유비쿼터스 서비스 온톨로지를 위한 설계 품질 모델

(A Design-phase Quality Model for Ubiquitous Service Ontology)

이 미 연 † 박 승 수 †† 이 정 원 †††
(Meeyeon Lee) (Seung Soo Park) (Jung-Won Lee)

요 약 유비쿼터스 컴퓨팅 환경에서 사용자에게 서비스를 자동으로 제공하기 위해 사용되는 동적 서비스 합성은 효과적인 서비스 기술 및 모델링 기법이 전제되어야 한다. 이를 위해 이전 연구에서는 유비쿼터스 환경 내의 기기의 기능을 추상화한 'u-서비스'라는 개념을 제안하였고, 온톨로지로 구조화하는 메커니즘과 u-서비스의 특성을 표현하기 위한 기술 규격을 정립한 바 있다. 하지만 u-서비스 온톨로지 설계 시에 효율성을 검증하기 위한 체계나 기준을 제공하지는 못하고 있다. 또한 기존의 소프트웨어 제품이나 컴퓨팅 시스템의 품질 평가 모델은 유비쿼터스 서비스의 특성을 고려하지 못하므로 직접 적용하기에도 적합하지 않다. 따라서 본 연구에서는 이전 연구에서의 u-서비스 온톨로지 구축 방법론을 바탕으로 효과적인 유비쿼터스 서비스 온톨로지를 설계하기 위한 품질 평가 모델을 제안한다. 유비쿼터스 서비스 온톨로지의 특성을 반영하여 모델링 목표와 설계 단계에서의 평가 지표를 도출하고 각 품질 속성을 정량화하기 위한 메트릭을 정의한다. 이전 연구에서 구축한 u-서비스 온톨로지에 적용한 결과, 제안한 품질 평가 모델을 통해 유비쿼터스 서비스 온톨로지의 적합성을 다각도로 분석하고 설계 개선을 위한 모델링 원칙을 제시할 수 있음을 보였다.

키워드 : 유비쿼터스 컴퓨팅, 서비스 온톨로지, 품질 모델, 품질 메트릭, 서비스 지향 아키텍처

Abstract Effective service description and modeling methodologies are essential for dynamic service composition to provide autonomous services for users in ubiquitous computing environments. In our previous research, we proposed a 'u-Service' as an abstract and structured concept for operations of devices in ubiquitous environments. In addition, we established the mechanism to structure u-Services as an ontology and the description specification to represent attributes of u-Services. However, it did not provide enough methods or standards to analyze and evaluate the effectiveness of the u-Service ontology in the design time. Since existing quality models for software products or computing systems cannot consider characteristics of ubiquitous services, they are not suitable for ubiquitous service ontology. Therefore, in this paper, we propose a quality evaluation model to design and modeling a good ubiquitous service ontology, based on our u-Service ontology building process. We extract modeling goals and evaluation indicators according to characteristics of ubiquitous service ontology, and establish quality metrics to quantify each quality sub-characteristics. The experiment result of the proposed quality evaluation model for u-Service ontologies which are

· 이 연구는 2009학년도 아주대학교 교내연구비 지원(20094720)에 의하여 이루어졌음
· 본 연구는 지식경제 프론티어 기술개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스 컴퓨팅 및 네트워크 원천기반기술개발사업의 10C2-T3-10M 과제로 지원된 것임

† 학생회원 : 이화여자대학교 컴퓨터정보통신공학과
ailmy@ewhain.net
†† 정 회 원 : 이화여자대학교 컴퓨터정보통신공학과 교수
sspark@ewha.ac.kr
††† 종신회원 : 아주대학교 전자공학부 교수
jungwony@ajou.ac.kr

논문접수 : 2010년 2월 8일
심사완료 : 2010년 4월 6일

정보과학회논문지 : 소프트웨어 및 응용 제37권 제6호(2010.6)

constructed for our previous works shows that we can analyze the design of ubiquitous service ontology from various angles, and indicate recommendations for improvement.

Key words : Ubiquitous Computing, Service Ontology, Quality Model, Quality Metric, SOA (Service-Oriented Architecture)

1. 서 론

유비쿼터스 컴퓨팅은 주변 환경과 사용자의 가변적인 상황(context)에 따라 지능적이고 자율적인 서비스를 제공함으로써 특정 목표의 달성을 지원하는 것을 목표로 한다. 즉, 사용자의 위치, 사용 가능한 기기 등과 같은 상황이 동적으로 변하는 유비쿼터스 환경에서 사용자가 원하는 환경을 조성하거나 특정 이벤트를 감지하고 처리하는 등의 자율 서비스를 제공할 수 있어야 한다. 적합한 서비스를 제공하기 위한 핵심 기술로는 정확한 상황 인식(context-awareness)과 그에 필요한 동적 서비스의 발견(dynamic service discovery) 및 합성(composition)[1,2] 등이 있다. 이를 위하여 유비쿼터스 환경의 상황 정보와 서비스 정보를 효과적으로 기술하고 모델링하는 것이 매우 중요하다. 이전 연구[3,4]에서는 그런 점에 착안하여 유비쿼터스 환경 내의 기능(operation; function)을 'u-서비스(u-Service)'라는 개념으로 추상화하고, SOA(Service-Oriented Architecture)[5]의 서비스 레이어링(service layering) 기법을 적용하여 이들을 추상화의 정도에 따라 계층으로 분류한 u-서비스 온톨로지를 제안하였다.

그런데 이 과정에서 u-서비스 온톨로지 설계를 위한 체계적인 기준이나 평가 지표 등이 절실하게 요구되었고, 기존의 일반 온톨로지에 적용되는 평가 모델은 직접 적용이 적절하지 못하다는 사실을 확인할 수 있었다. 일반적으로 온톨로지는 지식의 효율적인 재사용과 공유, 추론 시스템과 같은 응용 프로그램의 성능에 중대한 영향을 미치기 때문에 적절한 모델링과 그에 대한 평가가 필수적이다[6,7]. 다양한 유비쿼터스 환경의 u-서비스 온톨로지의 경우 그 효율성은 실제 환경에서 응용 시스템을 통해 테스트해보기 전에는 확인할 수 없다. 게다가 유비쿼터스 환경에서 발생 가능한 모든 상황을 예측하여 테스트하는 것도 불가능하므로 u-서비스 온톨로지 설계의 전반적인 평가는 매우 어렵다고 할 수 있다. 또한 비즈니스 도메인의 서비스를 대상으로 한 SOA의 설계 원리나 다양한 기존 품질 평가 모델을 유비쿼터스 환경에 직접 적용하는 것은 한계가 있다.

따라서 본 논문에서는 유비쿼터스 서비스 온톨로지에 특화된 품질 평가 모델을 개발하기 위해 소프트웨어 제품, SOA 시스템, 온톨로지 등 다양한 분야의 기존 품질 속성을 기반으로 유비쿼터스 서비스 모델링 평가에 적합한 속성을 도출하고 측정 메트릭을 제안하려 한다.

본 논문의 구성은 다음과 같다. 2장에서는 이전 연구에서 제안한 u-서비스 온톨로지에 대해 설명하고, 다양한 분야에서의 품질 평가 방법에 대해 간략히 소개한다. 3장에서는 기존의 품질 모델을 기반으로 가변적인 상황에 따라 동적으로 제공되어야 하는 유비쿼터스 서비스의 특성에 따라 품질 속성과 세부 속성을 도출하고, 각 항목을 정량적으로 수치화하기 위한 평가 메트릭을 정의한다. 4장에서는 u-홈과 공공 도메인에 대해 구축한 u-서비스 온톨로지를 평가 모델을 통해 평가한 결과를 비교 및 분석하고, 5장에서 결론 및 향후 연구방향을 제시한다.

2. 관련 연구

다양한 대상을 평가하기 위한 품질 모델들 중에서 대표적인 모델들을 표 1과 같이 정리할 수 있다.

ISO/IEC 9126[8-11]은 소프트웨어 제품의 품질을 평가하기 위한 6개의 품질 특성을 정의하고 27개의 부특성을 개발/테스트/사용 시에 정량적으로 측정할 수 있는 메트릭들을 제시하고 있다. 이 표준은 다양한 분야에서의 품질 모델 개발을 위한 기초가 되고 있다. 이를 기반으로 객체 지향 패러다임의 소프트웨어를 평가하기 위해 객체 지향적 특성을 반영하여 재정의한 모델도 제안되었다. QMOOD(Quality Model for Object-Oriented Design)[12]는 객체 지향적 설계 컴포넌트 도출, 메트릭 선정, 설계 속성 정의, 품질 속성 정의의 4단계를 통해 개발된 품질 모델이다. 이 모델은 객체 지향 시스템의 객체, 클래스, 메소드와 같은 개념들과 그들의 속성을 반영하고 있다. 또한 최근에 주목 받고 있는 SOA 분야에서는 서비스 지향적 특성에 맞게 시스템을 설계했는지를 평가하기 위한 연구도 진행되고 있다. SOA 설계 품질 모델[13]은 QMOOD의 4단계 개발 방법론을 도입하여 SOA 시스템에 적용하였다. 이 연구에서 제안된 품질 모델은 웹 서비스, 웹 서비스의 오퍼레이션, 그들 간에 주고 받는 메시지 등과 같은 요소들을 평가할 수 있는 항목들을 정의하고 있다. 하지만 위의 모델들은 각 평가 대상 고유의 특성만을 고려하고 있어서 유비쿼터스 서비스 온톨로지의 평가에 부적합하고 직접적인 적용도 어렵다.

UCQM[14]은 ISO/IEC 9126 모델 항목에 덧붙여 외부 환경 변화 감지, 유비쿼터스 접근, 다대다 접속 등을 비롯한 8가지의 유비쿼터스 컴퓨팅 시스템의 특성을 감

표 1 기존 품질 모델 비교

기존 품질 모델	특징	평가 대상	평가 시기
ISO/IEC 9126	S/W 제품의 품질 평가 항목 및 메트릭에 대한 표준	소프트웨어 제품	개발/테스트/사용 시
QMOOD	객체 지향(OO) 패러다임에 따라 S/W를 개발할 경우, 설계 단계에서 품질을 평가하기 위한 모델	객체 지향 소프트웨어	설계 시
SOA 시스템의 설계 품질 모델	ISO와 QMOOD 모델을 SOA 시스템에 적합하도록 특화한 모델	서비스 지향 시스템	설계 시
UCQM	ISO의 품질 모델을 유비쿼터스 시스템에 맞게 특화한 모델	유비쿼터스 컴퓨팅 시스템	실행 시
Web Service QoS	웹 서비스의 실행 시의 성능 평가를 위한 모델	웹 서비스	실행 시
Ontology 평가 모델	도메인(선언적) 온톨로지를 평가하기 위한 모델	온톨로지	설계 시

안한 품질 모델을 제시하고 있다. 이 모델을 통해 집, 회사, 마트 등과 같은 유비쿼터스 환경에서의 시스템 성능(기능성, 사용성, 이식성, 이동성)을 평가할 수 있다. 웹서비스의 QoS(Quality of Service)[15,16]는 웹 서비스의 성능 평가를 위해 등장한 모델들이다. 이 모델들도 평가 대상에 따라 특화되어 있고 시스템의 오류율, 시스템의 사용 용이성, 기능 제공 실패율, 사용자의 만족도 등이나 개별적인 웹 서비스에 대한 네트워크 상에서의 작업 처리 속도와 같은 실행 성능 평가를 대상으로 하고 있다.

