한국시뮬레이션학회 논문지
Vol. 16, No. 1, pp. 49-57 (2007. 3)

서비스 지향 아키텍처를 기반으로 한 웹서비스 시스템 모델링

이성규[1] · 진찬욱[2] · 김태석[1†]

System Modeling for Web Service based on Service-Oriented Architecture

Seong-Kyu Lee · Chan-Uk Jin · Tai-Suk Kim

ABSTRACT

Service-Oriented Architecture(SOA) is improving rapidly in IT Environment. Enterprise companies interest in implementation infrastructure based on SOA to adapt quick changes of large and complex distributed environment. SOA is a component model that inter-relates the different functional units of an application, called services, through interfaces and contacts between theses services. In this paper, we studied the concept of the relationship between SOA and standard web service core and therefore, we model the web service based on SOA. We implemented the application and business service architecture using web service that include the XML and SOAP. We confirmed that how the each SOA characters like interoperability, reusability, scalability and flexible business process adapted to web service and present a web service modeling that is maintained the neutrality using loose service coupling through the method of service model process and web service architecture designing methodology based on SOA.

Key words : Service-Oriented architecture, Web service, Business service, XML, SOAP, UDDI

요 약

서비스 지향 아키텍처(SOA)는 최근 IT환경에서 급격한 성장을 하고 있다. 거대하고 복잡한 분산 환경에서 재빠른 변화에 적용하기위해 SOA를 기반으로 한 인프라 구축을 기업에서 많은 관심을 가지게 되었기 때문이다. SOA는 인터페이스와 서비스간의 계약을 통하여 서로 다른 기능을 단위로 하는 응용프로그램이 상호 연관성을 가지는 컴포넌트 모델이다. 본 논문에서는 SOA와 핵심 웹서비스 표준에 관계된 개념을 웹서비스에 적용하기 위한 아키텍처를 설계하고, 그 내용에 따라 SOA를 기본으로 한 웹 서비스 시스템을 모델링 한다. 웹서비스는 XML과 SOAP를 기본으로 도입하여, 응용프로그램과 비즈니스 서비스의 설계를 구현한다. 이렇게 설계된 SOA기반의 웹서비스를 통하여 상호 운영성, 재 사용성, 확장성 및 유연한 비즈니스 프로세스 처리와 같은 SOA의 각 특징이 어떻게 적용되는지 확인하고, 서비스 모델 프로세스에 대한 방법과 SOA기반의 웹서비스의 아키텍처 설계방법을 통하여, 서비스 간의 느슨한 결합(Loose Coupling)으로 중립성을 유지하는 웹 서비스 모델링을 제시한다.

주요어 : 서비스 지향 아키텍처(SOA), 웹 서비스, 비즈니스 서비스, XML, SOAP, UDDI

1. 서 론

IT환경의 발전으로 1980년대에 들어서면서 기업은 업무에 IT를 적극적으로 적용하게 되었고, 각 환경에 맞는 응용프로그램 개발이 활발해지기 시작 하였다. 그 당시의 소프트웨어 개발 방법 형태는 다형성(Polymorphism)이나 캡슐화(Encapsulation)등을 기준으로 하였으나, 점점 처리해야 할 로직과 프로세스가 복잡해지고, 주고받아야 할 데이터의 특성이 중요하게 됨에 따라 1990년대 이후에는 개발 방법론의 형태로 인터페이스 기반(Interface-based)의 컴포넌트 개발 방법론이 주류를 이루게 되었다. 하지만 컴포넌트 기반의 개발 방법론은 현재와 같은 Web 기반의 IT환경에서 기업의 다양한 비즈니스 업무를 처리하기 위해서는 많은 노력과 시간이 필요하게 되었다. 또

2006년 12월 31일 접수, 2007년 3월 21일 채택
[1] 동의대학교 컴퓨터소프트웨어공학과
[2] 동의대학교 인터넷응용공학과
주 저 자 : 이성규
교신저자 : 김태석
E-mail; tskim@deu.ac.kr

한 업무에 맞게 잘 설계된 프로그램도 처리해야할 서비스의 내용이 변경되면 수정이 어렵거나 상호 연결된 다른 프로그램에도 영향을 미치게 된다. 서비스 지향 아키텍처(Service-Oriented Architecture : SOA)는 이러한 부분을 좀 더 쉽게 접근하고 해결하기 위해 최근 Microsoft, IBM, Sun Microsystem 및 Oracle과 같은 대표적인 IT업계에서 집중적으로 연구하고 있다. 사실 몇 년 전 부터 CORBA나 DCOM 등의 분산 객체 기술에서 SOA의 기본 개념이 사용되었으나 기술적인 미성숙과 벤더 마다 독자적인 데이터 포맷과 전송 프로토콜, 인터페이스 정의 언어를 사용하는 공개 표준의 부재 문제로 인하여 서로 다른 벤더 간의 상호 운용이 어려워 SOA는 주목 받지 못하였다. 하지만 최근 XML 기반의 웹서비스 기술이 등장하면서 SOA는 새롭게 조명되고 있으며, SOAP (Simple Object Access Protocol), WSDL (Web Services Discription Language), 그리고 UDDI (Universal Description, Discovery and Integration)등과 같은 공개표준이 정의되어 서로 다른 벤더 간의 상호 운용이 용이하게 됨에 따라 웹서비스는 최상의 SOA의 구현기술로 자리 잡고 있다[2].

