디지털 도서관을 위한 시맨틱 웹 서비스

김운[O] 김지현*, 김여정 강지훈
충남대학교 컴퓨터과학과
{ wkim[O], innias, jhkang}@cs.cnu.ac.kr, jhkim*@kopec.co.kr

Semantic Web Service for Digital Library System

Yun Jin[O], Ji-Hyeon Kim, Yeo-Jung Kim, Ji-Hoon Kang
Chungnam National University

요 약

현재 디지털 도서관 시스템에서 제공하는 정보검색 서비스는 **HTML, HTTP**등에 근간을 둔 구조적인 방법이지만, 시맨틱 웹 환경에서의 디지털 도서관 시스템은 보다 지능적이고, 상호 응용성이 뛰어난 시맨틱 웹 서비스에 근간을 둔 방법이 필요하다. 대부분의 시맨틱 웹 서비스 시스템에서는 사용자의 요구와 일대일로 매핑되는 서비스가 있다고 가정하지만 본 논문에서는 이런 경우가 매우 드물다고 가정한다. 이런 가정 하에 본 논문에서는 에이전트가 시맨틱 웹 서비스를 효율적으로 할 수 있는 새로운 **planning**방법과 **execution**방법에 대하여 제안하고자 한다. 본 논문에서 제안하는 시스템은 사용자가 시맨틱한 질의를 할 수 있고, 에이전트는 그 의미를 파악하여 가능한 모든 경우를 도입하여 planning을 하게 된다. 본 논문에서 제안하는 방법과 시스템은 새로운 분야에서의 시맨틱 웹 서비스모델로도 적용 가능하다.

1. 서론

디지털 도서관 시스템에서 자동적으로 웹 서비스를 하려면, 기계가 웹 서비스를 자동적으로 발견하고, 그런 웹 서비스들을 **planning**하고, 자동적으로 실행하여야 한다. 이렇게 하기 위해서는 기계가 이해할 수 있는 언어가 필요한데, 시맨틱 웹(**Semantic Web**)[1]의 탄생은 바로 이런 요구를 만족하였으며, 웹 서비스 환경을 풍성하게 하였다. 시맨틱 웹의 장점은 온톨로지를 사용하여 그 의미를 표현하는 것이다. 시맨틱 웹 서비스에서는 **DAML-S[2], OWL-S[3]**와 같은 언어를 이용하여 웹 서비스를 기술함으로써, 에이전트는 추론을 통하여 서비스를 발견하고, **planning**할 수 있다. 본 시스템에서는 **OWL-S**언어를 이용하여 **MPEG-7 IR, Dublin-Core & Text IR, Genre IR** 등 서비스 자원을 기술하였다.

웹 서비스를 **DAML-S**와 **OWL-S**로 기술하였다고 해서 사용자에게 원하는 서비스를 제공해줄 수 있는 것은 아니다. 사용자는 반드시 모호성이 적고, 구체적인 질의를 에이전트에게 제공해야만, 에이전트는 적은 노력과 시간을 들여 원하는 서비스를 제공할 수 있다. 본 시스템에서 사용자는 질의 언어로 **DQL[4]**를 사용하며, **DQL**에서 사용될 온톨로지는 시스템에서 제공하게 된다. 에이전트는 이렇게 작성된 **DQL** 문서로부터 **triple**을 추출하여 의미적으로 적합한 서비스를 찾는다. 또한 찾은 서비스는 **planning**을 통하여 실행 순서를 결정하게 되며, **planning**된 결과는 **executor**를 통해 실제 서비스를 실행하게 된다.

본 시스템에서는 에이전트의 효율성을 제고하기 위하여 **planning**함에 있어서, 정확히 매핑되는 서비스 뿐만 아니라 사용자 질의에 의미적으로 포함되는 서비스까지 고려하였으며, 실행 시 에러를 대비하여 **if-then-else, choice**등 프로그래밍 제어 구성자들을 **planning**에 포함 시켰다.

본 논문의 구성은 다음과 같다. 2에서는 관련연구를 소개하고, 3에서는 전체 구성 및 역할에 대해 살펴보며, 4에서는 에이전트 설계 및 구현에 대하여 살펴본다. 마지막 5에서는 결론 및 향후 연구에 대해 소개한다.

