효율적인 시스템 설계를 위한 클래스 간의 결합 척도☆

A Coupling Metric between Classes for Efficient System Design

최 미 숙* 이 종 석** 이 서 정***
Misook Choi Jongsuk Lee Seojeong Lee

요 약

최근 서비스 지향 시스템이 시스템의 기능적 단위인 서비스들을 재사용함으로 해서 개발 시간과 노력을 줄이는 특성 때문에 더욱 중요하게 부각되고 있다. 서비스의 재사용은 서비스들 간의 느슨한 결합에 의하여 효과적으로 이루어질 수 있고 서비스의 느슨한 결합은 컴포넌트 기반 시스템에 의존된다. 즉, 컴포넌트 기반 시스템은 클래스 간의 상호 의존이 강한 클래스들을 그룹화하여 도출하고 서비스 지향 시스템의 서비스는 컴포넌트 기반 시스템에 의존되어 설계된다. 따라서 효율적인 컴포넌트 기반 시스템 및 서비스 지향 시스템 설계를 위해서는 클래스 간의 결합도를 좀 더 정확히 측정할 수 있는 메트릭이 요구된다.

본 논문에서, 우리는 클래스 간의 구조적 특성, 동적 특성 그리고 0-1 사이로 정규화된 값을 적용한 클래스 간의 결합 척도를 제안한다. 우리는 Briand이 제안한 공리에 의해서 제안된 메트릭이 이론적으로 타당함을 증명하고 사례 연구를 통하여 정확성과 실용성을 제시한다. 우리는 기존 메트릭스와의 비교 분석을 통해서 제안된 메트릭의 평가 결과를 제시한다.

Abstract

Recently, service-oriented systems have been issued by their properties of reducing software development time and effort by reusing functional service units. The reusability of services can effectively promote through loose coupling between services and loose coupling between services depends on component-based system. That is, the component-based system is designed by grouping the tightly coupled classes of the object-oriented system and the service-oriented system is designed by the component-based system. Therefore, to design the component-based system and service-oriented system efficiently, a metric to measure the coupling between classes accurately needs.

In this paper, we propose a coupling metric between classes applying a structural property, a dynamic property, and the normalized value by 0-1. We prove the theoretical soundness of the proposed metric by the axioms of briand et al. and suggest the accuracy and practicality through a case study. We suggest the evaluation results of the proposed metric through a comparison with the conventional metrics.

☞ keyword : A Coupling Metric between Classes, Component-based System, Service-oriented System, Dependency Properties between classes, Normalization of Coupling

1. 서론

소프트웨어 개발은 객체 지향 개념과 컴포넌트 지향 개념을 거치면서 서비스 지향 개념으로 발전해 오고 있다. 서비스 지향 시스템은 현재 새로운 표준적 아키텍처 스타일이 되고 있고 소프트웨어 공학의 차세대 스텝으로서 자리잡아가고 있다[1]. 이러한 서비스 지향 시스템은 컴포넌트 기반 시스템에서 추상화 레벨이 한층 더 높아진 서비스들의 조합에 의해 구축되는 소프트웨어로 컴포넌트 기

* 정 회 원 : 우석대학교 기초및자연과학연구소 연구교수
khc67_kr@hanmail.net
** 정 회 원 : 우석대학교 컴퓨터교육과 교수
jong1007@nate.com
*** 정 회 원 : 한국해양대학교 컴퓨터공학부 조교수
sjlee@hhu.ac.kr
[2008/05/23투고 - 2008/05/27심사 - 2008/06/24심사완료]
☆ 이 논문은 2006년도 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-353-D00029).

반이다[2]. 컴포넌트 기반 시스템의 컴포넌트들은 상호 의존이 강한 클래스들을 그룹화 하여 보다 독립적인 컴포넌트를 식별한다[3]. 그리고 식별된 컴포넌트가 포함하고 있는 인터페이스들의 그룹화 된 단위는 서비스가 된다. 따라서 컴포넌트 기반 시스템이 잘 설계된다면 서비스 지향 아키텍처를 좀 더 효율적으로 설계할 수 있다. 즉, 컴포넌트 기반 시스템과 서비스 지향 시스템을 효율적으로 설계하기 위해서는 클래스들 간의 연결 강도를 정확히 평가할 수 있는 클래스들 간의 결합 척도가 필요하다.

기존의 클래스 기반의 결합 척도들은 평가 스케일[4]이 다른 특성으로 인하여 구조적 기반인 정적 메트릭과 메소드 호출 수 기반인 동적 메트릭을 따로 구분하여 측정하였다. 현재 객체 지향 설계 모델을 위한 클래스 다이어그램은 클래스 간의 관계인 상속 관계와 포함 관계가 대다수 50% 이상 포함되어 있다[5]. 따라서 하나의 시스템에 대하여 구조적 특성에 기반한 메트릭과 메소드 호출 수에 기반한 동적 메트릭을 각각 구분하여 측정한다면 구조적 특성과 동적 특성을 다 포함하고 있는 클래스 간의 관계에 대한 결합도를 정확하게 측정 및 평가할 수 없다. 즉, 구조적 특성과 동적 특성을 동시에 다 포함하여 측정할 수 있는 클래스 간의 결합 척도가 필요하다. 또한, 기존의 결합도는 0-1 사이로 정규화 되어 있지 않고 선형적으로 증가한다. 따라서 클래스 간의 결합도가 어느 범위에 있는지 평가하기가 용이하지 않다. 결합도가 0-1 사이로 정규화 된다면 클래스 간의 연결 강도의 정도를 효율적으로 평가할 수 있다. 따라서 본 논문에서는 이러한 문제점을 보완하여 좀 더 정확히 결합도를 측정하고 평가할 수 있는 메트릭을 제안한다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존 클래스의 결합 척도를 제시한다. 3 장에서는 본 논문에서 제안한 클래스 간의 결합 척도를 제시한다. 4 장에서는 3 장에서 제시한 결합 척도가 이론적으로 타당하다는 것을 Briand[6]이 정의한 공리에 적용하여 증명한다. 5 장에서는 온라인 도서 주문 시스템[7]을 이용하여 제안한 클래스 간의 결합 척도 적용 사례를 제시하고 기존 메트릭스와의 비교 분석을 통해서 그 결과의 정확성을 검증한다. 6장에서는 결론을 제시한다.

