韓國IT서비스學會誌
第10卷 第4號
2011年 12月

서비스 지향 아키텍처(SOA) 기반 소프트웨어의 새로운 결합도 메트릭

유 문 성*

New Coupling Metrics for SOA Based Softwares

Moon-Sung Yoo*

■ Abstract ■

Service Oriented Architecture (SOA) is rapidly emerging as the efficient approach in contemporary complex, heterogeneous computing environments. SOA increases the adaptability by loose coupling and its main feature is that three elements such as service provider, service requester and service registry are connected with each other systematically. To design the service-oriented system efficiently, a metric to measure the coupling between services accurately is needed.

In this paper, we propose four coupling metrics for SOA based softwares. First, we suggest a coupling metric for service-oriented systems by modifying an established coupling metric of object-oriented systems. Then we suggest another coupling metric which includes indirect coupling between services. We also suggest two relative coupling metrics to measure coupling between subsystems.

We investigate the theoretical soundness of the proposed metrics by the axioms of Briand et al. Finally, we apply the presented metrics to an industrial-scale case study.

Keyword : SOA, Service-Oriented Architecture, Coupling, Software Metrics, SOA Based Softwares, Service-Oriented Systems

논문투고일 : 2011년 07월 22일 논문수정완료일 : 2011년 09월 06일 논문게재확정일 : 2011년 09월 30일

* 상지대학교 컴퓨터정보공학부 교수

1. 서 론

서비스 지향 구조(SOA)[1, 8]는 현재 소프트웨어 개발의 최적의 아키텍처로 각광받고 있으며 기존의 객체 지향 개발과 컴포넌트 기반 개발을 뛰어넘는 방안으로 제시되고 있다. SOA는 전통적인 프로그램 중심의 설계/개발 방식에서 비즈니스 프로세스 관점에서 재활용 가능한 단위로 서비스를 설계/개발하게 함으로써 특정 프로세스나 서비스 변경 또는 내부/외부 시스템과의 비즈니스 통합시 효율적이고 빠른 대응이 가능하다는 점에서 그 의미가 깊다[2]. SOA 서비스 방식은 사용자 인터페이스로부터 메시징 채널까지 독립된 서비스 간의 느슨한 결합(loosly-coupled)을 통해 제공된다. 따라서 SOA는 컴포넌트 방식에 비해 비즈니스 서비스를 실행하고 서비스 컴포넌트를 운용하는데 있어 보다 넓고 확장 가능한 아키텍처를 제공한다.

SOA 기반 소프트웨어를 좀 더 효율적으로 설계하고 품질을 높이기 위하여 여러 가지 메트릭이 제안되었다[13-17]. 그 중 결합도는 서비스 간의 의존성 정도를 측정하는 메트릭이다. 서비스의 재사용은 결합도가 낮을수록 재사용되기가 용이하므로 시스템을 되도록 결합도가 낮게 구축하는 것이 필요하다. 기존의 결합도에 대한 연구는 주로 객체지향 시스템이나 컴포넌트 기반 시스템을 대상으로 이루어져 왔다. SOA 기반 소프트웨어가 현재 활발히 개발되고 있으므로 이 소프트웨어의 결합도에 대한 연구가 필요하다. 본 논문에서는 객체지향 시스템에서 많이 적용하는 결합도 메트릭을 SOA 기반 소프트웨어에 맞게 변형하였고 간접적으로 호출한 서비스의 결합도를 반영한 메트릭과 상대적 비교를 할 수 있는 메트릭을 제안하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로서 객체 지향 시스템과 컴포넌트 기반 소프트웨어의 결합도 메트릭과 기존의 SOA 기반 소프트웨어 결합도 메트릭을 소개한다. 제 3장에서는 SOA 기반 소프트웨어 관련 새로운 결합도 메트릭을 제시한다. 제 4장에서는 제 3장에서 제시한 결합도 메트릭이 이론적으로 바람직한 것인가를 Briand 등이 정의한 공리에 적용하여 조사한다. 제 5장에서는 온라인 쇼핑몰 주문 시스템을 이용하여 제안한 서비스 간의 결합 메트릭 적용 사례를 제시하고 제 6장에서는 결론을 맺는다.

