컴포넌트 설계를 위한 결합도 메트릭

최 미 숙[†] · 이 종 석[††] · 송 행 숙[†††]

요 약

소프트웨어 개발의 높은 생산성을 향상시키기 위한 재사용 기술로 컴포넌트 기반 개발 방법론은 널리 사용되게 되었다. 컴포넌트의 재사용을 향상시키기 위해서는 설계된 컴포넌트가 측정가능 해야 하므로 컴포넌트의 품질을 정량적으로 평가할 메트릭스가 필요하다. 따라서 본 논문에서는 컴포넌트의 특성을 반영한 컴포넌트의 결합도 메트릭을 제안한다. 또한 제안된 결합도 메트릭의 정확성을 검증하기 위해 사례연구를 제시하고 기존 결합도 메트릭스와의 비교 분석 결과를 제시한다. 제안된 결합도 메트릭은 좀 더 정확하게 컴포넌트의 품질을 평가하고 Briand이 제시한 결합도 메트릭의 필요조건을 만족한다.

키워드 : 컴포넌트 설계, 컴포넌트의 품질, 결합도, 컴포넌트의 평가

A Coupling Metric for Design of Component

Mi-Sook, Choi[†] · Jong-Seok, Lee[††] · Song Haeng-Sook[†††]

ABSTRACT

The component-based development methodology becomes famous as the reuse technology to improve the high productivity of software development. It is necessary component metrics for component-based systems, because the designed components should be measurable to improve the quality of the software. Therefore this paper propose a coupling metric for component design which is reflected in characteristics of component. This paper suggest a case study and comparative analysis result about conventional metrics to verify the accuracy of our coupling metric. The proposed coupling metric measure the quality of components accurately and satisfies necessary conditions of coupling metric suggested by Briand and others.

Key Words : Component Design, Quality of Component, Coupling of Component, Evaluation of Component

1. 서 론

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 고 품질의 시스템을 효율적으로 개발할 수 있는 기술이다[1]. 따라서 컴포넌트를 설계하는데 있어서 가장 중요한 점은 기능적 재사용을 위하여 컴포넌트가 얼마나 유사한 기능들의 그룹으로 이루어져 있으며, 컴포넌트와 컴포넌트의 조합에 의해서 새로운 기능을 수행해야 할 때에 한 컴포넌트의 영향이 다른 컴포넌트에 최소한이 되어야 한다. 즉, 컴포넌트 간에 결합도가 낮을수록 변경의 파급효과가 국소적이므로 컴포넌트의 설계를 수정하기가 쉽고 유지보수단계에서 소프트웨어가 효율적으로 관리되어질 수 있다. 또한 소프트웨어 메트릭을 사용한 정량적인 방법은 잠재해 있는 설계 결점을 발견하고 결점을 정정하기 위한 변형을 찾는데 개발자들에게 도움을 준다[2]. 따라서 보다 독립적인 컴포넌트를 설계하기 위해서는 컴포넌트의 결합도를 측정할 메트릭이 필요하다.

그러나 기존의 컴포넌트의 결합도 메트릭은 객체지향 기반의 클래스의 결합도를 그대로 적용하거나 약간만 수정하여 사용하는 경우가 대부분이기 때문에 보다 정확하게 컴포넌트의 결합도를 평가할 수 없다. 따라서 본 논문에서는 객체지향 기반의 클래스의 특성과 다른 컴포넌트의 특성을 찾아내고 그 특성을 적용한 결합도 메트릭을 제안한다. 본 논문에서 제안한 결합도 메트릭은 컴포넌트 분석단계나 설계단계에서 식별된 컴포넌트를 정량적으로 평가함으로써 좀 더 독립적인 컴포넌트를 설계하는데 개발 시간과 노력을 절감할 수 있다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존의 컴포넌트의 결합도와 클래스 기반의 결합도 메트릭스 방법과 문

† 정 회 원 : 우석대학교 컴퓨터공학과 초빙교수
†† 정 회 원 : 우석대학교 컴퓨터공학과 부교수
††† 정 회 원 : 한일장신대학교 건강생명정보학부 학부장
논문접수 : 2005년 4월 6일, 심사완료 : 2005년 5월 20일

제점을 살펴본다. 3 장에서는 컴포넌트의 특성에 따른 결합도를 정의하고 이론적으로 타당하다는 것을 제시한다. 4 장에서는 사례연구 및 비교 분석을 통해서 제안된 메트릭의 정확성 및 유효성을 평가한다. 5 장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

