논문 14-39C-08-02

The Journal of Korea Information and Communications Society '14-08 Vol.39C No.08
http://dx.doi.org/10.7840/kics.2014.39C.8.626

입출력 파라미터 특성을 이용한 REST 기반의 Open API 온톨로지 모델링 및 자동 매쉬업 방법

정 완•, 김 화 성°

REST-Based Open API Ontology Modeling and Automatic Mash-Up Method Using In/Output Properties

Jung wan•, Kim Hwa Sung°

요 약

기존의 매쉬업 서비스는 개발자 위주로 제작되어 모든 사용자의 취향과 목적에 맞춰 제공할 수 없기 때문에 사용자가 자동으로 매쉬업 서비스를 제작할 수 있는 방법에 대한 연구가 진행되었다. 자동 매쉬업 서비스를 제공하기 위해서는 매쉬업이 가능한 요소를 선별하는 방법이 핵심 연구이며 선행 연구에서는 REST 기반 Open API의 입출력 파라미터 이름으로 유사도를 비교하는 방법을 사용하였다. 하지만 유사도 비교만 이용하여 매쉬업 가능성을 판별할 경우에는 입출력 파라미터의 특성을 고려하지 못하기 때문에 의도하지 않은 결과의 출력이나 매쉬업이 불가능한 경우가 존재하였다. 본 논문에서는 잘못된 매쉬업의 결과를 줄일 수 있는 방법으로 입출력 파라미터의 특성을 고려하는 방안에 대해 제시하고 이를 선행 연구에서 제안한 온톨로지에 적용하여 확장하였다. 또한 확장된 온톨로지를 기반으로 매쉬업이 가능한 요소를 판별하는 알고리즘을 제안하고 자동 매쉬업 서비스의 구현 결과를 보였다.

Key Words : Automatic mash-up, REST-based Open API, Ontology Modeling, Open API parameter

ABSTRACT

Existing mash-up services could not be offered in accordance with the purposes and preferences of all users because they are created by the service developers. Therefore some precedent studies, which enable for individual users to create their own mash-up services automatically, have been conducted. In order to create automatic mash-up services, it is important to find elements to distinguish the possibility of mash-up. The precedent studies determine the possibility of mash-up through comparison of the similarity between input/output parameter names in the REST-based Open API. Only using the similarity to distinguish the possibility of mash-up, however, some unintended mash-up results can be occurred because the property of input/output parameters are not considered. In this paper, we propose the method considering the properties of input/output parameters to decrease the unintended mash-up results and extend ontology proposed in precedent studies by applying this property. And we propose the algorithm to distinguish the possibility of mash-up using the expanded ontology and describe the result of automatic mash-up services.

※ 본 연구는 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구 사업(2011-0025226) 지원을 받아 수행된 것 입니다.
• First Author : KwangWoon University Electronics and Communications Engineering, jung_wan@kw.ac.kr, 학생회원
° Corresponding Author : KwangWoon University Electronics and Communications Engineering, hwkim@kw.ac.kr, 종신회원
논문번호 : KICS2014-05-199, Received May 27, 2014; Revised August 14, 2014; Accepted August 14, 2014

Ⅰ. 서 론

웹 2.0의 특징은 '참여, 공유, 개방' 이라는 키워드로 나타낼 수 있으며 '개방' 이라는 키워드의 대표적인 예로 Open API가 소개되고 있다.[1] 웹에서의 API는 웹 서비스를 사용하기 위한 규약 혹은 규칙을 의미하며 , 누구나 사용할 수 있도록 공개된 API를 Open API라고 한다.[2] Open API는 서로 다르게 제공되는 정보나 서비스를 섞어서 하나의 서비스로 제공하는 '매쉬업(Mash-up)' 서비스에 많이 사용되고 있으며, Open API가 다양해짐에 따라 Open API를 활용한 매쉬업 서비스도 다양해지고 있다.[3] 하지만 기존의 Open API 기반 매쉬업 서비스는 개발자 위주로 제작되기 때문에 모든 사용자의 취향과 목적에 맞춰 제공할 수 없었다. 따라서 개인 사용자의 목적에 맞도록 사용자가 직접 제작할 수 있는 자동 매쉬업 제작 플랫폼이 필요하였다. 자동 매쉬업 뿐만 아니라 일반적인 매쉬업 서비스를 제공하기 위해서는 우선적으로 Open API에 대한 정보를 획득할 필요가 있다. Open API 정보와 매쉬업 서비스 정보를 제공하는 programmableWeb[4]에는 2014년 3월 24일 기준으로 11,196개의 Open API가 명시되어있다. Open API는 REST, SOAP, XML-RPC 등 다양한 프로토콜 및 아키텍처 스타일로 제공되며 programmableWeb에서는 API 중 REST 기반은 69%, SOAP 22%, JavaScript 5%, XML-RPC 2%로 각각 집계된다. 그 중 REST기반의 Open API는 웹 서비스를 위한 프로토콜 규약 없이 HTTP 기반으로 쉽게 동작하기 때문에 많은 Open API가 REST 기반으로 제공된다. 또한 매쉬업 서비스 제공에 있어서도 REST 기반의 Open API 가 더 많이 사용된다. 이와 같이 REST 기반 Open API가 갖는 여러 가지 장점으로 인해 기존의 자동 매쉬업 서비스에 대한 연구는 REST 기반의 Open API가 많이 사용되었다. [5-7] Open API를 이용한 자동 매쉬업에서 가장 큰 이슈사항은 매쉬업의 가능성 판별 방법에 대한 것이며, 기존 연구에서는 입출력 파라미터 이름의 구문적 유사도 비교를 통해 매쉬업 가능성을 판별하였다.[6,7] 하지만 유사도 비교만을 이용하여 매쉬업 가능성을 판별할 경우에는 입출력 파라미터의 특성을 고려하지 못하기 때문에 잘못된 매쉬업 결과가 생성되는 문제점이 발생하였다.

따라서 본 논문에서는 REST API의 입출력 파라미터 특성을 고려한 자동 매쉬업 서비스의 생성 방법에 대해 제안하였다. 먼저 API 정보와 매쉬업 가능한 요소의 정보를 저장하기 위해 선행 연구[7,8]에서 제안한 온톨로지 모델링을 사용하였다. 그리고 잘못된 매쉬업 결과를 줄일 수 있도록 입출력 파라미터의 특성을 정의하고 이를 선행 연구에서 제안한 온톨로지에 반영하여 잘못된 매쉬업 결과를 줄일 수 있는 확장된 온톨로지를 제안하였다. 마지막으로 확장된 온톨로지를 이용한 자동 매쉬업 서비스의 제공 방법을 제시하고 서비스 구현 결과를 보였다.

