SOA 프레임웍 아키텍처

TmaxSoft 김성익
신한은행 박정일

1. 서 론

최근의 비즈니스 환경은 과거 독립적인 조직 및 프로세스에 의해 주도되는 수직적 통합에서 고객, 공급자, 파트너 등 다수 기업과의 관계적 협업관계가 중시되는 수평적 통합환경으로 변화하고 있다. 은행의 대출금리가 신용카드 사용액과 연동되어 적용되는 복합상품의 출현은 그 대표적인 예라 할 수 있다. 이러한 관계적 협업이 중시되는 비즈니스 환경에서 경쟁력있는 기업은 급변하는 시장요구에 민첩하게 대응할 수 있어야 한다. 이러한 비즈니스 요구를 효율적으로 대응하기 위한 기업 IT 아키텍처로 대표되는 것이 '서비스 지향 아키텍처 (SOA: Service-Oriented Architecture)'이다. SOA는 전통적인 프로그램 중심의 설계/개발 방식에서 비즈니스 프로세스 관점에서 재활용 가능한 단위로 서비스를 설계/개발하게 함으로써 특정 프로세스나 서비스 변경 또는 내부/외부 시스템과의 비즈니스 통합시 효율적이고 빠른 대응이 가능하다는 점에서 그 의미가 깊다.

서비스 지향 아키텍처는 특정 기술이나 플랫폼에 종속적인 것은 아니다. SOA가 추구하는 것은 느슨한 결합(Loosely Coupled)을 가지고 상호연동할 수 있는 (Interoperable) 서비스들의 조합으로 어플리케이션 개발을 가능하게 하는 정보시스템 아키텍처이다. 즉 한 덩어리의 방대한 코드로 이루어진 어플리케이션들을 각각 개발하는 대신 각각의 비즈니스 기능을 수행하는 서비스를 구성하고, 이 서비스를 조합하거나 분리함으로써 비즈니스 프로세스들을 구현할 수 있게 하는 정보시스템 구축을 목표로 한다. 이러한 개념을 기업내 혹은 기업간 비즈니스 프로세스 차원에서 구현하게 되면, 그림 1 에서와 같이 수평적 통합환경의 IT 아키텍처를 구현할 수 있게 된다. 본 연구에서는 서비스 지향 아키텍처의 기본 원리들과 관련 기술을 살펴보고, 서비스 지향 아키텍처 구현의 핵심 인프라인 SOA 프레임웍(Framework) 구조를 소개하면서, 현장에서 핵심기술이 적용되는 방식을 보이고자 한다.

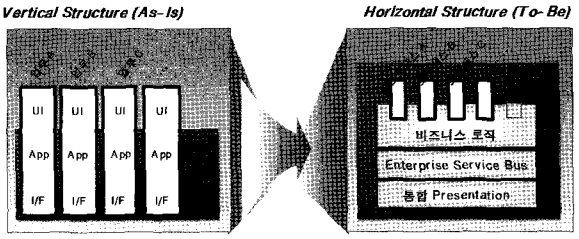

그림 1 수평적 통합환경으로의 변화

2. 서비스 지향 아키텍처

2.1 SOA(Service-Oriented Architecture) 정의

1996년 가트너(Gartner)그룹에 의해 처음 소개된 이후, SOA에 대한 많은 정의들이 존재하지만, 현재까지 기술 및 비즈니스적으로 깊이있게 적용할 수 있는 공식적인 정의는 OASIS(the Organization for the Advancement of Structured Information Standards) 그룹의 정의가 유일하다.

SOA Definition :

A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.[2]

IT 관점에서 SOA는 "느슨한 결합을 가지는 서비스(Loosely coupled services)를 지원하는 하나의 아키텍처 스타일(architectural style)" 이라고 할 수 있다. 서비스 지향 아키텍처 기반의 서비스들은 네트워크상에서 상호연동할 수 있어야 하고 비즈니스 프로세스를 구현하기 위해 조립되어질 수 있고, 동적으로 재구성되어질(re-configurable) 수 있도록 인터페이스 기반의 서비스 명세(description)를 제공한다. 그림 2[1]는 서비

스 지향 아키텍처의 요소들을 설명한다.