Chidamber-Kemerer의 CBO(Coupling Between Object Classes), RFC(Response For a Class)[3]는 가장 널리 알려지고 가장 근간이 되는 클래스 기반의 결합도 메트릭스들이다. 그 중 CBO는 한 클래스와 결합되어 있는 다른 클래스들의 개수로 정의된다. 두 클래스가 결합되었다는 것은 한 클래스 안에 선언된 메소드가 다른 클래스에 정의된 메소드 또는 인스턴스 변수를 사용하는 경우를 말한다. CBO의 값이 크면 클래스 설계의 모듈화와 재사용을 어렵게 만들며 변경에 대한 파급효과가 커서 유지 보수를 어렵게 한다. RFC는 한 클래스에서 다른 클래스에 대해 메소드 호출이 일어나는 것을 계산한다. RS가 한 클래스의 응답 집합(Response Set)이라고 할 때 RFC는 $RFC = |RS|$로 정의된다. 여기에서, $RS = \{M\} \cup all_i \{R_i\}$이다. 이때 $\{M\}$은 그 클래스 안에 선언된 메소드의 집합이고, $\{R_i\}$는 메소드 i에 의하여 호출되는 메소드들의 집합이다. 이는 클래스의 반응정도를 나타내며 호출되는 메소드가 많아질수록, 클래스의 테스트와 디버깅이 복잡하며 유지보수를 어렵게 만든다. Henderson-Sellers가 제안한 클래스 기반의 대표적인 결합척도[4]는 MPC(Message Passing Coupling)이다. MPC는 클래스 내에 정의된 send 문장의 수로 정의된다. 클래스 내에 정의된 send 문장의 수가 많아질수록 클래스의 테스트와 디버깅이 복잡하며 유지보수를 어렵게 만든다.

Lee[5]가 제안하는 컴포넌트의 결합도는 컴포넌트 식별을 위한 결합도로서 시퀀스 다이어그램에서 호출된 메소드의 호출 수를 이용한 동적 결합도와 클래스 간의 정적관계를 통한 정적 결합도를 통하여 컴포넌트의 결합도를 정의한다. Koh[6]는 기능을 실행하기 위해서 컴포넌트와의 상호작용 관계를 측정할 수 있도록 컴포넌트 사이에서 클래스의 오퍼레이션이 얼마나 공유되느냐의 정도로 결합도를 정의한다. Kim[7]은 기능 중에서 여러 개의 클래스들의 상호작용에 의해서 실행되어질 수 있는 기능을 조인트 액션(Joint Action)이라 정의하고 Kim이 제안한 결합도는 컴포넌트 간에 공유하는 조인트 액션의 수로 결합도를 정의한다. 따라서 기존의 컴포넌트의 결합도를 보면 단순히 메소드 호출 수, 컴포넌트 간에 공유하는 오퍼레이션의 수나 기능의 수를 사용하여 결합도를 정의한다. 즉, 기존의 컴포넌트의 결합도는 클래스 기반의 결합도와 거의 유사하다고 볼 수 있다.

그러나 클래스 기반의 메트릭스를 컴포넌트의 메트릭스로 그대로 적용했을 때의 문제점은 클래스의 구조적 특성과 컴포넌트의 구조적 특성이 다르다는 것이다. 클래스는 메소드와 속성의 합으로 구성되었지만 컴포넌트는 기능 및 구조적으로 밀접한 연관성이 있는 클래스들을 그룹화 하여 독립적인 단위로 개발되어지고 실행되는 것으로 한 컴포넌트는 밀접하게 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 메트릭스의 측정 요소가 클래스가 포함하고 있는 메소드와 속성의 관계를 중심으로 측정되어진다면 컴포넌트는 밀접하게 연관된 클래스들의 그룹으로 컴포넌트가 형성되어 있는지를 평가하기 위해서 클래스와 클래스 간의 정적, 동적 상호작용 관계, 컴포넌트의 기능 실현을 위한 클래스들과 인터페이스 메소드와의 공유관계, 그리고 컴포넌트와 컴포넌트 간의 관계를 중심으로 측정되어져야 한다는 것이다. 또한 컴포넌트는 여러 클래스들로 구성되어 있으므로 컴포넌트 내부나 외부의 클래스들의 객체를 생성, 삭제, 수정 그리고 참조하여 컴포넌트의 기능을 수행한다. 따라서 컴포넌트 간의 결합의 정도는 컴포넌트와 컴포넌트 간의 연관된 에지 수뿐만 아니라 컴포넌트 간에 메시지 호출의 유형 또한 고려하여야 한다. 그러나 클래스 기반의 결합도 메트릭스들과 기존의 컴포넌트의 결합도 메트릭스는 이러한 의미적 요소를 전혀 고려하지 않고 연관된 에지의 수나, 연관된 클래스들의 수에 의해서만 결정된다. 따라서 클래스 기반의 결합도 메트릭스나 기존의 컴포넌트 결합도 메트릭스로는 정확하게 컴포넌트를 평가할 수 없다.

3. 컴포넌트 평가를 위한 결합도 메트릭

본 절에서는 컴포넌트의 특성을 적용한 컴포넌트의 결합도 메트릭스를 정의하고 정의한 결합도 메트릭이 이론적으로 타당하다는 것을 검증하기 위하여 Briand이 제안한 프레임워크[8]에 적용하여 증명한다.