본 논문에서는 서비스 지향 아키텍처를 적용하여 웹서비스를 개발하기 위한 모델을 제시한다. SOA의 기본 개념인 자치적인 서비스(autonomous service)가 메시지 기반(message-based)으로 정책(Policy)을 통해 바인딩 하여 서로 커뮤니케이션하도록 웹서비스를 설계하는 방법과 서비스 소비자와 공급자 측면에서 비즈니스 프로세스가 어떻게 구현되는가를 설명한다. 본 논문의 구성은 다음과 같다. 2장에서 서비스 지향 아키텍처의 구성과 구성요소들 사이의 역할에 대해서 살펴보고 SOA구현을 위한 웹서비스를 정의한다. 3장에서는 서비스 지향 애플리케이션의 적용시 변화된 모습과 비즈니스 서비스의 효과에 대해 살펴보고, 그에 따라 웹 서비스를 어떻게 구성할 것인가를 설명한다. 4장에서는 웹서비스 시스템 구현 방법과 SOA의 기본 구성이 되는 서비스 프로세스 모델 방법을 설명하고, 웹서비스가 SOA를 기반으로 할 때의 아키텍처 설계 방법을 제시한다. 마지막 5장에서 SOA기반의 웹서비스 시스템에 대한 평가와 결론 및 향후 진행해야할 연구 방향을 설명한다.

2. SOA(Service-Oriented Architecture)

2.1 SOA 구성요소

서비스 지향 아키텍처는 로직의 개별 단위들이 서로 고립되지 않고 상호 작용하면서도 자율적으로 존재할 수 있도록 한다. 로직 단위는 원칙을 따라 공통성과 표준성을 유지하며 독립적으로 구성되며, SOA에서는 이러한 로직 단위를 서비스라고 한다.

SOA는 이러한 서비스들을 기반으로 하는 소프트웨어 아키텍처로 애플리케이션의 기능들을 사용자(Consumer)에게 적합한 크기로 공개한 서비스들의 집합을 제공하고 정책(Policy) 또는 프레임워크(Framework)를 통해 바인딩 가능하도록 구현한다. 이 때 서비스는 단일한 표준 기반의 인터페이스 형태를 통하여 구현하며 독립적으로 추상화되고, 호출(Invoke), 공개(Publish), 발견(Discovery)과 같은 오퍼레이션을 수행 한다. 즉, SOA란 서비스로 정의되는 분할(Decomposition)된 애플리케이션 조각들을 단위로 느슨하게 연결해 하나의 완성된 애플리케이션으로 만드는 아키텍처이다[3].

SOA의 정의에서 언급된 각 오퍼레이션은 서비스 소비자(Service Consumer), 서비스 레지스트리(Service Registry), 그리고 서비스 공급자(Service Provider)와 같이 3가지 구성요소로 이루어져 있으며, 각 구성요소 역할은 아래 그림 1과 같은 상호 관계를 갖는다.

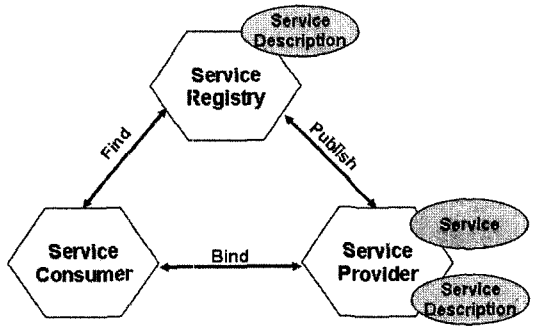


그림 1. SOA구성 요소와 상호 관계

- 서비스 공급자(Service Provider): 서비스 공급자는 서비스 명세서를 만들고 하나 이상의 서비스 레지스트리에 서비스 명세서를 등록하고 하나 이상의 서비스 요청자로부터 서비스 호출 메시지를 받을 책임이 있다.
- 서비스 소비자(Service Consumer): 서비스 소비자는 하나 이상의 서비스 레지스트리에 등록되어 있는 서비스 명세서를 찾고(Find), 서비스 공급자에 의해 호스팅 되는 웹서비스를 호출하거나 연결(Bind)하기 위하여 서비스 명세서를 사용할 책임이 있다.
- 서비스 레지스트리(Service Registry): 서비스 공급

