의미를 제공하는 웹 서비스 조합 도구의 설계[1]

윤혜정*, 김명화**, 이민수*
*이화여자대학교 과학기술 대학원 컴퓨터학과
e-mail:auroree@ewhain.net, munga@ewhain.net,
mlee@ewha.ac.kr

Design of a Semantic Web Service Composition Tool

Hyejung Yun*, Myunghwa Kim**, Minsoo Lee*
*Dept. of Computer Science & Engineering, Ewha Institute of Science and Technology, Ewha Womans University

요 약

워크플로우는 컴퓨터에 의해서 자동으로 실행되고 관리되는 업무 프로세스이다. 워크플로우 기술은 기업 환경의 변화와 함께, 비즈니스 프로세스 관리(BPM:Business Process Management)의 개념과 함께 발전해 왔다. 또한비즈니스를 수행하기 원하는 기업의 거래파트너간, 고객, 공급자들의 분리된 비즈니스 환경을 제거하고, 향상된 e-비즈니스를 수행하기 위한 통합 환경을 제공해주는 웹 서비스는 워크플로우를 구성하는 매우 중요한 개념이 되었고, 이러한 웹 서비스의 흐름을 정의하는 워크플로우 설계 언어가 많이 등장하였다. 그 중 가장 표준이 될 가능성이 유력한 BPEL4WS는 비즈니스 프로세스에 필요한 트랜잭션, 보상 등의 개념을 포함하여, 효율적으로 비즈니스 프로스세스를 구성할 수 있게 해 준다. 그러나 BPEL4WS는 1)고정된 WSDL파일을 참조하여, 그 흐름을 정적으로만 구성할 수 있다는 단점이 있고, 2)제한된 자원만을 갖고 있기 때문에, 의미를 부여할 수 없어 대체 서비스를 찾을 수 없다는 단점이 있다. 따라서 본 논문에서는 이러한 단점을 해결하기 위하여 기존의 WSDL에 의미(Semantic)을 부여하는 방법으로 이를 해결하고자 한다.

1. 서론

오늘날 데이터 중심의 정보기술은 프로세스 중심의 정보기술로 빠르게 변하고 있다. 프로세스 중심의 정보기술의 핵심은 바로 비즈니스 프로세스(Business Process)이며, 워크플로우는 비즈니스 프로세스 처리를 자동으로 할 수 있도록 한다.

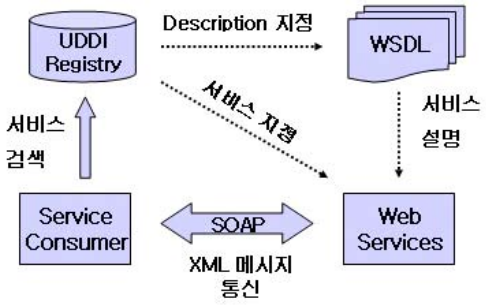
'웹서비스'라는 용어는 서로 호환되지 않는 프로그램을 인터넷 프로토콜을 이용해 연결되도록 하는 일련의 소프트웨어 명세다. 비즈니스 프로세스로 사용되기 위해 여러 개의 웹 서비스가 함께 작용할 필요가 있는데, 이를 위해 흐름을 조정하기 위한 언어가 필요로 하게 되었다. 그래서 워크플로우 표준 단체는 WSCI, WSFL, XLANG, BPEL4WS와 같은 많은 워크플로우 표준들을 내놓았고, 아직 표준이 정해지지는 않은 상태이다. 그러나 BPEL4WS (Business Process Execution Language for Web Services)가 OASIS에서 채택된 상태이므로 표준이 될 가능성이 높다. 그러나 BPEL4WS는 1)고정된 WSDL파일을 참조하여, 그 흐름을 정적으로만 구성할 수 있다는 단점이 있고, 2) 제한된 자원만을 갖고 있기 때문에, 의미를 부여할 수 없어 대체 서비스를 찾을 수 없다는 단점이 있다. 따라서 본 논문에서는 이러한 단점을 해결하기 위하여 기존의 WSDL에 의미(Semantic)을 부여하는 방법으로 이를 해결한다.

2. 관련연구

2.1. 웹서비스

웹 서비스는 크게 보았을 때 유, 무선 상에서 플랫폼과, 구현언어에 독립적인 컴포넌트 기반의 분산 컴퓨팅 서비스이다. 웹서비스의 특징으로는 플랫폼과 디

바이스 및 위치 독립적이라는 것과 동적인 기능(dynamic function), 기존(legacy) 시스템이 적용이 가능하다는 것이다. 그림 1은 웹 서비스 이용 과정을 보여주고 있다[1].



