http://dx.doi.org/10.3745/KIPSTD.2012.19D.1.049

클라우드 서비스의 재사용성 평가 및 향상 기법

오 상 헌[†] · 라 현 정[††] · 김 수 동[†††]

요 약

클라우드 컴퓨팅에서 서비스 제공자들은 다양한 어플리케이션들 중에서 재사용 특징과 공통성을 고려하여 개발하고 배포하며 서비스 사용자들은 어플리케이션을 구축하기 위해 서비스를 찾고 재사용한다. 그러므로 재사용성은 클라우드 서비스의 중요한 본질적인 특성이다. 서비스의 재사용성이 높으면, 투자대비 높은 수익을 올릴 수 있다. 클라우드 서비스는 기존의 소프트웨어 재사용성을 위한 품질모델은 전통적인 프로그래밍 패러다임에 나타나지 않는 특성을 가지고 있기 때문에 서비스 평가에 적용하기 어렵다. 본 논문에서는 클라우드 서비스 재사용성을 평가하기 위해 품질속성과 메트릭이 포함된 평가모델을 제시한다. 사례연구에서는 평가모델을 적용한 사례를 보여준다.

키워드 : 클라우드 서비스, 재사용성, 품질속성, 메트릭

Method to Evaluate and Enhance Reusability of Cloud Services

Sang Hun Oh[†] · Hyun Jung La[††] · Soo Dong Kim[†††]

ABSTRACT

In cloud computing, service providers develop and deploy services with common and reusable features among various applications, service consumers locate and reuse them in building their applications. Hence, reusability is a key intrinsic characteristic of cloud services. Services with high reusability would yield high return-on-investment. Cloud services have characteristics which do not appear in conventional programming paradigms, existing quality models for software reusability would not applicable to services. In this paper, we propose a reusability evaluation suite for cloud services, which includes quality attributes and metrics. A case study is presented to show its applicability.

Keywords : Cloud Services, Reusability, Quality Attributes, Metrics

1. 서 론

클라우드 컴퓨팅은 클라우드 서비스와 어플리케이션을 구축하기 위해 효과적인 패러다임으로 떠오르고 있다. 다양한 자원의 타입은 서비스로 배포될 수 있으며 많은 장점들을 제공한다[1]. 클라우드 컴퓨팅에서 서비스 제공자들은 다양한 어플리케이션들 중에서 재사용 특징과 공통성을 고려하여 개발하고 배포한다. 서비스 사용자들은 자신들의 어플리케이션을 위해 적합한 서비스를 찾고 서비스 구독을 통한 사용자들의 어플리케이션 구축을 위해 서비스를 재사용한다. 이러한 이유로 재사용성은 서비스의 품질평가에서 중요한 기준 중에 하나이다.

※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2009-0076392).
† 준 회 원 : 숭실대학교 컴퓨터학과 박사수료
†† 정 회 원 : 숭실대학교 모바일 서비스 소프트웨어공학센터 연구교수
††† 종신회원 : 숭실대학교 컴퓨터학부 교수(교신저자)
논문접수 : 2011년 8월 29일
수 정 일 : 1차 2011년 11월 4일, 2차 2011년 11월 21일
심사완료 : 2011년 11월 21일

재사용성은 재사용 할 수 있는 정도를 의미한다. 기존의 품질속성과 재사용성의 메트릭들은 CBD에서 컴포넌트와 객체지향 프로그램, 절차적 프로그램에 대부분 사용되었다[2]. 클라우드 컴퓨팅에서 서비스는 다른 프로그래밍 패러다임에서 나타나지 않는 고유한 특성을 가지고 있기 때문에 클라우드 서비스를 평가하는 품질모델은 이러한 점이 고려되어야 한다.

클라우드 서비스의 재사용성은 두 가지의 다른 관점으로 측정된다. 서비스 제공자 측면에서의 재사용성은 서비스 사용량과 수익에 대한 측정이다. 서비스 사용자 측면에서의 재사용성은 사용자들의 어플리케이션을 위해 서비스 사용과 적응, 그리고 발견할 수 있는 정도에 대한 측정이다.

본 논문에서 클라우드 서비스 재사용성 평가 모델은 서비스 사용자 측면을 고려하여, 2장에서는 관련연구를 작성하였고, 3장에서는 클라우드 서비스의 특징들을 요약하였다. 4장에서는 재사용성을 위한 품질속성을 5장에서는 품질속성과 관련된 메트릭을 설명하고, 6장에서는 제시한 재사용성 모델을 적용한 사례연구의 결과를 나타낸다. 그리고 7장에

서는 측정된 재사용성을 향상시킬 수 있는 가이드라인을 제시하며, 8장에서는 제시한 품질모델에 대해 분석과 평가를 한다.

제안된 평가모델은 클라우드 서비스 품질평가에 실용적으로 적용할 수 있고 정량적인 방법으로 재사용성 향상에 적용할 수 있고 생각한다.

2. 관련 연구

ISO/IEC 9126은 품질모델의 세가지 타입을 제공한다; 내부 메트릭, 외부메트릭, 그리고 사용품질 메트릭 [3][4]. 현재 제품을 평가하기 위한 표준인 ISO/IEC 9126은 정확하게 클라우드 서비스의 재사용성 평가에 적용하기에는 부족하다. 또한 다른 관련연구로는 단일시스템이나 컴포넌트 평가에 초점을 맞추고 있다. 그러므로, 전통적인 품질모델 ISO/IEC 9126을 클라우드 서비스 재사용성 평가를 위해 바꾸거나 클라우드 서비스의 재사용성을 평가하기 위한 새로운 프레임워크를 정의해야 한다.

Choi의 연구는 사용자 관점을 특별히 고려하여 제공될 서비스의 품질평가를 위한 메트릭을 제안하였다[5]. 이 연구에서는 품질메트릭의 가용성과 유용성, 품질속성의 모델 구조는 SOC(Service Oriented Computing)의 특성으로부터 도출되었다. SOC에서 말하는 서비스는 기업이나 조직의 내의 소프트웨어 자산에 대해 다른 소유권 통제 하에 있을 수 있는 분산된 역량을 구성하고 활용하기 위한 아키텍처 패러다임이다. 즉, 소프트웨어의 소유권이 있으므로 서비스의 수정 및 변경, 유지보수 등을 모두 수행해야 한다. 또한 기업이나 특정 조직 내에서만 사용할 수 있는 서비스이기 때문에 클라우드 서비스 보다 재사용성이 낮다. 반면 클라우드 서비스는 잠재적인 서비스 사용자들을 위해 공통적이고 재사용할 수 있는 특징들을 제공한다. 또한 서버, 저장장치, 소프트웨어 등의 자원을 구독하므로 소유하지 않고 필요 시 언제 어디서든 인터넷을 통해 서비스를 이용하는 방식이다. 클라우드 서비스 제공자는 가상화 기술을 바탕으로 분산된 자원을 통합하여 확장성 높은 서비스를 제공한다. 서비스 사용자는 인터넷을 이용하여 다양한 형태의 단말기를 이용하여 서비스를 이용할 수 있다. 또한 SOC의 서비스는 비즈니스 수준에서 서비스를 재사용하기 때문에 서비스의 규모나 복잡성이 높아 많은 컴퓨팅 자원을 필요로 하고 있다. 그러나 클라우드 서비스는 제공자 측에서 실행되며 서비스 사용자는 실행 결과를 브라우저나 경량 클라이언트 응용 프로그램을 통해 이루어지기 때문에 모바일 환경에서도 서비스를 이용할 수 있다. 따라서 클라우드 서비스의 중요한 특성 일부분은 SOC에 존재하지 않기 때문에 이 연구는 클라우드 서비스 품질측정에 적용할 수 없다.

Washizaki's의 연구는 소스코드 없이 컴포넌트의 외부로부터 얻을 수 있는 정보를 기반으로 블랙박스 컴포넌트의 재사용성을 측정하기 위한 메트릭을 제안하였다[6]. 이 연구는 컴포넌트 사용자 관점으로부터 블랙박스 컴포넌트를 위한 재사용성 모델을 정의하였다. 이 재사용성 모델에서, 컴포넌트 재사용성은 3개의 요소로 분리된다: 이해성(Understandability), 적응성(Adaptability), 이식성(Portability). 이 요소들은 다시 4개의 기준과 5개의 메트릭으로 나뉜다. 4개의 기준은 메타 정보의 존재여부, 식별 가능성, 커스터마이즈, 외부와의 의존도이다. 5개의 메트릭, *Existence of Meta-Information (EMI), Rate of Component Observability (RCO), Rate of Component Customizability (RCC), Self-Completeness of Component's Return Value (SCCr), and Self-Completeness of Component's Parameter (SCCp)*은 자바 빈(JavaBeans) 컴포넌트의 수를 통계 분석하여 신뢰 구간을 정의하였다. 또한 재사용성 모델을 기반으로 한 5개의 메트릭들을 결합하여 재사용성 메트릭을 제공한다. 그러나, 클라우드 서비스는 컴포넌트 보다 큰 재사용성 단위이다. 그러므로 품질요소와 메트릭은 클라우드 서비스의 재사용성을 평가하기 위해서 확장되어야 한다.

3. 클라우드 서비스의 특징

품질모델은 목표 컴퓨팅 패러다임의 고유한 특성을 고려하여 정의된다. 그러므로 품질모델을 정의하기 위해서는 클라우드 서비스의 핵심 특성을 식별하는 것이 전제조건이다[7]. 특성은 두 개 분야로 나뉜다: 본질적인 것과 서비스의 가치 있는 속성.