3.1 컴포넌트의 특성

클래스와 클래스의 관계는 연관관계, 상속관계 그리고 포함관계가 존재한다. 이 때 클래스들 간의 의존은 $<C_2, C_1> \in R$은 C_1이 C_2의 클래스에서 파생(Derived) 되었거나 C_1이 C_2의 클래스의 포함 객체이거나, C_1과 C_2가 메시지 호출이나 자료요소의 접근에 의해서 결정된다. 이러한 모든 경우는 C_1이 C_2에 의존적이며 C_2에 수정이 발생하였을 때 C_1은 영향을 받는다[9, 10]. 또한 클래스 간의 의존은 클래스들 간의 관계의 강도에 의하여 결정된다. 한 클래스 안에 의존관계가 있는 것 같이 컴포넌트들 간에도 의존성이 존재한다. 즉, 한 컴포넌트 내의 클래스와 다른 컴포넌트 내의 클래스들 간에 의존성이 존재한다면 두 컴포넌트는 의존성이 존재한다. 보통 컴포넌트 간의 의존성은 컴포넌트 내부의 클래스 간에 의존성이 아니라 컴포넌트의 인터페이스를 통해서 의존성을 나타낸다. 컴포넌트 내부에는 기능적으로 연관성이 많거나 상호 호출이 많은 클래스들을 하나로 묶으며 외부 컴포넌트 내의 호출이 없이 단지 인터페이스만을 통해

상호 호출할 수 있다. 컴포넌트 간의 의존은 하나의 컴포넌트 내의 클래스가 변경되면 다른 컴포넌트 내의 클래스에 변경을 야기할 수 있다. 만약 포함관계나 상속관계의 클래스들이 각각 A 컴포넌트와 B 컴포넌트에 존재한다면 A 컴포넌트 내의 클래스의 수정은 반드시 B 컴포넌트 내의 상속관계와 포함관계에 있는 클래스에 강한 영향을 미친다. 클래스 간의 관계는 포함관계나 상속관계만 존재하는 것이 아니라 연관관계 또한 존재한다. 클래스 간의 연관관계가 존재하는 클래스들이 각각의 컴포넌트로 포함된다면 컴포넌트 간은 연관관계가 존재한다고 정의된다. 만약 $<C_2, C_1> \in R$ 관계 시 클래스와 클래스의 의존 관계는 C2 객체가 C1 객체를 생성, 삭제, 수정, 조회하는 메소드 호출 유형에 따라서 C1 객체는 C2 객체에 의해서 수정 영향의 파급효과와 의존의 강도가 달라진다. 따라서 컴포넌트 C_i의 인터페이스의 기능을 실행하기 위하여 컴포넌트 C_j의 클래스를 참조하는 인터페이스 메소드를 호출할 경우 호출 에지가 존재하고 컴포넌트 C_i와 컴포넌트 C_j가 결합되었다고 정의할 수 있다. 이때 컴포넌트 C_i가 참조하는 컴포넌트 C_j의 클래스의 유형은 생성, 삭제, 수정, 그리고 참조가 된다. 컴포넌트 C_i 와 컴포넌트 C_j 간의 메소드 호출 유형이 생성과 삭제라면 가장 강한 의존 관계를 가졌다고 정의할 수 있다. 즉, 기능을 실행하기 위해서 데이터가 존재하지 않는다면 어떠한 기능도 실현되어질 수 없으므로 데이터를 생성하거나 삭제하는 메소드 호출은 가장 강한 종속관계를 유도한다. 컴포넌트 C_i와 컴포넌트 C_j 간의 메소드 호출 유형이 수정관계라면 데이터가 존재해서 기능을 실행할 수 있고 존재하는 데이터를 수정하는 관계이므로 생성, 삭제 보다는 약한 종속관계를 가지고 있다고 정의할 수 있다. 컴포넌트 C_i와 컴포넌트 C_j 간의 메소드 호출 유형이 참조관계라면 데이터를 수정하는 관계가 아니라 데이터를 참조만하는 관계이므로 가장 약한 관계가 존재한다고 정의할 수 있다. 따라서 컴포넌트 간의 결합의 정도는 메시지 호출 수에 의해서도 증가하지만 메시지 호출 유형에 의해서도 컴포넌트 간에 결합의 정도가 달라질 수 있으므로 본 논문에서는 메시지 호출 수, 메시지 호출 유형 및 클래스 간의 정적관계를 고려하여 결합도 메트릭스를 정의한다.

3.2 컴포넌트 평가를 위한 결합도 메트릭

본 절은 3.1절에서 제시한 컴포넌트의 특성을 적용하여 컴포넌트 결합도 메트릭을 정의 한다.

[정의 1] 클래스 간의 정적 결합도(a Static Coupling between Classes)

클래스와 클래스의 정적관계는 연관관계, 상속관계 그리고 포함관계가 존재하고 각각의 관계에 따라서 클래스간의 의존의 강도가 다르다. 따라서 각각의 클래스 간의 정적 관계의 강도에 따른 가중치를 부여하여 클래스 간의 정적 결합도 SCC를 다음과 같이 정의한다.

$$SCC(C_i, C_j) = \begin{cases} W_c, W_i, W_a\ (W_c > W_i > W_a),\ (C_i \neq C_j) \\ 0, \quad otherwise \end{cases}$$

[정의 1]에서 W_c, W_i, W_a는 각각 포함관계, 상속관계, 연관관계에서의 가중치를 의미하고 C_i와 C_j는 각각의 서로 다른 클래스를 의미한다.

