서비스 지향 아키텍처의 기반 기술과 구축 사례에 대한 연구

Research for SOA (Service Oriented Architecture) implementation case study and its base technologies

조재훈[a], 이상완[b]

[a]한국 아이비엠 소프트웨어 사업부
서울 강남구 도곡동, 674-12, 군인공제빌딩 16층
Tel: +82-11-898-6235, Fax:+82-2-3781-5151, E-mail:jhcho@kr.ibm.com

[b]동아대학교 공과대학 산업경영공학과
부산 사하구 하단2동 840, 604-714
Tel: +82-51-200-7692, Fax:+82-51-200-7697, E-mail:swlee@dau.ac.kr

Abstract

In order to succeed in today's changing business environment, the enterprise should adapt their process flexible. To increase their business process flexibility, they need to establish and end-to-end integration internally as well as among partners. Now that SOA (Service Oriented Architecture) is accepted as an upcoming IT standard to support flexible business processes and integration. In this paper, SOA is introduced. And the Web services and ESB (Enterprise Service Bus) are also introduced as a foundation technology to implement SOA-enabled solutions. The pilot project, which introduced in this paper, will be a good reference for future SOA study.

Keywords:
SOA, Web Services, ESB, System Integration

서론

기업의 사업 환경이 점점 복잡해지고 변화함에 따라 기업의 계속적인 성장을 위해서는 업무 절차나 흐름을 수시로 변화시킬 수 있는 융통성 있는 체제를 갖추어야 한다. 그 동안 기업의 IT 시스템들은 업무별로 별개로 구축되어, 이의 통합이 필요하게 되었다. 정보 시스템 간의 통합이 원활히 이루어 지지 않을 경우 융통성 있는 업무 체제 구축에 방해 요소가 되어, 통합의 표준이 필요하게 되었다. 그림1은 현재 기업들이 처한 업무 흐름의 예로서, 외부 환경의 변화에 따라 기업의 업무 흐름이 수시로 변해야 하며, 사용자와 공급자 등 모두를 고려한 시스템 통합 모델이 필요함을 알 수 있다. 이러한 시스템 통합을 위해서는 아래와 같은 몇 가지 "통합의 필수 요건" 이 필요하다.

- 상호 운영: 이 기종 하드웨어, 운영

환경, 프로그래밍 언어, 응용 소프트웨어간 상호 공동 이용 가능하여야 한다.

- 융통성: 비즈니스 환경 변화에 따른, 변경에 적절히 대응할 수 있어야 한다
- 단순화: 통합 기술은 간단하고 저렴해야 하며, 단순해야 한다.
- 민첩성(Agility): 비즈니스의 변경에 따라 신속하고 쉽게 변화를 수행하여야

통합의 필수 요건 중 가장 중요한 요소는 공동 이용 가능성이며, 이는 특정 밴드나 플랫폼에 종속되지 않는 기술이어야 한다. 아울러 기존의 시스템 투자를 보호할 수 있어야 하고, 개발자의 현재 보유 기술을 계속적으로 활용하고 응용할 수 있어야 한다.

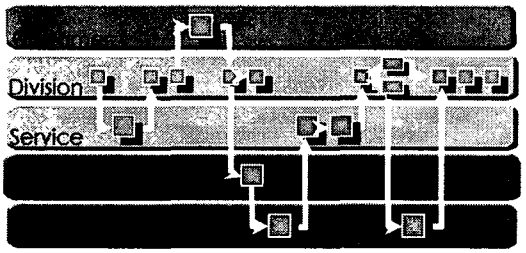

그림 1 기업들이 처한 업무의 흐름 [11]

본 논문에서는 앞에서 열거한 이러한 문제를 해결하기 위한 해결책으로, 전세계적으로 활발히 논의되고 있는 SOA (Service Oriented Architecture)에 대해서 여러 발표 자료를 바탕으로 소개하고, 이의 가장 중요한 기반 기술인 웹 서비스와 ESB (Enterprise Service Bus)에 대해 정리하고, SOA를 구축한 사례를 소개한다.