2. 관련 연구

기존의 결합도에 대한 연구는 주로 객체지향 시스템이나 컴포넌트 기반 시스템을 대상으로 이루어져 왔으며 여러 가지 결합도가 제안되었다[3, 4, 6, 7, 11, 12, 18]. 그 중 C&K metrics인 CBO(Coupling Between Objects)와 RFC(Response for a Class)가 대표적으로 사용되고 있다[11]. CBO(c)는 클래스 c에서 결합되어 있는 다른 클래스들의 수이다. 결합되어 있다는 것은 하나의 메소드가 다른 클래스의 메소드나 인스턴스를 사용함을 말한다. RFC(c)는 클래스 c에서 있는 메소드 개수와 클래스 c에서 호출한 다른 클래스의 메소드의 개수의 합이다.

SOA 기반 소프트웨어를 대한 여러 가지 메트릭 중 결합도에 관련된 것은 Huynh[14], Ma[15], Perepletchikov[17] 등의 연구가 있다.

Huynh[14]는 서비스 사이의 관계를 링크로 정의하였으며 링크당 평균 메소드 수와 평균 매개변수의 수 등을 계산하여 결합도를 구하였으나 이것은 링크의 복잡도로서 일반적인 결합도로 사용하는데는 적당치 않다.

Ma[15]는 결합도를 구하는데 호출하는 서비스의 개수뿐만 아니라 호출시 사용하는 메시지의 크기도 상대적인 복잡도를 적용하여 구하였다. 상대적인 복잡도를 구하는데 경험을 사용하여 구하기 때문에 객관적이고 일반적인 값을 제시하지 못하였으며 메시지의 크기까지 고려해야 하는 정당성을 구체적으로 제시하지 못하였다.

Perepletchikov[17]는 서비스를 intra와 extra로 구별하고 구현한 element와 interface 사이의 결합도, incoming 결합도와 outgoing 결합도 등을 구

별하여 구하였다. 요소들 사이의 결합도를 너무 세부적으로 정의하여 전체적인 시스템의 결합도를 파악하기 곤란하다.

기존의 결합도에는 두 가지 문제점이 있다. 첫째, 기존의 결합도는 직접 의존성만 반영하여 서비스(또는 클래스)와 서비스(또는 클래스)에 직접 관련이 안 되어있더라도 관련성이 있는 서비스(또는 클래스)간의 간접 의존성을 무시하여 정확한 결합도를 측정하지 못하였다.

둘째, 기존 결합도는 절대적인 값 즉 서비스(또는 클래스)나 메소드의 개수 등 단순히 구성 요소의 크기만을 나타내므로 시스템 간의 결합도를 상대적으로 비교할 수 없었다. 시스템이 클 경우는 일반적으로 절대적인 결합도의 값이 커지는데 그 시스템이 보다 작은 시스템에서 작은 결합도의 값을 갖는 시스템보다 결합도가 높다라고 말할 수 없다. 시스템간 결합도 비교를 하기 위해서는 절대적인 비교가 아닌 상대적인 비교가 필요하다.

3. 제안하는 SOA 기반 소프트웨어의 결합도

3.1 CBS1과 CBS2

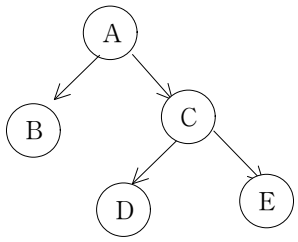
객체지향 시스템의 대표적인 결합도인 CBO를 서비스 지향 아키텍처에 맞게 적용하기 위해서는 클래스 개념을 서비스로 대치하여야 한다.