[정의 2] 클래스 간의 동적 결합도(a Dynamic Coupling between Classes)

클래스와 클래스 간의 동적 관계는 메소드 호출에 의하여 이루어지는데 클래스 간의 의존의 강도는 메소드 호출 수와 메소드 호출 유형에 따라 다르다. 따라서 각각의 클래스 간의 메소드 호출 유형에 따른 가중치를 부여하고 메소드 호출 수를 고려하여 클래스 간의 동적 결합도 DCC를 다음과 같이 정의한다.

$$DCC(C_i, C_j) = WM_c \sum_{m_i \in M(C_i), m_j \in M(C_j)} C(m_i, m_j) + WM_d \sum_{m_i \in M(C_i), m_j \in M(C_j)} D(m_i, m_j)$$
$$+ WM_w \sum_{m_i \in M(C_i), m_j \in M(C_j)} W(m_i, m_j) + WM_r \sum_{m_i \in M(C_i), m_j \in M(C_j)} R(m_i, m_j)$$

[정의 2]에서 $m_i \in M(C_i), m_j \in M(C_j)$ and $(C_i \neq C_j)$에 대하여 클래스 C_i와 C_j가 포함하고 있는 메소드의 집합을 $M(C_i), M(C_j)$라 하고 각각 클래스가 포함하고 있는 메소드를 m_i, m_j 라 정의한다.

또한 $C(m_i, m_j), D(m_i, m_j), W(m_i, m_j), R(m_i, m_j)$는 클래스 C_i, C_j 대한 메소드 $m_i \in M(C_i), m_j \in M(C_j)$에 대해서 클래스 간의 메소드 호출에 대한 유형을 정의한 것으로 각각은 생성, 삭제, 수정, 참조의 메소드 호출 유형을 의미한다.

또한 $WM_c, WM_d, WM_w, WM_r (WM_c, WM_d > WM_w > WM_r$ and $C_i \neq C_j)$은 클래스 간의 메소드 호출 유형에 따른 가중치를 의미하는 것으로 각각은 생성, 삭제, 수정, 참조에 대한 가중치를 의미한다.

따라서 클래스 간의 동적 결합도는 클래스 간의 메소드 호출 유형에 따른 메소드 호출 수에다 메소드 호출 유형에 따른 가중치를 곱한 결과를 각각 합한 값을 의미한다.

[정의 3] 컴포넌트의 결합도(a Coupling Of Component)

각 컴포넌트가 포함하고 있는 클래스에 대하여 클래스 간

의 정적 결합도와 클래스 간의 동적 결합도를 고려하여 컴포넌트 BC_k의 결합도 $CPC(BC_k)$를 다음과 같이 정의한다.

$$COC(BC_k) = \sum_{C_i \in BC_k, C_j \notin BC_k} SDC(C_i, C_j) \times \sum_{C_i \in BC_k, C_j \notin BC_k} DDC(C_i, C_j)$$

[정의 4] 컴포넌트 간의 결합도(a Coupling Between Component)

두 개의 컴포넌트 BC_l, BC_m에 대한 컴포넌트 간의 결합도 $CBC(BC_l, BC_m)$은 각각의 컴포넌트가 포함하고 있는 클래스들 $C_l \in BC_l, C_m \in BC_m$에 대한 정적 결합도와 동적 결합도에 의하여 정의되어 진다. 따라서 컴포넌트 간의 결합도는 다음과 같이 정의한다.

$$CBC(BC_l, BC_m) = \sum_{C_i \in BC_l, C_j \in BC_m} SDC(C_l, C_m) \times \sum_{C_i \in BC_l, C_j \in BC_m} DDC(C_l, C_m)$$

[정의 5] 시스템의 평균 결합도(a Average Coupling of System by Components)

하나의 시스템 S_p의 평균 결합도는 S_p를 구성하는 컴포넌트들 간의 결합도의 평균값을 의미한다. 시스템의 평균 결합도 $ACSC(S_p)$는 다음과 같다.

$$ACSC(S_P) = \frac{\sum_{\alpha=1}^{r} CBC_\alpha(BC_g, BC_y)}{r}$$

[정의 5]에서 r 은 조합 가능한 모든 컴포넌트 쌍의 수를 의미한다.

3.3 결합도 속성에 대한 이론적 평가

Briand[8]은 특정 소프트웨어 산출물에만 일반적이며, 정확한 수학적 개념에 근거하여 엄격한 수학적 프레임워크를 제안했다. 이 프레임워크는 메트릭에 대해 크기, 길이, 복잡도, 응집도, 결합도등의 분야에 대한 개념과 성질을 정의하였다. 이 프레임워크는 새로운 소프트웨어 메트릭을 정의할 때 만족해야할 기준으로 사용가능하다. 따라서 본 논문에서는 Briand이 정의한 프레임워크를 컴포넌트 결합도 메트릭에 적용하여 이론적 타당성을 검증한다.

결합도 속성에 대한 이론적 검증은 다음과 같다.

성질 1. Non-Negativity

(결합도의 측정값은 음수가 될 수 없다는 것을 의미한다.)

