DOI: 10.3745/KIPSTD.2008.15-D.4.513

서비스 지향 아키텍처의 클라이언트를 위한 실용적 프로세스 모델

이 재 유[†] · 김 수 동[††]

요 약

서비스 지향 아키텍처(Service-Oriented Architecture, SOA)는 여러 어플리케이션들에서 사용될 수 있는 범용적인 서비스를 개발하여 배포(Publish)하고 동적으로 발견(Discover), 조립(Composition)하여 어플리케이션을 만드는 기술이다. 따라서 SOA는 Publish-Discover-Invoke 형태의 컴퓨팅 방식을 사용하고 있으며, 이는 기존의 객체지향, CBD의 컴퓨팅 방식과 상당한 차이를 보인다. SOA의 서비스는 절차적 프로그래밍의 함수, 객체지향의 객체, CBD의 컴포넌트와는 다른 구성 단위(Unit)이며, 재사용성과 보편성이 서비스의 기본적인 요구사항이 된다. 또한 서비스 제공자와 소비자 간의 사전 상호인지나 정보의 공유없이, 서비스 제공자가 재사용을 고려하여 서비스를 개발하고, 서비스 사용자는 이를 검색한 후 사용하는 방식이므로 제공자와 소비자 간의 결합도가 매우 낮은 특징을 가지고 있다. SOA에서 필요한 서비스를 실행시간에 검색, 선택하여 사용하는 방식도 기존 컴퓨팅 방식에는 적용되지 않은 개념이다. 따라서, 객체지향 프로그래밍에서처럼 사용자 인터페이스를 만들고 서버의 기능을 JSP, RMI 등을 사용하여 호출하는 방식은 SOA 서비스 클라이언트 프로그램을 설계 하는데 잘 적용될 수 없다. SOA 서비스 클라이언트 개발을 위해서는 서비스의 특징과 SOA의 서비스 사용 절차를 설계 단계에서 적용할 수 있어야 하며, 이를 위하여 서비스 클라이언트 개발을 위한 실용적이고 체계적인 개발 프로세스가 정의되어야 한다. 그러나 아직까지 서비스 제공자 측면의 SOA 서비스 개발을 위한 프로세스조차 정의가 부족하고, 서비스 사용자 측면의 클라이언트 개발 프로세스는 소수의 가이드라인을 제외하고는 전무한 실정이다. 따라서 본 논문에서는 효율적인 서비스 검색과 실행을 위한 실용적이고 체계적인 개발 프로세스와 각 단계별 지침을 정의하고, 호텔 검색 및 예약 서비스 시스템을 통하여 제시된 개발 프로세스를 적용한다.

키워드 : 서비스 지향 아키텍처, 프로세스, 클라이언트 프로그램, SOA 표준

A Practical Process Model for Clients in Service-Oriented Architecture

Jae Yoo Lee[†] · Soo Dong Kim[††]

ABSTRACT

Service-Oriented Architecture(SOA) is an method to develop applications by developing and publishing reusable services which potentially be used in various applications, and discovering and composing right services dynamically. SOA adopts a paradigm of *publish-discover-invoke*, which is considerably different from object-oriented and component-based development(CBD) approaches. The *service* in SOA is different from *function* in procedural programming, *object* in object-oriented programming, and *component* in CBD, and its fundamental requirement is a high level of reusability and applicability. In SOA, service providers and service consumers are loosely coupled since the providers try to develop reusable services and the consumers try to locate right services without knowing much about the providers and their published services. Moreover, the process of searching, choosing and invoking right services is not presented in conventional programming paradigms. Therefore, conventional approaches to developing user interfaces and invoking the functionality on servers through JSP, and RMI in object-oriented programming cannot well be applicable to designing clients' programs in SOA. Therefore, there is a high demand for a practical and systematic process for developing clients' applications, and the such a process should be devised by considering key characteristics of services and SOA. However, little work on this area is known to date, and there has not a process for client side just except few guide lines for developing service client. In this paper, we propose a practical and systematic development process for developing clients' applications in SOA. Then, we define instructions for carrying out each activity in the process. To show the applicability of the proposed work, we show the result of applying our process in developing a services application for searching and booking hotels.

Key Words : Service-Oriented Architecture, Process, Client Program, SOA Standards

1. 서 론

서비스 지향 아키텍처(Service-Oriented Architecture, SOA)는 여러 어플리케이션들에서 사용될 수 있는 범용적인 서비스를 개발하고, 이를 재사용하여 어플리케이션을 신속하고 경제적으로 개발할 수 있는 기술로써, 학계 및 산업계에서 각광을 받고 있다. SOA는 서비스 제공자(Provider)가 공통으로 사용될 수 있는 기능을 서비스로 개발하여, 서비스 레지스트리(Registry)에 등록하고, 서비스 소비자(Consumer) 즉 서비스 사용자가 표준 검색 인터페이스인 UDDI API를 사

※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음.
† 준 회 원 : 숭실대학교 컴퓨터학과 석사과정
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2008년 2월 22일
수 정 일 : 1차 2008년 4월 1일
심사완료 : 2008년 4월 22일

용하여 필요한 서비스를 검색하고, 이들을 조립하여 실행한다. 이러한 SOA의 Publish-Discover-Invoke 형태의 컴퓨팅 방식은 기존의 객체지향, 컴포넌트 기반 개발(Component-Based Development, CBD)의 컴퓨팅 방식과 상당한 차이를 가지고 있다. SOA의 서비스는 절차적 프로그래밍의 함수, 객체, CBD의 컴포넌트와는 다른 구성 단위(Unit)이며, CBD 컴포넌트 개발에서는 선택적인 특성(Feature)이었던 재사용성 및 범용성이 SOA 서비스에서는 가장 기본적인 요구사항이 된다. 또한, 서비스 제공자와 소비자 간의 사전 상호인지나 정보의 공유없이, 서비스 제공자가 재사용을 고려하여 서비스를 개발하고, 서비스 사용자는 이를 검색한 후 사용하는 방식이므로 제공자와 소비가 간의 결합도(Coupling)가 매우 낮다. 이는 SOA의 클라이언트 프로그램이 서버에 있는 특정 서비스를 지정하여 개발하지 않아야 하는 방식으로 나타난다. 또한, SOA에서 서비스를 사용하기 전에 원하는 서비스를 검색하는 단계도 기존 컴퓨팅 방식에는 포함되어 있지 않는 절차이다.