[그림 1] 웹 서비스 이용 과정

2.2. BPEL4WS

BPEL은 웹 서비스에서 비즈니스 프로세스를 사용 가능케 하고 정의하는 데 쓰이는 언어다[2]. BPEL 비즈니스 프로세스는 웹 서비스를 이용해 BPEL을 기반으로 만들어지거나 아니면 서비스를 상호 교환하게 해준다. BPEL은 IBM 의 WSFL(Web Services Flow Language)과 마이크로소프트의 XLANG을 기반으로 만든 것이다[3]. 그리고 WSFL과 XLANG은 둘 다 WfMC.org와 BPML.org의 스펙에 영향을 받았다. BPEL4WS는 WSDL[4], XML Schema[5], XPath[6]에 의존되어 있다[7]. 이들 중 WSDL가 BPEL4WS에 가장 많은 영향을 주고, BPEL4WS는 WSDL이 동작하는 위에서 추가 된다.

2.3. 시멘틱 웹

서비스의 합성에 있어 의미(Semantic)를 부여하기 위한 노력은 계속 진행되고 있다. 시맨틱 웹 서비스(Semantic Web Services)는 시맨틱 웹과 웹 서비스가 갖고 있는 특징을 모두 갖춘다. 시맨틱 웹 기반의 웹 서비스 명세는 시맨틱 마크업 언어를 사용해 컨텐츠를 표현하여 서비스 합성(Composition)을 자동적으로 처리할 수 있게 한다.

2.3.1 온톨로지

서비스 요청자와 제공자는 올바르게 해석된 서비스 및 자원을 이용할 수 있어야 한다. 이를 위해 해당 서비스 및 자원이 의미상 분류를 위한 온톨로지를 가져야 한다. 온톨로지는 시맨틱 웹이 가능하게 하는 핵심 기술이자, 교환되는 데이터에 대한 개념 및 의미를 정의하는 수단을 제공한다. 온톨로지는 상징(Symbol)에 대한 인간의 이해와 그 이해를 기계가 처리 가능하게 하는 것을 연결시켜준다.

2.3.2 시맨틱 웹 서비스 기능 요구 사항

- 자동 웹 서비스 발견 기능(Automatic Web Service Discovery)
- 자동 웹 서비스 호출(Automatic Web Service Invocation)
- 자동 웹 서비스 조합 및 상호 운영 기능(Automatic Web Service Composition and Inter-operation)
- 자동 웹 서비스 실행 감시와 복구 기능(Automatic Web Service Execution Monitoring and Recovery)

2.3.3. DAML-S

시멘틱 웹 서비스의 출발점은DAML(DARPA Agent Markup Language)에서 파생된 DAML-S라는 웹 서비스의 역량과 성격을 명료하고 컴퓨터 인식 가능한 형태로 기술할 수 있는 상위 온톨로지(ontology)를 제공하기 위한 언어라 할 수 있다. DAML-S는 사용자가 웹 서비스를 자동으로 위치(loaction), 선택(selection), 합성(composition), 감독(monitoring)할 수 있게 하는 적절한 온톨로지를 제공하는 것이다. DAML-S의 구성 요소는 다음과 같다.

- Service Profile: 웹 서비스의 능력을 표현하고, 웹 서비스 설명을 위한 추가적인 특징 을 기술한다.
- ServiceModel: 웹 서비스의 기능을 묘사하며, 웹 서비스의 형태와 실행순서를 표현하는 Process Model과 이를 모니터링하기위한 Process Control Model로 구성된다.
- ServiceGrounding: 서비스의 실체화(실제 WSDL과의 매핑)을 담당하며, Process Model에서 표현된 웹 서비스와 WSDL로 표현된 웹 서비스간의 관계를 기술한다.

3. 의미 기반 웹 서비스 워크플로우 설계

3.1. BPEL4WS 설계파일

BPEL4WS는 현재 표준이 될 가능성이 가장 높은 웹 서비스 합성 언어이자, 높은 기능성(fucntionality)를 지녀, 웹 서비스 프로세스 효율적으로 합성하기 위한 언어이다. 그러나 자동화되고 지능적인 시맨틱 웹 서비스의 합성을 위한 기술로는 많이 부족하다. 이를

위해 기존의 BPEL4WS설계 도구에서 효율적인 방법으로 시멘틱을 지원하고자 한다. 우선 BPEL4WS 설계에 있어 필요한 파일이다.

