독립척도 기반의 비즈니스 컴포넌트 식별

최 미 숙† · 조 은 숙††

요 약

컴포넌트 아키텍쳐 설계를 위하여 재사용 가능한 독립적인 비즈니스 컴포넌트의 식별은 컴포넌트 기반 시스템 구축을 위하여 가장 중요한 작업이다. 그러나 기존 컴포넌트 기반 개발 방법론들의 컴포넌트 식별 방법은 대다수 개발자의 직관과 경험에 의존하고 있다. 또한 개발자들에 의해서 식별된 컴포넌트가 보다 독립적으로 잘 정의되었는지 평가할 기준이 없다. 따라서 본 논문에서는 개발자의 직관과 경험에 의존하는 비즈니스 컴포넌트 식별의 어려운 점을 보완하기 위하여 비즈니스 컴포넌트 식별의 평가 기준이 되는 메트릭을 컴포넌트의 특성을 부여하여 정의한다. 즉, 비즈니스 컴포넌트 식별을 위하여 컴포넌트 내의 응집도는 높고 컴포넌트 간의 결합도는 낮아야 하는 컴포넌트 특성을 적용한 응집척도와 결합척도를 제안한다. 또한 컴포넌트의 응집도와 결합도의 비율에 의하여 비즈니스 컴포넌트의 독립의 정도를 평가할 수 있는 독립척도를 제안한다. 본 논문에서 제안한 응집척도, 결합척도 그리고 독립척도를 사례에 적용하여 그 효율성을 평가한다.

Identification of Business Component based on Independence Metric

Mi-Sook Choi† · Eun-Sook Cho††

ABSTRACT

When constructing a component-based system, it is understood that identifying reusable and independent business components is of utmost importance. However, according to conventional component based developing methodologies, most of developers depend on their experience and/or intuition for identification of business components. Furthermore, there are no criteria to evaluate whether the identified business components are more independently defined or not. Therefore, we propose a component identification metrics to apply to component properties in order to complement the difficulties of identifying business components through developers' experience and/or intuition. The metrics defined are the criteria for identifying the business components and/or for evaluating the identified components. We propose both a cohesion metric, and a coupling metric, to which component properties are applied, wherein those properties can be understood by high cohesion in, and low coupling between, components. Moreover, we propose an independence metric that can evaluate the degree of independence for a particular component by ratio of the cohesion and coupling of components. The metrics that we propose are applied to case study which demonstrates the identification of more independent business components and the validity of our metrics.

키워드 : 컴포넌트(Component), 비즈니스 컴포넌트 식별(Business Component Identification), 응집척도(Cohesion Metric), 결합척도(Coupling Metric), 독립척도(Independence Metric)

1. 서 론

컴포넌트란 독립적인 기능을 수행할 수 있는 부품의 단위이다. 이러한 컴포넌트를 식별하는데 있어서 가장 중요한 점은 기능적 재사용을 위하여 컴포넌트가 유사한 기능들로 그룹화되어야 하며, 컴포넌트와 컴포넌트의 조합에 의해서 새로운 기능을 수행해야 할 때에 한 컴포넌트의 영향이 다른 컴포넌트에 최소한이 되어야 한다. 이러한 특성을 가진 컴포넌트를 식별하기 위해서 기존의 방법론들인 UML Components[1], RUP[3], CBD96[4] 그리고 UNIFACE[6] 등등은 개발자의 직관과 경험에 의하여 컴포넌트를 식별한다. 그러나 직관과 경험에 의하여 컴포넌트를 식별하는 것은 개발자들의 많은 노력과 시간이 필요하고 컴포넌트의 특성에 적합한 독립적인 컴포넌트를 식별하기가 쉽지 않다.

기존 개발 방법론들의 취약점을 보완하기 위해서는 독립적인 컴포넌트를 식별하기 위한 명확한 기준이 필요하다. 컴포넌트는 유사한 기능으로 이루어진 재사용의 단위이어야 하고 컴포넌트와 컴포넌트간의 의존관계는 최소한이 되어야 하므로 보다 독립적인 컴포넌트의 특성은 컴포넌트 내의 응집도는 높고 컴포넌트간의 결합도는 낮아야 한다[19, 20]. 즉, 보다 독립적인 컴포넌트 식별을 위한 기준을 설정하기 위해서는 컴포넌트의 응집척도와 결합척도가 필요하다. 그러나 클래스의 특성과 컴포넌트의 특성이 다르기 때문에 클래스 기반의 응집척도와 결합척도를 컴포넌트 식별을 위한 척도로 적용할 수는 없다. 따라서 보다 독립적인

† 정 회 원 : 우석대학교 컴퓨터공학부 강사
†† 정 회 원 : 동덕여자대학교 정보과학부 교수
논문접수 : 2004년 2월 23일, 심사완료 : 2004년 4월 9일

컴포넌트를 효율적으로 식별하기 위한 명확한 기준을 설정하기 위해서는 컴포넌트의 특성을 반영한 응집척도와 결합척도가 필요하다.

따라서 본 논문에서는 컴포넌트 기반 시스템에서 독립적인 컴포넌트를 잘 식별하고 평가할 수 있도록 컴포넌트 특성을 적용한 응집척도와 결합척도를 정의하고 응집도가 높고 결합도가 낮으면 낮을수록 보다 독립적인 비즈니스 컴포넌트들의 조합으로 시스템이 이루어지므로 본 논문에서는 응집도와 결합도의 비율에 의해서 비즈니스 컴포넌트들의 조합으로 이루어진 한 시스템이 보다 독립적인지를 평가하는 독립척도를 정의한다.

또한 본 논문에서 제안한 척도들을 사례에 적용하고 사례를 통하여 좀 더 독립적인 컴포넌트가 식별됨을 보임으로 그 효율성을 검증한다.

2. 관련 연구

2.1 클래스 기반의 메트릭스

가장 널리 알려진 응집척도로써 가장 근간이 되는 Chidamber-Kermerer의 LCOM(Lack of Cohesion in Methods)[10, 11], Henderson-Sellers가 1995년에 제안한 LCOM*[15], Briand-Daly-Wust가 제안한 Coh[21] 등등의 기존 클래스 기반의 응집척도는 공유 인스턴스 변수를 가지는 메소드쌍의 수 또는 메소드에 의해 참조되는 인스턴스 변수의 수에 의해서 응집척도를 정의하였다. 제시한 응집척도 중 LCOM*와 Coh는 메소드가 인스턴스 변수를 많이 참조하면 할수록 높은 값을 보이고, LCOM은 인스턴스 변수를 공유하는 메소드 쌍의 수가 많으면 많을수록 높은 응집도를 보인다. 그러나 메소드 쌍의 수에 의하여 측정되는 기존의 응집척도 들은 모든 메소드가 서로 공통된 인스턴스를 하나씩만 가지고 있는 클래스와 모든 메소드들이 아주 밀접하게 연관된 클래스나 모두 다 가장 큰 LCOM값인 0을 갖는다는 문제점이 있다. 또한 응집척도가 비율적으로 계산되지 않기 때문에 추출된 응집도의 값의 가치를 전체적으로 판단할 수 없다. LCOM*는 0값일 때 가장 응집도가 높게 측정되고 1값일 때 가장 응집도가 낮게 측정되므로 현상적으로 생각되는 응집도 값과 반대이므로 효율적이지 못하다. 따라서 가장 완벽한 응집척도인 Coh를 본 논문의 컴포넌트 식별 응집척도 정의시 적용한다.