본 논문의 구성은 다음과 같다. 본론 1절에서는 본 논문에서 필요한 관련 기술에 대해 간략하게 설명하고 2절에서는 기존 논문에서 제안한 Open API 정보 및 매쉬업 정보를 저장하는 온톨로지 모델링에 대해 기술한다. 3절에서는 입출력 파라미터의 특성을 고려하지 않은 온톨로지 방법에서 생길 수 있는 문제점과 이를 보완한 온톨로지 확장 모델에 대해 기술한다. 4절에서는 확장 모델을 기반으로 자동 매쉬업 서비스의 제공 방법에 대해 설명한다. 5절에서는 온톨로지 확장 모델을 바탕으로 자동 매쉬업 생성 과정을 실험 분석하고 결론으로 마무리한다.

Ⅱ. 본 론

2.1 관련 연구

2.1.1 REST 기반의 Open API 매쉬업

REST(Representational State Transfer)는 웹 페이지 접근에 관한 일종의 설계 방식을 의미한다. 기존의 웹 아키텍처는 웹이 갖는 본래 설계의 우수성을 활용하지 못하였기 때문에 네트워크 기반의 아키텍처인 REST를 제안하여 웹의 장점을 최대한 활용할 수 있도록 하였다.[9] REST는 서버에 결과를 요청하는 'Request' 전달 방식과 함께 URI를 구조적으로 표현하고 HTTP의 기본 메소드인 GET/PUT/POST/DELETE 만으로 리소스에 접근이 가능하다. 그러므로 REST 기반의 Open API는 HTTP 메쏘드와 URI만 이용하여 요청을 할 수 있으므로 다른 프로토콜 기반의 Open API 보다 간단하게 사용이 가능하다.

'Request'와 'Response'를 기준으로 REST 기반 Open API를 이용한 매쉬업 서비스는 다음과 같은 구조로 합성이 가능하다.

그림 1과 같이 어떤 하나의 웹 서비스가 갖는 응답(Response)의 'Output Data'는 또 다른 웹 서비스 요청(Request)의 'Input Data'로 전달되어 서로 다른 기능을 수행하는 Open API 1과 Open API 2는 하나의 매쉬업 서비스로 합성될 수 있다. REST 기반의 Open API는 'Output Data'가 다른 API의 URI가 갖는

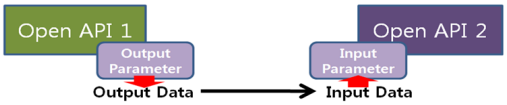

그림 1. 매쉬업 서비스 구조
Fig 1. Mash-up Service Structure

'Input Data'로 전달되는 구조를 갖기 때문에 매쉬업 과정이 다른 프로토콜 기반의 API보다 용이하다는 이점이 존재한다.

2.1.2 Open API의 온톨로지

온톨로지는 개념과 상관관계를 기술하는 도구이다. 즉, 특정한 단어가 나타내는 개념의 의미를 이해하고 지식을 표현하는데 온톨로지가 사용된다. 또한 특정 영역의 지식을 모델링하고 처리하여 구성원 간의 지식 공유 및 재사용을 가능하게 하는 중요한 요소이다. 따라서 온톨로지는 지식 표현, 정보시스템 개발, 시맨틱 웹 구축 등의 다양한 용도로 사용된다.

온톨로지의 정보 공유 및 재사용과 같은 장점을 이용하여 다양한 용도로 정보를 표현할 수 있기 때문에 Open API에 대한 정보도 온톨로지로 표현하여 온톨로지의 이점을 그대로 적용할 수 있다. 또한 온톨로지는 계층적으로 지식을 표현하는 것이 용이하다. REST 기반의 Open API는 HTTP 메쏘드와 계층적 구조로 표현할 수 있는 URI로 구성되어있기 때문에 온톨로지의 계층적 구조를 쉽게 적용할 수 있다. 그리고 온톨로지를 이용하여 구체적으로 나타나 있지 않는 암시적 의미나 내재하는 규칙까지도 이해하여 서로 다른 API 간의 정보 공유도 가능하다.

온톨로지 표현 언어로는 서브젝트(Subject), 프로퍼티(Propery), 오브젝트(Object) 트리플 구조를 갖는 RDF(Resource Description Frame) , 리소스간의 계층 구조 형성이 가능한 RDFs(RDF Schema) , RDF와 RDFs의 한계를 보완한 OWL (Web Ontology Language)이 존재한다.

2.2 자동 매쉬업 서비스를 제공하기 위한 온톨로지 모델링

Open API 기반의 자동 매쉬업 서비스 제공을 위해서는 Open API 내에서 매쉬업에 필요한 요소를 선별할 수 있어야 한다. 선별 과정에 앞서, Open API 에 대한 정보의 저장은 필수적이며 이를 위해 본 절에서는 선행 연구[7, 8]에서 제안하였던 온톨로지 모델링 방법에 대해 기술한다. 선행 연구에서 제안하였던 온톨로지 모델은 두 가지 부분으로 분류할 수 있다. 첫 번째는 Open API를 서비스, 메쏘드, 입출력 파라미터별로 분류하여 계층적으로 나타낸 'Open API 정보 온톨로지', 두 번째는 매쉬업 가능한 정보를 나타낸 '매쉬업 정보 온톨로지'이다.

2.2.1 Open API 정보의 계층적 온톨로지 모델링

Open API 에는 프로토콜, 서비스 종류, 메쏘드, 입출력 파라미터 등 많은 정보를 포함하고 있다. 예를 들어 , flickr[13]의 API를 살펴보면 모든 메쏘드는 REST 아키텍처 스타일[10]을 지원한다. REST 아키텍처 스타일을 기반으로 하는 flickr에서는 사진과 관련된 서비스를 제공하며 서비스 내에서는 등록된 사진의 위치 정보 , 등록자의 정보, 사진을 관리하는 정보 등 여러 기능을 제공하는 메쏘드(Method)들로 이루어져 있다. 또한 각각의 메쏘드에는 요청(Request)을 위해 구성되는 URI 정보와 응답(Resopnse)을 나타내기 위한 출력 형식이 존재한다. 이 뿐만 아니라 'flickr'와 같이 Open API를 제공하는 서비스 기관의 이름, Open API를 지원하는 프로그래밍 언어의 종류 등 Open API에 관련된 여러 정보들이 존재한다. 이 중 선행 연구에서는 자동 매쉬업을 위해 필요한 Open API 정보로 서비스 종류, 메쏘드, 요청과 응답 정보를 사용하여 계층적인 온톨로지로 구성하였다.

먼저 메쏘드와 요청 및 응답의 정보를 계층적 온톨로지로 모델링하는 방법에 대해 살펴본다. REST 기반의 Open API는 HTTP 메쏘드와 URI로 간단히 접근할 수 있기 때문에[11] 요청에 대한 온톨로지 모델링은 URI 정보를 이용하여 구축할 수 있다. URI의 경로는 UNIX의 파일 시스템과 같은 계층 구조를 가지고 있으며[12] 온톨로지 또한 계층 구조로 모델링 할 수 있기 때문에 REST 형식의 Open API 정보를 계층적 온톨로지 모델로 설계가 가능하다.

