2007년도 행내 현상논문

SOA를 활용한 정보시스템의 운영 효율성 제고 방안

- 당행 정보시스템에 서비스지향 아키텍처(SOA) 도입을 고려하기 위한 실질적 연구 -

전산정보국 품질관리반 조사역 양희정

< 차 례 >

<요 약>

< 요 약 >

그 동안 당행은 경영관리, 회계결제, 대외연계, 조사통계, 문서관리, 경제교육 등 모든 업무영역에서 다양한 플랫폼과 기술을 바탕으로 40여개 정보시스템을 구축 운영하고 있다. 시스템은 매우 안정적으로 운영되고 있으나 결재처리, 권한관리, 법규정보, 검색엔진, 코드관리, 설문조사 등과 같은 기능은 여러 시스템에서 각각 별도로 개발하여 중복 사용되고 있다.

정보통신 기술발전과 더불어 재사용을 통한 효율적인 시스템 개발을 목적으로 널리 사용되는 컴포넌트 활용기술도 아직까지는 플랫폼에 종속적인 한계를 가지고 있다. 따라서 상대적으로 연관도가 높은 동일한 환경에서 재사용성은 뛰어나지만, 서로 다른 환경에서 개발된 컴포넌트 재사용에는 상호운용의 문제가 발생한다.

이러한 정보시스템 재사용의 문제점을 보완하기 위한 해결책으로 서비스 지향 아키텍처인 SOA가 대두되었다. SOA는 업무 프로세스를 구성하는 개별 기능을 서비스 단위의 독립적인 기술로 유연하게 개발하고 웹서비스 기반의 기술표준을 지원하는 아키텍처이다. 서비스 단위로 재사용 및 공유가 가능한 SOA는 이기종 시스템 간 통합 구축에 적합한 구조로 되어 있다.

본 논문에서는 SOA의 정의 및 관련 기술표준으로 알려진 SOAP, UDDI, WSDL, BPM, ESB, EMB 등에 대해 살펴본 후 이와 관련된 구현 방법과 효과를 기술했다. 당행의 정보시스템 개발 및 운영 현황을 개략적으로 관찰하고 서비스 지향 관점에서 개선과제를 도출하였다. 다음으로 국내 기업의 SOA 구축사례 분석 결과를 참고하여 당행의 SOA 기반 정보시스템 구축을 위한 로드맵과 목표를 설정하고 실증적으로 효과를 분석하기 위해 파일럿시스템을 구축하여 SOA 프레임 워크와 방법론을 제시하였다.

SOA의 도입은 그 자체가 비즈니스의 목적이 아니라 효율적인 비즈니스를 위한 수단이며, 무작정 도입하기 보다는 IT자원의 상호 운용성과 재활용성 증대를 통한 비용절감 등 도입 목적을 분명히 한 후 추진하여야 한다.

당행의 경우 SOA 기반으로 정보시스템을 구축함으로써 시스템의 유연성을 확보하여 시스템간 서비스를 공유하고 IT 조직에 필요한 정보 습득기간을 단축 시킬 수 있다. 통합 환경에서는 시스템별 운영을 통합 유지정비 체제로 전환하는 것도 가시화될 수 있으며, 중장기적으로 IT 인프라에 대한 효율성도 제고할 것으로 기대된다.

Ⅰ. 서론

현재 전산 환경은 다양한 플랫폼과 기술을 기반으로 급속한 진화를 거듭하고 있으며, 비즈니스 환경 또한 과거 독립적인 조직에 의해 주도되는 수직적 관계에서 프로세스 기반의 수평적 관계로 변화하고 있다. 이러한 급변하는 시장요구에 민첩하게 대응하기 위해서는 고객, 공급자, 파트너 등 다수 기업과 협업관계가 중요하며 기업 생존을 위한 경쟁력 확보를 위해 업무효율을 극대화시켜야만 한다.

예컨대 회계결산, 인사급여, 자산관리 등 기업 활동의 대부분 업무가 정보시스템으로 구축되어 있지만 복잡 다기화 된 IT자원 환경에서 이를 통한 업무 효율을 개선하기 위해서는 시스템의 재사용과 통합, 비즈니스 프로세스 관리와 프로세스 통합이 절실히 요구된다. 여기서 재사용은 기존의 코드, 객체, 컴포넌트, 투자자원 등을 개발업무에 재활용하는 것이고, 시스템 통합은 이기종 분산 시스템 및 어플리케이션간의 통합을 나타낸다. 즉, 비즈니스 프로세스 관리와 통합은 기업 환경의 변화에 따라 비즈니스 프로세스를 수시로 융통성(Flexibility)있고 민첩성(Agility)있게 변화시킬 수 있는 정보화 능력을 의미한다.