정의 1 : CBS1(s)
서비스 s에서 직접 호출하는 서비스의 개수

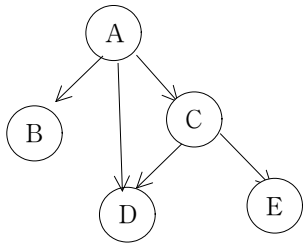
그런데 서비스는 직접 호출하는 것 이외에 간접 호출할 수도 있다. 즉 서비스 A가 서비스 B를 호출하고 서비스 B가 서비스 C를 호출하면 서비스 A는 서비스 C를 간접 호출하게 된다. 이 경우 서비스 A와 서비스 C사이에는 결합력이 있으므로 이런 간접 호출에 대한 것도 고려하여 서비스 A의 결합도를 계산하여야 한다.

예를 들어 [그림 1]과 같이 call graph로 표시할 수 있는 SOA 시스템을 가정하자. call graph에서 노드는 각 서비스를 나타내고 연결선은 서비스 호출을 의미한다. CBS1(A)는 A에서 직접 호출한 B와 서비스 C를 계산하여 2가 되지만 CBS1(C)는 C에서 직접 호출한 D와 서비스 E도 서비스 A가 간접 호출하였으므로 그 결합도도 고려에 넣어야 한다. 그 결합도는 호출하는 서비스와 호출되는 서비스가 멀리 있을수록 작은 값을 가져야 하므로 그 결합도는 두 서비스 사이의 거리가 d라고 할 때 $(\frac{1}{2})^{d-1}$로 한다. 만일 두 서비스 사이의 경로가 두 개 이상 존재하면 최단 경로(shortest path)를 두 서비스 사이의 거리 d로 한다.

CBS1(s)에 간접 호출에 대한 것도 고려하여 CBS2(s)를 제안한다.



[그림 1]



[그림 2]

정의 2 : CBS2(s)

$$CBS2(s) = \sum_{d=1}^{\max} (\frac{1}{2})^{d-1} \cdot n(d)$$

d : 서비스 s에서 간접 호출 가능한 서비스가 있을 때 s에서 그 서비스에 대한 최단 경로

(shortest path)의 길이
n(d) : s에서 길이가 d인 서비스의 개수
max : d의 최대값

그리고 시스템에 있는 모든 서비스에 대한 합계를 그 시스템에 대한 CBS1, CBS2로 나타낸다.

정의 3 : CBS1

$$CBS1 = \sum_{s \in S} CBS1(s),$$

정의 4 : CBS2

$$CBS2 = \sum_{s \in S} CBS2(s)$$

[그림 1]에서 CBS1(A) = 2, CBS1 = 4, CBS2(A) = 3 , CBS2 = 5이다. 한 서비스에서 다른 서비스로의 경로가 2개 이상인 경우의 예로서 [그림 2]로 표시한 시스템에서는 CBS1(A) = 3, CBS1 = 5, CBS2(A) = 3.5, CBS2 = 5.5이다. A에서 D의 호출 경로가 2개 있는데 경로의 길이가 작은 경로를 선택한다.

3.2 RCBS1와 RCBS2

많은 결합도가 클래스나 서비스의 개수등 단순히 구성요소의 크기만을 나타내므로 클래스나 서비스간 또는 서브시스템 간의 결합도를 비교하기가 곤란하다. 가령 시스템이 클 경우 서비스 개수가 많아져 CBS1의 값이 커졌다고 해서 그 시스템이 보다 작은 시스템에서 작은 CBS1의 값을 갖는 시스템보다 결합도가 높다라고 말할 수 없다. 시스템간 비교를 하기 위해서는 상대적인 결합도가 필요하다. 서비스의 개수가 |S|인 시스템에서는 서비스를 호출할 수 있는 최대의 수는 $|S|^2-|S|$이다. CBS1과 CBS2에 대한 상대적인 결합도를 구하기 위해서는 CBS1과 CBS2의 값을 $|S|^2-|S|$로 나누면 된다. 이렇게 되면 그 값은 0에서 1의 값을 갖게 되고 결합도의 상대적인 비교가 가능하게 된다. 다음과 같이 CBS1과 CBS2의 상대적인 결합도 RCBS1와 RCBS2를 다음과 같이 정의한다.