2. 관련 연구

Chidamber[8]는 결합 척도로 CBO를 제안하였는데 CBO는 결합되어 있는 다른 클래스의 수로 정의된다. 하나의 클래스와 관련이 있는 클래스들의 수를 계산함으로써 클래스의 결합도를 간단하게 측정할 수 있지만, 한 클래스 내에서 같은 클래스에 있는 멤버들을 여러 번 사용할 때도 한 번 사용한 것과 같은 값을 가진다. 따라서 올바르게 결합도를 측정할 수 없다.

Li[9]는 결합 척도를 세가지로 나누어서 정의하였다. 결합 척도의 첫번째는 상속성을 통한 결합도이다. 상속성을 측정하기 위해서 DIT와 NOC를 사용하였다. DIT는 그 클래스가 가지는 부모 클래스의 수이고 NOC는 그 클래스에 의해 직접적으로 영향을 받는 클래스의 수를 가리킨다. 두번째는 메시지 패싱을 통한 결합도(MPC: message-passing coupling)이다. MPC는 CBO를 발전시킨 방법으로서 클래스 내에서 정의된 send 문장의 수로 정의하였다. MPC는 클래스가 받은 메시지 수는 고려하지 않았다. 세번째는 추상 데이터형을 통한 결합도(DAC: data abstraction coupling)이다. DAC는 클래스 내에서 사용되는 추상 데이터형의 수로 정의하였다. 따라서 Li는 클래스 간의 구조적 관계인 상속성을 측정한 정적 메트릭과 메시지 호출을 통한 동적 메트릭을 구분하여 측정하였다. 상속성을 측정하기 위해서는 직접적으로 영향 받는 클래스의 개수로 측정하므로 CBO처럼 클래스에 있는 구성 요소들을 여러 번 사용할 경우에도 한 번 사용한 것과 같은 값을 가지므로 정확하게 측정할

수 없다. 또한 MPC는 클래스 내에 정의된 send 문장의 수로 측정하므로 메소드 타입을 고려하지 않고 0-1 사이로 정규화 되지 않으므로 의존 관계에 있는 클래스들 각각에 대하여 얼마나 강하게 의존되어 있는가를 평가할 수 없다.

Henderson[10]의 결합 척도는 Li가 제시한 상속에 의한 결합도는 생각하지 않고 DAC, MPC, RFC로 나누어 제시하였다. 그 중에 RFC만 Li가 제안한 방법과 다르다고 할 수 있다. RFC는 클래스에 있는 모든 메소드의 수에 각 클래스가 호출한 메소드의 수를 합한 것으로 결국 그 클래스에서 사용 가능한 메소드의 수이다. RFC 역시 메소드 호출 수 만을 고려하고 클래스 간의 메소드 호출에 대한 메소드 타입을 고려하지 않고 0-1 사이로 정규화 되지 않으므로 의존 관계에 있는 클래스들 각각에 대하여 얼마나 강하게 의존되어 있는가를 평가할 수 없다. 또한 상속관계의 클래스들 간의 결합도는 고려하지 않는다.

Yacoup et. al.[11]은 객체 기반의 양 방향 메소드 호출 수를 측정하는 동적 결합 척도인 EOC(Export Object Coupling), IOC(Import Object Coupling), OQFS(Object Request for Service), OPFS(Object Response for Service)를 제안했다. Yacoup et. al.[11]이 제안한 동적 결합척도 역시 메소드 호출 수 만을 고려하고 클래스 간의 메소드 호출에 대한 메소드 타입을 고려하지 않는다. 따라서 정확하게 동적 결합도를 측정할 수 없다.

[12]는 클래스 간의 상대적인 결합 척도(RCBC: Relative Coupling Between Classes)를 제안했다. RCBC는 의존 클래스가 호출하는 피 의존 클래스의 메소드들의 집합을 의존 클래스에서 호출하는 다른 모든 클래스에 속한 메소드들의 크기의 합으로 나누어서 클래스 간의 의존 강도를 측정한다. 즉, RCBC(c1, c2)는 클래스 c1이 시스템의 변화에 의해 재검사가 유발될 가능성 중 클래스 c2에 의해서 재 검사가 유발될 가능성을 나타내며 0에서 1사이의 값으로 클래스 간의 의존 강도를 측정한다. RCBC는 0-1 사이로 정규화 되어 클래스 간의 의존 강도의 정도를 제시할 수 있지만 클래스 간의 상속관계의 특성이나 메소드 타입을 고려하지 않고 있다.

3. 효율적인 시스템 설계를 위한 클래스 간의 결합 척도

3.1 클래스의 특성

클래스는 데이터 즉, 객체 없이는 그 기능을 실행할 수 없다. 따라서 클래스들 간의 데이터 생성 관계는 가장 중요한 관계이다. 즉, 클래스들 간의 데이터 생성에 의한 의존 관계의 파악은 클래스들 간의 연결 강도를 보다 정확히 측정해낼 수 있는 기반이 되는 것이다. 클래스의 데이터인 객체는 액터에 의해서 생성될 수가 있거나 다른 클래스에 의해서 생성될 수 있다. 만약 클래스의 데이터가 액터에 의해서 생성된다면 그 클래스는 독립적으로 존재할 수 있고 다른 클래스들은 그 클래스에게 데이터를 생성하는 메소드가 아닌 데이터를 참조하거나 수정(삭제포함)하는 메소드만을 호출하여 그 클래스를 사용한다. 그러나 다른 클래스에 의해서 자신의 클래스의 데이터가 생성된다면 자신의 클래스는 데이터를 생성해 주는 클래스 없이는 독립적으로 존재할 수 없으므로 데이터를 생성해 주는 클래스에 강하게 종속되고 자신의 데이터를 생성시킨 클래스가 삭제된다면 자신의 클래스 역시 삭제되므로 대부분 그러한 관계의 클래스들은 대부분 라이프 사이클이 같은 특성을 가지고 있다. 따라서 데이터 생성에 의한 클래스들 사이의 연결 강도는 가장 높다. 또한, 다른 클래스에 의해서 자신의 클래스의 데이터가 생성되었기 때문에 자신의 클래스를 접근하는 다른 클래스들은 데이터를 생성하는 메소드가 아닌 데이터를 참조하거나 수정(삭제포함)하는 메소드만을 호출하여 그 클래스를 사용한다. 따라서 클래스 간의 메소

드 호출 타입은 생성, 수정(삭제포함) 그리고 참조 메소드 타입으로 분류할 수 있으며 데이터 생성 메소드는 위에서 제시한 특성에 의하여 클래스 간에 가장 높은 연결 강도를 가지고 있다. 그 다음은 수정 메소드 타입이고 참조 메소드 타입은 연결 강도가 가장 약하다. 그 이유는 수정 메소드 호출은 존재하는 데이터의 값을 변경하거나 삭제하므로 그 데이터를 참조하여 기능을 수행하는 다른 클래스들은 영향을 받는다. 또한, 참조 메소드 호출은 존재하는 데이터의 값을 변경하는 것이 아니므로 그 데이터를 참조하여 기능을 수행하는 다른 클래스들은 전혀 영향을 받지 않기 때문이다.