지금까지 대부분의 전형적인 어플리케이션 개발 기법에서 비즈니스 프로세스는 어플리케이션의 코드 속에 숨겨지게 되었다. 구현된 비즈니스 프로세스를 파악하기 위해서는 코드를 직접 파악해야 하고, 새로운 비즈니스 요건 반영을 위해 프로세스를 변경하려면 코드 수준에서 어플리케이션 내의 의존관계를 따져 조심스럽게 변경하여야 한다. 이것은 비즈니스 프로세스 간에 밀접한 결합 관계를 형성하게 된다. 그 결과 프로세스를 잘 아는 현업 담당자들은 프로세스의 구현과정에서 멀어지고, 개발자는 어플리케이션 개발 및 변경에 많은 시간과 노력을 필요로 하게 되며, 결국 비즈니스 요건의 변화에 기업 정보시스템이 따라가지 못하는 문제가 생긴다.

어플리케이션을 환경변화에 유연하게 개발하기 위한 노력은 객체지향 개발방법론 이후 코드의 재사용 강화 방법을 중심으로 연구되기 시작했다. 그리고 객체에서 진화하여 객체의 다형성과 상속성을 배제하고 인터페이스를 통해 소스코드가 아닌 이진코드 형식으로 재사용할 수 있는 컴포넌트기반 개발방법론(Component Based Development, CBD)이 등장했다. 정보기술 환경이 발전하고 시스템의 요구사항이 복잡해짐에 따라 분산 환경의 시스템 간에 일부기능을 공유하는 형태로 시스템이 발전하고 있으며, CBD를 통해 소프트웨어가 개발됨에 따라 시스템의 기능 모듈들도 컴포넌트로 개발되고 있다. 분산 환경[1]에서의 컴포넌트는 특정 기술에 종속적이면서 상대적으로 연관도가 높아(Tightly Coupled) 동일한 플랫폼이나 프레임워크 내부에서의 재사용성은 매우 뛰어나다.

---

1) CORBA(Common Object Request Broker Architecture), DCOM(Distributed Component Object Model), COM+(Component Object Model+), EJB(Enterprise JavaBeans), J2EE(Java2 platform, Enterprise Edition), .NET 등을 기반으로 함

그러나 이기종 환경에서 개발된 기존 컴포넌트들을 재사용하고자 할 때는 플랫폼과 구현 언어에 의존성이 강하여 상호운용의 문제가 발생한다는 단점이 있다. 최근에는 이러한 컴포넌트를 보완하기 위한 소프트웨어 아키텍처로 구현 기술로부터의 독립성과 유연성이 높고 재사용성뿐만 아니라 이기종 시스템간 통합 구축에도 적합한 서비스 지향 아키텍처(Service Oriented Architecture, SOA)에 대한 관심이 집중되고 있다. SOA는 표준화된 인터페이스와 함께 XML 기반의 인터넷 표준메시징 프로토콜을 이용하여 느슨한 결합(Loosely Coupled)연결 방식을 통해 플랫폼에 유연하면서 표준화된 아키텍처와 개발 방법을 제공한다.

SOA에서는 프로세스를 업무 영역과 프로세스 영역으로 나누어, 각각 '비즈니스 서비스'와 '프로세스 서비스'로 구현한다. 비즈니스 서비스는 프로세스 모델에 의해 표현되는 프로세스 로직을 반영한 컴포넌트를 포함하고, 프로세스 서비스는 비즈니스 서비스간의 통합과 제어(Process Orchestration)를 하는 공통 인프라로 작용한다. 비즈니스 서비스가 실제 비즈니스 로직을 제공한다면, 프로세스 서비스는 통합을 위한 메시지 처리와 프로세스 오케스트레이션 즉, 언제, 어디서, 누구를, 어떻게 통합해야 하는 지 같은 제어 역할을 담당하게 된다. 응용 프로그램을 프레젠테이션-비즈니스 로직-데이터 계층으로 나누어 각각 독립적으로 구성한 3계층(3tier) 아키텍처에서 더 나아서 비즈니스 로직을 비즈니스 서비스와 프로세스 서비스로 분화시킨 것이다. 각 서비스들은 느슨하게 연결된 인터페이스를 갖는 독자적인 시스템으로서 서비스 사이에 발생하는 상호 작용이 단순화되기 때문에, 기존 서비스의 비즈니스 로직은 얼마든지 추가, 변경 및 확장될 수 있는 것이다.