이러한 근본적인 차이점으로 인해, 객체지향 프로그래밍이나 CBD에서처럼 서버에 있는 기능을 JSP, RMI 등을 사용하여 호출, 실행하는 방식은 SOA 서비스 클라이언트 프로그램 개발에 잘 적용될 수 없다. 즉, SOA 서비스 클라이언트 프로그램 개발을 위한 실용적이며 효과적인 개발 프로세스가 필요하다. 또한, SOA에서는 여러 서비스 제공자가 여러 서비스 레지스트리에 서비스를 등록하므로, 이들 중에서 최적의 서비스를 선정, 제공하는 프로세스가 필요하다. 그 뿐 아니라, WSDL, SOAP, UDDI, BPEL등 10여개 이상의 산업계 표준들을 적용하고 있는데, 이런 표준들간의 관계를 잘 고려하여 클라이언트 개발자가 효율적으로 SOA 표준을 준수하여 클아이언트 프로그램 개발을 지원해주는 프로세스가 요구된다.

Unified Process 등 객체지향 개발 프로세스나 방법론에서는 객체라는 기본 단위로 어플리케이션을 구성하고, 클래스와 상속 장치를 기반으로 설계를 하며, 라이브러리의 클래스를 포함(Include, Import)하는 형태로 재사용이 이루어진다. 하지만 이런 방식은 SOA의 기본 단위인 서비스나, 동적으로 서비스 발견, 그리고 포함 대신 서비스의 원격 실행(Remote Invocation)하는 장치를 지원하지 않는다. 또한, 그 간에 소개된; 서비스 지향 모델링 아키텍처(Service Oriented Modeling Architecture, SOMA) 등 SOA 개발 방법론들은 서비스 제공자가 서비스를 설계 구현하는 프로세스를 제공하지만, 서비스 사용자가 필요로 하는 서비스 클라이언트 프로그램을 지원하는 효과적인 프로세스는 포함하지 않고 있다.

본 논문에서는, SOA 표준을 준수하고, 원하는 서비스를 식별하여 클라이언트 프로그램을 구현, 실행하는데 필요한 프로세스와 세부 지침을 제공한다. 즉, 서비스 저장소에 등록된 서비스들을 검색하여, 서비스 클라이언트가 필요로 하는 최적의 서비스를 발견하고, 이를 구독하여 서비스 클라이언트 프로그램을 작성(생성)하고, 이를 이용하여 대상 서비스를 실행하는 전 과정을 다루는 프로세스를 제안하고, 세부 작업 지침을 제공한다.

본 논문의 구성을 다음과 같다. 제 2장에서는 현재의 SOA 개발 프로세스에 대한 관련기술과 SOA 환경 구축을 위한 주요 표준 기술들에 대하여 기술한다. 제 3장에서는 SOA에서의 서비스 검색, 발견, 실행을 위한 서비스 사용자 측면의 클라이언트를 작성하는 프로세스 모델과 각 단계별 지침들을 제안한다. 제 4장에서는 사례연구를 통하여 제안된 서비스 클라이언트 개발 프로세스와 지침의 실용성을 보여주고, 제 5장에서는 평가 및 결론을 제시한다.

본 논문에서 제시하는 프로세스를 사용할 경우, SOA 관련 표준이 적용된 개발 프로세스를 통한 개발 시간의 단축과 표준 준수를 통한 확장성있는 효율적인 개발 등의 장점으로 서비스 사용자를 위한 고품질의 서비스 클라이언트 프로그램의 작성이 가능하다.

2. 관련 및 기반 연구

서비스 지향 모델링 아키텍처(Service-Oriented Modeling Architecture, SOMA)는 서비스 식별, 서비스 상세화, 서비스 구현의 3가지 단계로 구성된다[1]. 서비스 식별 단계에서는 요구사항 분석을 통하여 식별된 서비스를 식별하고, 상세화 단계에서는 식별된 서비스와 컴포넌트, 서비스의 흐름, 메시지와 이벤트 등의 정보를 상세화한다. 상세된 정보를 통해 실제 구현에 응용될 수 있도록 명시한다. 도메인 분할, 서비스 목표 모델링, 기존 시스템 분석의 세가지 활동로 서비스를 식별하는데, 각 활동에 알맞게 하향식, 상향식, 절충식의 접근으로 서비스를 분석하고 식별한다. 식별 단계에 도출된 후보 서비스들은 상세화 단계에서 상세히 설계된 후 실제 서비스 구현 단계에서 맞춤 설계, 통합, 변환, 청약, 아웃 소싱 등의 방법을 통하여 산출한다. 이 기법에서는 SOA의 계층적 구조를 수행적 시스템 레이어, 상업적 컴포넌트 레이어, 서비스 레이어, 비지니스 프로세스 레이어 조합 레이어, 프레즌테이션 레이어, 통합 기술 레이어, QoS 레이어로 구성하는 것을 제시한다. SOMA는 서비스 설계와 구현, 즉 서비스 제공자를 위한 프로세스이며, 서비스 클라이언트 프로그램 설계와 구현 프로세스는 포함하지 않고 있다.

Erl Thomas의 연구는 서비스 지향 구조적 개발 생명주기를 제시하였다. 개발하고자 하는 서비스 후보의 도출과 설계를 위해 서비스지향 분석, 서비스지향 디자인, 서비스 개발, 서비스 테스팅, 서비스 배치, 서비스 관리단계를 제시한다[2]. 서비스 지향 구조적 개발 생명주기는 서비스 재사용성을 강조하였고, 비즈니스 프로세스를 구현하기 위한 과정을 상세화하였지만, 개발된 서비스를 이용하기위한 클라이언트의 개발은 다루고 있지 않다.

UDDI는 웹 서비스를 위한 웹 기반 분산 레지스트리의 표준이며, 서비스 제공자들이 개발한 서비스를 등록, 검색, 발견하기 위한 메커니즘을 제공한다[3]. UDDI 레지스트리는 3가지 종류의 'well-defined 인터페이스'를 통해 질의할 수 있다. 첫째, 웹 서비스 제공자는 서비스 자체에 대한 정보와 서비스 제공 회사에 대한 정보, 그리고 접근 정보 등과 같은

내용을 등록(Publish)한다. 둘째, 서비스 사용자는 원하는 서비스 제공자와 비지니스 서비스를 찾기 위해 검색(Discover)한다. 마지막으로 원하는 서비스를 검색했다면 서비스와 상호 거래를 위해 접근하기 위해 사용되는 정보들에 대한 바인드(Bind)를 수행한다. UDDI 버전 2.0에서는 데이터 구조를 총 5가지 구성 요소로 정의하고 있다. <tModel>: tModel은 UDDI 레지스트리의 테크니컬 명세를 나타낸다. 이 테크니컬 명세는 회사간에 데이터를 주고 받는 표준, 웹 서비스 인터페이스 정의 같은 모든 종류의 것들을 포함한다. <businessEntity> 엘리먼트: UDDI 데이터 계층구조의 최상위 엔트리이다. 이 엘리먼트는 회사 자체에 대한 정보(회사명, 회사주소, 전화번호 등)를 갖고 있다. <businessService> 엘리먼트: 서비스의 이름, 부수적인 기술내용, <categoryBag>, <bindingTemplate> 엘리먼트로 이루어져 있다. <bindingTemplate> 엘리먼트: 서비스에 관한 URL, 이메일 주소, 전화번호 등 어떠한 정보라도 담을 수 있다[4]. 서비스 사용자는 UDDI에 등록된 서비스의 검색을 위해 제공되는 UDDI 검색 API를 통해 UDDI 레지스트리의 검색 기능을 사용할 수 있다.

