컴포넌트의 응집성 측정

고 병 선[†] · 박 재 년[††]

요 약

이미 존재하는 기능의 조각인 컴포넌트를 조립함으로써 시스템의 개발 시간과 비용을 줄이고, 소프트웨어의 품질과 생산성을 향상시키고자 하는 컴포넌트 기반 개발 방법론이 새로운 재사용 기술로 나타나기 시작했다. 컴포넌트 기반 시스템은 컴포넌트의 조합으로 구성되기 때문에 개별 컴포넌트의 품질에 의해 영향을 받는다. 그러므로, 개발될 컴포넌트 시스템의 품질을 향상시키기 위해서는, 조립될 개별 컴포넌트의 품질에 대한 측정이 필요하다. 따라서, 본 논문에서는 컴포넌트 인터페이스와 내부의 클래스 또는 클래스들 사이의 관련성으로 컴포넌트 응집성을 측정하는 메트릭스를 제안한다. 이는 소프트웨어 개발 주기의 초기인 분석단계에 적용하여, 향후 개발될 컴포넌트의 기능적 응집 정도를 측정해 볼 수 있다. 컴포넌트의 기능 독립성을 예측 가능함으로써, 소프트웨어 개발에 대한 노력을 줄일 수 있으며 컴포넌트 재사용을 통한 시스템의 품질 향상을 가져올 수 있는 효과를 기대할 수 있다.

Measuring cohesion of a component

Byung-Sun Ko[†] · Jai-Nyun Park[††]

ABSTRACT

The component-based development methodology becomes famous as the new technology for reuse. That technology can help us easily develop a complex and large system by composing reusable components in short period with high-quality and low-cost. The component-based system may be developed by composing more than one component. So, the quality of component-based system is determined by individual component quality. Therefore, it is necessary to measure individual component quality for the improvement in quality of component-based system. Hence, in this paper, we propose new component metrics for measuring the cohesion as relationship between classes and interfaces or among classes. Those can be applied in the early stage of software development life cycle. So, we can measure the functional cohesion of component which will be developed. Predicting functional independence of a component, we expect to reduce the software developing cost & effort and improve software quality by reusing a component.

키워드 : 컴포넌트(Component), 응집도(Cohesion), 측정(Measurement)

1. 서 론

소프트웨어 재사용 기술은 소프트웨어의 품질과 개발 생산성을 향상시키기 위한 방안으로 제시되었다. 소프트웨어도 마치 하드웨어 부품처럼 기능의 조각을 조립하여 재사용 한다면, 단기간에 저비용으로 고품질의 소프트웨어를 생산할 수 있다고 생각되었다[1].

객체지향 개발(object-oriented development) 방법론은 1980년 초기부터 소프트웨어 재사용 문제의 해결책으로 사용되기 시작했다. 객체지향 개발 방법론을 통해 생산된 클래스들을 재사용하기 위해서는 즉, 자식 클래스가 부모 클래스의 데이터와 동작을 상속받기 위해서는 부모 클래스의 내부를 파악해야 한다. 그러나, 클래스의 내부를 파악하는 작업은 쉬운 일이 아니므로, 이는 클래스 재사용의 한계라 할 수 있다.

그후 1990년대에 컴포넌트 기반 개발(component-based development) 방법론이 소프트웨어 개발에서의 재사용 문제를 해결하기 위한 새로운 기술로 나타나기 시작했다. 이 방법은 이미 존재하는 기능의 조각인 컴포넌트를 조립함으로써 시스템을 개발하는 방법으로, 개발 시간과 비용을 줄이고 생산성을 향상시킬 수 있다. 컴포넌트는 복잡한 데이터와 동작은 내부에 숨기고 인터페이스만 외부에 분리되어 있어, 사용자나 개발자는 인터페이스를 통해서만 컴포넌트에 접근하거나 시스템을 개발할 수 있다. 이러한 컴포넌트의 캡슐화 특성은 에러의 영향을 제한시켜 유지보수를 용이하게 한다.