1996년 가트너 그룹(Schulte & Natis)에 의해 SOA가 소개된 이래로 특별한 주목을 받지 못하다가 SOA 개념을 실제로 구현하는 웹서비스가 등장하면서 근래에 다시 SOA는 중요한 패러다임이자 소프트웨어 아키텍처로 주목받고 있다. 그렇다고 웹서비스가 SOA를 실현시킨 유일한 기술은 아니다. 제한적인 범위 내에서 사용되기는 하지만 분산컴퓨팅이나 RPC(Remote Procedure Call) 기반 미들웨어 시스템들(CORBA, DCOM, RMI 등)도 SOA 개념을 구현한 기술이 될 수 있다.

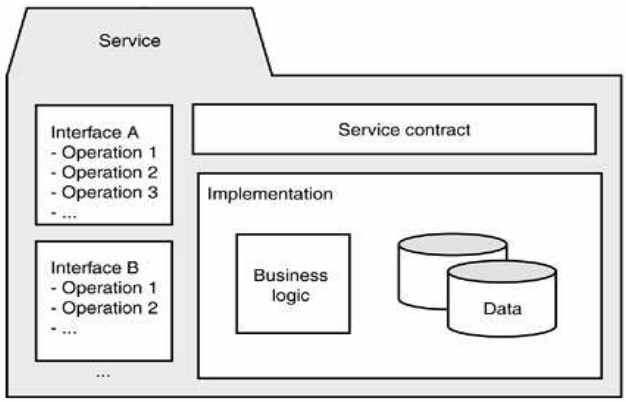
본 논문에서는 SOA 개념 및 SOA 관련 공개표준기술(Web Service, Business Process Execution Language, Web Service Description Language, Simple Object Application Protocol, Enterprise Service Bus)에 대해 살펴보고, SOA 구축사례를 참고하여 당행 정보시스템의 현황을 서비스 관점에서 분석하고 SOA기반 정보시스템 운영환경 구축을 위한 로드맵을 제시하고자 한다.

II. SOA에 대한 이론적 고찰

1. SOA 개요

가. SOA 정의

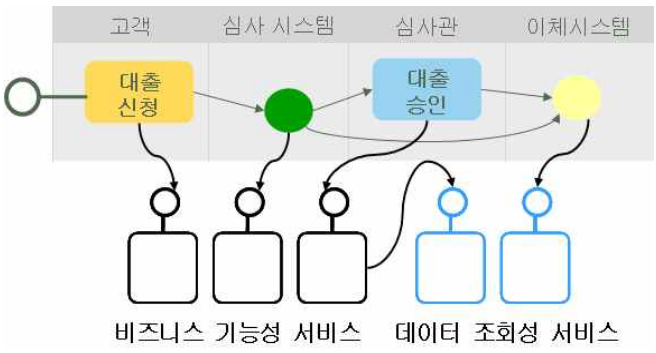
서비스 지향 아키텍처(SOA : Service Oriented Architecture, 이하 "SOA"라 표시)에서 언급되는 '서비스'는 반복 사용이 가능한 비즈니스 기능으로 정의할 수 있다. 서비스는 개별 기능을 수행하는 단위(어플리케이션을 구성하는 기본)로서 다른 서비스와는 독립적인 작업 로직을 담고 있는 업무단위를 말한다. 기업에서 SOA를 도입할 때 제일 먼저 해야 하는 것이 기업 내의 업무를 서비스라는 단위로 나누는 일이다. 예를 들면, 고객 신용도 조사, 신규 계좌 개설, 항공권 예약 등을 서비스로 정의할 수 있다.



[그림 1] 서비스의 정의

[그림 1]은 2004년 4월 SOA 심포지엄에서 세계적인 IT전문 시장 조사기관인 가트너그룹이 제시한 서비스 형태 모형이다.

'서비스 지향'이란 서비스를 서로 통합하여 비즈니스 요구사항을 충족하는 방식을 뜻한다. 다시 말하면 내부 비즈니스 각각의 독립된 기능을 서비스로 정의하고 정의된 서비스를 서로 연결하여 특정 기능을 제공하는 방식이다. 예를 들면 항공권, 호텔, 렌터카 예약, 비행기 탑승, 호텔 체크인 등과 같은 각각의 서비스를 결합하여 태국여행이란 목적을 달성하고 일본여행에서는 항공권, 호텔 예약 등의 서비스는 그대로 재활용하고 후지산 등반 등의 서비스를 대체하면 된다. 결국 SOA는 서비스지향 비즈니스를 지원하는 IT 아키텍처 방식이며 단순한 서비스의 집합이 아니고 서비스 간의 연결 방식 등에 대한 구체적인 내용을 포함하고 있어야 한다. 가트너 그룹은 [그림 2]와 같이 SOA는 서비스와 서비스 사용자로 구성된 어플리케이션 소프트웨어 위상으로 느슨하게 연결된 '1 대 1'의 관계로 이루어진다고 정의했다.