또한 온톨로지 평가를 위한 대부분의 연구[6,7,17]는 선언적 지식을 표현하는 도메인 온톨로지를 그 대상으로 한다. 도메인 온톨로지는 특정 도메인 내의 용어(vocabulary)나 개념(concept)과 그들간의 의미적인 관계를 표현한다. 반면에 서비스 온톨로지는 표현 단위가 '서비스'로서 선언적 지식이 아니라 특정 기능을 제공하는 일련의 절차 또는 행위를 표현하기 위해 세부 프로세스와 실행 순서, 실행에 관련된 입력/출력 파라미터와 같은 정보를 포함한다. 예를 들어, 사용자 정보(이름, 국적 등), 항공사 정보(비행 일정 등) 등은 도메인 온톨로지로 표현하고, 항공 티켓 구매 절차(회원 로그인, 항공편 검색, 결제 등)는 서비스 온톨로지에 포함된다. 이 두 타입의 온톨로지를 위한 표준 언어도 OWL(Web Ontology Language)[18]과 OWL-S(OWL for Services)[19]의 두 종류의 표준이 제정되어 있다. 두 타입의 온톨로지가 표현하는 대상이 서로 다르기 때문에 평가하는 요소가 달라야 하는 것은 당연하다.

3. u-서비스 온톨로지(u-Service Ontology)

일반적인 서비스 온톨로지는 웹 서비스나 비즈니스 서비스를 표현 대상으로 하며 유비쿼터스 환경에서의 서비스를 대상으로 하지 않고 있다. 따라서 본 연구의 이전 연구[3,4]에서는 유비쿼터스 서비스를 체계적으로 도출하고 표현하여 구축하기 위한 방법을 제안한 바 있다. u-서비스 온톨로지는 유비쿼터스 서비스를 'u-서비스'라는 개념으로 정의하고 이들을 효과적으로 표현 및 구조화한 서비스 온톨로지이다. 'u-서비스'는 유비쿼터스 환경에 존재하는 다양한 기기들의 기능을 추상화한 하나의 단위이다. u-서비스 온톨로지 구축은 우선 서비스를 제공하고자 하는 유비쿼터스 컴퓨팅 환경을 대표하는 상황들을 정의하고 각 도메인 별, 상황 별로 등장하는 기기와 기능들을 추출하는 작업으로 시작된다. 풍부한 u-서비스 도출을 위해, 대표 상황에 직접적으로 등장하지는 않지만 해당 도메인에서 제공될 수 있는 기능들도 추가하고 이들을 u-서비스로 추상화한다. u-서비스는 추상화 정도에 따라 3가지 레벨로 분류되며 실시간에 서비스의 동적 발견 및 조합을 위해 고려되어야 하는 정보를 정의한 u-서비스 기술 규격에 맞게 기술된다. u-서비스 기술 규격[3,4]의 14가지 항목 중에서 중요한 항목들은 다음과 같다.

AbsLevel: u-서비스의 3가지 추상화 레벨. 'ABSTRACT'는 최상위 추상화 레벨이고, 'COMPOSITE'은 중간 레벨, 'ATOMIC'은 가장 하위 레벨의 단말 u-서비스이다.

Child: ABSTRACT과 COMPOSITE u-서비스는 하위에 1개 이상의 다른 u-서비스들을 *Sequence, Choice* 와 같은 결합 속성을 통해 포함할 수 있다.

Object: 해당 u-서비스를 제공할 수 있는 플랫폼 혹은 스마트 기기.

Precondition-Status: 해당 u-서비스를 실행할 기기의 전제 상태.

Effect - Functional/Environmental Effect: 해당 u-서비스를 실행한 후에 발생하는 기능적('이미지 보여주기', '위험 경고' 등) 또는 환경적('소음 조절', '온도 상승' 등) 효과.

모든 u-서비스는 3가지 추상화 레벨로 분류되고 '기능적/환경적 Effect'를 기준으로 그룹화되어 u-서비스 온톨로지 내에서 계층 구조를 이루게 된다. 즉, Effect에 기반하여 도출된 최상위의 ABSTRACT u-서비스를 루트로 해서 특정 Effect와 연관된 u-서비스들을 묶어서 하나의 그룹을 형성한다.

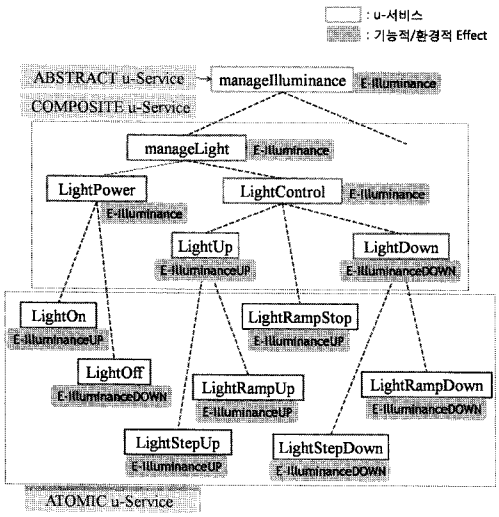


그림 1 u-서비스 온톨로지의 구조(조도 관련 Effect인 'E-Illuminance' 그룹의 일부분)

그림 1은 u-서비스 온톨로지의 구조를 트리 구조로 도식화한 그림으로 환경적 Effect의 하나인 조도 조절 Effect(E-Illuminance) 그룹의 일부분을 보이고 있다. 이전 연구를 통해 구축한 이 u-서비스 온톨로지는 u-홈과 공공 안전 관리 도메인에서의 수면 환경 조성, 불법 침입 감지 및 추적 등과 같은 유비쿼터스 서비스 제공을 위한 지식 베이스로 활용되었다.

4. 유비쿼터스 서비스 온톨로지를 위한 품질 모델 개발

유비쿼터스 서비스 온톨로지의 평가를 위하여 평가 대상의 특성, 평가 대상이 목표로 하는 도메인의 특성, 평가 적용 시의 특성을 모두 고려할 수 있는 새로운 평가 방법이 필요하다. 그러므로 기존의 품질 모델 개발 방법론에 따라 유비쿼터스 서비스 온톨로지의 품질 속성 및 세부 특성과 이를 정량화할 수 있는 품질 메트릭을 도출한다.

4.1 품질 모델의 필요성

"좋은" 유비쿼터스 서비스 온톨로지는 동적 서비스 합성이 가능하도록 서비스의 발견, 대체 서비스 발견, 오버로딩 기법[3,4]을 지원할 수 있어야 한다. 효과적인 유비쿼터스 서비스 온톨로지를 개발하기 위해서는 표 2의 첫 번째 열에 나열한 3가지 영역의 특성을 반영해야 한다. 즉, u-서비스 온톨로지는 유비쿼터스 환경을 대상으로 하고 SOA의 서비스 모델링 기법을 도입하여 온톨로지로 구축함으로써 효과적인 유비쿼터스 서비스 온톨로지로서의 역할을 할 수 있도록 하였다. 표 2의 두 번째 열을 보면, 유비쿼터스 서비스 온톨로지의 품질 평가 모델은 유비쿼터스 도메인에 적합하도록 서비스가 도출되고 구조화되었는지, 온톨로지의 장점을 잘 반영하였는지, SOA의 도입 목적에 맞게 설계되었는지를 평가할 수 있어야 한다. 유비쿼터스 서비스를 추출하고 계층화

표 2 u-서비스 온톨로지 설계를 위한 기반 영역의 특성에 따른 유비쿼터스 서비스 온톨로지의 설계 목표와 속성

설계 기반 영역	특성	설계 목표	속성
유비쿼터스 환경	유비쿼터스 서비스 온톨로지의 모델링 대상은 웹 서비스나 SOA가 주 대상으로 하는 비즈니스 도메인에서의 서비스가 아니라, 유비쿼터스 환경에서 제공될 수 있는 상황인식 서비스들이다. 따라서 이동성, 이질성 등과 같은 유비쿼터스 환경의 특성에 적합하게 설계하여야 한다.	• 위치, 현재의 환경 상태, 기기 상태와 같은 주변 상황(context)을 반영할 수 있어야 한다.	상황인식, 다형성
		• 유비쿼터스 서비스의 동적 발견, 동적 조합, 대체 서비스 발견이 가능하도록 필요한 서비스의 정보를 기술하여야 한다.	다형성
		• 비즈니스 서비스와 달리 복합 서비스의 하위 서비스들과 실행 순서가 고정되지 않으므로 워크플로우 형태의 정적인 서비스를 정의하지 않도록 한다.	조합성
		• 동적으로 변화하고 예측 불가능한 환경이므로, 유사한 효과를 발생할 수 있는 서비스가 다양하도록, 다양한 기기에 의해 제공될 수 있도록 설계한다.	대체성, 독립성
온톨로지	온톨로지라는 지식 표현 구조는 도메인 내의 개념을 표현하기 위한 정립된 체계를 정의하고 그들의 연관 관계를 의미적으로 표현하도록 한다.	• 유비쿼터스 환경 내의 목표 도메인에 대한 상황 정보를 풍부하게 체계적으로 표현하도록 한다.	상황인식, 지원범위, 풍부성
		• 유비쿼터스 서비스들을 단순하게 나열하기보다, 유사한 효과를 발생시킨다던가 유사한 종류의 기기에 의해 실행 가능하다던가 하는 등의 서비스들 간의 의미적인 관계를 표현하도록 한다.	대체성, 계층구조
SOA	u-서비스 온톨로지 설계 시에 적용했던 SOA의 서비스 모델링 기법을 효과적으로 반영하여 설계하여야 한다.	• SOA의 재사용성, 발견성, 자율성, 서비스 계약, 느슨한 결합과 같은 서비스 설계 원칙을 준수하여야 한다.	재사용성, 자율성, 느슨한 결합, 발견성, 서비스 계약, 추상화, 무상태 유지, 조합성
		• SOA의 서비스 레이어링 기법을 참고하되, 유비쿼터스 서비스의 특성에 적합하게 분류 및 계층화하여야 한다.	계층 구조

하는 설계 단계에서부터 위의 3가지 영역의 특성을 적절하게 반영하기 위한 설계 목표를 표 2의 세 번째 열과 같이 정리할 수 있고, 이들은 유비쿼터스 서비스 온톨로지의 평가 항목을 도출하는 기준으로서 4.2절에서 자세히 설명한다.