WSDL은 웹 서비스를 기술하기위해 사용되는 XML 기반의 언어이다[5]. WSDL은 웹 서비스가 제공하는 서비스 메소드와 각 메소드의 입/출력 데이터의 구조, 서비스 Endpoint와 바인딩 정보 등을 기술하는데 사용된다. 또한 WSDL은 마이크로소프트, IBM 등에 의해 주도된 UDDI의 기본이라 할 수 있다. 즉, UDDI는 서비스 제공자들이 서비스 내용을 네트워크 상에 스스로 등록할 수 있게 해주는 XML 기반의 Service Registry이며, WSDL은 서비스의 정보를 등록하고 표시하기위한 언어이다.

BPEL(Business Process Execution Language)은 비즈니스 프로세스의 재사용과 통합이 가능하도록 비즈니스 프로세스를 기술하는 웹 서비스 기반의 표준 모델링 언어로써, 비즈니스 프로세스와 그 프로세스에 관계된 파트너들 사이의 상호작용에 근거하여, 비즈니스 프로세스 행위를 기술하기 위해 필요한 모델 및 문법을 정의하는 언어이다[6]. BPEL은 WSDL에 정의된 서비스 인터페이스에 기반하여 만들어지며, WSDL에서는 불가능한 상태를 가진, 지속적인 상호작용의 표현이 가능하다.

SOAP은 XML과 HTTP를 기반으로 네트워크 상에 존재하는 각종 컴포넌트간의 호출을 효율적으로 할 수 있게 하는 통신 프로토콜이다[7]. 이는 다음과 같은 두 가지 특징이 있다. 첫째, XML 기반이므로 이 기종의 시스템간에 객체를 공유하여 사용할 수 있다. 둘째, 인터넷 표준인 HTTP를 사용함으로써 널리 사용되고, 기존 분산객체시스템에서의 방화벽 문제를 해결할 수 있다. 따라서 SOAP은 독자적인 기술로 개발되어 상호 운용성 문제가 있는 기존 분산객체시스템의 문제를 용이하게 해결할 수 있어 많은 관심이 증폭되고 있다.

JAX-WS는 SOAP 기반의 자바 웹 서비스 개발 기술로서, WSDL과 자바 사이의 맵핑 관계를 기술한다. JAX-WS는 웹 서비스 및 서비스 클라이언트를 만들 수 있는 도구인 wsimport와 wsgen을 제공하며, 서비스 제공자는 이를 이용하여 자바 5.0 기반의 웹 서비스를 작성할 수 있으며, 서비스 사용자는 wsgen을 통하여 서비스 Endpoint 클래스를 읽고 서비스 배포 및 호출에 필요한 모든 아티팩트를 생성한다. 즉, 웹 서비스와 연결되는 클래스를 생성하여 서비스 사용자가 일반 객체를 호출하는 방식으로 서비스를 이용가능하도록 지원한다.

3. 프로세스 모델 및 단계별 지침

본 장에서는 서비스 클라이언트의 프로세스 모델을 제안하며, 각 단계에 적용될 지침을 제시한다. 이 프로세스는 (그림 1)와 같이 세 가지 단계와 관련 산출물들로 구성된다.

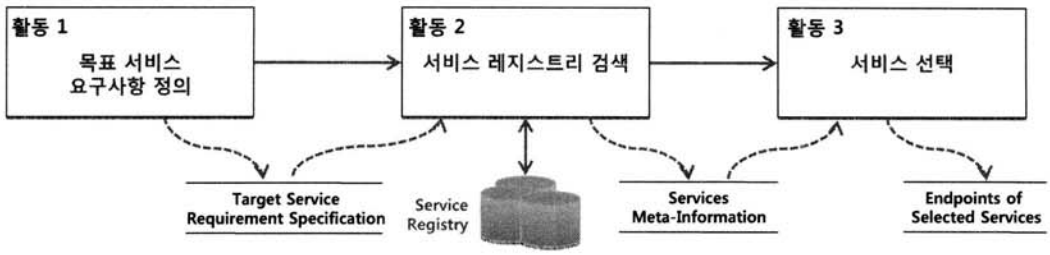
3.1 단계 1. 서비스 검색

SOA에서 서비스 제공자가 개발한 서비스의 메타(Meta) 정보를 기반으로 Service Registry에 등록하고, 서비스 사용자는 Service Registry에 등록된 서비스들을 검색하여 요구하는 서비스를 발견, 사용하게 된다[8]. 단계 1은 서비스 사용자가 원하는 서비스의 요구사항을 정의하고, 이 요구사항을 만족하는 서비스들을 Service Registry를 이용하여 검색하는 과정이다. 이 단계는 (그림 2)와 같이 세개의 활동들로 구성된다.

활동 1은 서비스 클라이언트가 요구하는 목표 서비스의 요구사항을 정의하는 과정이다. 서비스 요구사항은 세 가지의 항목으로 정의하는데, 즉, 기능적 요구사항, 비기능적 요구사항 및 인터페이스 요구사항이다. 기능적 요구사항은 사용자가 원하는 결과를 얻기위해 필요한 기능들로 나타나며, 비즈니스 로직에서 각각의 활동들로 나타난다. 이러한 활동들은 목적과 제공하는 기능으로 명세되어야한다.

비기능적 요구사항은 서비스의 품질 요소들로 이루어진 Quality of Service(QoS)와 서비스가 실행될 때 보장해야 하는 계약 성격의 Service Level Agreement(SLA)로 구성된다[9]. SOA 서비스는 사용자가 알지 못하는 다양한 서비스 제

(그림 1) 프로세스의 단계와 관련 산출물



(그림 2) 서비스 검색 단계의 세가지 활동들

〈표 1〉 목표 서비스 요구사항 명세서

Target Service Requirement Specification (TSRS)

1. Functional Requirements
   1.1. Functional Requirement #1
   .....
2. Non-Functional Requirements
   2.1. Non-Functional Requirement #1
   .....
3. Interface Requirements
   3.1 Interface Requirement #1
   .....