[그림 2] SOA의 구성 형태

SOA란 말 그대로 서비스를 지향하는 아키텍처를 구성하여 비즈니스 업무에 좀 더 효율적으로 활용하겠다는 개념이다. 물론 아직도 SOA를 웹서비스에서 파생된 개념으로 설명되기도 하지만 SOA는 어플리케이션을 서비스 단위로 구축하는 개발 아키텍처와 SOA 실현을 위한 규칙 및 공통 서비스 관리를 포함해서 광범위하게 발전하고 있는 포괄적인 의미의 아키텍처라 할 수 있다.

SOA는 새로운 아키텍처가 아니고 기존의 밀접한 결합형태의 객체지향 모델에 대한 느슨한 결합의 대안이라고 할 수 있다. 개발자 관점의 객체지향 모델에 의한 컴포넌트는 데이터베이스와 강력한 관계로 설계당시 구축된 환경에서는 충분히 잘 활용되지만 업무확장 및 타 시스템 연계가 어려운 구조가 된다. 따라서 한번 만들어지면 변동성이 적어 상대적으로 오래 동안 사용가능한 '프로세스 서비스'와 기업의 비즈니스 환경에 따라 변동 가능성이 상대적으로 높은 '비즈니스 서비스'를 분리하고 이를 연계하는 형태로 구성 되어야 한다. 이러한 비즈니스 단위의 컴포넌트를 메시지 형태로 연계하는 형태가 바로 'SOA'이다. 메시지 형태의 연계는 개발자 관점의 이진코드가 아닌 사용자 관점의 텍스트 메시지 포맷이므로 가독성이 뛰어나고 변환이 비교적 자유롭기 때문에 느슨한 결합이라고 한다.

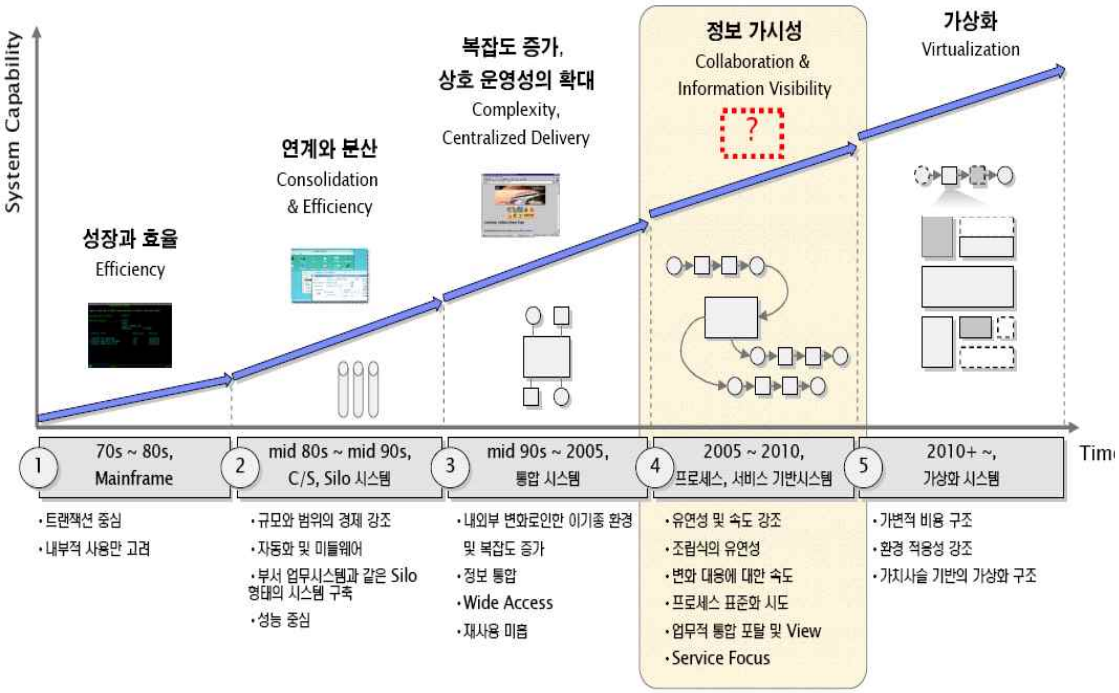
정부나 기업의 정책, 핵심 비즈니스, 파트너십, 산업 표준 등 비즈니스의 본질에 영향을 미치는 관련 요소들은 늘 변화하기 마련이며 이러한 환경에서 유연하게 대처할 수 있는 SOA기반의 느슨한 결합 시스템이 중요한 역할을 할 수 있다고 본다.