- 공통성: 클라우드 서비스는 잠재적인 서비스 사용자들을 위해 공통적이고 재사용 할 수 있는 특징들을 제공해야 한다. 서비스의 공통성이 높을수록 높은 ROI(Return on Investment)을 낼 수 있다.
- 인터넷을 통합 접속: 원격 노드에 배포된 클라우드 서비스는 인터넷을 통해 사용자들이 접속을 할 수 있다[1][8]. 따라서 표준 프로토콜을 준수하는 사용자들은 특정 소프트웨어의 설치 및 운영체제 요구사항 없이 클라우드 서비스에 접근 할 수 있다.
- 잘 정의된 인터페이스: 클라우드 서비스는 내부 정보를 노출하지 않으면서 서비스의 인터페이스를 통해 접근할 수 있다[1][9].
- 사용에 대한 비용지불: 전통적인 어플리케이션과는 달리, 클라우드 서비스는 서비스 사용자가 영구적으로 소유할 수 없다. 즉, 서비스 사용자들은 서비스를 구독해야 하고, 사용량에 따라 비용을 지불해야 한다[1][10].
- 가벼운 클라이언트 모델: 클라우드 서비스는 제공자 측에서 실행되며 서비스 사용자는 실행 결과를 브라우저나 경량 클라이언트 응용 프로그램을 통해서 확인 할 수 있다. 또한 사용자의 특정 데이터 셋을 저장하고 서비스 제공자 측면에서 유지보수 된다. 즉, 실행과 저장 그리고 유지보수는 서비스 제공자 측에서 모두 이루어지며, 소비자는 서비스 실행 결과만을 브라우저나 경량 클라이언트 응용 프로그램을 통해 확인 한다.

- 높은 가용성(Availability): 클라우드 서비스는 높은 가용성이 보장되어야 한다. 서비스의 가용성이 낮다면 서비스 이용 시 불편함과 사용자에게 부정적인 영향을 줄 수 있다[1][8][9]. 따라서 클라우드 컴퓨팅은 강력한 서버 클러스터와 안정적인 네트워크를 제공하는 강력한 컴퓨팅 아키텍처를 제공해야 한다.
- 높은 확장성(Scalability): 클라우드 컴퓨팅에서, 서비스 요청에 대한 양. 즉, 서비스 로드는 동적이고 예측할 수 없다. 클라우드 서비스는 서비스 요청이 절정(Peak Time)일 때, 높은 확장성을 제공할 수 있어야 한다. 서비스의 확장성이 낮다면, 서비스 요청이 절정일 때 서비스 사용자들은 많은 어려움을 겪고 이로 인해 서비스의 이용이 점차 낮아진다[1][11].

(그림 1)에서 위의 7개의 클라우드 서비스의 특징은 많은 문헌들을 통해 연구된 범용적인 특징들이며, 이러한 범용적인 특징을 기반으로 클라우드 서비스 재사용성을 측정하기 위한 품질속성들을 도출한다.

“잘 정의된 인터페이스”는 이해성과 연관성이 있다. 클라우드 서비스는 내부 정보를 노출하지 않으며 서비스의 인터페이스를 통해 접근할 수 있다. 즉, 서비스의 디스크립션에 인터페이스가 잘 정의 되어 있지 않다면 클라우드 서비스 사용자들은 디스크립션을 이해하지 못한다. 따라서 인터페이스가 잘 정의되어 있지 않다면 이해성이 낮아지기 때문에 전체적으로 클라우드 서비스의 재사용성이 낮아지므로 “잘 정의된 인터페이스”는 이해성과 밀접한 관계가 있다.

“공통성”과 “잘 정의된 인터페이스”는 홍보성과 연관성이 있다. 클라우드 서비스를 사용하는 대상이 누구인지 서비스 제공자는 알 수가 없다. 따라서 잠재적인 서비스 사용자들을 위해 공통적이고 재사용 할 수 있는 특징들을 많이 제공해야만 잠재적인 여러 사용자들이 서비스를 발견할 수 있다. 또한 서비스의 인터페이스가 잘 정의되어 있어야만 잠재적인 사용자들이 서비스를 발견하고 쉽게 이용할 수 있다.

“공통성”, “잘 정의된 인터페이스”와 “높은 확장성”은 적응성과 연관성이 있다. 서비스의 공통성과 잘 정의된 인터페이스를 통해 서비스 사용자들은 쉽게 커스터마이즈를 할 수 있다. 높은 확장성을 통해 클라우드 서비스가 서비스 요청이 절정일 때 다양한 옵션들을 제공할 수 있어야 한다. 만약 서비스 제공자가 서비스를 배포할 때 다양한 옵션들을 제공하지 않는다면 서비스 사용자들은 많은 어려움을 겪고 이를 통해 서비스의 재사용성은 낮아진다.

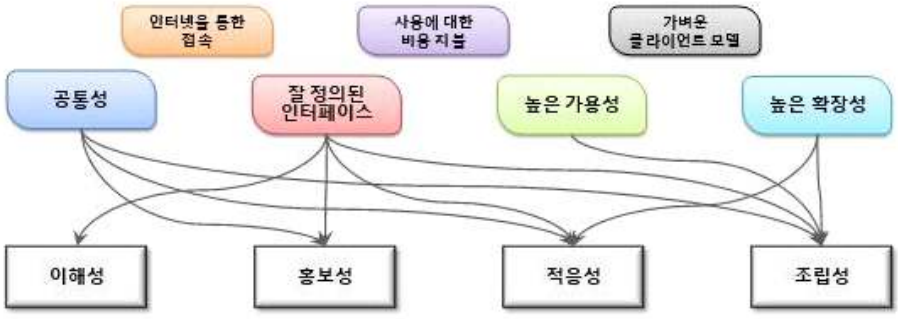
“공통성”, “잘 정의된 인터페이스”, “높은 가용성”, 그리고 “높은 확장성”은 조립성과 연관성이 있다. 공통성과 잘 정의된 인터페이스 그리고 높은 확장성이 잘 제공되어야 여러 어플리케이션들간의 인터페이스 공통성과 가변성 분석을 통하여 공통된 인터페이스 항목을 수정되지 않도록 고정적으로 설계하고, 가변적인 인터페이스 항목은 다시 가변점(Variation Point)와 각 가변점 별 가변치(Variants)를 반영하여 어플리케이션 별로 필요한 인터페이스로 특화 할 수 있도록 설계가 될 수 있다. 그리고 높은 가용성은 여러 서비스들이 연결되어 필요한 비즈니스 프로세스를 만들기 때문에 서비스 이용이 항상 가용해야 한다.

4. 서비스 재사용성을 위한 품질속성

본 장에서는 서비스 재사용성을 위한 품질속성을 정의한다. 품질모델은 실용적으로 널리 적용할 수 있고 신뢰할 수 있어야 한다. 본 논문에서는 McCall의 FCM (Factor Criteria Metrics)[12] 접근방법을 적용하며, 품질모델은 하향방식(Top-Down)으로 세가지 수준으로 구성되어 있다. 3가지 용어(요소, 기준, 메트릭)는 평가 및 평가 계층구조로 구성되어 있다. 요소는 관리수준에서 사용되며 기준은 디자인 수준에서 사용되고 메트릭은 제품 수준에서 사용된다.

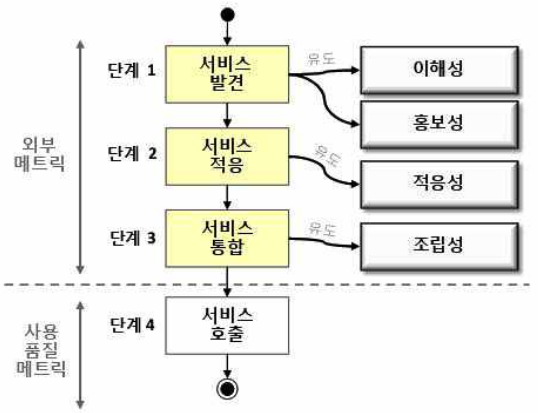
일반적이고 잘 적용될 수 있는 품질모델은 클라우드 서비스 재사용성의 일반적인 프로세스를 고려해야 한다. 본 논문에서는 [13][14][15][16]로부터 서비스 재사용 프로세스 자원들을 유도하여 (그림 2)와 같은 일반적인 서비스 재사용 프로세스를 정의하였다. 유도된 서비스 재사용 프로세스는 기존 연구들을 통해 범용적으로 사용되는 프로세스이기 때문에 본 논문에서는 유도된 서비스 재사용 프로세스를 클라우드 서비스 환경에 적용하여 서비스의 재사용성을 평가하기 위한 품질속성을 도출한다.

유도된 서비스 재사용 프로세스는 서비스 발견, 적응, 통합 그리고 호출이라는 4개 단계로 되어 있으며, 클라우드 서비스의 높은 재사용성을 위해 서비스는 여러 사용자들에



(그림 1) 클라우드 서비스의 특징과 품질속성들간의 관계

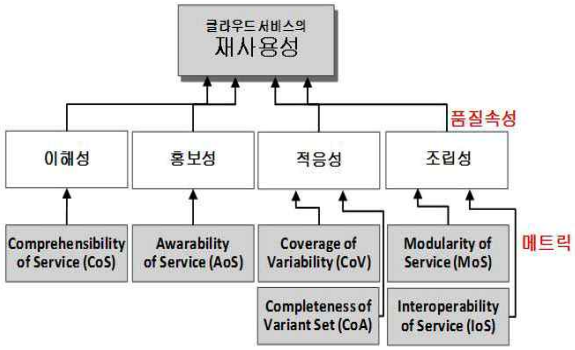
게 잘 발견되어야 하고 그러기 위해서는 서비스 디스크립션의 이해와 서비스가 잘 발견될 수 있도록 서비스의 신택틱, 시맨틱한 정보기술이 서비스 명세서에 기술되어 있어야 한다. 따라서 본 논문에서는 클라우드 서비스 재사용성 평가를 위해 이해성과 홍보성을 품질속성으로 도출하였고 이해성과 홍보성을 측정할 수 있는 메트릭들을 제안하였다. 또한 서비스 기반 어플리케이션의 목적에 맞도록 서비스를 효과적으로 적용할 수 있어야 한다. 즉, 다양한 기능이 포함된 도메인 분석 보고서와 디자인 명세서가 기술되어야 한다. 따라서 본 논문에서는 클라우드 서비스 재사용성 평가를 위해 적응성을 품질속성으로 도출하였고 이를 측정할 수 있는 메트릭을 제안하였다. 그리고 서비스들 간의 통합은 적응된 서비스와 목표 어플리케이션의 통합을 통해 비즈니스 프로세스를 구성하며 하나 이상의 서비스는 어플리케이션에 통합될 수 있어야 한다. 즉, 다른 서비스와 상호작용할 수 있는 오퍼레이션의 정보 등이 제공되어야 한다. 따라서 본 논문에서는 클라우드 서비스 재사용성 평가를 위해 조립성을 품질속성으로 도출하였고 이를 측정할 수 있는 메트릭을 제안하였다.



