling(Co_i)$라 한다.

$$Coupling(Co_i) = coupling\_calculation(InterR_{Co_i})$$ □

정의 6. 컴포넌트 설계 모델에 대한 컴포넌트의 구성 요소와 그들 사이의 상호작용 관계를 표현하는 컴포넌트 구성 그래프 CSG(Component Structure Graph)를 다음과 같이 정의한다.

$CSG=(V, E)$는 컴포넌트 기반 시스템의 컴포넌트 구성 요소들 사이의 상호작용을 나타내는 방향성 그래프(directed graph)이다. V는 정점의 유한 집합으로, $V=V_c \cup V_o$이다. V_c는 컴포넌트 설계에 나타난 클래스를 표현하는 정점들의 집합으로, 사각형으로 표현하며, V_o는 컴포넌트 설계에 나타난 인터페이스를 구성하는 오퍼레이션의 집합으로, 원으로 표현한다. E는 간선의 유한 집합으로 오퍼레이션이 사용하는 클래스들에 대한 방향 있는 실선으로 표현하며, $E=\{(x,y) \mid x \in V_o, y \in V_c, x \text{ uses } y\}$이다. 그리고 시스템의 개별 컴포넌트는 정점과 간선을 둘러싸는 외곽의 점선 사각형으로 표현한다. □

그림 2는 컴포넌트 메트릭을 사용하여 컴포넌트 설계의 품질을 측정하기 위한, 사례 연구의 비디오 대여 시스템에 대한 CSG이다.

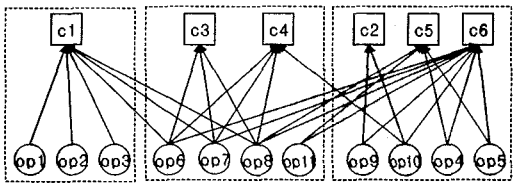

그림 2 컴포넌트 구성 그래프

정의 7. 컴포넌트 인터페이스의 오퍼레이션과 클래스들 사이의 상호작용은 오퍼레이션 사용도 행렬 OUM (Operation Use Matrix)으로 표현된다.

$OUM[i, j] = common\_use(op_i, op_j)$

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

OUM은 컴포넌트 인터페이스의 서로 다른 오퍼레이션이 컴포넌트의 서비스를 제공하기 위해 공통으로 사용하는 클래스가 어느 정도인지를 나타내는 행렬로, 정방 행렬이며 대칭 행렬이다. 행렬의 주대각 요소 부분은 오퍼레이션 자신이 사용하는 클래스 수를 나타내며, 비대각 요소 부분은 서로 다른 오퍼레이션 쌍이 컴포넌트의 서비스를 제공하기 위해 공통으로 사용하는 클래스의 수를 나타낸다. 따라서 OUM의 각 행은 임의의 오퍼레이션이 컴포넌트의 서비스를 제공하기 위해 다른 오퍼레이션들과 공통으로 사용하는 클래스의 수를 나타내는 행벡터로 이해할 수 있다. 행벡터는 오퍼레이션의 동작 유형을 나타내는 것으로, 특정 오퍼레이션이 다른 오퍼레이션들과 공통으로 사용하는 클래스의 많고 적음을 직관적으로 분별할 수 있다. 그림 3은 사례 연구인 비디오 대여 시스템의 OUM이다.

	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	0	1	1	1	0	0	0
2	1	1	1	0	0	1	1	1	0	0	0
3	1	1	1	0	0	1	1	1	0	0	0
4	0	0	0	2	2	1	1	2	1	1	2
5	0	0	0	2	2	1	1	2	1	1	2
6	1	1	1	1	1	4	4	4	2	2	1
7	1	1	1	1	1	4	4	4	2	2	1
8	1	1	1	2	2	4	4	4	2	2	2
9	0	0	0	1	1	2	2	2	3	3	1
10	0	0	0	1	1	2	2	2	3	3	1
11	0	0	0	2	2	1	1	2	1	1	2

그림 3 오퍼레이션 사용도 행렬

정의 8. 컴포넌트 인터페이스의 서로 다른 오퍼레이션의 동작 유형에 대한 유사도는 오퍼레이션 유사도 행렬 OSM(Operation Similarity Matrix)으로 표현된다.

$$OSM[i, j] = similarity(op_i, op_j)$$

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

OSM은 컴포넌트의 서비스를 제공하기 위한 오퍼레이션들의 동작 유형의 유사성을 나타내는 행렬로, 0에서 1 사이의 코사인 값으로 표현된다. OSM의 행과 열의 크기는 오퍼레이션 수이므로 정방 행렬이고, 행렬의 주대각 요소 부분은 동일한 오퍼레이션 사이의 유사도 값이므로 1이고, 대칭 행렬이다. 그림 4는 사례 연구인 비디오 대여 시스템의 OUM으로부터의 OSM이다.