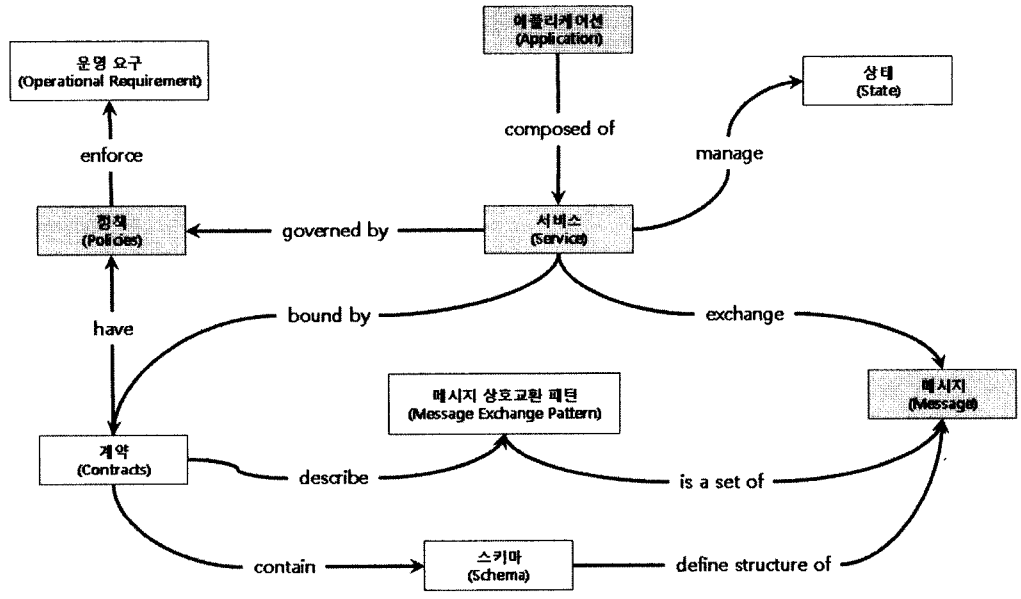


그림 2. SOA기반 서비스, 정책, 및 메시지의 상호 관계

자에 의해 등록된 서비스 명세서를 홍보하고 서비스 레지스트리 안에 보관되어 있는 명세서의 집합에서 서비스 요청자가 원하는 것을 찾을 수 있게 할 의무가 있다.

2.2 SOA의 특징과 구조

앞서 설명된 내용과 같이 SOA의 정의는 실제 아키텍처로서 구현되기 위해 다음과 같은 특성을 만족해야 한다.

- 느슨한 결합관계 : 비즈니스와 애플리케이션 기술은 서로를 인지할 수 있는 인터페이스만 고려한 나머지 기술 관련된 변화는 독립적인 서비스 로직으로 진화해야 한다.
- 조합성 : 서비스 조합성은 비즈니스 프로세스의 유연한 적용을 제공한다. 조합성은 개별적인 솔루션이 하나의 요구속성을 지원하는 한 독립적인 발전에 대하여 서로 지속적인 상호 작용을 할 수 있다.
- 상호 운영성 : 공개표준을 사용하여 다양한 벤더 환경에 대하여 상호 운영성을 갖는 서비스가 되도록 설계원칙을 가져야 한다. 이는 플랫폼에 독립적으로 수행되어 모든 환경에서 호출이 가능하게 한다.
- 재사용성 : 서비스 지향의 원칙에 따라 설계되는 서비스는 지금 당장 재사용 요구사항이 존재하지 않더라도 향후에 재사용할 수 있어야 하며, 이는 예측하지 않은 재사용성 기회를 가져온다.
- 확장성 : 서비스 로직이 적절히 잘 분할되어 있고 적정한 수준의 인터페이스 크기를 가지고 있을 때, 서비스가 제공하는 기능의 범위는 인터페이스의 변경 없이 확장 할 수 있다.
- 발견성 : 서비스 레지스트리 혹은 디렉터리의 형태와 같은 저장소에 서비스 명세를 관리하여 어떤 환경에서든 쉽게 등록, 호출 될 수 있어야 한다.

SOA의 이러한 특징들은 서비스와 각 연계되는 요소들의 구성관계를 정의하는데 기본원칙이 된다. 비즈니스 와 서비스의 각 요소들은 SOA에 특징에 따라 상호 연결되며 이 연결은 각각의 의미에 따라 프로세스의 요구에 맞는 흐름을 갖게 된다. 예를 들어, 서비스들의 집합으로 구성된 애플리케이션은 각 서비스가 상태(State)를 관리하고 정책(Policy)에 의해 통제되며, 스키마(Schema)를 포함하고 있는 계약(Contract)에 구속된다. 여기서 스키마는 메시지의 구조를 정의하고 이 메시지는 서비스의 상호 교환 정보가 된다. 이러한 상호 관계를 도식화하면 그림 2와 같은 다이어그램 형태로 표현할 수 있다.

3. 서비스 지향 애플리케이션

3.1 SOA적용 애플리케이션

이전의 SOA구현 문제점은 CORBA (Common Object

Request Broker Architecture), DCOM (Distributed Component object mode) 또는 JAVA RMI(Remote Method Invocation)와 같이 서로 다른 플랫폼 상에서 애플리케이션을 생성하는 경우에 이들 애플리케이션의 통합이 어려웠다. 그러나 웹서비스를 이용하여 이러한 문제를 극복할 수 있다. 현재 소프트웨어 기술을 주도하는 대부분의 벤더들이 웹서비스라는 하나의 표준에 동의함으로써 상호 운영성이 확보되었기 때문이다.

SOA를 웹서비스에 적용하여 설계하기 위해서는 비즈니스 프로세스 관점에서 서비스를 구성하면 효과적인 설계가 가능하다. 서비스는 프로세스 관점에 따라 두 가지 형태로 구분할 수 있는데, 첫 번째는 자신의 클라이언트나 파트너 또는 다른 조직에게 노출하기를 원하는 비즈니스와 두 번째, 기업 내 여러 시스템에 분산되어 있는 비즈니스 서비스를 해체하여 여러 시스템에서 공유할 수 있는 공유 비즈니스 서비스로 통합하는 것이다. SOA를 이용하지 않는 일반적인 애플리케이션 개발에서는 비즈니스 요구에 맞는 모듈을 구성하고 해당 모듈을 여러 채널 시스템에서 반복 사용하고 있었으나 SOA를 적용하게 되면 공유 비즈니스는 서비스에 따라 시스템을 구분하여 여러 채널에서 따로 중복 사용하지 않고 한곳에서 재사용(Reuse)할 수 있다.