소프트웨어 개발에 있어서 부품의 역할을 하는 독립적인 재사용 단위인 컴포넌트를 조립하여 시스템을 개발하는 컴포넌트 기반 시스템의 품질은 각 개별 컴포넌트의 품질에

† 준 회 원 : 숙명여자대학교 대학원 컴퓨터과학과
†† 정 회 원 : 숙명여자대학교 정보과학부 교수
논문접수 : 2001년 11월 17일, 심사완료 : 2002년 4월 24일

의해 결정된다. 즉, 새로 개발될 시스템의 품질은 재사용되는 컴포넌트의 품질에 영향을 받는다. 따라서, 개발될 시스템의 품질 향상과 컴포넌트의 보다 폭넓은 활용을 위하여, 각 컴포넌트의 품질 측정은 필수적이라 할 수 있으며, 이미 평가받은 컴포넌트를 사용하여 시스템을 구축함으로써 시스템 개발의 위험성을 줄일 수 있다는 장점을 갖는다[2-4].

따라서, 본 논문에서는 컴포넌트의 기능적 응집성을 측정하기 위한 새로운 컴포넌트 메트릭스를 제안한다. 객체지향 시스템에 적용되던 기존의 메트릭스는 클래스, 인스턴스 변수, 메소드, 상속성 등을 고려하여 클래스를 기반으로 측정하였다. 그러나, 컴포넌트는 하나 이상의 클래스와 인터페이스로 구성되므로, 객체지향 시스템에 적용되던 기존의 메트릭스를 그대로 컴포넌트 기반 시스템에 적용하는 것은 부적당하다. 그러므로, 기존의 클래스 수준의 객체지향 메트릭스와는 측정 단위가 다른 컴포넌트의 추가적인 정보를 고려한 컴포넌트 메트릭스를 제안하며, 이로써 컴포넌트의 기능적 응집성을 측정해 보고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 연구 배경으로 객체지향 메트릭스에 대해 살펴보고, 그 한계점에 대해 설명한다. 3장에서는 기본적 개념으로 컴포넌트의 정의와 컴포넌트의 기능적 응집성을 측정하기 위해 제안한 메트릭스에 대해 설명한다. 4장에서는 제안된 메트릭스의 이론적 적합성 평가와 사례 연구를 통해 결과를 분석해 보고, 마지막으로 5장에서 결론을 맺는다.

2. 연구 배경

2.1 여러 가지 객체지향 메트릭스

클래스는 객체지향 시스템의 기본 단위로, 메시지 전달이나 상속의 특성을 갖는다. 클래스 설계의 평가는 클래스의 메소드, 상속성, 결합도, 메시지 전달, 테스팅 등의 측면에서 이루어진다[5]. 본 절에서는 몇 가지의 객체지향 메트릭스에 대해 살펴보겠다.

Chidamber와 Kemerer[6]가 제안한 메트릭스는 대표적 클래스 중심 메트릭스이며, 다음과 같다. 클래스 당 가중 메소드의 수 WMC(Weighted Methods per Class)는 클래스에 정의된 각 메소드들의 복잡도의 합으로, 이는 클래스를 개발하거나 유지보수 하는데 드는 노력을 나타내는 척도로 사용될 수 있다. 이 값이 클수록 클래스에 선언된 메소드가 많아 하위 클래스에 상속되어 영향을 주는 부분이 많아짐을 의미한다. 객체간의 결합도 CBO(Coupling Between Object classes)는 상속성이 없는 클래스들 사이에서, 하나의 클래스가 다른 클래스의 메소드나 속성을 사용하여 결합한 수를 측정하는 것이다. 이 값이 크면 설계의 모듈화와 재사용을 어렵게 만들며, 변경에 대한 파급 효과가 커 유지보수를 어렵게 한다. 클래스에 대한 반응도 RFC(Response For a Class)는 클래스가 호출하는 모든 메소드의 수이다. 이는 클래스의 반응 정도를 나타내며, 호출되는 메소드가 많아질수록 객체의 복잡도는 증가하며, 유지보수에 많은 노력이 필요하다. 메소드의 응집 결여도 LCOM(Lack of COhesion in Methods)은 모든 메소드간의 조합 쌍 중에서 같은 메소드에 의해 공유되지 않는 인스턴스 수와 공유되는 인스턴스 수의 차이다. 메소드들이 같은 속성을 사용한다면 클래스의 유사도가 크고 매우 응집력이 높은 것으로, 이로써 해당 모듈의 캡슐화 정도를 알수 있다.

