웹 서비스 코리오그래피의 상호운용성 지원을 위한 XML 언어 개발

이완[a], 정범석[b], 박종헌[c]

[a] *아이디스*
135-973, 서울시 강남구 삼성동 159-9 도심공항타워 13층
Tel:+ 82-2-3429-9968, E-mail:nymphy@idis.co.kr

[b]*서울대학교 산업공학과 석사과정*
151-742, 서울시 관악구 신림9동 공과대학 39동 314호
Tel: +82-2-882-0504, E-mail: bumdol03@snu.ac.kr

[c]*서울대학교 산업공학과 교수*
151-742, 서울시 관악구 신림9동 공과대학 39동 306호
Tel: +82-2-88-7174, E-mail: jonghun@snu.ac.kr

Abstract

웹 서비스가 현재 산업에서 주요한 분산 컴퓨팅 기술로 대두되면서, 그에 다른 연구가 활발히 이루어지고 있다. 특히, 비즈니스 프로세스 관점에서 BPEL4WS (Business Process Execution Language for Web Services)와 같은 오케스트레이션의 연구가 많이 이루어지고 있다. 또한, 웹 서비스의 주요 구성 요소 중에 하나인 코리오크래피에 대한 부분은 W3C (World Wide Web Consortium)에서 현재 후보 권고안 수준인 WS-CDL (Web Services Choreography Description Language) 관련 연구가 이루어지고 있다. 이런 가운데 DERI (Digital Enterprise Research Institute)에서 WSMX (Web Services Execution Environment) 기반의 시맨틱 웹 서비스 분야의 코리오 그래피 연구를 시작하였고, 코리오그래피 불일치 (Mismatch)를 개념적으로 해결하는 방안인 프로세스 미디에이션 (Process Mediation)의 연구가 있었다.

이러한 연구들을 바탕으로 서로 다른 WS-CDL 상에서 특정 참여자가 코리오그래피 미디에이션을 하기 위해서 개발된 XML 언어 WS-CML (Web Services Choreography Mediation Language)을 본 논문에서 제안한다.

Keywords:

웹 서비스, 코리오그래피, 미디에이션, XML, 상호호환성, 프로세스 미디에이션

Introduction

오늘날 분산 컴퓨팅 기술에 있어서 관심도가 높고 연구가 활발한 분야가 바로 웹 서비스이다 [30]. 웹 서비스는 기존의 기술에 비하여 서비스 기술 및 탐색, 실행이 W3C(World Wide Web Consortium), OASIS (Organization for the Advancement of Structured Information Standards) 등의 표준제정단체의 활동으로 표준화의 완성도가 높으며, 여러 IT 관련 기업들이 웹 서비스를 지지함으로써 실용화가 빠르게 되고 있는 기술이다. 이러한 웹 서비스의 관심이 많은 가운데, 지금까지의 웹 서비스 자체 연구보다는 실제 적용 측면에서 생기는 연구가 필요하게 되었다.

코리오그래피(choreography)와 오케스트레이션(orchestration)은 웹 서비스 참여자의 외부 웹 서비스들의 순차적인 프로세스를 구성하고, 내부 비즈니스 논리대로 웹 서비스를 실행시키는 핵심 요소이다 [4]. 오케스트레이션은 한 파트너에 대하여 내부 비즈니스 프로세스 논리를 의미하며, OASIS에서 정의한 BPEL4WS (Business Process Execution Language for Web Services)가 널리 사용된다 [24]. 반면 코리오그래피는 두 파트너 사이에서 일어나는 메시지 교환 방식으로 의미하며, 일반적으로는 둘 이상의 웹 서비스 참여자가 상호작용을 할 때 생길 수 있는 웹 서비스의 프로세스를 의미한다.

현재 WS-CDL(Web Services Choreography Description Language)이 대표적인 코리오그래피 기술언어로 평가 받는다. 코리오그래피는 셋 이상의 파트너가 웹 서비스에 참여하게 될 경우 특히 요구된다. 여러 파트너들 사이에서 각각의 롤 (role)에 따라 언제 어떤 상호작용으로 누구의 웹 서비스를 요청할지를 기술하여 공유할 필요성이 있고 여기에 WS-CDL이 사용된다.