4.2 품질 세부 특성 정의

품질 세부 특성(Quality Sub-characteristics, 이하 QSC)은 개념적인 의미라기보다, 평가 대상을 구성하는 요소들로부터 직접적으로 측정 가능한 실질적인 속성이다[8,12]. 세부 특성을 도출하기 위해 앞 절에서 설명한 유비쿼터스 서비스 온톨로지의 설계 목표(표 2의 세 번째 열)를 표 2의 네 번째 열에 보인 것과 같은 속성들과 관련지을 수 있다. 이 속성들은 3가지 기반 영역의 특성들로부터 정의한 설계 지침들을 직접적으로 나타내는 용어들이다. 즉, 표 2의 첫 번째 행의 경우 유비쿼터스 환경이라는 도메인의 특성 상 유비쿼터스 서비스 온톨로지가 주변 상황에 대한 정보를 반영할 수 있어야 한다는 설계 목표는 '상황인식'과 '다형성'이라는 속성으로 지칭할 수 있다. '상황인식'은 유비쿼터스 서비스 온톨로지가 유비쿼터스 환경의 상황 정보를 표현할 수 있는 체계를 제공하고, 유비쿼터스 서비스도 상황 정보를 고려하여 선택될 수 있도록 기술되어야 한다는 속성을 의미한다. '다형성'은 유비쿼터스 환경에서는 같은 효과를 발생시키기 위해서라고 하더라도 실제 실행될 서비스가 상황에 따라 달라지므로 유비쿼터스 서비스 온톨로지를 통해 실행 시의 상황 정보를 고려하여 최적의 서비스를 발견 및 선택할 수 있어야 함을 뜻한다.

그림 2는 표 2의 설계 기반 영역들로부터 도출한 속성들로부터 품질 세부 특성과 품질 속성을 정의하는 과정과 연관성을 보이고 있다. 표 2의 네 번째 열의 15가지의 속성들은 개념적인 설계 목표로부터 도출하였기 때문에 의미가 중복되기도 하고 정량적으로 측정하기 어려운 속성들도 있다. 따라서 이러한 속성들 중에서 다음의 5개 속성들을 배제하였고, 그림 2에서 점선으로 표시한다.

- 상황인식: 이 속성은 유비쿼터스 서비스가 추구하는 가장 근본적인 목표로서, 단일 항목으로 정량화하기 어렵다. 즉, 모든 세부 특성들이 상황인식을 통한 동적 서비스 제공 가능성을 평가하고 있으며, 표현된 상황정보의 양적인 면은 '상황정보 풍부성' 항목으로 측정 가능하다.
- 발견성 & 서비스 계약: SOA의 서비스 설계 원칙으로, 서비스 정보를 기술한 계약(contract)을 공개함으로써 쉽게 발견될 수 있도록 해야 함을 뜻한다. 이 두 속성은 3장에서 언급한 u-서비스 기술 규격을 준수함으로써 자연스럽게 충족할 수 있기 때문에 별도의 평가 항목이 불필요하다.
- 추상화: 유비쿼터스 서비스라는 개념 자체가 '기기의 기능을 추상화하고 구조화한 단위'임을 의미한다. 단순한 기기의 기능인지 추상화된 유비쿼터스 서비스인지는 '독립성' 속성으로 측정할 수 있으므로 채택하지 않았다.
- 무상태 유지: 웹 서비스가 주고 받는 메시지의 처리와 같은 특성을 고려하는 속성으로 실시간의 상황에 대한 고려가 필수적인 유비쿼터스 서비스의 특성과 부합하지 않아 배제하였다.

다음의 속성은 각각 세부 항목으로 확장하여 대체하

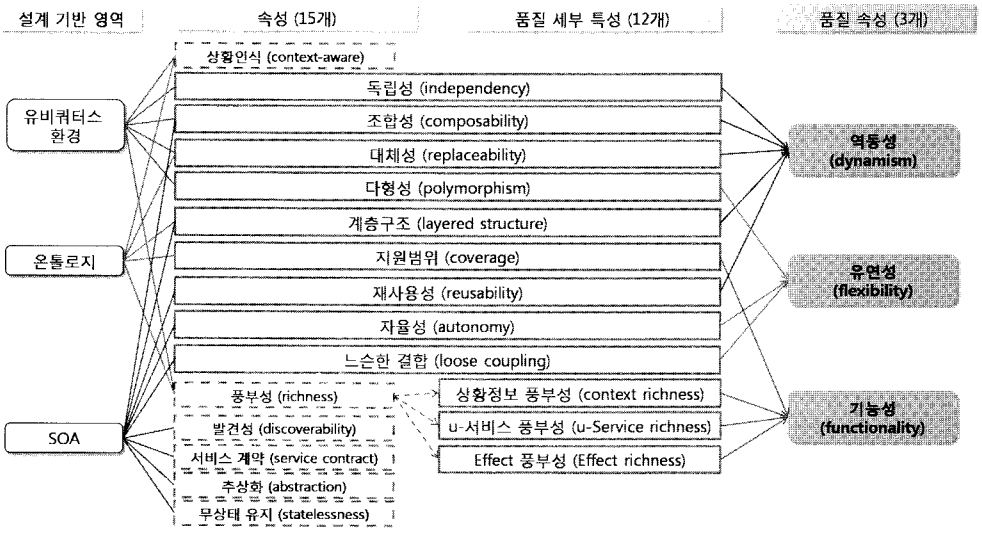


그림 2 품질 세부 특성 및 품질 속성 도출

였고, 그림 2에서 점선으로 표시한다.

- 풍부성: 도메인 정보와 유비쿼터스 서비스 정보의 양적인 면을 정량적으로 측정할 수 있는 세부 항목을 추가하여 3가지 항목으로 확장하였다.

최종적으로 선택한 12개의 품질 세부 특성은 그림 2와 표 3의 두 번째 열에 정리하였고, 각 세부 특성에 대해서는 4.4절에서 정량 메트릭과 같이 자세히 설명한다.

4.3 품질 속성 정의

평가 모델에 있어서의 품질 속성(Quality Characteristics, 이하 QC)은 평가 대상이 추구하는 특성 또는 목표이다[8,12]. 그림 2와 같이 유비쿼터스 서비스 온톨로지의 품질 평가 속성은 품질 세부 특성들을 관련성에 따라 분류하고 대표 속성을 선정하여 역동성, 유연성, 기능성의 3개 항목으로 분류하였다.

- **QC1: 역동성(Dynamism)** 유비쿼터스 서비스 온톨로지가 지원해야 하는 가장 중요한 속성으로서 유비쿼터스 환경에서 가장 적합한 서비스가 상황에 따라 동적으로 선택되고 상황 변화에 따라 대체될 수 있는 기반으로서의 역할을 해야 한다는 의미이다. 유비쿼터스 서비스들의 구조가 서비스의 상황에 따른 동적 발견/조합, 대체 서비스 발견과 같은 유비쿼터스 서비스의 특성을 잘 반영하고 있어야 하며(조합성, 대체성, 독립성), 이러한 동적 기능에 활용 가능한 관계가 효과적으로 표현되는 구조로(계층 구조, 재사용성) 설계되어야 한다.
- **QC2: 유연성(Flexibility)** 유비쿼터스 서비스들의 추상화 정도에 따른 계층 구조와 서비스들간의 관계가 정적이거나 귀속적이지 않아야 한다는 특성을 나타낸다. 서비스들이 강한 결속력으로 연관되어 있으면 실행 시에 서비스의 동적 선택 및 실행에 제한을 주게 되므로, 설계 단계에서 미리 방지하도록 해야 한다(느슨한 결합, 자율성). 또한 유비쿼터스 서비스의 특성상, 사용자의 추상적인 요구에도 적합한 서비스가 유연하게 선택되어 실행될 수 있는 구조를 가져야 한다는 의미도 포함하고 있다(다형성).
- **QC3: 기능성(Functionality)** 유비쿼터스 서비스 온톨로지의 역할을 충실히 수행할 수 있는 기본 바탕을 제공하느냐에 대한 평가 지표이다. 유비쿼터스 서비스 온톨로지는 유비쿼터스 환경에서 발생할 수 있는 상황을 최대한 지원할 수 있어야 하며, 유비쿼터스 서비스 자체뿐만 아니라 도메인과 상황에 대한 정보를 풍부하게 표현하고 있어야 한다(지원범위, 풍부성).

4.4 품질 메트릭 정의

각 세부 품질 속성(QSC)을 정량적으로 측정할 수 있는 공식으로서 품질 메트릭(Quality Metrics)을 정의하였고, 표 3의 세 번째 열에 개념적인 메트릭을 정리하였다. 세부 특성과 메트릭은 설계 상(design phase)의 효율성 평가만을 목표로 선정하였으며, 서비스 제공 시간, 실행 성공률, 실행 비용 등과 같은 실행 시(execution time)의 성능 평가는 배제한다. 유비쿼터스 서비스 온톨로지는 서비스 온톨로지 표준 언어인 OWL-S를 사용하기 때문에 OWL-S 구문으로부터 산출될 수 있는 메트릭도 각 세부 절에서 설명한다. 제안한 온톨로지 품질 모델이 이전 연구의 u-서비스 온톨로지 개발 방법론을 기반으로 정립되었기 때문에, 이후부터는 유비쿼터스 서비스 온톨로지의 표현 단위인 서비스를 'u-서비스'로 통칭한다.

4.4.1 QC1: 역동성(Dynamism)

유비쿼터스 서비스 온톨로지의 궁극적 목표라 할 수 있는 역동성은 유비쿼터스 서비스들의 전체적인 구조가 효과적으로 설계되었는지를 평가하며, 다음 5가지의 세부 속성과 각 메트릭을 사용한다.

- **QSC1: 조합성(Composability)**

SOA나 웹 서비스가 주로 다루는 비즈니스 서비스는 절차적 워크플로우(procedural workflow)가 예측 가능하고 중요하기 때문에 필요한 서비스를 실행 순서에 따라 미리 정의할 수 있다. 비즈니스 도메인에서의 '복합 서비스(composite service)'는 세부적인 기능을 수행할 수 있는 하위 서비스들이 실행 조건 및 순서에 따라 조합된 서비스를 의미한다. 하지만 유비쿼터스 도메인에서는 실행 순서나 절차가 큰 의미를 갖지 못하며 실시간의 상황 변화에 따라 필요한 하위 서비스가 달라지기 때문에 예측뿐만 아니라 미리 정의하기도 어렵다. 따라서 정적인 실행 절차를 정의하기보다는 상황을 고려해서 최적의 서비스를 찾아낼 수 있는 구조가 더 효과적일 것이다. 유비쿼터스 서비스 온톨로지의 설계에 있어서 복합 u-서비스를 다음과 같이 재정의하였다.

비절차적 조합(non-procedural composition)

조합을 구성하는 하위 서비스들의 실행 순서를 정적으로 명시하지 않는 조합의 타입.