(증명)

컴포넌트 간에 메시지 호출관계가 없다면 컴포넌트 BC_k에 대한 결합도인 $CPC(BC_k)=0$이고 메시지 호출관계가 하나라도 존재한다면 $C_i \in BC_k$, $C_j \notin BC_k$인 클래스 C_i, C_j에 대하여 $SDC(C_i, C_j) \neq 0$이고 $DDC(C_i, C_j) \neq 0$이기 때문에 $CPC(BC_k) > 0$이다. 따라서 컴포넌트의 결합도 $CPC(BC_k) \geq 0$을 만족한다.

성질 2. Null Value

(컴포넌트들 사이에 상호작용이 없다면 0 이라는 것을 의미한다.)

(증명)

컴포넌트 간에 호출관계가 존재하지 않는다면 컴포넌트 BC_k에 대한 $C_i \in BC_k$, $C_j \notin BC_k$인 클래스 C_i, C_j에 대하여 $SDC(C_i, C_j) = 0$이고 $DDC(C_i, C_j) = 0$ 이므로 컴포넌트의 결합도 $CPC(BC_k) = 0$이다.

성질 3. Monotononicity

(컴포넌트 내부 구성요소에는 변화 없이, 컴포넌트들 사이의 상호작용만 증가 한다면, 결합도는 감소하지 않는다는 것을 의미한다.)

(증명)

컴포넌트 내부의 구성요소에는 변화 없이 컴포넌트 간에 상호작용이 증가한다는 것은 컴포넌트 간에 메시지 호출이 증가한다는 것으로 컴포넌트 BC_k에 대한 $C_i \in BC_k$, $C_j \notin BC_k$인 클래스 C_i, C_j에 대하여 컴포넌트의 동적 결합도 $DCC(C_i, C_j)$는 메시지 호출이 증가한 만큼 증가한다는 것이다. 따라서 컴포넌트의 결합도 $CPC(BC_k)$는 감소하지 않는다는 것을 의미한다.

성질 4. Merging of Components

(두 컴포넌트를 결합하여 새로운 컴포넌트를 만든다면 새로운 컴포넌트의 결합도는 두 컴포넌트의 결합도의 합보다 증가하지 않는다는 것을 의미한다.)

(증명)

두 개의 컴포넌트를 병합한 후의 결합도 $CPC(BC_k)$는 각각의 컴포넌트의 결합도의 합에서 두 개의 컴포넌트 간에 메시지 호출에 의한 상호관계의 값인 동적 결합도 $DCC(C_i, C_j)$와 클래스들 간의 연관관계에 의해서 유도된 정적 결합도 $SDC(C_i, C_j)$를 뺀 값이 되므로 각각의 컴포넌트의 결합도의 합보다는 병합된 컴포넌트의 결합도가 크지 않다는 것을 의미한다.

성질 5. Disjoint Component Aditivity

(컴포넌트 간 상호작용이 전혀 없는 두 컴포넌트를 결합하여 새로운 컴포넌트를 만든다면, 결합도는 원래 두 컴포

넌트의 결합도의 합과 같다는 것을 의미한다.)

(증명)

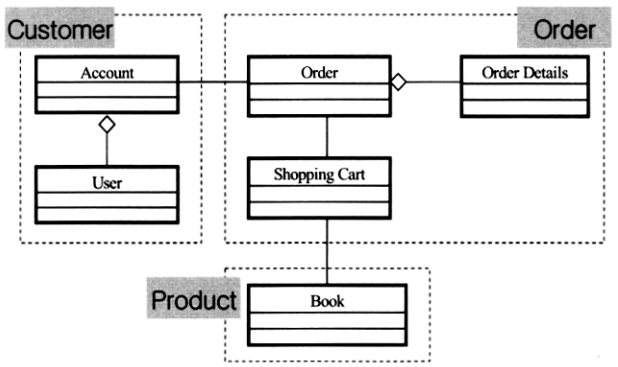
두 개의 컴포넌트 간에 상호작용이 전혀 없다면 $C_i \in BC_l, C_j \in BC_m$에 대하여 $DDC(C_i, C_j) = 0$이 되므로 두 개의 컴포넌트를 병합한 후의 결합도는 각각의 컴포넌트의 결합도의 합과 같다는 것을 의미한다. 두 컴포넌트 사이에 상호작용이 전혀 없기 때문에 두 컴포넌트를 하나로 결합할 경우 컴포넌트 사이의 상호작용은 그대로 유지되고 변화가 없으므로 기존 컴포넌트의 결합도의 합과 동일하다.

4. 사례 연구와 비교 평가

본 절에서는 본 논문에서 전자상거래 도메인을 선택하여 결합도 메트릭 적용 사례를 제시하고 기존 메트릭스와의 비교 평가를 통해서 그 효과성을 검증한다.

4.1 사례 연구

본 절에서는 온라인 북 주문 시스템[11]을 가지고 본 논문에서 제안한 결합도를 측정한다. 다음 (그림 1)은 온라인 북 주문 시스템의 클래스 다이어그램과 컴포넌트 구성도이다.



(그림 1) 클래스 다이어그램과 컴포넌트의 구성

다음 <표 1>는 온라인 북 주문 시스템의 기능 중 여러 개의 클래스들의 상호작용에 의해서 실현되는 유스케이스를 제시하였다. 또한 다음 <표 2>는 온라인 북 주문 시스템을 위한 ID 테이블이다.