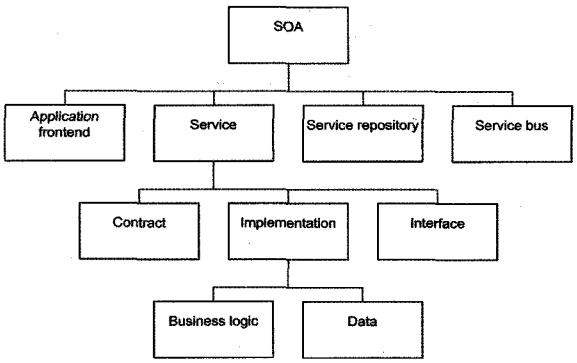

그림 2 Elements of SOA

2.2 SOA와 웹서비스

웹서비스는 XML과 인터넷 프로토콜을 이용하여 네트워크상에 분산된 서비스간의 상호연동이 가능하도록 하는 표준기술로서 업계에서 광범위하게 사용되고 있다. 서비스 지향 아키텍처는 웹서비스 표준에 기반하여 구현되어 질 수 있지만, 특정 기술이나 프로토콜에 종속적이지는 않으며, CORBA, Jini와 같은 기술 기반으로도 구현되어질 수 있다.

서비스 지향 아키텍처를 구현하기 위한 웹서비스 기본 표준은 다음과 같다

- XML(eXtensible Markup Language)
- HTTP(Hyper Text Transfer Protocol)
- SOAP(Simple Object Access Protocol)
- WSDL(Web Services Description Language)
- UDDI(Universal Description discovery and Integration)

2.3 SOA 기반 서비스 특징

SOA는 서비스와 사용자를 분리하여 분산된 환경에 위치한 서비스가 네트워크상에서 접근 가능하도록 함으로써 서비스 재사용성을 최대화 한다. 성공적인 SOA 기반의 서비스 정의 및 설계에 있어 고려해야 할 아키텍처상의 핵심 요건은 다음과 같다.

- Loose coupling :

다른 서비스에 대한 인식(awareness)을 제외한 서비스간 의존성(dependency)을 갖지 않아야 한다.

- Statelessness :

특정 액티비티(activity) 수행에 따른 상태정보를 서비스에 유지하지 않아야 한다.

- Composability :

복합서비스(composite service) 구현을 위해 서비스들은 조립되어질(assembled) 수 있고 조정되어질(coordinated) 수 있어야 한다.

- Autonomy :

서비스는 내포된(encapsulated) 로직에 대한 완전한 제어를 가져야 한다.

- Reusability :

재활용(reuse)이 가능하도록 비즈니스 로직을 단위 기능으로 분리하여 서비스화하여야 한다

- Contract :

서비스 명세(service description)에 정의된 통신규약을 준수해야 한다

- Discoverability :

외부에서 인지하고 접근가능하도록 설계되어야 한다.

상기 아키텍처상의 요건외에 서비스들은 다양하고 분산된 환경에서 상호운용성(interoperability)을 보장하도록 구현되어야 한다.

3. SOA 프레임웍

2006년 8월 OASIS 그룹이 SOA를 위한 참조모델(Reference Model) 1.0을 발표했지만, 가트너(Gartner) 그룹이 서비스 지향 아키텍처 개념을 소개한지 10년이 넘은 현재까지도, SOA 어플리케이션을 위한 구체적이고 견고한 설계/구축 방법론은 확립되지 못한 상태다. 본 연구에서는 SOA 어플리케이션의 구축 및 운영환경을 제공하는 SOA 프레임웍의 아키텍처 및 기능에 대해 소개함으로써 서비스 지향 정보시스템 구축에 참조모델을 제공하고자 한다.