Chidamber-Kermerer의 CBO, RFC, WMC, DIT, NOC는 가장 널리 알려지고 가장 근간이 되는 결합척도들이다[10, 11]. 또한 Henderson-Sellers는 결합척도[15]를 DAC(Data Abstraction Coupling), MPC(Message Passing Coupling) 및 RFC(Response For a Class)의 세 가지 측면으로 나누어 제시하였다. 클래스 기반의 결합척도 중 CBO는 한 클래스와 결합되어 있는 다른 클래스들의 개수로 정의되므로 클래스 간의 메소드 호출 수가 많다 하더라도 1값을 갖는 문제점이 있다. 나머지 MPC나 RFC는 메소드 호출 수가 증가하면서 결합도가 증가하므로 CBO 보다는 훨씬 정확하게 결합도를 측정할 수 있다. 그러나 기존의 결합척도에 의해서 측정된 값들은 0~1 사이로 정규화 되지 않는 문제점이 있다.

2.2 클래스 기반의 메트릭스의 문제점

클래스 기반의 척도를 컴포넌트의 척도로 그대로 적용했을 때의 문제점은 클래스의 구조적 특성과 컴포넌트의 구조적 특성이 다르다는 것이다. 즉, 클래스란 메소드와 인스턴스 변수들로 구성되고 유사한 객체들의 그룹으로 형성된다. 그러나 클래스들 보다 더 큰 개념인 컴포넌트는 기능 및 구조적으로 밀접한 연관성이 있는 클래스들을 그룹화하여 독립적인 단위로 개발되어지고 실행되는 것으로 한 컴포넌트는 밀접한 연관성이 있는 클래스들, 그리고 이 컴포넌트에 의해서 수행되어질 수 있는 기능인 시스템 인터페이스와 시스템 인터페이스의 기능을 실행하기 위한 인터페이스 메소드로 이루어져 있다. 따라서 클래스 기반 메트릭의 측정요소를 메소드와 인스턴스 변수, 메소드와 인스턴스 변수 간의 관계, 클래스, 클래스 간의 관계로 구성되었다면 컴포넌트 기반 메트릭의 측정요소는 클래스, 시스템 인터페이스, 인터페이스 메소드, 클래스와 인터페이스 메소드와의 관계, 컴포넌트, 컴포넌트 간의 관계로 구성되어야 한다는 것이다. 또한 한 컴포넌트는 여러 클래스들로 구성되어 있고 인터페이스 메소드는 이러한 클래스들의 객체를 생성, 삭제, 수정 그리고 참조하여 시스템 인터페이스의 기능을 수행한다. 따라서 컴포넌트 내의 클래스들과 인터페이스 메소드와의 결합의 정도는 인터페이스 메소드가 클래스들을 얼마나 공유하는가와 인터페이스 메소드가 클래스를 생성, 삭제, 수정, 그리고 참조의 유형에 따라서 달라진다. 또한 컴포넌트 간의 결합의 정도도 컴포넌트와 컴포넌트 간의 연관된 에지수 뿐만 아니라 컴포넌트 간에 메시지 호출이 클래스를 생성, 삭제, 수정, 그리고 참조의 유형에 따라 컴포넌트 간의 의존의 강도가 달라진다. 따라서 컴포넌트 내의 인터페이스 메소드와 클래스 간의 관계나 컴포넌트 간의 메시지 호출의 관계에서 인터페이스 메소드가 클래스를 생성, 삭제, 수정 그리고 참조하는 유형에 따라서 관련된 클래스를 그룹화하는데 의미적으로 중요한 요소가 된다. 그러나 클래스 기반의 척도들은 이러한 의미적 요소를 전혀 고려하지 않고 연관된 에지의 수들에 의해서만 결정된다. 따라서 관련 연구에서 제시한 기존 클래스 기반의 척도를 그대로 컴포넌트의 식별에 적용하는 것은 클래스와 컴포넌트의 차이점에 의하여 한계가 있다[14].

그러므로 본 논문에서는 독립적인 후보 컴포넌트를 잘 식별하기 위하여 클래스 기반의 기존 척도에 대한 문제점을 보완하고 컴포넌트의 특성에 적합한 응집척도와 결합척도를 제안하고 응집도와 결합도의 비율에 의해서 후보 컴포넌트들의 독립의 정도를 평가할 수 있는 독립척도를 제안한다.

3. 비즈니스 컴포넌트 식별을 위한 메트릭스

본 논문에서 제안하는 메트릭은 클래스 다이어그램과 유스케이스 다이어그램을 기반으로 컴포넌트의 특성을 부여한 응집척도와 결합척도 그리고 응집도와 결합도의 비율에 의해서 컴포넌트의 독립성을 평가할 수 있는 독립척도이다. 따라서 본 장에서는 컴포넌트의 특성을 제시하고 컴포넌트의 특성을 부여한 결합척도, 응집척도 그리고 독립척도를 정의한다.

3.1 분석 클래스 기반의 비즈니스 컴포넌트의 특성

컴포넌트란 독립적인 기능을 수행할 수 있는 부품의 단위이다. 이러한 컴포넌트를 식별하는데 있어서 가장 중요한 점은 기능적 재사용을 위하여 컴포넌트가 얼마나 유사한 기능들의 그룹으로 이루어져 있으며, 컴포넌트와 컴포넌트의 조합에 의해서 새로운 기능이 수행될 때에 컴포넌트 간의 의존성이 적어서 한 컴포넌트의 영향이 다른 컴포넌트에 최소한이 되어야 한다. 따라서 보다 독립적인 비즈니스 컴포넌트를 식별하기 위하여 메소드의 생성, 삭제, 수정 그리고 참조에 따른 비즈니스 컴포넌트 간의 의존의 강도를 제시하고 이를 비즈니스 컴포넌트 식별 메트릭에 적용한다.

컴포넌트 기반의 시스템은 시스템의 기능을 실행하기 위하여 유스케이스인 시스템 인터페이스로 구성된 시스템 컴포넌트가 존재하고 시스템 컴포넌트는 부품의 단위가 되는 여러 개의 비즈니스 컴포넌트들의 조합으로 이루어져 있다. 따라서 시스템은 비즈니스 컴포넌트들 간의 연관관계를 맺으며 시스템 컴포넌트의 인터페이스 기능을 실행한다. 또한 비즈니스 컴포넌트의 인터페이스 메소드들은 각 비즈니스 컴포넌트가 포함하는 클래스의 객체를 생성, 삭제, 수정 그리고 참조하여 기능을 수행하고 비즈니스 컴포넌트 간에도 생성, 삭제, 수정 그리고 참조의 메시지를 호출함으로 비즈니스 컴포넌트들의 기능을 실행한다.

여러 개의 클래스로 구성된 비즈니스 컴포넌트의 인터페이스 메소드들은 비즈니스 컴포넌트가 포함하는 클래스들의 객체를 조작하여 기능을 수행하는 것이므로 만약 비즈니스 컴포넌트 내에서 객체를 생성시키는 인터페이스 메소드가 존재하지 않아 객체가 생성되지 않는다면 이 객체를 참조하여 기능을 수행하는 다른 인터페이스 메소드들은 실행되어질 수 없고 또한 다른 비즈니스 컴포넌트가 포함하는 인터페이스 메소드에 의해서 비즈니스 컴포넌트의 객체가 생성된다면 두 비즈니스 컴포넌트 간의 관계는 종속적인 관계가 되고 서로 독립적으로 존재할 수 없다. 삭제 역시 마찬가지다.

비즈니스 컴포넌트 간에 수정 메소드 호출에 의해서 기능을 수행한다면 생성과 삭제 메소드 같이 객체를 생성하거나 삭제함으로 비즈니스 컴포넌트의 기능 실행과 객체의 구조에 관한 강한 영향을 부여하지 않고 비즈니스 컴포넌트가 포함하는 각 클래스에 대한 객체의 구조는 그대로 두고 객체 값의 변경만 이루어진다.