(그림 2) 재사용 프로세스와 유도된 품질 속성들

따라서, (그림 2)에서 단계 1은 목표 어플리케이션에 적용할 수 있는 후보 서비스들을 발견한다. 일반적으로 서비스 디스크립션 관찰을 통해 발견할 수 있고, 따라서 서비스 디스크립션은 이해성이 높아야 한다. 단계 1의 활동을 고려하여 본 논문에서는 두 개의 품질속성을 도출하였다: 이해성(Understandability)과 홍보성(Publicity). 단계 2는 어플리케이션에서 예상 가능한 서비스와 가용한 서비스 사이에서 부분적으로 일치한다면 목표 어플리케이션에 맞게 후보 서비스를 맞춘다[13]. 단계 2의 활동을 고려하여 본 논문에서는 품질속성으로 적응성(Adapatbility)을 도출하였다. 단계 3은 적응된 서비스와 목표 어플리케이션의 통합을 통해 비즈니스 프로세스를 구성하며 클라우드 컴퓨팅에서 하나 이상의 서비스는 어플리케이션에 통합될 수 있다. 단계 3의 활동을 고려하여 본 논문에서는 품질속성으로 조립성(Composability)을 도출하였다. 단계 4는 실시간에 서비스를 호출하면 목표 어플리케이션이 실행된다. 이 단계는 사용중 품질을 위한 속성이기 때문에, 본 논문에서는 단계 4에 대해서 고려하지 않는다.

식별된 품질속성들의 값을 계산하기 위한 메트릭을 정의한다. (그림 3)은 품질속성과 메트릭의 전체적인 구조를 보여준다.



(그림 3) 재사용성을 위한 품질속성과 메트릭

(그림 3)에서 이해성은 서비스 디스크립션의 품질을 측정하기 위해 *Comprehensibility of Service(CoS)*을 통해 측정된다. 홍보성은 새롭게 등록된 서비스를 서비스 사용자가 얼마나 효과적으로 인식할 수 있는지를 측정하기 위해 *Awarability of Service(AoS)*을 통해 측정된다. 적응성은 두 개의 메트릭을 통해 측정된다; *Coverage of Variablity (CoV)*와 *Compleness of Variant Set(CoA)*. 두 메트릭은 가변성과 커스터마이즈의 범위를 측정하는데 사용된다. 조립성은 두 개의 메트릭을 통해 측정된다: Modularity of Service (MoS)와 Interoperability of Service (IoS). MoS는 서비스의 응집성을 측정하고, IoS는 다른 서비스들과의 협력의 정도를 측정한다.

5. 서비스 재사용성을 위한 품질 메트릭

본 장에서는 서비스 재사용성의 품질속성을 위한 메트릭을 정의한다. 각 메트릭은 개요, 계산을 위한 공식, 그리고 설명을 정의한다.

5.1 이해성을 위한 메트릭

서비스는 일반적으로 서비스의 신택틱과 시맨틱한 특징을 명세한 디스크립션을 제공한다. 그리고, 서비스 디스크립션은 많은 설명 필드를 가지고 있다. 예를 들어, 서비스를 위한 WSDL(Web Services Description Language) 디스크립션은 표준에서 정의된 많은 XML 태그들을 포함한다.

이해성은 Comprehensibility of Service (CoS)로부터 측정되며 받아들일 수 있는 가독성(Readability) 필드의 수를 측정한다. CoS는 다음과 같이 계산된다;

$$CoS = \left(\frac{Number\ of\ Fields\ with\ Acceptable\ Readability}{Total\ Number\ of\ Fields} \right)$$

분모는 서비스 디스크립션에서 사용되는 전체 필드의 수이며 분자는 받아들일 수 있는 가독성 필드의 수를 의미한다. 즉, 필드의 이해성을 의미한다. 따라서 값의 범위는 0...1이다. CoS의 값이 높을 수록 서비스는 높은 이해성을 가진다.

5.2 홍보성을 위한 메트릭

클라우드 컴퓨팅에서, 서비스 제공자에 의해 등록된 서비스는 서비스 제공자에 의해 등록된 서비스는 서비스 저장소에 등록되며 등록된 서비스들은 서비스 발견을 위해 공개된다. 서비스의 홍보성이 낮다면 서비스들을 효과적으로 발견할 수 없으며, 이로인해 낮은 구독 결과가 발생한다.

홍보성은 *Awarability of Service (AoS)*을 통해 측정되며, AoS는 다음과 같이 계산된다;

$$AoS = \left(\frac{Number\ of\ Consumers\ with\ Service\ Discoverability}{Total\ Number\ of\ Potential\ Consumers} \right)$$

분모는 잠재적인 서비스 사용자들의 전체 수 이고 분자는 기존에 존재하는 서비스들을 UDDI(Universal Description Discovery, and Integration) 발견 기법과 서비스 분류 트리 탐색등 다양한 방법으로 등록된 서비스를 발견할 수 있는 사용자들의 수 이다.

따라서 값의 범위는 0...1이다. AoS의 값이 높을수록 서비스를 보다 쉽게 사용자들이 서비스를 이용할 수 있다.

5.3 적응성을 위한 메트릭

서비스의 적응성은 각 서비스 기반 어플리케이션의 목적을 위해 효과적으로 서비스를 적응할 수 있는 정도를 의미한다. 적응성은 두 개의 메트릭으로 측정된다; *Coverage of Variability (CoV)*와 *Completeness of Variant Set (CoA).*

CoV는 얼마나 많은 곳에서 서비스 사용자들이 커스터마이즈를 할 수 있는 정도를 측정한다. 즉 서비스가 얼마나 많은 가변점을 지원할 수 있는지를 의미한다. CoV는 다음과 같이 계산된다;

$$CoV = \left(\frac{Number\ of\ Variation\ Point\ Supported}{Total\ Number\ of\ Potential\ Variation\ Points} \right)$$

분모는 클라이언트 어플리케이션들의 집합을 예상하고 고려한 잠재적인 가변점들의 전체 수 이고, 분자는 서비스에서 제공하는 가변점들의 수 이다. 서비스는 블랙박스 형태로 제공되기 때문에 서비스 사용자들이 잠재적인 가변점의 존재 여부를 찾을 때 사용할 수 있는 유일한 방법은 서비스 명세서(Service Specification)이다. 서비스를 명세하기 위한 범용적인 표준은 존재하지 않지만 서비스 인터페이스(Service Interface), 서비스 기능성(Service Functionality), 서비스 품질(Quality of Service)와 같은 서비스 명세를 위한 중요 항목에 대한 공통적인 견해가 존재한다.

CoA는 얼마나 많은 옵션을 서비스가 제공할 수 있는지를 측정한다. 즉, 서비스 사용자는 목적을 위해 제공되는 옵션들 중에서 하나를 선택할 수 있다. CoA는 다음과 같이 계산된다;

$$CoA = \frac{\sum_{i=1}^{n} \left(\frac{Number\ of\ Variants\ Supported\ in VP_i}{Total\ Number\ of\ Potential\ Variants\ in VP_i} \right)}{n}$$

n은 서비스에서 제공되는 가변점들의 수 이고 분모는 각 가변점(잠재적인 클라이언트 어플리케이션의 집합을 고려하여 도출)에서 잠재적인 가변치들의 전체 수 이고, 분자는 각 가변점에서 가변치 수들의 합이다.

적응성 측정을 위한 메트릭에서 서비스에 제공하는 가변점들의 수는 값을 설정하여 커스터마이즈 할 수 있는 오퍼레이션들의 수를 집계하여 측정한다. 예를 들면 본 논문의 사례연구에서 사용된 G 맵은 서비스 사용자가 프로그래밍 방식을 통해 지도의 유형과 마커의 종류, 지도의 확대 및 축소 등을 설정할 수 있다. 따라서 G 맵에서 제공되는 가변점들의 수를 측정 할 수 있다.

주어진 두 개의 메트릭은 가중치(Weight)를 적용하여 적응성을 계산한다. 만약 같은 가중치를 사용한다면, 적응성은 두 메트릭 값의 평균이다. 따라서, 적응성 값의 범위는 0...1이다. 적응성의 값이 높을수록 서비스 사용자의 특정 요구사항을 쉽고 효과적으로 커스터마이즈 할 수 있다.

5.4 조립성을 위한 메트릭

서비스의 조립성은 두 개의 메트릭으로 측정된다; *Modularity of service(MoS)*와 *Interoperability of Service (IoS).* MoS는 서비스의 응집력을 측정한다. MoS는 다음과 같이 계산된다;

$$MoS = 1 - \left(\frac{Number\ of\ Elements\ with\ External\ Dependency}{Total\ Number\ of\ Elements} \right)$$

분모는 서비스에서 요소들의 전체 수 이고 분자는 외부 서비스의 의존한 요소들의 수 이다. 조립성 측정을 위한 메트릭에서 외부 서비스의 의존한 요소들의 수는 성공적으로 기능을 수행하는 다른 서비스와 상호작용을 하는 오퍼레이션의 수를 집계하여 측정한다. 이러한 정보는 서비스 디스크립션과 제약의 한 형태인 다른 디스크립션에서 명시된다. 만약 서비스가 완전히 포함된 경우에 MoS는 1이다.

IoS는 현재 서비스가 참여한 모든 비즈니스 프로세스에서 현재 서비스와 그것에 의존적인 서비스들 사이에 효과적인 상호작용의 정도를 고려하여 측정한다. IoS는 다음과 같이 계산된다;

$$IoS = \left(\frac{Number\ of\ Dependent\ Services\ with\ Acceptable\ Interaction}{Total\ Number\ of\ Dependent\ Services\ in\ Participating\ BPs} \right)$$

분모는 현재 서비스가 참여하고 있는 모든 잠재적인 비즈니스 프로세스에서 현재 서비스에 의존적인 서비스들의 전체 수 이다. 분자는 허용된 상호작용 능력을 가진 의존적인 서비스들의 수 이다. 허용된 상호작용 능력을 가진 의존적인 서비스들의 수는 각 서비스 호출에 대한 QoS(서비스 정보와 시간을 수신하는 시간에 대한 결과를 반환)를 유지한 이전 데이터를 활용하여 얻는다. 이러한 데이터는 허용할 수 있는 QoS 결과를 받아들일 수 있는 서비스의 수를 유도할 수 있다.