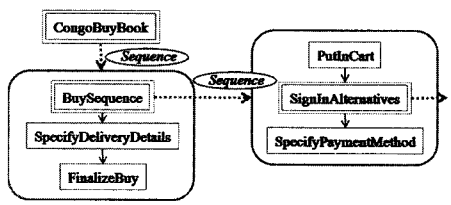
복합 u-서비스(composite u-Service)

특정 기능적 또는 환경적 효과(Effect)와 관련된 서비스들의 그룹으로서, 하위 서비스들로 구성되는 ABSTRACT과 COMPOSITE 타입의 u-서비스를 통틀어 지칭한다. 비절차적 조합 타입으로 설계하는 것이 효과적이다.

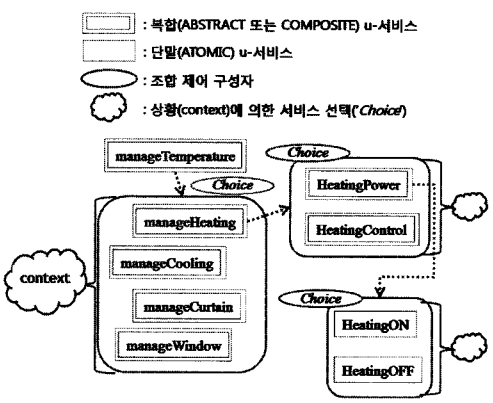
예를 들어, 그림 3(a)는 OWL-S의 대표적인 예제인 'Congo.com' 서비스[20]의 일부분을 도식화한 그림이다. 복합 서비스인 'CongoBuyBook' 서비스(온라인 책 구입 서비스)는 3개의 하위 서비스의 순차적인 실행으로 달성됨을 보이고 있다. 이와 달리, 유비쿼터스 서비스 온톨로지의 설계 상에서의 복합 서비스의 한 예인 'manage-

표 3 유비쿼터스 서비스 온톨로지의 품질 속성-세부 특성-메트릭

품질 속성(QC)	품질 세부 특성(QSC)	품질 메트릭 (개념적)	
역동성 (Dynamism)	조합성 (Composability)	비절차적 조합의 개수 / 전체 조합(복합 *u*-서비스)의 개수	(1)
	대체성 (Replaceability)	'*dynamic Effect*'의 개수 / 전체 Effect의 개수	(2)
	독립성 (Independency)	'*dynamic u*-서비스'의 개수 / 전체 *u*-서비스의 개수	(3)
	계층 구조 (Layered Structure)	모든 조합의 평균 계층 수 = $\frac{\sum \text{각 조합의 계층 수}}{\text{전체 조합(복합 } u\text{-서비스)의 개수}}$	(4)
	재사용성 (Reusability)	중복되는 *u*-서비스의 개수 / 전체 *u*-서비스의 개수	(5)
유연성 (Flexibility)	느슨한 결합 (Loose Coupling)	하위 *u*-서비스가 2개 이상인 조합의 개수 / 전체 조합(복합 *u*-서비스)의 개수	(6)
	자율성 (Autonomy)	필수 선행 *u*-서비스가 없는 *u*-서비스의 개수 / 전체 *u*-서비스의 개수	(7)
	다형성 (Polymorphism)	$\frac{\sum \text{각 복합 u-서비스의 다형 u-서비스의 개수}}{\text{전체 복합 } u\text{-서비스의 개수}}$	(8)
기능성 (Functionality)	지원범위 (Coverage)	*u*-서비스로 지원가능한 상황 개수 / 대표상황에 등장하는 기능(*operations*)의 개수	(9)
	상황정보 풍부성 (Context Richness)	전체 클래스(*Classes*)의 개수	(10)
	u-서비스 풍부성 (u-Service Richness)	전체 u-서비스의 개수	(11)
	Effect 풍부성 (Effect Richness)	전체 Effect의 개수	(12)



(a) 비즈니스 서비스의 예



(b) u-서비스의 예

그림 3 두 도메인에서의 복합 서비스의 예

Temperature' 서비스는 '온도 조절' 효과를 발생할 수 있는 서비스들의 그룹으로 형성된다(그림 3(b)). 즉, 이 복합 서비스는 실제 서비스 제공 시간이나 위치, 배치되어 있는 기기의 상태 등에 따라 하위의 서비스들 중에서 가장 적합한 서비스를 선택하여 제공할 수 있도록 하는 체계로 구성되어 있다.

웹 서비스 기술을 위한 표준 서비스 온톨로지 언어인 OWL-S에서는 복합 프로세스의 실행 순서 및 조건을 표현하기 위해 조합 제어 구성자(control constructs)를 사용한다. 이 총 8개의 제어 구성자를 절차 정의 여부에 따라 두 종류로 분류하였다. 즉, 다음과 같이, 하위 서비스들의 실행 순서가 포함되도록 하는 구성자와 그렇지 않은 구성자로 나누었다. 예를 들어, '*If-Then-Else*'는 조건에 따라 실행 순서가 정해지고, '*Sequence*'는 하위 서비스들이 모두 순서대로 실행되어야 한다. 반면에 '*Any-Order*'는 하위 서비스들이 모두 실행되어야 하지만 실행 순서는 정의하지 않으며, '*Choice*'는 실행 순서도 지정하지 않고 하위 서비스들 중에서 적합한 하나의 프로세스만 선택함으로써 유비쿼터스 서비스 온톨로지에서의 복합 서비스의 의미에 부합된다.

• 절차적 조합을 정의하는 제어 구성자: *If-Then-Else,*

Repeat-Until, Repeat-While, Sequence, Split, Split-Join

- 비절차적 조합을 정의하는 제어 구성자: *Any-Order, Choice*

따라서 유비쿼터스 서비스 온톨로지에서 복합 서비스는 워크플로우와 같은 절차적인 정적 조합을 지양하여야 하므로 '*Any-Order*' 또는 '*Choice*' 제어 구성자로 조합되도록 설계하여야 한다. 이 특성을 준수하여 이후의 예제를 설명하는 그림들에서(그림 4 ~ 그림 6) 조합의 상위 서비스와 하위 서비스를 연결하는 점선은 모두 '*Any-Order*' 또는 '*Choice*' 타입으로 가정한다.

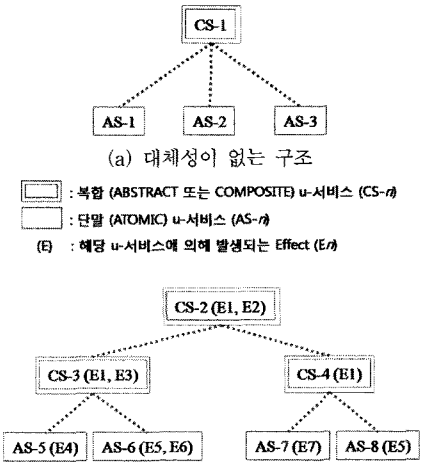
표 3의 메트릭 (1)을 통해 측정할 수 있으며, 모든 조합의 개수에 대한 비절차적 조합 즉, '*Any-Order*'와 '*Choice*' 조합의 비율로 계산한다. 다음은 OWL-S 언어의 구문으로 표현한 식이다. 유비쿼터스 서비스 온톨로지에서의 ABSTRACT과 COMPOSITE u-서비스는 OWL-S의 'composite process'로 표현되므로 정의된 인스턴스의 개수를 산출하고, OWL-S의 'control constructs' 중에서 '*Any-Order*'와 '*Choice*'의 인스턴스 개수를 산출하여 계산한다. 0 ~ 1의 값이 측정될 수 있고 측정치가 1에 가까울수록 높은 조합성을 지원한다고 볼 수 있다.

$$\text{조합성} = \frac{\text{'process:Choice'의 인스턴스의 개수} + \text{'process:Any-Order'의 인스턴스의 개수}}{\text{'process:CompositeProcess'의 인스턴스의 개수}} \quad (1\text{-}1)$$

• QSC2: 대체성(Replaceability)

유비쿼터스 환경은 위치나 기기와 같이 가변적인 요소가 많기 때문에 실행할 기기가 배치되어 있지 않거나 이미 서비스가 제공중임에도 불구하고 만족스럽지 않은 상황과 같은 다양한 환경과 조건에서도 사용자가 원하는 서비스를 지속적으로 자연스럽게 제공할 수 있어야 한다. 이를 가능케 하기 위해 유비쿼터스 서비스 온톨로지에서는 모든 u-서비스를 '(기능적/환경적) Effect'를 기초로 구조화하는데, 3장에서 언급하였듯이 Effect는 '서비스에 의해 발생되는 효과 또는 서비스 실행으로 조성되는 특정 상황'을 뜻한다. 이러한 구조를 활용함으로써 동일한 효과를 발생할 수 있는 서비스들을 파악하여 대체할 수 있도록 하여야 한다.

예를 들어, 그림 4(a)와 같이 Effect 정보가 전혀 없거나 Effect와 u-서비스가 일대 일 관계로 설계되어 있다면 해당 유비쿼터스 서비스 온톨로지 내에서는 서비스들 간의 대체를 지원할 수 없어 서비스 제공 확률과 만족도가 낮아지게 된다. 반면에 (b)의 경우에는 Effect에 따라 그룹화되어 있고 동일한 Effect를 발생하는 서비스들을 식별할 수 있으므로 한 서비스가 실행 불가능할 때 다른 서비스로 교체가 가능하게 된다(Effect '*E5*'를 발생할 수 있는 '*AS-6*'과 '*AS-8*' 서비스 간의 대체).



(a) 대체성이 없는 구조

(b) 대체성이 있는 구조

그림 4 유비쿼터스 서비스 온톨로지의 대체성

대체성을 측정하기 위한 메트릭인 표 3의 (2)에서 'dynamic Effect'는 다음과 같이 정의하고 위의 그림 4(b)에서는 *E1*과 *E5*가 'dynamic Effect'가 된다.

dynamic Effect

다중(2개 이상의) u-서비스에 의해 발현될 수 있는 Effect.

아래의 식에서 Effect는 SWRL 규칙의 이름으로 정의하므로 인스턴스의 개수를 산출한다. 조건 1은 Effect를 발생시키는 u-서비스 즉, OWL-S의 'atomic process'의 인스턴스 또는 'composite process'의 인스턴스가 2개 이상인 Effect를 찾아낸다. 측정치는 0과 1 사이의 값이며 1에 가까울수록 유비쿼터스 서비스 온톨로지의 대체성은 높아진다.

$$\text{대체성} = \frac{\text{조건1을 만족하는 Effect의 개수}}{\text{'expr:SWRL-Condition'의 인스턴스 중 Effect의 개수}} \quad (2\text{-}1)$$

조건1 : $((AP_n \text{또는} CP_n), hasResult, E_i)$ 그리고 $((AP_n \text{또는} CP_n)$의 개수 $\geq 2)$

단, $((AP_n \text{또는} CP_n), hasResult, E_i)$ 구문은
온톨로지의 triple구조를 의미하며
어떤 u-서비스(AP_n 또는 CP_n)가
어떤 Effect(E_i)를 결과로 발생한다는
OWL-S 표현법이다.