	1	2	3	4	5	6	7	8	9	10	11
1	1.000	1.000	1.000	0.365	0.365	0.778	0.778	0.727	0.426	0.426	0.365
2	1.000	1.000	1.000	0.365	0.365	0.778	0.778	0.727	0.426	0.426	0.365
3	1.000	1.000	1.000	0.365	0.365	0.778	0.778	0.727	0.426	0.426	0.365
4	0.365	0.365	0.365	1.000	1.000	0.738	0.738	0.849	0.778	0.778	1.000
5	0.365	0.365	0.365	1.000	1.000	0.738	0.738	0.849	0.778	0.778	1.000
6	0.778	0.778	0.778	0.738	0.738	1.000	1.000	0.980	0.862	0.862	0.738
7	0.778	0.778	0.778	0.738	0.738	1.000	1.000	0.980	0.862	0.862	0.738
8	0.727	0.727	0.727	0.849	0.849	0.980	0.980	1.000	0.868	0.868	0.849
9	0.426	0.426	0.426	0.778	0.778	0.862	0.862	0.868	1.000	1.000	0.778
10	0.426	0.426	0.426	0.778	0.778	0.862	0.862	0.868	1.000	1.000	0.778
11	0.365	0.365	0.365	1.000	1.000	0.738	0.738	0.849	0.778	0.778	1.000

그림 4 오퍼레이션 유사도 행렬

3.2 컴포넌트 응집도 CHI

인터페이스는 컴포넌트 내부나 외부의 클래스들과의 상호작용을 통해 컴포넌트의 서비스를 제공한다[3,4]. 인터페이스의 오퍼레이션이 컴포넌트 내부의 클래스들을 많이 사용한다면, 컴포넌트 내부 구성 요소들의 친화력인 오퍼레이션과 클래스들 사이의 연관 정도가 높아져 응집도가 높아지게 된다. 그러므로 유지보수가 용이한 품질 높은 시스템을 위해, 시스템에 대한 설계의 품질을 측정하기 위한 대표적인 척도인 응집도를 이용해 컴포넌트의 기능 독립성을 파악한다. 따라서 인터페이스가 컴포넌트의 서비스를 제공하기 위한 오퍼레이션의 동작 유형의 유사도에 의해 컴포넌트 응집도를 측정한다.

정의 9. 인터페이스에 의한 컴포넌트 응집도 CHI (CoHesion by component Interface)

$$CHI(Co_i) = \frac{\sum_{p=N1+1}^{N2} \sum_{q=N1+1}^{p} similarity(op_p, op_q)}{\frac{n_i(n_i+1)}{2}}$$

여기서, 임의의 컴포넌트 Co_i의 인터페이스의 오퍼레이션 전체 수는 n_i이다. p와 q는 현재 컴포넌트 내 오퍼레이션의 순서를 나타내는 일련번호로 $1 \le p \le n_i$이며, $1 \le q \le p$이다. i는 시스템 내 컴포넌트 Co_i의 순서를 나타내는 수로, $N1 = \sum_{r=1}^{i} n_{r-1}$로 순서상 현재 컴포넌트 이전까지의 오퍼레이션 수의 합이고, $N2 = \sum_{r=1}^{i} n_r$로 순서상 현재 컴포넌트까지의 오퍼레이션 수의 합이다. 그리고 $similarity(op_p, op_q)$는 p번째 오퍼레이션과 q번째 오퍼레이션 사이의 동작 유형의 유사성을 나타내는 오퍼레이션 유사도 값이다. □

임의의 컴포넌트에 대한 응집도는 컴포넌트 내부에 속한 오퍼레이션 쌍들의 오퍼레이션 유사도 값의 합을 가능한 오퍼레이션 쌍의 수로 나누어 계산한다. 그리고 OSM은 대칭 행렬이므로 계산의 중복을 피하기 위해, 오퍼레이션을 나타내는 일련번호인 p와 q에 대해 $p \ge q$인 경우만 고려한다.

3.2.1 응집도 속성에 대한 이론적 평가

소프트웨어 메트릭에 대한 검증(verification)은 메트릭이 주장된 속성의 적당한 숫자적 특징임을 보증하는 과정이다. 본 논문에서는 Briand[14]가 정의한 수학적 프레임워크를 이용해 컴포넌트 응집도와 결합도 메트릭의 이론적 타당성을 검증한다. 이 프레임워크의 속성들은 충분 조건적 속성은 아니지만 필요 조건적 속성으로, 메트릭이 엄격하고 덜 실험적이 되도록 한다.

응집도 속성에 대한 이론적 검증은 다음과 같다.

성질 1-1. Non-negativity and Normalization

응집도 측정값은 음수가 아니며 항상 일정한 구간 내에 존재해야 한다는 것을 의미한다. 컴포넌트 응집도는 컴포넌트 내 오퍼레이션 쌍에 대한 0에서 1 사이의 코사인 값인 오퍼레이션 유사도 값의 합을 가능한 오퍼레이션 쌍의 수로 나누기 때문에 0에서 1 사이의 값으로 정규화가 되며 음수의 값을 가질 수 없으므로, 이 성질은 만족된다.