정의 5 : RCBS1

$$RCBS1 = \frac{\sum_{s \in S} CBS1(s)}{|S|^2-|S|}$$

정의 6 : RCBS2

$$RCBS2 = \frac{\sum_{s \in S} CBS2(s)}{|S|^2-|S|}$$

[그림 1]에서 RCBS1 = $\frac{4}{20}$, RCBS2 = $\frac{5}{20}$이고 [그림 2]에서 RCBS1 = $\frac{5}{20}$, RCBS2 = $\frac{5.5}{20}$이다.

4. 제안한 메트릭의 이론적 검증

4.1 객체지향 소프트웨어에 적용한 메트릭의 이론적 검증

소프트웨어 메트릭에 대한 이론적인 검증은 Weyuker의 검증 방법[19]과 Briand[9]가 제안한 방법 등이 있다.

Weyuker의 검증 방법은 처음으로 소프트웨어 메트릭에 대한 이론적 검증을 시도한 것이지만 복잡도(complexity measure)에 관한 검증으로 본 논문이 제안한 결합도에 대한 검증으로서는 부적당하다.

객체 지향 소프트웨어에서 사용하는 메트릭에 대한 이론적 검증은 Briand[9]가 제안한 엄격한 수학적 공리를 주로 사용한다. 이 공리는 복잡도(complexity). 응집도(cohesion), 결합도(coupling) 등 여러 가지 소프트웨어 메트릭을 정의할 때 바람직한 이론적 검증을 제공한다. Briand가 원래 정의한 것은 모듈 시스템에서의 결합도 성질에 관한 것인데 객체지향 시스템에서는 클래스가 모듈의 역할을 하므로 아래에 5가지로 기술한 결합도가 가져야 할 바람직한 성질을 가지고 여러 가지 객체지

향 시스템에 대한 결합도 메트릭의 이론적 타당성을 판단하였다[6, 17].

coupling 1 : Nonnegativity
결합도의 측정 값은 음수가 될 수 없다

coupling 2 : Null Values
클래스들 사이에 상호작용이 없다면 결합도는 0이다.

coupling 3 : Monotonicity
클래스 내부 구성 요소에는 변화 없이 클래스들 사이의 상호작용만 증가한다면 결합도는 감소하지 않는다.

coupling 4 : Merging of Classes
두 클래스를 결합하여 새로운 클래스를 만든다면 새로운 클래스의 결합도는 두 클래스 각각의 결합도의 합보다 증가하지 않는다.

coupling 5 : Disjoint Class Additivity
클래스 간 상호작용이 전혀 없는 두 클래스를 결합하여 새로운 클래스를 만든다면 결합도는 원래 두 클래스의 결합도의 합과 같다.

4.2 제안한 결합도 메트릭의 이론적 조사

본 논문에서는 서비스 간의 연결 강도인 결합도를 측정할 수 있는 메트릭을 정의하였다. 따라서 제안한 메트릭의 이론적 타당성을 검증하기 위하여 Briand가 정의한 결합도가 가져야 할 바람직한 성질을 만족하는지 여부를 살펴보았다. SOA 기반 소프트웨어에 Briand가 적용한 기준을 적용하기 위하여 객체지향 소프트웨어에서 적용한 5가지 성질에서 클래스 개념을 서비스 개념으로 대치하여 조사하였다. Briand 자신의 연구에서도 밝혔듯이 바람직한 메트릭이 모든 이론적 성질을 만족할 필요는 없다[10].

coupling 1 : Nonnegativity
CBS1과 CBS2는 0이상이고 |S|는 양수이므로 CBS1과 CBS2, RCBS1과 RCBS2는 모두 음수가 될 수 없다.

coupling 2 : Null Values
만약 서비스들 간에 호출이 없다면 메소드 호출 타입에 따른 메소드 호출 수 CBS1과 CBS2는 0이다. 그러면 상대적인 결합 메트릭 RCBS1과 RCBS2도 0이 된다.