SOA (Service Oriented Architecture)

아키텍처란

아키텍처란 완성된 구조물의 전체적인 모양뿐 아니라 그 조각 하나하나가 어떻게 연계되어 있는지를 나타내는 개념이다. 구조물이 클수록 각 조각들간의 연관성이 더 복잡하다. 아키텍처의 범위는 다양한데 개인의 집 한 채에서부터 공항 이나 건물 전체 또는 도시 전체를 하나의 아키텍처로 생각해 볼 수 있다. 서비스 아키텍처는 도시 전체를 계획하는 것과 같다고 생각할 수 있으며, 빌딩 하나에 초점을 맞추는 것이 아니라 도시 전체의 연결 및 연계와 밀접한 관련이 있다. 소프트웨어 아키텍처는 소프트웨어 구성 요소가 무엇인지를 정의하고 이들 구성 요소 각각의 기능을 설명한 것이며, 구성 요소들 간의 연결 관계뿐 아니라 각 구성 요소들의 기술적인 구조, 제약 사항, 특징 등을 설명한다. 아키텍처는 시스템에 대한 청사진을 제시하는 것으로서 구축 시 참조해야 하는 높은 수준의 암묵적 계획이다. [2]

아키텍처 층(層)

통합 측면에서 IT 아키텍처는 그림 2와 같이 4단계 층으로 구분해 볼 수 있다.

- Linux, UNIX, J2EE, .Net과 같은 IT구현 기술과 관련된 기술 층
- ERP, CRM, HR등과 같은 응용 소프트웨어 층
- 주문, 고객, 사원 등 서비스 별로 구분한 서비스 층
- 비즈니스 프로세서 흐름에 관한 업무

절차 층

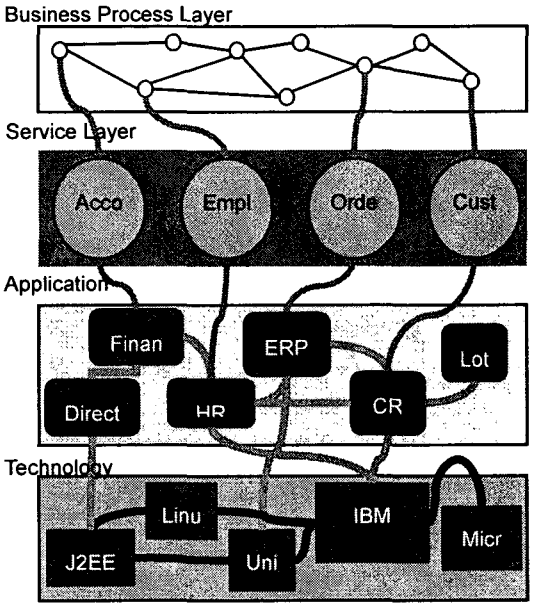

그림 2 아키텍처 층 [12]