Wei & Henry[7]가 제안한 메트릭스는 다음과 같다. 응집 결여도 LCOM은 클래스 내의 메소드들이 공통으로 가지는 인스턴스가 없는 즉, 지역 메소드의 분리 집합의 개수로 정의하였다. 결합도는 세 가지 측면으로 제시하였다. 상속성을 통한 결합도는 상속 계층도의 높이와 자식 노드의 수로 측정하며, 높을수록 좋다. 객체지향에서는 재사용을 위해 상속성을 강조하기 때문에 이는 높을수록 좋으나, 지나친 결합도는 캡슐화와 정보 은닉을 해칠수 있으므로 피해야 한다. 메시지 전달을 통한 결합도는 메시지 전달이 많은 클래스간의 결합도가 높다는 것을 의미하고, 추상 데이터형을 통한 결합도는 클래스 내에서 정의된 추상 데이터형의 수로 정의하였는데, 이 값이 크다는 것은 다른 클래스의 정의에 연관된 자료 구조가 많다는 것을 의미한다.

Bieman와 Kang[8]은 LCOM[6]과 유사한 TCC(Tight Class Cohesion)와 LCC(Loose Class Cohesion)을 제안하였다. 두 메소드가 하나 이상의 공통 인스턴스 변수를 사용할 때는 직접적으로 연결되었다고 하고, 다른 간접적인 연결 메소드에 의해 연결되었을 때는 간접적으로 연결되었다고 한다. TCC는 클래스에서 직접적으로 연결된 메소드 수의 비율로 구하고, LCC는 클래스의 최대 연결수에 대해 직접적과 간접적으로 연결된 메소드 수의 비율로 구하며, 이 두 값은 클수록 응집도가 높다.

Binder[9]는 공용 속성과 보호된 속성의 비율 PAP(Public And Protected)와 데이터 멤버에 대한 공용 접근도 PAD(Public Access to Data members)를 제안하였다. PAD는 캡슐화에 위배되는 다른 클래스의 속성에 접근할 수 있는 클래스나 메소드의 수를 나타낸다.

2.2 객체지향 메트릭스의 한계

객체지향 시스템은 속성과 메소드가 캡슐화된 클래스를 기반으로 개발된다. 그러므로, 대부분의 객체지향 메트릭스는 객체지향 시스템의 기본 단위인 클래스에 기반을 두고, 클래스, 메소드, 상속성, 캡슐화 등을 고려하여 품질을 측정하였다. 그러나, 컴포넌트 기반 시스템은 하나 이상의 컴포넌트로 이루어지며, 또한 컴포넌트는 하나 이상의 클래스와 인터페이스로 구성되므로, 객체지향 메트릭스를 그대로 컴포넌트에 적용하는 것은 부적당하다. 그러므로, 상호작용이

이루어지는 컴포넌트 내 클래스들과 인터페이스와 같은 추가적인 정보를 고려한 새로운 컴포넌트 메트릭스가 필요하다. 따라서, 본 논문에서는 컴포넌트 메트릭스를 제안하여, 컴포넌트의 구성에 대한 기능적 응집성을 측정하고자 한다.

3. 컴포넌트의 응집성 측정

3.1 컴포넌트의 정의

객체지향 시스템의 클래스들은 그들의 상호 작용을 기반으로 하나의 기능을 제공하기 위한 독립적인 소프트웨어 단위인 컴포넌트로 묶일 수 있으며, 클래스보다 큰 재사용 단위이다. 인터페이스는 컴포넌트 내부의 클래스들에 의해 수행되는 서비스에 접근하기 위한 오퍼레이션들의 집합으로, 재사용되는 컴포넌트의 기능은 인터페이스를 통해서만 제공된다[4, 10, 11].