클래스들 사이의 관계가 포함 관계일 경우, A 클래스가 B 클래스와 C 클래스를 포함하고 있다면 A 클래스가 B 클래스와 C 클래스의 데이터를 생성시키는 관계이고 A 클래스의 메소드 호출은 그대로 B 클래스와 C 클래스의 메소드 호출로 전파된다. 포함 관계의 클래스들은 라이프 사이클이 같다. 또한 클래스들 사이의 관계가 상속 관계일 경우, A 클래스가 부모 클래스이고 B 클래스가 자식 클래스라면 자식 클래스인 B 클래스의 데이터 생성은 동시에 부모 클래스인 A 클래스의 데이터를 생성 시킨다. 따라서 B 클래스에 의존된 A 클래스는 라이프 사이클이 같으며 B 클래스의 메소드 호출은 A 클래스의 메소드 호출로 전파될 수 있다. 클래스들 사이가 연관 관계라면 대부분 데이터 생성 관계가 아닌 데이터 참조나 데이터 수정에 의한 메소드 호출 관계이다. 그러나 클래스들 사이의 관계가 연관 관계라 하더라도 환경적 특성에 의하여 데이터를 생성 시키는 관계가 존재할 수 있으며 그들 간의 관계는 라이프 사이클이 같지 않을 수도 있다. 따라서 클래스들 사이에 메소드 호출에 의한 데이터 생성 관계 파악은 구조적 관계인 상속 관계, 포함 관계 그리고 연관 관계의 특성을 자동으로 부여한다. 즉, 클래스들 사이에 메소드 호출 타입을 분류하고 메소드 호출 타입에 따른 메소드 호출 수를 결합 척도에 부여한다면 클래스들 간의 관계인 구조적 관계의 특성과 메소드 호출에 의한 동적 관계의 특성을 다 포함하여 측정하므로 클래스 간의 연결 강도인 결합도를 정확히 측정할 수 있다. 또한 메소드 호출 타입에 의한 클래스 간의 연결 강도는 다르므로 메소드 호출 타입에 따라 가중치를 부여하여야 한다.

클래스들 사이에 상호 작용은 메소드 호출에 의해서 이루어진다. 클래스들 간의 상호 작용에 의한 연결 강도는 메소드 호출 타입에 따른 메소드 호출 수에 의존되어 선형적으로 증가되는 특징을 가지고 있다. 그러나 클래스 간의 연결 강도가 선형적 증가 특성을 가지고 있다고 하더라도 결합도가 0-1 사이로 정규화 된다면 클래스들 간의 결합의 정도를 효율적으로 평가할 수 있다.

3.2 클래스 간의 결합 척도

본 절은 3.1 절에서 제시한 특성을 적용하여 클래스 간의 결합 척도를 정의 한다.

[정의 1] 시스템을 구성하는 클래스들

시스템은 유한개의 클래스들로 구성되어 있으므로 시스템을 S라 하고 시스템에 포함된 클래스들을 $C_i(i=1...l)$라 하면, 시스템은 다음과 같이 정의한다.

$$S=\{C_1, C_2, ..., C_l\}$$

[정의 2] 클래스가 포함하는 메소드들

각 클래스가 포함하는 메소드들인 $M(C_j)$ $(j=1..k)$는 다음과 같이 정의한다.

$$M(C_j)=\{m_{j1}, m_{j2}, ..., m_{jn}\}$$

[정의 3] 클래스들 사이의 상호 작용

클래스들 간의 상호작용은 메소드 호출에 의해

서 이루어진다. 서로 다른 클래스들 $C_g(1 \le g \le m), C_g(1 \le g \le m), C_y(1 \le y \le m)$ 에 대해서 메소드 $m' \in M(C_g), m \in M(C_y)$ 이 존재한다고 할 때 메소드 m' 이 m 을 호출한다면 (m', m) 으로 정의한다. 또한, 서로 다른 클래스 $C_g(1 \le g \le m)$, $C_y(1 \le y \le m)$ $C_g(1 \le g \le m), C_y(1 \le y \le m)$ 에 대해서 클래스 C_g 와 C_y 간의 메소드 호출에 의한 상호 작용은 다음과 같이 정의한다.

$$Calling\ (C_g, C_y) = \{m \in M(C_y) \mid for\ some\ g \neq y \quad \exists m' \in M(C_g)\ s.t.\ (m', m)\}$$

$$Called\ (C_g, C_y) = \{m' \in M(C_g) \mid for\ some\ g \neq y \quad \exists m \in M(C_y)\ s.t.\ (m, m')\}$$

메소드 호출 유형인 생성(Create), 수정(Update), 참조(Read)에 의한 클래스 간의 상호 작용은 다음과 같이 정의한다.

a. 클래스 C_g 와 C_y 간의 메소드 호출이 수정일 경우

$$Calling(C_g, C_y) =: U(\overrightarrow{Cg, Cy}),$$

$$Called(C_g, C_y) =: U(\overleftarrow{Cg, Cy})$$

b. 클래스 C_g 와 C_y 간의 메소드 호출이 참조일 경우

$$Calling(C_g, C_y) =: R(\overrightarrow{Cg, Cy}),$$

$$Called(C_g, C_y) =: R(\overleftarrow{Cg, Cy})$$

c. 클래스 C_g 와 C_y 간의 메소드 호출이 생성일 경우

$$Calling(C_g, C_y) =: C(\overrightarrow{Cg, Cy}),\ Called(C_g, C_y) =: C(\overleftarrow{Cg, Cy})$$

[정의 4] 클래스들 사이에 메소드 호출 유형에 의한 가중치

메소드 호출 유형에 따라서 클래스 간의 연결 강도가 달라지므로 메소드 호출 유형에 따라서 가중치(W)를 부여한다. 따라서 클래스 간의 메소드 호출 유형에 따른 가중치를 다음과 같이 정의한다.