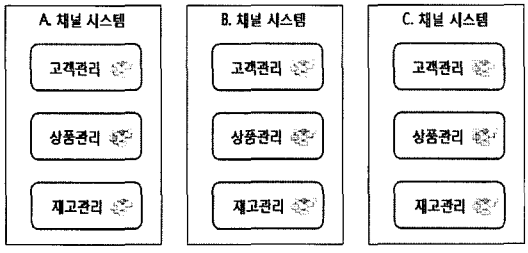


그림 3. SOA적용 이전 애플리케이션 모델

그림 3과 그림 4 은 SOA를 적용하기 전, 후에 따라 애플리케이션 모델이 어떻게 변화되는지를 나타낸다. 두 그림의 차이에서 보는 바와 같이 SOA를 기반으로 하는 애플리케이션은 비즈니스에 단일화된 작업 중심적인 뷰(Unified, Task-Oriented View)를 제공하고 불투명한 블랙박스 시스템에 비즈니스 투명성을 제공하는 하부 구조로 대체할 수 있음을 알 수 있으며 변화에 대하여 민첩하고, 유연하고, 효과적으로 대응할 수 있게 된다.

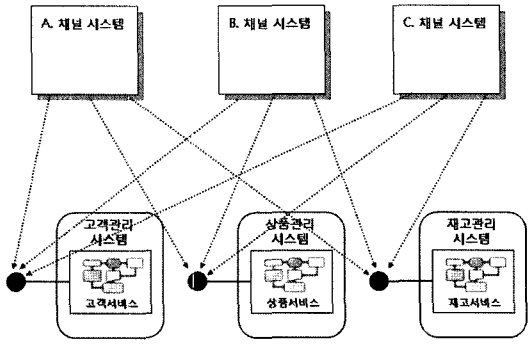


그림 4. SOA 적용 이후 애플리케이션 모델

3.2 비즈니스 서비스(Business Service)

SOA기반의 서비스는 모두 메시지(Message)를 통해 커뮤니케이션을 한다. 여기서 말하는 메시지란 하나의 서비스가 다른 서비스로 전송하는 정보의 단위로서 스키마(Schema)를 사용하여 구조화되어 전송되어야 하며 메시지를 이해하는데 필요한 모든 정보를 포함하거나 참조함으로써 개념적으로 스스로 충족될 수 있어야 한다. 이렇게 서비스 간 커뮤니케이션에 사용되는 메시징 모델은 사실 스키마를 통해 어렵지 않게 표현할 수 있다. 하지만 SOA 특성을 지원하고 서비스 지향 원칙을 준수하면서 기업이나 조직의 전산 환경시스템에 적용할 수 있는 수준으로 웹서비스 모델을 구현하는 것은 메시지의 요청과 전송을 처리해야 할 비즈니스 서비스를 얼마나 효과적으로 정의 하였는가에 따라 구현 여부가 결정된다.

비즈니스 서비스(Business Service)는 비즈니스 프로세스(Business Process)와 비즈니스 로직(Business Logic), 그리고 비즈니스 실체(Business Entity)등을 캡슐화하는 기본적인 구성단위로 사용한다. 그리고 각각의 비즈니스 서비스가 다른 비즈니스 서비스에 의존하지 않고도 독립적으로 해당 로직을 제어할 수 있으며 해당 로직을 제어하는데 필요한 정보를 스스로 포함하고 있는 자치성(Autonomous)은 SOA에서 사용되는 비즈니스 서비스가 가지고 있어야 할 중요한 특성이다.

비즈니스 서비스는 SOA를 적용하지 않는 일반적인 개발 아키텍처의 객체(Object)나 컴포넌트(Component)와 비교할 수 있다. 객체는 작은 입자(Fine-grained)를 이용한 밀접한 결합(Tightly-coupled)으로 비유되는 반면 비즈니스 서비스는 큰 입자(Coarse-grained)를 이용한 느슨한 결합(Loosely-coupled)으로 비유할 수 있다.

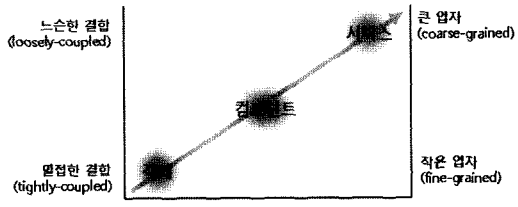


그림 5. 객체, 컴포넌트 및 서비스의 특성 비교

4. SOA기반의 웹서비스

4.1 SOA기반의 웹서비스 구조

최신 SOA는 웹서비스 기술을 바탕으로 발전하고 있다. 서비스 지향을 구축하기 위해 사용되는 웹서비스의 개념과 기술은 SOA 특징에 영향을 받기도 하고 주기도 한다. 그러나 웹서비스 구현 기술은 여러 기관에 의해 지속적으로 발전되고 있었기 때문에 획일화된 표준을 정의하기가 어려웠다. W3C는 웹서비스를 URI로 인식하는 소프트웨어 애플리케이션으로서, 서비스의 인터페이스와 바인딩 가능한 XML 결과로 나타낼 수 있고, 인터넷 기반의 프로토콜을 통하여 XML 기반 메시지를 사용하는 다른 소프트웨어 에이전트들과의 직접적인 상호 작용을 지원해야 한다고 정의하고 있다[1].

가트너 그룹은 웹서비스를 다음과 같이 정의한다. 웹서비스는 분산 컴퓨팅을 수행하기 위해 SOAP, WSDL, UDDI등의 표준 기술을 적용한 소프트웨어 컴포넌트이다. 이는 위험도가 높은 비즈니스 전략 수행에 상대적으로 위험도가 낮은 기술을 이용한다[4]. 또한, IT 분야 컨설팅 전문기관인 Ovum은 웹서비스를 위치나 구현 모델에 관계없이, 다른 프로그램이 발견할 수 있고, 호출할 수 있는 소프트웨어 시스템의 기능으로서 웹서비스는 구현이 아니라 인터페이스라고 정의한다.