단, AP_n 또는 CP_n : 어떤 atomic process 또는 composite process

• QSC3: 독립성(Independency)

유비쿼터스 서비스 온톨로지는 기기들의 기능(operations)들을 나열한 단순 목록이 아니라, 다양한 환경에서 다양한 기기에 의해 제공될 수 있는 단위로 추상화한 개념이어야 한다[3,4]. 다시 말해서 같은 서비스라 하더라도 다양한 실행 플랫폼이나 기기에 의해 제공 가

능한 구조로 설계한다면, 이미 다른 서비스를 실행하고 있거나 고장 등과 같은 기기 자체의 유효성 문제로 인한 서비스의 제공 실패율을 감소시킬 수 있을 것이다. u-서비스 기술 규격 항목 중에서 'Object' 항목에 해당 서비스를 실행할 수 있는 기기를 명시하도록 규정하고 있는데, 도메인 온톨로지에 기기에 대한 정보를 풍부하게 포함하고 u-서비스 기술 시 활용하여야 한다.

그림 5에 보인 것과 같이 단일 기기(*D-1* 또는 *D-2*)에 의해서만 실행될 수 있는 *AS-1*, *AS-2*, *AS-3* 서비스는 해당 기기가 비치되어 있지 않거나 고장 난 상태라면 제공이 아예 불가능한 반면에 *AS-4*나 *AS-5* 서비스는 어느 한 클래스의 기기들이 문제가 있다고 하더라도 다른 기기로 대체 실행할 수 있으므로 특정 플랫폼에 독립적이라 할 수 있다.

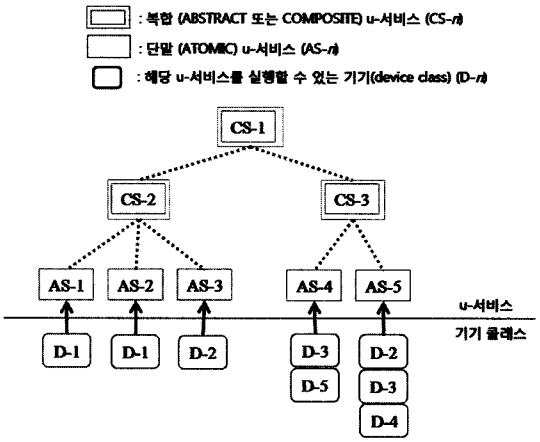


그림 5 유비쿼터스 서비스 온톨로지의 독립성

메트릭 (3)에서 'dynamic u-서비스'는 다음과 같이 정의할 수 있다. 그림 5의 예에서는 *AS-4*, *AS-5*, *CS-3*, *CS-1*이 'dynamic u-서비스'가 된다.

dynamic u-서비스

다중(2개 이상) 플랫폼 혹은 기기에 의해 실행 가능한 u-서비스.

아래의 식 (3-1)에서 조건 2는 u-서비스를 실행할 수 있는 기기 클래스가 존재해야만 서비스 제공이 가능하므로 전제 조건이 되고 기기 클래스가 2개 이상인 u-서비스들만을 찾아낸다. 역시 0~1의 값으로 산출되는데 높은 수치가 높은 독립성을 나타낸다.

$$\text{독립성} = \frac{\text{조건2를 만족하는 } AP_i \text{ 또는 } CP_i\text{의 개수}}{AP_i\text{의 개수} + CP_i\text{의 개수}} \quad (3-1)$$

조건2 : $((AP_i, \text{또는 } CP_i), hasPrecondition, Cls_n)$ 그리고
(Cls_n의 하위클래스의 개수 ≥ 2)

단, $((AP_i, \text{또는 } CP_i), hasPrecondition, Cls_n)$ 구문은
온톨로지의 triple 구조를 의미하며
어떤 u-서비스(AP_i, 또는 CP_i)가 실행되기 위해서는
어떤 기기 클래스(Cls_n)가 있어야 한다는
전제조건을 OWL-S 표현법이다.

단, AP_i, 또는 CP_i : 어떤 atomic process 또는 composite process

단, Cls_n : 클래스(classes)

• QSC4: 계층 구조(Layered Structure)

SOA의 서비스 모델링 기법은 서비스의 크기와 역할에 따라 3 단계로 분류하여 계층화한다[5]. 이를 참조로 유비쿼터스 서비스 온톨로지는 도출한 u-서비스들을 단순히 단일 레벨로 나열하지 않고 추상화 정도에 따라 계층적 구조로 설계한다. 3장의 u-서비스 기술 규격에서 언급하였듯이, u-서비스는 ABSTRACT, COMPOSITE, ATOMIC의 3가지 타입으로 분류하고 ABSTRACT과 ATOMIC은 단일 레벨로, COMPOSITE은 다중 레벨로 구성하여 3단계 이상의 계층을 이루게 된다. 이러한 구조는 Effect와 기능 및 특징에 따라 u-서비스들을 그룹화하는 과정에서 자연스럽게 형성될 수 있으며 유비쿼터스 서비스 온톨로지의 동적 기능을 가능케 하는 효율적인 기반이 된다.

이러한 면에서 이 '계층 구조' 속성이 '조합성' 속성과 유사하거나 부수적인 속성이라고 볼 수도 있다. 하지만 표 2의 3가지 주요 특성 중에서 '조합성'은 유비쿼터스 도메인의 특성을 반영하는데 초점을 맞추는 반면에 '계층 구조' 속성은 온톨로지의 계층 구조를 활용한다는 특성에 초점을 맞추고 있어 구별된다.

표 3의 메트릭 (4)를 사용하여 정량화하고, 모든 조합의 평균 계층 수로 판단할 수 있다. 즉, 계층을 나누는 역할을 하는 복합 u-서비스의 계층 수의 평균치를 계산한다. 예를 들어, 그림 4(a)와 (b)가 각각 독립된 온톨로지라고 가정할 경우, (a)의 계층 구조는 1/1=1이고, (b)는 1+1+2/3=1.33이다.

다음의 식에 보인 것과 같이 유비쿼터스 서비스 온톨로지에서의 복합 u-서비스인 ABSTRACT과 COMPOSITE u-서비스는 'composite process'의 인스턴스 개수에 해당한다. 조합의 계층 높이는(하위 조합의 최대 계층 높이+1)로 계산하고 하위 조합이 없을 경우에는 당연히 1이다.

$$\text{계층구조} = \frac{\sum CP_i\text{의 최대 계층 개수}}{\text{'process:CompositeProcess'의 인스턴스 개수}} \quad (4-1)$$

단, CP_i의 최대 계층 개수는
$\sum \max(CP_i\text{의 하위조합의 최대계층개수}) + 1$

이 메트릭은 1 이상의 값이 계산된다. 단일 계층으로 이루어진 경우에 1 값을 갖게 되고 수치가 높을수록 많

은 계층을 가짐을 의미한다. 많은 계층이 반드시 효율적인 구조는 아니지만, 이 '계층 구조' 속성은 계층의 단일/다중 여부만 고려하기 때문에 위의 식에서도 최대 계층 높이를 계산하였다. 계층의 효율성은 '느슨한 결합' 속성으로 판단할 수 있으며 4.4.2절의 유연성 품질 속성에서 설명한다.

• QSC5: 재사용성(Reusability)

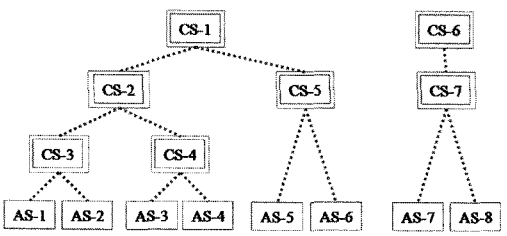
여기에서의 '재사용성'은 유비쿼터스 서비스 온톨로지의 설계 시에 이미 정의되어 있는 u-서비스를 활용함을 의미한다. u-서비스의 특성에 따라 가능한 한 다른 u-서비스들과 많은 관계로 연결될수록 즉, 다양한 조합을 형성할수록 u-서비스로 추상화하는 의미도 있을 것이며 서비스 대체와 같은 동적 기능도 효과적으로 지원할 수 있을 것이다. 따라서 단지 많은 u-서비스를 정의하거나 u-서비스의 단순 분류를 지양한다. 예를 들어, 그림 6(a)의 경우 유비쿼터스 서비스 온톨로지의 구조를 형성하면서 15개의 u-서비스들이 크게 두 분류로 배타적으로 나뉘어 있고 단일 관계(점선)만을 가지고 있다. 반면에 (b)는 *CS-2* 그룹이 *CS-6* 복합 서비스와 연관됨을 파악하여 재사용함으로써 *CS-2* 그룹에 속한 서비스들이 다양한 목적으로 활용되도록 할 수 있다. 이와 같이, u-서비스의 개수만 많이 늘리지 않고 u-서비스를 최대한 추상화하여 도출하고 특성을 파악하여 구조화할 필요가 있다.

이 속성은 메트릭 (5)를 통해 측정할 수 있고, u-서비스의 전체 개수에 대해 조합 형성 시 중복 사용된 u-서비스의 개수의 비율로 계산한다. OWL-S에서는 복합 프로세스를 생성할 때 *'Perform'* 제어 구조를 통해 하위 서비스를 표현하고, 이미 정의된 서비스를 참조할 경우에 *'rdf:resource'*를 사용한다.

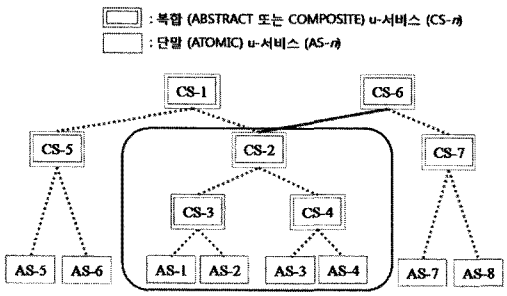
재사용성= (5-1)

'process:Perform'의 인스턴스 중, 'rdf:resource'를 사용하는 인스턴스의 개수 / 'process:AtomicProcess'의 인스턴스 개수+ 'process:CompositeProcess'의 인스턴스 개수

이 메트릭을 통해 그림 6(a)의 예는 0/15=0의 재사용성을, (b)는 7/15=0.47의 재사용성을 가짐을 알 수 있다. 0 이상의 측정치가 산출되며 높을수록 재사용성이 높음을 뜻한다.



(a) 재사용성이 없는 구조



(b) 재사용성이 있는 구조

그림 6 유비쿼터스 서비스 온톨로지의 재사용성

4.4.2 QC2: 유연성(Flexibility)

u-서비스들간의 관계가 종속적이지 않게 설계되었는지에 대한 특성으로 다음의 3가지 세부 속성으로 나눌 수 있다.