예를 들어, twitter API의 요청은 URI spec에 따라 표 1과 같이 나타낼 수 있다. 서비스를 요청할 때 URI는 'URI scheme', 'Host Name' , 'Path'를 지닌 일정한 URI를 갖고 있으며 쿼리 파라미터(Query Parameter)의 변경에 따라 메쏘드 내에서 다양한 요청이 이루어진

표 1. twitter API 요청의 URI spec
Table 1. URI Specification of twitter API Request

URI Specification	Example
URI Scheme	http
Host Name	api.twitter.com/
Path	search/tweets.json
Query Parameter	q=seoul

다. 여기서 변하지 않는 일정한 주소는 'Base URI'로 정의한다.[12] 이와 같이 각각 다른 기능을 수행하는 메쏘드마다 Base URI는 다르게 구성되므로 '메쏘드 또는 Base URI'를 기준으로 Open API 정보를 분류하여 온톨로지로 모델링 할 수 있다. 메쏘드마다 각각의 쿼리 파라미터를 가지며 하나의 메쏘드가 여러 쿼리 파라미터를 가질 수 있기 때문에 메쏘드는 쿼리 파라미터보다 상위에 있는 계층적 구조로 표현할 수 있다.

또한 각각의 메쏘드는 요청 뿐만 아니라 응답도 갖기 때문에 이에 대한 정의도 필요하다. 요청에 대한 응답으로 XML과 JSON등의 데이터 포맷을 따르며 메타 데이터를 포함한다. 그림 2와 같이 XML 데이터 포맷을 따르는 예에서는 요소(Element) 및 속성(Attribute)을 메타 데이터로 정의할 수 있다.

메타 데이터도 각각의 메쏘드에 대해 존재하므로 쿼리 파라미터와 같이 메소드를 상위 계층으로 갖는 구조로 온톨로지 모델링한다.

따라서 메쏘드마다 요청에는 쿼리 파라미터, 응답에는 메타 데이터가 존재하며 이를 Open API 메쏘드의 입출력 파라미터로 정의할 수 있다. 입력과 출력을 따로 구별할 수 있도록 온톨로지 모델링에서는 메쏘드 계층과 입출력 파라미터 계층의 관계를 각각 'InputValue', 'OutputValue' 프로퍼티로 정의한다. 그리고 메쏘드의 상위 계층에는 어떤 서비스와 관련된 메쏘드인지를 명시해주기 위해 '서비스 카테고리' 계층을 정의한다. 그림 3은 flickr API의 'Open API 정보 온톨로지'를 표현한 것으로 서비스 카테고리, 메쏘드, 입출력 파라미터 순의 계층적 구조를 나타내고 있다.

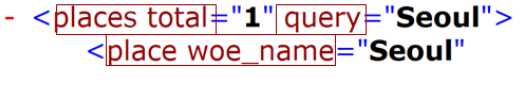

그림 2. XML 출력 예시
Fig. 2. Example of XML Response

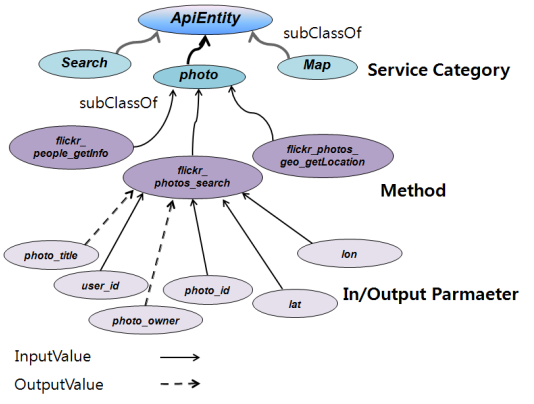

그림 3. Open API 정보 온톨로지
Fig. 3. Open API Information Ontology

2.2.2 매쉬업 정보 온톨로지의 모델링

앞서 기술하였듯이 REST 기반 Open API의 자동 매쉬업 서비스를 제공하기 위해서는 어떤 입출력 파라미터들이 매쉬업 가능한지 알 수 있어야 한다. 이를 위해 매쉬업이 가능한 입출력 파라미터의 정보를 나타내는 '매쉬업 정보 온톨로지'에 대해 기술한다.

입출력 파라미터가 서로 매쉬업이 가능하기 위해서는 그림 1과 같이 파라미터 간의 데이터 전달을 통해 매쉬업 서비스가 정상적으로 수행될 수 있어야하며 파라미터의 이름이 의미적, 구문적으로 유사해야 한다. 즉, 매쉬업 가능한 입출력 파라미터들은 의미적으로 유사한 이름을 가지고 있기 때문에 별도의 그룹으로 생성할 수 있으며 이들을 대표하는 단어로 그룹을 나타낼 수 있다.

따라서 같은 그룹 내에 존재하는 입출력 파라미터들은 의미적으로 유사하기 때문에 서로 모두 매쉬업이 가능하며 각각의 그룹은 매쉬업 가능한 입출력 파라미터 정보를 가질 수 있게 된다. 이를 계층적인 온톨로지로 모델링하면 그룹을 나타내기 위해 새로운 계층을 생성할 수 있고 그림 4와 같이 정의할 수 있다. 그룹을 나타내기 위해 새로운 계층은 '파라미터 그룹' 계층으로 정의하며 '입출력 파라미터' 계층과 'ParameterOf' 프로퍼티를 갖도록 정의한다.

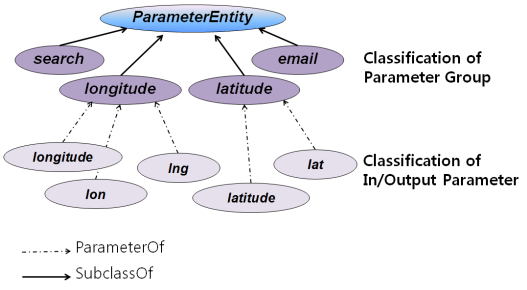

그림 4. 매쉬업 정보 온톨로지
Fig. 4. Mash-up Information Ontology

2.3 입출력 파라미터 특성을 고려한 확장된 온톨로지 모델링

본 절에서는 입출력 파라미터의 특성에 대해 제시하고 이를 고려하지 않은 온톨로지 모델링에서 발생할 수 있는 문제점에 대해 기술한다. 그리고 문제점을 해결할 수 있도록 선행 연구에서 제안한 온톨로지에

입출력 파라미터의 특성을 적용하여 확장된 온톨로지를 제시한다.

2.3.1 입출력 파라미터의 특성

이전 절에서 제시한 '파라미터 그룹' 내에는 서로 매쉬업이 가능한 입출력 파라미터로 구성되어 있다. 즉, 같은 파라미터 그룹 클래스를 갖는 입출력 파라미터는 서로 매쉬업이 가능하다.