a. 클래스들 C_g 와 C_y 간의 메소드 호출이 수정일 경우의 가중치

$$W(U(\overrightarrow{Cg, Cy})) = W(U(\overleftarrow{Cg, Cy})) := W_u$$

b. 클래스들 C_g 와 C_y 간의 메소드 호출이 참조일 경우의 가중치

$$W(R(\overrightarrow{Cg, Cy})) = W(R(\overleftarrow{Cg, Cy})) := W_r$$

c. 클래스들 C_g 와 C_y 간의 메소드 호출이 생성일 경우의 가중치

$$W(C(\overrightarrow{Cg, Cy})) = W(C(\overleftarrow{Cg, Cy})) := W_c$$

클래스들 간의 메소드 호출 타입에 의한 종속의 강도는 생성 > 수정 > 참조 순이다. 따라서 클래스들 간의 메소드 호출 유형에 의한 가중치의 크기는 $W_c > W_u > W_r$ 순으로 부여한다.

[정의 5] 클래스들 간에 메소드 호출 타입에 의한 메소드 호출 수

서로 다른 클래스 $C_g(1 \le g \le m), C_y(1 \le y \le m)$ 간에 메소드 호출 유형에 대한 메소드 호출 수는 다음과 같이 정의한다.

a. 클래스들 C_g 와 C_y 간의 메소드 호출이 수정일 경우의 메소드 호출 수

$$|U(\overrightarrow{Cg, Cy})| + |U(\overleftarrow{Cg, Cy})| := nuc$$

b. 클래스들 C_g 와 C_y 간의 메소드 호출이 참조일 경우의 메소드 호출 수

$$|R(\overrightarrow{Cg, Cy})| + |R(\overleftarrow{Cg, Cy})| := nrc$$

c. 클래스들 C_g 와 C_y 간의 메소드 호출이 생성일 경우의 메소드 호출 수

$$|C(\overrightarrow{Cg, Cy})| + |C(\overleftarrow{Cg, Cy})| := ncc$$

[정의 6] 클래스들 간에 메소드 호출 타입에 의한 가능한 최대 메소드 호출 수

서로 다른 클래스 $C_g(1 \le g \le m), C_y(1 \le y \le m)$ 간에 메소드 호출 유형에 대한 가능한 최대 메소드 호출 수는 다음과 같이 정의한다.

a. 클래스들 C_g 와 C_y 간의 메소드 호출이 수정일 경우의 가능한 최대 메소드 호출 수

$$\max(|U(\overrightarrow{Cg,Cy})|+|U(\overleftarrow{Cg,Cy})|) := mnuc$$

b. 클래스들 C_g 와 C_y 간의 메소드 호출이 참조일 경우의 가능한 최대 메소드 호출 수

$$\max(|R(\overrightarrow{Cg,Cy})|+|R(\overleftarrow{Cg,Cy})|) := mnrc$$

c. 클래스들 C_g 와 C_y 간의 메소드 호출이 생성일 경우의 가능한 최대 메소드 호출 수

$$\max(|C(\overrightarrow{Cg,Cy})|+|C(\overleftarrow{Cg,Cy})|) := mncc$$

[정의 7] 클래스 간에 결합 척도

(A Coupling Metric between Classes)

본 논문에서 제안하는 클래스 간의 결합도는 클래스 간의 연결 강도를 나타낸다. 따라서 클래스 간의 결합 척도 CMC 는 클래스 $Cg(1 \le g \le m), Cg(1 \le g \le m), Cy(1 \le y \le m), g \ne y$ 에 대해서 다음과 같이 정의한다.

$$CMC\ (C_g, C_y) = W_u \times \frac{nuc}{mnuc} + W_r \times \frac{nrc}{mnrc} + W_c \times \frac{ncc}{mncc}$$

$$W_u \times \frac{nuc}{mnuc} + W_r \times \frac{nrc}{mnrc} + W_c \times \frac{ncc}{mncc}$$

따라서 본 논문에서 제안하는 CMC 는 생성(Create), 수정(Update), 참조(Read)등, 각각의 메소드 호출 타입에 따른 가중치에다 각각의 메소드 호출 타입에 따른 클래스들 간의 메소드 호출 수를 클래스들 간의 가능한 메소드 호출 수로 나눈 결과 값을 곱한 값의 합으로 클래스 간의 결합 척도를 계산한다. 가중치 W_c =0.6, W_u =0.3, W_r =0.1을 적용하고 CMC 의 값은 0-1 사이로 정규화 된다.

4. 제안된 메트릭의 이론적 검증

Briand[6]은 특정 소프트웨어 산출물에만 적용되지 않도록 일반적이며 정확한 수학적 개념에 근거하여 엄격한 수학적 공리를 제안했다. 이 공리는 새로운 소프트웨어 메트릭스를 정의할 때 만족해야 할 기준을 제공해 주고 있다. 본 논문에서는 클래스 간의 연결 강도인 결합도를 정확히 측정할 수 있는 메트릭을 정의하였다. 따라서 제안된 메트릭의 이론적 타당성을 검증하기 위하여 Briand[6]이 정의한 프레임워크에 적용하여 증명한다.

[결합도성질 1] Nonnegativity

(결합도의 측정 값은 음수가 될 수 없다는 것을 의미한다.)

증명)

만약 클래스들 간에 메시지 호출이 없다면 메소드 호출 타입에 따른 메소드 호출 수 $ncc = 0$, $nrc = 0$, $nuc = 0$ 이므로 클래스 간에 결합 척도 CMC=0 이 된다. 또한 클래스들 간에 메시지 호출이 있다면 메소드 호출 타입에 따른 메소드 호출 수 $ncc \ne 0$, $nrc \ne 0$, $nuc \ne 0$ 이므로 클래스 간에 결합 척도인 CMC $\succ 0$ 이다. 따라서, 본 논문에서 제안하는 클래스 간에 결합 척도 CMC 는 음수가 될 수 없으므로 $CMC \ge 0$ 인 결합도 성질 1을 만족한다.

[결합도성질 2] Null Values

(클래스들 사이에 상호작용이 없다면 0라는 것을 의미한다.)