• QSC6: 느슨한 결합(Loose Coupling)

여기에서 '결합'은 복합 u-서비스를 생성할 때의 조합 관계 즉, 상위 레벨과 하위 레벨의 관계를 의미한다. 복합 서비스가 단일 하위 서비스로 이루어질 경우에는 두 서비스 간의 관계를 긴밀(tight)하다고 볼 수 있으며 이는 불필요하고 무리한 계층화의 결과이다. 예를 들어, 위의 그림 6(a)에서 *CS-6* 서비스는 CS-7 서비스 하나만으로 구성된 복합 서비스로 두 서비스는 긴밀하게 연결되어 있다.

표 3의 메트릭 (6)을 통해 측정 가능하며, 복합 u-서비스 중에서 조합을 구성하는 하위 u-서비스가 2개 이상인 것들의 비율로 산출된다. 조합을 이진 형식(binary)으로 표현하는 OWL-S 구문에서는 단일 하위 서비스로 이루어진 복합 u-서비스의 경우에 *'nil'*을 포함하므로 다음과 같이 계산할 수 있다. 즉, 그림 6(a)는 6/7=0.86이다. 높은 측정치가 느슨한 결합으로 구성되었음을 뜻한다(0~1).

느슨한 결합= (6-1)

'nil'을 갖지않는 CP_i의 개수 / 'process:CompositeProcess'의 인스턴스 개수

• QSC7: 자율성(Autonomy)

어떤 u-서비스를 실행하기 위해 반드시 선행되어야 하는 서비스가 존재한다면 해당 u-서비스의 실행 자율성이 없다고 할 수 있다. 예를 들어, 본 연구에서 구축한 u-서비스 온톨로지에서 *'HeaterUP'* u-서비스는 *'HeaterON'* u-서비스가 선행되어야 실행 가능하다고 가정하였으므로, 이럴 경우에 *'HeaterUP'* u-서비스는 자율성이 낮다고 볼 수 있다. 따라서 선행 서비스가 반드시 필요한 경우도 있을 수 있으나, u-서비스의 도출 및 구

조 설계 단계에서부터 각 u-서비스가 선행 서비스가 없도록 하는 것이 좋다.

메트릭 (7)에서처럼 모든 u-서비스 중에서 선행 서비스가 없는 u-서비스의 비율을 측정하며, u-서비스의 실행 전제 조건을 명시하는 *'hasPrecondition'* 구문을 통해 알 수 있다. 높은 수치가 높은 자율성을 의미하게 된다(0~1).

$$\text{자율성} = \frac{\text{'process:hasPrecondition'에 다른 u-서비스가 명시되지 않은 u-서비스의 개수}}{\text{'process:AtomicProcess'의 인스턴스 개수} + \text{'process:CompositeProcess'의 인스턴스 개수}} \quad (7-1)$$

• QSC8: 다형성(Polymorphism)

이 속성은 객체 지향 패러다임에서 차용한 개념으로서, 유비쿼터스 서비스 온톨로지에서의 다형성을 '복합 u-서비스(ABSTRACT 또는 COMPOSITE)는 상황과 조건에 따라 하위의 u-서비스들 중에서 최적의 서비스로 선택되어 실행될 수 있음'으로 재정의하였다. 이 개념은 유비쿼터스 환경에서 사용자가 원하는 단말 서비스를 정확하게 지정하지 않고 모호하고 추상적으로 목적(goal)을 기술하더라도 유비쿼터스 서비스 온톨로지의 구조를 통해 서비스 제공 시에 가장 적합한 서비스를 선택하고 실행할 수 있음을 포함한다. 예를 들어, 그림 3(b)의 경우 사용자가 '온도 조절'을 원하더라도 어떤 조건 하에서는 '난방 장치 가동' 서비스가, 다른 조건 하에서는 '창문 열기' 서비스가 제공될 수도 있다. 위의 그림 6(b)에서 복합 u-서비스 *CS-1*은 실행 시의 주변 상황이나 사용자의 선호도 등에 따라 최종적으로 *AS-1*으로 실현될 수도 있고 *AS-3* 또는 *AS-6*으로 실현될 수도 있을 것이다.

다형성을 정량화하기 위한 표 3의 메트릭 (8)에서 '다형 u-서비스'는 다음과 같이 정의한다.

다형 u-서비스(polymorphic u-Service)

각 복합 u-서비스의 다형 u-서비스는 실행 시의 상황에 따라 바인딩될 수 있는 하위 레벨의 u-서비스를 뜻한다. 즉, 해당 u-서비스의 조합을 형성하는 모든 하위 u-서비스들이다.

이 정의에 따라, 그림 6(b)에서 *CS-4*의 다형 u-서비스는 2개(*AS-3, AS-4*), *CS-1*은 10개가 된다. 이 경우의 다형성 수치는 (10+6+2+2+2+10+2)/7=4.86이다. 아래의 식에서처럼 각 복합 u-서비스를 조합하는 하위 u-서비스들을 재귀적으로 모두 계산할 수 있다. 이 속성은 0 이상의 수치로 나타나며 높을수록 다형성이 높음을 나타낸다.

$$\text{다형성} = \frac{\sum_{\text{각 } CP_i\text{의 'process:composedOf'로 조합된 모든 sub-u-서비스의 개수}}}{\text{'process:CompositeProcess'의 인스턴스 개수}} \quad (8-1)$$

4.4.3 QC3: 기능성(Functionality)

유비쿼터스 서비스 온톨로지 자체의 크기와 표현된 정보의 양에 초점을 맞추는 속성으로서 다음 4가지 세부 속성과 해당 메트릭을 통해 측정할 수 있다.

• QSC9: 지원범위(Coverage)

유비쿼터스 서비스 온톨로지 구축은 유비쿼터스 환경의 모든 상황을 예측할 수 없으므로 몇 개의 대표 상황들을 기초로 시작된다. 따라서 최소한 대표 상황에 등장하는 기능들은 모두 u-서비스로 도출되어 완전하게 지원할 수 있어야 한다.

이 속성은 메트릭 (9)를 사용하여 측정한다. 명시적인 작은 단위의 기능뿐만 아니라 추상적으로 기술되어 있는 큰 단위의 기능들이 도출한 u-서비스들을 통해 지원 가능한지 수치화한다. 예를 들어, 다음과 같은 상황을 가정할 수 있다.

- 상황 1: 저녁 10시가 넘어 사용자 A씨가 침대에 눕자 커튼이 닫히고, 형광등이 꺼진다. 대신 협탁 스탠드가 켜지고 조용한 음악이 나온다.
- 상황 2: 사용자 B씨가 수면을 취할 동안, 온도는 24°C로 유지하고 소음에 방해받지 않도록 한다.

유비쿼터스 서비스 온톨로지를 사용하여 위의 상황 1에서의 '커튼이 닫힌다'라는 기능은 *'CurtainClose'* ATOMIC u-서비스로, 상황 2의 '온도 유지' 기능은 *'manageTemperature'* ABSTRACT u-서비스로 지원할 수 있다.

0~1 사이의 값이며 1에 가까울수록 지원력이 높다고 볼 수 있다.

• QSC10: 상황 정보 풍부성(Context Richness)

상황인식 및 추론을 통한 동적 서비스 제공을 위해서는 상황정보에 대한 효과적인 모델링이 필요하다. 이러한 정보를 체계적으로 표현하기 위해 유비쿼터스 서비스 온톨로지에 상황 온톨로지(context ontology)가 포함되고, 도메인 내의 위치, 시간, 사람에 대한 정보를 비롯한 상황 정보에 대한 구조, 그들의 특성과 관계를 표현하여야 한다. 그 중에서도 가장 중요한 것은 기기에 대한 정보로서, 유비쿼터스 환경에 배치될 수 있는 기기를 최대한 풍부하게 정의 및 분류하고 그들의 특징(현재 상태, 에너지 효율, 위치 등)을 표현하여야 u-서비스의 속성 기술 시에 활용할 수 있다.

따라서 상황 온톨로지의 크기를 알 수 있는 개념의 양(concept quantity)을 측정함으로써 상황정보의 풍부성을 평가한다. 전체 클래스의 개수를 계산하는 메트릭 (10)은 OWL에서 모든 클래스의 최상위 클래스인 'owl:Thing'의 하위 클래스이므로 다음과 같이 재정의할 수 있다.

$$\text{상황정보풍부성} = \text{'owl:Thing'의 모든 하위클래스(Classes)의 개수} \quad (10-1)$$

측정치는 0 이상의 값으로서, 높은 수치일수록 상황 정보가 풍부하게 표현되었다고 해석할 수 있다.

• QSC11: u-서비스 풍부성(u-Service Richness)

유비쿼터스 서비스 온톨로지를 다양한 유비쿼터스 상황에 적용할 수 있으려면 최대한 많은 u-서비스를 풍부하게 포함하고 있어야 할 것이다.

메트릭 (11)의 전체 u-서비스의 개수는 OWL-S에서의 모든 'atomic'과 'composite process'의 인스턴스 개수로 알 수 있다. 0 이상의 수치로, 높은 값일수록 풍부한 u-서비스를 포함하고 있음을 나타낸다.

u-서비스 풍부성= (11-1)
'process:AtomicProcess'의 인스턴스 개수+
'process:CompositeProcess'의 인스턴스 개수

• QSC12: Effect 풍부성(Effect Richness)

유비쿼터스 서비스 온톨로지의 구조는 기능적 또는 환경적 Effect를 기본으로 이루어져 있다. 따라서 유비쿼터스 환경에서 발생할 수 있는 Effect를 최대한 모두 포함하여야 한다.

메트릭 (12)에 보인 것과 같이 u-서비스 기술 시에 사용된 모든 Effect의 개수를 산출한다. Effect는 OWL-S에서 SWRL 규칙의 이름으로 표현하므로 다음의 식으로도 나타낼 수 있다.

Effect 풍부성= (12-1)
'expr:SWRL-Condition'의 인스턴스 중 Effect의 개수

0 이상의 값으로, 높은 수치일수록 Effect가 풍부하게 도출되었음을 의미한다.

4.5 품질 속성 측정

4.3절에서 정의한 품질 속성은 각 품질 속성을 구성하는 세부 품질 속성들로부터 산출될 수 있다. 그림 7과 그림 8은 품질 속성의 세부 속성들 간에 영향을 미치는 관계를 보이고 있는데, 높은 관계성을 실선으로, 약한 관계성을 점선으로 나타내었다. 그림 7에 보였듯이 조합성을 통한 의미 있는 복합 서비스들의 생성이 전제되어야 다른 4개 세부 속성들도 보장할 수 있으므로 조합성이 역동성을 위한 가장 중요한 세부 속성이라고 할 수 있다. 또한 그림 8과 같이 다형성은 다른 2개의 세부 속성들에 영향을 미칠 뿐만 아니라 유연성을 향상시키는 가장 중요한 속성이다.

이 관계도에서 나타나는 세부 속성들의 중요도에 따라 가중치를 부여하였고, 표 4와 같이 품질 속성을 정량적으로 계산할 수 있는 식을 도출하였다. 기능성의 4가지 세부 속성들은 각 대상의 양을 측정하는 개별적인 속성들이므로 동일한 가중치를 할당한다. 표 4의 식에 의해 산출된 수치를 비교하면 평가 대상 온톨로지의 강점과 단점뿐만 아니라 개선 방향도 각 속성 별로 확인할 수 있다.