컴포넌트에 대한 정의는 여러 가지가 있으나, 본 논문에서는 컴포넌트를 특정 기능을 제공하기 위한 클래스들의 묶음으로, 외부에서 컴포넌트 내부의 클래스들의 서비스에 접근하기 위해서는 인터페이스를 이용한다고 하겠다.

3.2 응집성 측정

객체지향 시스템에서 클래스 응집성을 측정한다는 것은 클래스 내의 속성과 메소드가 얼마나 연관 되었는지의 정도를 측정하는 것이다[12-14]. 본 연구에서는 클래스의 속성과 이에 상호 작용하는 메소드의 연관 정도인 클래스의 응집성을 컴포넌트로 확장하여 컴포넌트의 응집성이라 하겠다. 응집도란 모듈 구성 요소들이 얼마나 연관되어 있는지를 측정하는 소프트웨어의 속성이다. 높은 응집도를 갖는 독립적인 모듈은 하나의 기능을 수행하는 분리될 수 없는 모듈로, 소프트웨어 복잡도를 감소시켜 소프트웨어의 품질과 신뢰성을 향상시키며 유지보수를 쉽게 할 수 있는 핵심으로 간주될 수 있다.

본 장에서는 컴포넌트의 응집성을 측정하기 위해 컴포넌트 메트릭스를 제안한다. 컴포넌트의 응집성 측정은 임의의 한 컴포넌트에 대해, 인터페이스와 컴포넌트 내부의 클래스 사이의 연관 정도를 측정하는 인터페이스에 의한 응집도 *RCP*(the Ratio of Class-type Parameters to interface's parameters)와, 내부의 클래스가 속성으로 다른 클래스를 사용하는 정도에 의한 클래스에 의한 응집도 *RCA*(the Ratio of Class-type Attributes to class's attributes)의 두 가지로 측정한다. 다시 말하면, 시스템의 임의의 한 컴포넌트 *Co*에 대해, 컴포넌트의 응집성 측정 $M(Co)$는 *RCP*와 *RCA*의 두 가지 요소에 의해 이루어진다.

$$M(Co) = \{RCP,\ RCA\}$$

인터페이스는 관련된 여러 오퍼레이션들의 그룹으로, 컴포넌트 내부의 클래스들에 공통적으로 접근하기 위한 방법을 제공한다. 컴포넌트는 특정 기능을 제공하는 독립된 소프트웨어 단위이기에, 인터페이스가 내부의 클래스들을 공통으로 사용하는 정도가 많을수록, 그 컴포넌트의 기능적 응집성은 높다할 수 있다. 그러므로, 내부의 클래스를 인터페이스가 얼마나 공통적으로 사용하는지에 의해 컴포넌트의 응집성을 측정할 수 있으며, 이는 인터페이스의 매개변수 중 클래스 타입인 매개 변수의 비율로 구한다.

[정의 1]은 임의의 한 컴포넌트에 대해, 인터페이스의 오퍼레이션이 매개변수로 내부의 클래스를 얼마나 공통적으로 사용하는지의 정도를 측정하는 것으로 *RCP*(the Ratio of Class-type Parameters to interface's parameters)라 하겠다. 측정값이 컴포넌트의 크기에 영향을 받지 않도록 하기 위해, 전체 매개변수의 수에 대한 비율을 구하며, 기능적 응집성의 상대적 비교를 위해 0에서 1사이의 값으로 정규화 시킨다.

[정의 1] 임의의 한 컴포넌트를 *Co*라 하고, 컴포넌트의 인터페이스가 $Op_1, Op_2, Op_3, \cdots, Op_n$의 n개의 오퍼레이션으로 이루어질 때, 각 오퍼레이션의 매개변수를 P라 하자.

$$RCP(Co) = \frac{\sum_{k=1}^{n} |P_C(Op_k)|}{\sum_{k=1}^{n} |P(Op_k)|}$$

여기서,

$P(Op_k) = \{p \mid p$는 Op_k의 *parameter*$\}$

$P_C(Op_k) = \{p_c \mid p_c$는 Op_k의 *class type*인 *parameter*$\}$

이 *RCP*는 인터페이스가 컴포넌트 내부의 클래스를 얼마나 많이 사용하는지 즉, 인터페이스와 내부 클래스 간에 강한 연관 관계가 있는지를 측정하게 된다. 따라서, 인터페이스가 컴포넌트 내부의 클래스와 얼마나 연관되는지의 응집성을 통해, 컴포넌트의 기능적 독립성을 확인할 수 있다.