나. SOA 필요성

최근 비즈니스 환경변화는 기존 IT 시스템이 그 변화를 따라가기 어려울 정도로 빠르기 때문에, 기존 IT 시스템을 얼마나 유연하게 만들 수 있느냐가 성공의 관건이라 할 수 있다. 기업이 비즈니스 유연성을 확보함으로써 외부변화에 반응하여 빠르게 비즈니스를 바꿀 수 있도록 하는 온 디맨드(on-demand) 비즈니스를 구현하기 위해서는 IT 시스템의 유연성이 필수적이다. SOA는 온 디맨드 운영환경에서 비즈니스 유연성을 가능하게 하는 인프라스트럭처를 제공한다. 하지만 기존 IT 시스템만으로 이러한 비즈니스 유연성을 만족시키기가 쉽지 않다. 따라서 전통적인 IT 시스템과는 다른 패러다임이 필요하고 이러한 필요사항에서 나온 것이 바로 서비스 지향 아키텍처이다.

IT의 패러다임이 과거 중앙집중식 환경 하에서 클라이언트/서버(Client/Server) 환경으로 그리고 다시 분산 컴퓨팅으로 발전하면서 Java RPC나 DCOM, CORBA 같은 초기의 분산 객체에서 발전하여 컴포넌트, 그리고 다시 서비스에 이르기까지 진화해 오고 있다. 이들 분산 객체나 컴포넌트, 그리고 서비스 들은 모두가 위치나 프

로토콜 등 이종의 환경에 상관없이 따로 또 같이 통합시스템으로 운영이 요청되고 있으며, 이는 프로세스와 서비스 기반 환경에서 가능한 것이다. 복잡한 프로세스는 향후 보다 지능적인 가상화 환경으로 발전될 것으로 예상된다.



[그림 3] 컴퓨팅 환경의 변화

2. SOA 관련 기술 표준

SOA는 비즈니스 요구사항을 신속히 해결하기 위해 어플리케이션에 포함된 개별 기능들을 조립 및 재사용할 수 있는 상호운용이 가능한 표준 기반 서비스로 재구성하는 아키텍처로 여기서는 이와 관련된 주요 기술 표준에 대해서 살펴보고자 한다.

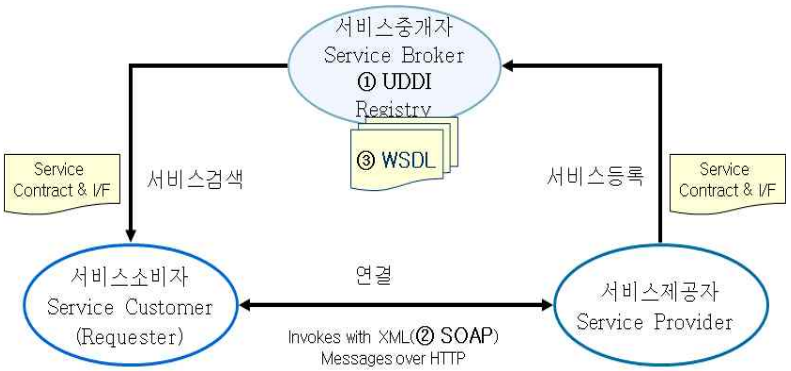
가. 웹서비스

웹서비스는 SOA 개념을 실제로 구현한 기술의 하나로서 서비스 컴포넌트간의 호출에 XML, HTTP, SOAP 등 인터넷 표준 기술 기반의 메시지를 이용하는 인터페이스들의 집합이라 할 수 있다. 2000년 W3C[2]는 웹상에서 벤더 중립적인 접근 프

2) W3C(The Worldwide Web Consortium)는 웹을 위한 표준을 개발하고 장려하는 조직으로 회원기구, 정직원, 공공기관이 협력하여 웹 표준을 개발하는 국제 컨소시엄이다. W3C가 제정한 WWW 관련 표준 목록은 CSS, CGI, DOM, HTML, RDF, SVG, SOAP, SMIL, WSDL, XHTML, XML, XML 정보 집합, XPath, XQuery, XSLT 등이 있다.