〈표 1〉 온라인 북 주문 시스템의 유스케이스

ID	UseCase
U1	Order Book
U2	Cancel Order
U3	Update Order
U4	Search Order
U5	Update Shopping Cart
U6	Add Book to Shopping Cart
U7	Create Account
U8	Delete Account

〈표 2〉 클래스 이름과 ID 목록

ID	Class Name
A	User
B	Account
C	Order
D	Order Details
E	Shopping Cart
F	Book

다음 <표 3>는 온라인 북 주문 시스템의 기능 실행을 위하여 <표 1>에서 제시한 각 유스케이스별로 시퀀스 다이어그램을 분석하여 클래스 간에 메소드 호출 수와 메소드 호출 유형(생성, 수정, 삭제, 조회)을 분석하여 제시하였다. 메시지 호출 방향은 행 클래스에서 열 클래스이고 셀 내부는 메시지 호출 유형(개수)를 제시하였다. 예를 들면 표 2의 4행 2열에서 메시지 호출 방향은 C->A이고 메시지 호출 유형(Write)과 개수(3)는 W(3)이다.

〈표 3〉 클래스 간의 메소드 호출 수와 메소드 호출 유형

Class \ Class	A	B	C	D	E	F
A						W(2)
B	C(1),D(1)					
C	W(3)	R(5)		C(1),W(1) D(1),R(1)	R(2),D(2)	
D						
E						R(2)
F						

다음 <표 4>는 클래스들의 관계에 의한 정의 1에 의한 정적 결합도 $SCC(C_i, C_j)$를 제시한다. 정적 결합도의 가중치는 $W_c > W_i > W_a$ 이므로 실험적 근거에 의하여 5 > 3> 1 을 부여하였다.

〈표 4〉 클래스 간의 정적 관계에 의한 가중치 적용

Class \ Class	A	B	C	D	E	F
A		5				
B			1			
C				5	1	
D						
E						1
F						

다음 <표 5>는 정의 2에 의하여 클래스들의 메소드 호출 유형과 호출 수에 의한 동적 결합도를 제시한다. 동적 결합도 $DCC(C_i, C_j)$의 가중치는 메소드 호출 유형에 따라서 $WM_c, WM_d, WM_w, WM_r (WM_c, WM_d > WM_w > WM_r$ and $C_i \neq C_j)$이므로 실험적 근거에 의하여 6 > 2 > 1을 부여하였다.

〈표 5〉 클래스 간의 메소드 호출 수와 메소드 호출 유형에 의한 가중치 적용

Class \ Class	A	B	C	D	E	F
A		6*1+6*1=12	2*3=6			2*2=4
B			1*5=5			
C				6*1+2*1+6*1+1*1=15	1*2+6*2=14	
D						
E						2*2=4
F						

(그림 1)에 의하여 온라인 북 주문시스템의 컴포넌트는 Customer, Order, Product이다. 따라서 온라인 북 주문시스템의 컴포넌트에 대한 결합도를 [정의 3]에 의하여 계산한다. 따라서 온라인 북 주문시스템 컴포넌트의 결합도 $COC(BC_k)$는 다음과 같다.

$$COC(Customer) = 1 \times 5 = 5$$
$$COC(Order) = 1 \times 5 + 1 \times 4 = 9$$
$$COC(Product) = 1 \times 4 = 4$$

다음은 [정의 4]에 의하여 온라인 북 주문 시스템의 컴포넌트 간의 결합도 $CBC(BC_l, BC_m)$는 다음과 같다.

$$CBC(Customer, Order) = 5$$
$$CBC(Order, Product) = 4$$

다음은 [정의 5]에 의하여 온라인 북 주문 시스템의 평균 결합도 $ACSC(S_p)$는 다음과 같다.

$$ACSC(S_p) = (5+4)/3 = 3$$

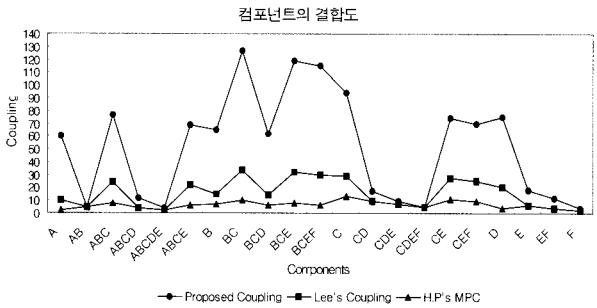
4.2 비교 평가

본 절에서는 본 논문에서 제안한 컴포넌트 설계를 위한 결합도 메트릭의 유용성을 증명하기 위해서 기존의 결합도 메트릭스를 적용하여 그 결과를 비교 평가해 본다. 비교 대상인 기존의 메트릭스로는 Henderson-Sellers가 제안한 MPC (Message Passing Coupling)[4], Lee[5]의 결합도 메트릭을 사용한다.

