한국산업정보학회논문지 제6권 제4호 (2001. 12)

객체 지향 시스템에서의 클래스 결합도 척도

(Coupling Measures for Classes in Object-Oriented System)

이 종석, 천 은홍*
(Jong-Seok Lee, Eun-Hong Cheon)

요 약 소프트웨어의 품질 측정의 중요성이 커짐에 따라 새로운 소프트웨어 척도를 개발하기 위한 수많은 노력이 나타났다. 객체 지향 개발 방법론의 중요성에서 볼 때 이러한 경향이 나타나는 특별한 분야는 객체 지향 시스템의 결합도 측정이다. 본 논문에서는 객체 지향 시스템의 결합도를 측정하기 위해 메소드에 의한 응집도 COM과, 클래스에 의한 결합 COC를 제안하였다. 그리고 이를 Briand가 제안한 결합도 성질을 이용하여 평가하고, C++ 언어로 작성된 시스템에 적용하여 다른 결합도 척도와 비교하였다.

Abstract As the importance of software measurement increases, there are more researches developing the new software measures. Given the importance of object-oriented development techniques, one specific area where this has occurred is coupling measurement in object-oriented systems. This thesis presents the coupling measure COM based on the methods and COC based on the classes in measuring the coupling of object-oriented systems. And, it evaluates them using the coupling properties suggested by Briand while it compares them with other coupling measures by applying them to a system developed in the language of C++ programming language.

1. 서 론

제품의 품질과 개발 프로세스를 향상시켜 주기 위해서는 소프트웨어 제품의 특징들을 이해하는 것이 중요하다. 그런데 DeMarco가 '측정할 수 없는 것은 이해할 수 없다.'고 이야기한 것처럼, 이해하기 위해서는 반드시 측정할 수 있어야 한다[4]. 이와 같이 소프트웨어 개발 시장에서 소프트웨어 품질은 점점 중요한 위치를 차지하고 있고, 이는 소프트웨어 척도 분야를 더욱 중요하게 만들고 있다.

그리고 현재에는 캡슐화(encapsulation), 상속성(inheritance), 다형성(polymorphism) 등과 같은 개념을 이용하는 객체 지향 방법론이 많이 사용되고 있다. 이는 객체 지향 방법론이 기존의 절차적 방법론보다는 유지보수와 재사용을 더 잘 지원하기 때문이다. 하지만 개발자의 무지 혹은 미숙함으로 인하여 객체 지향 방법론의 장점들을 충분히 살리지 못할 수도 있기 때문에, 단지 객체 지향 방법론만을 도입한다고 해서 좋은 품질의 소프트웨어를 개발할 수 있는 것은 아니다. 그러므로 소프트웨어가 객체 지향 방법론에 따라 잘 구성되었는지를 정량화하기 위한 척도들이 점점 더 요구되고 있는 실정이다.

이러한 이유 때문에 수많은 척도들이 소프트웨어 품질을 측정하기 위해 등장하였다. 이중에서 두 모듈 간의 의존도를 측정하는 결합도는 소프트웨어 품질을 측정하는 중요한 척도이다. Stevens et al.은 결합도를 '한 모듈에서 다

* 우석대학교 컴퓨터공학과

른 모듈로의 결합에 의하여 생긴 관련 정도'라고 정의하였다[5]. 그러므로 모듈 간의 결합도가 강하면 강할수록 이 모듈들은 이해하고 변경하는 것이 더 어렵고, 이에 따라 시스템은 더욱 복잡해 진다[1].

오늘날 객체 지향 시스템을 위한 수많은 결합도들이 제안되고 있다. 하지만 객체 지향시스템에서는 절차적 시스템과는 달리 결합도를 구성하는 메카니즘들이 서로 다르기 때문에 이를 간단한 방법으로 측정하는 것은 어렵다. 따라서 많은 척도들이 척도 모음(metric suite)을 이용하여 결합도를 정의하고 있다. 그러나 너무 많은 척도들이 모인 것은 각 척도값들 간에 차이가 생겼을 때 이를 적절히 해석하지 못하는 단점이 있다. 이를 해결하기 위해 본 논문에서는 결합도를 구성하는 메카니즘들의 대부분을 측정하는 결합도 척도로 COM과 COC를 제안한다.