로토콜로 SOAP를 정의하여 XML을 기초로 한 메시지 형식이 HTTP를 통하여 통신할 수 있는 전송 프레임워크를 수립하였으며 이듬해 웹서비스 인터페이스를 기술하기 위한 표준 언어로 WSDL을, 서비스를 동적으로 발견하기 위한 표준 메커니즘을 사용할 수 있는 UDDI 명세를 각각 발표하여 웹서비스의 플랫폼이 구현되었다.

웹서비스는 표준 인터넷 기술만을 사용하여 인터넷을 통해 제공되기 때문에 플랫폼에 무관하고 상호간 완벽한 호환성을 보장한다. 이는 COM+가 Microsoft 제품에 종속적이고, J2EE가 Java 언어에 종속적인 것을 극복한 것이라고 볼 수 있다.



[그림 4] 웹서비스의 기본 구성요소

웹서비스의 기본구성요소는 [그림 4]와 같이 서비스 제공자, 서비스 소비자, 서비스 중개자로 구성되어있으며, 기본적인 오퍼레이션은 서비스 등록(Register/Publish), 검색(Find), 연결(Bind/Invoke)이다. 웹서비스에서는 SOA의 각 구성요소들과 오퍼레이션을 구현하기 위해 SOAP, UDDI, WSDL의 세 가지 핵심기술을 활용한다.

XML은 eXtensible Markup Language의 약자로 1998년에 공식 발표된 HTML을 잇는 인터넷 언어로 HTML같은 마크업 언어(Markup Language)이지만 데이터의 종류, 송수신 측의 저장방식, 사용방식 등에서 데이터를 사용할 수 있고 사용자(웹페이지 작성자)가 자신만의 데이터 구조를 직접 만들 수도 있다. 이러한 장점 때문에 XML 웹페이지는 그 자체가 하나의 구조화된 데이터베이스가 되어 이를 적절히 사용할 수 있는 특징을 지닌다. 따라서 XML 웹서비스라 함은 어플리케이션들이 플랫폼, 프로그래밍 언어(Java, .Net, C 등)에 종속되는 것이 아닌 독립적으로 서로 통신할 수 있도록 하는 표준화된 기술을 말한다.

SOAP은 Simple Object Access Protocol로 XML 기반의 메시지 포맷으로서 서비스 소비자와 제공자가 직접 커뮤니케이션 할 수 있는 방법을 제공하며, 전송 프로토콜(HTTP, SMTP, FTP, JMS 등)에 독립적이기 때문에 다양한 전송 프로토콜을 사용하여 전달된다. 예를 들어, 웹서비스를 인터넷상에서 접근 가능하도록 하려면 HTTP를 전송 프로토콜로 채택할 수 있으며, 특정기관 내에서만 이용하도록 하려면 JMS(Java Messaging Standards)와 같은 프로토콜을 채택할 수도 있다.

UDDI는 Universal Description, Discovery, and Integration으로 웹서비스 등을 등록하고 이를 실시간으로 검색할 수 있는 공용 디렉토리 또는 프로토콜의 집합체인데, 웹에서 상호 온라인 거래를 원활히 하고 전자상거래의 상호 운용에 필요한 비즈니스 이름, 제품, 위치 혹은 웹서비스 등의 목록을 작성하여 사용자에게 제공하는 것으로 웹서비스라는 분산된 XML 기반 정보 레지스트리용으로 개발되었다.

WSDL은 Web Service Description Language로 웹서비스 시스템 기능을 명세화시키는 방법을 표준화하기 위해 만들어진 XML 기반의 문서 포맷으로 웹서비스 제공자가 자신의 서비스를 외부에서 이용할 수 있도록 사용법을 알려주는 인터페이스 언어로 CORBA의 IDL(Interface Definition Language)에 해당한다. WDSL 문서에는 서비스 정의, 추상적인 인터페이스 정의, 구체적인 구현정보 등이 포함된다. 웹서비스 제공자는 보유한 서비스를 서비스 중개자인 UDDI 레지스트리(Registry)에 등록함으로써 서비스 소비자들에게 그 서비스를 광고하고 이용할 수 있도록 한다.