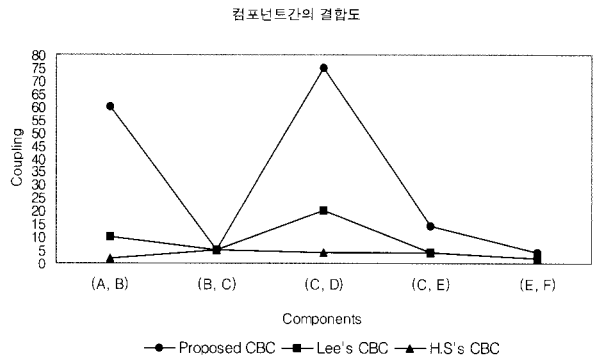
다음 (그림 2)는 각 클래스를 가능한 모든 경우의 컴포넌트로 조합한 후 각 후보 컴포넌트의 결합도 메트릭 $COC(BC_k)$를 측정한 결과이다.

(그림 2)에서는 본 논문에서 제안한 결합도와 Lee의 결합도가 비슷한 곡선이 그려지는 것을 확인할 수 있다. 이는 Henderson-Sellers가 제안한 MPC(Message Passing Coupling)[4]와 다르게 정적 결합도인 클래스 간의 관계적 특성에 의하여 가중치를 적용하였기 때문이다.

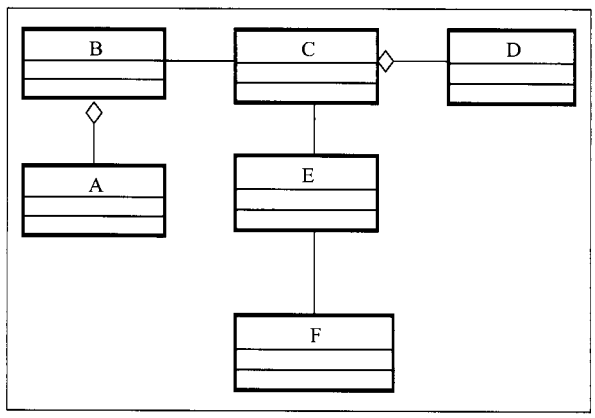
다음 <표 6>은 단일 클래스를 후보 컴포넌트라 정의하고 각 컴포넌트 간의 결합도 $CBC(BC_l, BC_m)$를 비교하였다.



(그림 2) 후보 컴포넌트들의 결합도



(그림 3) 컴포넌트 간의 결합도



(그림 4) ID를 적용한 클래스 다이어그램

<표 6>에서 제시한 컴포넌트 간의 결합도 결과를 보면 Henderson-Sellers의 MPC에 의한 컴포넌트 간의 결합도는

〈표 6〉 컴포넌트 간의 결합도 적용 결과

CBC(BC_i, BC_m)	Proposed CBC	Lee's CBC	H.S's CBC	# of method call
CBC(A, B)	60	10	2	2
CBC(B, C)	5	5	5	5
CBC(C, D)	75	20	4	4
CBC(C, E)	14	4	4	4
CBC(E, F)	4	2	2	2

〈표 7〉 후보 시스템의 평균 결합도를 적용한 결과

Proposed ACSC(Sp)		Lee's ACSC(Sp)		H.S.'s ACSC(Sp)	
Candidate System	ACSC(Sp)	Candidate System	ACSC(Sp)	Candidate System	ACSC(Sp)
AB, CDE, F)	**3**	**AB, CD, E, F**	**1.83**	**A, BCD, E, F**	**1.33**
AB, CD, E, F	3.83	ABCD, E, F	2	A, BCDE, F	1.33
ABCDE, F	4	ABCDE, F	2	A,BC, DE, F	1.33
AB, CDEF	6	AB,CDE, F	2.33	AB, CD, E, F	1.83
A, BCD, E, F	13	A, BCD, E, F	5.33	AB, CDE, F	2.33

메소드 호출 수와 똑 같음을 알 수 있다. 또한 본 논문에서 제안한 컴포넌트 간의 결합도와 Lee가 제안한 컴포넌트 간의 결합도는 컴포넌트 간에 포함관계와 상속관계가 존재하는 경우는 결합도가 높게 나오는 반면 컴포넌트 간의 관계가 연관관계일 경우, Lee가 제안한 컴포넌트 간의 결합도의 결과는 메소드 호출 수와 똑같이 나옴을 알 수 있다. 이는 클래스 간의 관계가 연관관계일 경우 메소드 호출 유형의 중요도에 따라서 컴포넌트 간의 결합의 강도가 달라지는데 그러한 면은 전혀 고려하지 않았기 때문이다.

다음은 결합도가 낮은 컴포넌트들을 조합하여 가능한 후보 시스템의 평균 결합도 $ACSC(S_p)$를 측정하였다.

<표 7>에 의하여 Henderson-Sellers의 MPC와 Lee에 의한 결과를 보면 후보 시스템들 중에 평균 결합도가 제일 낮은 후보 시스템에 포함된 컴포넌트들은 전혀 우리의 직관과 일치하지 않는다. 본 논문에서 제안한 결과를 보면 평균 결합도가 제일 낮은 후보 시스템에 포함된 컴포넌트들은 (AB), (CDE), (F)로 **(user, account), (Order, Order Details, Shopping Cart), (Book)**이다. 이 결과는 우리의 직관과 일치한다. 따라서 본 논문에서 제안한 컴포넌트의 결합도 메트릭은 컴포넌트의 특성을 적용하여 컴포넌트 간의 정적관계와 동적관계를 다 고려하였고 동적관계 시 메소드의 중요도에 따라 가중치를 부여하였기 때문에 설계된 컴포넌트들을 좀 더 정확하게 측정한다는 것을 기존의 결합도 메트릭스와의 비교 분석을 통하여 확인할 수 있었다.