통합은 모든 층에서 수행되지만 추구하는 방식이나 내용은 층마다 다르다. 기술 층에서의 통합은 물리적인 기술의 연결이고, 응용 프로그램 층에서의 통합은 구현된 응용 프로그램이나 구성 요소의 연계를 의미한다. 이 단계에서의 통합 과제는 SAP 자료를 Siebel로 어떻게 변환하느냐와 같은 어떤 특정 사실에 초점을 맞추어야 한다. 이 단계에서는 어떤 변화가 일어나면 통합 시스템도 함께 변경해야 한다. 반면 서비스 층에서의 통합은 자원들로부터 독립적이어서, 고객과 주문을 어떻게 연결하느냐와 같이 각 서비스 단위에 초점을 맞춘다. 마지막으로 업무 처리 층에서의 통합은 구현된 응용 프로그램이나 기술과는 상관없이 서비스들만을 고려 한다. 아키텍처 층을 분리하는 가장 중요한 이유는 변경에 따른 영향을 최소화하자는 것이다. 어떤 참여자도 다른 참여자의 기술적 변경에 따라 변경되어서는 안 되도록, 서비스 층에서의 통합이 매우 중요하다. 그래서 웹 서비스 기술을 이용한 느슨한 결합 방식을 사용한다. 서비스 층은 응용 소프트웨어의 제공이나 요청으로부터 자유로울 수 있도록 한다. 서비스 요청자는 내부 시스템이 어떻게 구성 되어있는지에 관심이 없다 다만 제공된 서비스에만 관심을 가질 뿐이다. 마찬가지로 서비스 제공자도 내부에 숨겨진 응용 소프트웨어 등에는 관심이 없다. 서비스란 비즈니스의 논리적 단위이다. 서비스 단위들간에 흐름을 통제하거나, 자료 형식을 변환시키는 노력이 필요하므로, 이들 서비스 단위들을 비즈니스의 논리적 흐름과 분리할 필요가 있다. 서비스들은 서비스 제공자와 요청자 간의 연속적인 네트워크 관계를 가진다. 이런 서비스가 이행되는 과정들의 네트워크 집합이 SOA라고 생각해 볼 수 있다. [12]

SOA의 정의

Mark Colan의 정의에 의하면 서비스 지향 아키텍처란 "서비스를 요청하거나 제공하여 응용 프로그램 기능이 가능하도록 하는 정책이나 관례 또는 골격이다" [8] SOA는 데이터와 애플리케이션을 표준 블록 단위로 나눠 하나의 서비스로 구성한 뒤 웹 서비스 기술 등을 적용해 각 서비스를 조합 또는 재사용할 수 있게 한다 내부 애플리케이션과 서비스를 통합하는 것은 물론이고 협력사 · 하도급 업체 등의 외부 시스템까지 연계할 수 있다.

SOA를 적용할 경우 얻을 수 있는 기대 효과로는 크게 개발 및 유지 비용의 감소와 비즈니스 환경 변화에 대한 적응 속도 증가를 들 수 있다. SOA는 비즈니스를

여러 측면에서 변화시킨다. 비즈니스 구성 방법, 어플리케이션 구축 및 관리하는 방법, 조직이 이용하는 기술 등 모든 면에서 효율적인 방식으로 변화가 생긴다. 서비스 형태로 구성된 어플리케이션을 통해, 그 기능을 다른 비즈니스를 위해 재사용한다. 어플리케이션의 생명주기를 늘리고, 새로운 기능을 개발하는 비용을 줄임에 따라 기존 응용 소프트웨어의 ROI (Return on Investment)를 증가시킨다. 서비스로 구성된 기존 어플리케이션은 새로운 어플리케이션과 잘 연계하여 실제 시스템에 배치되어 보다 빠르게 최종 사용자에게 서비스를 제공한다. 이러한 서비스 민첩성으로 인하여 기업은 새로운 제품과 서비스에 대한 경쟁력의 기반이 되며 변화하는 비즈니스 환경에 신속히 대응할 수 있다.

SOA에 대한 전망

Gartner 자료에 의하면, 웹 서비스에 기반한 SOA로의 변화는 과거 단말기 기반 아키텍처에서 Client/Server 아키텍처로의 변화에 견줄만한 사고의 틀을 변화시킬 것을 예견하고 있다. 다음은 Gartner의 예측 자료이다. [6]

- 2008년까지, 새로운 개발 프로젝트의 80%는 SOA일 것이다. (0.8 probability).
- 2008년까지, SOA는 코드의 재사용 율을 100%이상 증가시킬 것 이다. (0.8 probability).
- 2010년까지, 소프트웨어 매출 성장의 80%는 SOA에 기반한다 (0.7 probability).

