SOA(Service-Oriented Architecture)와 웹 서비스

충남대학교 이경하 · 이규철*

1. 서 론

1996년 Gartner 그룹에 의해 소개된 이래 서비스 지향 아키텍처(SOA; Service Oriented Architecture)[1]는 최근에 많은 지지를 얻고 있는 웹 서비스, ebXML 등의 모태가 되는 개념으로 새로운 소프트웨어 아키텍처로 주목받고 있다. Gartner에 따르면 2006년까지 전 세계 비즈니스 애플리케이션의 80% 이상이 SOA를 기반으로 개발될 것이라고 전망되고 있다. 웹 서비스(Web Services)[2]는 이러한 SOA의 개념을 구현하는 소프트웨어 컴포넌트로써 XML 기반의 메시지를 이용하는 인터페이스들의 집합이라 할 수 있다.

사실 SOA는 이미 CORBA나 DCOM 등의 분산 객체 기술에서도 그 기본 개념이 사용되었으나, 이들은 기술적인 미성숙 및 공개 표준의 부재와 주요 소프트웨어 벤더들 간 협력 부재로 인하여 그리 큰 주목을 받지 못하여 왔다. 하지만 XML 기반의 웹 서비스 기술이 등장하면서 SOA는 다시 중요한 패러다임으로 인식되고 있다.

본 고에서는 SOA의 개념과 특징, 구성 요소에 대해 살펴보고, 웹 서비스의 기본 아키텍처에 대해 설명하도록 한다. 또한 SOA와 웹 서비스와의 관계에 대해서 설명하고, 마지막으로 SOA를 통한 도입 효과와 향후 전망에 대해 논하도록 한다.

2. SOA로의 진화 과정

초기의 프로그램들은 어셈블리어를 이용하여 작성되었으나, 1950년대에 들어서 FORTRAN과 같은 고급 언어가 등장함에 따라 프로그래머들은 프로그램을 보다 쉽게 작성할 수 있게 되었다. 그러나 초기의 프로그래밍 언어들은 문제마다 서로 다른 접근법을 이용함으로써 대형 프로그램의 개발에 있어 개발이 복잡하게 되는 문제를 가지고 있었다. 이러한 문제는 잘 정의된 제어 구조와 코드 블록, 순환 호출과 지역 변수를 지원하는 부 프로그램을 지원하는 구조적 프로그래밍 언어를 통해 많은 부분이 해결되어 왔으며, 현재에는 C++, Java, C#과 같은 객체지향 프로그래밍을 이용하여 개발자가 객체라는 보다 직관적인 방법을 통해 프로그램을 개발할 수 있게 되었다.

객체지향 언어의 출현 이후에는 소프트웨어 개발 시 유사한 코드와 알고리즘의 재개발을 막고, 코드의 재사용을 강화하는 방법이 강구되어 왔다. 코드의 재사용이란 코드가 동작하는 방식이나 하는 일을 사용자가 정의할 수 있으면서도, 보다 커다란 모듈을 작성할 때는 재사용하기에 충분할 만큼 정형화된 코드를 작성해야 함을 의미한다.

코드의 재사용을 위해서는 객체와는 달리 다형성과 상속성을 배제하고 잘 정의된 인터페이스를 통해 소스 코드가 아닌 이진 형식으로 재사용할 수 있어야 한다. 이러한 문제의 해결을 위해 최근까지 컴포넌트기반 개발 방법론(CBD; Component-Based Development)에 대한 많은 연구가 진행되어 왔다. CBD에서는 여러 시스템에 의해 사용될 수 있도록 인터페이스들을 가지는 소프트웨어 조각 또는 소프트웨어의 빌딩 블록을 컴포넌트라 하고, 이들의 생성과 통합, 재사용을 통한 소프트웨어의 개발을 이루어 왔다.

이 중 분산 컴포넌트란 하드웨어 플랫폼, 운영체제, 소프트웨어 등의 환경에 제약 받지 않고 분산 환경에서 개별적, 독립적으로 실행될 수 있는 컴포넌트를 의미한다. 이러한 분산 환경에서의 컴포넌트 기술들로는 최근까지 CORBA, COM+, EJB 등이 소개되어 왔다.

이상적인 분산 환경에서는 사용자들이 분산 컴포넌트의 이용에 어떠한 제약도 없어야 하며, 이를 위해서 분산 컴퓨팅 환경은 아래의 조건을 충족해야만 한다.