비즈니스 컴포넌트 간에 참조 메소드 호출에 의해서 기능을 수행한다면 객체의 구조나 객체 값의 변경을 일으키지 않고 객체의 값만 참조하므로 비즈니스 컴포넌트의 내부 혹은 비즈니스 컴포넌트 간에는 전혀 영향을 받지 않는다.

따라서 시스템 인터페이스의 기능을 수행하기 위해서 비즈니스 컴포넌트간의 의존의 강도는 생성, 삭제가 가장 높고 그 다음이 수정이고 마지막으로 참조의 관계가 된다. 그러므로 시스템 컴포넌트의 인터페이스 기능 실행이 비즈니스 컴포넌트의 생성이나 삭제 메소드 호출에 의해서 이루어진다면생성과 삭제 메소드에 대하여 최대의 가중치를 적용하고 그 다음의 가중치는 수정 메소드 그리고 참조 메소드에 가장 낮은 가중치를 적용함으로 보다 독립적인 비즈니스 컴포넌트를 식별한다.

3.2 비즈니스 컴포넌트 식별을 위한 메트릭스

다음은 본 논문에서 제안하는 비즈니스 컴포넌트 식별 메트릭인 결합척도, 응집척도 그리고 독립척도를 정의한다.

3.2.1 비즈니스 컴포넌트의 결합척도

시스템 컴포넌트를 구성하는 비즈니스 컴포넌트 BC_i와 BC_j가 존재한다면 비즈니스 컴포넌트 BC_i는 시스템 인터페이스의 기능을 실행하기 위하여 비즈니스 컴포넌트 BC_j의 인터페이스 메소드를 호출한다. 이 때 BC_i와 BC_j간에 호출 에지가 존재하고 비즈니스 컴포넌트 BC_i와 BC_j는 연관관계가 존재한다고 정의할 수 있다. 이 때 비즈니스 컴포넌트 BC_i가 호출하는 비즈니스 컴포넌트 BC_j의 메소드 유형은 생성, 삭제, 수정 그리고 참조가 된다. 소프트웨어를 컴포넌트의 단위로 조합하는 부품의 특성과 유지 보수 단계에서 소프트웨어의 수정에 따른 영향의 범위나 수정 영향에 따른 의존의 강도를 줄여야 하는 컴포넌트의 특성상 컴포넌트 간의 연관된 에지 수, 즉 메시지 호출 수 뿐만 아니라 컴포넌트 간의 의존의 강도에 의하여 메시지 호출 유형을 반드시 고려하여야 한다. 따라서 메시지 호출 유형에 따라 컴포넌트 간에 의존의 강도가 달라지기 때문에 보다 독립적인 컴포넌트를 식별하기 위해서는 메시지 호출 유형에 따라서 가중치를 두어야 한다. 또한 비즈니스 컴포넌트 간의 결합척도 뿐만 아니라 시스템의 결합척도 또한 정의한다.

[정의 1] 시스템 컴포넌트를 구성하는 비즈니스 컴포넌트

시스템 컴포넌트는 n개의 비즈니스 컴포넌트인 BC_1, BC_2, BC_3, …, BC_n으로 구성되어 있으므로 하나의 시스템 컴포넌트 S_p를 구성하고 있는 n개의 비즈니스 컴포넌트들의 집합을 다음과 같이 정의한다.

$$\forall p = \{1 \cdots t\} \text{ and } \forall i = \{1 \cdots n\},\ \exists S_p \text{ and } BC_i, \text{ such that } S_p = \{BC_1, BC_2, BC_3, \cdots, BC_n\} \quad (1)$$

[정의 2] 시스템 컴포넌트를 구성하는 시스템 인터페이스

시스템 컴포넌트는 기능을 나타내는 a개의 유스케이스,

즉 a개의 인터페이스, I_1, I_2, I_3, …, I_a로 구성되어 있다. 따라서 하나의 시스템 컴포넌트 S_p를 구성하고 있는 시스템 인터페이스를 SI라 정의하면 시스템 인터페이스는 다음과 같다.

$$^{\forall}k = \{1 \cdots a\} \text{ and } ^{\forall}p = \{1 \cdots t\},\ \exists I_k \text{ and } S_p, \text{ such that}$$
$$SI(S_p) = \{I_1, I_2, I_3, \cdots, I_k\} \quad (2)$$

[정의 3] 비즈니스 컴포넌트를 구성하는 비즈니스 인터페이스

시스템 인터페이스의 기능을 수행하기 위해서 비즈니스 컴포넌트 BC_i가 포함하는 비즈니스 인터페이스는 m개의 인터페이스 메소드들의 집합으로 구성되어 있다. 비즈니스 컴포넌트 BC_i를 구성하고 있는 비즈니스 인터페이스를 BI라 정의하면 비즈니스 인터페이스는 다음과 같다.

$$^{\forall}i=\{1 \cdots t\} \text{ and } ^{\forall}z=\{1 \cdots n\},\ \exists BC_i \text{ and } BM_{iz}, \text{ such that}$$
$$BI(BC_i) = \{BM_{i1}, BM_{i2}, \cdots, BM_{im}\} \quad (3)$$

[정의 4] 비즈니스 컴포넌트 $\mathbf{BC_i}$와 비즈니스 컴포넌트 $\mathbf{BC_j}$사이의 결합척도

인터페이스 메소드가 클래스를 참조하는 유형에 따라서 가중치(Weight) W는 다음과 같이 W(C), W(D), W(W), W(R)라고 정의한다. 이 때 C는 Create, D는 Delete, W는 Write, R는 Read를 의미한다. BC_i의 비즈니스 인터페이스 $\mathbf{BI(BC_i) = \{BM_{i1}, \cdots, BM_{in}\}}$, BC_j의 비즈니스 인터페이스 $\mathbf{BI(BC_j) = \{BM_{j1}, \cdots, BM_{jg}\}}$이라면 BC_i의 비즈니스 인터페이스 메소드가 BC_j의 비즈니스 인터페이스 메소드를 호출하는 에지의 수의 합을 **E**라 하고 다음과 같이 정의한다.

$$E = E1 + E2$$
$$:E1 = \sum Call\ (\overrightarrow{BC_i, BC_j}),\ E2 = \sum Call\ (\overleftarrow{BC_i, BC_j}) \quad (4)$$

따라서 비즈니스 컴포넌트 BC_i의 인터페이스 메소드가 비즈니스 컴포넌트 BC_j를 호출할 경우 하나의 시스템 컴포넌트를 구성하고 있는 컴포넌트 BC_i와 BC_j 사이의 결합도 $CBC(BC_i, BC_j)$는 다음과 같이 정의한다.

O = BC_i의 인터페이스 메소드 개수,
Q = BC_j의 인터페이스 메소드 개수,
BW = 최대 가중치라 정의한다.

$\{BC_i\}(i = 1 \cdots m)$, $\{BC_j\}(j = 1 \cdots n)$라 할 때,

$$CBC(BC_i, BC_j) = \frac{\sum_{k=1}^{z} Wk(BC_i, BC_j)}{O \times Q \times BW} \quad (5)$$

결합척도를 0에서 1 사이로 정규화시키기 위해서 비즈니스 컴포넌트 간의 메시지 호출 유형에 대한 가중치의 합을 컴포넌트 간의 메시지 호출이 가능한 모든 경우의 수에다 최대 가중치를 곱한 값, 즉, 결합도의 최대값으로 나눈 값을 정의한다.