일반적으로 WS-CDL은 완성된 웹 서비스의 코리오그래피를 보고 웹 서비스를 구성하는 Top-Down 방식을 목적으로 하지만 참여자가 다른 웹 서비스에 참여하기를 원한다면 Bottom-Up 방식으로 접근해야 한다. 즉, 여기서 다른 코리오그래피 간에 충돌이 발생하게 되는데, 이를 코리오그래피 충돌이라고 정의한다.

본 논문은 이기종 웹 서비스의 코리오그래피에 대하여 미디에이션을 통한 해결 방안을 제시하는 것을

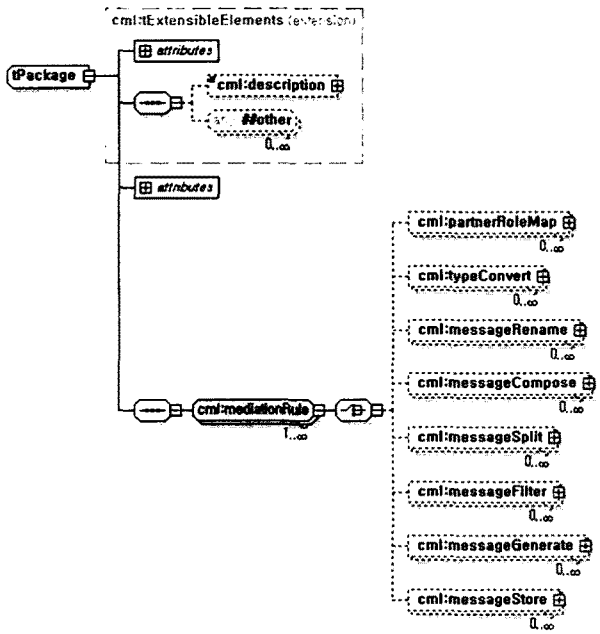
목적으로 한다.

먼저 두 개의 대상 WS-CDL에 대한 코리오그래피 충돌의 발견 방법이 필요하다. 본 논문 2장에서 코리오그래피 충돌에 대하여 분류 및 정의하고, 발견 방법을 제시하여 이를 해결하는 방안으로 XML (eXtensible Markup Language) [28]로 기술한 언어, WS-CML (Web Services Choreography Mediation Language)을 제안한다. 3장에서는 이렇게 작성된 WS-CML을 실제 코리오그래피 미디에이터가 처리하는 방식과 그 아키텍처를 다루고, 4장에서는 예제를 통해 WS-CML의 개발 방법 절차와 유효성을 검증한다.

코리오그래피 충돌 해결

WS-CML

WS-CML은 WS-CDL로 표현되어 있는 상호호환적인 이기종 코리오그래피 충돌을 해결하는 방법을 적어 놓은 XML 언어이다. 이는 다음 <figure 1>과 같다.



<Figure 1> WS-CML 스키마 구조도

본 절에서는 코리오그래피 충돌을 해결하기 위해 먼저 코리오그래피 충돌을 분류하고 이에 대한 해결 방안으로 WS-CML을 제시한다. WS-CML은 필요한 정보들을 WS-CDL에 참조하기 대문에, 많은 정보를 담기보다는 간단한 기능 위주로 구성되어 있다.

WS-CDL에서 일어나는 충돌 상황은 크게 3가지가 있다. 해당 웹 서비스에 참여하는 참여자들의 정보가 다를 때, 서로 상호 교환하는 메시지의 이름이나 타입이 다른 경우, 실제로 메시지 상호교환이 일어날 시에 필요한 메시지를 보내지 못하거나, 불필요한 메시지가 전송되어 오거나, 원래 자신의 WS-CDL이 계획한 순서대로 메시지 교환을 하지 못하는 경우에 발생된다.

다음 <Table 1>은 각각의 충돌 상황의 분류와 그 때 적용할 WS-CML 기능에 대한 관계를 나타낸다.