- 개발 언어에 상관없이 서비스가 제공되어야 한다.
- 컴포넌트가 특정 플랫폼에 종속되지 않아야 한다.
- 제공되는 서비스의 유지보수가 용이해야 한다.

이전의 COM+, EJB, CORBA와 같은 분산 컴포넌

* 종신회원

트 기술들 또한 위의 조건을 충족시키고자 고안되었지만, SOA에서의 서비스와는 명확한 차이를 가지고 있다. 이들은 자신이 가지는 프레임워크 내부에서 비즈니스 로직을 어떻게 잘 모듈화해서 최대의 효율성을 가지게 하는가에 주안점을 두고 있다. 즉, 각자의 원래 방식으로는 아주 효율적으로 개발, 이용될 수 있지만, 자신의 플랫폼 경계를 넘어설 경우에는 호환성에서 큰 문제점을 보이게 된다. 이에 반해 SOA는 표준화된 인터페이스와 메시징 프로토콜을 이용한 약결합(loosely coupled) 방식의 연결 방식을 제공함으로써 플랫폼에 보다 유연하면서 표준화된 개발 방법을 제공한다.

3. SOA의 개념

3.1 SOA의 정의

서비스란 생산된 재화를 운반, 배급하거나 생산이나 소비에 필요한 노무를 제공하는 것을 의미하며, SOA에서는 하나의 컴포넌트가 다른 컴포넌트와 인터페이스 계약을 통해 제공되는 행동으로 정의한다.

SOA란 이 서비스들을 기반으로 하는 소프트웨어 아키텍처로 '애플리케이션의 기능들을 사용자에 적합한 크기로 공개한 서비스들의 집합으로 이의 제공, 사용에 관한 정책이나 적용 또는 프레임워크'로 정의할 수 있다. 또한 SOA에서의 서비스는 아래의 추가적인 요구사항을 만족해야만 한다.

- **플랫폼에 독립적**: SOA에서의 서비스는 표준화된 방법을 통해 모든 환경에서 호출이 가능해야 한다. 이는 서비스 호출 메커니즘이 널리 채택된 표준에 근거해야 함을 의미한다.
- **약결합 방식**: 서비스는 그 이용에 있어 내부 자료구조나 지식을 필요로 해서는 안된다.
- **위치 투명성의 지원**: 서비스는 그들의 정의와 위치 정보를 UDDI와 같은 저장소에 저장하고 여러 클라이언트를 통해 그들의 위치와 상관없이 등록, 호출될 수 있어야 한다.

즉, SOA란 서비스라 불리는 분할된 애플리케이션 조각들을 단위로 약결합으로 연결하여 하나의 완성된 애플리케이션을 개발하기 위한 소프트웨어 아키텍처라 할 수 있다.

3.2 SOA의 구성 요소

논리적으로 SOA에서의 서비스는 서비스 인터페이스와 구현(implementation)으로 구성된다. 서비스는 일반적으로 소프트웨어에서 구현된 비즈니스 함수이며, 잘 구성된 인터페이스를 통해 외부에 노출된다. 이 서비스 인터페이스는 서비스 요청자가 호출할 수 있는 연산들에 대한 설명이며, 서비스 명세에는 서비스 요청자가 호출할 수 있는 모든 인터페이스가 명시적으로 기술되어야 한다[3,4].

또한 서비스는 그 이용 방법에 따라 단순(simple) 서비스와 복합(composite) 서비스로 구분될 수 있다. 복합 서비스란 단순 서비스들을 조합하여 특정 비즈니스 프로세스를 운용할 수 있도록 구성된 서비스를 의미하며, 이렇게 구성된 서비스들의 집합은 다시 하나의 서비스로 이용될 수가 있다. 그림 1은 SOA의 기본 구성 요소를 보이고 있다.