웹서비스는 UDDI를 매개로 서비스의 등록과 검색이 이루어지고, 서비스를 요청하는 자와 제공자 간에 등록된 서비스 거래가 SOAP 기반에서 WSDL 형태로 이루어지는 표준화된 인터페이스 모델로 정의할 수 있다. 특히 웹서비스는 개인이 웹 브라우저를 통해 사용하는 단순한 웹페이지 서비스와 달리, 시스템과 시스템 사이에서 서비스를 제공하고 요청할 수 있도록 인터페이스 표준에 맞추어 제공된 독립적이고 모듈화 되어있는 비즈니스 어플리케이션을 일컫는다.

나. BPM

BPM(Business Process Management)은 조직 내·외부를 포괄하는 전체 업무 프로세스와 유기적인 통합이 가능하도록 실시간으로 비즈니스 프로세스 프로세스를 생명주기 전체에 걸쳐서 일관성 있게 관리하기 위한 경영관리기법이다. 업무 프로세스는 근본적으로 조직 내에서 다른 프로세스와 항상 연계되어 이루어지는 것이며, 더 나아가 조직 외부의 타 프로세스들과도 관련성을 갖는다. 그렇기 때문에 업무 프로세스는 거래(Transaction)와 협력(Collaboration) 네트워크 속의 한 부분이 된다. BPM은 경영패러다임 차원과 정보기술 패러다임 차원으로 분리되는데, 이러한 양 차원을 통합적으로 구현할 수 있는 지원하는 솔루션 도구로서 BPMS(Business Process Management System)가 사용된다.

BPMS는 반드시 새로운 기술을 요구하는 것이 아니라 대부분은 기존 기술에 프로세스 관리 개념을 포괄하는 확장된 형태로 발전하고 있다. 프로세스 생명주기 전체, 즉 모형(모델링), 검증 및 최적화, 실행과 운영, 모니터링과 통제, 측정과 분석, 개선 등에 걸쳐 표준이 필요하게 되었으며, 여러 국제표준들이 제정되어 활발한 활

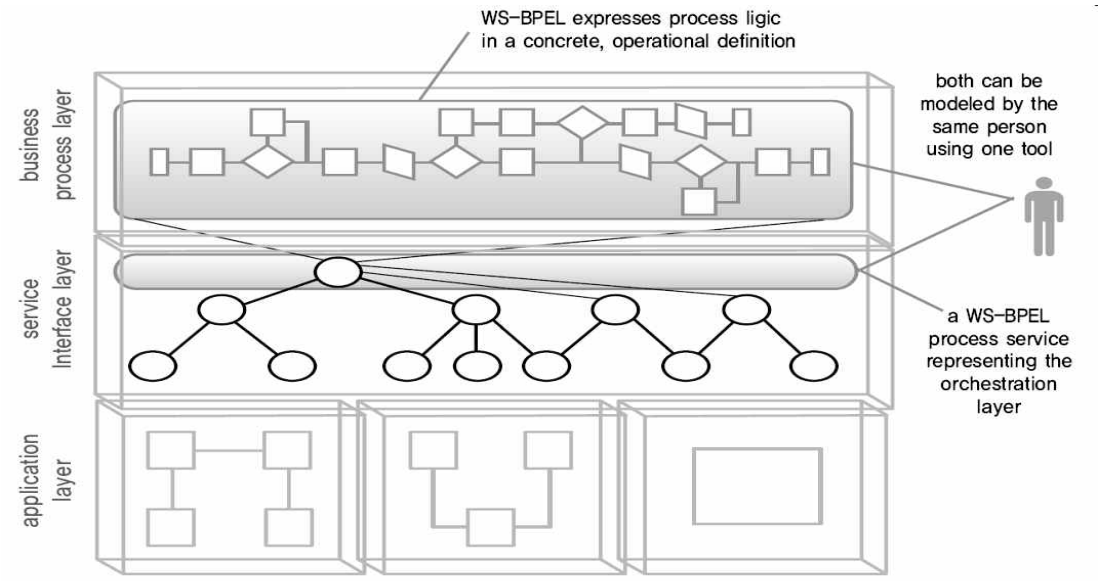
동을 하고 있다. 현재의 BPM 표준들은 다음과 같이 일부 기능들, 즉 모형, 실행과 운영, 모니터링과 통제에 초점이 맞추어져 중복된 표준으로 정의되어 있다.