공자들에의해 개발되고, 또한 서비스가 수정, 보완 등 진화(Evolve)될 수 있기 때문에, 원하는 서비스의 QoS와 SLA를 정의하는 것이 고품질 서비스 검색을 위하여 중요하다.

서비스의 인터페이스 요구사항은 클라이언트가 서비스를 사용하기 위하여, 미리 정해진 서비스 인터페이스가 있는 경우에 정의하면 된다. 예를 들면, 사용자가 실행할 클라이언트 프로그램이 특정 도메인에서 이미 정해진 표준 인터페이스를 준수해야 할 경우이다.

이와 같은 정보들은 <표 1>의 목표 서비스 요구사항 명세서(Target Service Requirement Specification, TSRS) 형태로 문서화하고, 활동 2에서 Service Registry 검색에 사용된다. 서비스 사용자는 TSRS에 명시된 항목들을 기반으로 목표 서비스를 실현하기위한 기능들을 기존 서비스들과 매칭하여 필요 서비스 리스트를 뽑아낸다.

활동 2는 정의된 요구사항 명세를 기반으로 Service Registry를 검색하여 사용가능한 서비스들의 리스트를 획득하는 과정이다. 이용가능한 서비스의 식별은 TSRS에 명시된 사항들을 기반으로 SOA 표준 서비스 레지스트리, 즉 UDDI (Universal Description, Discovery and Integration) 에 등록된 서비스들의 메타 정보를 비교하는 과정으로 이루어진다. SOA 표준을 준수한 Service Registry는 UDDI 표준에 따라 서비스 등록시 서비스 제공자 관련 정보, 표준화된 카테고리에 따른 분류, 서비스 Endpoint, 서비스 설명과 같은 서비스 관련 정보를 함께 등록하고, 이는 메타정보로써 서비스 사용자에게 제공된다. 따라서 서비스 사용자의 메타정보 접근을 위하여 Service Registry는 UDDI API를 제공하며, 이를 이용하여 사용자는 TSRS와 메타정보를 비교할 수 있다. 대표적인 검색 메소드로 FindBusiness( ), FindService( )등이 있다. FindBusiness( ) 메소드는 다음과 같이 여러개의 입력 매개변수들을 가지고, 이 검색 조건에 만족하는 비지니스들의 목록(List)을 반환한다[10].

public BusinessList FindBusiness(Vector, DiscoveryURLs, IdentifierBag, CategoryBag, TModelBag, FindQualifiers, int);

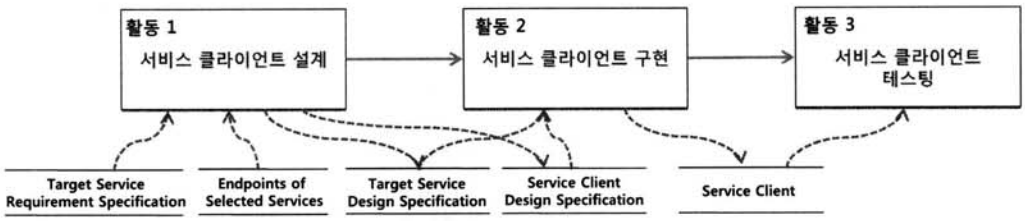
활동 3은 전달받은 서비스 리스트의 각 서비스 메타정보를 분석하여 서비스를 선택하는 과정이다. 사용자가 요구하는 기능을 제공하는 서비스가 복수 개가 있을 경우, 기능적 요구사항과 인터페이스 요구사항이 충족하는 어떤 서비스라도 선택하면 되나, 비기능적 요구사항은 그 충족도(Conformance)를 고려하여 최적 서비스를 선택하여야 한다. 즉, 각 후보(Candidate) 서비스의 QoS, SLA 값들과 요청된 서비스의 QoS, SLA값들을 비교하여, 각 후보 서비스의 충족도를 계산한 후, 가장 높은 충족도를 가진 서비스를 선택한다. 사용자는 선택된 서비스에 실제 위치에 접근, 사용하기위해 서비스 Endpoint를 습득해야하며, 이는 선택된 서비스의 메타정보에서 서비스의 Endpoint에 대한 정보를 획득하는 과정으로 이루어진다. 이 단계에서 획득된 Service Endpoint를 이용하여 다음 단계에서는 서비스 클라이언트 클라이언트에서 사용가능한 객체을 생성한다. 다음의 코드는 Endpoint를 가져오기 위해서 사용되는 UDDI API 메소드 호출 순서를 나타낸다.

```
// Get Bisiness Serice binding template
BindingTemplate bt = (BindingTemplate)b_vector.get(1);

AccessPoint ap = bt.getAccessPoint();

// Get Endpoint Address
ENDPOINT_URL = ap.getText();
```

클라이언트가 선택한 서비스의 Endpoint 획득을 위해, 해당 서비스의 Binding Template을 먼저 획득한다. Binding Template은 Service Registry에 등록된 서비스의 메타정보와 실제 서비스 간의 연결정보들을 포함한다. AccessPoint는 서비스의 실제 위치 주소를 가리키며, getAccessPoint() 메소드를 통하여 획득된다. 획득된 AccessPoint는 Service Endpoint와 동일한 값을 포함하므로, 해당 값을 Endpoint로 반환한다.

3.2 단계 2. 클라이언트 프로그램 작성

클라이언트 프로그램 작성 단계에서는 이전 단계에서 획



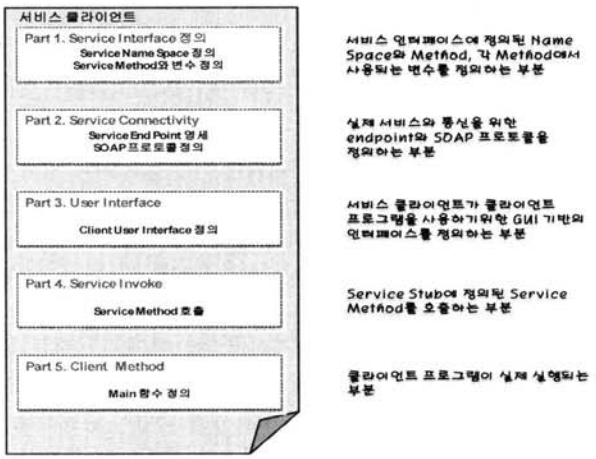
(그림 3) 클라이언트 프로그램 작성 단계의 세가지 활동들

득된 Service Endpoint를 이용하여 선택된 서비스들에 접근, 실행하는 클라이언트 프로그램을 설계, 구현한다. 선택된 서비스가 복수 개일 경우 서비스 사용자는 BPEL을 이용하여 선택된 서비스들 간의 조합(Composition)을 수행한다. 단계 3은 (그림 3)과 같이 3개의 활동들로 구성된다.