이러한 웹서비스의 정의내용들과 앞서 설명한 SOA의 특징을 정리할 때, SOA를 기반으로 하는 웹서비스는 다음과 같은 특징을 가져야 한다.

표준체계로 정의되고 기술플랫폼으로 구현되어 추상화 수준이 높은(벤더 중립적) 형태로 존재한다.

웹서비스, 서비스 명세, 그리고 메시지를 포함하는 핵심 구축 단위이다.

커뮤니케이션 규약은 WSDL 기반의 서비스 명세를 중심으로 한다.

메시지 프레임워크는 SOAP의 기술과 개념으로 구성된다.

서비스 명세의 등록과 발견 아키텍처는 UDDI를 통해 구현된다.

메시징 패턴과 조합을 지원하는 잘 정의된 아키텍처를 필요로 한다.

SOA를 위한 웹서비스에서는 WSDL, UDDI, SOAP의 세 가지 핵심 기술을 활용해야 한다. WSDL은 서비스 공급자가 자신이 제공하는 서비스에 대하여 기술할 수 있도록 하여 서비스 사용자와 서비스 레지스트리의 서비스에 대하여 이해할 수 있도록 한다. UDDI는 서비스 레지스트리에 서비스 사용자가 정의한 웹서비스를 등록하고 서비스 사용자가 발견할 수 있도록 하는 표준화된 방법을 제공하며 SOAP는 서비스 공급자와 서비스 사용자가 서로 통신을 할 때 따라야 할 메시지 형태 및 구성 방식에 필요한 표준을 제공한다.[5]

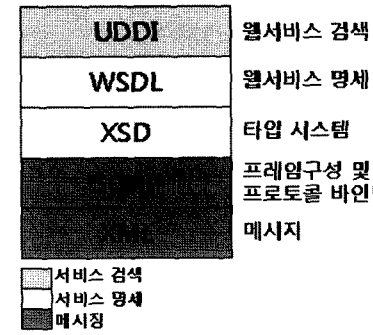


그림 6. SOA기반의 웹서비스 구조

표준기술의 사용은 SOA의 특징인 조합성이나 상호 운영성을 높일 수 있고 내부적으로는 XML로 메시지를 주고받는다. 결국, 그림 6의 구조와 같이 SOA기반의 웹서비스는 XML 메시지를 SOAP 프로토콜을 사용하여 처리하고 XML 스키마로 정의된 메시지는 WSDL을 사용하여 서비스 인터페이스를 기술하며 UDDI를 통하여 등록 및 검색이 가능하도록 설계한다.

4.2 SOA기반의 서비스 시스템 구현

4.1에서 정의한 웹서비스 구조를 SOA의 각 구성요소인 서비스 공급자, 서비스 소비자 그리고 서비스 레지스트리와 연결하여 전체 시스템의 일부로서 역할을 수행하도록 구현해야 한다. 구성요소와 웹서비스의 연결은 일련의 처리 순서에 따라 진행 되는데, 특히 서비스 소비자가 서비스 요청을 하기 이전에 먼저 서비스 제공자로부터 필요한 서비스들이 레지스트리 또는 그와 같은 역할을 수행하는 곳에 등록이 되어 있어야 한다. 그리고 등록된 서비

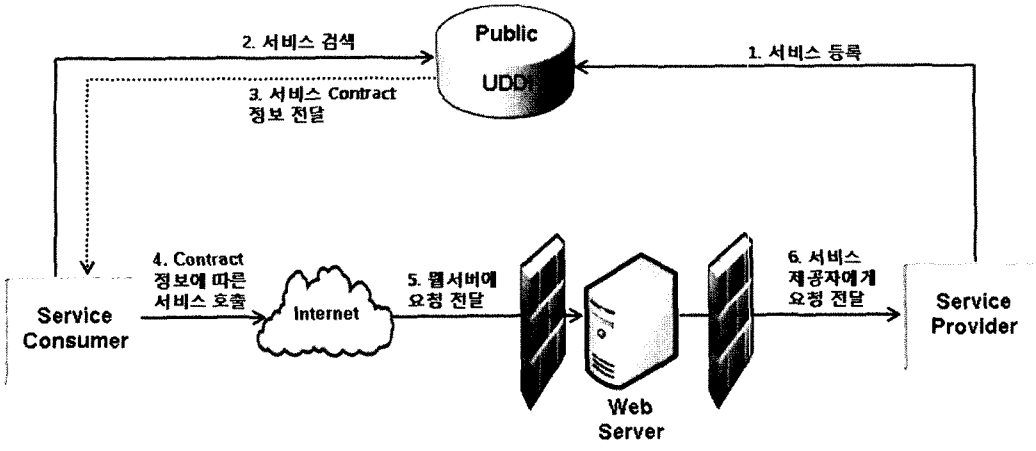


그림 7. SOA기반의 서비스 시스템

스 중에서 소비자가 요청한 사항과 일치하는 서비스 있다면 서비스 레지스트리로부터 해당 서비스의 계약(Contract)정보를 전달 받고 소비자는 그 계약 정보를 웹서버에 호출한다. 웹 서버는 요구 내용에 포함된 계약 정보를 확인하고 해당 서비스를 실제 수행할 수 있는 서비스 제공자에게 요청을 전달한다. 이러한 일련의 과정을 그림 7과 같다.