다음 장에서는 관련 연구를 수행하고, 3장에서는 결합도 척도를 제안하여 이를 결합도 성질을 이용하여 평가한다. 4장에서는 본 논문에서 제안된 결합도 척도를 C++ 언어로 된 소프트웨어에 실제 적용하여 그 결과를 분석하고, 5장에서는 결론과 향후 연구에 관해 말할 것이다.

2. 관련 연구

2.1 CBO(Coupling Between Objects)

CBO는 [2]에서 '상속되지 않고 결합되어 있는 다른 클래스의 수'로 정의하였다. 여기서 결합되어 있다는 것은 하나의 메소드가 다른 클래스의 메소드나 애트리뷰트를 사용한다는 것을 의미한다. [3]에서 CBO는 다시 정의되었는데, 이는 '결합되어 있는 다른 클래스의 수'로 상속을 포함하고 있다. 이 방법은 하나의 클래스와 관련이 있는 클래스들의 수를 계산함으로써 클래스의 결합도를 간단히 측정할 수 있지만, 한 클래스 내에서 같은 클래스 내에 있는 멤버들을 여러 번 사용할 때도 한 번 사용한 것과 같은 값을 가진다.

또한 [6]에서는 외부 클래스에 있는 애트리뷰트와 메소드의 합으로 CBO를 정의하였는데, 이 방법에서는 같은 애트리뷰트와 메소드를 여러 번 호출하여도 이를 구별할 수 없다.

2.2 RFC(Response for Class)

RFC는 클래스에 있는 모든 메소드의 수에 각 클래스에 의해 호출된 메소드의 수를 합한 것[2][3][7]으로, 결국 그 클래스에서 사용 가능한 메소드의 수라고 할 수 있다. 그런데 여기에는 잠재적으로 실행 가능한 메소드도 포함되므로, 직접적으로 호출되는 메소드뿐만 아니라 간접적으로 호출되는 메소드까지도 포함한다.

2.3 MPC(Message Passing Coupling)

MPC는 [4]에서 '클래스에서 정의된 send 문장의 수'로 정의하였다. 이러한 send 문장의 수는 내부 메소드들의 구현이 다른 클래스에 있는 메소드에 얼마나 종속되어 있는지 가리킨다. 하지만 한 클래스 내에서 여러 개의 다른 클래스로 메시지를 패싱한 결과와, 한 클래스에게 같은 메시지를 여러 번 패싱한 결과가, 앞의 경우가 결합도가 더 높다는 것을 직관적으로 알 수 있음에도 불구하고 같은 값을 가진다. 이러한 MPC는 [7]에서도 마찬가지로 정의하고 있다.

2.4 DAC(Data Abstraction Coupling)

DAC는 [4]에서 '추상 자료형(abstract data type)의 수'로 정의하였다. 여기에서 추상 자료형은 시스템에 있는 클래스로, 클래스에서 정의된 애트리뷰트의 형이다. 추상 자료형의 애트리뷰트의 수는 다른 클래스의 정의에 종속적인 자료 구조의 수를 가리키므로, DAC 값이 크다는 것은 다른 클래스의 정의에 연관된 자료 구조가 많다는 것을 의미한다. 그러나 [4]에서는 DAC값이 추상 자료형의 개수인지, 추상 자료형의 애트리뷰트의 개수인지 정확하게 정의하지 않은 반면에, [7]에서는 원격 추상 자료형의 수로 DAC를 정의하여, 어떤 클래스가 다른 클래스의 추상 자료형을 여러 번 사용한다고 해도 그 값은 항상 1이다.