예를 들어, 위,경도(latitude, longitude)를 나타내는 입출력 파라미터는 다음과 같이 매쉬업 정보 온톨로지에 명시된다. foursquare[14]는 'latitude'와 'longitude'를 출력 파라미터로 'lat'과 'lng'으로 정의하고 있으며, flickr는 입력 파라미터로 foursquare와 비슷하게 'lon'과 'lat'으로 정의한다. 따라서 그림 4와 같이 foursquare의 'lat', 'lng'와 flickr의 'lat', 'lon'은 각각 latitude, longitude로 정의한 파라미터 그룹에 포함되기 때문에 foursquare 위경도 출력 데이터가 flickr의 입력 데이터로 전달되는 매쉬업 서비스를 생성할 수 있다. 하지만 모든 웹 서비스에서 제공하는 입출력 파라미터들이 제안한 '매쉬업 정보 온톨로지' 모델을 충족시키는 것은 아니다. twitter의 출력 파라미터 'user_id'와 flickr의 입력 파라미터 'user_id' 예는 이와 같은 문제점을 잘 보여준다. 'user_id' 파라미터는 온톨로지로 명시할 때와 데이터의 전달이 수행될 때 각각 문제점이 발생한다.

먼저 twitter의 'user_id'와 flickr의 'user_id'를 온톨로지로 명시할 때, 매쉬업을 위해 두 파라미터는 서로 다른 API의 파라미터이지만 파라미터의 이름이 같기 때문에 온톨로지에서 'user_id'라는 이름을 갖는 하나의 리소스로 정의된다. 따라서 'user_id'는 twitter 메쏘드, flickr 메쏘드 두 개의 서로 다른 메쏘드와 프로퍼티 관계를 갖게 된다. 이와 같이 같은 이름을 갖는 파라미터는 서로 다른 웹서비스에서 제공하더라도 선행 연구에서 제안한 온톨로지 상에서는 구별할 수 없고 이름이 충돌되는 문제점이 발생한다.

두 번째 문제점은 'user_id'가 갖는 데이터가 매쉬업을 위해 전달될 때 발생한다. flickr의 'user_id'는 사용자를 구분하기 위해 고유한 데이터 값을 부여하고 twitter 역시 사용자를 구분하기 위해 twitter API 내에서만 유효하고 고유한 데이터 값을 부여한다. 앞서 제시한 첫 번째 문제점을 해결하여 flickr의 'user_id'와 twitter의 'user_id'를 서로 구별할 수 있는 방법이 존재하더라도 'user'id'라는 공통된 단어가 존재하기 때문에 구문적, 의미적으로 유사하여 두 파라미터는 매쉬업 가능하다고 판단한다. 하지만 'user_id'가 갖는 데이터는 각각의 웹 서비스 내에서만 고유한 데이터이므로 다른 웹 서비스에 이용될 경우 잘못된 매쉬업 결과가 나타날 수 있다. 예를 들어 '95243107@N03'과 같은flickr의 고유한 'user_id' 데이터 값이 twitter API 의 'user_id' 데이터 값으로 입력되는 경우 twitter는 결과를 반환하지 못하거나 flickr의 데이터와는 전혀 다른 데이터 값으로 인식하는 문제점이 발생하게 된다.

이와 같은 문제점은 선행 연구에서 제시한 온톨로지에서 입출력 파라미터의 특성을 고려하지 못하였기 때문에 발생한다. 따라서 본 논문에서는 입출력 파라미터의 특성을 고려할 수 있는 온톨로지 모델링을 위해 다음과 같이 입출력 파라미터가 갖는 특성을 정의한다.

1. 입출력 파라미터는 서로 다른 웹 서비스에서 정의한 파라미터라도 이름이 구문적으로 일치하는 특성을 갖는다.
2. 일부 입출력 파라미터 데이터에는 유효 범위를 갖는 특성이 존재한다.

2.3.2 입출력 파라미터 특성을 고려한 온톨로지 모델링의 확장

파라미터의 구문적, 의미적 비교만 수행하여 매쉬업 가능성을 판별할 경우 입출력 파라미터의 특성을 고려하지 못하기 때문에 앞서 제시한 문제점이 드러난다. 따라서 기존 연구의 온톨로지를 확장하여 앞서 제시한 파라미터 특성을 고려할 수 있는 온톨로지 모델링에 대하여 제안한다.

앞서 정의한 입출력 파라미터의 첫 번째 특성 때문에 온톨로지에서 입출력 파라미터가 충돌하는 문제점을 발생시켰다. 이를 해결하기 위해서 온톨로지에 API를 제공하는 기관의 이름을 입출력 파라미터의 네임 스페이스(Namespace)로 정의한다. 네임 스페이스는 이름은 같지만 다른 의미를 가지고 있는 요소나 속성이 서로 충돌하는 것을 예방하기 위한 방법이다. 따라서 flickr 의 'user_id'는 'flickr:user_id' twitter의 'user_id'는 'twitter : user_id' 로 정의할 수 있다. 이로 인해 이름이 일치하는 파라미터들의 구별이 가능하며 온톨로지에서 발생하는 이름의 충돌 문제를 피할 수 있다.

그리고 입출력 파라미터의 두 번째 특성으로 인해 발생한 문제점에 대해서는 온톨로지에서 새로운 프로퍼티를 정의하여 해결한다. 'user_id' 파라미터 예에서 이름이 일치하는 문제점은 네임 스페이스를 정의하여

해결하였지만, 매쉬업 가능한 API를 탐색할 때에는 네임 스페이스를 정의하더라도 두 번째 문제점이 해결되지 않는다. 예를 들어 'flickr:user_id'와 'twitter:user_id'의 텍스트 유사도를 비교한다면 user_id 부분이 일치하기 때문에 매쉬업 가능하다고 판별할 수 있는 문제점이 발생한다. 따라서 고유한 데이터 값을 갖는 'user_id'처럼 입출력 파라미터가 갖는 데이터 값이 특정 웹 서비스 내에서만 유효하다면, 이를 구분하기 위해 별도의 식별이 필요하다.[15] 그러므로 데이터의 유효 범위를 한정하기 위해 프로퍼티를 추가적으로 정의한다. 기존의 입출력 파라미터 계층과 메쏘드 계층 간의 프로퍼티는 'InputValue'와 'OutputValue'로 정의하였다. 하지만 데이터의 유효 범위를 갖는 파라미터를 위해서 'uniqueInputValue'와 'uniqueOutputValue' 프로퍼티를 추가적으로 정의한다. 이와 같이 추가적인 프로퍼티를 정의하여 온톨로지에서 유효한 데이터 범위를 갖는 파라미터들을 구분할 수 있다.