3.1 권장 어플리케이션 아키텍처

기존의 어플리케이션이 요구되는 핵심 비즈니스 로직과 수행시 관계되는 다른 기능을 연동하는 로직을 한 덩어리로 구현하는 프로그래밍 모델을 갖는 반면, 서비스 지향 어플리케이션은 핵심 비즈니스 로직과 연동 로직(Flow Rule)을 분리하여(그림 3), 서비스간 의존성(dependency)을 최소화하는 느슨한 결합(loosely-coupled) 구조를 갖는 점이 어플리케이션 아키텍처상의 핵심 요소이다.

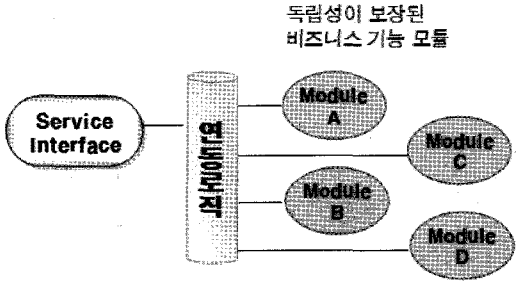

그림 3 비즈니스로직과 연동로직의 분리

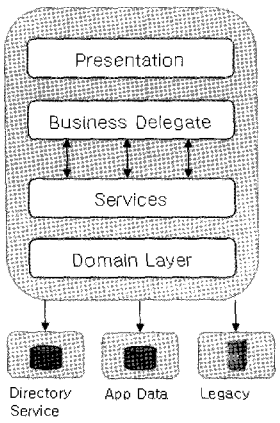

그림 4 권장 어플리케이션 아키텍처

그림 4는 SOA 구축 원리들(principles)을 준수하면서 모든 레벨에서의 재사용을 가능하게 하는 어플리케이션 아키텍처를 보여준다. 즉 서비스를 사용자(Presentation or Service consumer)로부터 분리하여 서비스의 독립성과 재사용성을 보장하고, 기업내 정보자원에 대한 추상화 layer를 통해 통합적인 view를 제공함으로써 정보원천시스템 (source system) 변화에 대한 영향도를 최소화하고, 정보자산(Information assets)에 대한 재사용성을 최대화할 수 있다. 권장 어플리케이션 아키텍처의 각 계층(Layer) 설명은 다음과 같다.

- Presentation Layer :

Presentation 서비스를 제공한다. 기업내 각 어플리케이션에 대한 뷰(view)를 통합된 UI로 제공하고, 통합 인증(SSO : Single Sign-On) 기능과 권한정보관리 기능을 제공한다.

- Business Delegate Layer :

Presentation layer와 Services layer간 통신 및 인터페이스를 담당한다. Presentation layer 및 네트워크상의 서비스 사용자(service consumer)에 대하여 서비스 호출에 요구되는 복잡하고 상세한 사항에 대한 추상화를 제공한다. 또한 비즈니스 프로세스 관리(Business Process Management, BPM) 기능과 멀티채널통합 기능을 제공한다.

- Services Layer :

비즈니스 기능을 제공한다. 요구되는 비즈니스에 대한 진입점(entry point)을 제공하여, 비즈니스 처리의 복잡하고 상세한 사항에 대한 추상화를 제공함으로써 서비스 사용자(service consumer)와 서비스가 밀결합(tightly-coupled)되지 않도록 하는 아키텍처를 가진다. Services layer는 stateless 서비스들로 구성되며, 이러한 서비스들은 2.3절에 기술된 특징을 가진다. 트랜잭션 처리, 부하분산 등 어플리케이션 서버 기능을 제공한다.

- Domain Layer :

영속적인(persistent) 속성을 가지는 비즈니스 객체들에 대한 접근 서비스를 제공한다. 데이터 접근 객체가 대표적인 예이며, 기업내 정보자원에 대한 통합적인 추상화를 제공해야 한다. 최근 금융, 통신 인더스트리에 활발하게 적용되고 있는 상품 팩토리(Product Factory)는 기존 서비스 프로그램에 하드코딩되어 있던 상품 정보를 데이터베이스화하고 접근 서비스를 제공하는 Domain