2.5 Ce and Ca(efferent and afferent Coupling)

[8]에서 공통적인 목적을 함께 수행하는 클래스의 집합을 카테고리(category)라고 정의한 후, 이 카테고리 안에 있는 클래스들과 연관이 있는 이 카테고리 밖에 있는 클래스들의 수로 Ca를 정의하고, 카테고리 밖에 있는 클래스들과 연관이 있는 이 카테고리 안에 있는 클래스들의 수로 Ce를 정의하였다. 그러나 클래스들 간의 종속 관계를 구성하는 것을 정확하게 언급하지 않았다.

2.6 COF(Coupling Factor)

COF는 상속을 통해서 연관이 되지 않은 클래스 간의 클라이언트-서버 관계로 정의하고, 값이 0과 1 사이에 오도록 정규화시켰다[9]. 그러나 다형성이나 메소드 오버로딩(overloading)에 대해서는 언급하지 않았다.

3. 결합도 척도

결합도는 모듈이 다른 모듈과 얼마나 밀접하게 연관되어 있는지 측정하는 척도이므로, 작은 결합도를 가진 모듈이 다른 모듈에 영향을 받지 않고 독립적인 기능을 수행할 수 있다. 그러나 객체 지향 프로그래밍은 재사용성을 위해 상속을 요구하기 때문에 상속을 통한 결합도는 높을수록 좋다고 할 수 있다. 그러나 상속을 통한 결합도가 너무 높다면 이 경우에도 역시 캡슐화와 정보 은닉을 나쁘게 만든다. 더구나 부모 클래스로부터 메소드나 애트리뷰트를 상속받는다는 것은 부모 클래스의 성질을 이해하여야 하기 때문에 좋은 의미의 결합도라고 하더라도 클래스를 이해하는데 어려움을 준다. 따라서 상속을 통한 결합도도 다른 의미의 결합도와 마찬가지로 클래스의 복잡도를 높인다고 할 수 있다.

또한 두 클래스가 연관되는 방법은 한 클래스의 메소드가 다른 클래스의 메소드나 애트리뷰트를 사용하는 경우와, 클래스 내에서 인스턴스의 형으로 다른 클래스를 사용하는 경우이다. 그런데 이 두 가지는 서로 관계가 있는 부분도 있지만, 기본적으로는 다른 메카니즘이기 때문에 별개로 생각하여야 한다.

그러므로 본 논문에서는 두 가지 경우 각각에 대하여 결합도를 제안하고자 한다. 이때 상속을 통한 결합도도 다른 결합도와 같이 취급하여 결합도를 측정한다.

3.1 메소드에 의한 결합도

3.1.1 척도 정의

두 클래스가 메소드를 통하여 연관되는 경우는 다음과 같다.[1]

(1) 메소드가 애트리뷰트를 참조
(2) 메소드가 다른 메소드를 호출
(3) 메소드가 다른 메소드로부터 포인터를 받음1).

이러한 경우들을 만족시키기 위해 클래스 내에 있는 메소드가 참조하는 다른 클래스의 구성 요소의 집합 SA, SM, SP를 다음과 같이 정의한다.

정의 1. SA(Set of Attribute)

$$SA(c) = \{ra(c) \notin c \mid ra(c)\}$$

ra(c) : 클래스 c에 있는 메소드가 참조하는 애트리뷰트

정의 2. SM(Set of Method)

$$SM(c) = \{im(c) \notin c \mid im(c)\}$$

im(c) : 클래스 c에 있는 메소드가 호출하는 메소드

정의 3. SP(Set of Pointer to method)

$$SP(c) = \{rp(c) \notin c \mid rp(c)\}$$

rp(c) : 클래스 c에 있는 메소드가 받는 메소드의 포인터

이 SA, SM, SP에 있는 원소들은 메소드에서 사용되는 외부 요소로 결합도에 영향을 주는 요소들이다. 따라서 각 집합의 cardinality의 합이 메소드에 의해 생긴 결합도라고 할 수 있다.