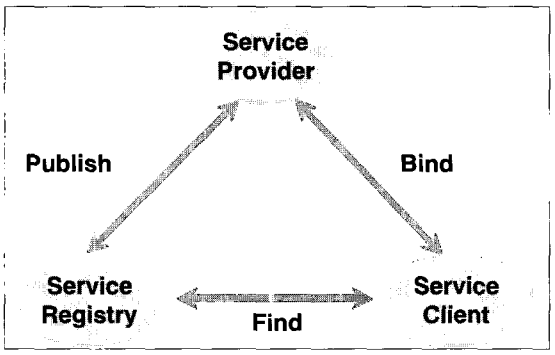


그림 1 기본 서비스 지향 아키텍처

- **서비스 요청자**(Service Client): '서비스 제공자'에 의해 제공되는 하나 이상의 서비스를 이용
- **서비스 제공자**(Service Provider): '서비스 요청자'가 호출 시 입력하는 값을 가공하여, 그에 해당하는 결과를 제공한다. 경우에 따라 '서비스 제공자'는 또 다른 '서비스 제공자'의 서비스를 사용하는 '서비스 사용자'가 될 수 있다.
- **서비스 레지스트리**: 서비스에 대한 기술 정보(description)를 저장, 검색할 수 있게 한다. '서비스 제공자'는 자신이 제공하는 서비스를 등록하고, '서비스 요청자'는 자신이 원하는 서비스를 검색, 호출할 수 있다.

이러한 SOA의 기본 아키텍처는 SOAP, WSDL, UDDI 표준을 기반으로 하는 웹 서비스 아키텍처와 일치한다. 하지만, 위의 SOA의 기본 구성 요소는 실세계에서 필요한 서비스 조합과, 트랜잭션 관리, 조정, 보안 등과 같은 소프트웨어 개발 시 취급해야 하는 다른 문제들을 충분히 고려하지 않고 있다. 이러한 문제는 [3,4]에서 언급되었으며, 이에 따라 아래와 같이 확장된 서비스 지향 아키텍처가 고려되었다. 이 아키텍처는 기본 SOA를 하위 계층으로 하여 두 개의 계층을 추가시킨다. 서비스 통합 계층(service composition layer)은 복수의 기존 서비스를 하나의 서비스로 통합하고자 할 때 필

요한 역할과 기능들을 포함한다. 이러한 기능에는 조정 기능과, 적합성, 모니터링, QoS 보장이 존재한다. 통합 서비스는 서비스 통합자(service aggregator)에 의해 하나의 컴포넌트로써 이용된다. 또한 서비스 통합자는 하나로 통합한 서비스를 제공함으로써 서비스 제공자가 될 수 있다. 또 상단은 관리 계층으로 서비스의 실제 이용에 필요한 인증과 권한, SLA(Service-Level Agreement)과 서비스 연산에 대한 보증과 지원 기능이 요구됨을 보이고 있다.

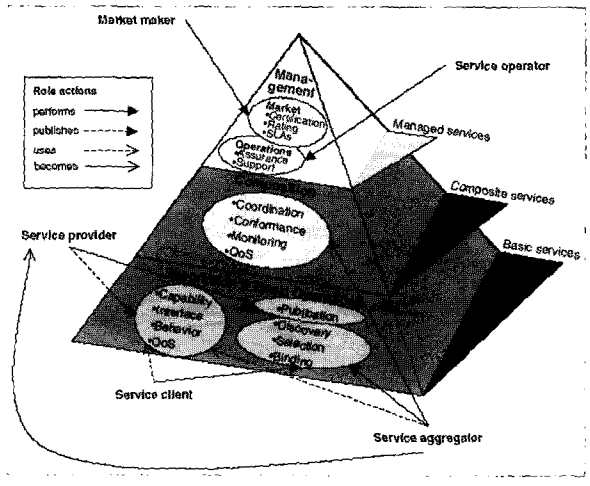


그림 2 확장 서비스 지향 아키텍처

4. SOA의 특징

SOA가 기존 소프트웨어 아키텍처와 차이를 보이는 몇가지 특징을 나열하면 아래와 같다.