2. 관련연구

현재 웹 서비스에 대하여 많은 연구들을 하고 있으며, 대체로 두 개 큰 분류로 나눌 수 있다[5]. 하나는 기업 또는 비즈니스 월드 등에서 제안하고 개발한 **XML**기반의 웹 서비스 프로세스 모델링과 실행언어로써, **BPEL4WS[6]**를 대표적 예로 들 수 있다. 그러나 이런 방법들은 기본적으로 모두 구문적이며 시맨틱을 위한 처리방법을 고려하지 않는다. 다른 하나는, **W3C**의 시맨틱 웹 위원회를 중심으로 웹 서비스에 대한 온톨로지로 정의하고 이를 **Planning**을 통하여 추론하는 방법에 초점을 맞추고 있다. **DAML-S**와 **OWL-S**는 바로 이런 목적을 위해 고안된 웹 서비스 기술 언어이다. 이

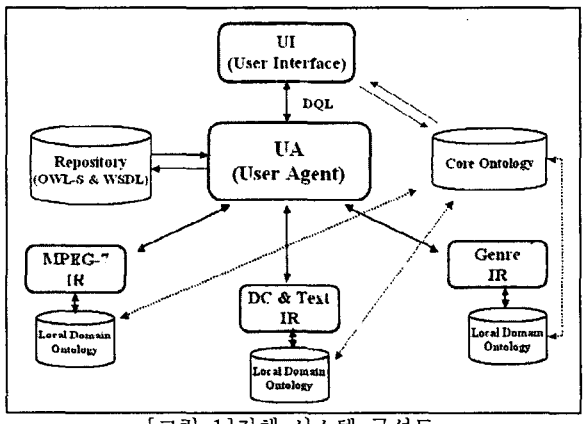
* 이 연구는 BK21 충남대학교 정보통신 인력 양성 사업단과 소프트웨어 연구 센터의 지원을 받았음.

런 표준적 언어를 이용하여 웹 서비스를 기술하면, 에이전트는 자동적으로 웹 서비스에 대하여 추론하여 발견할 수 있고, 서비스 의존성에 근거하여 실행순서를 결정할 수도 있어 지능적인 서비스를 할 수 있다는 장점이 있다. 본 논문에서 제안하고자 하는 방법은 **OWL-S**를 이용하여 디지털 도서관에서 제공하는 웹 서비스를 기술하였으며, **planning**을 통하여 웹 서비스를 발견하고, **composition**한다.

웹 서비스 **planning** 방법들이 여러 가지로 제안되었다. [7]에서는 웹 서비스를 **planning**함에 있어서 **output** 데이터 타입에만 의존하는 방법을 택하고 있다. 그러나 본 시스템에서의 **planning**방법에서는 서비스 **matching**을 **output**뿐만 아니라 **input, precondition, effect**까지도 고려한다. [8]에서는 각 서비스를 **DAML-S**로 기술하고, 이런 서비스들을 **planning operator**로 변환하여 **JESS**[9]와 같은 전문가 시스템의 **Rule**로 사용한다. [10]에서도 [8]과 비슷한 방법을 택하고 있다. 이런 논문들의 기본 가정은 원하는 서비스가 정확히 매핑될 때의 경우를 고려하여 만든 시스템들이다. 그러나 [11]에서는 에이전트가 서비스 **match**함에 있어서 인터넷 환경에서 일대일 매핑되는 서비스를 찾기란 아주 어렵다고 주장한다. 본 시스템에서는 이런 점을 감안하여 일대일 정확히 매핑되는 서비스를 찾는 경우 외에도 의미적으로 포함되는 경우도 후보 서비스로 고려하였다.

3. 전체 시스템 구성 및 역할

전체 시스템은 [그림 1]과 같이 **UI(User Interface), UA(User Agent), Repository,** 및 웹 서비스 자원들 **(MPEG-7, Dublin Core & Text IR, Genre IR)** 등 **4** 개 부분으로 구성되었다. 각 부분의 역할을 살펴보면 다음과 같다.



[그림 1]전체 시스템 구성도

UI는 사용자로부터 질의를 받아들이는 부분으로써, 사용자는 시맨틱한 질의로 시스템에서 제공하는 온톨로지를 이용할 수 있고, 일반 용어만 입력하는 키워드 기반의 검색과 비슷한 방식으로 질의를 할 수도 있다. 예를 들어, 한 사용자가 시맨틱 질의방식으로 화가 **"Gogh"**가 그린 그림을 찾고 싶다면, 입력으로 시스템에서 제공하는 클래스 중 **painter**를 선택하고 또한 시스템에서 제공하는 속성**(Property)**중 **name**을 선택한 다음 값으로 **'Gogh'**를 입력하고, 출력으로 **picture**과 그 속성 **Genre**도 선택한다. 키워드 기반 질의 방식과 비슷한 방법은 용어만 입력하면 시스템이 그 용어가 가질 수 있는 의미를 체크하고, 만약 이 용어가 여러 의미가 있으면 사용자에게 문의하는 방법을 취한다.