<Table 1> 각 충돌 상황에 대한 WS-CML 기능 유도

WS-CDL 충돌상황	충돌 분류	WS-CML 기능
파트너 정보가 다를 때	타입 충돌	Partner Mapping
메시지 내부의 타입이 다를 때	타입 충돌	Type Converting
메시지 이름이 다를 때	타입 충돌	Message Renaming
메시지 내의 세부 정보 타입이 다를 때	타입 충돌	Message Composing
	타입 충돌	Message Splitting
필요 없는 메시지를 수신 때	구조 충돌	Message Filtering
필요한 메시지를 수신하지 못할 때	구조 충돌	Message Generating
메시지 순서가 맞지 않을 때	구조 충돌	Message Storing

타입 충돌 해결

타입 충돌은 코리오그래피의 role, relationship, participant와 같은 파트너 정보의 타입이나 이름이 다른 WS-CDL과 다르거나, 같은 정보를 담고 있는 메시지 타입이나 이름이 다른 WS-CDL에서 정의된 것과 다른 경우 발생된다. 파트너 정보 타입이 다른 경우에는 role만 매핑시켜주면 relationship과 participant들에 대하여는 고려할 필요가 없다. 아래 <Table 2>에는 예제 cdl1과 cdl2에서 일어나는 충돌의 타입과 WS-CML의 해결방안을 설명한다.

<Table 2> cdl1과 cdl2 간의 타입 충돌표

cdl1	cdl2	충돌 타입	WS-CML
…	…	…	…
ClientRole	BuyerRole	롤 이름	Partner Mapping
creditCardNumber	creditCardNumber	세부 정보 타입	Type Converting
itineraryResult	scheduleResult	메시지 이름	Message Renaming
couponSelect	creditCardSend	메시지 타입	Message Composing
itinerarySend	scheduleSend	메시지 타입	Message Splitting
…	…	…	…

위의 <Table 2>를 통하여 충돌 타입에 따른 WS-CML 적용 방안에 대하여 알아볼 수 있다. 충돌 타입 열에서 '롤 이름'은 WS-CDL 간의 roleType이 다른 것으로서, 그에 따른 participantType, relationshipType, channelType, token들이 다르다는 의미이다. WS-CML에서는 roleType만 참조하기 때문에 서로 다른 roleType에 대해서만 고려한다. '세부 정보 타입'은 세부 정보의 타입이 다른 경우로 같은 세부 정보인데 WS-CDL에 따라 Int 타입과 String 타입으로 정의되어 충돌을 일으키는 경우가 있다. 세부 정보의 타입은 simpleType 형태인 XML 스키마 표준 문서에서 사용되는 primitive datatypes, derived datatypes들 중 하나로 결정된다 [20]. '메시지 이름'은 메시지 타입이 같지만 이름만 다른 경우로 메시지 이름을 수정하면 된다. '메시지 타입'은 메시지 내의 세부 정보들이 다르게 구성되어 있어서 메시지 타입이 달라 충돌이 일어나는 경우로, Message Composing과 Message Splitting을 통해 메시지 타입을 수정하는 방안이 필요하다.

Partner Mapping: 두 WS-CDL에서 같은 롤이지만 roleType이 다르게 표현되었을 경우에 사용된다.

```
...
<partnerRoleMap source="cdl1:ClientRole"
target="cdl2:BuyerRole" />
<partnerRoleMap source="cdl1:ServerRole"
target="cdl2SellerRole" />
...
```

Type Converting: 두 WS-CDL에서 정의된 메시지 간에 세부 정보 타입이 다른 경우에 사용된다.

```
...
<typeconvert fromMessage="cdl1:creditCardSend"
toMessage="cdl2:creditCardSend">
  <variable from="/creditCardNumber"
to="/creditCardNumber"/>
</typeConvert>
...
```

Message Renaming: 두 WS-CDL에서 똑 같은 타입으로 정의된 메시지라도 이름이 다르게 정의되는 경우가 있다.

```
...
<messageRename fromMessage="cdl2:scheduleResult"
toMessage="cdl1:itineraryResult" />
...
```

Message Composing: 한 WS-CDL에만 기술되어있는 메시지를 다른 WS-CDL에서 필요한 세부 정보들을 추출하여 만드는 방법을 Message Composing에서 제공한다.

```
...
<messageCompose toMessage="creditCardSend">
  <variable fromMessage="cdl1:creditCardNumber"
query="/creditCardNumber" to="/creditCardNumber" />
  <variable fromMessage="cdl1:couponSelect"
query="/coupon" to="/coupon" />
</messageCompose>
...
```