- **상호운용성**: 모듈 간 결합도가 높을수록 한 모듈의 변화가 다른 모듈에 영향을 미치는 파문 효과(ripple effect)를 발생하게 되며, 파문 효과가 클수록 시스템의 유지 보수는 어려워진다. SOA는 표준화된 메시지를 이용하여 서비스를 호출, 이용하는 약결합 방식이므로, 기존의 컴포넌트에서보다 운용 플랫폼에 대한 상호운용성이 높은 특징을 가진다.
- **위치 투명성**: SOA의 서비스는 네트워크 주소로 접근 가능한 인터페이스를 가지고 있으며, 이 네트워크 주소는 서비스 레지스트리를 통해 발견되어진다. 서비스의 이용은 해당 프로세스가 실행 중에 동적으로 발견, 호출될 수가 있다.
- **프로세스 중심**: 한 시스템에서 서비스 또는 애플리케이션 통합에 관한 요구는 비즈니스 프로세스의 자동화라는 요구에 의해 발생된다. 실제로 비즈니스 서비스는 타인에게 자신의 서비스를 제공하는 반면 프로세스 서비스는 이들 간의 통합을 위한 공통 인프라를 제공한다. SOA에서는 프로세스 서비스를 별도의 구성 요소로 두어 통합에 필요한 메시지 처리와 서비스 오케스트레이션을 담당하게 하고 있다. 그렇게 함으로써 통합에 참여하는 애플리케이션이 자신의 고유 기능에만 집중할 수 있도록 한다.

  이런 구조는 각 애플리케이션들의 서비스 진입 지점을 프로세스 서비스로 단일화하고, 업무 프로세스 추적을 가능하게 해준다. 이를 통해 비즈니스 외적인 예외 사항을 보다 쉽게 처리하게 할 수도 있고, 각 애플리케이션들은 업무 프로세스 변경에 독립적일 수 있게 된다.

  물론 프로세스 서비스는 서비스 조합과 관리를 위해 메시지 처리, 메시지 연계(Correlation), 비즈니스 트랜잭션 처리, 영속적 프로세스 관리 기능 지원해야만 한다.

위의 내용에서 보다시피 SOA는 엔터프라이즈 애플리케이션의 개발 시 개발의 유연성을 보장할 수 있도록 고안된 하나의 소프트웨어 아키텍처라 할 수 있다. 하지만 SOA는 컴포넌트 간 상호운용성을 보장함으로써 모듈의 재사용성을 높인다는 면에서는 컴포넌트기반 개발 방법론의 연장선상에 위치하며, ROI(Return of Investment)의 증대와 플랫폼간의 독립성을 보장한다는 면에서 MDA(Model Driven Architecture)[5]와 밀접한 관련을 맺는다.

5. 웹 서비스 기본 아키텍처

5.1 SOA와 웹 서비스

SOA는 웹 서비스 개념보다 먼저 출현하였으며, 웹 서비스보다 포괄적인 개념이다. SOA는 소프트웨어 아키텍처에 가깝고, 웹 서비스는 이러한 아키텍처를 실현하기 위한 기술들의 모음이라고 할 수 있다. 이름에서 알 수 있듯이 SOA는 소프트웨어 아키텍처로, 특정 기술의 집합이 아니며 기술로부터 독립적이다. 비즈니스 환경에서 SOA의 순수한 정의는 "호출 가능한 잘 정의된 인터페이스를 갖는 독립된 기능의 서비스로 정의한 애플리케이션 아키텍처"이다. 반면, 웹 서비스는 기술의 집합체이며 SOA의 개념을 구체화 한 것이다.

하지만 웹 서비스는 단순한 SOA의 구현만은 아니다. 웹 서비스는 SOA 구현의 Best Practice라 할 수 있다. 왜냐하면 SOA의 근본적인 개념을 고스란히 실현할 수 있도록 플랫폼 독립적인 프로토콜과 기술을 채택하였기 때문이다. 웹 서비스는 HTTP, XML, SOAP, WSDL, UDDI 등의 표준을 기본 구조로 이용할 하고 있으며 이들 기술은 특정 플랫폼에 독립적이며, 표준화된 기반을 가지고 있으며, 또한 상호운용성을 충분히 보장하고 있다.

웹 서비스는 XML 표준을 기반으로 개발된 표준화된 XML 메시지를 통해 네트워크 상에서 접근 가능한 연산들의 집합을 기술한 인터페이스[6]로 기존의 웹과는 분명한 차이[7]를 보인다.

- **단순성** : 웹 서비스는 XML 기반이므로 기존의 분산 컴퓨팅 모델에 비해 보다 단순하고 확장이 용이한 모델을 제공한다.
- **상호운용성** : 웹 서비스는 교환 메시지를 포함한 모든 정보 표현에 있어 XML을 이용하며, 기존의 웹 환경 위에 바로 구현할 수 있는 이점을 제공함으로써 이기종 시스템 간의 상호운용성을 극대화하였다.
- **표준 기반** : 웹 서비스는 XML 기술을 기반으로 한 개방형 표준들의 지원을 받는다.
- **빠른 발전 및 업계의 지원** : 웹 서비스는 Microsoft, IBM, SAP, HP, Oracle을 비롯한 주요 IT 업체들이 지원을 받으며, 빠른 발전 속도와 함께 다양한 개발 도구의 지원을 받고 있다.