coupling 3 : Monotonicity
서비스 내부 구성 요소에는 변화 없이 서비스들 사이의 상호작용만 증가한다면 서비스 간에 메시지 호출 수가 증가하는 것이므로 CBS1과 CBS2도 증가한다.

$$\text{RCBS1} = \frac{\sum_{s \in S} CBS1(s)}{|S|^2 - |S|}$$ 과

$$\text{RCBS2} = \frac{\sum_{s \in S} CBS2(s)}{|S|^2 - |S|}$$ 도 모두 분자가 증가하므로 RCBS1과 RCBS2도 증가한다.

coupling 4 : Merging of Services
두 서비스 S1과 S2에 대하여, 두 서비스를 합한 새로운 서비스를 S라고 하자. 그러면 다음과 같은 수식이 성립한다.

$$\begin{aligned} |CBS1(S)| &= |CBS1(S1)| + |CBS1(S2)| \\ &\quad - |CBS1(S1 \cap S2)| \\ &\leq |CBS1(S1)| + |CBS1(S2)|) \end{aligned}$$

CBS2도 마찬가지로 이 성질을 만족한다.

그러나 RCBS1이나 RCBS2에 대해서는 |S|의 값이 1 감소하여 새로운 서비스의 분자, 분모값이 모두 감소하므로 이 성질을 항상 만족하지는 않는다.

coupling 5 : Disjoint Service Additivity
두 서비스 S1과 S2에 대하여, 두 서비스를 합한 새로운 서비스를S 라고 하자. 그러면 다음과 같은 수식이 성립한다.

$$|CBS1(S)| = |CBS1(S1)|+|CBS1(S2)| - |CBS1(S1 \cap S2\ )|$$

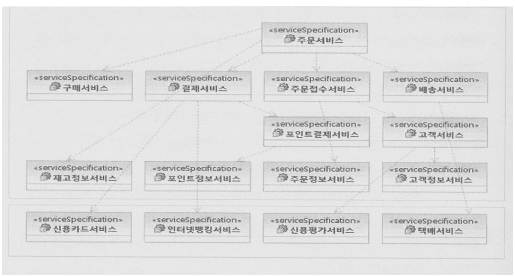
그런데 $|CBS1(S1 \cap S2)| = 0$이므로 $|CBS1(S)| = |CBS1(S1)|+|CBS1(S2)|$이 성립한다. CBS2도 마찬가지로 이 성질을 만족한다.

그러나 RCBS1이나 RCBS2에 대해서는 |S|의 값이 1 감소하므로 이 성질을 만족하지 않는다.

요약하면 주로 객체지향 시스템에서 사용된 Briand등이 제안한 메트릭에 대한 공리를 SOA 기반 소프트웨어에 관하여 본 논문에서 제안한 메트릭에 대해 조사하였더니 호출수만 계산하는 CBS1과 CBS2은 모든 조건을 만족하였지만 상대적 결합도를 측정하는 RCBS1과 RCBS2는 coupling 4와 coupling 5를 만족하지 못하였다.

5. 사례연구

제안한 결합도 메트릭을 온라인 쇼핑몰 주문 시스템[5]에 적용하였다. 이 시스템의 서비스 모델은 [그림 3]에 나타내었다. 각 서비스에 대하여 결합도 계산시 편리를 위하여 서비스마다 기호를 <표 1>과 같이 정의하였다.