성질 1-2. Null Value

컴포넌트 내부 구성 요소들 사이에 상호작용이 없다면, 응집도 측정값은 0이라는 것을 의미한다. 컴포넌트의 구성 요소들 사이에 상호적 관련성이 전혀 없는 최악의 경우라면 응집도는 0이므로, 이 성질은 만족된다.

성질 1-3. Monotonicity

컴포넌트 내부 구성 요소의 양적 변화 없이 구성 요소들 사이의 상호작용만 증가한다면, 응집도는 감소하지 않는다는 것을 의미한다. 컴포넌트의 설계에서 인터페이스와 내부 클래스 사이의 상호작용이 추가된다면 하나의 컴포넌트로 통합되기 위한 내부적 관계의 증가를 의미하므로 응집도는 감소하지 않는 것을 나타내므로, 이 성질은 만족된다.

성질 1-4. Cohesive Components

서로 관련성이 전혀 없는 두 컴포넌트를 합친다면 응집도는 원래 두 컴포넌트의 최대 응집도보다 크지 않다는 것을 의미한다. 서로 상호작용을 하지 않는 컴포넌트들을 하나의 컴포넌트로 결합할 경우, 관련 없는 요소들이 함께 결합되어 컴포넌트의 구성 요소들 사이에 분리 집합이 생기게 되고, 이는 하나의 컴포넌트로 결합되기 위한 응집적 관계의 감소를 의미한다. 하나로 통합된 컴포넌트의 크기는 커지나 내부적 구성 요소간의 상호작용은 증가하지 않으므로, 응집도는 이전의 최대 응집도보다 증가하지 않아, 이 성질은 만족된다.

3.3 컴포넌트 결합도 CPI

컴포넌트 기반 시스템은 독립된 기능 단위인 컴포넌트가 조합되어 시스템을 구성하게 되고, 조합되는 컴포넌트들은 시스템의 서비스를 제공하기 위해 인터페이스를 통해 컴포넌트 간 상호작용을 하게 된다[3,4]. 그러므로 유지보수가 용이한 품질 높은 시스템을 위해, 시스템에 대한 설계의 품질을 측정하기 위한 대표적인 척도인 결합도를 이용해 컴포넌트의 기능 독립성을 파악한다. 따라서 인터페이스가 컴포넌트의 서비스를 수행하기 위한 컴포넌트 내부의 오퍼레이션과 다른 컴포넌트의 오퍼레이션들과의 상호작용에 대한 오퍼레이션 유사도에 의해 컴포넌트 결합도를 측정한다.

정의 10. 인터페이스에 의한 컴포넌트 결합도 CPI (CouPling by component Interface)

$$CPI(Co_i) = \left(\sum_{p=M+1}^{M} \sum_{q=1}^{t} similarity(op_p, op_q) - \sum_{p=M+1}^{M} \sum_{q=M+1}^{M} similarity(op_p, op_q) \right) \times \frac{1}{n_i(t-n_i)}$$

여기서, 시스템 내 임의의 컴포넌트 Co_i의 오퍼레이션 수는 $|op(Co_i)| = n_i$ 이다. 그리고 시스템 내 모든 컴포넌트 인터페이스의 오퍼레이션 전체 수는 t로, 각 컴포넌트 인터페이스의 오퍼레이션 수의 합으로 $t = n_1 + n_2 + \ldots + n_c$이다. □

임의의 컴포넌트에 대한 결합도는 현재 컴포넌트의 오퍼레이션과 다른 컴포넌트에 속한 모든 오퍼레이션과의 쌍에 대한 오퍼레이션 유사도 값의 합을 가능한 오퍼레이션 쌍의 수로 나누는 방법으로 계산된다. 오퍼레이션 유사도 값은 0에서 1 사이의 코사인 값이므로, 0에서 1 사이의 범위로 결합도는 정규화된다. 현재 컴포넌트의 오퍼레이션과 다른 컴포넌트의 오퍼레이션과의 오퍼레이션 쌍에 대한 오퍼레이션 유사도 값의 합은 두 오퍼레이션 사이의 직접적인 상호작용과 간접적인 상호작용을 모두 포함하므로, 결합도 측정에 포함한다. 한 컴포넌트의 오퍼레이션과 다른 컴포넌트의 오퍼레이션들 사이의 동작 유형이 유사할수록 컴포넌트 간 결합도는 높다 할 수 있으며, 동작 유형이 상이할수록 컴포넌트가 서로 관련 없이 독립적으로 동작을 하게 되는 것을 의미하므로, 컴포넌트 결합도는 낮다고 할 수 있다.