속성과 연산으로 이루어지는 클래스는 속성으로 일반 데이터 타입의 변수나 클래스 타입의 변수로 다른 클래스를 사용할 수 있다. 클래스가 변수의 형으로 다른 클래스를 사용할 경우는 클래스들끼리 연관 정도가 높아져 컴포넌트의 기능적 응집성이 높다 할 수 있다. 그러므로, 클래스가 다른 클래스를 변수의 형으로 얼마나 사용하는지에 의해 클래스 타입인 변수의 비율로 컴포넌트의 응집성을 측정할 수 있다.

[정의 2]는 임의의 한 컴포넌트에 대해, 컴포넌트 내부의 클래스가 다른 클래스를 데이터 타입으로 얼마나 사용하는지의 정도를 측정하는 것으로, *RCA*(the Ratio of Class-type Attributes to class's attributes)라 하겠다. 측정값은

컴포넌트 내부의 클래스가 다른 클래스를 데이터 타입으로 사용하는 속성 수를 클래스의 전체 속성 수로 나누기 때문에, 0에서 1사이의 값으로 정규화되며, 컴포넌트 내의 클래스의 수에 영향을 받지 않게 된다.

[정의 2] 임의의 한 컴포넌트를 Co라하고, 컴포넌트 내부에 $C_1, C_2, C_3, \cdots, C_m$의 m개의 클래스가 있을 때, 각 클래스의 속성을 A라 하자.

$$RCA(Co) = \frac{\sum_{k=1}^{m} |A_C(C_k)|}{\sum_{k=1}^{m} |A(C_k)|}$$

여기서,

$A(C_k) = \{a \mid a$는 C_k의 *attribute*$\}$

$A_C(C_k) = \{a_c \mid a_c$는 C_k의 *class type*인 *attribute*$\}$

이 RCA는 컴포넌트 내부의 클래스들 사이에 얼마나 강한 연관 관계가 있는지를 측정하게 된다. 따라서, 클래스의 속성이 클래스 타입의 변수이면, 클래스들끼리 서로 사용 횟수가 많아져 강한 연관 관계를 갖게 되어, 여러 클래스들이 하나의 컴포넌트로 묶이게 된 사실에 대해 명확한 이유를 제시한다 할 수 있다.

4. 평가 및 적용

4.1 평 가

Weyuker[15]는 절차적(procedural) 프로그램을 위한 복잡도 측정에 요구되는 성질들을 정의하였다. 이 성질은 메트릭스의 이론적 평가를 위해 많이 사용되었으나, 객체지향 시스템의 특성을 반영하기에는 부족한 점이 있어 많은 지적을 받아왔다.

따라서, 제안된 컴포넌트 메트릭스의 적합성을 Briand[16]가 제안한 응집도 측정이 가져야만 하는 필요조건적 성질과 Weyuker의 성질 중 Merha[17]가 재정의한 응집도 관련 성질에 적용하여 평가해 본다.

먼저, Briand[16]가 제안한 응집도 측정이 가져야만 하는 필요조건적 성질과 그에 대한 평가는 다음과 같다.

성질 1 : Non-negativity and Normalization

응집도는 구성 요소들 사이에 전혀 관련이 없는 최악의 경우라면 0이므로, 음수(negativity)는 아니다. 그리고, 구성 요소들 사이에 최대의 관련성을 갖는다면, 응집도는 모듈의 크기에 비례하는 최대값을 가지므로, 한 모듈의 응집도는 [0, max]의 구간을 만족한다. 그러나, 응집도가 모듈의 크기에 영향을 받지 않도록 하기 위해서는, 동일한 구간에 응집도 값이 존재하게 하여 서로 다른 모듈들 간에 비교가 가능하도록 정규화(normalization)가 되어야 한다. RCP와 RCA는 비율로 [0, 1]의 구간으로 정규화 되었으므로, 성질 1을 만족한다.