주어진 두 개의 메트릭은 가중치를 적용하여 조립성을 계산한다. 만약 같은 같은 가중치를 사용한다면, 조립성은 두 메트릭 값의 평균이다. 따라서 조립성 값의 범위는 0...1이다. 조립성의 값이 높을 수록 다른 서비스와 통합하여 서비스 사용자의 특정 요구사항을 쉽고 효과적으로 커스터마이즈 할 수 있다.

5.5 재사용성을 위한 메트릭

서비스의 재사용성 평가는 4개의 품질속성과 메트릭으로 측정된다. 이러한 메트릭은 별도로 정의되어 있기 때문에, 본 논문에서는 서비스 재사용성 평가를 위해 각 메트릭들을 통합하여 계산한다.

각 메트릭들을 결합하는 쉬운 방법은 각 메트릭 값의 평균을 계산하는 것이다. 그러나, 도메인에 따라 각 메트릭은 재사용성의 정도를 결정하는데 서로 다른 영향을 준다. 따라서 각 메트릭의 다른 가중치 값을 적용하여 재사용성을 다음과 같이 평가한다;

$$Resuability = \sum_{i=1}^{4} W_i \cdot QA_i$$

QA_i는 품질속성 i번째를 위한 메트릭이고, i의 범위는 1...4이며 재사용성 평가를 위해 4개 품질속성을 정의한다. 그리고 W_i는 각 품질속성 QA_i를 위한 가중치 값이다. W_i는 도메인의 특성과 다른 요인들을 고려하여 정의되며, W_i의 모든 합은 1이다.

수식을 통해 재사용성의 값은 0...1 사이의 값을 가지며, 높은 값을 가질수록 서비스가 높은 재사용으로 설계되었다.

6. 재사용성 메트릭을 위한 사례연구

일반적으로 품질모델은 실제 프로젝트에 적용할 수 있는 개념이다. 품질모델의 추상적인 메트릭으로 인해, 메트릭의 값을 객관적으로 의도하거나 직접적으로 사용할 수 없기 때문에 품질평가자에 의해 주관적으로 결정한다. 제시한 메트릭을 적용하기 위해, 본 논문에서는 서로 다른 서비스 사업자가 제공하는 3개의 맵 서비스에 메트릭을 적용하여 사례연구를 수행한다; *G* 맵, *N* 맵, and *D* 맵. 본 논문에서 제시한 메트릭을 적용하기 위해 각 맵 서비스는 재사용성 측면을 고려하여 설계되었다. 첫 번째, 각 서비스 디스크립션에는 오퍼레이션과 이와 관련된 속성들이 기술되어 있으며 두 번째, 서비스를 검색하여 잘 발견할 수 있도록 신택틱 정보기술과 시맨틱 정보기술 등이 서비스 명세서에 기술되어 있으며 세 번째, 서비스에 필요할 것으로 예상되는 다양한 기능을 포함한 도메인 분석 보고서와 디자인 명세서(가변치와 가변점 정보를 제공) 네 번째, 다른 서비스와 상호작용 할 수 있는 오퍼레이션의 정보(이 정보는 서비스 디스크립션과 제약의 한 형태인 다른 디스크립션에서 명시된다.)등을 기반으로 본 논문에서 제안한 메트릭을 활용하여 클라우드 서비스 재사용성을 평가한다.

따라서, 본 사례연구에서는 각 변수들의 특징들을 고려하여 값을 수집한다. 각 변수의 값을 수집하기 위한 방법으로 History Log를 이용하거나 설문조사를 통해 얻을 수 있고, 설계모델에서 변수들과 관련된 값들을 통해 수집할 수 있으며, 서비스 제공자가 제공한 값들로부터 수집할 수 있다. 본 논문에서는 각 변수에 대한 값을 수집하기 위해 각 변수들의 특징을 고려하여 위에서 제시한 여러 방법들을 이용해 변수들의 값을 수집한다.

6.1 이해성을 위한 측정

CoS은 서비스 사용자가 제공된 서비스의 기능을 얼마나 쉽고 효과적으로 이해할 수 있는지를 측정하고 두 개의 변수로 계산된다. *Number of Field with Acceptable Readability*는 사용자들로부터 서비스 디스크립션에서 얼마나 많은 오퍼레이션과 관련된 속성들을 이해할 수 있는지를 측정한다. 그리고 *Number of Field with Acceptable Readability*의 값은 사용자들로부터 설문조사를 통해 수집되었다. 이 메트릭의 경우 객관적으로 의도하거나 직접적으로 사용할 수 없기 때문에 품질평가자에 의해 주관적으로 결정한다. 그리고 *Total Number of Fields*는 서비스 명세서를 통해 서비스 디스크립션에 등록된 모든 오퍼레이션과 관련된 속성들을 집계하여 획득한다.

따라서 <표 1>에서와 같이, 3개의 CoS 값을 얻었다; G 맵은 0.833, N 맵은 0.667, D 맵은 0.727.

〈표 1〉 이해성의 측정 결과

분자 & 분모	G 맵	N 맵	D 맵
Number of Fields with Acceptable Readability	10	6	8
Total Number of Fields	12	9	11
Total	**0.833**	**0.667**	**0.727**

이 값들을 해석하면, 각 디스크립션에 명세 된 모든 필드에서 G 맵 필드의 83%를, N 맵 필드의 67%를, D 맵 필드의 73%를 이해했다는 결론을 얻었다.

6.2 홍보성을 위한 측정

AoS는 서비스 사용자가 얼마나 쉽게 서비스를 발견하고 접근할 수 있는지를 측정하며, 두 개의 변수로 계산된다.

*Number of Consumers with Service Discoverability*는 후보 서비스 검색 후 서비스 사용결정을 한 사용자들의 수와 같은 다양한 정보를 포함한 통계 데이터로부터 획득한다. *Total Number of Potential Consumers*는 서비스 제공자가 ROI와 잠재된 사용자들을 고려하여 서비스 개발 여부를 결정하고 비즈니스 분석의 결과를 통해 얻는다.

따라서 <표 2>에서와 같이, 3개의 AoS 값을 얻었다; G 맵은 0.91, N 맵은 0.828 D 맵은 0.87.

〈표 2〉 홍보성의 측정 결과 (Number in ten thousands)

분자 & 분모	G 맵	N 맵	D 맵
Number Of Consumers with Service Discoverability	91	58	74
Total Number of Potential Consumers	100	70	85
Total	**0.91**	**0.828**	**0.87**

이 값들을 해석하면, 분류 트리를 찾거나 적합한 쿼리를 전송한 잠재적인 사용자는 G 맵은 91%, N 맵은 83%, D 맵은 87% 라는 결론을 얻었다.

6.3 적응성을 위한 측정

적응성은 서비스 사용자들이 자신의 요구에 맞게 서비스를 커스터마이즈 할 수 있는 능력을 측정하며, 두 개의 메트릭으로 계산된다; CoV와 CoA. CoV는 두 개의 변수를 사용하여 서비스가 제공하는 가변점들의 수를 기준으로 적응성을 평가한다. *Number of Variation Points Supported*는 값을 설정하여 커스터마이즈 할 수 있는 오퍼레이션들의 수를 집계하여 측정한다. *Total Number of Potential Variation Points*를 얻기는 어렵다. 본 사례연구에서는 서비스 사용자가 서비스에 필요할 것으로 예상되는 다양한 기능을 포함한 도메인 분석 보고서와 디자인 명세서로부터 값을 얻는다. 예를 들면 G 맵은 서비스 사용자가 프로그래밍 방식을 통해 지도의 유형과 마커의 종류, 지도의 확대 및 축소 등을 설정할 수 있다. 값은 실행 시에 변경할 수 있다. 그러나 그것들은 선호도로 저장되며, 어플리케이션 실행 시에 복원된다. 따라서 G 맵 에서 제공되는 몇 가변점들을 판단할 수 있다.

따라서 <표 3>에서와 같이, 3개의 CoV 값을 얻었다; G 맵은 0.833, N 맵은 0.667, D 맵은 0.75.

〈표 3〉 CoV의 측정 결과

분자 & 분모	G 맵	N 맵	D 맵
Number of Variation Points Supported	5	2	3
Total Number of Potential Variation Points	6	3	4
Total	**0.833**	**0.667**	**0.75**

CoA는 두 개의 변수를 사용하여 가변점을 위한 가변치의 수를 기준으로 적응성을 평가한다. *Number of Variants supported in* VP_i는 특정 VP_i에 대한 모든 옵션을 집계하여 얻는다. 예를 들면, G 맵은 뷰 타입의 가변점 설정을 위한 두 가지 뷰를 제공한다; 맵 뷰와 위성 뷰 즉, 가변치의 수는 2이다. 그리고 Total Number of Potential Variation Points는 VP_i에서 잠재적인 가변치의 전체 수는 디자인 명세서를 통해 얻는다.

따라서 <표 4>에서와 같이, 3개의 CoA 값을 얻었다; G 맵은 0.71, N 맵은 0.5, D 맵은 0.75.

〈표 4〉 CoA의 측정 결과

분자 & 분모	G 맵	N 맵	D 맵
Number of Variants supported in VP_i	5	2	3
Total Number of Potential Variants in VP_i	7	4	5
Total	**0.71**	**0.5**	**0.6**

CoV와 CoA의 값을 통해, <표 5>에서와 같이 가중치를 적용하여 적응성을 측정한다. 가중치는 품질평가자에 의해 주관적으로 결정된다; 그러나 가중치의 합은 1이 되어야 한다. 가변점 없이는 변화가 없기 때문에, 본 논문에서는 CoV가 CoA보다 적응성을 측정하는데 더 영향력이 있다고 결론을 내렸다. 따라서 CoV의 가중치를 0.7, CoA의 가중치를 0.3으로 할당 하였다.

〈표 5〉 적응성의 측정 결과

G 맵	(0.7 Í 0.833) + (0.3 Í 0.71)	**0.796**
N 맵	(0.7 Í 0.667) + (0.3 Í 0.5)	**0.616**
D 맵	(0.7 Í 0.75) + (0.3 Í 0.6)	**0.705**

이 값들을 해석하면, G 맵은 비교적 다른 서비스들보다 서비스 사용자의 특정 요구사항을 커스터마이즈 하는데 더 효과적이였다.