웹 기술의 발전단계로 구분해 볼 때 웹 서비스 기술의 출현 이전을 제 1세대 웹으로, 웹 서비스의 출현 이후를 제 2세대 웹으로 분류할 수 있다[7].

제 1세대 웹에서는 서비스를 이용하기 위해 사용자가 브라우저에 해당 서비스를 제공하는 사이트의 URL을 입력하거나 검색 엔진 등을 이용하여 컨텐츠 또는 웹 어플리케이션에 접근한다. 즉, 1세대의 웹은 서비스 지향이라기보다는 컨텐츠, 애플리케이션 지향적이며, 사용자만이 서비스를 이용한다는 점에 있어 사용자 중심적인 특징을 가진다. 반면에 제 2세대의 웹은 서비스 지향적인 특징을 보이며, 또한 사용자뿐만 아니라 애플리케이션 간의 통신이 가능하며, 서비스 레지스트리를 이용한 서비스의 검색과 동적 바인딩이 가능하다. 또한 최근에는 온톨로지를 이용한 시맨틱 웹의 구축을 통하여 에이전트 기반의 지능형 검색과 통합을 구현하기 위한 노력이 이루어지고 있다.

표 1 웹 서비스의 진화

1세대 웹: 웹 서비스의 출현 이전	2세대 웹: 현재의 웹 서비스	3세대 웹: 시맨틱 웹 서비스
• 브라우저 기반의 이용 • 사용자 중심 • 컨텐츠, 애플리케이션 지향 • URL 만을 이용한 수동적 접근	• 애플리케이션 기반의 이용 • 사용자, 애플리케이션 중심 • 서비스 지향적 • 레지스트리를 이용한 검색, 동적 바인딩	• 에이전트 기반의 이용 • 사용자, 애플리케이션 중심 • 목적 지향적 • 온톨로지 기반의 서비스 검색, 관리, 통합

5.2 웹 서비스 아키텍처

위와 같은 특징들을 지원하기 위해 웹 서비스는 크게 서비스 작성과 기술(description), 그리고 등록(registration), 발견(discovery), 호출(invocation)로 구성된 개발단계를 가지고 있으며, 각 개발단계의 지원을 위한 관련 표준들을 이용하고 있다. 현재 웹 서비스에서 이용하고 있는 기반 표준으로는 SOAP, WSDL, UDDI가 존재한다.

SOAP(Simple Object Access Protocol)[8]은 Microsoft에서 제안하여 현재 W3C에서 표준화를 진행하고 있는 XML 프로토콜 표준으로, 웹 서비스의 요청 및 응답에서 사용되는 메시지 형식을 정의하고 있으며, WSDL(Web Services Description Language)[9]은 웹 서비스 이용에 필요한 인터페이스와 입/출력 메시지의 형식을 기술하기 위해 이용된다. UDDI(Universal Description, Discovery and Integration)[10]는 웹 서비스에 대한 디렉토리 서비스를 지원하기 위해 개발된 분산 레지스트리 표준으로 웹 서비스를 등록하고, 검색/바인딩하기 위한 메커니즘을 제공한다.

현재의 웹 서비스는 이들 기술을 기반으로 구현되어 있으며, 이는 그림 1에서 보인 SOA의 기본 구조와 일치한다. 즉 UDDI 레지스트리가 SOA의 기본 구조에서의 서비스 레지스트리로서 동작하며, 이 때 서비스 간에 교환되어지는 메시지는 SOAP을, 서비스의 바인딩과 호출을 위해 필요한 인터페이스의 명세는 WSDL를 이용하도록 되어 있다. 이들 표준들은 모두 XML로 작성되었으며, 또한 기존의 웹 환경에 그대로 이식할 수 있다는 점 때문에, 적은 이식 비용과 플랫폼에 독립적이라는 특징을 가지게 된다.