[그림 3] 온라인 쇼핑몰 주문 시스템의 서비스 모델

〈표 1〉 서비스, 기호 정의표

서비스 명	주문 서비스	구매 서비스	결제 서비스	주문접수 서비스
기호	A	B	C	D

서비스 명	배송 서비스	포인트 결재 서비스	고객서비스
기호	E	F	G

서비스 명	재고정보 서비스	포인트정보 서비스	주문정보 서비스	고객정보 서비스
기호	H	I	J	K

서비스 명	신용카드 서비스	인터넷뱅킹 서비스	신용평가 서비스	택배 서비스
기호	L	M	N	O

서비스 A에 대한 각 결합도 메트릭을 다음과 같이 계산한다. A에서 직접 호출 가능한 서비스는 B, C, D, E, H이므로 CBS1(A) = 5이다. CBS2(A)를 구하기 위해서 서비스 A에서 간접 호출 가능한 서비스를 찾아야 한다. A에서 최단 경로가 2인 간접 호출 서비스는 C에서 호출한 F, H, L과 D에서 호출한 G, J와 E에서 호출한 O가 있다. A에서 최단 경로가 3인 간접 호출 서비스는 G에서 호출한 N이유일하다. 그러므로 CBS2(A) = 5+0.5×6+0.25×1 = 8.25가 된다. 다른 서비스에 대한 CBS1과 CBS2 값도 동일한 방식으로 구하고 모든 서비스에 대한 합계가 CBS1과 CBS2가 된다. 서비스 개수가 모두 15개 이므로

$$\text{RCBS1} = \frac{\sum_{s \in S} CBS1(s)}{15^2 - 15},$$

$$\text{RCBS2} = \frac{\sum_{s \in S} CBS2(s)}{15^2 - 15}$$

이고 이들의 값을 계산한 결과를 <표 2>에 제시하였다.

결과를 보면 직접 의존성만 고려한 결합도 CBS1보다 간접 의존성까지 고려한 CBS2는 당연히 더 큰 값을 가지며 상대적인 결합도의 값은 0과 1사이의 값을 가진다. 상대적인 결합도를 다른 시스

템이나 현 시스템을 수정할 경우 수정한 시스템 결합도와 비교하여 어느 시스템이 결합도 측면에서 더 바람직한가를 판단할 수 있다.

〈표 2〉 온라인 쇼핑몰의 주문 시스템에 대한 결합도 메트릭 값

메트릭	CBS1	CBS2	RCBS1	RCBS2
값	14	19.25	0.067	0.092

6. 결 론

본 논문은 새로이 각광받기 시작한 SOA 기반 소프트웨어에 대한 결합도 메트릭을 제안하였다.기존 객체지향 기반의 결합도 메트릭에서 가장 많이 사용하는 CBO를 SOA에 맞게 수정한 CBS1과 직접 호출 외에 간접 호출까지 고려한 결합도 CBS2를 제안하였다. 또한 CBO, CBS 등은 서비스 간 또는 서브 시스템 간 결합도의 상대적인 평가를 할 수 없으므로 상대적인 결합도 메트릭인 RCBS1과 RCBS2를 제안하였다. 그리고 본 논문에서 제안한 결합 메트릭이 이론적으로 바람직하다는 것을 검토하기 위하여 Briand 등이 제안한 결합도 성질에 따라 조사하였다. 마지막으로, 제안한 결합 메트릭의 실용성을 검증하기 위하여 적용 사례를 제시하였다. 새로운 메트릭의 제안으로 SOA 기반 소프트웨어의 결합도를 보다 정확하게 또한 상대적으로 평가할 수 있게 되었다.

앞으로의 연구과제는 이 메트릭을 사용하여 SOA 기반 소프트웨어의 유지보수, 리팩토링, 시스템 분해 등에 효율적으로 사용될 수 있음을 실제 사례를 통하여 성능을 실증적으로 검증하는 것이다.

참 고 문 헌

[1] 고현희, 궁상환, 박재년, “아키텍처 기반 설계 방식에 대한 평가기능이 통합된 소프트웨어 설계 방법론”, 「정보과학회논문지 : 소프트웨어 및 응용」, 제l34권, 제7호(2007), pp.625-634.

[2] 권수갑, “SOA 개념과 동향”, 「방송과 기술」, 제121권(2006), pp.110-121.

[3] 박성희, 홍의석, 우치수, 김태균, “객체 지향 프로그램에서 응집도, 결합도 측정 메트릭 집합”, 「정보과학회논문지(B)」, 제25권, 제12호(1998), pp.1779-1787.