그런데 다음과 같은 경우를 생각해 보자.

경우 1. 동일한 메소드를 여러 번 호출하는 경우
경우 2. 여러 개의 다른 메소드를 호출하는 경우

경우 1에서 클래스 내의 내부 메소드가 외부의 동일한 메소드를 호출한 횟수는 내부 메소드의 구현이 다른 클래스에 있는 메소드와 연관되어 있음을 가리킨다. 따라서 메소드를 호출한 횟수는 결합도와 관계가 있다.

그러나 경우 1과 경우 2를 MPC의 경우에서처럼 같은 값을 가지는 것으로 취급한다면 문제가 있다. 왜냐하면 직관적으로도 분명히 여러 개의 다른 메소드를 호출하는 경우가 동일한 메소드를 여러 번 호출하는 것보다 더 강력하게 결합되어 있음을 알 수 있다. 그러므로 경우 2가 경우 1보다 더 높은 값을 가져야 한다.

이에 본 논문에서는 동일한 메소드를 호출할 때마다 상수를 곱한다. 이 상수는 동일한 메소드를 호출하면 그 횟수에 따라 결합도의 강도는 늘어나지만, 점차로 그 영향력은 감소하기 때문에 값의 범위는 0에서 1사이에 있어야 한다. 이러한 것은 메소드를 호출하는 경우뿐만 아니라, 애트리뷰트를 참조하는 경우와, 메소드를 가리키는 포인터를 받는 경우 모두에 적용될 수 있다. 따라서 메소드에 의한 결합도 척도 COM을 다음과 같이 정의한다.

정의 4. COM(Coupling Of Method)

$$COM(c) = \bigcup_{x \in SA(c)} \sum_{i=0}^{\phi(x)-1} a^i + \bigcup_{y \in SM(c)} \sum_{j=0}^{\phi(y)-1} a^j + \bigcup_{z \in SP(c)} \sum_{k=0}^{\phi(z)-1} a^k$$

$\phi(x)$: 요소 x의 사용 횟수

여기에서 상수 값은 0보다 크고, 1보다 작은 값으로서, 0에 근접한 값을 가지면 동일한 외부 요소를 참조할 때 그

1. 메소드가 공통 변수를 공유하는 경우가 있지만 이는 객체 지향 패러다임에는 맞지 않으므로 본 논문에서는 제외하였다.

횟수가 결합도 척도 COM에 영향을 거의 미치지 않고, 1에 가까운 값을 가지면 동일한 외부 요소를 여러 번 참조한 결과와 다른 외부 요소들을 참조한 결과의 차이가 거의 없다.

3.1.2 척도의 수학적 성질

본 논문에서는 [10]에서 제시한 결합도 척도의 성질을 이용하여 COM을 평가하고자 한다. 이를 위해 COM의 첫 번째 항, 두번째 항과 세번째 항을 각각 SAI, SMI, SPI라 가정한다.

성질 1. 결합도는 0이상이다.

COM 값은 음수가 나올 수 없다. 따라서 이 성질을 만족한다.

성질 2. 모듈 간에 관계가 없다면 결합도는 0이다.

외부 요소를 전혀 참조하지 않는 클래스가 있다면 그 class의 SA, SM, SP의 cardinality가 모두 0이 되므로, SAI, SMI, SPI의 값이 모두 0이다. 따라서 COM 값은 0이 되므로 성질 2를 만족한다.

성질 3. 모듈 간에 관계를 더하면 결합도는 감소하지 않는다.

새로운 외부 메소드를 호출하는 문장이 클래스에 더해진다면 그 클래스의 SAI와 SPI의 값은 동일하지만, SMI의 값이 증가하게 되므로 COM 값이 증가한다. 따라서 이 성질을 만족한다.

성질 4. 두 개의 모듈을 합쳐서 만든 모듈의 결합도는 두 모듈의 결합도의 합보다 크거나 같다.