웹 서비스 (Web Services)

웹 서비스는 플랫폼이나 벤더에 무관한 개방형 표준을 지향하는 기술로써, 인터넷 네트워크를 통해 다수의 비즈니스 시스템들을 표준화된 기술로 결합시켜 고객이 원하는 정보나 응용 기능을 구현하는 기술이다. 웹 서비스는 기존의 다른 소프트웨어처럼 그 기능에 대한 완벽한 정의를 통하여 구성하는 것이 아니라, 고객의 다양한 요구에 맞춰 서로 주고 받는 데이터 표준에 대한 정의를 규정한다. 이와 같은 열린 정의를 통해서 매우 유연하고 이질적인 운영 시스템, 프로그램 언어간의 의사 소통의 차이를 극복해 주는 역할을 한다. Mark Colan의 정의에 따르면, 웹 서비스는 SOAP이나 HTTP와 같은 표준 네트워크를 통하여 전달할 수 있는 WSDL로 쓰여진 소프트웨어 구성 요소이다. [9] 웹 서비스는 IBM, Microsoft 및 몇몇 회사에 의해서 개발되어 W3C(World Wide Web Consortium)나 OASIA (The Organization for the Advancement of Structured Information Standards)와 같은 표준 단체에서 이에 대한 표준화 작업을 수행하고 있다. 그림3은 웹 서비스의 기본 개념과 기본 기술을 설명한 그림이다. 웹 서비스는 서비스 제공자, 요청자, 중개자 간의 세 가지 오퍼레이션 (Publish, Find, Bind)을 통하여 정의된다. 이때 각각 WSDL (Web Services Description Language), UDDI (Universal Description, Discovery, and Integration), SOAP (Simple Object Access Protocol)과 같은 웹 서비스 표준 기술들이 사용되고 있다.

웹 서비스는 어떤 어플리케이션 서비스를 중재자에 등록하고 등록된 서비스를 공급자로부터 요청자로 실제 전달하는 과정을 통하여 구현된다. WSDL과 UDDI는 어플리케이션 서비스를 중재자에 등록하기 위한 표준 기술이고, SOAP는 등록된 어플리케이션 서비스를 서비스 공급자로부터 서비스 요청 자에게 실제 전달하기 위한 표준 기술이다.

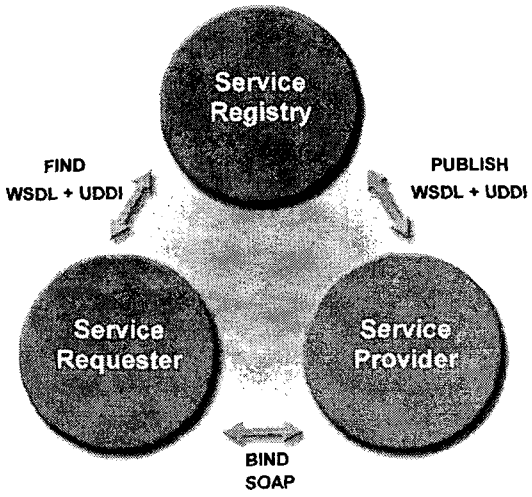

그림 3 웹 서비스 기본 개념과 기술 [3]

ESB (Enterprise Services Bus)

ESB의 개념

ESB는 SOA를 구현하기 위한 새로운 개념의 미들웨어로, 여러 장소에 걸쳐 다양한 어플리케이션 유형간 최적의 방식으로 정보를 배포할 수 있는 아키텍처 패턴이다. ESB 패턴은 메시지 지향적, 이벤트 중심적, 그리고 서비스 지향적 통합 방식에 기반을 두고 이 방식들을 하나로 묶는다.[10] 서비스 제공자와 서비스 사용자를 직접 연결하지 않고 ESB를 통하여 연결하여, 필요에 따라, 소비자가 알아차리지 못하는 사이에, 한 서비스 제공자를 다른 서비스 제공자로 대체할 수 있게 한다 실직적 의미에서 ESB는 원활하게 상호 연동하는 분산 통합 서버(허브)의 집합으로서, 과거에는 융통성 있고 쉽게 변경 가능한 방식으로 연동하지 못했던 어플리케이션들의 통합을 목표로 다양한 보완 서비스를 제공한다. ESB를 개념적으로 살펴보면 그림 4와 같다.