증명)

만약 클래스들 간에 메시지 호출이 없다면 메소드 호출 타입에 따른 메소드 호출 수 $ncc = 0$, $nrc = 0$, $nuc = 0$ 이므로 클래스 간에 결합 척도 $CMC = 0$ 이 된다. 따라서, 본 논문에서 제안하는 클래스 간에 결합 척도 CMC 는 결합도 성질 2를 만족한다,

[결합도성질 3] Monotonicity

(클래스 내부 구성 요소에는 변화 없이 클래스들 사이의 상호작용만 증가한다면 결합도는 감소하지 않는다는 것을 의미한다.)

증명)

클래스 내부 구성 요소에는 변화 없이 클래스들 사이의 상호작용만 증가한다면 클래스 간에 메시지 호출 수가 증가하는 것이므로 $ncc \le ncc'$, $nrc \le nrc'$, $nuc \le nuc'$ 이 성립된다.

따라서, 다음과 같은 식이 성립되므로 클래스 간의 결합도가 그만큼 증가한다는 것이다.

$$CMC(C_g, C_y) =$$

$$W_u \times \frac{nuc}{mnuc} + W_r \times \frac{nrc}{mnrc} + W_c \times \frac{ncc}{mncc}$$

$$\le W_u \times \frac{nuc'}{mnuc} + W_r \times \frac{nrc'}{mnrc} + W_c \times \frac{ncc'}{mncc}$$

$$= CMC'(C_g, C_y)$$

그러므로 본 논문에서 제안하는 클래스 간에 결합 척도 CMC 는 결합도성질 3을 만족한다.

[결합도성질 4] Merging of Classes

(두 클래스를 결합하여 새로운 클래스를 만든다면 새로운 클래스의 결합도는 두 클래스 각각의 결합도의 합보다 증가하지 않는다는 것을 의미한다.)

증명)

두 클래스 $Cg(1 \le g \le m), Cy(1 \le y \le m), g \ne y$ 에 대하여, 두 클래스를 합한 새로운 클래스를 C 라고 하자. 그러면 다음과 같은 수식이 성립한다.

$$|\text{Coupling}(C)| = |\text{Coupling}(C_g)| + |\text{Coupling}(C_y)| - |\text{Coupling}(C_g \cap C_y)| \text{ à } ①$$

①의 수식에 대하여 각각은 다시 다음과 같이 정의할 수 있다.

$$|\text{Coupling}(C_g)| = \sum_{C_y \notin C_g} CMC(C_g, C_y)$$

$$|\text{Coupling}(C_y)| = \sum_{C_g \notin C_y} CMC(C_y, C_g) \text{ à } ②$$

$$|\text{Coupling}(C_g \cap C_y)| = CMC(C_g, C_y)$$

②의 수식을 ①의 식에 대입해 보면 다음과 같은 수식이 성립된다.

$$|\text{Coupling}(C)| = |\text{Coupling}(C_g)| + |\text{Coupling}(C_y)| - |\text{Coupling}(C_g \cap C_y)|$$

$$= \sum_{C_y \notin C_g} CMC(C_g, C_y) + \sum_{C_g \notin C_y} CMC(C_y, C_g) - CMC(C_g, C_y) \le \sum_{C_y \notin C_g} CMC(C_g, C_y) + \sum_{C_g \notin C_y} CMC(C_y, C_g)$$

$$= |\text{Coupling}(C_g)| + |\text{Coupling}(C_y)|$$

따라서 Coupling(C_g)+Coupling(C_y)>=Coupling(C) 이다.

그러므로 본 논문에서 제안하는 클래스 간에 결합 척도 CMC 는결합도성질 4를 만족한다.

[결합도성질 5] Disjoint Class Aditivity

(클래스 간 상호작용이 전혀 없는 두 클래스를 결합하여 새로운 클래스를 만든다면 결합도는 원래 두 클래스의 결합도의 합과 같다.)

증명)

두 클래스 $Cg(1 \le g \le m), Cy(1 \le y \le m), g \ne y$ 에 대하여, 두 클래스를 합한 새로운 클래스를 C 라고 하자. 다음 ①과 같은 수식이 성립된다.

$$|Coupling(C)| = |Coupling(^{C_g})|+|Coupling(^{C_y})| -|Coupling(^{C_g} \cap ^{C_y})|à \quad ①$$

두 클래스 $Cg(1 \le g \le m), Cy(1 \le y \le m), g \ne y$ 에 대하여 상호작용이 없다면 $CMC(C_g,C_y)=0$ 이다. 따라서 다음 ②와 같은 수식이 성립된다.

$$|Coupling(^{C_g} \cap ^{C_y})|=^{CMC(C_g,C_y)}=0이다. à \quad ②$$

수식 ①에 ②를 적용하면 다음과 같은 수식이 성립된다.

$$|Coupling(C)| = |Coupling(^{C_g})|+|Coupling(^{C_y})| -|Coupling(^{C_g} \cap ^{C_y})|$$

$$= |Coupling(^{C_g})|+|Coupling(^{C_y})| -0$$

$$= |Coupling(^{C_g})|+|Coupling(^{C_y})|$$

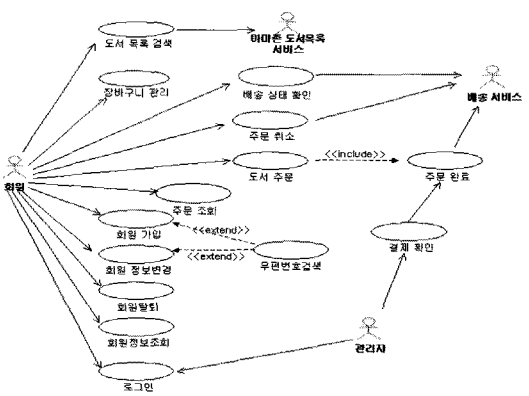
그러므로 본 논문에서 제안하는 클래스 간에 결합 척도 CMC 는결합도성질 5를 만족한다.

5. 사례연구 및 비교 평가

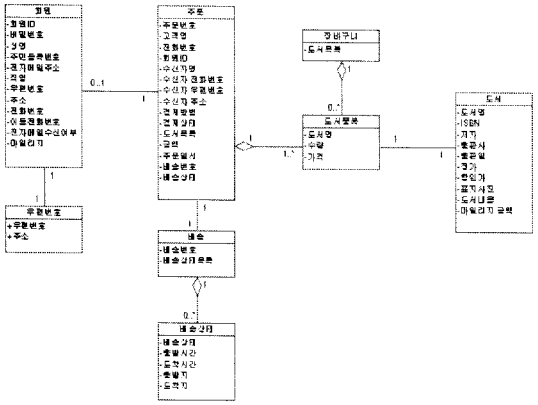
본 장에서는 온라인 도서 주문 시스템[7]을 이용하여 제안한 클래스 간의 결합 척도 적용 결과를 제시하고 기존 메트릭스와의 비교 평가를 통해서 그 결과의 정확성을 검증한다.