활동 1은 이전 단계에서 선택된 서비스들을 이용하는 서비스 클라이언트 프로그램을 설계하는 과정이다. 이 과정은 선택된 서비스들의 조합과 조합된 서비스를 이용하는 실제 클라이언트 프로그램을 설계하는 단계로 이루어진다. Service Registry에서 검색/선택된 서비스들은 블랙박스 형태로 내부 로직이나 구현 부분을 사용자의 입장에서는 접근할 수 없다. 따라서 서비스의 조합 과정에서 Endpoint를 이용한 서비스 참조를 통하여 해당 서비스가 업데이트되어도 클라이언트 프로그램의 내부 구조는 영향을 받지 않는다.

서비스 조합 설계는 사용자가 요구하는 서비스가 둘 이상의 서비스들로 이루어졌을 때 수행된다. SOA에서 조합서비스(Composite Service)는 BPEL을 이용하여 하나의 서비스에서 둘 이상의 서비스를 호출하는 방식으로 이루어진다. 따라서 조합서비스를 구성하기 위해서는 비지니스 프로세스(Business Process, BP)를 설계해야 한다. 비지니스 프로세스 설계는 다음과 같은 2가지 단계로 이루어진다; 첫째, TSRS를 분석하여 비지니스 로직을 도출한다. 비즈니스 로직은 서비스가 제공하는 기능을 제공하기위한 서비스 내부 구조를 나타내며, 내부에 호출되는 서비스들의 순서와 데이터들 간의 맵핑 관계를 포함한다. 둘째, 도출된 비지니스 로직을 기반으로 비지니스 프로세스를 모델링한다. 비지니스 프로세스 모델링(Business Process Modeling, BPM)은 BP를 모델링하기 위한 SOA 표준으로써, BP 설계에 필요한 표기법들을 제공한다. 이 과정을 통하여 목표 서비스 디자인 명세서(Target Service Design Specification, TSDS)가 산출된다.

클라이언트 프로그램 설계는 설계된 BP 혹은 완전한 하나의 서비스를 사용하는 서비스 사용자용 프로그램을 설계하는 과정이다. 클라이언트 프로그램은 사용되는 서비스가 둘 이상 일 때는 BP를, 하나일 때는 서비스 자체를 호출하여 사용하기 때문에 항상 하나의 서비스를 호출한다. 서비스 클라이언트 프로그램은 서비스 사용자가 식별된 서비스를 이용할 수 있도록 서비스가 제공하는 기능성, 각 기능별 요구되는 속성값, 서비스 사용자의 입력에 대한 반환값과 같은 내용을 제공해야하며, 서비스 인터페이스와 서비스/클라이언트 사이의 연결을 정의해야 한다. 클라이언트 프로그램의 설계는 객체지향의 설계 기법을 이용하여 클래스 다이어그램 등의 UML 표기법으로 표현되고, 설계된 내용은 서비스 클라



(그림 4) 서비스 클라이언트 프로그램 구조

이언트 디자인 명세서(Service Client Design Specification, SCDS)로 문서화된다.

활동 2는 TSDS와 SCDS에 설계된 내용을 기반으로 BP와 실제 서비스 클라이언트를 구현하는 부분이다. 비지니스 프로세스 모델링에 기술된 설계 모델은 OASIS BPEL에 따라 작성이 되고, Service Endpoint를 통하여 외부 서비스들을 호출한다. 작성된 BP는 WSDL 기반의 BPEL 인터페이스를 통하여 하나의 서비스처럼 외부에서 접근이 가능하다.

본 논문에서 제시하는 서비스 클라이언트 프로그램은 서비스 인터페이스를 통하여 서비스와 연결되므로 서비스의 내부 로직의 변화로부터 자유롭고, (그림 4)의 5가지 요소들로 구성된다.

Part 1은 서비스의 메소드와 변수를 정의하는 부분에서는 클라이언트가 사용하는 서비스의 메소드와 변수에 대하여 정의한다. 실제 서비스에서 제공하는 메소드는 클라이언트에서 사용할 수 있도록 정의되어야 하며, 서비스의 Namespace와 메소드에서 사용되는 변수들의 속성, 변수명 등이 정의된다. 예를 들면, 5장의 사례연구에서 서비스의 SearchHotel 메소드는 다음과 같이 클라이언트 내부에 정의된다.

```
@WebMethod
    @WebResult(name = "string", targetNamespace = "http://www.openuri.org/", partName = "Body")
    public String searchHotel(
        @WebParam(name = "string", targetNamespace = "http://www.w3.org/2001/XMLSchema", partName = "city")
        String city,
        @WebParam(name = "string", targetNamespace = "http://www.w3.org/2001/XMLSchema", partName = "rating")
        String rating);
```

Part 2는 서비스와의 메시지 통신을 정의하는 부분에서는 클라이언트와 해당 서비스간의 상호작용을위해 Service Endpoint와 전송 프로토콜을 정의한다. Service Endpoint는 실제 서비스가 배치되어있는 곳의 주소를 가르키는 값으로 클라이언트와 서비스간의 상호작용을 위해 반드시 정의되어야한다. 또한 클라이언트와 서비스간의 통신을 위해 상호간 표준 전송 프로토콜이 정의되어야한다. 따라서 클라이언트에서 서비스와 통신에 사용하는 SOAP 프로토콜을 다음과 같이 정의한다[11].

```
@WebService(name = "ProcessSoap", targetNamespace = "http://www.openuri.org/")
```

Part 3는 클라이언트와의 인터페이스를 정의한다. 클라이언트의 인터페이스는 서비스 인터페이스를 참조하여 서비스가 제공하는 기능을 실행하기위해 필요한 변수와 속성값 등을 클라이언트가 처리가능한 형태로 제공하고, 서비스 실행 결과를 출력한다. 클라이언트 유저 인터페이스는 단순히 클라이언트가 입력하는 데이터와 클라이언트에 정의된 변수를 매핑하는 역할을 수행하지만, 클라이언트의 관점에서 얼마나 편리하게 서비스를 이용할 수 있는가에 대한 측면이 중요하다.

Part 4는 Part 1에서 정의된 서비스의 메소드를 클라이언트에서 사용하기위해 실제 요청하는 부분이다. 클라이언트에서 실제 서비스 메소드의 사용은 미리 정의된 서비스 메소드와 속성 등에 대한 정의를 이용하여 Service Endpoint에 위치한 실제 서비스와 SOAP 프로토콜을 이용한 메시지 통신의 과정으로 이루어진다.

```
public String Search(String aCity, int aRating){
        String result = "";
        String outputMessage = "";
        ......
        org.openuri.www.SearchHotelLocator service = new org.openuri.www.SearchHotelLocator();
        org.openuri.www.SearchHotelSoap port = service.getSearchHotelSoap();

        result = port.clientRequestwithReturn(aCity, aRating);
        ......
        Return result;
}
```

Part 5는 이전 부분에서 정의된 서비스 인터페이스와 메소드 등을 이용하여 서비스를 실행하는 클라이언트의 메인 함수를 정의한다.