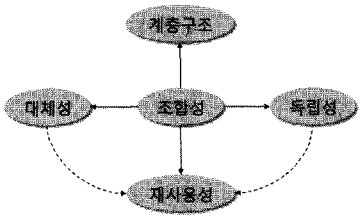


그림 7 역동성의 세부 속성들 간의 관계도

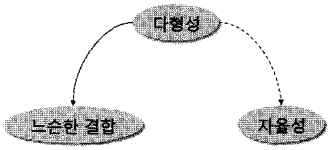


그림 8 유연성의 세부 속성들 간의 관계도

표 4 품질 속성 정량화

품질 속성		정량화 식
QC1:	역동성	(0.3 × 조합성) + (0.2 × 대체성) + (0.2 × 독립성) + (0.15 × 계층 구조) + (0.15 × 재사용성)
QC2:	유연성	(0.3 × 느슨한 결합) + (0.3 × 자율성) + (0.4 × 다형성)
QC3:	기능성	(0.25 × 지원범위) + (0.25 × 상황정보 풍부성) + (0.25 × u-서비스 풍부성) + (0.25 × Effect 풍부성)

5. 적용 및 평가

위에서 정립한 품질 평가 모델이 유비쿼터스 서비스 온톨로지에 실제로 적용 가능한지, 전반적인 평가가 가능하지를 검증하기 위해, 실제 서비스 온톨로지에 적용해보았다. 선정한 실험 대상은 대표적인 OWL-S 예제인 'Congo.com' 서비스[20]와 'Bravo Air' 서비스[21]에 대한 서비스 온톨로지와 이전 연구에서 구축한 u-서비스 온톨로지의 두 가지 버전이다. 표 5는 실제로 품질 모델을 적용하여 측정하기 전에 실험 대상을 비교 분석한 내용이다. Bravo Air와 Congo.com 서비스 온톨로지는 비즈니스 서비스를 표현 대상으로 했기 때문에 품질 메트릭에 사용되는 항목들 - 표 5의 4, 6, 7, 8번째 행 - 자체가 아예 없어서 측정이 불가능한 속성이 많다. 뿐만 아니라, u-서비스 온톨로지와 비교하여 서비스 개수를 비롯한 온톨로지 크기가 현저하게 작아서 평가 결과를 비교하는 것이 무의미하다고 판단하여 실험 대상에서 제외하였다.

따라서 본 연구에서 정립한 품질 평가 모델을 적용에

표 5 실험 후보 서비스 온톨로지 비교 및 실험 적합 여부

비교 항목 \ 대상	Bravo Air	Congo.com	u-서비스 온톨로지 v.1.0	u-서비스 온톨로지 v.3.0
설명	온라인 항공 예약 서비스	온라인 서적 구매 서비스	u-서비스 온톨로지의 초기 버전.	가장 최근 버전으로서, 대표 상황 확충으로 u-서비스를 비롯한 서비스/도메인 정보가 추가됨.
도메인	비즈니스 (웹 서비스)	비즈니스 (웹 서비스)	유비쿼터스	유비쿼터스
대표 상황 개수	없음	없음	12	52
서비스 개수	7	14	125	264
Effect 개념	없음	없음	91개 Effect	217개 Effect
기기 개념	없음	없음	61개 기기 클래스	94개 기기 클래스
기술 규격	없음 (웹 서비스)	없음 (웹 서비스)	u-서비스 기술 규격	u-서비스 기술 규격
OWL-S 코드	790 라인	2,052 라인	20,304 라인	33,177 라인
실험 적합 여부	X	X	O	O

적합하다고 판단한 두 가지 버전의 u-서비스 온톨로지에 적용해보았다. u-서비스 온톨로지는 본 연구의 일환으로 u-홈과 공공 안전 도메인을 대상으로 2007년도부터 개발 및 업데이트되고 있으며, 온톨로지 편집 도구인 Protégé[22]를 사용하여 구축하였다. 버전 1.0은 본 연구의 초기에 u-서비스 온톨로지의 구축 방법론 및 기술 규격 등을 정립하면서 구축한 온톨로지로서, 제안한 설계 원리를 충실히 반영하는 것을 최우선 목표로 설계하였다. 반면에 버전 3.0은 많은 대표 상황 지원과 양적인 확장을 위해 버전 1.0에 되도록 많은 u-서비스와 클래스 등을 추가하였다.

표 6에 실험 대상인 2개의 u-서비스 온톨로지에 메트릭을 적용하여 계산한 실제 결과값을 보인다.

- **QSC1: 조합성** 3.0 버전이 복합 u-서비스의 개수도 늘었지만 대부분이 비절차적 조합으로 형성되었음을 나타내고 있다.
- **QSC2: 대체성** 3.0 버전에서 Effect의 개수는 2배 이상이 증가하였지만 대부분이 'dynamic Effect'가 아니기 때문에 측정치가 더 떨어졌다. u-서비스가 추가되면서 Effect가 거의 일대 일로 비례하여 증가하였음을 나타내는데, 이는 대표 상황에 기반한 u-서비스 도출의 부작용 때문으로 분석된다. 즉, 대표 상황에 등장하는 기능에 의존하여 u-서비스를 도출하게 됨으로써 추상화하거나 확장하지 못하는 한계로 인한 문제점이라고 할 수 있겠다. 예를 들어, '수상한 침입자의 도주를 CCTV로 추적'하는 상황으로부터 'CCTV ZoomIn'을 비롯한 관련 u-서비스가 추가되면서 'E-ZoomIN'과 같은 Effect도 일대 일로 계속 추가만 되었다는 의미이다. 따라서 평가 결과, u-서비스 온톨로지 3.0은 u-서비스를 더 풍부하게 도출하거나 Effect의 단위(granularity)를 조정 또는 다양화하는 등의 구조 개선이 필요하다.
- **QSC3: 독립성** 3.0 버전이 더 낮은 독립성을 보이고 있으며 이 결과는 추가된 u-서비스들이 대부분 실행 플랫폼과 일대 일로 연관됨을 뜻한다. 역시 대표 상황에 등장하는 기기의 기능을 u-서비스로 추상화하지 못하고 그대로 목록화만 했음을 알 수 있다. 위에서 언급한 CCTV 관련 u-서비스가 하나의 예일 것이다. 도메인 온톨로지의 기기 클래스를 다양화하거나 유사한 u-서비스들을 하나의 u-서비스로 통합할 필요가 있겠다.
- **QSC4: 계층 구조** 3.0 버전의 온톨로지의 평균 계층의 높이가 더 낮다. 1.0 버전에 비해 많은 조합을 이루고는 있지만 2개 이상의 다중 계층을 이루는 복합 u-서비스의 비율이 낮음을 알 수 있다.
- **QSC5: 재사용성** 3.0 버전이 조금 더 높은 수치를 보이고는 있지만 여전히 u-서비스들이 단순 분류되었음을 나타낸다. 즉, 기존의 u-서비스들이나 이미 추가된 u-서비스들간의 연관성을 고려하지 못해 새로운 u-서비스를 생성하여 추가만 하였음을 알 수 있다.

표 6 품질 모델 적용 결과 - 세부 속성

품질 세부 속성 \ 대상	버전 1.0	버전 3.0
QSC1: 조합성	41/45=0.91	96/99=0.97
QSC2: 대체성	21/91=0.23	24/217=0.11
QSC3: 독립성	64/125=0.51	106/264=0.4
QSC4: 계층 구조	79/45=1.76	163/99=1.65
QSC5: 재사용성	22/125=0.18	62/264=0.23
QSC6: 느슨한 결합	45/45=1	96/99=0.97
QSC7: 자율성	29/125=0.23	159/264=0.6
QSC8: 다형성	315/45=7	741/99=7.48
QSC9: 지원범위	51/51=1	177/177=1
QSC10: 상황정보 풍부성	69	115
QSC11: u-서비스 풍부성	125	264
QSC12: Effect 풍부성	91	217

- **QSC6: 느슨한 결합** 이 속성의 측정치를 통해 3.0 버전의 온톨로지가 몇 개의 불필요한 계층(조합)을 포함하고 있음을 알 수 있다.
- **QSC7: 자율성** 3.0 버전의 추가된 u-서비스들의 대부분이 필수 선행 서비스가 없도록 설계되었음을 의미한다.
- **QSC8: 다형성** 3.0 버전이 더 높은 다형성을 보이고 있는데 이는 u-서비스를 추가하면서 그룹화 및 계층화를 효과적으로 시행한 결과라고 할 수 있겠다. 따라서 3.0 버전의 u-서비스들이 평균적으로 다형 u-서비스를 많이 갖게 되면서 추상적인 목적 기술(goal description)을 지원할 수 있을 뿐만 아니라, 상황에 따라 다양한 u-서비스들로 바인딩될 수 있는 가능성이 높아질 것이다.
- **QSC9:** 지원범위 두 버전 모두 대표 상황에 입각해 u-서비스를 도출하였음을 알 수 있다.
- **QSC10: QSC11: QSC12: 상황정보 풍부성 & u-서비스 풍부성 & Effect 풍부성** u-서비스 온톨로지가 업데이트되면서 선언적 클래스와 u-서비스, Effect가 확연하게 증가하였다.

위의 결과를 3가지 품질 속성 별로 그림 9, 그림 10, 그림 11에 보인다.