[정의 5] 하나의 시스템 컴포넌트의 결합척도

하나의 시스템 컴포넌트 Sp의 결합도는 Sp를 구성하는 비즈니스 컴포넌트들 간의 총 결합도에 대한 평균 값이다. 시스템 컴포넌트의 결합척도는 다음과 같다.

$\{BC_i\}(i = 1 \cdots m)$, $\{BC_j\}(j = 1 \cdots n)$, $\{CBCx\}(x = 1 \cdots u)$이라 할 때,

$$CBC(S_p) = \frac{\sum_{x=1}^{u} CBC_x(BC_i, BC_j)}{u} \quad (6)$$

3.2.2 비즈니스 컴포넌트의 응집척도

본 논문에서는 클래스 기반의 응집척도 중 가장 완벽한 응집척도인 Coh[21]를 컴포넌트의 구조적 특성에 맞추어 클래스의 인스턴스 변수를 컴포넌트의 클래스로 대치하고 클래스의 메소드를 컴포넌트의 인터페이스 메소드로 대치한다. 또한 컴포넌트 내의 응집도는 인터페이스 메소드가 클래스의 객체를 호출하는 유형에 따라서 달라질 수 있으므로 한 컴포넌트의 인터페이스 메소드가 객체를 호출하는 유형에 따라서 가중치를 적용하여 응집척도를 적용한다. 컴포넌트 기반의 시스템은 여러 개의 컴포넌트 조합으로 시스템의 기능을 실행하므로 한 컴포넌트의 응집척도 뿐만 아니라 시스템의 컴포넌트 응집척도 또한 정의한다.

[정의 6] 비즈니스 컴포넌트를 구성하고 있는 클래스

비즈니스 컴포넌트 BCi는 클래스들의 집합인 Ci를 포함하고 있으므로 q개의 클래스들을 포함하고 있는 비즈니스 컴포넌트는 다음과 같이 정의한다.

$$^{\forall}\{i = 1 \cdots n\}, \text{ and } ^{\forall}\{g = 1 \cdots q\},\ \exists BC_i \text{ and } C_g, \text{ such that}$$
$$BC_i = \{C_1, C_2, \cdots, C_q\} \quad (7)$$

[정의 7] 비즈니스 컴포넌트의 응집척도

인터페이스 메소드가 클래스를 참조하는 유형에 따라서 가중치(Weight), W는 다음과 같이 W(C), W(D), W(W), W(R)라고 정의 한다. 이때 C는 Create, D는 Delete, W는 Write, R는 Read를 의미한다. 비즈니스 컴포넌트의 인터페이스 메소드 집합 $BI(BC_i)$가 비즈니스 컴포넌트의 클래스들의 집합 Cg의 객체를 참조하여 기능을 수행한다고 할 때 한 컴포넌트 내의 응집척도 $CHC(BC_i)$는 다음과 같다.

비즈니스 컴포넌트의 클래스의 개수 = l ,
비즈니스 컴포넌트의 인터페이스 메소드의 개수 =k ,
BW = 최대 가중치라 정의한다.

$\{BC_i\}(i = 1 \cdots n)$라 할 때,

$$CHC(BC_i) = \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} W(BM_i(C_j))}{l \times k \times BW} \quad (10)$$

따라서 본 논문이 제안한 컴포넌트 기반의 응집척도는 한 비즈니스 컴포넌트가 포함하고 있는 클래스와 인터페이스 메소드 간의 관계의 중요도에 따라 가중치를 부여함으로 해서 시스템 인터페이스의 기능을 수행하기 위해서 클래스와 인터페이스 메소드 간의 관계가 강한 결합의 관계를 가지고 있는 경우는 반드시 한 컴포넌트 내로 포함하도록 한다.

[정의 8] 시스템 컴포넌트의 응집척도

시스템 컴포넌트 S_p의 응집척도는 S_p를 구성하는 비즈니스 컴포넌트들의 각 응집도 값에 대한 평균 값이다. 따라서 아래의 수식에서 y는 시스템 컴포넌트 S_p를 구성하는 비즈니스 컴포넌트의 총 개수가 된다.

$S_p(p = 1 \cdots t)$, $BC_q(q = 1 \cdots y)$라 할 때,

$$CHC(S_p) = \frac{\sum_{q=1}^{y} CHC(BC_q)}{y} \quad (11)$$

3.2.3 비즈니스 컴포넌트 식별을 위한 독립척도

잘 정의된 독립적인 컴포넌트는 응집도는 높고 결합도는 낮아야 한다. 따라서 식별된 후보 비즈니스 컴포넌트들이 얼마나 독립적으로 시스템 컴포넌트를 구성하고 있나를 평가할 수 있는 독립척도(Component Independence Metric) CIM(Sp)는 다음과 같이 정의한다.

[정의 9] 시스템 컴포넌트의 독립척도

비즈니스 컴포넌트들로 조합된 시스템 컴포넌트의 독립척도(Component Independence Metric) CIM(Sp)는 다음과 같이 정의한다.

$\{S_p\}(p = 1 \cdots w)$이라 할 때,

$$CIM(S_p) = \frac{CHC(S_p)}{CBC(S_p)} \quad (12)$$

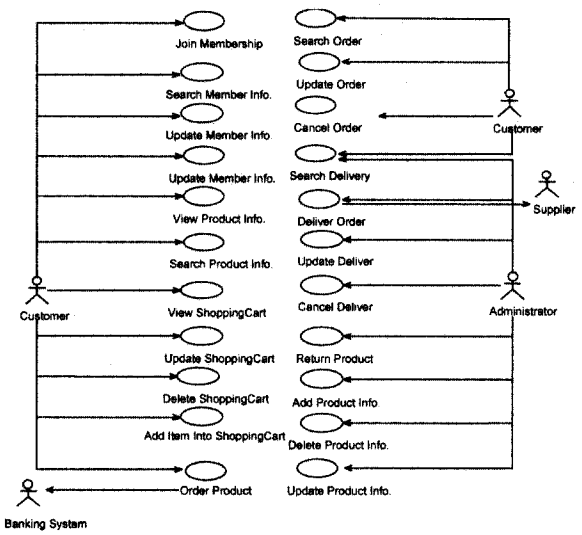
즉, 비즈니스 컴포넌트들로 구성된 한 시스템의 독립척도는 응집도/결합도로 정의 되므로 독립척도의 값이 높을수록 한 시스템의 후보 컴포넌트들은 보다 독립적으로 잘 정의되었다고 결정할 수 있다.

4. 사례 연구

다음은 보다 독립적인 비즈니스 컴포넌트를 식별하기 위하여 본 논문에서 제안한 비즈니스 컴포넌트 식별 메트릭인 응집척도, 결합척도 그리고 독립척도를 전자 상거래 기반의 주문시스템 사례에 적용하여 그 결과를 제시한다.

4.1 전자상거래 도메인의 유스케이스 다이어그램

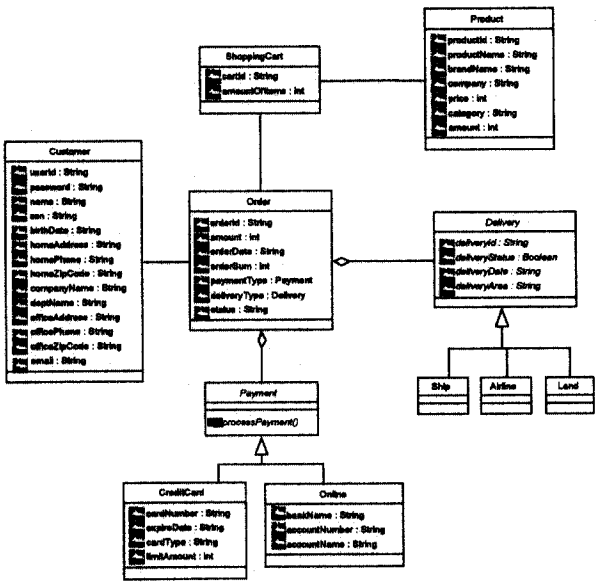
위의 다이어그램은 요구사항 분석 프로세스를 통하여 식별된 주문 시스템에 대한 유스케이스 다이어그램이다. 추출된 유스케이스는 22개이다.