UA는 **UI**로부터 **DQL**문서의 질의를 받아 이 문서를 파싱하여 적합한 서비스를 **Repository**에 요청한다. **UA**는 크게 **Planner**와 **executor**로 구성 되었는데 **Planner**는 **UI**의 **DQL**를 파싱하여 **triple** 집합을 추출하고, 이렇게 추출된 **triple**집합을 이용하여 **Repository**로부터 서비스를 얻는다. 이렇게 얻은 서비스는 실행 순위에 따라 트리를 형성하게 된다. **Executor**는 **Planning** 트리에 근거하여 실제 서비스를 실행하게 되는데 이 때 트리의 각 노드에는 웹 서비스를 외에도, 프로그래밍 제어 구성자와 상태 표시자가 포함되어 있다. 이런 추가 정보는 현재의 실행상황을 파악하기 위함이다. **Executor**가 실제 서비스와 연결할 때 어떤 방식으로 연결할지를 결정하기 위해 해당 서비스의 **WSDL**문서를 참조하게 되는데 이 문서 역시 **Repository**로부터 가져오게 된다.

Repository는 각 서비스를 기술한 **OWL-S**문서와 각 서비스의 **WSDL**문서, 그 밖에 온톨로지 등을 저장하게 된다.

마지막으로, 검색 서비스 자원이 되는데, 이런 검색 서비스 자원은 실질적으로 특정한 서비스를 제공하게 된다. 예를 들면, **MPEG-7 IR**은 이미지, 동영상 등 **MPEG-7** 형식으로 된 문서에 관한 서비스를 하게 된다. 그러나, 기존의 키워드 검색 방법이 아니라 키워드 대신 의미의 단위인 **triple**를 이용하여 검색하는 방법을 택하고 있다.

4. 시스템 에이전트 설계 및 구현

시맨틱 웹 서비스를 함에 있어서 에이전트의 노력, 시간 등을 줄이는 것은 매우 중요하다. 여기서 말하는 노력은 에이전트가 질의 의미와 매핑되는 웹 서비스들을 어떻게 발견하고 또한 **planning**하는가 하는 것을 말하며, 시간은 에이전트가 노력에 소요한 시간과 실질적인 서비스를 실행하는데 걸린 시간의 합을 말한다. 아무리 많은 노력을 들여 사용자가 원하는 서비스를 찾았다고 할지라도 많은 시간이 걸린다면 아무런 의미가 없으며, 시간적으로는 적절하게 적은 노력을 들인다고 할지라도 사용자가 원하는 서비스를 제공해 주지 못하거나 또는 제공한다 하더라도 서비스가 실행되다가 에러가 발생하거나 하는 등 생각하지 않던 일이 발생한다면 이 역시 의미 없는 웹 서비스가 될 것이다.

4.1 Planner

Planner는 **executor**의 전단계로써, **executor**가 사용자의 요구에 맞는 서비스를 해줄 수 있도록 프로세스 모델을 형성해 준다. 예를 들어, 다음과 같은 **triple** 집합이

있다고 하자.

```
(?V1, Painter 'Gogh')
(?V1, Genre, ?V2)
```

위의 예제에서 사용자는 **Picture(?V1)**와 **Picture**의 **Genre(?V2)**을 검색하려고 한다. 그런데 **V1**의 **Painter**를 **'Gogh'**를 알고 있으므로 이를 입력으로 하는 서비스를 찾아야 한다. 대부분의 **planner**는 **Painter**를 입력으로, **Picture**와 **Genre**를 출력으로 하는 서비스를 찾으며, 기본적으로 이런 서비스가 있다고 가정한다. 그러나 본 시스템에서는 이렇게 정확히 일치한 경우가 드물다는 가정 하에 **Painter**를 입력으로 받는 서비스를 찾은 다음 그 들을 시작으로 출력 **Picture**와 **Genre**를 목표(**Goal**)로 정하고 이것이 얻을 수 있을 때까지 **planning** 트리를 형성한다. 이렇게 형성된 트리가 바로 **Planning** 결과이며, 트리를 방문하는 순서가 서비스 실행순서가 된다. [그림 2]와 [그림 3]은 실제 위의 예를 이용하여 **planning**한 결과이다.

```
RS : Painter / Picture, Genre

PS1 : Painter, Nationality / Picture
PS2 : Painter, Picture / Genre
PS3 : Painter / Nationality
```