한 클래스가 다른 클래스의 내부 메소드를 호출하는 경우, 두 클래스를 합치면 한 클래스의 SMI 값이 줄어들게 되므로 각 클래스의 SMI 값을 합한 것보다 합친 클래스의 SMI 값이 작다. 또한 두 클래스가 동일한 외부 메소드를 똑같이 호출한다면, 두 클래스를 합친 클래스의 SMI 값은 1보다 작은 상수를 곱함에 따라 각 클래스의 SMI 값을 합한 것보다 작다. 따라서 합친 클래스의 COM 값이 작게 되므로 이 성질을 만족한다.

성질 5. 두 개의 모듈이 서로 관련이 없을 경우 두 모듈을 합쳐서 만든 모듈의 결합도는 두 모듈의 결합도의 합과 같다.

두 클래스가 서로 연결이 되어 있지 않지만, 동일한 외부 메소드를 호출하는 경우를 가정하자. 비록 외부 메소드를 호출하는 횟수는 두 클래스의 합과 같지만 동일한 메소드를 호출할 때마다 그 강도가 줄어들고 이는 1보다 작은 상수에 의해 SMI 값에 영향을 미치기 때문에 두 클래스를 합친 클래스의 SMI 값은 각 클래스의 SMI 값을 합한 것보다 작게 된다. 따라서 이 성질은 만족하지 않는다.

3.2 클래스에 의한 결합도

3.2.1 척도 정의

두 클래스가 클래스를 통하여 연관되는 경우는 다음과 같다.

(1) 클래스가 클래스 애트리뷰트의 형
(2) 클래스가 메소드의 파라미터이거나 반환형
(3) 클래스가 메소드의 내부 변수의 형.

클래스에서도 메소드에서와 같이 참조하는 외부 클래스의 집합 SCA, SMP, SLV를 정의할 수 있다.

정의 5. SCA(Set of Class' Attribute)

$$SCA(c) = \{at(c) \notin c \mid at(c)\}$$

at(c) : 클래스 c의 애트리뷰트의 형으로 쓰이는 외부 클래스

정의 6. SMP(Set of Method's Parameter)

$$SMP(c) = \{mp(c) \notin c \mid mp(c)\}$$

mp(c) : 클래스 c의 메소드의 파라미터이거나 반환형으로 쓰이는 외부 클래스

정의 7. SLV(Set of method's Local Variable)

$$SLV(c) = \{lv(c) \notin c \mid lv(c)\}$$

lv(c) : 클래스 c에 있는 메소드의 내부 변수의 형으로 쓰이는 외부 클래스

메소드에 의한 결합도에서와 같이 동일한 클래스를 이용하여 클래스 내에 여러 개의 애트리뷰트를 선언하는 것보다는 여러 개의 다른 클래스로 애트리뷰트를 선언하는 것이 더 결합도가 높다고 할 수 있다. 이러한 것은 SMP나 SLV에서도 마찬가지이다. 따라서 결합도 척도 COC는 사용 횟수가 늘어남에 따라 결합도 값은 커지지만 그 영향력은 감소되도록 외부 클래스를 사용한 횟수만큼 1보다 작은 상수를 곱한다.

정의 8. COC(Coupling Of Class)

$$COC(c) = \bigcup_{x \in SCA(c)} \sum_{i=0}^{\mu(x)-1} \beta^i + \bigcup_{y \in SMP(c)} \sum_{j=0}^{\mu(y)-1} \beta^j + \bigcup_{z \in SLV(c)} \sum_{k=0}^{\mu(z)-1} \beta^k$$

μ(x) : 클래스 x의 인스턴스 수

3.2.2 척도의 수학적 성질

평가를 위해 COC의 첫번째 항, 두번째 항, 세번째 항을 각각 SCAI, SMPI, SLVI라 가정한다.

성질 1. 결합도는 0이상이다.

COC 값은 음수가 나올 수 없다. 따라서 이 성질을 만족한다.

성질 2. 모듈 간에 관계가 없다면 결합도는 0이다.