입출력 파라미터의 특성으로 인해 발생하는 문제점을 해결하기 위해서 기존의 온톨로지 모델을 수정 및 확장한 결과는 그림 5와 같다. 첫 번째 문제점을 해결하기 위해서 각각의 입출력 파라미터는 서비스 기관의 이름을 네임 스페이스로 갖는 것을 확인할 수 있다. 입출력 파라미터 뿐만 아니라 메쏘드 역시 서비스 기관에 따라서 분류되기 때문에 네임 스페이스를 가질 수 있다. 이러한 분류는 특정 영역을 구체화한다는 관점에서 'API 정보 온톨로지'를 추상화의 정도에 따라 상위 온톨로지와 하위 온톨로지로 구분할 수 있다.[16] 따라서 메쏘드와 입출력 파라미터 정보를 하위 온톨로지인 'Domain Ontologies'로 정의하여 특정 영역에 관한 온톨로지로 나타낼 수 있다. 상위 온톨로지는 다양한 영역에 적용될 수 있는 기본적이고 보편적인 개념을 나타내는 정보로 'Upper Ontology'로 정의한다. 또한 두 번째 문제점을 해결하기 위해서 'place_id', 'photo_id','user_id'와 같이 데이터의 유효 범위를 갖는 입출력 파라미터는 새로 정의한 프로퍼티를 이용하여 다른 입출력 파라미터와 구별할 수 있음을 보인다.

2.4 매쉬업 서비스 구조 분석 및 자동 매쉬업 제공 방법

앞서 살펴 본 것과 같이 매쉬업의 가능성을 판별하는 요소로 입출력 파라미터를 사용하였다. 데이터의 전달로 매쉬업 서비스를 제공할 수 있으며 기존에 제공되는 매쉬업 서비스도 이와 같은 동작을 수행한다. 하지만 기존 매쉬업 서비스는 데이터 전달 구조에 따라 분류될 수 있고 자동적으로 매쉬업 서비스를 제공할 때는 분류된 형식을 모두 고려해야 한다. 따라서 본 절에서는 이미 생성된 기존 매쉬업 서비스의 데이터 전달 구조를 분석하고 앞서 정의한 온톨로지를 이용하여 자동으로 매쉬업할 수 있는 방법에 대해 기술한다.

2.4.1 기존 매쉬업 서비스의 데이터 전달 구조

REST 기반의 Open API를 이용한 자동 매쉬업 서비스는 파라미터의 정보를 이용하여 합성 가능함을

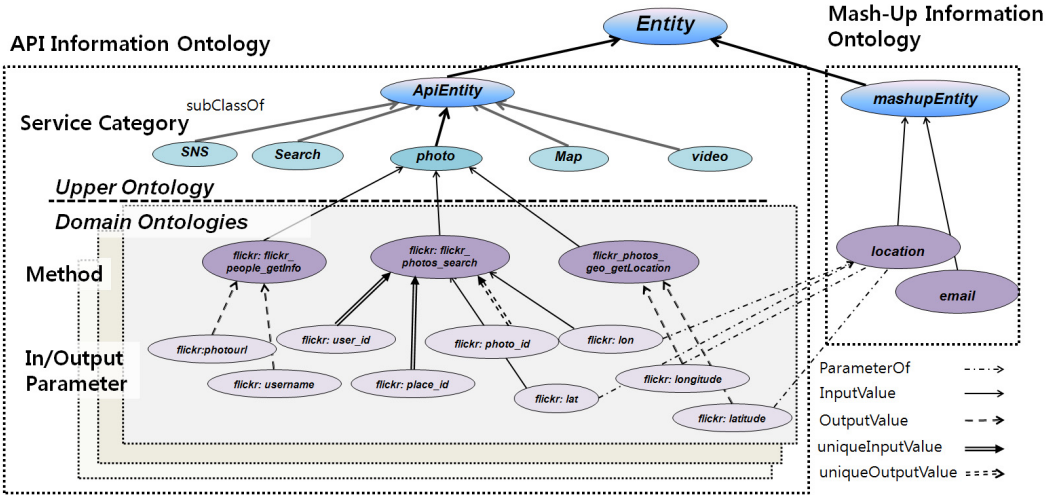

그림 5. Open API 정보의 계층적 온톨로지 모델링
Fig. 5. Hierarchical ontology modeling of Open API information

판별 할 수 있다. 기존의 매쉬업 서비스 또한 합성 가능한 파라미터간의 데이터 전달을 통해 수행된다. 하지만 기존의 매쉬업 서비스는 파라미터간의 데이터 전달 형태에 따라 구조의 차이점을 갖고 있으며 구조의 차이점을 분석하기 위해 programmableWeb에 게시되어 있는 기존의 매쉬업 서비스를 사용하였다.

기존에 제공되는 매쉬업 서비스를 바탕으로 데이터 전달 형태에 따른 분류는 그림 6을 바탕으로 3가지로 분류할 수 있다. 첫째, 'Type A'는 사용자의 입력 값 또는 미리 정해진 데이터 값을 여러 웹 서비스에 요청하고 그에 대한 응답들을 취합하여 보여주는 형태이다. 예를 들어 'Amazon and ebay compairison Shopping[17],'과 같은 매쉬업 서비스는 사용자의 제품과 관련된 쿼리 값을 Amazon API와 eBay API에 각각 요청하고 결과 값을 동시에 보여줌으로서 제품의 가격을 비교해볼 수 있는 서비스이다.

두 번째, 'Type B'는 하나의 메쏘드에서 출력되는 데이터 값이 다른 메쏘드의 입력 값으로 전달되는 형태이다. 'Auction Search Kit'[18] 는 eBay API와 Google map API를 합성한 서비스이다. eBay의 메쏘드는 등록된 상품의 위경도 정보를 반환하고, Google Map API에서 제공하는 메쏘드는 위경도 정보를 입력받아 지도 상에 해당 지점을 표기한다.

마지막으로 'Type C'는 두 형태에 해당되지 않는 매쉬업 서비스로 인터페이스를 변형하거나 데이터 전달을 외부에서 알아볼 수 없는 형태이다.

위와 같은 분류 기준에 따라 programmableWeb에 게시되어 있는 82개의 매쉬업을 분류한 결과는 그림 7과 같다. 분류 결과인 그림 7을 통해서 Type A와 Type B의 비율이 전체 제공되는 매쉬업 서비스 중 약 81%에 해당하는 것을 확인할 수 있다. 이는 데이터의 전달을 이용한 매쉬업 서비스가 많은 비율로 기존에도 제공되고 있으며, 자동 매쉬업 서비스를 제공하기 위해서는 두 가지의 형태가 모두 고려되어야 한다는 것을 확인할 수 있다.