(그림 1) 주문 시스템의 유스케이스 다이어그램

4.2 전자상거래 도메인의 클래스 다이어그램

다음은 주문 시스템 컴포넌트의 각 유스케이스의 이벤트 흐름을 분석하여 RUP 분석 프로세스를 통하여 도출된 클래스 다이어그램이다.



(그림 2) 주문 시스템의 클래스 다이어그램

4.3 주문시스템의 유스케이스와 클래스의 CRWD 메트릭스 테이블

다음은 주문 시스템의 유스케이스인 시스템 인터페이스의 기능을 실행하기 위해서 포함된 클래스의 각 객체에 대하여 생성, 수정, 조회 그리고 삭제 중에 어떠한 메소드를 적용하는지를 나타내는 테이블을 정의한다.

〈표 1〉 주문 시스템의 유스케이스와 클래스 간의 CRWD 매트릭스 테이블

클래스 / 인터페이스	Cuatomer	Order	Payment	Delivery	Shopping Cart	Product
Join Membership	C					
Delete Membership	D					
Update Membership	W					
Search Membership	R					
Order Product	R	C	C	C	C	W
Cancel Order	R	D	D	D	D	W
Update Order	R	W	W	W	W	W
Search Order	R	R	R	R		
Search Delivery	R	R	R	R		
Deliver Order	R	R	R	W		
Update Deliver	R	W	W	W		
Cancel Deliver	A	W	W	D		
View Product						A
Search Product						A
view ShoppingCart					A	
DeleteShoppingCart					D	
Update ShoppingCart					W	A
Add Item to Shopping					C	A
Add Product						D
Update Product						W
Return Product	A	W				W
Delete Product						D

4.4 CRWD 메트릭스 테이블에 가중치 적용과 클래스 간의 메소드 호출관계 정의

〈표 2〉 주문 시스템의 CRWD 메트릭스 테이블에 대한 가중치의 적용

클래스 / 인터페이스	Cuatomer	Order	Payment	Delivery	Shopping Cart	Product
Join Membership	20					
Delete Membership	20					
Update Membership	3					
Search Membership	1					
Order Product	1	20	20	20	20	3
Cancel Order	1	20	20	20	20	3
Update Order	1	3	3	3	3	3
Search Order	1	1	1	1		
Search Delivery	1	1	1	1		
Deliver Order	1	1		3		
Update Deliver	1	3	3	3		
Cancel Deliver	1	3	3	20		
View Product						1
Search Product						1
view ShoppingCart					1	
DeleteShoppingCart					20	
Update ShoppingCart					3	1
Add Item to Shopping					20	1
Add Product						20
Update Product						3
Return Product	1	3				3
Delete Product						20

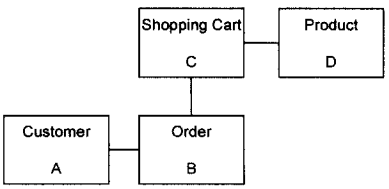
다음은 주문 시스템의 유스케이스 기능을 실행하기 위해서 객체들 간에 메소드 호출관계(A가 B를 호출할 경우 : A → B)를 정의하고 인터페이스 메소드 호출 유형에 따라서 가중치를 부여한다(C : Create=20, D : Delete = 20 > W : Write = 3 >R : Read = 1).

4.5 각 후보 컴포넌트와 후보 시스템에 대한 메트릭의 적용

클래스 다이어그램에서 클래스들 간의 연관관계를 참조하여 가능한 한 모든 경우의 후보 컴포넌트를 도출한다. 그 다음 후보 컴포넌트에 포함된 클래스들이 중복되지 않도록 조합하여 가능한 모든 경우의 후보 시스템을 도출한 후도출된 각 후보 시스템에 대하여 본 논문에서 제안한 응집척도, 결합척도, 독립척도를 적용하여 계산한다. 후보 시스템들 중에 독립도가 가장 높은 후보 시스템에 포함된 컴포넌트들이 비즈니스 컴포넌트로 결정된다.

4.5.1 재 정의된 클래스 다이어그램

클래스 다이어그램에서 Order는 Payment와 Delivery를 포함하고 있으므로 강한 종속관계를 가지므로 반드시 하나의 컴포넌트로 포함되어야 한다. 따라서 Order Class = Order +Payment+Delivery이고 다시 도출된 클래스 다이어그램은 다음 그림과 같다.



(그림 3) 재 정의된 클래스 다이어그램

4.5.2 가능한 한 모든 경우의 후보 컴포넌트

클래스들의 관계를 고려하여 가능한 한 모든 경우의 클래스의 조합으로 이루어진 후보 컴포넌트를 도출한다.

〈표 3〉 가능한 모든 경우의 후보 컴포넌트

후보 컴포넌트	포함된 클래스	후보 컴포넌트	포함된 클래스
C_1	A	C_8	ABC
C_2	B	C_9	BCD
C_3	C		
C_4	D		
C_5	AB		
C_6	BC		
C_7	CD		

4.5.3 컴포넌트 조합으로 이루어진 가능한 한 모든 경우의 후보 시스템

모든 클래스들이 포함되지만 클래스들이 중복되지 않게 후보 컴포넌트들을 조합하여 가능한 모든 경우의 후보 시스템을 추출한다.

〈표 4〉 가능한 모든 경우의 후보 시스템

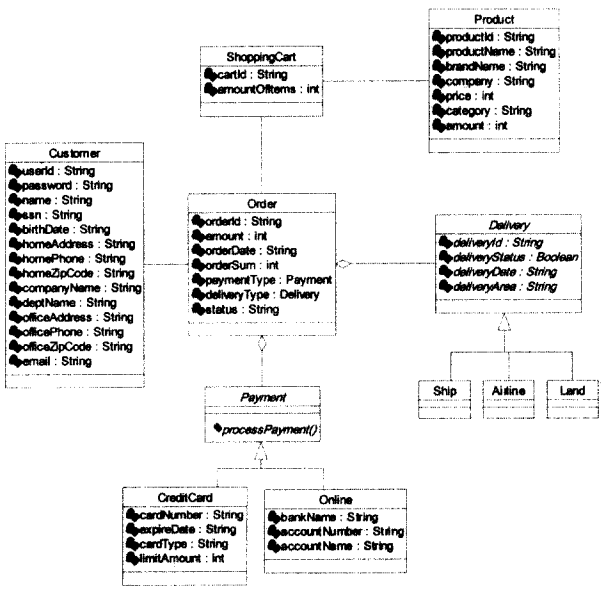
후보 시스템	포함된 후보 컴포넌트	후보 시스템	포함된 후보 컴포넌트
S_0	C_1, C_2, C_3, C_4	S_5	C_8, C_4
S_1	C_5, C_3, C_4	S_6	C_1, C_9
S_2	C_1, C_6, C_4		
S_3	C_1, C_2, C_7		
S_4	C_5, C_7		

4.5.4 메트릭스의 적용과 비즈니스 컴포넌트의 식별

추출된 후보 시스템에 대하여 본 논문에서 제안한 메트릭을 적용하고 가장 독립도가 높은 후보 시스템이 가장 독립적인 후보 컴포넌트들의 조합으로 구성되어 있다. 따라서 가장 독립도가 높은 후보 시스템 안에 포함된 후보 컴포넌트가 비즈니스 컴포넌트로 식별된다. 그러므로 도출된 각 후보 시스템에 대하여 본 논문에서 제안한 응집척도, 결합척도, 그리고 독립척도를 적용하여 비즈니스 컴포넌트를 추출한다.