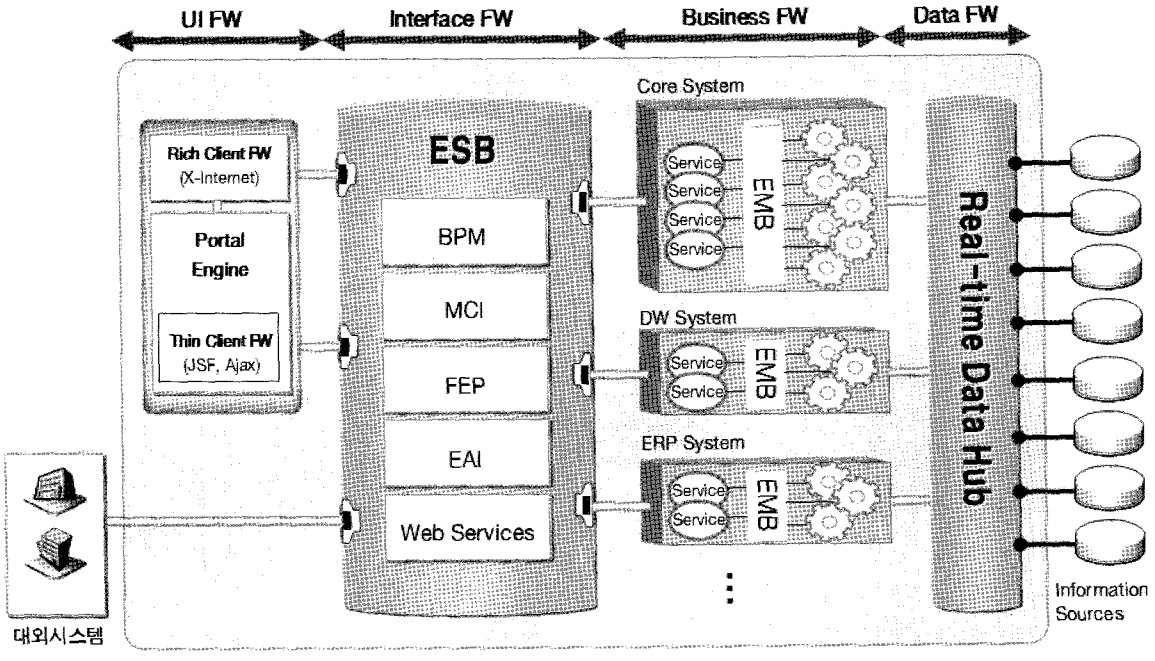

그림 5 SOA 통합 프레임웍 모형

Layer의 특화된 기능으로 볼 수 있다.

권장 어플리케이션 아키텍처를 기반하여 설계되고 구축된 어플리케이션은 Presentation과 비즈니스 로직을 느슨한 결합구조로 구성함으로써, 안정적(reliable)이고 확장성(scalable)있는 시스템 구성이 가능하며, 서비스 조합을 통한 새로운 어플리케이션 구성이 가능한 이점을 갖는다.

3.2 SOA 통합 프레임웍 모형

SOA 프레임웍은 그림 4의 권장 어플리케이션 아키텍처를 지원하여, 각 layer별 컴포넌트의 개발환경을 제공하고 각 layer에서 요구되는 공통 서비스 기능 및 layer 간 인터페이스 방안을 제공한다. 그림 5[3]는 SOA 통합 프레임웍 모형을 보여준다.

SOA 통합 프레임웍은 UI 프레임웍, Interface 프레임웍, Business 프레임웍, Data 프레임웍으로 구성되며, 각 프레임웍의 설명은 다음과 같다.

3.3 UI Framework

UI 프레임웍은 어플리케이션의 Presentation layer의 개발환경과 공통기능을 제공하며 기업 업무의 단일 접점 구축으로 다양한 업무를 통합할 수 있는 기반을 제공한다. 기술적으로 Portal 서비스와 Thin 혹은 Rich Client를 제공하는 Portal / Web UI 통합 개발환경을 제공하고, 개인화 (Personalization), 통합인증(SSO), 권한관리 등 공통서비스 기능을 제공한다. 그림 6[3]은 UI 프레임웍(포탈) 아키텍처 예를 보여준다.