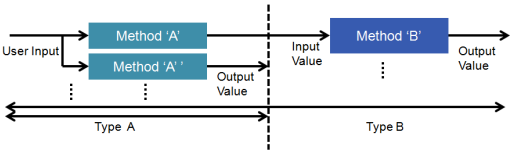

그림 6. 매쉬업 구조의 분류
Fig. 6. Classification of Mash-up Structure

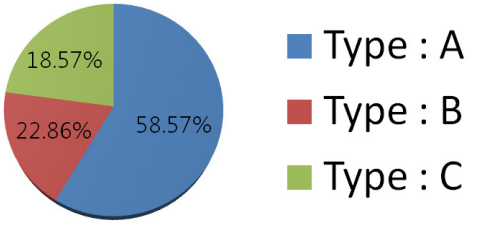

그림 7. 매쉬업 구조의 비율
Fig. 7. Percentage of Mash-up structure

2.4.2. 매쉬업 가능한 파라미터의 탐색 알고리즘

자동 매쉬업을 생성하기 위해서는 서로 매쉬업 가능한 입출력 파라미터들을 탐색하는 방법이 필요하다. 이를 위해 확장된 온톨로지를 기반으로 그림 8과 같이 매쉬업 가능한 입출력 파라미터들을 탐색하는 알고리즘에 대해 제안한다.

우선 사용자가 어떤 서비스와 관련된 매쉬업 서비스를 받고자 하는지 알기 위하여 초기 값으로 사용자의 서비스 선택을 입력 받는다. 그리고 입력받은 'Service A'에 해당하는 입출력 파라미터를 "Search_QueryParameter" 함수를 이용하여 탐색한다. 서비스 카테고리, 메쏘드, 입출력 파라미터 순으로 구성되어 있는 'Open API 정보 온톨로지'를 기반으로, 'Search_QueryParameter' 함수는 사용자에게 입력받은 서비스의 하위 계층에 존재하는 입출력 파라미터 탐색을 단계적으로 수행한다. 온톨로지의 트리플 관계에 따라 탐색된 입출력 파라미터가 'parameterOf' 프로퍼티를 갖는 경우, 그림 5와 같이 오브젝트(Object)는 파라미터 그룹을 갖게 되며 매쉬업 가능하다고 판별할 수 있다. 유효 범위를 갖는 파라미터의 매쉬업 구분을 위해 유효 범위를 갖지 않는 파라미터와 구분을 하여 저장한다.

탐색된 파라미터 그룹 'Normal', 'Unique'에는 서로 매쉬업 가능한 입출력 파라미터가 하위 계층에 존재한다. 다른 메쏘드의 입력 파라미터로 데이터가 전달되기 위해서는 각각 프로퍼티 'InputValue'와 'uniqueInputValue'를 갖는지를 판단한다. 조건을 만족시키는 파라미터를 저장하고 사용자에게는 입력받은 서비스와 저장한 파라미터에 해당하는 서비스가 서로 매쉬업 가능하다는 결과를 보여준다.

2.5 온톨로지 모델링을 활용한 자동 매쉬업 서비스 구현

이번 절에서는 설계한 온톨로지를 바탕으로 사용자에게 자동으로 매쉬업 서비스를 제공한 결과에 대해 기술한다. 사용자가 매쉬업 서비스를 받는 과정은 그림 9와 같다. 사용자는 매쉬업 받고자 하는 서비스 종류

```
service_name   A = user inputs service 'A' ;
Resource[] query_parameter= func Search_QueryParameter(service_name   A ) ;

for(i=0 ;i<query_parameter.size() ;i++ )    {
    if( Get Property (Subject == query_parameter[i]) ) == "parameterOf" ) {
        if( Get Property(Subject == query_parameter[i]) == "uniqueInputValue" || "uniqueOutputValue" ) {
                Resource Unique[ j ] = Get Object (subject == query_parameter[i] && property == ("parameterOf" )  ;
        else if( Get Property(Subject == query_parameter[i]) == "InputValue" || "OutputValue" ) {
                Resource Normal [ h ] = Get Object (subject == query_parameter[i] && property == ("parameterOf" );
}
for (j=0; j<Normal.size() ; j++ )  {
    if( Get Subject( (Property == "parameterOf")&& Object == Normal ) != null ) {
        if(Get Object (Property == "uniqueInputValue") {
                save the subject mash_up_parameter [ n ] ; n++; }        }        }
for (h=0; h< Unique.size()   ; h++ )  {
    if( Get Subject( (Property == "parameterOf")&& Object == Unique ) != null) {
        if(Get Object (Property == "InputValue") {
                save the subject mash_up_parameter [ n ] ; n++; }        }        }

search the service name 'B' of mash_up_parameter[n];
Mash up of Service name 'A' and 'B ' is available
```

그림 8. 매쉬업 가능한 입출력 파라미터의 탐색 알고리즘
Fig. 8. Search Algorithm of Mash-up possible In/Output parameters

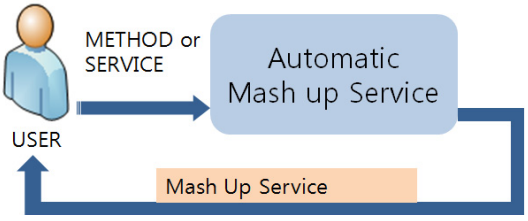

그림 9. 자동 매쉬업 서비스의 제공 과정
Fig. 9. Process of providing automatic Mash-up service

나 메쏘드를 입력하면 합성된 서비스의 결과를 얻을 수 있다. 서비스 종류의 다양성을 고려하고 사용자의 간단한 입력 처리를 위해서 Open API를 서비스 카테고리 별로 분류하였기 때문에, 사용자는 서비스 입력만으로도 매쉬업 서비스를 제공받을 수 있다.

본 논문에서는 매쉬업 가능한 입출력 파라미터가 자동으로 파라미터 그룹을 생성한다고 가정한다. 그리고 자동 매쉬업 서비스 구현은 Jena 2.6.4[19] 라이브러리와 Java 언어를 기반으로 작성되었다.

2.5.1 사용자의 서비스 카테고리 선택 및 API 검색

구현된 플랫폼은 매쉬업에 많이 사용되는 서비스 항목 'Photo', 'SNS', 'Mapping', 'Video', 'Search'를 선별하여 테스트 하였다.

서비스를 나타내는 리스트는 미리 설계한 온톨로지에서 서비스 카테고리를 검색하여 그림 10과 같이 사용자에게 보여준다. 사용자에게 서비스 카테고리를 입력 받은 후 해당 카테고리 내에 있는 API를 검색하게 된다. 'Photo' 카테고리를 입력 받았다고 가정한다

```
The Service Category
1.Photo
2.SNS
3.Map
4.Video
5.Search
 Choose the Number of the Service Category :
1
```

그림 10. 서비스 카테고리 입력을 위한 단계
Fig. 10. Steps for inputting the category of service

면 flickr API를 포함한 사진과 관련된 웹 서비스의 메쏘드들이 검색된다.

2.5.2 매쉬업 판별 알고리즘 수행 및 탐색

메쏘드의 기능에 따라 서비스의 분류가 세분화되어 있는 경우, 세분화된 서비스에 적은 수의 메쏘드가 포함되어 있기 때문에 매쉬업 가능한 메쏘드를 한정지을 수 있다. 하지만 본 논문에서는 위에서 정의한 5개의 서비스로 한정지으며 구현의 편의성을 고려하여 사용자에게 메쏘드도 선택하여 입력받을 수 있게 구현하였다. 구현에서는 사용자가 filckr의 'flickr.photos.search' 메쏘드를 선택한 것으로 가정하였다.