활동 3은 작성된 서비스 클라이언트를 테스팅하는 과정이다. 클라이언트 테스팅은 사용자가 입력하는 데이터에 따른 예상되는 결과와 실제 결과를 비교하는 과정으로 이루어지며, 이 때 두 결과 사이에 차이가 발생하였을 시에는 활동 2로 돌아가서 잘못된 부분을 수정한 후 다시 테스트를 진행한다. 테스트 과정에서는 단순히 실행 결과값만을 비교하는 것이 아닌 TSRS에 명시된 QoS와 SLA의 준수 여부를 확인하는 것도 중요한 비교 기준이 된다.

3.3 단계 3. 서비스 클라이언트 실행

서비스 실행 단계에서는 작성된 서비스 클라이언트 프로그램을 이용하여, 선택된 서비스를 실행한다. 클라이언트 프로그램 작성에 사용된 언어를 지원하는 미들웨어나 시스템 소프트웨어 상에서 실행하며, 서비스 실행 결과를 출력하는 과정이다.

서버측에서는 서비스 실행을 위해 첫째, 제공될 서비스를 개발하고, 둘째, 개발한 서비스는 서비스 사용자가 이용가능하도록 UDDI에 검색에 필요한 정보와 함께 등록한다. 서비스 작성자는 다양한 환경에서 서비스가 실행가능하도록 WSDL, SOAP, BPEL 등의 웹 서비스 표준 기술을 준수하여 서비스를 개발해야 하며, UDDI 등록시 함께 제공되는 정보들도 서비스 사용자의 검색을 위해 상세히 기술되어야한다.

서비스 클라이언트 프로그램에서는 UDDI 레지스트리에서 제공하는 각 실행환경에 따른 API를 이용하여 UDDI 검색이 가능한 클라이언트를 만들고, 이를 이용하여 UDDI에 등록된 서비스들 중 최적의 서비스를 검색한다. 검색된 서비스의 Endpoint를 이용하여 클라이언트 프로그램을 작성하고, 실행한다. 실행된 클라이언트는 실제 서비스와 연결되고, 연결된 서비스가 요구하는 데이터를 사용자가 입력한다. 입력된 데이터는 해당 서비스 시스템으로 전송되고, 시스템 내부에서 처리된 후 결과 데이터를 클라이언트로 반환하다.

4. 사례 연구

본 논문에서 제시된 프로세스 모델을 호텔 검색 및 예약 서비스에 적용하여, 제안한 프로세스가 어떻게 적용되는지를 설명한다. 사례연구에 사용되는 도메인 업무 흐름을 요약한다.

사례연구는 호텔서비스 도메인의 검색 및 예약 서비스를 대상으로 하며, 서비스가 제공하는 기능은 다음과 같다; 첫째, 서비스 사용자가 여행하는 지역에 위치한 호텔들을 검색하여 예약 가능한 리스트를 반환한다. 서비스 사용자는 제공되는 예약가능 호텔 리스트 중에서 이용하고자 하는 호텔 선택이 가능하다. 둘째, 서비스 사용자가 선택한 호텔의 예약 서비스 시스템과 연결하여 예약을 위해 입력해야할 정보를 서비스 사용자로부터 입력받아서 해당 예약 서비스 시스템에 전송한다. 전송된 정보들은 시스템 내에서 처리되어 이상이 없는 경우 예약이 수행되며, 호텔 예약 아이디를 결과값으로 반환하다.

4.1 서비스 검색

서비스 검색 단계에서 클라이언트의 요구사항에 맞는 서비스를 검색하기위해 Service Registry에 등록된 서비스 중 알맞은 도메인에 있는 서비스들을 검색하여, 각 서비스의 메타정보를 기반으로 최적의 서비스를 선택한다.

본 사례연구에서 Service Registry는 UDDI를 말하며, 등록된 서비스를 검색하기위하여 해당 UDDI에서 제공하는 검색관련 API를 사용하였다. UDDI는 검색관련 API로 findBusiness(), findService() 등과 같은 총 9개의 메소드들을 지원하며, 본 장에서는 다음의 코드와 같이 사용이 된다.

서비스 검색

서비스 검색 결과

(그림 5) UDDI API를 이용한 서비스 검색 및 결과

```
public String findingHotelService(String keyword) throws UDDIException,
TransportException {
        Vector names = new Vector();
        names.addElement(new Name(keyword));

        // find Services
        ServiceList list = proxy.find_service(null, names, null, null, null,0);
        ServiceInfos sinfos = list.getServiceInfos();

        Vector service_vector = sinfos.getServiceInfoVector();
    ...
        // Get Bisiness Serice binding template
        BindingTemplate bt = (BindingTemplate) b_vector.get(1);

        AccessPoint ap = bt.getAccessPoint();

        // Get Endpoint Address
        ENDPOINT_URL = ap.getText();
}
```

클라이언트에 정의된 findingHotelService() 메소드는 UDDI API를 이용하여 검색된 서비스의 Endpoint를 반환한다. findService()는 서비스 검색을 위해 제공되는 API 로써 서비스 키, 이름, … 등을 이용하여 실행되며, 사례연구에서는 서비스의 이름을 이용하여 서비스 검색을 실행하고, Service Endpoint 값을 반환한다. (그림 5)는 UDDI API를 이용한 서비스의 검색과 그 결과를 보여준다.

서비스 검색을 위하여 키워드를 이용하여 UDDI를 검색하고, 서비스 Endpoint를 획득한다. 예, 키워드 = Hotel Service, 서비스 검색 결과 = Service Endpoint를 획득한다.

4.2 클라이언트 프로그램 작성

클라이언트 프로그램 작성 단계에서는 선택된 서비스의 WSDL 인터페이스에 명시된 정보를 이용하여 실제 서비스를 실행하기위한 클라이언트 프로그램 개발을 위한 프로세스를 포함한다.