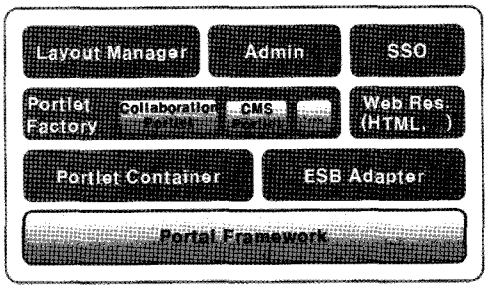

그림 6 UI Framework (Portal)

3.4 Interface Framework

인터페이스 프레임웍은 기업 내부/외부 어플리케이션 간의 통합, 비즈니스 프로세스 자동화를 위한 기반을 제공하여 서비스 지향 아키텍처 구현에 있어 매우 중요한 기능을 담당한다. 기존 시스템 아키텍처에서 인터페이스 프레임웍은 통합채널(MCI : Multi Channal Integration), 어플리케이션 통합(EAI : Enterprise Application Integration), 비즈니스 프로세스 관리(BPM : Business Process Management), 대외계(FEP : Front-End Processer) 등 인터페이스 대상 또는 통합목적별로 분리된 구조를 갖는 것이 일반적이었지만, SOA 프레임웍에서는 서비스 버스(ESB:Enterprise Service Bus)를 바탕으로 통합된 인터페이스 프레임웍을 제공한다. 그림 7은 Enterprise Service Bus 아키텍처 예를 보여준다.

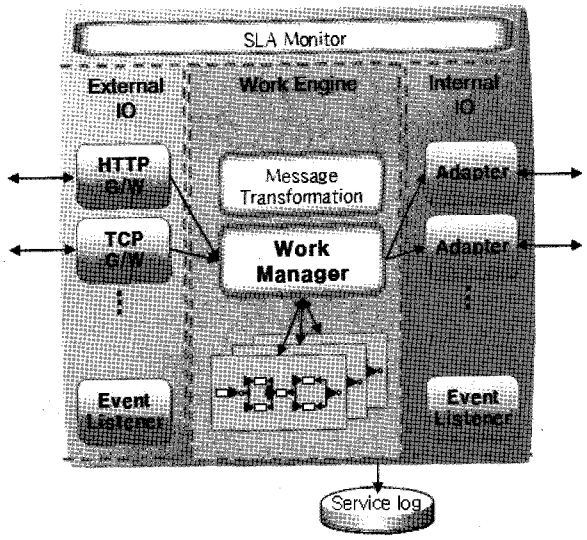

그림 7 Enterprise Service Bus Architecture

ESB는 비즈니스 프로세스를 구성하는 서비스간의 처리 흐름을 Rule화하여 ESB Work Engine이 서비스간 연계를 처리하기 때문에, 서비스는 순수 비즈니스 기능 로직만 처리하도록 설계되고, 서비스간 의존성을 최소화하여(loosely coupling), 서비스의 조립 및 재구성을 가능하게 한다. 또한 서비스 버스는 다양한 프로토콜에 기반한 동기(synchronous)/비동기(asynchronous) 메시지를 처리하여 규약(configuration rule)에 따라 목적지(destination)로 라우팅(routing)하는 Message Broker 기능과 메시지 변환(transformation) 기능을 제공한다.

인터페이스 프레임웍이 제공하는 또 다른 기능은 다음과 같다.

- SLA(Service Level Agreement) Monitoring :

서비스 사용자와 서비스간의 모든 메시지 흐름(flow)을 서비스 버스가 제어하기 때문에 ESB는 서비스 레벨에서의 모니터링을 제공할 수 있는 유일한 위치에 있게 되며, 인터페이스 프레임웍은 SLA(Service Level Agreement) Monitoring 기능을 제공한다.

- Service Registry :

서비스는 Network 상의 분산된 환경에서 접근 가능하여야 하고, runtime binding을 통해 사용되어질 수 있다. Service Registry 기능은 서비스에 대한 카탈로그(catalog) 정보를 publish 하고, 네트워크상에서 이

용되어질 수 있는 메카니즘을 제공한다.