〈표 5〉 우리의 메트릭스를 후보 시스템에 적용한 결과

후보 시스템	응 집 도	결 합 도	독 립 도
S_0(A, B, C, D)	0.38	0.0113	33.19
S_1(AB, C, D)	0.38	0.0138	27.33
S_2(A, BC, D)	0.26	0.0037	69.76
S_3(A, B, CD)	0.30	0.0147	20.34
S_4(AB, CD)	0.26	0.0201	13.06
S_5(ABC, D)	0.24	0.0037	64.95
S_6(A, BCD)	0.20	0.0038	53.62

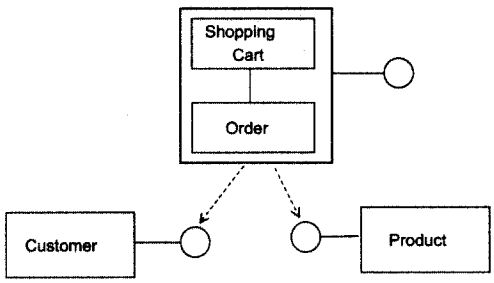


(그림 4) 우리의 메트릭스에 의해서 식별된 비즈니스 컴포넌트들

잘 정의된 컴포넌트는 응집도는 높고 결합도는 낮아야 하므로 평균적으로 독립도가 가장 큰 값이 보다 독립적이다. 따라서 후보 시스템 S_2가 가장 독립도가 높으므로 S_2가 포함하고 있는 후보 컴포넌트들이 비즈니스 컴포넌트들로 식별된다. 그러므로 결정된 시스템은 후보 시스템 중 독립도가 가장 높은 S2가 결정되고 식별된 비즈니스 컴포넌트는 A, BC, D가 결정되었다. A 비즈니스 컴포넌트는 Customer가 되고, BC로 이루어진 비즈니스 컴포넌트는 Order와 Shopping Cart가 되고, D컴포넌트는 Product이 된다. 따라서 그 결과는 (그림 4)와 같다.

4.6 결정된 컴포넌트 아키텍처

다음은 본 논문에서 제안한 메트릭스를 적용하여 식별된 비즈니스 컴포넌트를 중심으로 정의된 컴포넌트 아키텍처이다.



(그림 5) 주문 시스템의 컴포넌트 아키텍처

5. 제안한 컴포넌트의 특성을 적용한 메트릭스의 분석, 기대효과 및 비교평가

다음은 본 논문에서 제안한 메트릭스를 전자상거래 기반의 주문시스템에 적용하였을 경우와 클래스 기반의 객체지향 메트릭을 그대로 적용하였을 경우의 컴포넌트 식별 결과를 분석하고 기대효과 및 비교평가를 제시한다.

5.1 제안한 컴포넌트의 특성을 적용한 메트릭스의 분석

본 논문에서 제안한 컴포넌트 식별 메트릭스를 평가하기 위하여 본 절에서는 전자 상거래 도메인의 주문 시스템을 중심으로 적용된 클래스 기반의 객체지향 메트릭스의 결과와 본 논문에서 제안한 비즈니스 컴포넌트 식별 메트릭스의 결과를 중심으로 비교 분석하였다.

5.1.1 클래스 기반의 메트릭스를 적용하였을 경우

클래스 기반의 메트릭스의 적용은 클래스 기반의 응집척도 중 가장 완벽한 응집척도인 Henderson-Sellers[15]의 LCOM*를 보완하여 응집도가 높을수록 1 값을 갖고 응집도가 낮을수록 0 값을 갖도록 변형한 Coh를 적용하고 결합척도는 Henderson-Sellers[15]가 제안한 MPC(Message Passing Coupling)를 적용한다. 이 때 MPC는 0~1 사이로 정규화 되지 않아 독리도를 측정할 수 없으므로 본 논문에서는 MPC의 결과 값을 0과 1 사이로 정규화시켜 적용하였다.

Henderson-Sellers[15]가 제안한 응집척도와 결합척도는 클래스 기반의 메트릭스 중 가장 대표적인 Chidamber-Kemmerer [10, 11]가 제안한 응집척도 LCOM과 CBO의 문제점을 보완한 가장 완벽한 메트릭스이다. 클래스 기반의 메트릭스를 적용하여 비즈니스 컴포넌트를 식별한 결과는 다음과 같다.

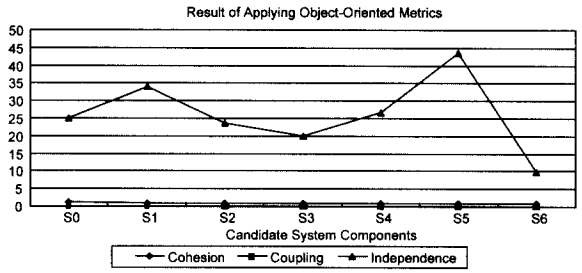
〈표 6〉 가중치를 적용하지 않은 주문 시스템의 매트릭스 테이블

클래스 / 인터페이스	A	B			C	D
	Cuatomer	Order	Payment	Delivery	Shopping Cart	Product
Join Membership	1					
Delete Membership	1					
Update Membership	1					
Search Membership	1					
Order Product	1	1	1	1	1	1
Cancel Order	1	1	1	1	1	1
Update Order	1	1	1	1	1	1
Search Order	1	1	1	1		
Search Delivery	1	1	1	1		
Deliver Order	1	1		1		
Update Deliver	1	1	1	1		
Cancel Deliver	1	1	1	1		
View Product						1
Search Product						1
View ShoppingCart					1	
DeleteShoppingCart					1	
Update ShoppingCart					1	1
Add Item to Shopping					1	1
Add Product						1
Update Product						1
Return Product	1	1				1
Delete Product						1

〈표 7〉 객체지향 메트릭스를 적용한 결과

후보시스템	응 집 도	결 합 도	독 립 도
S_0(A,B,C,D)	1.00	0.0402	24.90
S_1(A,B,C,D)	0.95	0.0279	34.01
S_2(A,B,C,D)	0.89	0.0377	23.60
S_3(A,B,C,D)	0.90	0.0449	20.00
S_4(A,B,C,D)	0.77	0.0289	26.65
S_5(A,B,C,D)	0.78	0.0180	43.48
S_6(A,B,C,D)	0.75	0.0789	9.75

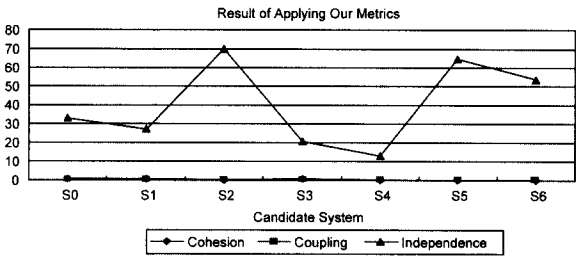
먼저 클래스 기반의 객체지향 메트릭스를 주문 시스템에 적용해 본 결과를 보면 클래스 A는 Customer, 클래스 B는 Order, 클래스 C는 Shopping Cart 그리고 클래스 D가 Product이라고 할 때 독립도가 가장 높은 시스템은 S5(ABC, D)이고 그 다음이 S1(AB, C, D)이다. 이 때 클래스 간의 에지수는 BD < CD < BC < AB 순으로 많다. 이 결과는 컴포넌트의 특성을 부여한 의미적 가중치를 적용하지 않고 객체지향 메트릭을 그대로 적용하였기 때문에 클래스 간의 에지 수가 가장 많은 클래스들이 컴포넌트로 그룹화 되었을 때 높은 독립도를 나타내고 있다. 따라서 클래스 기반의 메트릭을 적용하였을 경우에 클래스와 클래스간의 연관된 에지의 수에 의해서만 독립도가 결정되므로 식별된 컴포넌트는 직관적으로 생각하는 컴포넌트와 완전히 다른 컴포넌트가 식별됨을 알 수 있었다.