현재 주요 IT 업체들은 위의 표준들을 기반으로 여러 웹 서비스 아키텍처를 설계, 제안하고 있으며, 이러한 웹 서비스 아키텍처는 계층적으로 나뉘어진 개념적 스택 구조로 정의된다. 웹 서비스 아키텍처는 웹 서비스가 실제 엔터프라이즈 환경에 도입 시 필요한 기능들과 이들 기능들을 구현한 기술들 간의 상호운용성을 보장하기 위해 요구된다. 이에 따라 W3C에서는 Web Services Architecture WG에서 그림 3과 같은 웹 서비스 아키텍처를 정의하고 있다. 여기에서는 각 기술 별로 계층적으로 분리시킴으로써 상호운용성을 보장할 수 있도록 하고 있다. 실제 엔터프라이즈 환경에 웹 서비스를 도입하기 위해서는 여러 가지 고려 사항들이 발생할 수 있으며 이러한 것들로는 메시지 전달의 신뢰성, 보안, 트랜잭션의 지원, 프로세스의 통합과 조정, 관리 기능 등이 존재한다. 이러한 사항들은 그림 2에서 보인 확장 서비스 지향 아키텍처에서도 언급된 바 있다. 즉, 웹 서비스란

XML과 웹을 기반으로 하는 SOA의 구현을 위한 통합 기술이라 할 수 있다.

현재 SOAP, WSDL, UDDI을 제외한 다른 요구 사항에 관한 기술들은 표준이 제정이 되어 있지 않은 상태로 각 기능에 따라 여러 기술들이 난립하고 있다. 하지만 이런 문제는 조만간 해결이 될 것이며, W3C의 웹 서비스 아키텍처와 확장 서비스 기반 아키텍처에서 언급한 여러 기능들에 대한 표준이 정의될 것으로 기대된다.

하지만 해당 기능에 대한 표준화가 완료되었다 하더라도, 동일한 표준을 지원하는 개발 도구간에서 이들에 대한 호환성을 지속적으로 테스트하고 조정할 필요성이 존재한다. WS-I(Web Services Interoperability)는 웹 서비스 기술들에 대한 상호운용성을 보장하기 위한 목적으로 웹 서비스 관련 업체들이 조직한 기구로 현재 UDDI 2.0, WSDL 1.1, SOAP 1.1, XML Schema 1.0과 같은 주요 웹 서비스 기술 표준들의 이용에 관한 프로파일과 적합성 테스트를 위한 프로그램의 배포, 웹 서비스 사용 시나리오와 예제 응용의 개발 작업을 수행하고 있다.

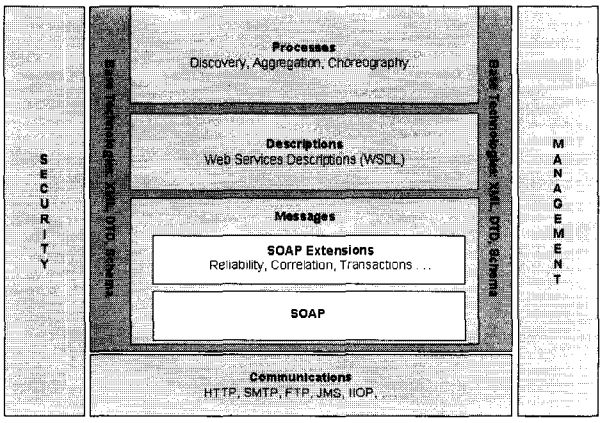


그림 3 웹 서비스 아키텍처

현재의 웹 서비스는 엔터프라이즈 환경에서의 모든 문제를 해결해 줄 수는 없지만, 표준화 작업과 기술 도입을 통하여 빠르게 정착될 것이며, SOA의 구현 기술로의 이용 가능성이 매우 높다고 할 수 있다.