3.3.1 결합도 속성에 대한 이론적 평가

결합도 속성에 대한 이론적 검증은 다음과 같다.

성질 2-1. Non-negativity

결합도 측정값은 음수가 아니라는 것을 의미한다. 결합도는 컴포넌트 간의 관련성을 나타내는 값으로, 하나의 컴포넌트는 다른 컴포넌트와 상호작용을 하거나 하지 않을 수 있으므로, 결합도는 음수의 값을 가질 수는 없다. 따라서 이 성질은 만족된다.

성질 2-2. Null Value

컴포넌트들 사이에 상호작용이 없다면 결합도 측정값은 0 이라는 것을 의미한다. 컴포넌트가 다른 컴포넌트와 상호작용을 전혀 하지 않으면서 독립적으로 서비스를 제공할 수도 있으므로, 이 성질은 만족된다.

성질 2-3. Monotonicity

컴포넌트 내부 구성 요소에는 양적 변화 없이 컴포넌트들 사이의 상호작용만 증가한다면, 결합도는 감소하지 않는다는 것을 의미한다. 결합도는 컴포넌트 간의 상호작용 정도를 나타내는 값으로, 컴포넌트와 다른 컴포넌트 사이의 외부적 상호작용이 증가한다면 결합도는 감소하지 않으므로, 이 성질은 만족된다.

성질 2-4. Merging of Components

컴포넌트들을 결합하여 하나의 컴포넌트로 만든다면, 새로이 결합된 컴포넌트의 결합도는 결합 이전 각 컴포넌트의 결합도의 합보다 증가하지는 않는다는 것을 의미한다. 컴포넌트가 결합되는 경우, 컴포넌트 간의 외부적 상호작용의 일부는 새로이 결합되어 만들어진 컴포넌트의 내부적 상호작용으로 포함될 수 있으므로, 컴포넌트들 사이의 상호작용의 정도를 나타내는 결합도는 결합 이전의 컴포넌트들의 결합도의 합보다 증가하지는 않아, 이 성질은 만족된다.

성질 2-5. Disjoint Component Additivity

서로 상호작용을 하지 않는 컴포넌트들을 결합하여 하나의 컴포넌트로 만든다면, 결합된 컴포넌트의 결합도는 결합 이전의 각 컴포넌트의 결합도 합과 동일하다는 것을 의미한다. 컴포넌트들 사이에 상호작용이 전혀 없기 때문에 결합 이전과 이후의 컴포넌트들 사이의 상호작용은 그대로 유지되고 변화가 없으므로, 기존 컴포넌트들의 결합도의 합과 동일하다. 따라서 이 성질은 만족된다.

3.4 컴포넌트 독립도 IDI

독립된 기능을 제공하는 소프트웨어 단위인 컴포넌트의 설계에 대해 응집도는 높고 결합도는 낮아야 한다는 일반적이고 추상적인 기준만 있을 뿐, 응집도나 결합도 측정 결과가 바람직한지 또는 그렇지 않은지를 판단하기 위한 구체적인 기준 값이 제시되지는 않고 있다. 따라서 개별 컴포넌트의 설계에 대해 내부적 구성 요소와는 응집도를 외부적 구성 요소와는 결합도를 측정해, 컴포넌트의 기능 독립성을 종합적으로 판단하기 위한 인터페이스에 의한 컴포넌트 독립도를 측정한다.

정의 11. 인터페이스에 의한 컴포넌트 독립도 IDI (InDependence by component Interface)

$$IDI(Co_i) = CHI(Co_i) - CPI(Co_i)$$ □

임의의 컴포넌트에 대한 독립도는 컴포넌트 응집도와 컴포넌트 결합도의 차이로 계산한다. 일반적으로 바람직한 컴포넌트 설계는 응집도는 가능한 높아야 하며 결합도는 가능한 낮아야 하므로, 응집도에서 결합도를 뺀 값으로 독립도를 구한다. 본 논문에서 제안한 컴포넌트 응집도와 결합도는 측정값이 모두 0에서 1 사이의 값으로 정규화가 되었으므로, 컴포넌트 독립도는 -1에서 1 사이의 값으로 정규화가 된다. 응집도가 가장 높고 결합도가 가장 낮은 바람직한 컴포넌트 설계일 경우, 독립도는 1에 가까운 값을 가질 것이며, 응집도가 가장 낮고 결합도가 가장 높은 바람직하지 못한 컴포넌트 설계의 경우, 독립도는 -1에 가까운 값을 가질 것이다. 독립도의 값이 -1에서 1 사이의 값으로 정규화가 됨으로, 컴포넌트 독립도가 양수의 값이면 컴포넌트는 독립적인 기능 단위가 될 것이라고 판단 가능하다. 그리고 독립도가 음수의 값이면 컴포넌트는 서비스를 수행하기 위해 독립적이기보다는 다른 컴포넌트에 의존적일 것이라고 판단

가능하다. 따라서 컴포넌트 독립도를 통해 컴포넌트 응집도와 결합도를 종합적으로 평가하여 독립적인 기능 단위로의 컴포넌트 설계에 대한 평가를 쉽도록 한다.