4.3 SOA기반의 비스 모델링

웹서비스를 구축하기 위해서 일반적으로 업무분석 이나 요구사항을 처리하기 위한 단위항목 설계 등의 작업을 수행 한다. SOA를 위해서는 이러한 작업을 하기에 앞서 서비스 모델링을 위한 과정을 거쳐야 한다. 이 과정은 개발에 필요한 서비스의 집합을 잘 정의하고 서비스 레이어와 애플리케이션 서비스 레이어를 구분하며 여러 가지 정보를 구조화 하는 작업이다. 업무내용이나 전산환경에 따라 서비스 모델링 과정을 구조화 하는 방법이 여러 가지 있을 수 있으나, 본 논문에서는 비즈니스 프로세스 관점에서 공통적으로 적용할 수 있도록 애플리케이션, 비즈니스 그리고 오케스트레이션 서비스 레이어를 중심으로 서비스 모델링 프로세스를 구성하였다.

오케스트레이션이란 자동화된 비즈니스와 다양한 기존 환경을 통합하기 위한 EAI(Enterprise Application Integration)와 같은 미들웨어 제품의 기능중 하나로서 워크플로우(Workflow)엔진을 이용하여 두 개 이상의 다른 애플리케이션 간의 상호 작용을 쉽게 해준다. 따라서 오케스트레이션을 이용하게 되면 기존의 자동화된 프로세스를 개별적으로 재개발하지 않고 다른 프로세스에 연결할 수 있으며 복잡한 솔루션 환경을 쉽게 처리할 수 있다.

SOA에서는 상호 운영성이란 특성을 위해 표준화된 기술을 사용해야 하는데 오케스트레이션 또한 표준화된 산업스펙을 가지고 있으며 이것을 웹서비스 비즈니스 프로세스 실행언어(WS-BPEL)로 정의하고 있다.(2002)

서비스 모델링 프로세스는 비즈니스 요구사항을 정의하는 단계에서 시작하여, 모두 13개의 단계로 구성한다. 각 단계의 구성은 아래 그림 8과 같다.

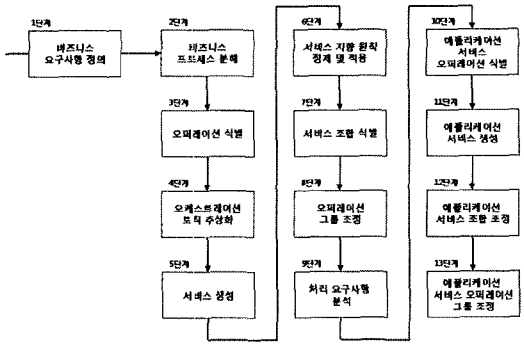


그림 8. 서비스 모델링 프로세스

3단계에 있는 오퍼레이션 식별 단계는 비즈니스 서비스가 자동화 할 수 없거나 수작업이 필요한지 여부를 판단하고 기존 시스템 로직이 수행하는 프로세스 존재 여부 등을 확인하는 과정이다. 4단계 오케스트레이션 로직 추상화 단계는 비즈니스 규칙이나 예외 로직, 순차적 로직 등을 식별하여 추상화 작업을 하는 단계이며 이 단계를 완료 후 서비스 생성단계를 거쳐 6단계 SOA정제 및 적용 단계를 수행한다. 6단계에서는 재사용성, 자율성, 발견성, 상

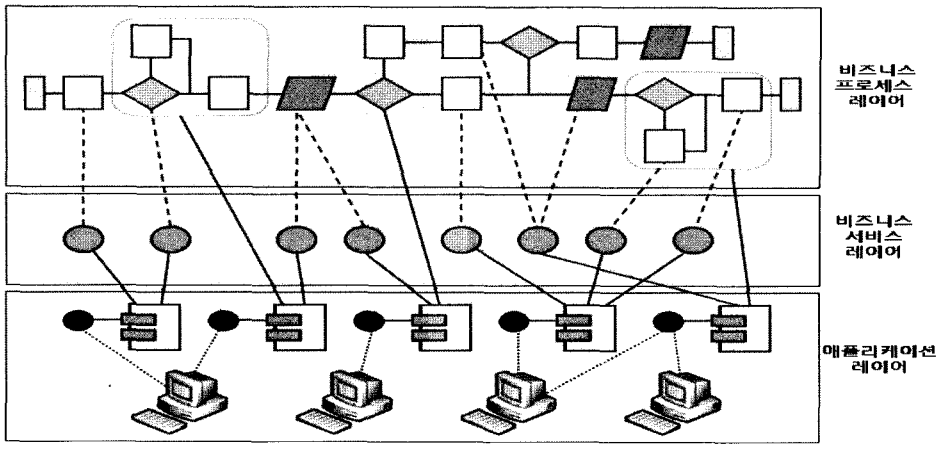


그림 9. SOA기반 웹서비스 시스템 아키텍처

호 운영성 등과 같은 SOA의 핵심 원칙을 적용하는 단계이다. 후반부에 있는 애플리케이션 서비스 조합 조정단계는 6단계에서 식별한 원래 시나리오를 다시 훑어보는 과정으로 애플리케이션 서비스 조합을 정교하게 매핑할 수 있도록 한다. 그리고 13단계에서는 12단계에서 시나리오를 매핑한 결과로 애플리케이션 서비스 오퍼레이션 정의나 그룹을 결정하여 서비스 생성을 완료한다.