유스케이스의 기능을 실행하기 위하여 어떤 클래스가 필요하고 각각의 클래스는 어떠한 타입의 메소드를 몇번 호출하는지를 분석하기 위하여 유스케이스를 구체화한 시퀀스 다이어그램을 통하여 분석한다. 또한, 액터의 요청으로부터 그 결과를 출력하는 한 단위가 되는 기능적 서비스 단위를 이벤트라 한다. 하나의 유스케이스는 액터 관점에서 하나의 이벤트를 포함하거나 여러 개의 이벤트를 포함할 수 있다. 각각의 이벤트들은 이벤트에 의한 기능을 실현하기 위하여 클래스의 메소드를 호출한다. 따라서 본 논문에서 클래스 간에 최대 가능한 메소드 호출 수, 클래스 간의 메소드 호출 타입과 메소드 호출 수를 좀 더 정확하게 파악하기 위하여 이벤트 기반으로 시스템을 분석한다.

다음은 온라인 도서 주문 시스템의 유스케이스 다이어그램과 클래스 다이어그램을 그림 1과 그림 2에서 제시하고 각 유스케이스의 기능을 실현하기 위하여 분석한 클래스 간의 메소드 호출 타입과 메소드 호출 수를 다음 표 2에서 제시한다.



(그림 1) 유스케이스 다이어그램



(그림 2) 클래스 다이어그램

(표 1) 클래스 이름 정의표

클래스	회원	도서	도서 품목	장 바구니	도서 주문	배송	배송 상태	우편 번호
이름	A	B	C	D	E	F	G	H

위의 표 2는 이벤트 기반의 참여 클래스와 클래스의 메소드 호출 타입을 분석하여 정의하였다. 다음 표 3은 분석된 표 2를 기반으로 클래스 간의 메소드 호출 타입과 메소드 호출 수를 분석하여 정의 하였다. 표 3에서 A는 회원 클래스, H는 우편번호 클래스, E는 주문 클래스, F는 배송 클래스, G는 배송상태 클래스, C는 도서품목 클래스, D는 장바구니 클래스, B는 도서 클래스이다. 메시지 호출 방향은 행 클래스 -> 열 클래스이고 셀 내부는 메소드 호출 타입(개수)을 표시 하였다. 메소드 호출 유형인 R은 Read, U는 Update, C는 Create를 의미한다. 예를 들면 표 3의 2행 9열에서 메소드 호출 방향은 회원 클래스 -> 우편번호 클래스이고 메소드 호출 타입(개수)은 Read(1)을 의미한다. 우리는 클래스 간의 메소드 호출 수 그리고 메소드 호출 타입을 고려한다.

(표 2) 유스케이스, 이벤트 그리고 포함 클래스의 메소드 호출 타입

유스케이스	이벤트	A	B	C	D	E	F	G	H
회원조회	회원정보조회	R							
회원가입	기등록회원체크	R							
	중복아이디체크	R							
	회원정보등록	C							
	우편번호 검색								R
회원정보변경	회원정보조회	R							
	회원정보변경	U							
	우편번호 검색								R
회원탈퇴	회원정보조회	R							
	회원탈퇴	U							
로그인	로그인	R							
장바구니 관리	장바구니 담기		R	C	C				
	장바구니 조회			R	R				
	장바구니 변경			U	U				
	장바구니 비우기			U	U				
도서주문	회원정보조회	R							
	도서주문			R	R	C			
	주문완료			C		U	C,R	C,R	
주문조회	주문조회					R			
	주문상세조회					R			
주문취소	주문취소			U		U	U	U	
결제확인	미결제주문조회					R			
	주문완료			C		U	C,R	C,R	
배송상태확인	배송상태확인					R	R	R	
도서목록검색	도서목록조회		R						
	도서상세조회		R						
우편번호검색	우편번호검색								R

(표 3) 클래스들 사이의 메소드 호출 타입과 메소드 호출 수

Class	A	B	C	D	E	F	G	H
A								R(1)
B								
C		R(1)						
D		R(1)	C(1), U(2), R(1)					
E	R(1)		C(1), U(1), R(1)	R(1)		C(1), U(1), R(2)		
F							C(1), U(1), R(2)	
G								
H								

다음은 본 논문에서 제안한 클래스 간의 결합 척도는 표 3의 분석한 결과를 기반으로 계산한다. 클래스 간의 의존의 정도는 메소드 호출 타입에 따라 다르므로 각각 가중치를 다르게 부여하였다. 클래스 간에 메소드 호출 타입 중 가장 강한 의존 관계를 가지고 있는 생성(Create)은 0-1 사이의 값을 기준으로 했을 때 중간 보다 좀 더 높은 값을 부여하였고 수정(Update)은 중간 보다 낮은 값을 부여하였다. 또한, 조회(Read)는 가장 낮은 값을 부여하였다. 즉, 메소드 호출 타입에 따른 가중치를 조회(Read)=0.1, 수정(Update)=0.3, 그리고 생성(Create)=0.6으로 부여하였다. 또한, 제안한 클래스 간의 결합 척도인 CMC를 계산하기 위해서 메소드 호출 타입에 따른 최대 가능한 메소드 호출 수를 파악해야 한다. 어느 시스템이나 클래스 간의 데이터 생성 메소드는 최대한 1개이다. 따라서 생성 메소드 타입의 가능한 최대 메소드 호출 수는 1로 부여하였다. 또한, 수정이나 조회 메소드의 최대 메소드 호출 수는 시스템의 특성에 따라서 다르게 파악되므로 표 3에서 제시한 시스템의 메소드 호출 수를 보고 부여하였다. 즉, 표 3에서 제시하고 있듯이 클래스 간의 생성 메소드 호출 수를 뺀 나머지 클래스 간의 최대 메소드 호출 수는 3이다. 따라서 수정이나 조회 메소드의 최대 메소드 호출 수는 3으로 부여하였다. 본 논문에서 제안한 가중치는 몇 개의 시스템 사례에 적용한 결과가 우리의 직관과 일치함을 확인할 수 있었다.