클래스 내에서 외부 클래스를 전혀 참조하지 않는다면 그 클래스의 SCAI, SMPI, SLVI의 값이 모두 0이 되므로 성질 2를 만족한다.

성질 3. 모듈 간에 관계를 더하면 결합도는 감소하지 않는다.

새로운 클래스를 애트리뷰트의 형으로 사용하는 문장이 클래스에 더해진다면 그 클래스의 SCAI의 값이 증가하고, SMPI와 SLVI의 값은 동일하기 때문에 COC 값이 증가한다. 따라서 이 성질을 만족한다.

성질 4. 두 개의 모듈을 합쳐서 만든 모듈의 결합도는 두 모듈의 결합도의 합보다 크거나 같다.

한 클래스가 다른 클래스를 애트리뷰트의 형으로 사용하는 경우, 두 클래스를 합치면 한 클래스의 SCAI 값이 줄어들게 되므로 각 클래스의 SCAI 값을 합한 것보다 합친 클래스의 SCAI 값이 작다. 또한 두 클래스가 동일한 클래스를 애트리뷰트의 형으로 사용한다면, 두 클래스를 합친 클래스의 SCAI 값은 1보다 작은 상수를 곱함에 따라 각 클래스의 SCAI 값을 합한 것보다 작다. 따라서 합친 클래스의 COC 값이 작게 되므로 이 성질을 만족한다.

성질 5. 두 개의 모듈이 서로 관련이 없을 경우 두 모듈을 합쳐서 만든 모듈의 결합도는 두 모듈의 결합도의 합과 같다.

두 클래스가 서로 연결이 되어 있지 않지만, 동일한 클래스를 애트리뷰트의 형으로 사용하는 경우를 가정하자. 비록 클래스를 애트리뷰트의 형으로 사용하는 횟수는 두 클래스의 합과 같지만 동일한 클래스를 애트리뷰트의 형으로 사용할 때마다 그 강도가 줄어들고, 이는 1보다 작은 상수에 의해 SCAI 값에 영향을 미치기 때문에 두 클래스를 합친 클래스의 SCAI 값은 각 클래스의 SCAI 값을 합한 것보다 작게 된다. 따라서 이 성질은 만족하지 않는다.

본 연구에서 제안한 결합도를 결합도 성질에 대하여 CBO와 비교한 결과는 <표 1>과 같다. 비록 COM이나 COC 모두 성질 5를 만족하지는 못하지만, 다른 결합도 척도가 만족하는 성질들을 최소한 만족하므로 결합도 척도로서 적합하다고 볼 수 있다.

<표 1> Briand의 결합도 성질 비교

	성질 1	성질 2	성질 3	성질 4	성질 5
CBO	O	O	O	O	X
COM	O	O	O	O	X
COC	O	O	O	O	X

4. 실험 및 평가

4.1 적용 데이터

본 논문에서 제안한 결합도 척도를 분석하기 위해 C++ 언어로 작성된 프로그램을 이용하여 테스트하였다. 이 프로그램은 객체 지향 시스템을 위한 개발 환경 시스템(DOOD)으로, 4명의 대학원생들이 18개월 동안 작성한 것이다. 이 시스템의 크기는 <표 2>와 같다.

이 시스템의 CBO, RFC, MPC, DAC를 계산한 결과는 <표 3>에 나타나 있다.

<표 2> 시스템 크기

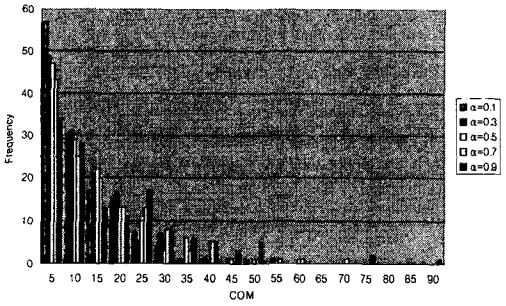
		줄 수	클래스	메소드	애트리뷰트
전 체		26184	134	677	1489
클래스	최대	1549	-	140	38
	최소	56	-	1	0