6. SOA와 웹 서비스의 도입 효과 및 전망

엔터프라이즈 애플리케이션의 개발을 위해서는 기 구축된 애플리케이션들을 필요에 따라 통합하거나 시스템의 변경이 용이해야 한다. 즉, 비즈니스 유연성(agility)을 보장하기 위한 소프트웨어 개발 방법이 요구된다. SOA는 비즈니스 유연성을 적은 비용으로 보장받으면서 엔터프라이즈 애플리케이션을 구현하기 위해 고안된 아키텍처로 빠른 시장 접근과, 개발 비용의 절감, 끊임없이 변화하는 비즈니스 프로세스에의 용이한 시스템 변경이 가능하다는 점에서 기존의 소프트웨어 아키텍처에 비해 우수한 특징을 가지고 있다. 웹 서비스는 이러한 SOA가 실현가능하도록 필요한 기반 기술들을 가지는 통합 방법을 제공함으로써 엔터프라이즈 환경에서의 시스템 개발과 통합에 적합하다. 또한 웹 서비스는 최근의 그리드 컴퓨팅[11]의 기반 기술로 활용되면서 그 영역이 플랫폼으로까지 확산되는 경향을 보이고 있으며, 시맨틱 웹과의 연동을 통한 지능형 서비스 검색과 통합 또한 가능할 것으로 기대된다.

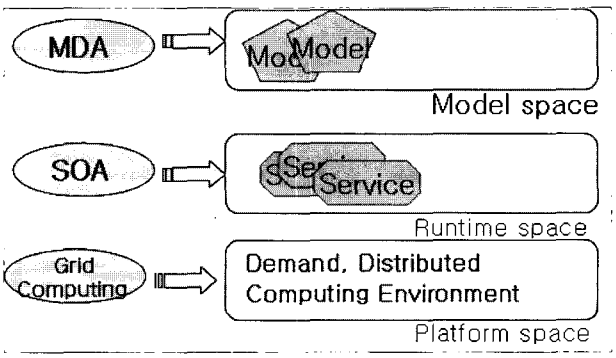


그림 4 MDA, SOA, 그리드 컴퓨팅

참고문헌

[1] Roy W. Schulte Yefim V. Natis, "Service Oriented Architecture," Gartner Group, SSA Research Note SPA-401-068, 1996.

[2] W3C Web Services WG, "Web Services Architecture," http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, W3C Working Group Note 11 February 2004

[3] M.P. Papazoglou et al. "Service-Oriented Computing," Communications of ACM Vol. 46, No. 10, pp.25-28, Oct 2003.

[4] M.P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics and Directions," Proc. of 4th Int'l Conf. on Web Information System Engineeriong, 2003.

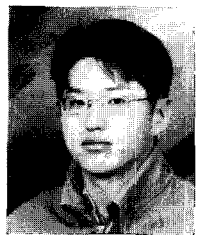
[5] D. S. Frankel. "Applying MDA to Enterprise Computing," OMG Press and Wiley Publishing, Inc, 2003.

[6] Header Kreger, IBM Software Group, "Web Services Conceptual Architecture(WSCA 1.0)," http://www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, May 2001.

[7] Fabio Casati, et al. "Models and Languages

for Describing and Discovering E-Services," Tutorial, Semantic Web Working Symposium, Stanford, USA, July 2001.
[8] Nilo Mitra, "SOAP Version 1.2 Part 0: Primer," W3C Recommendation, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/, June 2003.
[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] Joshy Joseph, Craig Fellenstein, "Grid Computing," Prentice Hall PTR, Dec, 2003.

이 경 하



1998 충남대학교 정보통신공학과(공학사)
2000 충남대학교 대학원 정보통신공학과(공학석사)
2003 충남대학교 대학원 박사과정 수료
현재 동 대학원 전문연구요원
관심분야 : 데이터베이스, XML, 정보통합
E-mai : bart@ce.cnu.ac.kr

이 규 철

1984 서울대학교 컴퓨터공학과(공학사)
1986 서울대학교 대학원 컴퓨터공학과(공학석사)
1990 서울대학교 대학원 컴퓨터공학과(공학박사)
1989~현재 충남대학교 컴퓨터공학과 교수
1994~1995 미국 IBM Almaden Research Center 객원 연구원
1995~1996 미국 Syracuse University, CASE Center 객원 교수
1997~1998 학술진흥재단 부설 첨단학술정보센터,파견교수
1999~2001 행자부 전자문서유통 활성화사업(XML), 표준 책임자
1999~현재 한국정보과학회 논문편집위원
2000~현재 한국 ebXML 전문위원회 위원장
2001~현재 전자상거래 표준화 통합 포럼, 전자거래 기반기술 위원회 부위원장
2003~현재 웹 코리아 포럼, 웹 기술분과위원회 위원장
관심분야 : 데이터베이스, XML, 정보통합, 멀티미디어 시스템, e-비즈니스 시스템
E-mail : kclee@ce.cnu.ac.kr