Message Splitting: Message Composing과 비슷하지만 긴은 반대로 한 메시지에 있는 여러 세부 정보들을 쪼개어 새로운 메시지를 만드는 기능이다.

```
...
<messageSplit fromMessage="cdl1:itinerarySend"
toMessage="scheduleSend">
  <variable from="/fromLocation" to="/fromLocation" />
  <variable from="/toLocation" to="/toLocation" />
  <variable from="/date" to="/date" />
</messageSplit>
...
```

구조 충돌 해결

구조 충돌은 Message Generating, Message Storing, Message Filtering의 3가지로 나뉜다. 아래의 <Table 3> 두 항공사간의 구조 충돌에 대한 예로서, 충돌 타입을 정의하고 WS-CML의 해결 방안을 제시한 표의 일부분이다.

<Table 3> 구조 충돌의 예와 해결 방안

cdl1	cdl2	충돌 타입	WS-CML
...	...	...	...
satisfactionSend	없음	메시지 차단	Message Filtering
	없음	메시지 부재	Message Generating
itinerarySend	scheduleSend	시간차 전송	Message Stroing
....	...	...	...

충돌 타입에 따라 WS-CML 적용 방안이 다른데, '메시지 차단'은 어떤 롤에서 보내는 메시지는 있는데 그 메시지를 받는 롤이 아무도 없을 때 생기는 충돌로 MessageFiltering을 사용한다. '메시지 부재'는 특정 롤이 받아야 하는 메시지를 아무도 보내주지 않는 경우 생기고, 이는 Message Generating을 사용해야 한다. 충돌 타입에서 '시간차 전송의 의미는 한쪽 WS-CDL에서 메시지를 전송하지만, 다른 쪽에서는 나중에 받도록 설계되어 있는 경우 생긴다. 이는 Message Storing을 통하여 해결이 가능하다.

Message Filtering

```
...
<messageFilter fromMessage="cdl1:satisfactionSend" />
...
```

Message Generating

```
...
<messageGenerate toMessage="cdl1:ackCouponSelected" />
...
<messageGenerate toMessage="cdl1:payTypeSelection">
  <assign target="/payTypeSelection" source="creditCardPayment" />
</messageGenerate>
...
```

Message Storing

```
...
<messageStore fromMessage="tns:scheduleSend" toMessage="cdl2:scheduleSend" />
...
```

코리오그래피 미디에이터

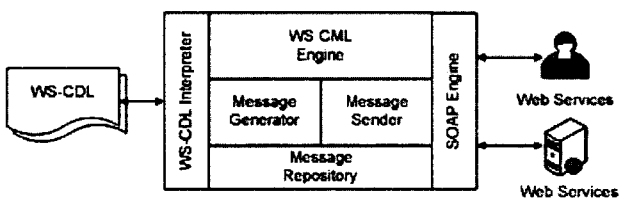
역할

코리오그래피 미디에이터는 웹 서비스 참여자들 사이에서 코리오그래피 충돌을 해결하여 상호운용성 향상에 기여하려는 목적을 가지고 있다. 이는 한쪽 롤에 임베디드되어 그 롤과 상호작용하는 모든 상호작용을 관리한다.

현재 WS-CDL은 실행 언어이기보다는 기술 언어로서, 웹 서비스를 구성하고자 할 때 사용된다. 이렇게 WS-CDL은 코리오그래피를 정의할 때보다 실제로 구축되어 있는 웹 서비스 상에서 코리오그래피가 작지켜지도록 하는데에는 아무런 역할을 하지 못한다. 즉, 코리오그래피 미디에이터를 위해서는 실행 가능한 WS-CDL이나 다른 코리오그래피 룰에 대한 규약 언어의 개발이 필요하다.

아키텍처

코리오그래피 미디에이터의 구조는 다음 <Figure 2>와 같다.



<Figure 2> 코리오그래피 미디에이터 아키텍처

미디에이션 하고자 하는 롤이 받거나 보내는 메시지는 코리오그래피 미디에이터의 SOAP Engine을 통한다. 이렇게 받은 메시지는 WS-CML Engine이 해석한 WS-CML 룰에 따라 이벤트가 발생한다. WS-CML의 룰에 따라 새로운 타입의 메시지가 생성이 되어야 하면, Message Generator는 WS-CML Engine이 디자인 한 대로 실제 메시지 인스턴스를 만들고 이를 Message Repository에 저장한다. 메시지를 전송해야 하는 경우에는 WS-CML Engine이 Message Sender에게 Message Repository에 있는 특정 메시지를 SOAP Engine에게 전달하게 한다. 특정 메시지를 전송하지 않아야 하는 경우에는 WS-CML Engine이 Message Sender에게 전달 명령을 내리지 않으면 전달이 안되기 때문에, 특정 메시지의 차단이 가능하다.