4. 컴포넌트 메트릭을 이용한 컴포넌트 재설계

인터페이스는 컴포넌트의 서비스를 제공하기 위해 오퍼레이션이 컴포넌트 내부나 외부의 클래스들과 상호작용을 하는데, 이러한 오퍼레이션의 동작 유형의 유사도에 의해 인터페이스에 의한 컴포넌트 응집도, 결합도, 독립도를 측정한다. 그리고 유사도가 높은 오퍼레이션들은 같은 컴포넌트 인터페이스에 묶이는 것이 좋다는 클러스터링 원리에 의해 컴포넌트 모델을 재설계하여 설계 모델의 적절성, 기능 독립성 등의 시스템 설계의 품질을 향상시키고자 한다. 그림 5는 컴포넌트 설계 모델로부터 컴포넌트 응집도, 결합도, 독립도를 측정하기 위해 필요한 정보를 추출하고 측정하며, 이를 통해 컴포넌트 모델을 재설계하는 전체 과정을 나타낸다.

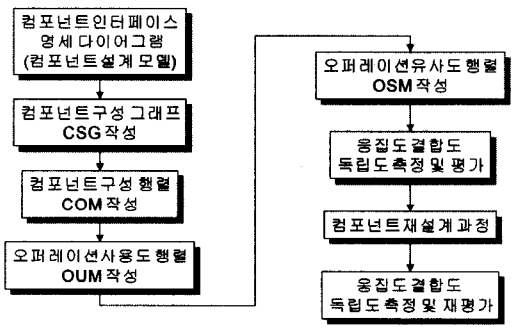

그림 5 컴포넌트 측정 및 재설계 전체 과정

다음은 클러스터링 원리와 컴포넌트 메트릭을 이용해 컴포넌트 설계 모델을 재정비하기 위한 컴포넌트 재설계 과정에 대해 단계별로 설명한다.

단계 1. 분할 컴포넌트를 선택하고, 이를 분할한다.

분할 컴포넌트는 컴포넌트 내부 구성요소들 사이의 상호작용이 독립적이고 관련성이 서로 없어, 응집도가 가장 낮은 컴포넌트이다. 컴포넌트 인터페이스의 오퍼레이션과 내부의 클래스가 공통으로 상호작용을 하는 부분이 없이 서로 독립적으로 상호작용을 하는 경우, 컴포넌트는 서로 독립적인 부분으로 분할될 수 있다. 그러므로 분할 컴포넌트를 먼저 선택하고 분할하여 응집도를 향상시킬 수 있으며, 이는 결과적으로 독립도를 향상시키게 된다.

컴포넌트가 분할 컴포넌트 인지의 판단은 OUM을 통해 가능하다. OUM 행렬로부터 특정 오퍼레이션을 나타내는 행벡터를 선택한 후, 선택된 행벡터에 대해 0이 아닌 값을 포함하는 열벡터의 원소가 모두 0이라면, 해당 컴포넌트를 분할 컴포넌트로 판단한다.

단계 2. 독립도가 가장 낮은 컴포넌트를 선택하여, 이를 분할한다.

시스템 내에 분할 컴포넌트가 더 이상 없다면 독립도가 가장 낮은 컴포넌트를 선택한 후, 이 컴포넌트 인터페이스의 오퍼레이션들에 대해 평균보다 낮은 오퍼레이션 유사도를 갖는 오퍼레이션들을 분리하여 컴포넌트를 분할한다.

컴포넌트 독립도는 컴포넌트 응집도에서 결합도를 뺀 값이므로, 높은 독립도를 위해서는 높은 응집도를 갖는 것이 충분조건이 된다. 컴포넌트 응집도는 컴포넌트 내 오퍼레이션들의 유사도의 합으로 계산되므로, 컴포넌트 내 오퍼레이션들 중 평균보다 낮은 오퍼레이션 유사도를 갖는 오퍼레이션들을 분리하면 컴포넌트의 응집도는 향상될 것이다.

단계 3. 시스템 내 모든 컴포넌트에 대해 오퍼레이션 유사도 값의 평균이 가장 근사한 컴포넌트들을 선택하여 하나로 결합한다.

컴포넌트는 오퍼레이션이라는 개체가 모인 군집이라고 생각할 수 있으므로, 서로 유사도가 밀접한 오퍼레이션들만 선택하고 묶어 새로운 컴포넌트로 만드는 클러스터링 방법을 이용하면 응집도가 높은 컴포넌트가 될 것이다. 클러스터링 방법으로는 컴포넌트의 구성 요소인 오퍼레이션들의 유사도 평균값에 대해 분산(variance)이 작은 컴포넌트들을 하나의 컴포넌트로 묶는 군 평균법을 이용한다. 분산이 작은 컴포넌트란 유사도가 높은 오퍼레이션들만 모인 것이므로, 컴포넌트 내부 구성 요소의 관련성이 많아져 응집도가 높은 컴포넌트라 생각할 수 있고, 결과적으로 독립도가 높은 컴포넌트이기 때문이다.