성질 2 : Null Value

모듈의 구성 요소들 사이에 전혀 관련이 없는 최악의 경우라면 응집도는 0이다. 인터페이스가 내부의 클래스를 사용하는 경우가 전혀 없거나, 컴포넌트 내의 클래스가 다른 클래스를 사용하는 경우가 전혀 없다면, RCP와 RCA는 요소들 사이의 관련 정도를 측정하므로 해당되는 경우가 없다면 0이 되므로, 성질 2는 만족한다.

성질 3 : Monotonicity

모듈의 내부에 관계를 더하면 모듈의 구성요소들이 하나의 모듈로 통합(encapsulation)되기 위한 관계의 증가를 의미하므로, 모듈의 응집도를 감소시키지는 않는다. 인터페이스가 내부의 클래스를 사용하는 경우나 컴포넌트 내의 클래스가 다른 클래스를 데이터 타입으로 사용하는 경우가 증가하게 되면, RCP와 RCA는 증가한다. 따라서, 응집도가 감소하지 않고 증가하게 되므로, 성질 3을 만족한다.

성질 4 : Cohesive Modules

서로 관련이 없는 두 모듈의 합성으로 만들어진 새로운 모듈의 응집도는 원래 두 모듈의 최대 응집도 보다 크지 않다. 서로 관련이 없는 두 개의 모듈을 하나로 합칠 경우 즉, 인터페이스의 매개변수가 클래스를 사용하는 경우나 컴포넌트 내의 클래스가 다른 클래스를 데이터 타입으로 사용하는 경우가 새로이 발생하지는 않는다. 따라서, RCP와 RCA의 값은 기존 두 모듈의 값에 대해 증가하지 않으므로, 성질 4를 만족한다.

Merha[17]가 재정의한 응집도 관련 성질과 그에 대한 평가는 다음과 같다.

성질 5 : Renaming

모듈의 이름을 바꾸어도 그 측정값에는 변함이 없다. 제안된 메트릭은 컴포넌트의 이름이나 그 내부 클래스의 이름에 기반을 두지 않으므로, 측정값에 영향을 주지 않기 때문에, 성질 5는 만족된다.

성질 6 : Transitive

서로 다른 모듈의 응집도 값 사이에, 이행적 성질을 만족한다. 제안된 메트릭스는 성질 1 에서 설명되었듯이, 0에서 1사이의 값으로 정규화 되었기 때문에, 측정값들 사이에 상대적 크기 비교로 이행적 성질을 만족하므로, 성질 6은 만족된다.

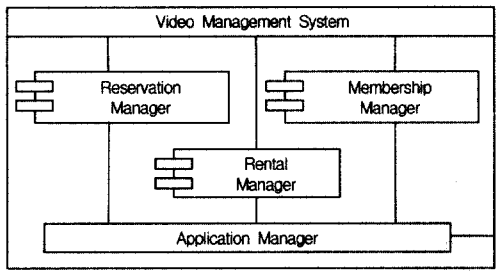
성질 7 : Coarse

응집도가 다른 모듈들이 존재한다는 것을 의미한다. 즉,

서로 다른 측정값을 갖는 컴포넌트들이 존재한다는 것을 의미한다. 한 컴포넌트는 인터페이스와 내부의 클래스들로 구성된다. 따라서, 같은 인터페이스와 같은 클래스들로 구성되며, 또한 같은 관계를 이루며 구성되는 컴포넌트는 하나 이상 존재할 수 없다. 그러므로, 서로 다른 두 컴포넌트로부터 계산된 측정값이 다르다는 사실은 쉽게 알 수 있으므로, 성질 7은 만족된다.