4.4 SOA기반 웹서비스 시스템 아키텍처 설계

SOA기반을 위한 웹 시스템과 SOA서비스 모델링 프로세스가 완료되면, 실제 웹 서비스를 구성하기 위한 아키텍처를 설계 할 수 있다. 웹 서비스 아키텍처는 비즈니스 프로세스 레이어, 서비스 인터페이스 레이어 그리고, 애플리케이션 레이어로 구성하였다.

① 비즈니스 프로세스 레이어: 로직에 따라 비즈니스 서비스를 통합하는 프로세스 중심의 서비스가 배치된다. 또한 이 서비스의 자동화를 높이기 위해 프로세스와 프로세스 사이 또는 프로세스 내부에서 서비스와 서비스간의 재사용성을 고려하여 공유 비즈니스 서비스의 역할을 가능하게 한다.

② 비즈니스 서비스 레이어: 비즈니스의 핵심 기능을 직접적으로 구현한 비즈니스 서비스를 제공한다. 태스크(Task) 중심의 비즈니스 서비스와 엔티티(Entity) 중심의 비즈니스 서비스와 같이 두 가지 비즈니스 서비스 모델로 구성할 수 있다. 태스크 중심 비즈니스 서비스는 특화된 로직을 캡슐화 하여 서비스를 제공하기 때문에 재사용성이 낮고, 엔티티 중심 비즈니스 서비스는 특화된 거래 내용에 따른 단위 항목을 캡슐화 하는 서비스를 제공하기 때문에 재사용성이 높고 프로세스에 독립적인 서비스를 생성하는데 유용하다.

③ 애플리케이션 레이어: 서비스 아키텍처의 소비자 또는 사용자 관점에서 정의되는 개별 애플리케이션의 솔루션을 표현하는 레이어다. 비즈니스 서비스 레이어에서 제공하는 공유 비즈니스 서비스를 사용하여 비즈니스가 직면해 있는 문제에 대하여 애플리케이션을 구성한다. 그러나 공유 비즈니스 서비스가 제공되지 않는 경우 애플리케이션 레이어 자체에서 개별적으로 구현할 수 있다.

그림 9는 각 레이어들의 아키텍처 뷰를 보여 주고 있다. 특히 비즈니스 프로세스 레이어의 경우 동일한 로직이 표시 되어 있으며 이러한 부분과 같이 프로세스 내에서 재사용될 수 있는 비즈니스 로직의 식별은 아키텍처 설계에서 중요한 작업이다. 그리고 어플리케이션 레이어에서는 요구에 맞는 서비스가 비즈니스 서비스에 존재하지 않는 경우 직접 비즈니스 프로세스 레이어에 접근하여 해당 로직을 사용할 수 있도록 설계하였다.

4.5 SOA기반 웹서비스 시스템의 평가

SOA를 기반으로 한 웹서비스와 컴포넌트 기반의 애플리케이션에 대하여 조합성, 상호 운영성 및 재사용성 등과 같은 SOA특징을 비교 분석 하기 위해 이미 컴포넌트 기반으로 작성된 주문-이행업무의 샘플 프로그램을 SOA 기반의 웹서비스로 구성하여 차이점을 확인 하였다. 비교

를 위한 각 애플리케이션의 구성 환경은 다음과 같다.

- 컴포넌트 기반 Application 구성 환경
  - Language : Acucobol
  - Client/Server 기반 프로그램
  - 공통 모듈은 컴포넌트 기반 단위 Section 파일
- SOA기반의 Web Service 구성 및 개발 환경
  - 개발 도구 : MS Visual Studio 2005
  - IIS 6.0 기반의 ASP.NET 웹 프로그램
  - 공통 모듈은 업무 단위의 서비스 호출 방식

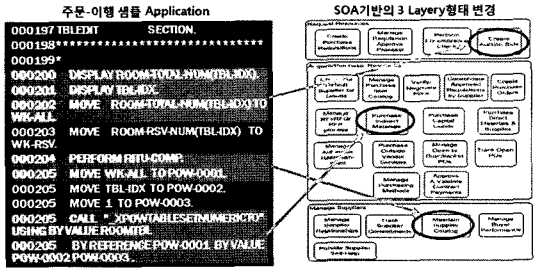
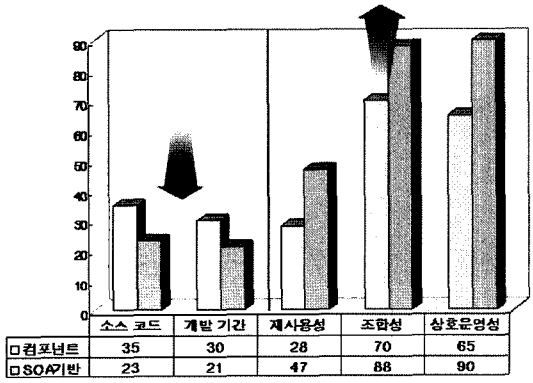


그림 10. 애플리케이션 구성 매핑

기본 애플리케이션을 SOA기반의 웹서비스로 재구성하기 위하여 본 논문에서 제안한 바와 같이 비즈니스 레이와 그에 속하는 서비스를 구성하였으며, 두 애플리케이션에 대한 매핑 구성은 그림 10과 같다.