예제

시나리오

본 장에서는 두 개의 항공사에서 예약 대행 업무를 하려는 여행사를 예로 들어보고자 한다.

K 항공사의 비행기표 예약 서비스 코리오그래피는 다음과 같다.
- 비행기표 할인 서비스로 쿠폰 선택.
- 출발지, 도착지, 날짜, 한도 가격 입력.
- 계속적으로 검색 가능.
- 무통장 입금, 신용카드 결제 중 선택.
- 무통장 입금 시 해당 코드 받음.
- 신용카드 유효성 검사.
- 결과를 예/아니오로 알려줌.

A 항공사의 코리오그래피는 다음과 같다.
- 출발지, 도착지, 시간 입력.
- 신용카드로만 결제 가능하기 때문에 추가 할인 쿠폰 정보를 함께 보냄.
- 신용카드 유효성 검사.
- 결과를 예/아니오로 알려줌.

WS-CML 개발

다음 <Table 4>은 두 항공사간의 코리오그래피 사이에 충돌이 발생하는 경우와 그에 대한 WS-CML 해결책을 보여주고 있다.

<Table 4> K항공사와 A항공사의 충돌 타입

K항공사 CDL	A항공사 CDL	충돌 타입	WS-CML
ServerRole	SellerRole	롤 이름	Partner Mapping
ClientRole	BuyerRole	롤 이름	Partner Mapping
couponSelect	creditCardSend	메시지 타입	Message Composing
creditCardSend	CreditCard Send	메시지 타입	Message Composing
creditCardNumber	creditCard Number	세부 정보 타입	Type Converting
itineraySend	scheduleSend	메시지 타입	Message Splitting
itineraryResult	scheduleResult	메시지 이름	Message Splitting
couponSelect	creditCardSend	시간차 전송	Message Storing
ackCouponSelected	없음	메시지 부재	Message Generating
itinerarySend	scheduleSend	시간차 전송	Message Storing
satisfactionSend	없음	메시지 차단	Message Filtering
ackSatisfaction	없음	메시지 부재	Message Generating
payTypeSelection	없음	메시지 차단	Message Filtering
ackPayType	없음	메시지 부재	Message Generating

WS-CML 문법 및 위의 <Table 4>를 참조하여 다음과 같은 WS-CML을 만들 수 있다.

```
...
(01) <partnerRoleMap source="cdl1:ClientRole" target="cdl2:BuyerRole" />
(02) <messageCompose toMessage="creditCardSend">
(03) <variable fromMessage="cdl1:creditCardNumber" query="/creditCardNumber" to="/creditCardNumber" />
(04) <variable fromMessage="cdl1:couponSelect" query="/coupon" to="/coupon" />
(05) </messageCompose>
(06)<messageStore fromMessage="tns:creditCardSend" toMessage="cdl2:creditCardSend" />
...
```

(01)의 partnerRoleMap은 두 WS-CDL의 roleType이 다르게 정의되어 있는 경우 cdl1의 ClientRole과 cdl2의 BuyerRole이 실제적으로 같은 롤이라는 의미를 내포한다. (02)에서 (05)까지의 messageCompose는 cdl1의 creditCardNumber의 creditCardNumber 세부 정보와 cdl1의 couponSelect의 coupon 세부 정보를 합쳐서 새로운 메시지 타입을 만든다. (06)은 (02)의 message-Compose에서 정의된 메시지를 cdl2의 creditCardSend와 매핑시키는 작업이지만, 실제적으로는 코리오그래피 미디에이터 내부 DB에 메시지를 저장시켜 놓고 cdl2에서 해당 메시지가 필요할 때 질의하여 전송한다.