- SOA Repository :

서비스의 생명주기(life cylce)에 걸쳐 서비스에대한 메타데이터(metadata)를 관리하는 기능

3.5 Business Framework

비지니스 프레임웍은 비즈니스 기능을 담당하는 서비스의 개발환경과 운영환경을 제공한다. 느슨한 결합구조를 가지고(Loosely coupled), 단위 기능화되어 재사용이 가능하며(Reusable), 비즈니스 프로세스를 구현하기 위해 조립되거나(assembled) 재구성되어질(re-configurable) 수 있는 서비스를 개발하는 것은 매우 높은 설계능력과 비즈니스 요건에 대한 깊이있는 지식 없이는 불가능한 일이다. 현재까지 SOA 어플리케이션을 위한 구체적이고 견고한 설계/구축 방법론이 널리 확립되지 못한 것도 같은 맥락에서 이해될 수 있다.

그림 8의 EMB(Enterprise Module Bus) 아키텍처는 서비스간 밀결합(tightly coupling)의 원인이 되는 모듈 호출 Flow를 Rule 기반으로 EMB 엔진에서 처리하고, 비즈니스 기능 모듈은 순수 비즈니스 로직만으로 구성되게 하여, 모듈의 조합만으로 서비스를 개발할 수 있도록 하는 새로운 방식의 SOA 어플리케이션 개발을 가능하게 한다.

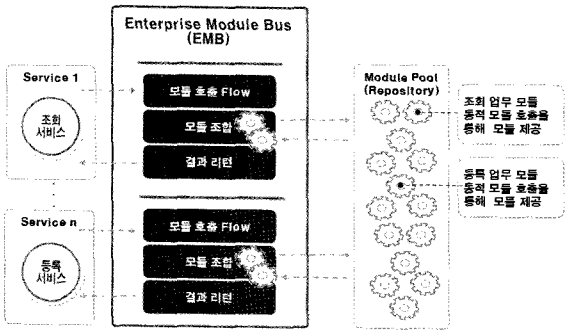

그림 8 EMB 아키텍처

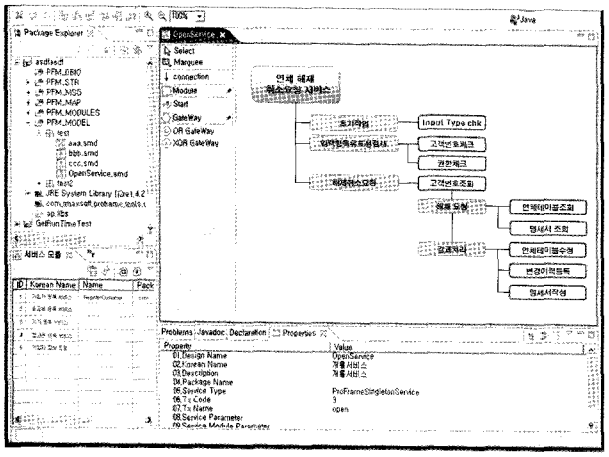

그림 9 서비스 처리흐름의 가시화

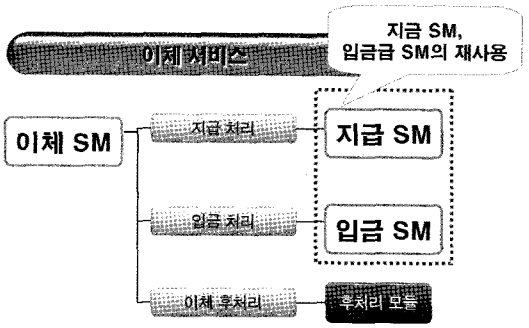

그림 10 복합서비스 개발 모형

또한 EMB는 서비스내 비즈니스 처리 흐름을 가시화하여 비즈니스 기능모듈의 조립/재구성, 재사용을 용이하게 하며, 서비스 내의 모든 수행 흐름을 모듈 버스에서 제어하기 때문에 Run time시 기능모듈 레벨의 모니터링이 가능하다.(그림 9)

EMB 기반 시스템에서 단위서비스를 결합하는 복합서비스(Composite Service)의 개발은 단위서비스의 재사용 및 조립만으로 개발이 가능하다.(그림 10)

비즈니스 프레임웍에서 제공해야 하는 또 다른 기능 컴포넌트는 다음과 같다.