단계 4. 결합된 컴포넌트가 가장 낮은 독립도를 가질 때 재설계 과정을 종료한다.

컴포넌트 재설계 과정은 오퍼레이션 유사도 값에 의해 오퍼레이션들을 분리하거나 묶는 과정을 통해 컴포넌트가 분할되고 결합되는 과정을 반복하게 된다. 컴포넌트 응집도를 저해하는 요소로 낮은 오퍼레이션 유사도 값의 오퍼레이션이 분리되고, 컴포넌트 응집도를 향상시키기 위해 오퍼레이션 유사도 평균이 근사한 컴포넌트들이 다시 결합된다. 분리된 오퍼레이션이 다른 컴포넌트와 결합되어 새로운 컴포넌트가 만들어진 경우, 해당 컴포넌트가 가장 낮은 독립도를 갖는다면 이 컴포넌트는 다시 분할되어야 한다. 이러한 경우, 결합과 분할이라는 이전의 재설계 과정을 다시 반복하게 될 것이므로, 최근에 결합된 컴포넌트가 가장 낮은 독립도를 가져 다시 분할되어야 한다면 컴포넌트 재설계 과정을 종료한다.

5. 사례 연구 및 비교 평가

5.1 사례 연구

컴포넌트 설계 모델에 대해 응집도, 결합도, 독립도를 측정해 보고, 클러스터링 원리를 이용한 컴포넌트 재설계 과정을 적용하여 컴포넌트의 기능적 독립성을 향상시키기 위한 과정을 사례 연구를 통해 확인한다. 사례 연구의 비디오 대여 시스템은 고객이 비디오에 대한 정보를 검색하고, 비디오를 대여 및 반납하기 위한 시스템이다.

그림 6은 비디오 대여 시스템의 초기 컴포넌트 설계에 대한 인터페이스 책임 다이어그램으로, 컴포넌트 아키텍처와 인터페이스를 확인할 수 있다. 비디오 테이프에 대한 정보를 관리하는 TapeMgr 컴포넌트와 비디오 대여 및 반납에 대한 정보를 관리하는 RentMgr 컴포넌트 그리고 고객에 대한 정보를 관리하는 CustomerMgr 컴포넌트로 구성되며, 각 컴포넌트가 제공하는 서비스는 ITapeMgt 인터페이스, IRentMgt 인터페이스, ICustomerMgt 인터페이스로 정의된다. 초기 컴포넌트 설계 모델에 대해 컴포넌트 구성 요소인 클래스와 오퍼레이션 사이의 상호작용에 대한 관계를 그림으로 그린 CSG는 그림 2이고, OUM은 그림 3, OSM은 그림 4이다. 그리고 초기 컴포넌트 설계 모델에 컴포넌트 재설계 과정을 적용하여 컴포넌트 설계 모델을 재설계한 결과의 CSG는 그림 7과 같다.