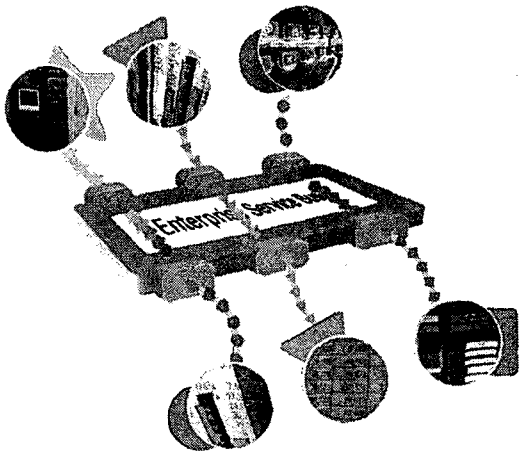

그림 4 ESB의 개념

ESB 개념 중 가장 중요한 하나는 어플리케이션 및 그 개발자가 인프라상의 서비스 또는 영구적인 리소스의 위치에 신경 쓸 필요가 없다는 것이다. 즉 ESB가 있으면 이러한 서비스 및 리소스에 더 쉽게 그리고 가급적 투명하게 연결할 수 있으며, 따라서 이기종 환경 통합에 필요한 수고가 줄어든다. ESB는 새로운 상호 운용성을 제시하며, 정보가 필요한 시점에 이를 필요로 하는 사람에게 소통되게 함으로써 의사 결정에서의 대응력 및 정확성을 높인다.

ESB의 특징

ESB는 단순히 서비스들을 연결시켜주는 제품이 아니고, 개념적인 구성으로서,

서비스들이 연결되는 SOA 고속도로이다. 이 고속도로는 지역이나, 운송 방법이나 수단, 이 기종 플랫폼도 모두 연결 가능하고, 다양한 프로토콜의 사용이나 전환이 가능하도록 하여준다. ESB는 아래와 같은 특징을 가지고 있다. [1]

- 보편성 (Pervasiveness)
- 표준 기반 어플리케이션 통합
- 고도의 분산 통합 환경
- 변환 (Transformation)
- 확장성 (Extensibility)
- 이벤트 기반 SOA
- 업무 절차의 흐름(Process Flow)
- 보안과 신뢰성
- 자율성과 원격 구성 관리
- XML 기반 데이터 형태

ESB에서 구현되는 것들

ESB는 엔터프라이즈 내부 또는 엔터프라이즈 하위 내부에서 전개되는 상황을 설명해야 한다. 그리고 가용 리소스 및 흐름을 나타내고 표현할 수 있도록 시각적으로 설명할 수 있어야 한다. ESB는 소스, 대상, 데이터베이스, 논리적 처리 등 다양한 노드간의 흐름을 설명할 수 있어야 한다. ESB에서 더 많은 사항을 설명할 수 있으면 더 많은 기능이 자동화 된다. 적용 가능한 리소스 및 네트워크 대여 폭을 설정하여 시스템의 위치를 정하고 잠재적으로 문제가 발생할 영역을 찾아내고 이 문제 영역을 어떻게 예방하거나 해결할 것인지 결정할 수 있게 해야 한다.

기존 어플리케이션이 상호 연결되는 방식을 바꾸거나 새로운 어플리케이션을 도입해야 하는 경우, ESB는 중요한 역할을 할 수 있다. 유연하고 관리 가능한 방식으로 어플리케이션을 연결하는 일은 앞으로도 중요한 과제로 남을 것이다. 대부분의 기업에게 있어 그러한 통합을 더 쉽고 저렴하게 실현하는 것이 가장 큰 숙제이다. 어플리케이션 X가 어플리케이션 Y와 통신하게 할 수 있지만, X(또는 Y)가 N개의 다른 어플리케이션과도 통신할 수 있다면 더 큰 이익을 누릴 수 있다. 이러한 기능을 더욱 간단하게 구현하고 작동시키는 것이 성공적인 ESB구현의 핵심이다. [10]

사례 연구

본 사례는, 모 항공사의 어떤 업무에 SOA 개념을 적용시켜, 실 업무에 적용할 수 있는지 평가해 보기 위한 프로젝트이다.