- .bpel : 웹 서비스 흐름 정의 파일
- .wsdl : 흐름 정의 파일에 대한 설명 파일과 각 웹 서비스의 설명파일이며 1개 이상
- .project , bpel.xml, build.xml: 설계한 파일의 프로젝트 실행 설명 파일

이들중 주요한 파일은 bpel과 wsdl이다. 본 논문에서는 이 WSDL에 의미를 부여하여, BPEL4WS문서로 DAML-S 의 기능을 만족하게 한다.

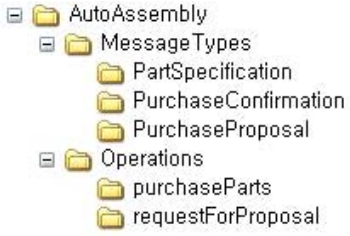
3.2. 온톨로지 설계

본 절에서는 자동차 부품을 사는 프로세스를 예제로 비즈니스 흐름을 정의한다고 가정한다. 자동차 부품을 사는 WSDL문서는 다음과 같다.

```
<wsdl:message name="OperationRequest">
<wsdl:part name="in0" type="auto1:PartDetails"/>
</wsdl:message>
<wsdl:message name="OperationResponse1">
<wsdl:part name="return"
        type="auto1:PurchaseInvoice"/>
</wsdl:message>
<wsdl:message name="OperationResponse2">
<wsdl:part name="return"
        type="auto1:Confirmation"/>
</wsdl:message>
<wsdl:portType name="AutoAssembly">
<wsdl:operation name="requestForInvoice"
        parameterOrder="in0" >
<wsdl:input message="inaa:OperationRequest"
        name="InvoiceRequest"/>
<wsdl:output message="inaa:OperationResponse1"
        name="InvoiceResponse"/>
<wsdl:operation name="purchaseOfPart"
        parameterOrder="in0" ">
<wsdl:input message="inaa:OperationRequest"
        name="PurchaseRequest"/>
<wsdl:output message="inaa:OperationResponse2"
        name="PurchaseResponse"/>
</wsdl:operation>
</wsdl:portType>
```

[표1] WSDL파일

이 WSDL문서를 그림 2의 자동차 부품 프로세스 온톨로지와 비교해보자. purchaseOfPart와 request ForInvoice연산은 purchaseParts의 온톨로지와 requestForProposal 온톨로지로 각각 매핑 될 수 있고. 메시지인 PartDetails타입과 Confirmation타입, PurchaseInvoice타입은 각각 PartSpecification 온톨로지와 PurchaseConfirmation 온톨로지, Purchase Proposal온톨로지로 매핑 될 수 있다.



[그림 2] 자동차 부품 프로세스 온톨로지

3.3. WSDL에 의미부여 및 WSDL과 연동

3.3.1. Grounding

그라운딩(Grounding)은 통신 프로토콜, 서비스에 접근하기 위해 사용되는 포트와 관련된 정보를 구체화하기 위해 사용한다. 그라운딩 클래스는 서비스 명세에 포함된 앨리먼트를 정의하기 위한 과정으로 WSDL과 연결될 수 있다. DAML-S와 독립적으로 개발된 WSDL은 추상적(abstract) 사양과 구체적인 연산이 분리된 점을 활용해 서비스 인터페이스 정의와 서비스 구현을 분리하여 반영한다. 두 언어는 다음과 같이 매치가 가능하다.

- DAML-S의 Automic process 는 WSDL의 연산(operation)과 일치한다.
- DAML-S의 Automic process의 입력(input)과 출력(output)집합은 WSDL의 메시지(message)의 개념과 일치한다.

3.3.2. WSDL의 확장

본 논문에서는 기존의 Grounding과 유사한 방법으로 기존 WSDL을 확장하는 방식을 택하였다. 이 방법은 시멘틱 웹에서 필요한 메시지와 연산에 필요한 의미를 각 메시지와 연산에 추가하여, BPEL4WS 설계 문서 전체를 DAML-S로 변환하지 않고도 의미(Semantic)을 제공한다는 데에 의의가 있다.

3.3.2.1 메시지(Message) 부분의 온톨로지 매핑

입력(input)과 출력(output) 인 메시지(Message) 부분은 XML 스키마에 의해 정의된 부분이다. 각 메시지를 DAML-S의 기술 클래스인 DAML-OIL과 매핑시킨다. 표 2에서 TravelDetails타입과 Confirmation

타입은 TicketInformation온톨로지와 Confirmation Message온톨로지로 매핑되었다.