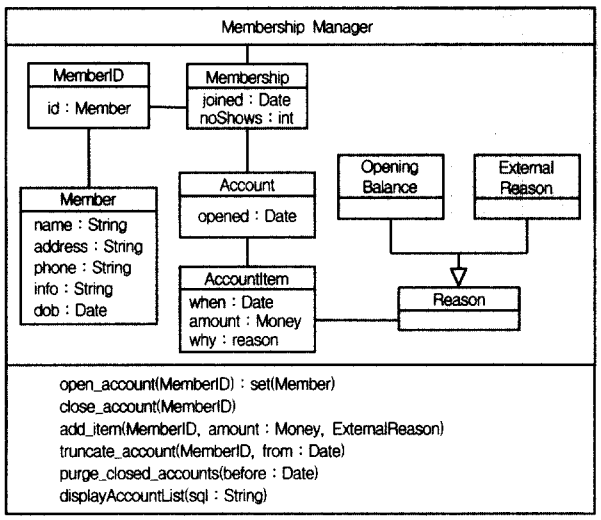
제안된 메트릭스는 컴포넌트 응집성 측정을 위한 메트릭스로서의 필요조건을 모두 만족하므로, 제안된 메트릭스의 적합성에 대한 이론적인 평가가 이루어졌다 할 수 있다.

4.2 사례 연구



(그림 1) 비디오 관리 시스템 구성

(그림 1)은 사례로 택한 비디오 관리 시스템의 구성으로, Reservation Manager, Membership Manager, Rental Manager의 세 컴포넌트와 화면 구성을 위한 Application Manager라는 컴포넌트로 구성된다.



(그림 2) Membership Manager의 구성

(그림 2)는 Membership Manager 컴포넌트 다이어그램으로, 여기에 본 논문에서 제안한 메트릭스를 적용해 컴포넌트의 응집성을 측정해 보겠다. 첫째, 인터페이스에 의한 컴포넌트의 응집도 *RCP*는 인터페이스 오퍼레이션의 전체 매개변수의 수는 9개이고, MemberID, ExternalReason 이라는 클래스가 매개변수로 사용된 수는 5개이므로, $\frac{5}{9}$ 이다. 둘째, 클래스에 의한 컴포넌트의 응집도 *RCA*는 컴포넌트 내부 클래스들의 전체 속성의 수는 12개이고, 그중 다른 클래스를 속성으로 사용하는 속성은 2개이므로, $\frac{2}{12}$ 이다.

사례로 택한 비디오 관리 시스템의 각 컴포넌트 다이어그램에 대해 본 논문에서 제안한 메트릭스를 적용한 결과 측정값은 <표 1>와 같다. 1에 가까운 가장 큰 *RCP* 값을 갖는 Reservation Manager 컴포넌트는 다른 컴포넌트에 비해 내부의 클래스가 인터페이스의 매개변수로 가장 많이 사용되었음을 알 수 있다. 반대로, 가장 작은 *RCA* 값을 갖는 Rental Manager 컴포넌트는 다른 컴포넌트에 비해 내부의 클래스들끼리 서로 연관되는 정도가 적음을 알 수 있다.

〈표 1〉 사례 연구의 결과 측정값

측정 \ 컴포넌트	*Memerbership Manager*	*Rental Manager*	*Reservation Manager*
RCP	*0.556*	*0.625*	*0.917*
RCA	*0.167*	*0.091*	*0.227*

5. 결 론

객체지향 시스템의 기본 단위인 클래스들은 그 상호 작용을 기반으로 하나의 기능을 제공하기 위한 독립적인 소프트웨어 단위인 컴포넌트로 묶일 수 있다. 컴포넌트 인터페이스는 소프트웨어 모듈인 컴포넌트의 서비스를 나타내며, 외부로 보이게 하는 역할을 한다. 존재하는 컴포넌트를 조립하여 시스템을 개발하는 컴포넌트 기반 시스템의 품질은 각 개별 컴포넌트의 품질에 영향을 받는다. 그러므로, 컴포넌트의 보다 폭넓은 활용과 시스템의 품질 향상을 위하여, 각 컴포넌트의 품질을 측정하기 위한 방법이 필요하다.