프로젝트의 배경과 범위

사례에서 선정한 SOA개념을 적용한 업무는 항공 램프 운영 시스템이다. 그림 5와 같이, 램프 운영 시스템은 비행기 운항에 필요한 비행기 기종 서비스, 고객 관련 정보, 비행 운항, 승무원 관리, 탑승이나 화물 등 비행에 관련된 모든 정보가 포함되었다.

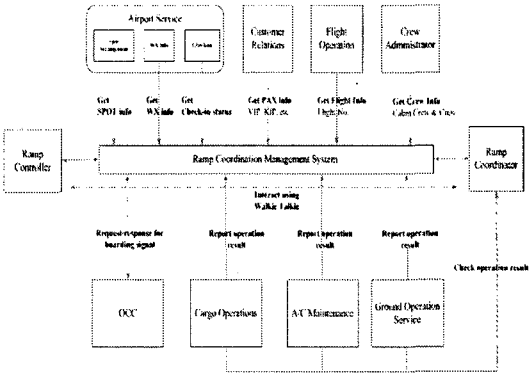

그림 5 항공 램프 관리 시스템. [7]

프로젝트의 구축 전 준비 과정

구축 전 준비 과정에서는 고객의 요구 사항을 정확히 분석하는 것이 중요하다. 본 프로젝트에서 가장 중요한 고객의 요구 사항은 '램프 관리 시스템 관련 모든 정보를 실시간으로 공유하고, 기존 시스템에 대한 투자를 보호하라' 는 것이었다.

프로젝트 구축 과정

파일럿 프로젝트 구축은 SOA구축 4단계 방법을 적용하였다. 비즈니스 모델을 SOA가 요구하는 서비스 모델로 변경하기 위하여 SOMA (Service Oriented Modeling Architecture)라는 체계화된 검정된 방법론을 사용하였고, 각 서비스들을 하부 목표와 연계시킨 후 각각 비즈니스 목표와 연계시켰다. ESB 인프라 선정을 위해서는, 필요한 몇 가지 옵션들을 나열한 후, 각자의 장단점을 살펴 WAS 5.x + WBI Event Broker + MQ를 선정하였다. 본 파일럿 프로젝트에 적용된 SOA구축 4단계 방법은 다음과 같다.

Step 1: 현재의 비즈니스 프로세서를 세부 단위 업무로 분할한다. 이때 분할된 단위 업무를 요소(element)라 하고, 각 요소에 대하여 응용 소프트웨어에 의하여 충족된 비즈니스의 기능에 대하여 정의하고, 프로세서에 의하여 사용되는 데이터가 무엇인지 확인하고, 시스템에 의하여 제공되는 서비스는 무엇인지, 각 구성 요소의 사용자가 누구(사람 또는 다른 응용 소프트웨어)인지를 확인 정의한다.

Step 2: 앞에서 정의한 요소들을 서비스화 한다. 서비스화의 의미는 서비스의 사용자와 소비자를 정의하고, 웹 서비스 기술을 적용하여 새로 개발하고, 이미 사용중인 업무는 웹 서비스로 변환시킬 수도 있다. 이때 모든 서비스는 WSDL 인터페이스로 생성된다. 물론 이때 서비스에 의하여 사용된 데이터의 구조는 XML에 의하여 정의된다

Step 3: 서비스들을 ESB와 연결한다. 물론 ESB는 이미 설치되어 있어야 한다. ESB는 IBM등에서 제공하는 여러 제품의 기능 등을 검토하여 적절히 선정되었다고 가정한다. 이때 ESB는 서비스 요청자의 위치, 수송 방법, 위치나 시간 또는 형식 등에 상관없이 신뢰성 있는 메시지를 전송 및 중계하게 한다.