메쏘드가 선택되어 입출력 파라미터가 검색되면 프로퍼티로 'parameterOf'를 갖는지 탐색한다. 선택한 'flickr.photos.search' 메쏘드와 'flickr.geo.getLocation' 메쏘드의 'parameterOf' 프로퍼티를 갖는 입출력 파라미터에 대해 중심적으로 나타내면 표 2와 같다. 두 메쏘드 간의 매쉬업이 가능한 입출력 파라미터는 이탤릭체로 표시하였다.

표 2. flickr 메쏘드의 입출력 파라미터
Table 2. In/Output parameters of flickr method

'flickr.photos.geo.getLocation'
Input Parameter
api_key ,*photo_id* , extras
Output Parameter
location:latitude, location:longitude, locaction: context, *location:place_id, location:woeid, county:place_id* …
'flickr.photos.search'
Input Parameter
api_key, user_id, tags, text, group_id, *woe_id, place_id*, media, *lat, lon*, radius …
Output Parameter
photo:id, photo:owner, photo:secret, photo:server, photo:farm, photo:title …

표 3. flickr.photos.search 메쏘드의 요청 및 응답
Table 3. Request and Response of flickr.photos.search method

Case that can mash-up
Type A
Nothing
Type B
1. flickr.photos.search 'photo:id' -> flickr.photos.geo.getLocation 'photo:id' 2. flickr.photos.geo.getLocation 'location:woeid' -> flickr.photos.search 'woe_id' 3. flickr.photos.geo.getLocation 'location:place_id' ->flickr.photos.search 'place_id' 4. flickr.photos.geo.getLocation 'location:latitude' ,'location:longitude' ->flickr.photos.search 'lat','lon'

'parameterOf' 프로퍼티를 갖는 파라미터 중에서 입출력 파라미터가 데이터의 유효 범위 특성을 갖는지 확인한다. 이를 바탕으로 두 메쏘드 간의 매쉬업 가능한 경우에 대해 나타내면 표 3과 같다. 두 메쏘드의 매쉬업 가능한 입출력 파라미터 중 'phto_id' 에를 살펴보면 다음과 같은 과정을 수행하며 매쉬업 가능한 메쏘드의 탐색 결과를 얻을 수 있다.

'photo_id'는 flickr API 내에서 고유한 데이터를 갖게 되므로 입출력 파라미터는 'unique-' 프로퍼티를 갖는다. 'photo:id'는 그림 13과 같이 'unique utputValue' 프로퍼티와 'flickr.photos.search' 메쏘드를 오브젝트로 갖고, 'parameterOf' 프로퍼티와 'flickr_photo_id'인 파라미터 그룹을 오브젝트로 가진다. 그림 8과 같이 'flickr_photo_id' 파라미터 그룹은 'Unique[]' 행렬에 저장이 된다. 그 다음으로는 'flickr_photo_id' 파라미터 그룹을 이용해 'flickr. hotos.search' 메쏘드와 매쉬업 가능한 파라미터를 탐색하며, 파라미터 탐색 조건은 'flickr_photo_id' 오브젝트와 'parameterOf' 프로퍼티를 갖고 또한 'uniqueInputValue' 프로퍼티를 갖는 것이다. 이 중 'flickr.photos.geo.getLocation' 메쏘드의 'flickr:photo_id' 입력 파라미터는 'uniqueInputValue' 프로퍼티를 가져 조건을 만족하므로 매쉬업 가능한 파라미터로 저장을 한다. 결과적으로 'flickr.photos.search' 메쏘드와 'flickr.photos.geo.getLocation' 메쏘드는 'photo_id' 파라미터를 이용하여 매쉬업 서비스를 생성할 수 있다.

2.5.3 사용자의 데이터 입력 및 데이터 파싱

앞서 살펴보았듯이 매쉬업 가능 메쏘드 탐색 알고리즘을 이용하여 'flickr.photos.geo.getLocation'과 'flickr.photos.search' 메쏘드가 서로 매쉬업 가능하다는 것을 알 수 있었다. 이 사실을 바탕으로 사용자에게 매쉬업 서비스가 제공되기 위해서는 메쏘드 간의 데이터 전달 과정이 필요하다. 'photo_id' 파라미터를 이용하여 'flickr.photos.search' 메쏘드와 'flickr. hotos.geo.getLocation' 메쏘드가 매쉬업 되는 경우 'flickr.photos.search' 메쏘드의 입력 값으로 place_id가 필요하다. 그림 11과 같이 URI를 생성하여 요청하면 'flickr.photos.search' 의 출력 결과가 XML 데이터 포맷으로 반환된다. 매쉬업 가능 메쏘드 탐색 알고

```
 Created URI : http://api.flickr.com/services/rest/?method=flickr.photos.search&api_k
ey=d34519fb351b60da76f04b28ff9abb06&place_id=Y1pL8yxTUb7rOocm
xml data :
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
<photos page="1" pages="2268" perpage="250" total="566985">
  <photo id="9848678063" owner="129596866@N08" secret="8c0c1a7dba" server="3687" f
arm="4" title="Airbus 380" ispublic="1" isfriend="0" isfamily="0" />
```

그림 11. flickr.photos.geo.getLocation 메쏘드의 'photo id' 파라미터에 대한 출력 결과
Fig. 11. Output result of 'photo id' parameter of flickr.photos.geo.getLocation method

```
<photo id="9848678063">
  <location latitude="50.050374" longitude="8.572028" accuracy="16" context="0" p
lace_id="_zI89TFWUrN_Hk0" woeid="709411">
    <neighbourhood place_id="_zI89TFWUrN_Hk0" woeid="709411">Frankfurt Main</neigh
bourhood>
    <locality place_id="Z0FRRfRXV7oIeew" woeid="650272">Frankfurt</locality>
    <county place_id="nLiTmf5QUL.SGmzwgQ" woeid="12597087">Frankfurt</county>
    <region place_id="Y1pL8yxTUb7rOocm" woeid="2345485">Hesse</region>
    <country place_id="h7eZVDlTUb50Btij9Q" woeid="23424829">Germany</country>
  </location>
```

그림 12. 두 메쏘드의 매쉬업이 가능한 경우
Fig. 12. Case that can mash-up of the two methods

리즘에서 'photo_id'를 이용하여 합성하므로 XML 결과에서 파싱 과정을 거쳐 'photo_id'에 해당하는 데이터 값 '9848678063'을 저장한다. 그리고 저장한 데이터 값을 이용하여 그림 12와 같이 URI를 생성하여 요청하고 XML 데이터 포맷으로 결과를 얻을 수 있다.