5. 결론 및 향후 연구과제

본 논문에서는 객체지향 기반의 클래스의 특성과 다른 컴포넌트의 특성을 찾아내고 그 특성을 적용한 결합도 메트릭을 제안하였고 제안된 결합도 메트릭의 정확성을 검증하기 위해 사례연구와 비교 분석 결과를 제시하였다. 기존의 결합도 메트릭스와의 비교 분석을 통하여 본 논문에서 제안한 결합도 메트릭이 설계된 컴포넌트를 좀 더 정확하게 정량적으로 측정함을 확인하였고 Briand이 제시한 결합도 메트릭의 필요조건을 만족함을 증명을 통하여 제시하였다. 본 논문에서 제안한 결합도 메트릭은 컴포넌트 분석단계나 설계단계에서 식별된 컴포넌트를 정량적으로 평가함으로써 좀 더 독립적인 컴포넌트를 설계하는데 개발 시간과 노력을 절감할 수 있다. 향후 연구과제로는 컴포넌트의 특성을 세분화하여 컴포넌트의 평가를 위한 메트릭스에 적용하는 것이다.

참 고 문 헌

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

[2] H. Sahraoui, R. Godin, and T. Miceli, "Can Metrics Help to Bridge the Gap Between the improvement of OO Design Quality and its Automation?", In Proceedings of International Conf. on Software Maintenance, pp.154-162, 2000.

[3] S.R. Chidamber and C.F. Kemerer, "A Metric Suite for Object-Oriented Design", IEEE Transactions on Software Engineering, Vol.17, No.6, pp.636-638, 1994.

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

[5] Jong Kook Lee, Seung Jae Jung and Soo Dong Kim, "Component Identification Method with Coupling and Cohesion", Proceedings of Asia Pacific Software Engineering Conference, pp.79~88, 2001.

[6] Byung-Sun Ko, Jai-Nyun Park, "Improvement of Component Design using Component Metrics", Journal of KISS: Software and Applications, pp.980-990, 2004.

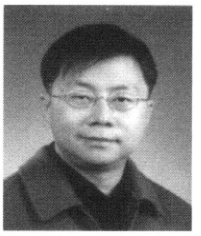
[7] Hyung Ho Kim and Doo Whan Bae, "Component Identification via Concept Analysis", Journal of Object

Oriented Programming, 2001.
[8] L.C. Briand, S. Morasca, and V.R. Basili, "Property-based software engineering measurement", IEEE Trans. Software Eng., Vol.22, No.1, pp.68-86.
[9] David C. Kung, Jerry. Gao, Pei Hsia, F. Wem, Y. Toyoshima and C. Chen, "Change Impact Identification in Object Oriented Software Maintenance", Proceedings International Technical Conference on Ciecuit/Systems, Computers and Communications, 1999.
[10] David C. Kung, Jerry Gao and Pei Hsia, "Class Firewall, Test Order, and Regression Testing of Object-Oriented Programs", Journal of Object-Oriented Programming, pp. 51-65, 1995.
[11] Doug Rosenberg, Kendall Scott, "Applying Use Case Driven Object Modeling with UML", Addison-Wesley, 2001.

최 미 숙

e-mail : khc67_kr@hanmail.net
1990년 전북대학교(이학사)
1994년 숙명여자대학교 일반대학원 컴퓨터과학과(이학석사)
2002년 숙명여자대학교 일반대학원 컴퓨터과학과(이학박사)
1995년~1999년 나주대학 소프트웨어 개발과 전임교수
2004년~현재 우석대학교 컴퓨터공학과 초빙교수
관심분야 : 소프트웨어 개발 방법론, CBD 방법론, 소프트웨어 아키텍쳐, 소프트웨어 메트릭

이 종 석



e-mail : jong1007@nate.com
1988년 서울대학교 계산통계학과(이학사)
1990년 서울대학교 대학원 계산통계학과(이학석사)
2001년 서울대학교 대학원 전기컴퓨터공학부(공학박사)
1993년~현재 우석대학교 컴퓨터공학과 부교수
관심분야 : 객체지향 소프트웨어 공학, 컴포넌트기반 개발, 소프트웨어 메트릭

송 행 숙

e-mail : songhs@mm.hanil.ac.kr
1981년~1985년 우석대학교 수학과(이학사)
1986년~1988년 전북대학교 대학원 컴퓨터과학과(이학석사)
1989년~1995년 아주대학교 대학원 컴퓨터과학과(공학박사)
1988년 전주대학교 전자계산소 조교
1991년~1994년 전북대학교, 아주대학교 컴퓨터과학과 강사
1995년~1997년 시멘틱스 소프트웨어 하우스연구소 소장
1997년 한일장신대학교 컴퓨터정보통신학부 교수
2005년~현재 한일장신대학교 건강생명정보학부 학부장
관심분야 : 컴퓨터 그래픽스, 컴퓨터 게임, 객체지향 소프트웨어 개발