[4] 이종석, 우치수, “객체 지향 시스템에서의 클래스 응집도와 결합도 메트릭”, 「정보과학회논문지 : 소프트웨어 및 응용」, 제27권, 제6호(2000), pp.595-606.

[5] 전병선, 「SOA, What and How」, 와우북스, 2008

[6] 최미숙, 이종석, 이서정, “효율적인 시스템 설계를 위한 클래스 간의 결합 척도”, 「인터넷정보학회논문지」, 제9권, 제5호(2008), pp.85-97.

[7] 화지민, 이숙희, 권용래, “객체 지향 시스템에서의 클래스 간 의존성 강도 측정을 위한 커플링 척도”, 「정보과학회논문지 : 컴퓨팅의 실제 및 레터」, 제14권, 제1호(2008), pp.81-85.

[8] Arsanjani, A., “Service-oriented modeling and architecture”, *IBM Developer Works*, 2004.

[9] Briand, L. C., S. Morasca, and V. R. Basili, “Property-based software engineering measurement”, *IEEE Trans. Software Eng.*, Vol. 22, No.1(1996), pp.68-86.

[10] Briand, L. C. and J. Wuest, “Empirical Studies of Quality Models in Object- Oriented Systems”, *Advances in Com puters*, Vol.59 (2002), pp.97-166.

[11] Chidamber, S. R. and C. F. Kemerer, “A Metrics Suite for Object Oriented Design”, *IEEE Tr. on SE*, Vol.20, No.6(1994), pp.476-493.

[12] Gui, G. and P. Scott, “Coupling and cohesion measures for evaluation of component reusability,” *Proceedings of the 2006 inter-*

national workshop on Mining software repositories(2006), pp.18-21.

[13] Hirzalla, M., J. Cleland-Huang, and A. Arsanjani, "A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures", *Service-Oriented Computing-ICSOC 2008 Workshops*, (2009), pp. 41-52.

[14] Huynh, Q. T., T. Q. Pham, and Q. V. Tran, "The Reusability and Coupling Metrics for Service Oriented Softwares", 2007.

[15] Ma, Q., N. Zhou, Y. Zhu, and H. Wang, "Evaluating service identification with design metrics on business process decomposition", *IEEE International Conference on Services Computing*(2009), pp.160-167.

[16] Perepletchikov, M., C. Ryan, and K. Frampton, "Cohesion metrics for predicting maintainability of service-oriented software", *Qsic*, (2007), pp.328-335.

[17] Perepletchikov, M., C. Ryan, K. Frampton, and Z. Tari, "Coupling Metrics for Predicting Maintainability in Service-Oriented Designs", *presented at 18th Australian Conference on Software Engineering*, Melbourne, Australia(2007), pp.329-340.

[18] Washizaki, H., T. Nakagawa, Y. Saito, and Y. Fukazawa, "A coupling-based complexity metric for remote component based software systems toward maintainability estimation", In *APSEC Proceedings*, Washington, DC, USA, IEEE Computer Society (2006), pp.79-86.

[19] Weyuker, E. J., "Evaluating software complexity measures", *IEEE Trans. Softw. Eng.*, Vol14, No.9(1988), pp.1357-1365.

◈ 저 자 소 개 ◈

유 문 성 (msyoo@sangji.ac.kr)

현재 상지대학교 컴퓨터정보공학부 교수로 재직 중이며 서울대학교에서 수학으로 이학사(BS)를, 인디아나 대학교 컴퓨터학과에서 석사(MS)를, 루이지애나대학교 컴퓨터학과에서 컴퓨터학 박사(Ph.D.)를 취득하였다. 석사 취득 전, 현대건설 전산실에서 프로그래머 및 분석가로 근무하였고 한국개발연구원(KDI) 전산실에서 계량 경제 분석 관련 선임 연구원 및 소프트웨어 개발자로 근무하였다. 주요 연구관심분야는 객체지향시스템, 소프트웨어 공학, 소프트웨어개발, 유비쿼터스 컴퓨팅 등이다.