3.3.3.2 선조건과 이펙트 태그 삽입

시맨틱 웹에서 각 연산은 많은 선조건(precondition)과 이펙트(effect)를 가진다. 논문에서는 선조건과 이펙트를 별도의 태그로 연산의 자식 노드로 추가하였다. 표 2에서 볼 수 있듯이 requestForInvoice에서 선조건으로 Valid Customer가 추가되었고 InvoiceSent를 이펙트로 지정하였다.

```
<wsdl:definitions
targetNameSpace="auto.org"xmlns="http://schemas.xmlsoap.org/wsdl
...
xmlns:OTOnt=http://dwlab.ewha.ac.kr/project/SWS/AutoAssemblyOntology.daml
xmlns:OTExt=http://dwlab.ewha.ac.kr/project/SWS/WSDLExtension"
...
<wsdl:message name="OperationRequest">
<wsdl:part name="in0" type="auto1:PartDetails"
OTExt:onto-concept="OTOnt:PartSpecification"/>
</wsdl:message>
<wsdl:message name="OperationResponse1">
<wsdl:part name="return" type="auto1:
PurchaseInvoice"
OTExt:onto-concept="OTOnt:PurchaseProposal"/>
</wsdl:message>
<wsdl:message name="OperationResponse2">
<wsdl:part name="return" type="auto1: Confirmation "
OTExt:onto-concept="OTOnt:PurchaseConfirmation"/>
</wsdl:message>
<wsdl:portType name="AutoAssembly">
<wsdl:operation name="requestForInvoice"
parameterOrder="in0"
OTExt:operation-concept="OTOnt:requestForProposal
">
<wsdl:input message="inaa:OperationRequest"
name="InvoiceRequest"/>
<wsdl:output message="inaa:OperationResponse1"
name="InvoiceResponse"/>
<OTExt:precondition name="ValidCustomer"
OTExt:precondition-concept="OTOnt:ValidCreditCustomer"/>
<OTExt:effect name="InvoiceSent"
OTExt:effect-concept=
"OTOnt:InvoiceDelivered"/> </wsdl:operation>
<wsdl:operation name="purchaseOfPart"
parameterOrder="in0"
OTExt:operation-concept="OTOnt:purchaseParts">
<wsdl:input message="inaa:OperationRequest"
name="PurchaseRequest"/>
<wsdl:output message="inaa:OperationResponse2"
name="PurchaseResponse"/>
 <OTExt:precondition name="ValidCustomer"
OTExt:precondition-concept="OTOnt:ValidCreditCustomer"/>
```

[표 2] WSDL를 시맨틱 개념으로 확장한 WSDL

4. 결론

본 고에서는 웹 서비스 합성 기술인 BPEL4WS에서 고정된 WSDL파일을 참조하여 정적 바인딩만 가능하다는 문제점과 제한된 자원만을 가져 의미를 부여할 수 없다는 문제점을 WSDL에 의미(Semantic)을 부여하는 방법으로 해결하였다. 기존 DAML-S의 Grounding과 유사한 방식으로, 우선 온톨로지를 구성하여 메시지와 연산으로 구분하고 각각을 매핑한 후, 메시지에서는 입력(input)과 출력(output)메시지에 각각의 온톨로지를 매핑하였고, 연산도 온톨로지에 매핑할 뿐 아니라, 연산에 필요한 선조건(precondition)과 이펙트(effect)를 자식노드로 추가하고, 이를 온톨로지와 매핑하였다. 이로써, BPEL4WS는 의미를 부여하기 위해 모든 파일을 DAML-S로 변환하지 않고도 WSDL을 통해 이를 제공할 수 있다는 장점을 가진다.

참고문헌

[1] 오지훈, 서비스 도메인 온톨로지를 이용한 시멘틱 웹 서비스 합성, 원광대학교 대학원, 2003

[2] Oracle, BPEL 서버를 이용한 웹 서비스 통합, 오라클 기술 백서, 2004

[3] Mikko Laukkanen, Heikki Helin, composing Workflows of Semantic Web Services, Workshop on Web Services and Agent-based Engineering (AAMAS), 2003

[4] W3C, http://www.w3.org/TR/wsdl

[5] W3C, http://www.w3.org/XML/Schema

[6] A.Berglund, S. Boag, D.Chamberlin, M.F.Fernndez, M. Kay, J.Robie and J.Simon, "XML Path Language (XPath) 2.0", W3C Working Draft, Nov. 2003. http://www.w3.org/TR/xpath

[7] Kunal Verma, Rama Akkiraju, et, al. On Accommodating Inter Dependencies in Web Process Flow Composition, American Association for Artificial Intelligence (www.aaai.org), 2004