다음 표 4는 The CBO of C.K.[8]에 의해서 분석된 결과인 클래스들 사이의 관계를 정의하였다.

(표 4) CBO[8]에 의해서 분석된 클래스들 사이의 관계

Class	A	B	C	D	E	F	G	H
A					1			1
B								
C		1		1				
D		1						
E			1			1		
F							1	
G								
H								

표 5는 Yacoup et. al.[11]에 의해서 분석된 클래스들 사이의 메소드 호출 수를 정의하였다.

(표 5) Yacoup et. al.[11]에 의해서 분석된 클래스들 사이의 메소드 호출 수

Class	A	B	C	D	E	F	G	H
A								1
B								
C		1						
D		1	3		3			
E	1		3	1		4		
F							3	
G								
H								

다음 표 6은 기존 메트릭과 본 논문에서 제안한 메트릭과의 비교 분석을 위하여 클래스들 사이의 결합도를 계산하여 제시하였다.

(표 6) 클래스들 사이의 결합도

Coupling(Cl,Cm)	Our's CMC	Yacoup's EOC+IOC	C.K's CBO
Coupling (A, H)	0.03	1	1
Coupling (A, E)	0.03	1	1
Coupling (E, D)	0.03	1	1
Coupling (E, C)	0.703	3	1
Coupling (E, F)	0.706	4	1
Coupling (D, C)	0.803	3	1
Coupling (C, B)	0.03	1	1
Coupling (D, B)	0.03	1	1
Coupling (F, G)	0.706	3	1

위의 표 6에서 제시한 결합도를 측정하기 위한 그 계산 과정은 다음과 같다. 즉, 표 3에서 제시한 클래스 D와 C의 경우 메소드 호출 타입과 메소드 호출 수는 C(1), U(2), R(1)이다. 따라서 Coupling (D, C) = $0.6\times\frac{1}{1}+0.3\times\frac{2}{3}+0.1\times\frac{1}{3}$ = 0.6+0.2+0.03=0.803이다. 또한, 클래스 B와 C의 경우 메소드 호출 타입과 메소드 호출 수는 R(1)이다. 따라서 Coupling(B, C)= $0.6\times\frac{0}{1}+0.3\times\frac{0}{3}+0.1\times\frac{1}{3}$ =0+0+0.03이다.

다음은 본 논문에서 제안한 결합 척도가 4장에서 증명한 결합도의 성질에 부합됨을 다음과 같이 수식 전개 과정을 통하여 제시하고자 한다. [결합도 성질 1]의 경우, 만약 두 개의 클래스 사이에 메소드 호출이 없다면 각 메소드 호출 타입에 대한 메소드 호출 수는 0 이다. 따라서 두 클래스 사이의 결합도는 $0.6\times\frac{0}{1}+0.3\times\frac{0}{3}+0.1\times\frac{0}{3}$ =0이다. 결합도의 최소 값은 0 이기 때문에 결합도가 음수가 될 수 없음을 만족한다. 또한, 클래스들 사이의 상호 작용이 없다면 결합도는 0이다라는 [결합도 성질 2]를 만족한다. [결합도 성질 3]의 경우, 클래스 내부 구성 요소에는 변화 없이 클래스들 사이의 상호 작용만 증가한다면 클래스들 간의 메소드 호출 수가 증가한다는 것으로 예를 들면 클래스 D와 C의 경우 C(1), U(2), R(1)에 대한 메소드 호출 수가 C(1), U(**3**),R(**2**)로 증가한다면 Coupling (D, C)= $0.6\times\frac{1}{1}+0.3\times\frac{3}{3}+0.1\times\frac{2}{3}$ =0.6+0.3+0.06=0.906으로 결합도는 감소하지 않는다는 [결합도 성질 3]을 만족한다. [결합도 성질 4]의 경우, 클래스 E, F 그리고 G에 대하여 클래스 간의 결합도 Coupling (E, F)=0.706, Coupling (F, G)=0.706 이므로 각각의 클래스 F, G에 대한 결합도는 Coupling(F)=1.412와 Coupling(G)=0.706 이다. 따라서 클래스 F와 G를 결합하여 새로운 클래스 H를 만든다면 Coupling(H)=0.706이 된다. 즉, 두 클래스를 결합하여 새로운 클래스 H를 만든다면 새로운 클래스 H의 결합도는 두 클래스 F와 G, 각각의 결합도의 합보다 증가하지 않는다는 것을 알 수 있다. 따라서 [결합도 성질 4]를 만족한다. [결합도 성질 5]의 경우, 클래스 간 상호 작용이 전혀 없는 두 클래스인 A(회원)와 B(도서)를 결합하여 새로운 클래스 X를 만든다면 각각의 클래스에 대한 결합도는 Coupling(A)= 0.06, Coupling(B)=0.03이고 A와 B를 결합한 클래스 X에 대한 Coupling(X)=0.09이다. 따라서 클래스 간 상호 작용이 전혀 없는 두 클래스, A, B를 결합하여 새로운 클래스 X를 만든다면 새로운 클래스 X의 결합도는 원래 두 클래스 A, B의 결합도의 합과 같다. 따라서 [결합도 성질 5]를 만족한다.

다음은 기존의 메트릭스와 본 논문에서 제안한 결합척도를 비교하고자 한다. 표 6의 CBO of C.K. [8]는 클래스 간의 구조적 그리고 동적 특성을 고려하고 있지 않으므로 클래스 간의 결합도가 정확히 측정되어지지 않음을 알 수 있다. 또한, 표 6에서 제시하고 있듯이 Yacoup et. al.[11]은 메소드 호출 타입을 고려하지 않기 때문에 클래스 간의 관계가 연관 관계라 하더라도 클래스 간의 조회(Read) 타입의 메소드 호출이 많은 경우에 상속