Step 4: 서비스들을 결합, 통합하여 SOA를 완성한다.

프로젝트 구축 후 결과물

프로젝트 구축 후 완성된 결과물 전달 및 교육이 진행되었다. 프로젝트의 결과물로는 관련 서류와 운영 가능한 PoC (Prove of Concept)시스템이 전달되었다. 서류에는 아키텍처 설계 문서, 서비스 모델 문서, 구성 요소 설계 문서가 포함되었으며, PoC 시스템에는 선택된 파일럿 프로세서, 사용자 인터페이스 서비스, 응용 프로그램 서비스, ESB 등이 포함되었다. 프로젝트의 결과, 비즈니스의 목표에 잘 부합된 SOA 파일럿 시스템을 성공적으로 구축하였으며, 축적된 경험과 기술을 바탕으로 향후 다른 업무에 추가 적용할 수 있는 기반을 갖추었다. SOA를 구축하면, 외부 환경이나 시장의 변화에 민첩하게 대응할 수 있고, 작성된 서비스들은 재 사용 가능하여 관리가 용이하고, 비용 효과적임을 알 수 있었다.

결론

사례 연구에서 살펴본 바와 같이, SOA개념을 적용한 비즈니스 프로세서를 실제 업무에도 적용할 수 있었다. 또 기존 투자를 보호하면서 SOA를 구축할 수 있음을 확인하였고, 재 사용 가능한 서비스의 활용으로 기업은 외부 환경의 변화에 민감하게 대처하고 향후 쉽게 확장 가능하여, 비즈니스 목표에 잘 연계된 IT를 구축할 수 있었다.

비즈니스 통합의 필요성 증가에 따라, 통합을 서비스 중심으로 생각하는 것은 분명히 앞서나간 개념이고 상당히 큰 변화임에 틀림없다. 개별 어플리케이션의 통합 개념을 뛰어넘는 서비스 지향 아키텍처가 향후 IT의 통합 표준이 될 수 있으리라 생각한다.

참고 문헌

Types of references are as follows:

- For a Book, see [1], [2], [3] and [4]
- For a Journal Article, see [5]
- For a Gartner research report, see [6]
- For a IBM paper, see [7],[8],[9],[10],[11]
- For a Web site, see [12]

[1] Dave Chappell, (2004) "Enterprise Service Bus", Publisher: O'Reilly.

[2] Dirk Krafzig, Karl Banke, Dirk Slama, (2004) "Enterprise SOA: Service-Oriented Architecture Best Practices" The Coad Series, Prentice Hall Professional Technical Reference.

[3] Eric Newcomser and Greg Lomow, (2005) "Understanding SOA with Web Services", Addison Wesley.

[4] Thomas Erl (2004) "Service-oriented Architecture; A field guide to integrating XML and Web Services", *PTR.*

[5] Pierpaolo Baglietto, Massimo Maresca, Andrea Parody, and Nicola Zingirian, (2005) "Stepwise deployment methodology of a service oriented architecture for business communities", Information and Software Technology, Italy, Vol 47, pp 427-436.

[6] Simon Hayward, (2005) "Positions 2005: Service-Oriented Architecture Adds Flexibility to Business Processes", Gartner research, ID number:G00126409.

[7] Xin Sheng Mao, Lead Architect, (2005) Manager of China SOA Design Center, Service Oriented Architectures Architecture Workshop

[8] Mark Colan, (2004) "Service Oriented Architecture and Web Services", IBM, Technical Presentation Material

[9] Mark Colan, (2004) "A Technical Overview of Web Services", IBM, Technical Presentation Material.

[10] Stephen Todd, (2005) Understanding Web Services on ESB, IBM ON Demand solution white paper.

[11].Daniel Sabbah, (2005) "IBM Executive SOA Summit" IBM presentation material.

[12] www.cbdiforum.com, "CBDI insight for Web Services & Software Component Practice", Tutorial: Service Orientation and Web Services