6.4 조립성을 위한 측정

조립성은 서비스가 다른 서비스와 함께 효과적으로 조립되는 능력정도를 측정하며 두 개의 메트릭으로 계산된다; MoS와 IoS. MoS는 두 개의 변수로 서비스의 독립성을 평가한다. *Number of Elements with External Dependency*는 성공적으로 기능을 수행하는 다른 서비스와 상호작용을 하는 오퍼레이션의 수를 집계하여 측정한다. 이 정보는 서비스 디스크립션과 제약의 한 형태인 다른 디스크립션에서 명시된다. 그리고 Total Number of Elements는 서비스 디스크립션의 모든 오퍼레이션을 집계하여 얻는다.

따라서 <표 6>에서와 같이, 3개의 MoS 값을 얻었다; G 맵은 0.778, N 맵은 0.8, D 맵은 0.83.

〈표 6〉 MoS의 측정 결과

분자 & 분모	G 맵	N 맵	D 맵
Number of Elements with External Dependency	2	1	1
Total Number of Elements	9	5	6
Total	**0.778**	**0.8**	**0.83**

IoS는 허용할 수 있는 QoS와 다른 서비스에 대한 결과를 제공하는 서비스를 고려하여 상호운영의 정도를 평가하며 두 개의 변수를 통해 계산된다.

*Number of Dependent Services with Acceptable Interaction*의 값은 각 서비스 호출에 대한 QoS(즉, 서비스 정보와 시간을 수신하는 시간에 대한 결과를 반환)를 유지한 이전 데이터를 활용하여 얻는다. 이러한 데이터와 함께, 허용할 수 있는 QoS결과를 받아들일 수 있는 서비스의 수를 유도할 수 있다. *Total Number of Dependent Services in Participating BPs*는 첫 번째로 서비스가 참여된 비즈니스 프로세스의 리스트를 알아내고, 비즈니스 프로세스에서 의존적인 서비스의 수를 집계하여 측정한다.

따라서 <표 7>에서와 같이, 3개의 IoS 값을 얻었다; G 맵은 0.91, N 맵은 0.83, D 맵은 0.89.

〈표 7〉 IoS의 측정 결과

분자 & 분모	G 맵	N 맵	D 맵
Number of Dependent Services with Acceptable Interaction	48	35	42
Total Number of Dependent Services in Participating BPs	53	42	47
Total	**0.91**	**0.83**	**0.89**

MoS와 IoS의 값을 통해, <표 8>에서와 같이 가중치를 적용하여 조립성을 측정한다. 가중치는 품질평가자에 의해 주관적으로 결정된다; 그러나 가중치의 합은 1이 되어야 한다. 맵 서비스는 경로를 탐색하거나 주소를 가져오는 상당히 응집력 있는 기능성들을 제공하므로, 본 논문에서는 IoS가 MoS보다 조립성을 측정하는데 더 영향력이 있다고 결론을 내렸다. 따라서, MoS의 가중치를 0.25, IoS의 가중치를 0.75로 할당 하였다.

〈표 8〉 조립성의 측정 결과

G 맵	(0.25 Í 0.778) + (0.75 Í 0.91)	**0.877**
N 맵	(0.25 Í 0.8) + (0.75 Í 0.83)	**0.823**
D 맵	(0.25 Í 0.83) + (0.75 Í 0.89)	**0.875**

이 값들을 해석하면, G 맵, N 맵, D 맵은 비슷한 조립성의 값을 가진다. 즉, 유사한 노력으로 다른 서비스와 함께 통합되고 있다는 결론을 얻었다.

6.5 재사용성을 위한 측정

위의 계산된 4가지의 품질속성의 값을 결합하여 세 개의 맵 서비스들의 재사용성을 평가한다. 그리고 각 품질속성을 위한 가중치를 결정해야 한다. 맵 서비스의 특성과 본 논문의 사례연구의 목적을 고려하여 모두 똑같이 4가지의 품질속성에 0.25씩 할당한다.

따라서, <표 9>에서와 같이 3개의 맵 서비스를 위한 재사용성의 값을 얻었다.

〈표 9〉 맵 서비스 재사용성을 위한 측정 결과

품질속성	결과 값			가중치
	G 맵	*N* 맵	*D* 맵	
이해성	0.833	0.667	0.75	0.25
홍보성	0.91	0.828	0.87	0.25
적응성	0.796	0.616	0.705	0.25
조립성	0.743	0.6	0.667	0.25
재사용성	0.82	0.677	0.748	-
비율	82%	67.7%	74.8%	-

<표 9>의 결과값들을 해석하면, G맵 서비스는 다른 서비스들 보다 재사용성이 높게 평가되었다. 즉, G 맵 서비스는 높은 재사용성을 고려하여 설계되었기 때문에 사용자들이 서비스 디스크립션에서 많은 오퍼레이션과 관련된 속성들을 이해할 수 있도록 기술되어 있고, G맵 서비스가 잘 발견될 수 있도록 신택틱, 시맨틱 정보기술이 서비스 명세서에 잘 반영되어 있으며, 다양한 기능을 포함한 도메인 분석 보고서와 디자인 명세서 그리고 다른 서비스와 상호작용할 수 있는 오퍼레이션의 정보 등이 다른 두 개의 맵 서비스들보다 상세히 제공되어 본 논문에서 제안한 메트릭에 잘 반영되었다.

7. 재사용성 향상을 위한 가이드라인

본 장에서는 5장에서 정의된 품질메트릭을 기반으로 사례연구의 품질 결과를 품질평가자의 판단에 의해 적용될 재사용성 향상을 위한 가이드라인을 제안한다.

7.1 이해성 향상

사용자는 서비스 인터페이스만을 이용하여 서비스의 기능을 이해하므로, 서비스 인터페이스가 서비스의 기능을 효과적으로 명세해야 사용자의 이해성을 향상시킬 수 있다. 즉, 서비스의 신택틱한 정보만을 이용하여 서비스의 기능을 정확하게 이해하는 것은 어렵다. 그러므로, 서비스의 기능성 정보를 신택틱한 측면 외에 시맨틱한 정보도 포함해야 서비스의 이해성이 향상될 수 있다.

• WSDL 혹은 RESTful 기반의 신택틱한 정보향상

서비스의 신택틱한 측면을 잘 정의하기 위해 서비스의 호

출방법이 잘 정의되어야 한다. 현재 널리 사용되고 서비스 호출 방법은 WSDL과 RESTful을 이용하는 방법이 있다. WSDL의 경우 요소들 중에서 'types'는 교환 메시지에 대한 설명과 이때, 사용되는 데이터 형식이 잘 정의 되어 있지 않거나 설명이 부족하다면 서비스의 인터페이스만 이용하는 사용자는 이해하기 힘들기 때문에 'types'부분에서 메시지의 설명과 데이터 형식이 잘 정의되어 있어야 한다. 'message'는 메시지가 어떤 시스템에서 정의되고 관련이 있는지를 논리적으로 명확하게 정의되어 있어야 한다. 'portType'는 오퍼레이션의 집합이기 때문에 각 오퍼레이션들의 입□출력 메시지가 어떠한 형태로 참조되는지 정의되어야 한다. 'binding'은 앞서 정의된 'portType'에 정의된 메시지에 대해 명확하고 구체적으로 프로토콜과 데이터형식이 잘 명세되어야 한다.

또한 도구를 이용하여 Java 코드를 WSDL로 맵핑 할 수 있다. 대부분 Java 유형은 직접 표준 XML 유형으로 매핑된다. 다음 (그림 4)는 Java코드를 WSDL의 'types'의 부분과 매핑하는 내용이다. 표준 XML 유형으로 직접 매핑될 수 없는 Java 유형은 wsdl:types 섹션에서 생성되고, Java Bean 패턴과 일치하는 Java 클래스는 xsd:complexType으로 매핑된다.

다음은 RESTful을 이용하여 서비스를 접근하는 방법이다. RESTful는 리소스 중심의 표현, 전달, 접근 방식의 특성으로 인해 4가지의 기본원칙을 잘 준수해야 서비스의 신택틱한 정보가 향상될 수 있다. 4가지의 기본 원칙은 리소스의 구별, Representation을 통한 리소스 조작, 서술적 메시지 어플리케이션 상태에 대한 엔진 하이퍼미디어다. 이 4가지 원칙 중 서비스이 이해성과 관련이 있는 항목은 서술적 메시지 이다. 서비스 사용자들을 위해 어떻게 처리되는지에 대한 충분한 정보를 제공되어야 한다. 예를 들어 어떤 파서를 불러야 하는지 모를 경우, MIME type 과 같은 인터넷 미디어 타입의 사용을 들 수 있다. 미디어 타입만 가지고도, 클라이언트는 어떻게 그 내용을 처리해야할 지 알 수 있어야 한다. 즉, 메시지를 이해하기 위해 그 내용까지 살펴봐야 한다면, 그 메시지는 서술적이 아니다. 'application/xml' 미디어 타입의 경우 코드 다운로드가 되지 않으면 그 내용을 가지고 무엇을 해야할지 충분히 알 수 없기 때문이다.

• OWL-S 기반의 시맨틱한 정보향상

대부분의 서비스는 시맨틱한 정보를 서비스 인터페이스에 기술하지 않다. 따라서 본 논문에서는 OWL-S을 이용하여 시맨틱한 정보를 기술하는 방법에 대한 가이드라인 제안한다. OWL-S는 온톨로지는 서비스 개요를 기술하는 서비스 프로파일(Profile)과 서비스 실행과 연관된 프로세스 정보를 제공하는 프로세스 모델, 그리고 서비스 매핑과 관련된 서비스 그라운딩(Grounding)등 3가지 구성요소를 포함하고 있다.

프로파일은 기능적 명세 즉, IOPE(Inputs, Outputs, Preconditions, Effects)을 기반으로 하기 때문에 서비스 제공자는 서비스의 시맨틱 정보을 기술을 하기 위해 서비스 서비스에 의해 요구되는 입력, 생성되는 출력에 대한 정보, 서비스 실행에 필요한 조건과 서비스 실행에 의한 효과등을 기술해야 하며, 또한 비 기능적 명세는 서비스 이름, 서비스 제공자 정보 등을 서비스 제공자가 상세하게 기술해야 한다. 프로세스 모델은 서비스 사용자가 쉽게 이해할 수 있도록 프로세스 단위에서 서비스 조합을 위한 정보를 기술하는 것으로 다른 서비스와의 조합과 실행과정 등을 제공자가 상세히 기술해야 한다. 마지막으로 그라운딩은 WSDL과 매핑 정보를 포함하는 부분이기 때문에 서비스 그라운딩 정보를 통해 통신 프로토콜, 사용포트, 메시지 형식 등 컴퓨터 프로그램이나 에이전트가 실제 해당 서비스에 접근할 수 있는 방법을 서비스 사용자는 이를 구체화하여 기술해야 한다.