	소스 코드	개발 기간	재사용성	조합성	상호운영성
□컴포넌트	35	30	28	70	65
□SOA기반	23	21	47	88	90

그림 11. 애플리케이션 형태별 비교 결과

비교 작업을 위한 데이터 추출은 컴포넌트기반 애플리케이션에서는 모듈개수, SOA기반 애플리케이션에서는 서비스 단위 수를 계산하여 각 애플리케이션 형태별 전체 합계에 대한 재사용 가능한 수치와 플랫폼 간의 이행 작업이 가능한 구성단위 수를 비율로 나타내었다.

그 결과 그림11과 같이 SOA를 적용한 경우 개발 기간과 소스코드의 길이는 낮은 수치를 나타내고 있으며, 나머지 SOA의 특징부분에 대해서는 컴포넌트 기반의 애플리케이션 보다 향상된 데이터가 측정 되었다.

5. 결론 및 향후연구

기업환경에서는 신규 어플리케이션 개발을 할 때 기존 환경을 통합하거나 새로운 시스템으로 이전할 필요가 있다. SOA는 이러한 요구에 대하여 재사용성, 유연성 등을 제공하는 최고의 소프트웨어 아키텍처이다. SOA의 이러한 특징을 공개 표준 기술을 사용하는 웹서비스에 접목하여 엔터프라이즈 환경에서 개발 비용의 절감과 지속적으로 변화하는 서비스의 요청에 대하여 빠른 대응이 가능하게 하여 시스템 개발과 통합에 가장 적합한 기반 기술로 전망되고 있으며 기존의 소프트웨어 아키텍처에 비해 우수한 특징을 확인 할 수 있다.

본 논문에서는 SOA기반의 웹서비스를 위한 시스템 구성도를 제시하여 SOA의 개념이 웹서비스 시스템에서 어떤 과정과 흐름으로 처리되는 지를 살펴보았다. 그리고 서비스라는 SOA의 핵심요소를 적용하기 위한 서비스 모델링 절차의 단계별 Flowchart와 SOA기반 애플리케이션의 전체 아키텍처를 3개의 레이어로 구분하여 실제 분산 환경에서 적용가능 한 설계안을 보여주고 있다.

향후 연구로 본 논문에서 제시된 SOA기반의 웹서비스 모델링 방법을 실제 플랫폼(J2EE 또는 .NET 프레임워크)상에 적용하는 방안을 계속 연구 중에 있으며 기존 웹서비스와 성능 비교와 평가가 요구된다.

참 고 문 헌

1. David Booth, "Web Services Architecture", http://www.w3.org/TR/ws-arch/, W3C Working Group Note 11 2004, 2.
2. David Sprottand Lawrence Wilkes, "Understanding SOA", CBDI Journal 2003,9.
3. Munindar P. Singh, Michael N. Huhns, "Service - Oriented Computing : Semantics, Processes, Agents", John Wiley & Sons.
4. D.Plummer, "Gartner's Internet Strategies Commentary COM-12-9640", Gartner, Feb 2001.
5. David Sprott, "Understanding the Component and Web Services Market", CBDI Journal, May 2001.
6. Thomas Erl, "Service-Oriented Architecture: Concepts Technology and Design".

7. Park, D.S., Shin, H.J. and Kim, H.K., "A Study on the Development Web Services Component Based Service Oriented Architecture", Journal of Korea Multimedia Society Vol. 7. No. 10, pp. 1496-1504.
8. W3C Web Services WG. "Web Services Architecture" http://www.w3.org/TR/ws-arch/. W3C Working Group Note 11 February 2004.
9. Roberto Chinnici, "Web Services Description Language (WSDL) Version 2.0 Part 1:Core Language," http://www.w3.org/TR/2004/WD-wsdl20-20040803/, W3C Working Draft 3 August 2004.
10. "UDDI Technical White Paper", http://uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, UDDI.org, September 2000.
11. Lee, K. H, Lee K. C, "Service - Oriented Architecture and Web Service", Information Science Journal 2004, 10 Vol. 22, No. 10.
12. Mark Endrei, Lee, Jenny Ang, "Patterns: Service-Oriented Architecture and Web Services", IBM Redbooks.
13. "OASIS Web Services Business Process Execution Language TC", OASIS WSBPEL Technical Committee, http://www.oasis-open.org/committees/wsbpel/charter.php.

이 성 규 (12284@deu.ac.kr)

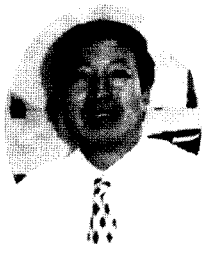
1993년 경남대학교 컴퓨터공학과 학사
1998년 광운대학교 정보통신학과 석사
2006년~현재 동의대학교 컴퓨터소프트웨어공학과 박사과정
2006년~현재 동의대학교 컴퓨터소프트웨어공학과 겸임 부교수

관심분야 : 서비스 지향 아키텍처, 웹서비스, 데이터베이스

진 찬 욱 (jinchanuk@hotmail.com)

1999년 동의대학교 컴퓨터공학과 학사
2006년~현재 동의대학교 인터넷응용공학과 석사과정

관심분야 : 서비스 지향 아키텍처, 웹서비스, XML



김 태 석 (tskim@deu.ac.kr)

1982년 경북대학교 전자공학과
1992년 일본 KEIO대학 이공학부 계산기과학전공 공학박사
1992년 일본 KEIO대학 이공학부 객원연구원
2000년~2003년 동의대학교 전산정보원장
2003년~2005년 동의대학교 교무처장
1993년~현재 동의대학교 컴퓨터소프트웨어공학과 교수

관심분야 : 웹서비스시뮬레이션, 원격강의, 자연어처리