관계나 포함 관계의 클래스 간의 결합도 보다 높게 평가될 수 있다. 그림 2의 클래스 다이어그램에서 클래스 E와 F는 연관 관계이다. 클래스 F와 G, 클래스 E와 C 그리고 클래스 D와 C는 포함 관계이다. 표 6에서 제시하고 있듯이 본 논문에서 제안한 결합 척도인 CMC는 포함 관계의 클래스들이 결합도가 높게 나옴을 알 수 있고, 클래스 E와 F처럼 연관 관계라 하더라도 클래스 간의 관계가 데이터를 생성시키는 관계라면 결합도가 높게 나옴을 알 수 있다. 또한 메소드 호출 타입에 대한 가중치를 다르게 부여하기 때문에 클래스 간의 조회(Read) 타입의 메소드 호출이 많은 연관 관계의 경우에 상속 관계나 포함 관계의 클래스 간의 결합도 보다 높게 평가될 수는 없다. 따라서 클래스 간의 구조적 특성과 동적 특성을 만족하고 이는 우리의 직관과 일치한다. CMC는 0-1 사이로 정규화되므로 클래스 간의 결합의 정도를 명시적으로 평가할 수 있다. 클래스 간의 상속 관계나 포함 관계시 데이터의 생성은 동적 바인딩에 의해서 이루어 진다. 그러나 시스템 분석 시에 데이터 생성관계에 의해서 클래스 간의 상속 관계나 포함 관계를 고려하기 때문에 동적 바인딩의 경우를 부분적으로 고려한다고 할 수 있다.

따라서 다음 표 7은 본 논문에서 제안한 메트릭과 기존의 메트릭과의 몇 가지의 비교 요소들에 의하여 정성적으로 평가한다.

(표 7) 기존 메트릭과의 정성적 비교 평가

메트릭스 / 비교 요소	기존의 객체지향 메트릭스	제안된 메트릭
메소드 호출 수	고려됨	고려됨
클래스 간의 구조적 특성	부분적으로 고려됨	고려됨
클래스 간의 동적 특성	부분적으로 고려됨	고려됨
동적 바인딩	거의 고려되지 않음	부분적 고려됨
결합도의 정규화	고려되지 않음	고려됨
메소드 호출 타입	고려되지 않음	고려됨

6. 결론

본 논문은 기존 클래스 기반의 메트릭스에서 제안하지 않았던 클래스 간의 관계에 대한 구조적 특성과 메소드 호출 수에 의한 동적 특성을 동시에 부여하고, 클래스 간의 결합도가 0-1 사이로 선형적 증가하도록 정규화 시켜 보다 정확히 측정하고 평가할 수 있는 결합 척도를 제안하였다. 또한, 본 논문에서 제안된 결합 척도가 이론적으로 타당하다는 것을 검증하기 위하여 Briand이 제안한 프레임워크에 적용하여 증명하였다. 마지막으로, 제안한 결합 척도의 정확성과 실용성을 검증하기 위하여 적용 사례를 제시 하였고 기존 메트릭스와의 비교 분석을 통해서 본 논문에서 제안한 결합 척도가 좀 더 정확하게 정량적으로 측정됨을 확인하였다. 따라서, 클래스 간의 상호 의존이 강한 클래스들을 그룹화하여 도출되는 컴포넌트 기반 시스템의 컴포넌트를 효율적으로 평가 및 설계할 수 있으며 또한, 컴포넌트 기반 시스템에 의존되어 설계되는 서비스 지향 시스템의 서비스를 식별하는데 효율적으로 적용될 수 있다.

참 고 문 헌

[1] K. H. Bennett and J. Xu, "Software Services and Software Maintenance," Proceedings of the 7thEuropeanConferenceonSoftwareMaintenanceandRe-engineering,Benevento,2003.

[2] A. Arsanjani, "Service-oriented modeling and architecture," *IBMdeveloperWorks*,Nov.,2004.

[3] John Chesseman, John Daniels, "UML Components: A Simple Process for Specifying Component-Based Software," AddisonWesely,2001

[4] N.E. Fenton, "Software Metrics: A Rigorous Approach," Chapman and Hall, London,1991.

[5] 최 미숙, 이종석, "컴포넌트 기반 시스템에서 클래스들 간의 정적 그리고 동적 특성을 적용한 컴포넌트 메트릭스," 정보과학회 논문지, 제 33권 3호, pp.301-315, 2006

[6] 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, 1996

[7] 전병선, J2EE Enterprise System, 객체지향 CBD 개발 방법론, 영진닷컴, 2004

[8] 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

[9] Li, W., and S. Henry, "Object Oriented Metrics that predict Maintainability", Jounal of Systems and Software, Vol 23, No. 2, 1993

[10] Henderson-Sellers, Brian, "Object-Oriented Metrics," Prentice Hall, 1996

[11] Sherif M. Yacoub, Hany H. Ammar, and Tom Robinson, "Dynamic Metrics for Object Oriented Designs," Software Metrics Symposium Proceedings, 1999

[12] 화지민, 이숙희, 권용래, "객체지향 시스템에서의 클래스 간 의존성 강도 측정을 위한 커플링 척도," 정보과학회 논문지, 제 14권 1호, pp.81- 85, 2008

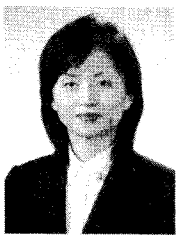
◐ 저 자 소 개 ◑

최 미 숙

1990년 전북대학교 수학과 졸업 (이학사)
1994년 숙명여자대학교 컴퓨터과학과 석사과정 졸업 (이학석사)
2002년 숙명여자대학교 컴퓨터과학과 박사과정 졸업 (이학박사)
1995년~1999년 나주대학 소프트웨어 개발과 전임교수
2004년~2006년 우석대학교 컴퓨터공학과 초빙교수
2006년~현재 우석대학교 기초및자연과학연구소 연구교수
관심분야 : 소프트웨어 개발 방법론, 컴포넌트 기반 시스템, 소프트웨어 매트릭, SOA

이 종 석

1988년 서울대학교 계산통계학과 학사
1990년 서울대학교 계산통계학과 석사
2001년 서울대학교 컴퓨터공학 전공 박사
1993년~현재 우석대학교 컴퓨터교육과 교수
관심분야 : 소프트웨어 공학, 소프트웨어 복잡도, 컴포넌트 기반 시스템, SOA



이 서 정

1989년 숙명여자대학교 전산학과 졸업(이학사)
1991년 동대학교 대학원 전산학과 석사과정 졸업 (이학석사)
1998년 동대학교 대학원 전산학과 박사과정 졸업 (이학박사)
1998년~2003년 동덕여자대학교 강의교수
2004년 숭실대학교 정보미디어연구센터 연구교수
2005년~현재 한국해양대학교 조교수
관심분야 : 소프트웨어 개발방법론, SOA, 임베디드 소프트웨어 설계