- Notification :

한번의 통지(notification)로 모든 어플리케이션에 전달되는 기능. 다양한 채널 및 동기/비동기 통지 방식을 제공해야 한다.

- Service proxy :

서비스 구현의 복잡성과 서비스의 위치를 추상화하는 기능.

- Logging Services :

오류 또는 액티비티(activity)를 추적할 수 있는 서비스. 로깅 서비스는 전사적으로 공통적이고 표준화되어야 함.

- Exception handling :

예외사항(exceptions)을 처리하고 통지하는 메카니즘. 전사적으로 공통적이고 표준화되어야 함.

상기 기능 외에 비즈니스 프레임웍은 어플리케이션 서버에서 제공하는 트랜잭션 관리, 부하분산, Naming 서비스 등을 제공하거나 사용할 수 있어야 한다.

3.6 Data Framework

데이터 프레임웍은 서비스 계층(Service Layer)에 상위 수준으로 추상화된 데이터 스키마(data schema)를 제공하여 데이터 통합 효과를 제공하며, 전사 정보자원의 구성 방식에 유연성을 제공한다. 그림 11은 다양한 형태로 분산되어 있는 정보자원에 대한 통합된 뷰(view)를 제공하는 데이터 프레임웍 아키텍처를 보여준다.

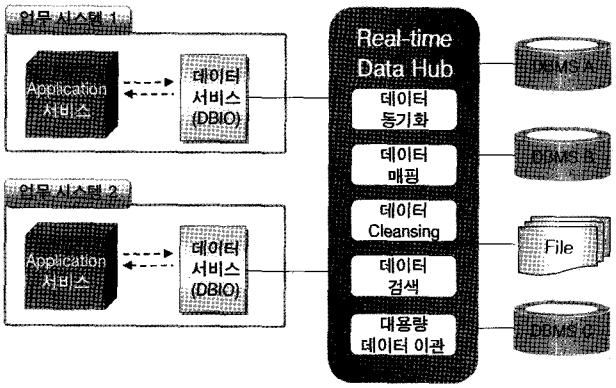

그림 11 데이터 프레임웍 아키텍처

어플리케이션 서비스는 표준 질의언어(Query Language)를 통해 원하는 데이터 자원에 접근할 수 있으며, Data Hub는 연결된 데이터 자원에 대하여 요청된 Query를 수행하고 그 결과를 서비스에 제공한다. Data Hub는 규칙기반의 데이터 동기화 기능과 표준화된 데이터 접근 서비스(DBIO)에서 요구되는 포맷으로의 데이터 매핑(mapping), Hub 레벨에서의 대용량 데이터 이관기능을 제공하여 서비스 계층에서의 데이터 처리를 단순화하고 빠른 처리성능을 보장한다. 또한 검색 프레임웍(Search Framework)을 제공하여 어플리케이션에서 공통적으로 자주 접근되면서 복잡한 질의조건(criteria)을 가지는 데이터에 대한 빠르고 효율적인 접근 기능을 제공한다. 즉 어플리케이션에서는 검색 조건(criteria)만으로 검색 프레임웍에 질의하게 되고, 검색 프레임웍 안에 정의된 해당 조건에 대한 query가 실행되어 정의된 형태로의 결과값을 어플리케이션에 제공하게 된다.