[그림 2] 서비스 요청과 검색된 서비스들

[그림 2]에서 **RS**는 사용자가 요청한 서비스이다. 이때 **Repository**에 요청을 하면, **Painter**를 입력으로 하는 서비스 후보 **PS**가 3개 된다. 그러나, **PS1**과 **PS2**는 현재 알 수 없는 **Nationality**와 **Picture**를 입력으로 하기 때문에 제일 처음 실행될 수 없다. 다만 **PS3**만이 현재 사용자로부터 받은 정보를 이용하여 실행 가능하다. 그러나, **PS3**는 **Nationaliry**를 출력으로 하기 때문에 **PS1**이 실행 가능하며, **PS1**의 실행 결과를 이용하여 **PS2**가 실행 가능하다. 즉 최종 목표에 도달하여 사용자의 요구를 만족하게 된다. [그림 3]은 서비스를 찾는 과정과 그것을 이용하여 순서를 정한 결과이다.

```
---------- selected PSS : 0 name : ps3
---------- selected PSS : 1 name : ps1
---------- selected PSS : 2 name : ps3
---------- selected PSS : 3 name : ps1
---------- selected PSS : 4 name : ps2
---------- selected PSSName : 0 name : ps3
---------- selected PSSName : 1 name : ps1
---------- selected PSSName : 2 name : ps2
```

[그림 3] **Planning**한 결과

4.2 Executor

Executor는 웹 서비스 제공자가 제공하는 서비스와 연결하여 실질적인 서비스 결과를 받아오는 부분이다. 실제 웹 서비스와 연결하여 서비스 받기 위하여 해당 서비스의 **WSDL**[12]문서가 필요하다. 이런 **WSDL** 문서 역시 **repository**에 저장되어 있으며, **planning**결과와 함께 이 문서를 받게 된다. **Executor**는 이 **WSDL**문서를 파싱하여 실제 웹 서비스의 인터넷 URL, 통신 프로토콜, **input**파라메터 개수, 및 타입 등을 추출하게 되며 이렇게 추출된 정보를 기반으로 웹 서비스와 통신을 하게 된다.

5. 결론

본 논문에서는 디지털 도서관 시스템을 위한 시맨틱 웹 서비스에 대하여 설계하고, 개발 방법론에 대하여 제안하였다. 본 시스템에서는 시맨틱한 웹 서비스를 찾기 위하여 정확히 **match**되는 경우가 드물기 때문에 질의를 온톨로지 계층구조를 이용하여 일반화된 질의를 만들었다. 이렇게 일반화된 질의는 해당되는 웹 서비스를 쉽게 찾을 수 있다는 장점을 갖고 있다. 향후 연구로는 이 방법론에 근거하여 개발 중인 본 시스템을 성공시켜, 타 분야에 적용 가능한 모델로 사용 가능하도록 한다.

5. 참고문헌

[1] W3C Semantic Web. http://www.w3.org/2001/sw.

[2] DAML-S, http://www.daml.org/services/daml-s/0.9/.

[3] OWL-S http://www.daml.org/services/owl-s/1.0/

[4] DAML Query Language(DQL) http://www.daml.org/dql/

[5] Biplav Srivastava, and Jana Koehler, "Web Service Composition - Current Solutions and Open Problems," ICAPS 2003 workshop on Planning for Web Services, 10 June 2003, Trento, Italy.

[6] Business Process Execution Language for Web service (BPEL4WS), http://www-106.ibm.com/developerworks/library/ws-bpel/

[7]Mark Carman, Luciano Serafini, Paolo Traverso, "Web Service Composition as Planning," ICAPS 2003, Workshop on planning for Web Services, 10 June 2003, Trento, Italy.

[8] Sheshagiri, M., desJardins, M., Finin, T., "A Planner for composing Service Described in DAML-S," ICAPS 2003, Workshop on planning for Web Services, 10 June 2003, Trento, Italy.

[9] Java Expert System Shell (JESS), http://web.njit.edu/all_topics/Prog_Lang_Docs/html/jess51/

[10]Fikes, R., Nilsson, N., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, 2(3/4), 1971.

[11] Sycara K., Widoff S., Klusch M., Lu J. G., "LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace," Autonomous Agents and Multi-Agent Systems, 5, PP173-203, 2002.

[12] Web Services Description Language(WSDL), http://www.w3.org/TR/wsdl.