결론

비즈니스 프로세스 관리 측면에서도 웹 서비스가 활발히 쓰이고 있는 반면, 웹 서비스의 코리오그래피는 표준화 자체도 현재 진행 중일만큼 완성도가 높지 않으며, 산업에 적용할 방안에 대하여도 확정적이지 않은 상태이다.

코리오그래피의 기술 언어 WS-CDL에 대하여도 최근 2년간 빠르게 표준화 작업 절차를 밟으며 개발중인 언어이기 때문에 여러 가지 문제 제기 및 개선의 여지가 남아있다. 코리오그래피 충돌도 이러한 관점에서 나온 것으로, WS-CDL은 Top-Down 방식으로 전체 프로세스가 명확하고 새로운 웹 서비스를 구축할 때 사용되는 것으로 의도하였으나, 현실에서는 실행되고 있는 웹 서비스의 코리오그래피를 수정하기 어렵다. WS-CDL이 나오기 앞서 현재 많은 곳에서 이미 웹 서비스가 구현되었고, 상용화 되는 추세이기 때문에, 구축된 웹 서비스에 WS-CDL로 새롭게 기술된 코리오그래피를 제시한다 하더라도, 모든 웹 서비스 벤더들이 이를 따르지 못한다는 한계점이 있다.

본 논문에서는 이러한 여러 가지 한계점을 인식하고 이미 기존에 구축되어 있는 웹 서비스에 사용자가 새롭게 참여할 때 발생할 수 있는 문제인 충돌을 미리 정의하여 참여자가 겪어야 하는 불편함을 해결하고자 하였다. 이를 해결하기 위한 방안으로 WS-CML 언어를 제시한다.

코리오그래피 충돌은 메시지 타입 충돌과, 상호작용을 제어하는 구조 충돌로 크게 2가지 타입이 이러한 충돌의 해결 방안으로 Partner Mapping, Type Converting, Message Renaming, Message Composing, Message Splitting이 있고, 구조 충돌에 대해서는 Message Filtering, Message Generating, Message Storing이 있다. 이러한 것들을 병렬적으로 정리한 XML 문서가 WS-CML이고, 이는 코리오 그래피 미디에이터의 WS-CML Engine 상에서 미디에에이션을 제어하는 핵심 언어이다.

본 연구를 통하여 지금까지의 접근 방법(Top-Down)이 아닌 Bottom-up 방식의 접근을 통해, 기존의 참여자가 서로 다른 웹 서비스와 통신하기 위한 방안을 제시하였다. 이는 코리오그래피 간의 충돌이 있다고 하여도 사용자는 자신의 프로세를 크게 변경하지 않고 통신이 가능하다는 측면에서, 웹 서비스 구성원들의 다양한 웹 서비스 참여를 유도한다.

본 연구에서는 코리오그래피 미디에이션에서 같은 도메인의 비슷한 웹 서비스에 대하여 적용했으나, 좀 더 도메인을 확장하여 미디에이션 범위를 넓히는

것과 보다 복잡한 코리오그래피 구조를 지닌 WS-CDL을 미디에이션 하는 것이 추구 연구 과제라 고 할 수 있겠다.

Acknowledgement

이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임 (No. R01-2007-000-11167-0)

참고 문헌

[1] A. Arkin, S. Askary, S. Fordin, et al., "Web Services Choreography Interface (WSCI) 1.0," W3C Note, 2002. [Online]. Available: http://www.w3.org/TR/wsci/

[2] A.Banerji, C.Bartolini, D.Beringer, et al., "Web Services Conversation Language (WSCL) 1.0," W3C Note, 2002. [Online]. Available: http://www.w3.org/TR/wscl10/

[3] A. Barros, M.Dumas, and P. Oaks, "A Critical Overview of the Web Services Choreography Description," *Business Process Trends*, Vol.3, 2005

[4] C.Peltz, "Web Services Orchestration and Choreography," *IEEE Computer*, Vol. 36, No. 10, pp. 46-52, 2003

[5] D.Box, D. Ehnebuske, G. Kakivaya, et al., "Simple Object Access Protocol (SOAP) 1.1," W3C Note, 2000. [Online]. Available: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[6] D. Raman, H. Lausen, and U. Keller, "Web Service Modeling Ontology (WSMO)," Final Draft, 2006. [Online]. Available: http://www.wsmo.org/TR/d2/v1.3/