7.2 홍보성 향상

서비스 제공자에 의해 등록된 서비스는 서비스 저장소에

Java Code

```
public abstract class Base {
   public Base() {}
   public int a;                  // mapped
   private int b;                 // mapped via setter/getter
   private int c;                 // not mapped
   private int[] d;               // mapped via indexed setter/getter

   public int getB() { return b;}      // map property b
   public void setB(int b) {this.b = b;}

   public int[] getD() { return d;}    // map indexed property d
   public void setD(int[] d) {this.d = d;}
   public int getD(int index) { return d[index];}
   public void setB(int index, int value) {this.d[index] = value;}

   public void someMethod() {...}      // not mapped
}

public class Derived extends Base {
   public int x;                  // mapped
   private int y;                 // not mapped
}
```

Mapping

WSDL Code

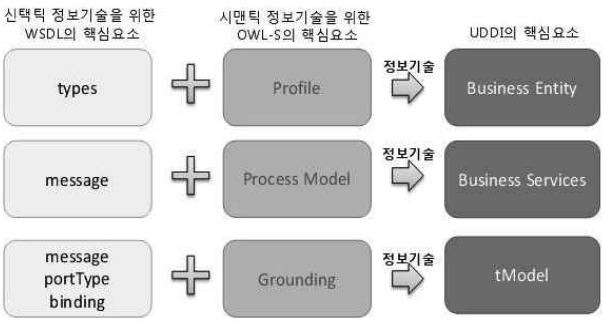
```
<xsd:complexType name="Base" abstract="true">
   <xsd:sequence>
     <xsd:element name="a" type="xsd:int" />
     <xsd:element name="b" type="xsd:int" />
     <xsd:element name="d" minOccurs="0"
       maxOccurs="unbounded" type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Derived">
   <xsd:complexContent>
      <xsd:extension base="ns:Base">
         <xsd:sequence>
            <xsd:element name="x" type="xsd:int" />
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

(그림 4) Java 코드와 WSDL간의 매핑

등록되며 등록된 서비스들은 서비스 발견을 위해 공개된다. 등록된 서비스들이 잘 검색될 수 있도록 가이드라인을 제시하여 홍보성을 향상시킨다. 따라서 본 절에서는 UDDI을 통해 서비스가 잘 검색 되도록하는 방법을 제안한다.

UDDI의 핵심요소들 중에서 Business Entity는 Business 객체에 대한 이름, 설명, URL, 식별 및 분류를 위한 정보가 정확하게 작성되어야 하며, Business Service는 Business 객체가 제공하는 서비스에 대한 논리 정보가 명확하고 이해하기 쉽게 작성되어야 한다. tModel은 서비스 이용에 필요한 통신 프로토콜, 메시지 형식, 서비스 이용 규칙등과 같은 기술 표준들이 잘 작성되어야 한다.

(그림 5)는 서비스가 잘 발견될 수 있도록 신택틱과 시맨틱 서비스 명세서를 기반으로 UDDI 핵심요소를 기술하는 방법을 보여준다.



(그림 5) 신택틱 / 시맨틱 서비스 명세서 기반의 UDDI 핵심요소 기술

(그림 5)을 통해, UDDI의 Business Entity에서 서비스 이름, 설명, URL, 식별 및 분류를 위한 정보가 정확하게 작성되기 위해 시맨틱적으로 기능적 명세는 IOPE를 기반으로 작성하고, 비 기능적 명세는 서비스 이름, 서비스 제공자 정보 등을 기술하며, 신택틱적으로는 교환될 메시지에 대한 설명과 사용되는 데이터 형식을 서비스 제공자가 정확하게 기술한다.

UDDI의 Business Service에서 서비스에 대한 논리정보를 명확하게 작성하기 위해 신맨틱적으로 프로세스 단위의 서비스 조합을 위한 정보와 다른 서비스와의 조합과 관련된 내용을 기술하고, 신택틱적으로는 메시지가 어떤 시스템에서 정의되고 관련되어 있는지를 논리적으로 제공자가 기술한다.

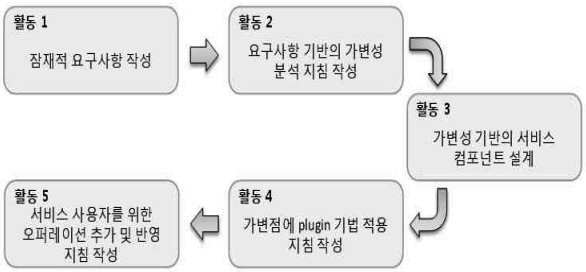
UDDI의 tModel에서 서비스 이용에 필요한 통신 프로토콜, 메시지 형식, 서비스 이용 규칙등과 같은 기술 표준들을 작성하기 위해서, 시맨틱적으로 서비스 그라운딩 정보를 통해 통신 프로토콜, 사용포트, 메시지 형식 등 프로그램이나 에이전트가 서비스에 접근할 수 있는 방법을 기술하고, 신택틱적으로는 메시지가 어떤 시스템에서 정의되어 있는지, 각 오퍼레이션의 입□출력 메시지를 참조할 수 있게 하고, 메시지에 대한 구체적인 프로토콜과 데이터 형식을 서비스 사용자가 기술한다.

7.3 적응성 향상

서비스는 블랙박스 형태로 제공되기 때문에, 서비스 사용자들이 잠재적인 가변점의 존재 여부를 찾을 때 사용할 수 있는 유일한 방법은 서비스 명세서이다. 따라서 서비스 제공자가 C&V 분석을 통해 서비스 가변성을 잘 제공해야 서비스 사용자가 이를 통해 원하는 어플리케이션을 쉽게 만들 수 있다. 본 연구에서는 적응성을 향상시키기 위한 방법으로 두 가지를 제안한다; 서비스 제공자가 수행해야 하는 활동과 서비스 사용자가 수행해야 하는 활동이다.

• 서비스 제공자가 수행해야 하는 활동

적응성 향상을 위해 서비스 제공자가 수행해야 하는 활동은 (그림 6)과 같다.

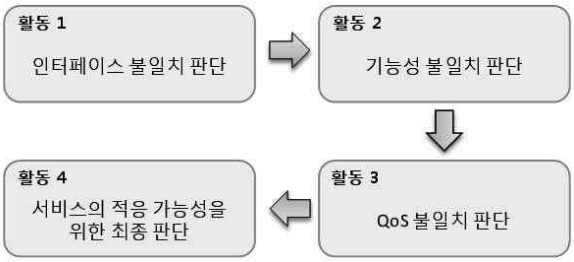


(그림 6) 서비스 제공자가 수행해야 하는 활동

활동 1에서 서비스 제공자는 개발하고자 하는 서비스의 도메인을 고려하여 잠재적 요구사항을 작성해야 한다. 잠재적 요구사항을 작성하기 위해서는 여러 요구사항들을 고려하여 가변성을 분석하는 지침을 제공한다. 활동 2에서 가변성을 분석하는 지침을 작성하기 위해서는 우선 클라우드 서비스에서 발생가능 한 가변점의 종류들을 식별한다. 가변점의 종류는 클라우드 서비스의 인터페이스, 워크플로우, 서비스 컴포지션 등이 있다. 활동 3은 가변성을 서비스 컴포넌트에 반영하여 설계를 한다. 활동 4는 open 가변성을 위해 각 가변점에 대한 plugin 기법을 적용하는 지침등을 작성한다. 마지막으로 활동 5에서는 서비스 제공자가 서비스 컴포넌트에 반영된 가변성을 사용자가 설정할 수 있도록 서비스 명세서에 관련된 오퍼레이션을 추가 및 반영하는 지침등을 작성한다. 지침 기술에는 WSDL에서 가변성과 관련된 'Required Interface를 기술하는 방법'에 대한 내용 등을 기술할 수 있다.

• 서비스 사용자가 수행해야 하는 활동

서비스를 선택할 때, 서비스 사용자들은 서비스 명세서만을 참고하여 그 서비스의 적응 가능성을 결정한다. 즉 서비스 인터페이스 불일치는 서비스 명세서로부터 도출되는 것을 암시한다. 따라서 서비스 명세서에 명시된 요소 항목을 기반으로 인터페이스 불일치, 기능성 불일치, QoS 불일치를 통해 서비스의 적응 가능성을 판단한다. 적응성 향상을 위해 서비스 사용자가 수행해야 하는 활동은 (그림 7)과 같다.



(그림 7) 서비스 사용자가 수행해야 하는 활동

서비스 사용자는 인터페이스 불일치를 판단하기 위해 다음 내용을 기반으로 판단할 수 있다. 인터페이스 불일치는 SVC_j의 오퍼레이션 시그니처(Signature)가 FEA_i에서 기대하는 시그니처와 완전하게 일치하지 않을 때 발생한다. 일반적으로 서비스는 블랙박스 형태로 제공되기 때문에, 이미 발행된 서비스 인터페이스는 수정할 수 없다. FEA_i에서 기대하는 인터페이스의 수정이 불가능한 유형이다. 예를 들어, 만약 FEA_i가 표준화된 API를 채택한다면 그것의 인터페이스는 수정되지 않는다. 이것이 인터페이스의 정적 어댑테이션이 필요로 하는 경우에 해당된다. 이러한 내용을 기반으로 서비스 사용자는 적응 가능성을 판단할 수 있다. (FEA = Feature, SVC = Service)

서비스 사용자는 기능성 불일치를 판단하기 위해 다음 내용을 기반으로 판단할 수 있다. 기능성 불일치는 $Fn(SVC_j)$와 $Fn(FEA_i)$의 기능성 사이에 존재하는 차이다. 여기서 $Fn(SVC_j)$와 Fn(FEA_i)는 각각 SVC_j 와FEA_i에서 제공하는 기능성을 나타낸다. 서비스 인터페이스가 서비스 사용자가 원하는 인터페이스의 형태와 일치하더라도 기대하는 기능성과 서비스의 기능성이 완전히 일치하지 않을 수 있다. 다음 (그림 8)은 두 기능성을 비교할 때 가능한 네 가지 경우를 보여준다.

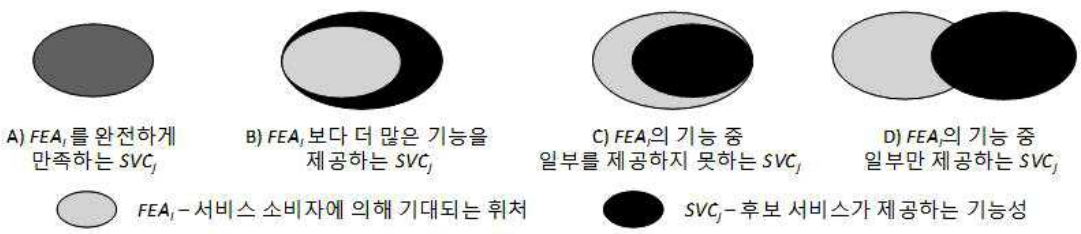