4.2.1 WSDL 인터페이스

서비스 검색 단계에서 획득된 Service Endpoint를 이용하여 서비스가 배치된 실제 주소로 접근하여, 서비스 WSDL 인터페이스에 접근한다. 서비스 인터페이스는 클라이언트가 서비스를 실행하는데 필요한 메소드와 매개변수, 속성 등의 정보를 포함하고있고, 이를 이용하여 클라이언트 프로그램이 작성된다. 다음은 클라이언트에서 사용되는 정보를 포함하는 WSDL 인터페이스의 중심 코드를 보여준다.

```
<s:element name="clientRequestwithReturn">
  <s:complexType>
    <s:sequence>
      <s:element name = "city" type = "s:string" minOccurs = "0"/>
      <s:element name="rating" type="s:int"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="clientRequestwithReturnResponse">
  <s:complexType>
    <s:sequence>
      <s:element name = "clientRequestwithReturnResult" type =
"s:string" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
```

WSDL 인터페이스의 내부 코드를 살펴보면, <s:element> 부분에서는 서비스가 제공하는 메소드와 해당 메소드에서 사용하는 매개변수를 정의한다. 예를들면, clientRequestwithReturn 메소드와 String 형태의 city, int 형태의 rating 매개변수가 정의되어있다. 클라이언트가 이 메소드에 city와 rating 형태의 입력값들을 입력하면 서비스는 String 형태의 리턴값을 반환한다

```
<wsdl:message name="clientRequestwithReturnSoapIn">
  <wsdl:part name="parameters" element = "s0 : clientRequest with
Return"/>
</wsdl:message>
<wsdl:message name="clientRequestwithReturnSoapOut">
  <wsdl:part name="parameters" element="s0:clientRequest with
ReturnResponse"/>
</wsdl:message>
```

<wsdl:message> 부분은 다수의 입/출력 매개변수들을 하나의 집합으로 정의한다. clientRequestwithReturnSoapIn 과 clientRequestwithReturnSoapOut 가 입력과 출력 매개변수들의 집합으로 정의되었다.

```
<wsdl:portType name="SearchHotelSoap">
  <wsdl:operation name="clientRequestwithReturn">
    <wsdl:documentation/>
    <wsdl:input message="s0:clientRequestwithReturnSoapIn"/>
    <wsdl:output message="s0:clientRequestwithReturnSoapOut"/>
  </wsdl:operation>
</wsdl:portType>
```

<wsdl:portType> 에서는 앞서 정의된 매개변수들의 집합을 서비스에서 제공하는 메소드 단위로 다시 하나의 집합으로 정의한다. searchHotelSoap는 clientRequestwithReturnSoapIn과 clientRequestwithReturnSoapOut으로 이루어진 메소드를 나타낸다.

```
<wsdl:binding name = "SearchHotelSoap" type = "s0:SearchHotel
Soap">
    <soap:binding transport = "http://schemas.xmlsoap.org/soap
/http" style="document"/>
    <wsdl:operation name="clientRequestwithReturn">
      <soap:operation soapAction="http://www.openuri.org/client
RequestwithReturn" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

<wsdl:binding> 에서는 정의된 메소드를 실제 서비스에서 제공하는 메소드와 매칭시키는 역할을 수행한다.

```
<wsdl:service name="SearchHotel">
    <wsdl:documentation/>
    <wsdl:port name="SearchHotelSoap" binding = "s0:Search
HotelSoap">
      <soap:address location = "http://FF:7051/Reservation_Web/
processes/SearchHotel.jpd"/>
    </wsdl:port>
  </wsdl:service>
```

<wsdl:service> 는 실제 서비스와 WSDL 인터페이스 내에 정의된 portType들 간의 연결을 정의한다.

WSDL 인터페이스에서 실제 서비스가 제공하는 메소드와 매개변수 등과 연결을 정의하였다면, 클라이언트에서는 WSDL 인터페이스에 정의된 정보들을 이용하여 WSDL 인터페이스와 연결하여, 서비스가 제공하는 메소드를 사용한다.

4.2.2 서비스 인터페이스 정의

서비스 클라이언트 프로그램에는 서비스의 WSDL 인터페이스와 통신하기위한 서비스 인터페이스 부분이 정의되어야 하며, 사례연구에서는 JAVA의 JAX-WS를 이용하여 클라이언트 내에 서비스 인터페이스와 서비스 Endpoint를 정의한다. 다음은 서비스와의 연결설정을 위한 SearchHotelLocator 클래스와 SearchHotelSoap 인터페이스의 주요 코드를 보여준다.

```
public class SearchHotelLocator extends org.apache.axis.client.
Service implements org.openuri.www.SearchHotel {

    // Use to get a proxy class for SearchHotelSoap
  private final java.lang.String SearchHotelSoap_address = "http://
203.253.23.244:7051/Reservation_Web/processes/SearchHotel.jpd";

    // The WSDD service name defaults to the port name.
    private java.lang.String SearchHotelSoapWSDDServiceName
= "SearchHotelSoap";

  public org.openuri.www.SearchHotelSoap getSearchHotelSoap()
throws javax.xml.rpc.ServiceException {
      java.net.URL endpoint;
      try {
        endpoint = new java.net.URL(SearchHotelSoap_address);
      }
      catch (java.net.MalformedURLException e) {
        throw new javax.xml.rpc.ServiceException(e);
      }
      return getSearchHotelSoap(endpoint);
    }
```

SearchHotelLocator 클래스에서는 서비스와의 연결을 위해 서비스 Endpoint를 사용하며, Endpoint 주소를 이용하여 JAX-WS에서 정의된 SOAP 프로토콜을 이용한 통신을 설정한다.

```
public interface SearchHotelSoap extends java.rmi.Remote {
    public java.lang.String clientRequestwithReturn(java.lang.
String city, int rating) throws java.rmi.RemoteException;
}
```

SearchHotelSoap 인터페이스는 실제 서비스에서 제공하는 메소드와 연결되는 웹메소드의 인터페이스를 정의하는 부분이며, 클라이언트에서는 정의된 웹메소드의 인터페이스의 정보를 이용하여 실제 서비스가 제공하는 메소드를 사용한다.

4.2.3 서비스 메소드 호출

서비스 클라이언트에서 실제 서비스가 제공하는 메소드들을 이용위해 서비스 메소드를 호출해야하며, 앞서 정의된 서비스 인터페이스에서 정의된 메소드를 사용한다. 다음은 클라이언트에서 서비스 메소드를 호출하는 코드를 보여준다.