<표 3> DOOD 시스템의 결합도

	CBO	RFC	MPC	DAC
최대	34	169	113	32
평균	5.9	19	13.5	4.58
중앙값	5	12	7	4
표준편차	4.53	21	17.6	4.07

4.2 데이터 분석

0.1부터 0.9까지 상수 값을 변화시켜 시스템에 적용했을 때의 COM의 분포도와 통계는 각각 <그림 1>, <표 4>와 같다. <그림 1>을 보면 이 시스템은 상수 값에 상관없이 편중되어 있으며, COM 값이 25이하인 클래스들이 거의 대부분임을 알 수 있다. 그러나 동일한 외부 요소의 참조 횟수가 작은 클래스일 경우에는 상수 값이 COM에 거의 영향을 주지 못하는 반면에, 동일한 외부 요소를 여러 번 참조하는 클래스는 상수 값에 따라 COM이 큰 차이를 보인다.

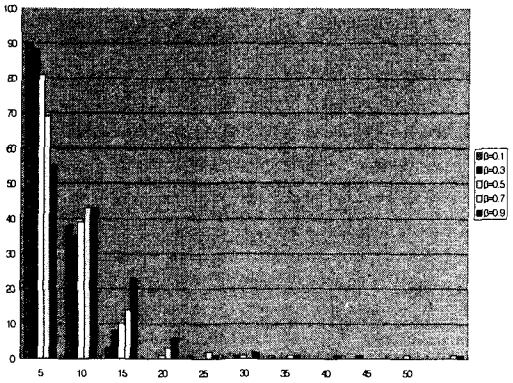


<그림 1> COM의 분포도

<표 4> 척도 COM의 통계값

α	0.1	0.3	0.5	0.7	0.9
최대	50	54	60	68	90
평균	9.29	10.35	11.5	13.32	16.83
중앙값	6	8	9	10	13
표준편차	8.68	9.44	10.53	12.31	16.1

이러한 경우는 COC에서도 마찬가지인데, 상수 값을 변화시켜 가며 시스템에 적용했을 때 COC의 분포도와 통계는 각각 <그림 2>와 <표 5>와 같다.



<그림 2> COC의 분포도

<표 5> 척도 COC의 통계값

β	0.1	0.3	0.5	0.7	0.9
최대	35	41	50	63	91
평균	4.65	5.17	5.86	6.74	8.44
중앙값	4	4	5	5	6
표준편차	4.31	4.86	5.63	6.95	9.83

COM과 COC의 다른 결합도 척도와의 상관관계는 <표 6>과 같다.

<표 6> 결합도 척도와의 상관관계

	COM				
	0.1	0.3	0.5	0.7	0.9
CBO	0.544	0.561	0.57	0.571	0.553
RFC	0.643	0.65	0.653	0.653	0.651
MPC	0.881	0.896	0.91	0.916	0.908
DAC	0.504	0.522	0.532	0.533	0.52
	COC				
	0.1	0.3	0.5	0.7	0.9
CBO	0.948	0.942	0.926	0.9	0.839
RFC	0.559	0.602	0.636	0.674	0.732
MPC	0.589	0.617	0.631	0.651	0.675
DAC	0.998	0.992	0.98	0.957	0.899

COM은 메소드를 통한 결합도이므로, send 문장의 개수를 나타내는 MPC와 특히 상관관계가 높고, 사용 가능한 메소드의 수인 RFC와도 상관관계가 높다. CBO와 DAC는 메소드보다는 외부 클래스에 의한 결합도의 성격이 강하므로, COC와 상관관계가 아주 높다. 그런데 COC에서 값이 0.1일 때 CBO, DAC와의 상관관계가 가장 높고, 값이 0.9일 때 RFC, MPC와의 상관관계가 가장 높다. 이는 CBO와 DAC는 클래스의 사용 횟수를 고려하지 않고, MPC는 사용 횟수를 고려하기 때문이다.