(그림 8)에서 A)유형은 $Fn(SVC_j)$와 $Fn(FEA_i)$가 완전 일치하여 SVC_j가 어떠한 어댑테이션 없이 사용 가능한 것이다. B)유형은 $Fn(SVC_j)$가 FEA_i에서 요구하지 않는 추가적인 기능성을 제공하는 경우이다. 이와 같은 경우에, 서비스가 제공하는 추가적인 기능성이 성능 상의 부작용을 도출할 수 있다. C)유형은 $Fn(SVC_j)$가 $Fn(FEA_i)$의 일부분을 제공하고 나머지 부분은 부족한 경우이다. D)유형은 $Fn(SVC_j)$와 $Fn(FEA_i)$가 완전히 중복되지 않고 일정 부분만 일치하는 경우이다. 즉, B)와 C)유형을 통합한 경우이다. 기능성 불일치는 B), C), D) 유형에서 발생한다. 기능성 불일치는 인터페이스 불일치가 발생할 때 자주 발생하는 것에 유의해야 한다. 그 이유는 서비스 오퍼레이션과 같은 같이 기능성 모듈이 입력 매개변수를 통해 지원되는 데이터와 값을 다루기 때문이다. 따라서, 매개변수와 관련된 인터페이스 불일치는 기능성 불일치를 유발시킬 가능성이 높다. 위 내용을 기반으로 서비스 사용자는 적응 가능성을 판단 할 수 있다.

서비스 사용자는 QoS 불일치를 판단하기 위해 다음 내용을 기반으로 판단할 수 있다. SVC_j의 SLA에 명시된 서비스 품질이 FEA_i에서 기대한 서비스 품질 수준을 완전히 만족하지 못하는 문제 상황을 나타낸다. SVC_j의 인터페이스와 기능성이 FEA_i에서 기대하는 것과 일치하더라도, 허용할 수 없을 정도로 낮은 값의 서비스 품질 속성은 QoS 불일치를 야기시킨다. 따라서 이러한 내용을 기반으로 서비스 사용자는 적응 가능성을 판단 할 수 있다.

7.4 조립성 향상

조립성을 향상하기 위해, 본 연구에서는 두 가지의 효과적인 지침을 제시하는데, 즉 모듈성과 인터페이스 관점의 지침이다. 조립성의 메트릭을 보면, 분자에 해당되는 외부 항목들과의 의존도가 주된 요소로 작용한다. 즉, 높은 모듈성을 가지는 서비스는 다양한 어플리케이션 개발에 포함되어 사용될 경우, 주변 모듈과의 의존도가 낮으므로 결과적으로 관련 서비스들과의 연결하여 필요한 비지니스 프로세스로 효과적으로 통합된다.

모듈화는 시스템의 복잡성을 관리하기 위한 기본 지침으로 지속적으로 연구가 되어 왔다. 즉, 복잡한 시스템을 더 작고 덜 복잡한 서브 시스템으로 구성하여 원칙적인 방법으로 이 서브 시스템들을 다시 결합함으로써, 시스템의 유연성을 높이는 것뿐만 아니라 보다 쉽게 시스템의 이해하는 것이 가능하며 시스템의 개발 시간을 단축한다. 모듈을 설계하여 시스템의 복잡한 설계 결정들이나 시스템의 변화에 대한 결정을 설계하는 것을 외부에 숨길 수 있다. 따라서, 각 서비스를 설계 시, 그 서비스가 제공하는 기능성의 응집도(Cohesion)이 높도록 설계하고, 주변 서비스들과의 의존도(Coupling)을 낮도록 설계해야 한다. 따라서, 하나의 서비스가 잘 정의된 하나의 기능성을 제공하도록 기능 범주를 설정하여야 한다. 한 서비스가 복수개의 기능군을 제공한다면 너무 큰 서비스 단위가 되어 재사용성과 조립성을 낮추게 된다. 또한, 한 기능을 복수개의 서비스들이 연동되어 제공



(그림 8) 서비스 기능 불일치가 가능한 경우

한다면, 해당되는 여러 서비스들을 통합하고 이들간의 실시간 메시지 교환의 부하들으로 조립성이 낮게 된다.

두 번째 지침은 인터페이스의 적절성(Appropriateness)을 들 수 있다. 서비스는 잠재적으로 여러 어플리케이션에서 범용으로 사용되어야 하며, 그 사용되는 방식을 서비스의 인터페이스를 통해서 메시지를 교환하는 방법이다. 따라서, 여러 어플리케이션들간의 인터페이스 공통성과 가변성 분석을 통하여 공통된 인터페이스 항목을 수정되지 않도록 고정적으로 설계하고, 가변적인 인터페이스 항목은 다시 가변점과 각 가변점 별 가변치를 반영하여 어플리케이션 별로 필요한 인터페이스로 특화 할 수 있도록 설계하여야 한다.

예를 들면, 구글의 맵 서비스 인터페이스는 다양한 맵 어플리케이션들이 요구하는 조금씩 다른 인터페이스를 모두 지원하기 위해서, 공통된 맵 기능에 해당되는 서비스 인터페이스는 고정하고, 가변적인 부분은 추상클래스, 템플릿클래스, 객체 대치성의 장치를 이용하여 어플리케이션 별로 필요한 인터페이스를 사용할 수 있도록 설계하였다. 그 결과 대부분의 맵 어플리케이션에서 불일치의 문제없이 서비스 형태 즉 Mashup 형태로 조립되어 사용하는데 효과적이다.

8. 이론적 분석과 품질모델 평가

본 장에서는 메트릭들을 검증하기 위해 Kitchenham[17]의 접근방식에 따라 이론적 분석을 수행하고, 올바른 메트릭 평가를 위해 4개의 기준를 기반으로 평가한다. 이론적 분석은 명확하게 정의된 기준과 관련하여 검증된 측정방식을 제공한다. 또한 IEEE Std. 1061와 Ejiogu의 연구를 통해 4개의 기준을 도출하여 제안된 품질모델을 측정한다.

8.1 이론적 분석(Theoretical Analysis)

Kitchenham의 접근방식의 메트릭 분류에 따르면, 본 논문에서 제시한 메트릭의 값을 직접 측정하지 않기 때문에 직접적으로 메트릭을 사용한다.

속성의 유효성에 대한 이론적 분석은 다음과 같다. 3장에서 클라우드 서비스는 다음과 같은 특성들을 제시하였다; 공통성, 인터넷을 통합 접속, 잘 정의된 인터페이스, 사용에 대한 비용지불, 가벼운 클라이언트 모델, 높은 가용성, 높은 확장성 등의 특성으로부터 관찰된 품질속성을 도출되었다; 만약 클라우드 서비스 제공자가 높은 특성을 제시한다면, 특성의 높은 수준은 품질속성의 가치를 높이는데 기여하고, 관련 품질속성의 가치를 높이는데 기여한다. 따라서, 품질속성의 높은 가치와 관련된 특성을 지원하는데 높은 수준으로 제시한다. 예를 들면, 만약 클라우드 서비스 제공자가 높은 수준의 가용성을 제시한다면, 관련된 속성인 적응성과 조립성에 높은 가치를 제공한다. 따라서, 본 논문에서 제시한 속성은 Kichenham의 프레임워크에 따라 유효하다는 결론을 얻었다.

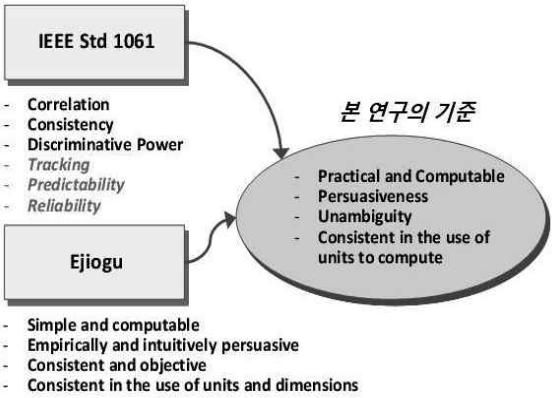
구성단위 유효성에 대한 이론적 분석은 다음과 같다. 위에서 설명한 내용을 기반으로 각 속성은 클라우스 서비스로부터 제시되었다. 다시 말해서, 속성을 측정하기 위해 사용된 메트릭은 클라우스 서비스의 특징으로부터 제시된다. 그리고 클라우드 서비스는 유도된 특성을 구성단위로 대신한다. 따라서 메트릭은 측정되는 속성의 구성단위를 사용한다. <표 10>에서처럼, 각 메트릭에서 사용된 구성단위를 요약하였다. 예를 들면, 구성단위인 *'Number of Variants Supported in VP_i'*는 다양성의 풍부함을 측정하는데 사용된다. 그리고 클라우드 서비스의 적응성을 명세하기 위한 대표적인 구성단위는 가변점이다. 그러므로, 구성단위 '가변점' 측정은 다양성의 풍부함을 측정하기 위해 적합한 구성단위이다. 요약하면, 각 메트릭에서 사용되는 구성단위 측정은 속성을 측정하기 위해 적합한 방법이다. 그러므로 본 논문에서는 재사용 모델에서 사용된 모든 경우의 구성단위가 유효하다고 주장한다.

〈표 10〉 각 메트릭에서 사용된 구성단위

메트릭	측정	구성단위
CoS	분모	가변치
	분자	가변점
AoS	분모	가변치
	분자	가변점
CoV	분모	가변치
	분자	가변점
CoA	분모	가변치
	분자	가변점
MoS	분모	가변치
	분자	가변점
IoS	분모	가변치
	분자	가변점

8.2 평가

제시한 재사용성 평가 모델의 유용성을 검토한다. 본 논문에서 제시한 4개의 메트릭의 유용성과 실용성을 Ejiogu[18]의 연구와 IEEE Std. 1061[19] 을 통해서 유도된 기준을 사용하여 평가한다. Ejiogu은 효과적인 소프트웨어 메트릭이 포함하고 있어야 하는 속성의 집합들을 정의하였다. IEEE Std 1061은 메트릭 검증을 위해 6개의 유효한 기준을 정의하였다. (그림 9)에서, Ejiougu의 연구와 IEEE Std. 1061는 본 논문에서 제시한 메트릭 평가를 위해 기준을 정의하는데 필요한 기본적인 역할을 한다. IEEE 1061에서 3가지의 기준인 연관성(Correlation), 일관성(Consistency). 구별능력(Discriminative Power)을 채택했다. 나머지 3가지의 기준(Tracking, Predictability, and Reliability)을 포기한 이유는 다음과 같다. 재사용성 모델은 개발이 완료되어야만 적용가능하다. 그러므로, 서비스 품질의 변화를 추적할 필요가 없다. IEEE Std 1061에서 기준 예측은 기준을 평가하기 위한 변수로서 시간을 포함하였다. 그러나 이미 개발된 클라우드 서비스와는 관련없이 없다.



(그림 9) 메트릭을 평가하는 기준 유도

따라서 각 정의된 기준과 각 기준을 제시된 재사용성 품질 모델에 적용한다.

실용성과 계산성(Practical and Computable)의 기준은 제안된 메트릭이 실제 프로젝트에 적용될 수 있는지와 계산은 오디네이트(Ordinate) 노력이나 시간에 요구해서는 안된다. 평가결과와 근거는 본 논문에서 제시한 4가지 메트릭은 각 메트릭에서 사용되어 측정되는 계산기법은 실제 적용할 수 있다.

설득력(Persuasiveness)의 기준은 메트릭은 제품 속성에 대한 엔지니어의 직관적인 개념을 고려해서 만족해야 할 품질속성과 메트릭 사이에서 명확한 관계가 존재해야 한다. 평가결과와 근거는 본 논문에서 제시한 4가지 메트릭은 고객의 직관적인 개념을 만족시킬 수 있다. 그리고 품질속성의 관계는 클라우드 서비스의 특성으로부터 유도된 메트릭이다.

명료성(Unambiguity)의 기준은 메트릭 결과는 모호한 결과를 얻을 수 없다. 결과를 명확하게 해석하기 위해 메트릭 값의 범위를 최대 및 최소의 값으로 경계하고 있어야 한다. 평가결과와 근거는 제안된 메트릭의 결과는 명확하고 모호하지 않다. 5장에서 정의한 바와 같이, 각 메트릭은 0...1의 가능한 값의 범위를 가지고 있다.