```
org.openuri.www.SearchHotelLocator service = new org.
openuri.www.SearchHotelLocator();
org.openuri.www.SearchHotelSoap port = service.getSearch
HotelSoap();
```

코드를 살펴보면, 실제 서비스의 위치와 서비스와의 연결에 대해 정의된 SeachHotelLocator에 대한 인스턴스를 생성하고, 이를 서비스 인터페이스가 정의된 SearchHotelSoap의 인스턴스에 연결을 한다. 클라이언트에서는 이렇게 생성된 인스턴스를 이용하여 서비스에 접근, 이용할 수 있다.

4.3 서비스 실행

이번 단계에서는 이전 단계에서 작성된 클라이언트 프로그램을 이용하여, 실제 서비스를 구독 및 실행한다. 클라이언트 프로그램을 실행하게되면, (그림 6)와 같이 설계/구현된 사용자 인터페이스에 따라 나타난다. 그림에서 보이는 클라이언트 프로그램은 검색된 서비스에 포함된 두 가지 하위 서비스, 즉 Search Hotel, Make Reservation 서비스들을 나타내고, 각각 해당 서비스 메소드를 호출하는 부분과 연결되어있다.

호텔 검색 및 예약 서비스를 이용하기위해, 첫 번째 예약 가능한 호텔을 검색할 필요가 있고, 이러한 기능은 하위 서비스인 Search Hotel 서비스에서 제공된다. 사례연구에서는 서울 지역에 위치한 5 등급의 예약 가능한 호텔을 멈검색하기위해 (그림 7)과 같이 City Name = Seoul, Rating = 5의 입력값을 이용하여 검색하고 예약 가능한 호텔리스트(Crown과 Sheraton)를 검색 결과로 받았다.

(그림 6) 클라이언트 프로그램의 User Interface

서비스 사용자는 검색된 호텔 리스트를 이용하여 Make Reservation 서비스를 실행할 수 있다. 사용자가 검색된 호텔 리스트 중에서 예약을 원하는 호텔을 선택하면 (그림 8)과 같이 예약을 위해 요구되는 데이터 입력을 위한 창으로 넘어가고, 입력된 데이터는 public ReservationajaID makeReservation (String hotelName, String username, String address, String contact, String eMail, Date sDate, Date eDate) 함수로 전송된다. 사용자의 호텔 예약에 대한 Reservation ID를 받는다.

5. 평가 및 결론

SOA는 여러 어플리케이션에서 사용할 수 있는 범용적인 서비스를 개발하고, 이를 재사용하여 어플리케이션을 신속하고 경제적으로 개발할 수 있는 기술이다. 따라서, 서비스 레지스트리에는 여러 종류의 많은 서비스들이 등록되어 있는 경우가 많은데, 서비스 사용자가 표준 검색 인터페이스인 UDDI API를 사용하여 필요한 서비스를 검색하고, 이들을 클라이언트 어플리케이션에 포함하여 실행한다. 본 논문에서는 서비스 사용자의 측면에서 보다 쉽고 정확하게 자신이 원하는 최적의 서비스를 식별하고, 이를 이용하는 클라이언트 프로그램을 위한 개발 프로세스를 제시하고, 사례연구를 통하여 이를 검증하였다.

제시된 프로세스는 세개의 단계와 세부 활동으로 구성되며, 각 활동별 세부 작업 지침을 제안하고, 관련된 예제를 보였다. 특히, 이 프로세스는 WSDL, SOAP, UDDI등의 SOA

(그림 7) Search Hotel 서비스 실행 및 결과

(그림 8) Make Reservation 서비스 실행 및 결과

표준들을 기반으로 제안하였고, 실용적인 적용을 위한 세부 지침을 제공하였다. 즉, 서비스 사용자가 자신이 원하는 서비스를 식별, 클라이언트 프로그램을 구현, 실행하는데 필요한 전과정의 세부 지침을 제공하였다. 제시된 개발 프로세스를 활용하여, 서비스 클라이언트 프로그램을 개발하면, 개발 시간과 비용이 절약되고 고품질의 클라이언트 프로그램 작성이 가능하다.

참 고 문 헌

[1] Arsanjani, A., "Service-Oriented Modeling and Architecture (SOMA)," IBM DeveloperWorks, 2004, http://www-128.ibm.com/developerworks/webservices/library /ws-soa-design1 (accessed September 26, 2007).

[2] Erl, T., Chapter 6 to Chapter 12 of *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, 2005.

[3] Clement, L, Hately, A., Riegen, C., and Rogers, T. eds., *UDDI Version 3.0.2*, UDDI Spec Technical Committee Draft, OASIS, Oct. 2004, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm (accessed September 26, 2007).

[4] J. Colgrave and K. Januszewski, "Using WSDL in a UDDI Registry, Version 2.0.2," *Technical Report.*, OASIS UDDI Spec TC, 2004.

[5] Booth, D. and Kevin, C. eds., *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*, W3C Recommendation, W3C, 26 June, 2007, http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/ (accessed September 26, 2007).

[6] Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C.K., Khalaf, R., König, D., Marin, M., Mehta, V., Thatte, S., Rijn, D., Yendluri, P., and Yiu, A. eds., *Web Services Business Process Execution Language Version 2.0*, OASIS Standard, OASIS, 11 April, 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (accessed September 26, 2007).

[7] Gudgin, M., et al., *Simple Object Access Protocol Version 1.2 Part 1:Messaging Framework (Second Edition)*, W3C Recommendation, W3C, 27 April, 2007, http://www.w3.org/TR/2007/REC-soap12-part1-20070427/(accessed September 26, 2007).

[8] Papazog, M. and Heuvel, W., "Service-Oriented Design and Development Methodology," *International Journal of Web Engineering and Technology*, InderScience Publisher, Vol.2, No.4, pp.412-442, 2006.

[9] Menasce, D.A., "Mapping service-level agreements in distributed applications," *IEEE Internet Computing*, Vol.8, No.5, pp. 100-102, 2007

[10] Bellwood, T., *UDDI Version 2.04 API Specification*, OASIS, 19th July 2002, http://uddi.org/pubs/ProgrammersAPI_v2.htm

[11] Jagannadham, D., Ramachandran, V., Kumar, H.N., "Java2 distributed application development(Socket, RMI, Servlet, CORBA) approaches, XML-RPC and web services functional analysis and performance comparison," *Communications and Information Technologies*(ISCIT '07), International Symposium, pp.1337-1342, Oct. 17-19, 2007.

이 재 유

e-mail : jylee81@otlab.ssu.ac.kr

2007년 홍익대학교 컴퓨터정보통신학과 (공학사)

2007년~현 재 숭실대학교 컴퓨터학과 석사과정

관심분야 : 소프트웨어 공학(Software Engineering), 서비스 지향 아키텍처(SOA)

김 수 동

e-mail : sdkim@ssu.ac.kr

1984년 Northeast Missouri State University 전산학(학사)

1988년 / 1991년 The University of Iowa 전산학(석사/박사)

1991년~1993년 한국통신 연구개발단 선임연구원

1994년~1995년 현대전자 소프트웨어연구소 책임연구원

1995년 9월~현 재 숭실대학교 컴퓨터학부 교수

관심분야 : 서비스 지향 아키텍처(SOA), 객체지향 S/W공학, 컴포넌트 기반 개발 (CBD), 소프트웨어 아키텍처