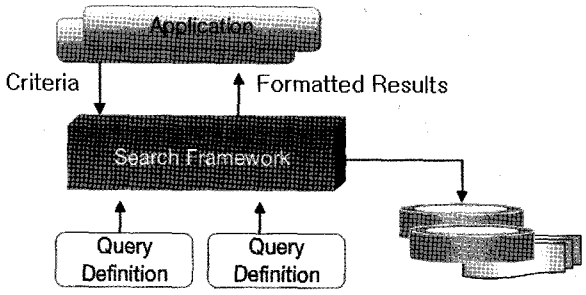

그림 12 Search Framework

4. 결 론

이상으로 서비스 지향 아키텍처를 구축하기 위해 요구되는 서비스의 특성을 도출하고, SOA 어플리케이션 구축의 핵심 인프라인 Framework의 아키텍처와 요구되는 기능에 대해 살펴보았다. SOA 기반 시스템 구축은 정보자산의 재활용, 정보 중복의 최소화, 유연한 시스템간 연계를 가능하게 함으로써 비즈니스 관점에서의 전략적 움직임(strategic move)을 용이하게 한다. 본 연구에서 살펴본 SOA 통합 프레임웍 아키텍처는 각 기업에서 서비스 지향 정보시스템을 구축할 때, 필요하게 될 참조 아키텍처(Reference Architecture)로서 유용하게 활용되어질 수 있을 것이다.

한 가지 유의할 것은 서비스 지향 아키텍처가 궁극적으로 모든 문제를 해결해주는 정보시스템 아키텍처는 아니라는 것이다. SOA 기반 서비스가 상태정보를 유지하지 못한다는(stateless) 특성은 현실적으로 많은 경우에 제약으로 작용한다. 또 SOA 프레임웍의 많은 추상화로 인한 성능상의 문제, 자원사용의 비효율성은 비용 증가로 귀결되고, 표준화 작업이 현재 진행형이라는 부분도 정보시스템 구축시 위험요인으로 관리되어져야 한다.

서비스 메타데이터(services metadata)의 관리 문제와 서비스간 상호연동을 위한 메시지의 범람 문제에 대해서는 앞으로의 연구과제로 남긴다.

〈약어 정리〉

SOA	Service Oriented Architecture
XML	eXtensible Markup Language
SSO	Single Sign-On
BPM	Business Process Management
ESB	Enterprise Service Bus
EMB	Enterprise Module Bus
FEP	Front-End Processor
EAI	Enterprise Application Integration
MCI	Multi Channel Integration
SLA	Service Level Agreement

참고문헌

[1] Dirk Krafzig, Karl Banke, and Dirk Slama. "Enterprise SOA", Prentice Hall, 2005

[2] OASIS, "Reference Model for Service Oriented Architecture 1.0" 2006. 8.

[3] 박대연, "차세대시스템 개발의 New Paradigm", TmaxSoft, 2006. 11.

[4] SOA Practitioners Guide Part 2: SOA Reference Architecture, 2006, 9.

[5] SOA Practitioners Guide Part 1: Why Service-Oriented Architecture?, 2006, 9.

[6] Service Orineted Architecture (http://en.wikipedia.org/wiki/Service-oriented_architecture), 2006.

[7] Erl, Thomas, "Service-Oriented Architecture: Concepts, Technology, and Design." Prentice Hall, 2005.

〔8〕 Gamma, Erich / Helm Richard, "Design Patterns" Addison-Wesley, 2003.10. 9.
〔9〕 Shaw, Mary/ Garlan, David, "Software Architecture : Perspectives on an Emerging Discipline", Prentice Hall, 1996.
〔10〕 Bass, Len/ Clements, Paul/ Kazman, Rick, "Software Architecture in Practice", Addison-Wesley, 2003.
〔16〕 TmaxSoft, Internal technical materials

김 성 익

1994 한국과학기술원 기계공학과(공학사)
1996 한국과학기술원 경영공학전공(공학석사)
1996~2002 동양시스템즈 기술연구소
2002~현재 TmaxSoft EA컨설팅 그룹장
관심분야 : S/W 아키텍처, 웹서비스, Ubiquitous computing
E-mail : sothis@tmax.co.kr

박 정 일

1989 중앙대학교 회계학과(경영학 학사)
1995 중앙대학교 MIS전공 (경영학 석사)
1982~현재 신한은행 IT서비스부 팀장
관심분야 : S/W 아키텍처, 전문가시스템
E-mail : jipark@shinhan.com