구성단위 사용에 일관성(Consistent in the use of units to compute)의 기준은 메트릭의 수학적 계산은 단위의 특별한 조합으로 측정을 위해 구성단위가 일관성이 있어야 한다. 평가결과와 근거는 측정에 사용되는 단위는 일관성이 있다. 각 메트릭의 측정 계산을 위해 본 연구에서는 <표 10>과 같이 가변치와 가변점을 실용적인 단위로 사용하였다.

9. 결 론

클라우드 컴퓨팅에서 서비스 제공자는 다양한 어플리케이션들 중에서 재사용 특징과 공통성을 고려하여 서비스를 개발하고 등록하며, 서비스 사용자는 어플리케이션을 구축할 때 서비스를 찾거나 재사용한다. 따라서 재사용성은 클라우드 서비스의 중요한 본질적인 특성이다. 서비스의 재사용성이 높으면 투자대비 높은 수익을 낼 수 있다. 따라서 본 논문에서는 클라우드 서비스를 위한 재사용성 평가 모델을 제안하였다.

클라우드 컴퓨팅과 서비스 지향 아키텍처에서 서비스와 관련된 연구를 통해 클라우드 서비스의 특성을 식별하였다. 그리고 서비스 기반 어플리케이션을 개발하는 일반적이고 대표적인 프로세스로부터 4가지의 품질속성을 도출하였다; 이해성은 서비스 사용자들이 얼마나 효과적으로 서비스 기능을 이해할 수 있는지를 측정하고, 홍보성은 서비스 사용자가 얼마나 쉽게 서비스를 발견할 수 있는지를 측정하고, 적응성은 큰 어려움 없이 서비스 사용자가 얼마나 효율적으로 서비스를 커스터마이즈 할 수 있는지를 측정하고, 조립성은 비스니스 프로세스에서 다른 서비스와 얼마 쉽게 결합할 수 있는지를 측정한다. 품질속성을 기반으로 4가지의 품질속성을 측정하기 위해 메트릭을 정량적으로 정의하였다. 본 논문에서는 3가지의 맵 서비스를 재사용성 평가에 적용하여 사례연구의 결과를 보였다. 제안된 평가 모델은 클라우드 서비스 품질평가에 실용적으로 적용할 수 있고 정량적인 방법으로 재사용성 향상에 적용할 수 있다.

참 고 문 헌

[1] Gillett, F., "Future View: The New tech Ecosystems of Cloud, Cloud Services, And Cloud Computing," Making Leaders Successful Every Day, FORRESTER Research, August, 2008.

[2] Frakes, W. and Isoda, S., "Success factors of systematic reuse," IEEE Software, Vol.11, No.5, September, 1994.

[3] Software Engineering－Product Quality－Part 1: Quality Model. ISO/IEC 9126-1, June, 2001.

[4] Software Engineering－Product Quality－Part 2: External Metrics. ISO/IEC TR 9126-3, July, 2003.

[5] Choi, S.W., Her, J.S., and Kim, S.D., "QoS Metrics for Evaluating Services from the Perspective of Service Providers," *In Proceedings of IEEE International Conference on e-Business Engineering (ICEBE 2007)*, pp.622-625, 2007.

[6] Washizaki, H. et al., "A metrics suite for measuring reusability of software components," *In proceedings of METRICS'03*, September, 2003, pp.211 - 223.

[7] Lee, J.Y., Lee. J.W., Cheun, D.W., and Kim, S.D., "A Quality Model for Evaluating Software-as-a-Service in Cloud Computing," *In Proceedings of the 7th ACIS International Conference on Software Engineering Research, Management, and Applications (SERA 2009)*, pp.261-266, 2009.

[8] Aymerich, F.M., Fenu, G., and Surchis, S., "An Approach to a Cloud Computing Network," *In Proceedings of the 1st International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2008)*, pp.113-118, 2008.

[9] Wan, L., Tao, J., and Kunze, M., “Scientific Cloud Computing: Early Definition and Experience,” *In Proceedings of the 10th International Conference on High Performance Computing and Communications (HPCC 2008)*, pp.825-830, 2008.

[10] Zhang, L., Li, H., and Lam, H., “Services Computing: Grid Applications for Today,” *IEEE IT Professional Journal*, Vol.6, pp.5-6, 2004.

[11] Lee, J.Y. and Kim S.D., "Software Approaches to Assuring High Scalability in Cloud Computing," *In Proceedings of the 7th IEEE International Conference on e-Business Engineering (ICEBE 2010)*, pp.300-306, 2010.

[12] McCall, J.A., Richards, P.K., and Walters, G.F., Factors in Software Quality, *US Rome Air Development Center Reports*, Vol.I, II, III, RADC-TR-77-369, 1977.

[13] Her, J.S., La, H.J., and Kim, S.D., “A Formal Approach to devising a Practical Method for Modeling Reusable Services,” *In Proceedings of the 7th IEEE International Conference on e-Business Engineering (ICEBE 2008)*, pp.221-228, 2008.

[14] Frakes, W. and Terry, C., “Software Reuse: Metrics and Models,” *ACM Computing Surveys*, Vol.28, No.2, June, 1996.

[15] Kim, S.D., "Software Reusability," *Wiley Encyclopedia of Computer Science and Engineering*, Vol.4, pp.2679-2689, 2009.

[16] Tian, J., S*oftware Quality Engineering*, Wiley Inter-Science, 2005.

[17] B. Kitchenham, S. Pfleeger, N. Fenton, “Towards a framework for software measurement validation,” *IEEE Transactions on Software Engineering 21* (12) (1995).

[18] L. Ejiogu, *Software Engineering with Formal Metrics*, QED Publishing, 1991.

[19] IEEE standard for a software quality metrics methodology, IEEE Std. 1061-1998, 1998.

오 상 헌

e-mail : sanghun1004@gmail.com
2004년 건양대학교 정보전산학과(공학사)
2006년 숭실대학교 컴퓨터학과(공학석사)
2006년~현 재 숭실대학교 컴퓨터학과 박사수료
관심분야 : 모바일 컴퓨팅 (Mobile Computing), 서비스 지향 컴퓨팅 (Service Oriented Computing), 클라우드 컴퓨팅(Cloud Computing), 품질 공학(Quality Engineering)

라 현 정

e-mail : hjla80@gmail.com
2003년 경희대학교 우주과학과(이학사)
2006년 숭실대학교 컴퓨터학과(공학석사)
2011년 숭실대학교 컴퓨터학과(공학박사)
2011년~현 재 숭실대학교 모바일 서비스 소프트웨어공학센터 연구교수
관심분야 : 서비스 지향 컴퓨팅(Service-Oriented Computing), 소프트웨어 아키텍처(Software Architecture), 모바일 컴퓨팅(Mobile Computing)

김 수 동

e-mail : sdkim777@gmail.com
1984년 Northeast Missouri State University 전산학(학사)
1988년/1991년 The University of Iowa 전산학(석사/박사)
1991년~1993년 한국통신 연구개발단 선임연구원
1994년~1995년 현대전자 소프트웨어연구소 책임연구원
1995년 9월~현 재 숭실대학교 컴퓨터학부 교수
관심분야 : 객체지향S/W공학, 소프트웨어 아키텍처(Software Architecture), 클라우드 컴퓨팅(Cloud Computing), 모바일 컴퓨팅(Mobile Computing)