5. 결 론

본 논문에서는 객체 지향 시스템의 결합도 척도로서 메소드에 의한 결합도 COM, 클래스에 의한 결합도 COC를 제안하였다. 이 COM과 COC에서는 동일한 외부 요소나 클래스를 사용할 때마다 결합도의 값은 커지지만 그 영향력은 줄어들도록 1보다 작은 상수를 곱하고 메소드나 클래스에 의해 생기는 모든 종류의 결합도를 합하였다.

본 논문에서 제안한 척도들은 Briand가 제안한 결합도 성질 중 5번만 제외하고는 모두 만족한다. 또한 이를 C++ 프로그래밍 언어로 작성된 시스템에 적용하여 다른 결합도 척도들과 비교한 결과 COM은 RFC, MPC와 상관관계가 높았고, COC는 CBO, DAC와 상관관계가 높았다.

앞으로의 연구 과제는 더 많은 실제 데이터를 이용하여 정확하게 시스템의 성질을 반영하는 상수 α와 β 값을 찾아내고, 복잡도나 이해가능성과 같은 시스템의 외부 성질과의 관계를 조사하여 COM과 COC의 타당성을 입증하는 것이다.

참 고 문 헌

1. L.C. Briand, John W. Daly, and J.K. Wst, "Unified Framework for Coupling Measurement in Object-Oriented Systems", IEEE Trans. Software Eng., vol. 25, no. 1, pp. 91-121, 1999

2. S.R. Chidamber and C.F. Kermerer, "Towards a Metric Suite for Object Oriented Design", A Paepcke, ed., Proc. Conf. Object-Oriented Programming: Systems, Languages and Applications, OOPSLA'91, Oct. 1991.

3. S.R. Chidamber and C.F. Kermerer, "A Metric Suite for Object Oriented Design", IEEE Trans. SE., vol. 20, no. 6, pp. 476-493, 1994.

4. W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability", J. Systems and Software, vol. 23, no. 2, pp. 111-122, 1993.

5. W. Stevens, G. Myers, and L. Constantine, "Structured Design", IBM Systems Journal, vol. 13, no. 2, pp. 115-139, 1974

6. J. A. Lewis, "Quantified Object-Oriented Development: Conflict and Resolution", Proc. 4th SW Quality Conf. vol. 1, pp.220-229, 1995.

7. B. Henderson-Sellers, Object-Oriented Metrics Measures of Complexity, Pretice Hall, 1996.

8. R. Martin, "OO Design Quality Metrics-An Analysis of Dependancies", position paper, proc. Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, OOPSLA'94, Oct. 1994.

9. F. Abreu, M. Goulao, and R. Esteves, "Toward the Design Quality Evaluation of Object-Oriented Software Systems", Proc. Fifth Int'l Conf. Software Quality, Austin, Texas, Oct. 1995.

10. L. Briand, S. Morasca, V. R. Basili, "Property-Based Software Engineering Measurement", IEEE Transactions on SE, Vol.22, NO.1, pp.68-85, Jan. 1996.

이 종 석 (Jong-Seok, Lee)

1988년 2월 서울대학교 계산통계학과 졸업(이학사)

1990년 8월 서울대학교 대학원 계산통계학과 졸업(이학석사)

2001년 2월 서울대학교 대학원 전기컴퓨터공학부 졸업(공학박사)

1993년 3월 ~ 현재 우석대학교 컴퓨터공학부 조교수

관심분야 : 소프트웨어 공학,

천 은 홍 (Eun-Hong Cheon)

1981년 2월 광운대학교 응용전자 공학과(공학사)

1985년 2월 아주대학교 전자공학과 (공학석사)

1998년 8월 아주대학교 컴퓨터 공학과(공학박사)

1988년 9월 ~ 현재 우석대학교 컴퓨터공학과 부교수

관심분야 : 컴퓨터네트워크, 정보보안