(그림 6) 객체지향 메트릭스를 적용한 결과 분포도

5.1.2 본 논문에서 제안한 메트릭스를 적용하였을 경우 (4장 참조)

또한 본 논문에서 제안한 비즈니스 컴포넌트 식별 메트릭스를 주문 시스템에 적용해 본 결과를 보면 독립도가 가장 높은 시스템은 S2(A, BC, D)이다. 이 때 클래스 간의 에지수를 보면 A 클래스와 B 클래스 사이의 메시지 호출 수가 가장 많음에도 A 클래스와 B 클래스가 분리된 형태의 후보 비즈니스 컴포넌트의 조합으로 구성된 시스템 S2가 독립도가 가장 높았다. 즉, 본 논문에서 제안한 메트릭에 의해서 도출된 결과 값을 보면 후보 시스템의 결과 값이 논리적으로 타당하게 분포되어 있음을 알 수 있고 우리의 직관과 일치되는 컴포넌트가 식별됨을 알 수 있다.



(그림 7) 제안된 메트릭스를 적용한 결과 분포도

따라서 클래스와 클래스 간의 메시지 호출이 생성이나 삭제의 유형이라면 메시지 호출 수가 1이라 하더라도 강한 연관관계를 가지고 있으므로 그룹화 되어야 하는데 이러한 관계는 단순히 클래스 간의 메시지 호출 수에 의해서 계산되는 클래스 기반의 객체지향 메트릭스로는 인식될 수 없으므로 컴포넌트의 특성이 부여되지 않는 객체지향 메트릭스를 이용해서는 비즈니스 컴포넌트를 올바르게 식별할 수 없다.

그러므로 본 논문에서 제안하고 있는 컴포넌트의 특성을 부여한 메트릭을 적용하여 보다 독립적인 비즈니스 컴포넌트를 식별하는 것은 타당한 것이다.

5.2 기대효과 및 비교 평가

위에서 제시한 분석 결과를 종합해 보면 본 논문에서 제안하는 컴포넌트 식별을 위한 메트릭스의 기대효과는 다음과 같이 정리할 수 있다.

① 본 논문에서 제안한 응집척도와 결합척도를 0~1 사이로 정규화 시키고 독립척도를 제안함으로 응집도와 결합도의 비율에 의해서 비즈니스 컴포넌트들이 얼마나 독립적인가를 측정할 수 있다.

② 본 논문에서 제안한 컴포넌트 식별 방법은 클래스들 간의 수직적관계와 수평적 관계를 다 고려하고 컴포넌트의 특성을 적용한 의미적 가중치를 척도에 적용하여 컴포넌트를 식별함으로 보다 더 독립적인 비즈니스 컴포넌트가 식별될 수 있다.

③ 비즈니스 컴포넌트들의 가능한 모든 경우의 조합에 의한 후보 시스템의 응집도, 결합도 그리고 독립도를 측정함으로 전체적으로 후보 시스템이 포함하고 있는 비즈니스 컴포넌트들의 독립의 정도를 판단할 수 있으므로 개발자가 정량적 결과에 의해서 비즈니스 컴포넌트를 좀 더 용이하게 식별할 수 있다.

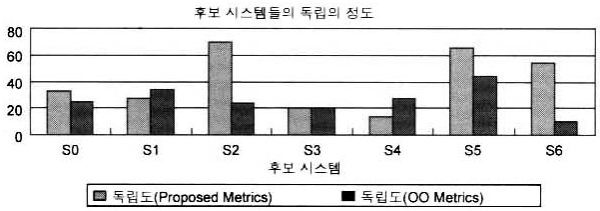
④ 개발자가 직관에 의해서 비즈니스 컴포넌트를 식별한 후 식별된 컴포넌트가 독립된 비즈니스 컴포넌트들로 구성 되었는지를 확인하고 평가할 때에 유용하게 적용되어질 수 있다.

다음은 몇 가지의 비교 요소들에 의하여 본 논문에서 제안된 메트릭스와 객체지향 메트릭스의 비교 결과를 제시함으로 우리가 제안한 비즈니스 컴포넌트 식별 메트릭스를 평가한다. 특히 객체지향 메트릭스는 객체와 컴포넌트의 특징과 관련된 다양한 특성을 고려하지 않는다. 그러나 우리의 메트릭스는 그러한 다양한 특성들을 고려함으로 더욱 타당한 비즈니스 컴포넌트를 식별하고 평가한다.

〈표 8〉 객체지향 메트릭스와 제안된 메트릭스의 비교 평가

메트릭스 / 비교요소	○○ 메트릭스	제안된 메트릭스
가중치	고려되지 않음	고려됨
메소드 호출 수	고려됨	고려됨
메소드 호출 방향	고려되지 않음	고려됨
메소드 호출의 유형	고려되지 않음	고려됨
클래스 간의 관계적 특성	고려됨	고려됨
고려 범위	클래스	고려됨
컴포넌트의 특성	고려되지 않음	고려됨
컴포넌트 식별 결과의 타당성	낮다	높다
메트릭스의 정규화	부분적으로 고려됨	고려됨
식별 컴포넌트의 독립의 정도	고려되지 않음	고려됨

다음 그림은 후보 시스템들이 얼마나 독립적인 비즈니스 컴포넌트들로 구성되었는지를 평가하기 위하여 본 논문에서 정의한 독립도를 적용한 결과와 객체지향 메트릭스에 의하여 응집도와 결합도를 측정한 후 본 논문에서 정의한 독립도를 적용한 결과의 분포표이다. 독립도가 가장 높은 값을 가진 시스템이 가장 독립적인 비즈니스 컴포넌트들로 구성되어 있으므로 가장 독립도가 높은 시스템에 포함된 비즈니스 컴포넌트들이 우리가 식별해야 내야 하는 보다 독립적인 비즈니스 컴포넌트가 된다. 따라서 다음의 분포표를 보면 본 논문에서 제안한 결과가 타당함을 알 수 있다.



(그림 8) 독립도(Independence Metric)의 비교 결과

6. 결 론

컴포넌트 기반 시스템의 목적은 기능적으로 재사용이 가능한 단위로 소프트웨어가 구축되는 것이며 유지보수 단계에서 소프트웨어의 수정에 대한 영향이 시스템 전체에 미쳐서 이미 구축된 시스템을 사용할 수 없도록 만드는 문제점을 보완하기 위하여 보다 독립적인 컴포넌트 단위로 소프트웨어를 구축하는 것이다. 따라서 컴포넌트 기반 시스템의 부품 단위가 되는 비즈니스 컴포넌트는 기능적 재사용이 가능하도록 응집도가 높아야 하고 소프트웨어의 수정에 대한 영향이 국부적으로 미치도록 보다 독립적이어야 하므로 컴포넌트 간의 결합도는 낮아야 한다. 이러한 특성을 가진 컴포넌트들로 시스템을 구축했을 때 컴포넌트 기반 시스템의 목적이 달성된다. 따라서 컴포넌트 기반 시스템에서 가장 중요한 점은 보다 독리적인 비즈니스 컴포넌트를 식별하는 것이다.