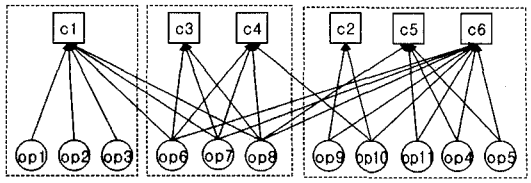

그림 7 재설계 과정 후 CSG

표 1 재설계 과정 전과 후의 측정 결과

	재설계 과정 전	재설계 과정 후
응집도	0.947	0.968
결합도	0.646	0.640
독립도	0.300	0.328

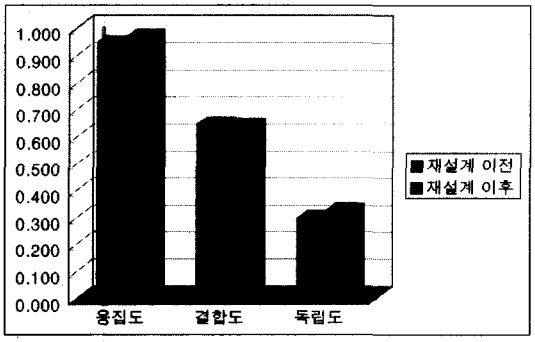

그림 8 재설계 과정 전과 후의 측정 결과 비교

표 1은 컴포넌트 재설계 과정 전과 후의 응집도, 결합도, 독립도 측정값을 나타낸 표이고, 그림 8은 측정 결

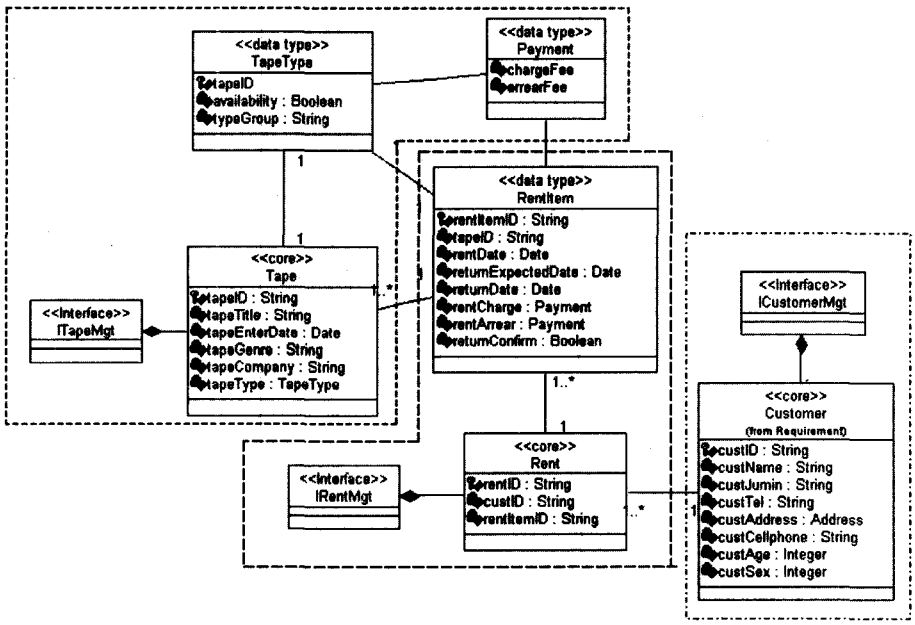

그림 6 인터페이스 책임 다이어그램

과를 비교하기 쉽도록 그래프로 나타낸 것이다. 결과로 알 수 있듯이, 본 논문에서 제안한 클러스터링 원리를 이용한 컴포넌트 재설계 과정이 독립된 기능 단위가 되기 위한 컴포넌트의 설계 품질을 향상시킴을 확인했다. 비디오 대여 시스템의 경우, 재설계 과정을 통해 컴포넌트 아키텍쳐에는 변화 없이, 각 컴포넌트를 구성하는 오퍼레이션의 위치 변화만으로 컴포넌트의 기능 독립성이 향상되었음을 확인했다.

5.2 비교 평가

본 논문에서 제안한 컴포넌트 메트릭의 유용성을 증명하기 위해 컴포넌트 설계 모델에 기존의 컴포넌트 메트릭을 적용하여 응집도와 결합도를 계산해 본다. 본 논문의 컴포넌트 메트릭의 측정 결과와 기존 컴포넌트 메트릭의 측정 결과를 비교해봄으로써, 컴포넌트 인터페이스의 특성을 반영한 본 논문의 컴포넌트 메트릭의 필요성 및 효용성을 확인한다.

비교 대상인 컴포넌트 메트릭으로는 본 논문의 컴포넌트 메트릭과 가장 유사한 Choi[7]의 응집도와 결합도 메트릭을 사용한다.

컴포넌트 설계 모델에 기존 컴포넌트 메트릭을 적용한 응집도와 결합도 측정 결과는 표 2에 있다. 그림 9는 기존 컴포넌트 메트릭의 측정 결과와 본 논문에서 제안한 컴포넌트 메트릭의 측정 결과를 비교하여 그래프로 나타낸 것이다.

측정 결과 컴포넌트 재설계 과정 전과 후에 대해 시스템의 응집도는 향상되었음을 알 수 있다. 기존 컴포넌트 응집도의 경우 향상된 정도가 측정 결과 값에서 두드러지게 드러나나, 이는 메트릭 측정 과정에 가중치를 사용함으로써 얻어진 결과이다. 가중치를 사용할 경우, 가중치 사용에 대한 번거로움과 가중치로 어떤 값을 사용할 지에 대한 명확한 근거가 제시되어야 하나, 그렇지 못하다는 문제점이 있다. 그러므로 본 논문에서 제안한 컴포넌트 인터페이스의 특성을 반영한 컴포넌트 메트릭은 가중치를 사용하지 않고 재설계 과정 전과 후의 응집도 향상을 측정할 수 있음을 확인했다. 이로써, 본 논문의 컴포넌트 메트릭이 사용하기 쉬우며, 컴포넌트 메트릭의 효용성을 밝히게 된다.