[7] E.Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," W3C Note, 2001. [Online]. Available: http://www.w3.org/TR/wsdl/

[8] E. Cimpian and A. Mocan, "WSMX Process Mediation Based on Choregraphies," *Proceedings of the 3rd International Conference on Business Process Management*, Nancy, France, Sep. 2005

[9] F. Curbera, R. Khalaf, N. Mukhi, et al., "The Next Step in web Services," *Communications of the ACM*, Vol. 46, No. 10, pp 29-34, 2003

[10] F.Leymann, "Web Services Flow Language," IBM, May 2001

[11] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, "Web Services," *Springer*, 2004

[12] J. Clark and S. DeRose, "XML Path Language (XPath) 1.0," W3C Recommendation, 1999. [Online]. Available: http://www.w3.org/TR/XPath/

[13] J. de Bruijn, H. Lausen, and R. Krummenacher, "The Web Services Modeling Language (WSML)," Final Draft, 2005. [Online]. Available: http://www.w3.org/TR/d16/d16.1/v0.21/

[14] M. H. Brestein, J. R. Hoobs, O. Lassila, et al. "DAML-S: Web Service Description for the Semantic Web," *Proceedings of the 1st International Semantic Web Conference of the Semantic Web*, Sarinia, Italia, Jun. 2002

[15] M. P. Singh and M. N. Hunhns, "Service_Oriented Computing," WILEY, 2005

[16] M. Zaremba, A. Haller, M. Zaremba, and M. Moran, "WSMX-Intrastructure for execution of semantic web services," *Proceedings of the 3nd International Conference on Web Services*, San Diego, Californai, USAL, Jul. 2004

[17] M. Zaremba, M. Moran, and T. Haselwanter, "WSMX Architecture," Final Draft, 2005. [Online]. Available: http://www.wsmo.org/TR/d13/d13.4/v0.3/

[18] N. Kavantzas, D. Burdett, G. Ritzinger, et al., "Web services Choreography Description Language (WS-CDL) 1.0," W3C Candidate Recommendation, *2005*. [Online]. Available: http://www.w3.org/TR/ws-cdl-10/

[19] O. Zimmermann, M. Tomlinson, and S. Peuser, "Perspectives on Web Services," *Springer*, 2003

[20] P. V. Biron, K, Permanente, and A. Malhotra, "XML Schema Part2: Datatypes Second Edition," W3C Recommendation, 2004. [Online]. Available: http://www.w3.org/TR/schema-2/

[21] S. Narayanan and S. A. Mcllraith, "Simulation, Verification and Automated Composition of Web Services," *Proceedings of the 11th International Conference on World Wide Web*, Honolulu, Hawaii, May 2002

[22] S. Pemberton, D. Austin, T. Celik, et al., "The Extensible HyperText Markup Language (Second Edition)," *W3C Recommendation*, 2000. [Online]. Available: http://www.w3.org/TR/html/

[23] S. Thatte, "XLANG: Web Services for Business Process Design," Microsoft, 2001

[24] T. Andrews, F. Curbera, Y. Goland, et al., "Business Execution Language for Web Services Version 1.1," IBM, 2003

[25] T. Bellwood, L. Clement, D. Ehnebuske, et al. "UDDI Version 3.0," UDDI Spec Technical Committee Specification, 2002. [Online]. Available: http://uddi.org/pubs/uddi-v3.00-published-20020719.htm

[26] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, Vol. 284, No.5, pp. 34-43, 2001

[27] T. Berners-Lee, R. Fielding, U. Irvine, And L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," Internet RFC 2396, 1998. [Online]. Available: http://www.isi.edu/in-notes/rfc2396.txt

[28] T. Bray, J. Paoli, C. M. Sperberg-McQueen, et al., "Extensible Markup Language (XML) 1.0 (Fourth Edition)," W3C Recommendation, 2006. [Online]. Available: http://www.w3.org/TR/xml/

[29] W. M. P. van der Aalst, A. H. M. Ter Hofstede, B. Kiepuzewski, and A. P. Barros, "Workflow Patterns," *Distributed and Parallel Databases*, Vol. 14, No. 1, pp. 5-51, 2003

[30] W. Vogels, "Web Services Are Not Distributed Objects," *IEEE Internet Computing*, Vol. 7, No. 6, pp. 59-66, 2003