따라서, 본 논문에서는 컴포넌트의 응집성을 측정하기 위한 새로운 컴포넌트 메트릭스를 제안하였다. 새로운 컴포넌트 메트릭스는 클래스의 속성과 상호 작용하는 메소드의 연관 정도인 클래스의 응집도를 컴포넌트로 확장하여 컴포넌트의 응집성을 측정하였다. 컴포넌트 응집도는 임의의 한 컴포넌트에 대해, 인터페이스와 컴포넌트 내부의 클래스 사이의 연관 정도를 측정하는 인터페이스에 의한 응집도와 내부의 클래스가 속성으로 다른 클래스를 사용하는 정도에 의한 클래스에 의한 응집도의 두 가지로 측정한다. 이는 소프트웨어 개발 주기의 분석단계의 컴포넌트 다이어그램에 적용하여, 향후 개발될 컴포넌트의 기능적 응집 정도를 측정해 볼 수 있다. 컴포넌트의 기능 독립성을 개발 초기 단계에 예측 가능함으로써, 시스템 개발을 위한 이후 단계의 비용과 노력을 줄일 수 있으며, 컴포넌트 재사용을 통한 시스템의 품질 향상을 가져올 수 있는 효과를 기대할 수 있다.

향후 연구과제로 충분한 사례 연구를 통한 메트릭스를 통한 실제적 유효성 검증이 요구된다.

참 고 문 헌

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

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

[3] Clemens Szyperski, "Component Software : Beyond Object-Oriented Programming," ACM Press and Addison-Wesley, 1998.

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

[5] Roger S. Pressman, "Software Engineering, A Practitioner's Approach," 4th ed., McGrawHill, 1998.

[6] 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, 1994.

[7] W. Li, S. Henry, "Object Oriented Metrics Which Predict Maintainability," Journal of Systems and Software, Vol.23, No.2, pp.111-122, Nov., 1993.

[8] J. Bieman, B. K. Kang, "Cohesion and reuse in an object-oriented system," Proceedings of ACM Symposium on Software Reusability (SSR'95), pp.259-262, Apr., 1995.

[9] R. Binder, "Testing Object-Oriented Systems : A Status Report," American Programmer, Vol.7, No.4, pp.22-29, Apr., 1994.

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

[11] D. D'Souza, A. Wills, "Catalysis : Component and Framework based development," Addison-Wesley, 1999.

[12] 김성애, 최완규, 이성주, "객체지향 패러다임에서 저해요인에 기반한 응집도 척도", 정보처리학회논문지, 제7권 제11호, pp.3372-3383, 2000.

[13] 채흥석, 권용래, 배두환, "객체지향 시스템의 클래스에 대한 응집도", 한국정보과학회논문지(B), 제26권 제9호, pp.1095-1105, 1999.

[14] M. H. Samadzadeh, S. J. Khan, "Stability, Coupling, and Cohesion of Object-Oriented Software Systems," Proceedings of the 22nd Annual ACM Computer Science Conference (CSC'94), pp.312-319, Mar., 1994.

[15] E. J. Weyuker, "Evaluating software complexity measures," IEEE Transactions on Software Engineering, Vol.14, No.9, pp.1357-1365, 1988.

[16] L. Briand, S. Morasca, V. R. Basili, "Property-based Software Engineering Measurement," IEEE Transactions on Software Engineering, Vol.22, No.1, pp.68-85, 1996.

[17] Bindu Mehra, "A Critique of Cohesion Measures in the object-Oriented Paradigm," Master thesis, department of computer science, Michigan Technological university, 1997.

고 병 선

e-mail : kobs@sookmyung.ac.kr

1995년 숙명여자대학교 컴퓨터과학과 졸업(학사)

1998년 숙명여자대학교 일반대학원 컴퓨터과학과(석사)

1998년~현재 숙명여자대학교 일반대학원 컴퓨터과학과(박사수료)

관심분야 : 재사용, 메트릭스, 객체지향프로그래밍

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 : 시스템개발방법론, 모델링, 시뮬레이션, 메타DB