표 2 기존 컴포넌트 메트릭 적용 : 재설계 과정 전/후

	재설계 과정 전	재설계 과정 후
응집도	0.196	0.409
결합도	0.065	0.070

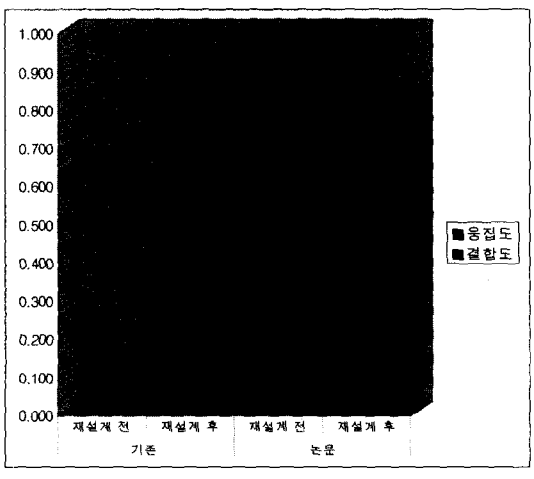

그림 9 기존 컴포넌트 메트릭과의 측정 결과 비교

6. 결 론

본 논문은 컴포넌트 인터페이스의 특성을 반영한 컴포넌트 응집도, 결합도, 독립도 메트릭을 제안했다. 이는 컴포넌트 인터페이스가 컴포넌트의 서비스를 제공하기 위해 얼마나 독립적으로 기능을 수행하는지를 측정한다. 그리고 독립적 기능 단위로 좀더 바람직한 컴포넌트 설계가 되기 위한 클러스터링 기법을 사용한 컴포넌트 재설계 과정을 제안했다.

컴포넌트의 서비스를 제공하기 위한 오퍼레이션의 상호작용은 컴포넌트 구성 그래프로 표현되고, 오퍼레이션이 컴포넌트의 서비스 제공을 위해 다른 오퍼레이션들과 공통으로 사용하는 클래스에 대한 정보는 오퍼레이션 사용도 행렬로, 오퍼레이션들의 동작 유형이 어느 정도 유사한지의 정보는 오퍼레이션 유사도 행렬로 표현된다. 컴포넌트 기반 시스템 개발 초기 단계에서 컴포넌트 설계 모델에 대해 컴포넌트의 응집도, 결합도, 독립도를 측정하여, 독립적 기능 단위로 바람직한 컴포넌트 설계인지를 판단한다. 그리고 독립적 기능 단위로의 컴포넌트의 설계 품질을 향상시키기 위해 오퍼레이션 유사도 값에 클러스터링 원리를 사용한 컴포넌트 재설계 과정을 적용한다. 개발 이전에 독립적인 기능 단위가 되기 위한 컴포넌트의 설계 품질 측정과 향상을 통한 재설계 과정의 유용성은 사례 연구 및 비교 평가를 통해 확인했다. 향후 연구과제는 제안한 컴포넌트 메트릭을 많은 실무 데이타에 적용하여 실험적 입증을 하는 것이다.

참 고 문 헌

[1] The Software Engineering Institute(SEI) in Carnegie Mellon University, "Component-Based Software Development / COTS Integration," http://www.sei.cmu.edu/str/descriptions/cbsd_body.html, January 1997.

[2] 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, pp.372-408, Addison-Wesley, 2002.

[3] Clemens Szyperski, Dominik Gruntz, Stephan Murer, Component Software: Beyond Object-Oriented Programming, 2nd Edition, pp.35-82, Addison-Wesley, 2002.

[4] John Cheesman, John Daniels, UML Components: A Simple Process for Specifying Component-Based Software, pp.25-36, 103-146, Addison-Wesley, 2001.

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

[6] 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.

[7] 최미숙, 분석 클래스 기반의 컴포넌트 식별 메트릭스와 컴포넌트 식별 방법에 관한 연구, 숙명여자대학교 박사학위논문, 2002년 8월.

[8] 차석빈 외 5인, 다변량 분석의 이론과 실제, pp. 253-282, 학현사, 2001.

[9] 성웅현, 경영통계 자료 분석, pp.405-428, 무역경영사, 1997.

[10] Hind Kabaili, Rudolf K. Keller, Francois Lustman, Class Cohesion as Predictor of Changeability: An Empirical Study, Hermes Science Publications, Paris, France, 2001.

[11] Bindu Mehra, A Critique of Cohesion Measures in the object-Oriented Paradigm, Master Thesis, Department of Computer Science, Michigan Technological University, 1997.

[12] George T. Heineman, William T. Councill, Component-Based Software Engineering : Putting the Pieces Together, pp.307-320, Addison-Wesley, 2001.

[13] Desmond F. D'Souza, Alan C. Wills, Object, Component and Framework with UML : The Catalysis Approach, pp.45-184, Addison-Wesley, 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

박사(2003년 8월). 숙명여자대학교 정보과학부 컴퓨터과학전공. Major in Computer Science, Division of Information Science, Sookmyung Women's University. 연구분야는 소프트웨어 공학

박 재 년

정교수. 숙명여자대학교 정보과학부 컴퓨터과학전공. Major in Computer Science, Division of Information Science, Sookmyung Women's University. 연구분야는 소프트웨어 공학