따라서 컴포넌트 기반 시스템에서 독립적인 비즈니스 컴포넌트를 효율적으로 식별하고 평가하기 위해서는 응집척도와 결합척도가 반드시 필요하다. 그러나 객체지향 시스템의 클래스와 컴포넌트 기반 시스템의 컴포넌트 특성은 다르기 때문에 객체지향 시스템의 척도를 그대로 적용할 수 없으므로 컴포넌트 특성에 적합한 척도가 필요하다. 따라서 본 논문에서는 보다 독립적인 비즈니스 컴포넌트의 식별을 위해서 컴포넌트 특성에 적합하도록 메소드 호출 타입에 따라서 의비적 가중치를 부여한 응집척도, 결합척도를 제안하고 응집도와 결합도의 비율에 의해서 시스템의 독립의 정도를 측정할 수 있는 독립척도를 제안하였다. 또한 전자상거래를 위한 주문 시스템 사례에 적용하여 본 논문에서 제안한 메트릭스에 의하여 보다 독립적인 비즈니스 컴포넌트가 식별되는지를 평가하였다. 제안한 메트릭스를 사례에 적용해 본 결과 직관적으로 독립적인 비즈니스 컴포넌트를 식별한 결과와 제안한 메트릭스를 적용하여 추출된 비즈니스 컴포넌트가 같음을 보였다.

또한 객체지향의 척도를 그대로 사례에 적용해 본 결과 직관적으로 컴포넌트를 식별한 결과와 올바르지 않은 결과를 보였으며 컴포넌트의 특성을 적용하지 않으므로 해서 더욱 더 정밀하게 값이 산출되지 않음을 알 수 있었다.

또한 개발자들의 직관과 경험에 의하여 컴포넌트를 식별하였다 하더라도 식별된 컴포넌트가 잘 식별되었는지 평가할 방법이 없다. 그러나 본 논문에서 제안하는 방법은 모든 경우의 후보 비즈니스 컴포넌트를 도출하고 도출된 후보 비즈니스 컴포넌트를 조합하여 모든 경우의 후보 시스템을 도출한다. 도출된 각 후보 시스템에 대하여 응집도, 결합도, 그리고 독립도를 산출하기 때문에 어떠한 형태로 비즈니스 컴포넌트가 조합되어야 시스템이 가장 독립적인지 판단할 수 있고 개발자들의 직관과 경험에 의하여 컴포넌트를 식별한 결과가 어느 정도의 독립도를 가지고 있는지 평가할 수 있다. 또한 컴포넌트의 크기(Granularity)를 결정하는데도 상당히 도움을 줄 수 있다.

향후 연구과제로는 본 논문에서 제안한 컴포넌트의 응집척도와 결합척도를 많은 실무 데이터를 이용한 실험을 통하여 그 타당성을 좀 더 입증하고 취약점을 보완해 나가는 것이다.

참 고 문 헌

[1] John Chessman, John Daniels, UML Components, Addison-Wesley, pp.67-120, 2001.

[2] Desmond Francis Dsouza, Alan Cameran Wills, Objects, Component, and Frameworks with UML : the Catalysis approach, Addison Wesley, 1999.

[3] Ivar Jacobson, Grady Booch, James Rumbaugh, The Unified Software Development Process, Addison-Wesley, 1999.

[4] John Dodd, "Identifying & Scoping CBD96 Components," Texas Instruments Inc., 1999.

[5] Cool Software Korea, CBD Project Guide using Cool, Cool Software Korea, 2000.

[6] Compuware corp., "UNIFACE Development Methodology : UNIFACE V. 7.2," Compuware Corp., 1998.

[7] W. Burg, S. Hawker, D. Hale, K. Mclnnis, A. Parrish, S. Sharpe, R. Woolridge, "Exploring a Comprehensive CBD Method, Use of CBD/e in Practice.," The 3rd International Workshop on Component-Based Software Engineering., 2000.

[8] Michael Siff and Thomas Reps, "Identifying Modules via Concept Analysis," ICSM97, 1997.

[9] Misook Choi, Hyunhee Koh, Yongik Yoon, Jaenyun Park, "Component Identification based on Usecase.," Proceeding of the ACIS on Computer and Information Science, pp.203-210, 2001.

[10] S. R. Chidamber and C. F. Kemerer, "Towards a Metrics Suite for Object-Oriented Design," OPSLA91, Phoenix, Arizona, USA, pp.197-211, 1991.

[11] S. R. Chidamber and C. F. Kemerer, "A Metric Suite for Object-Oriented Design," IEEE Transactions on Software.

[12] M. Lorenz and J. Kidd, Object-Oriented Software Metrics, A Practical Guide, Prentice-Hall, 1994.

[13] Snelting, G. and TIP. F., "Reengineering class hierarchies using Concept Analysis.," In Foundations of software Engineering, FSE-6, ACM, pp.99-110, 1998.

[14] Eunsook Cho, Minsun Kim, Soodong Kim, "Component Metrics to Measure Component Quality," IEEE APSEC, 2001.

[15] Henderson-Sellers, Brian, Object-Oriented Metrics, Prentice-Hall, 1996.

[16] D. Kung, Jerry Gao, Pei Hsia, F. Wem, Y. Toyoshima and C. Chen, "Change Impact Identification in Object Oriented Software Maintenance," Proceedings of International Technical Conference on Circuit/Systems, Computers and Communications, 1999.

[17] Hyungho Kim, Doohwan Bae, "Component Identification via Concept analysis," Journal of Object Oriented Programming, 2001.

[18] Lee Sang Duck, Yang Young Jong, Cho Eun Sook, Kim Soo Dong, "COMO : A UML-Based Component Development Methodology," Proceeding of IEEE APSEC, pp.54-61, 1999.

[19] S. Mancoridis, B. S. Mitchell, Y. Calm, E. R. Gransner, "Bunch : A Clustering Tool for the Recovery and Maintenance of Software System Structures," ICSM, 1999.

[20] Jong Kook Lee, Seung Jae Jung, Soo Dong Kim, "Component Identification Method with Coupling and Cohesion," Proceeding of IEEE APSEC, 2001.

[21] 채홍석, 권용래, 배두환, "객체지향 시스템의 클래스에 대한 응집도", 한국정보과학회논문지, 제2권 제9호, pp.1095-1104, 1999.

최 미 숙

e-mail : khc67_kr@hanmail.net
1990년 전북대학교(이학사)
1994년 숙명여자대학교 일반대학원 컴퓨터과학과(이학석사)
2002년 숙명여자대학교 일반대학원 컴퓨터과학과(이학박사)
2003년~현재 우석대학교 컴퓨터공학부 강사
관심분야 : 소프트웨어 개발 방법론, CBD 방법론, 소프트웨어 아키텍쳐

조 은 숙

e-mail : escho@dongduk.ac.kr
1993년 동의대학교 자연과학대학 전산통계학과(이학사)
1996년 숭실대학교 공과대학 컴퓨터학과(공학석사)
2000년 숭실대학교 공과대학 컴퓨터학과(공학박사)
2002년~2003년 한국전자통신연구원 초빙연구원
2000년~현재 동덕여자대학교 정보과학부 강의전임교수
관심분야 : 소프트웨어 모델링, CBD 방법론, 소프트웨어 아키텍처, 웹서비스 컴퓨팅