따라서 flickr.photos.search 메쏘드와 flickr.photos.geo.getLocation 메쏘드를 'photo_id' 파라미터를 이용하여 매쉬업하는 것은 특정 지역에 등록되어 있는 사진들의 정확한 위치 정보를 얻을 수 없었던 문제점을 해결하고 매쉬업 서비스를 사용자에게 제공할 수 있다.

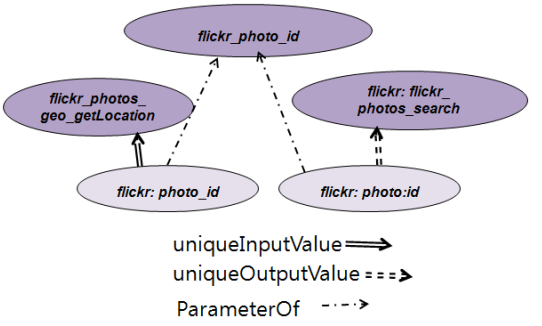

그림 13. 'photo id' 파라미터의 연관 관계
Fig. 13. Association of 'photo id' parameter

Ⅲ. 결 론

본 논문에서는 REST 기반의 Open API를 이용하여 자동 매쉬업 서비스를 제공할 때 고려해야하는 입출력 파라미터의 특성에 대해 제시하고 이를 적용한 온톨로지의 모델링 및 자동 매쉬업 방법에 대해 기술하였다.

입출력 파라미터의 특성을 적용한 온톨로지 모델링은 선행 연구에서 제시한 온톨로지를 확장 및 수정을 통해 재사용하였다. 선행 연구에서 제시한 온톨로지에서는 입출력 파라미터 특성을 고려하지 못하였기 때문에 입출력 파라미터를 온톨로지로 명시할 때와 데이터의 전달이 수행될 때 각각 문제점이 발생하였다. 이와 같은 문제점을 해결하기 위해 네임스페이스를 정의하여 온톨로지에서 발생하는 이름 충돌 문제를 해결하였고 새로운 프로퍼티를 정의하여 데이터의 유효 범위를 갖는 파라미터를 구분 할 수 있도록 설계하였다. 또한 사용자에게 자동 매쉬업 서비스를 제공하기 위해서 기존에 제공되는 매쉬업 서비스의 구조를 분석하고 입출력 파라미터의 데이터가 전달되는 구조를 두 가지로 정의하였다. 이를 기반으로 제안한 '매쉬업 가능한 매쏘드 탐색 알고리즘'을 통해 실제 사용되는 REST 기반의 API를 이용하여 합성된 서비스의 제공이 가능하다는 것을 구현 결과를 통해 확인할 수 있었다.

향후에는 REST 기반의 Open API 뿐만 아니라 SOAP 기반 및 XML-RPC 형식을 따르는 Open API까지 확장하여 적용하는 연구가 필요하다. 또한 본 논문에서 가정한 파라미터 이름을 이용한 구문적, 의미적 유사도 비교 과정의 자동화 방안에 대한 연구가 필요하다.

References

[1] J. Musser and Tim O'Reilly, *Web 2.0 Principles and Best Practices*, CA: O'Reilly Media., pp. 5-54, 2006.

[2] R. Yee, *Pro Web 2.0 Mashups: Remixing Data and Web Services*, Apress, pp. 3-20, 2008.

[3] C. C. Tsai, C. J. Lee, and S. M. Tang, "The Web 2.0 Movement: MashUps Driven and Web Services," *WSEAS Trans. COMPUTERS*,

vol. 8, no. 8, pp. 646-651, Aug. 2009.

[4] ProgrammableWeb, *ProgrammableWeb-APIs, Mashups and the Web as Platform(2014)*, Retrieved May, 21, 2014, from http://www.programmableweb.com./

[5] Y. J. Lee, "Resource Matchmaking for RESTful Web Services," *J. KIIT*, vol. 11, no. 8, pp. 135-143, Aug. 2013.

[6] S. I. Kim and H. W. Kim, "API selection and automatic open API composition method based on REST protocol," *J. KICS,* vol. 38C, no. 07, pp. 587-594, 2013.

[7] W. Jung, S. I. Kim, and H. S. Kim, "Ontology Modeling for REST Open APIs and Web Services Mash-up Method," in *Proc. Int. Conf. Inf. Netw.(ICOIN 2013)*, pp. 523-528, 2013.

[8] W. Jung, S. I. Kim, and H. S. Kim, "REST-based open API ontology modeling for automatic mash-up," in *Proc. KICS Winter Conf.*, pp. 832-833, 2013.

[9] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertion, Dept. Inform. Comput. Sci., Univ. of California, IRVINE, U.S.A., 2000.

[10] L. Richardson and S. Ruby, *RESTful Web Service*, 1st Ed., CA: O'Reilly Media, May 2007.

[11] Y. M. Park, A. K. Moon, H. K. Yoo, Y. C. Jung, and S. K. Kim, "SOAP-based Web Services vs. RESTful Web Services," *Electron. Telecommun. Trends,* vol. 25, no. 2, pp. 112-120, Apr. 2010.

[12] D. Gourley, B. Totty, M. Sayer, A. Aggarwal, and S. Reddy, *HTTP: The Definitive Guide*, 1st Ed., CA: O'Reily Media, pp. 3-42, Sept. 2002.

[13] flickr, *Flickr Services(2014)*, Retrived March, 15, 2014, from http://www.flickr.com/services/api/

[14] FOURSQUARE, *foursquare for Developers (2014)*, Retrived May, 21, 2014, from https://developer.foursquare.com/

[15] W. Jung and H. S. Kim, "Ontology modeling for solving the problems of automatic mash up service using the similarity between the query parameter name" in *Proc. KICS Fall Conf.*, pp. 516-517, 2014.

[16] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist, Second Edition: Effective Modeling in RDFS and OWL*, 2nd Ed., MA: Morgan Kaufmann, pp. 13-25, 2011.

[17] uGuX.com, Amazon and eBay Comparison Shopping, Retrived May, 21, 2014, from http://ugux.com/shopping/

[18] AuctionSearchKit., *Auction Search Kit(2014)*, Retrived May, 21, 2014, from http://www.auctionsearchkit.co.uk/

[19] The Apache Software Foundation, Apache Jena - Home, Retrived May, 21, 2014, from http://jena.apache.org/

정 완 (Jung Wan)

2012년 2월 : 광운대학교 전자통신공학과 졸업

2012년 3월~현재 : 광운대학교 전자통신공학과 석사과정

<관심분야>시맨틱 웹, 데이터 마이닝, RESTful 웹 서비스

김 화 성 (Kim Hwa Sung)

1981년 2월 : 고려대학교 전자공학과 졸업

1983년 2월 : 고려대학교 전자공학과(석사)

1996년 : Lehigh Univ. 전산학(박사)

1984년 3월~2000년 2월 : ETRI 책임 연구원

2000년 3월~현재 : 광운대학교 전자통신공학과 교수

<관심분야> Wireless Internet, NGN 미들웨어 환경, Streaming service