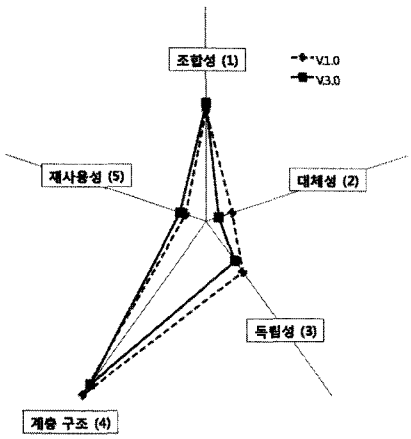


그림 9 역동성 비교(v.1.0 > v.3.0)

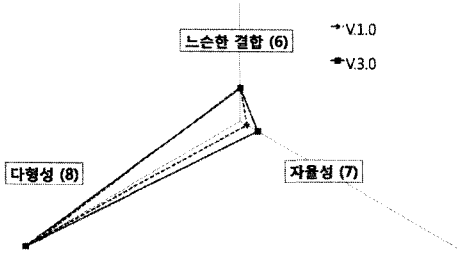


그림 10 유연성 비교(v.1.0 < v.3.0)

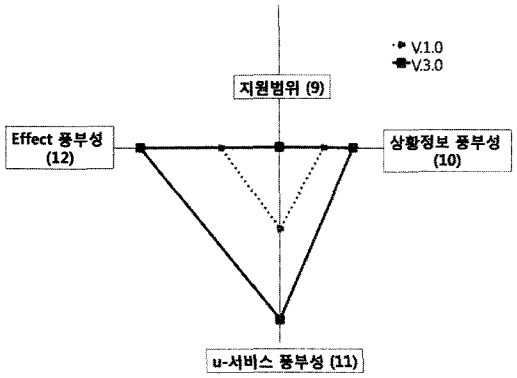


그림 11 기능성 비교(v.1.0 < v.3.0)

표 7 품질 모델 적용 결과 - 품질 속성

품질 속성 \ 대상		버전 1.0	버전 3.0
QC1:	역동성	0.712	0.675
QC2:	유연성	3.169	3.463
QC3:	기능성	71.5	149.25

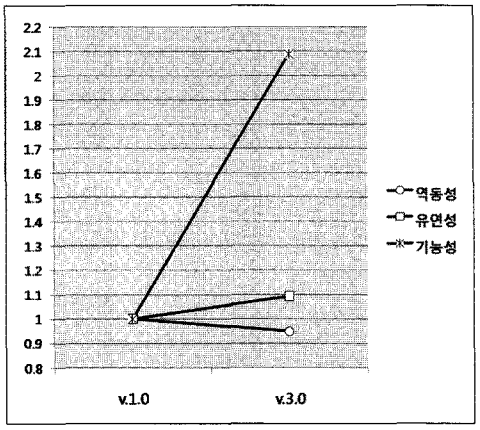


그림 12 품질 속성 비교

또한, 두 버전의 u-서비스 온톨로지에 대해 다음 표 7과 같이 각 품질 속성 별로 평가할 수 있다. 그림 12는 위의 결과치를 버전 1.0을 기준으로 정규화한 그래프로서, 각 속성 별로 두 버전 간의 품질을 비교할 수 있다.

- **QSC1: 역동성** 3.0 버전이 더 낮아졌다. 이는 5가지 세부 속성 중에서 3가지에 대해 3.0 버전의 수치가 낮기 때문이다. 이러한 결과는 새로 추가된 u-서비스들이 특정 실행 플랫폼과 그 기능에 종속되어 있어 추상적인 개념으로서의 u-서비스로서의 동적 특성을 나타내지 못하고 있음을 보여주고 있는 것이다. (1.0 버전 > 3.0 버전)
- **QSC2: 유연성** 3.0 버전에서 유연성이 향상되었음을

알 수 있다. 3가지 세부 속성에 대한 측정치를 통해 3.0 버전이 좀 더 나은 유연성을 갖는다고 할 수 있다. 온톨로지의 계층 구조를 활용하여 u-서비스가 다양한 추상화 레벨로 사용될 수 있고 개별 u-서비스들의 실행 상의 의존성이 낮아 유연하게 실행될 것이다(1.0 버전 < 3.0 버전).

- **QSC3: 기능성** 4가지 세부 속성 결과를 바탕으로 1.0 버전에 비해 3.0 버전의 기능성이 월등히 높다고 판단할 수 있다. 상황정보나 u-서비스 도출 의 양적 면에서는 3.0 버전이 향상되었으며 더 풍부한 정보를 포함하고 있다고 평가할 수 있다(1.0 버전 < 3.0 버전).

종합해보면 3.0 버전의 온톨로지가 기능성 면에서는 뚜렷한 향상을 보이고 있고, 유연성 면에서도 개선되었음을 알 수 있다. 이를 통해 3.0 버전이 양적으로 많은 내용을 포함하고 있고 개별 u-서비스들의 효과적인 표현이나 설계를 통해 더 많은 유비쿼터스 환경에서의 상황을 지원할 수 있을 것으로 평가할 수 있다. 하지만 낮아진 역동성 측정치가 나타내듯이 동적으로 변화하는 상황에 대처하여 서비스를 선택하거나 대체하기에는 역부족일 것이다. 너무 초기 대표 상황에 편중하였고 포괄적인 상황을 고려하지 못함으로 인해 기능과 기기에 종속적인 경향이 있고 u-서비스로의 추상화와 관계 파악에 소홀하였기 때문이다. 결과적으로 3.0 버전의 온톨로지를 사용할 경우 비교적 단순한 상황은 충분히 지원 가능하겠지만 동적으로 변화하는 상황에서의 u-서비스들간 혹은 기기간의 동적 대체에 있어서는 실패할 가능성이 크다고 할 수 있다. 이를 개선하기 위해 u-서비스 도출 과정에서 기능 레벨을 벗어난 추상화를 염두에 두어야 하고 u-서비스 자체뿐만 아니라 Effect와 도메인 정보의 다양화가 필요할 것이다.

6. 결 론

유비쿼터스 서비스 온톨로지는 유비쿼터스 컴퓨팅 실현에 핵심적인 동적 서비스 발견 및 합성의 필수적인 지식 기반으로서, 도메인 정보뿐만 아니라 u-서비스에 대한 정보를 효과적으로 표현하고 있어야 한다. 이에 본 논문에서는 설계 단계에서 유비쿼터스 서비스 온톨로지의 효율성을 평가하기 위한 품질 평가 모델을 제안하였다. 품질 모델은 u-서비스 온톨로지의 특성을 반영하고 있으며 실행 시에 특정 u-서비스가 성공적으로 실행되었는지와 같은 성능 측정이 아니라 u-서비스를 도출하고 구조를 설계하는 과정에서의 설계 원칙 및 평가 지표를 제시하고 있다. 유비쿼터스 서비스 온톨로지가 목표로 해야 하는 3가지 품질 속성을 선정하고 12가지 세부 속성과 정량화를 위한 메트릭을 정의하여 유비쿼터스 서비스 온톨로지가 얼마나 효과적으로 설계되었는지 평가할 수 있을 것이다. 향후에는 품질 모델의 타당성과 완전성(completeness)의 검증을 통한 모델 개선과 이를 바탕으로 유비쿼터스 서비스 온톨로지의 설계 과정에서 실시간으로 평가 결과뿐만 아니라 수정 방향을 제공하는 자동 평가 시스템의 개발도 고려할 것이다.

참 고 문 헌

[1] D. Chakraborty, F. Perich, A. Joshi, T. W. Finin, and Y.Yesha, "A Reactive Service Composition Architecture for Pervasive Computing Environments," *Proc. of the 7th Personal Wireless Communications Conference*, pp.53-62, Oct. 2002.

[2] D. Chakraborty and A. Joshi, "Dynamic Service Composition: State-of-the-Art and Research Directions," *Technical Report TR-CS-01-19*, University of Maryland, Dec. 2001.

[3] M. Lee, S. Park and J. Lee, "Ontology-based Service Layering for Facilitating Alternative Service Discovery," *Proc. of the 2th International Conference on Ubiquitous Information Management and Communication*, pp.482-487, Jan. 2008.

[4] M. Lee, J. Lee, S. Park and W. Cho, "Ontology-based Service Description and Overloading Method for Ubiquitous Computing," *Journal of Korea Information Processing Society*, vol.15-B, no.5, pp.465-476, Oct. 2008. (in Korean)

[5] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, 2005.

[6] A. Gomez-Perez, "Some Ideas and Examples to Evaluate Ontologies," *Proc. Of the 11th Conference on Artificial Intelligence for Applications*, pp.299-305, Feb. 1995.

[7] H. Ning and D. Shihan, "Structure-Based Ontology Evaluation," *Proc. Of IEEE International Conference on e-Business Engineering (ICEBE)*, pp.132-137, Oct. 2006.

[8] "ISO/IEC 9126-1 Software Engineering-Product Quality-part 1: Quality Model," *ISO/IEC Technical Report*, 2001.

[9] "ISO/IEC 9126-2 Software Engineering-Product quality-part 2: External Metrics," *ISO/IEC Technical Report*, 2003.

[10] "ISO/IEC 9126-3 Software Engineering-Product quality-part 3: Internal Metrics," *ISO/IEC Technical Report*, 2003.

[11] "ISO/IEC 9126-4 Software Engineering-Product quality-part 4: Quality in Use Metrics," *ISO/IEC Technical Report*, 2004.

[12] J. Bansiya and C.G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment," *IEEE Transactions on Software Engineering*, vol.28, no.1, pp.4-17, Jan. 2002.

[13] B. Shim, S. Choue, S. Kim and S. Park, "A Design Quality Model for Service-Oriented Archi-

tecture," *Proc. of the 15th Asia-Pacific Software Engineering Conference*, pp.403-410, Dec. 2008.
[14] S. Oh, S. Kim and S. Rhew, "UCQM: A Quality Model for Practical Evaluation of Ubiquitous Computing Systems," *Journal of KIISE: Software and Applications*, vol.34, no.4, pp.342-358, Apr. 2007. (in Korean)
[15] S. Lee and D. Shin, "Web Service QoS in Multi-Domain," *Proc. of the 10th International Conference on Advanced Communication Technology*, pp.1759-1762, Feb. 2008.
[16] J. Jang, D. Shin and K. Lee, "Fast Selection of Composite Web Services Based on Workflow Partition," *Journal of KIISE: Software and Applications*, vol.34, no.5, pp.431-446, May. 2007. (in Korean)
[20] S. Tartir, I. B. Arpinar and A.P. Sheth, "Ontological Evaluation and Validation," *In Theory and Applications of Ontology (TAO)*, vol.2, Springer-Berlin, 2008.
[21] D. L. McGuinness and F. van Hamelen, "OWL Web Ontology Language Overview," *W3C Member Submission*, 2004.
[22] D. Martin, et al., "OWL-S: Semantic Markup for Web Services," *W3C Member Submission*, 2004.
[23] CongoService.owl, http://www.daml.org/services/owl-s/1.1/examples.html
[24] BravoAirService.owl, http://www.daml.org/services/owl-s/11/examples.html
[25] Protégé, http://protege.stanford.edu/

이 미 연
2003년 이화여자대학교 컴퓨터학과 학사 2005년 이화여자대학교 컴퓨터학과 석사 2007년~현재 이화여자대학교 컴퓨터정보통신공학과 박사과정. 관심분야는 유비쿼터스 컴퓨팅, 온톨로지, SOA, 인공지능

박 승 수
1974년 서울대학교 수학과 학사. 1976년 한국과학기술원 전산학 석사. 1988년 미국 Texas대학 전산학 박사. 1988년~1991년 미국 Kansas대학 컴퓨터학과 조교수. 1991년~현재 이화여자대학교 컴퓨터공학과 교수. 관심분야는 인공지능, 시맨틱웹, 온톨로지

이 정 원
1993년 이화여자대학교 전자계산학과 학사. 1995년 이화여자대학교 전자계산학과 석사. 1995년~1997년 LG종합기술원 주임연구원. 2003년 이화여자대학교 컴퓨터학과 박사. 2003년~2006년 이화여자대학교 컴퓨터학과 BK교수, 전임강사(대우). 2006년~현재 아주대학교 정보통신대학 전자공학부 조교수. 관심분야는 SOA, 유비쿼터스 컴퓨팅, 임베디드 소프트웨어