[표 1] 프로세스 관련 기술 표준

o 프로세스 모형(모델링)	:	BPMN, IDEF0, IDEF3, PSL, UML 2.0
o 프로세스 실행과 운영	:	WS-BPEL(BPEL 또는 BPEL4WS), BPML, BPSS, XPDL, WSCI
o 모니터링과 통제	:	BPQL, BAML

이중에서 BPM과 관련이 있는 표준은 프로세스 실행언어인 WS-BPEL(Web Service Business Process Execution Language)과 프로세스 모델링 언어인 BPMN(Business Process Modeling Notation)이 있다. WS_BPEL은 [그림 5]와 같이 비즈니스 서비스와 프로세스 서비스간의 통합과 제어 역할을 수행하고, BPMN은 플로우차트에 기반을 둔 프로세스 모델링 언어로서 비즈니스 지향적인 프로세스 모델링과 IT 지향의 실행 언어사이의 간격을 채워주는 다리 역할을 수행한다. 즉, 실행 가능한(프로그래밍 가능한) 프로세스 모델링을 위하여 BPEL4WS로의 매핑을 지원하고 있다.

BPM은 기업 내 핵심 업무를 기준으로 단위 업무의 진행 방향, 진행 속도 등을 시스템이 자동으로 결정하고, 이를 정확하게 업무 담당자에게 전달하기 때문에 업무 속도를 혁신적으로 단축시킨다는 평을 받고 있다. 실제로 그동안 BPM 시스템을 구축한 기업 사례를 통해 수백 개의 업무 프로세스가 절반 혹은 수십 개 단위로 단축된 것이 속속 입증되고 있다.

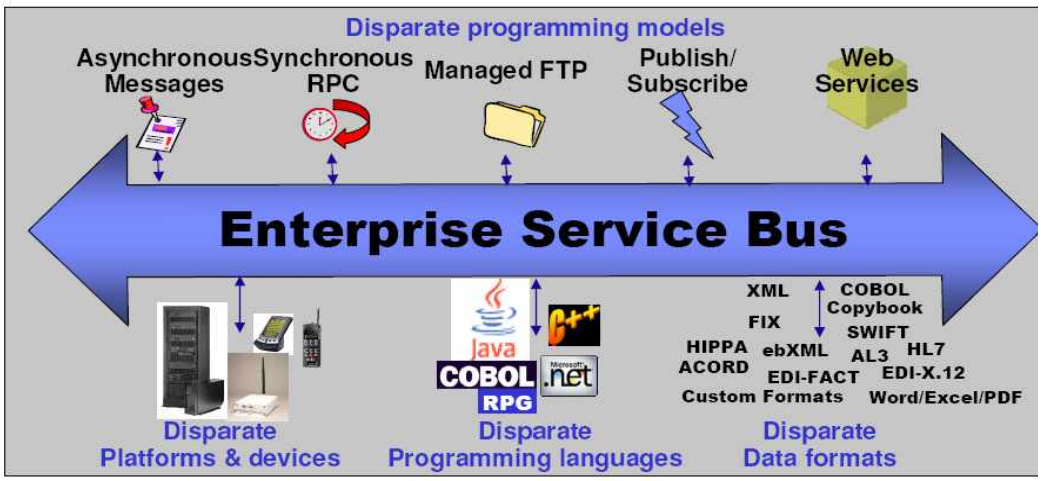


[그림 5] SOA에서 WS-BPEL의 역할

다. ESB

ESB(Enterprise Service Bus)는 SOA 구현을 위한 새로운 개념의 미들웨어로, 다양한 환경에서 개발된 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. 서비스 제공자와 서비스 사용자를 직접 연결하지 않고 ESB를 통하여 연결함으로써, 소비자가 알아차리지 못하는 사이에, 한 서비스 제공자를 다른 서비스 제공자로 대체할 수 있게 한다. 실질적 의미에서 ESB는 원활하게 상호 연동하는 분산 통합서버(Hub)의 집합으로서, 모든 어플리케이션의 통합을 목표로 유연한 연동 서비스를 제공한다.

ESB는 단순히 서비스들을 연결시켜주는 제품이 아니고, 개념적인 구성으로서, 서비스들이 연결되는 SOA 고속도로이다. 이 고속도로는 지역이나, 운송방법이나 수단, 이 기종 플랫폼도 모두 연결 가능하고, 다양한 프로토콜의 사용이나 전환이 가능하도록 하여준다. ESB의 기능을 개념적으로 살펴보면 [그림 6]과 같다.



[그림 6] ESB의 기능

기존 어플리케이션이 상호 연결되는 방식을 바꾸거나 새로운 어플리케이션을 도입해야 하는 경우, 유연하고 관리 가능한 방식으로 어플리케이션을 연결하는 일은 앞으로도 중요한 과제로 남을 것이다. 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 ESB 구현의 핵심이다.

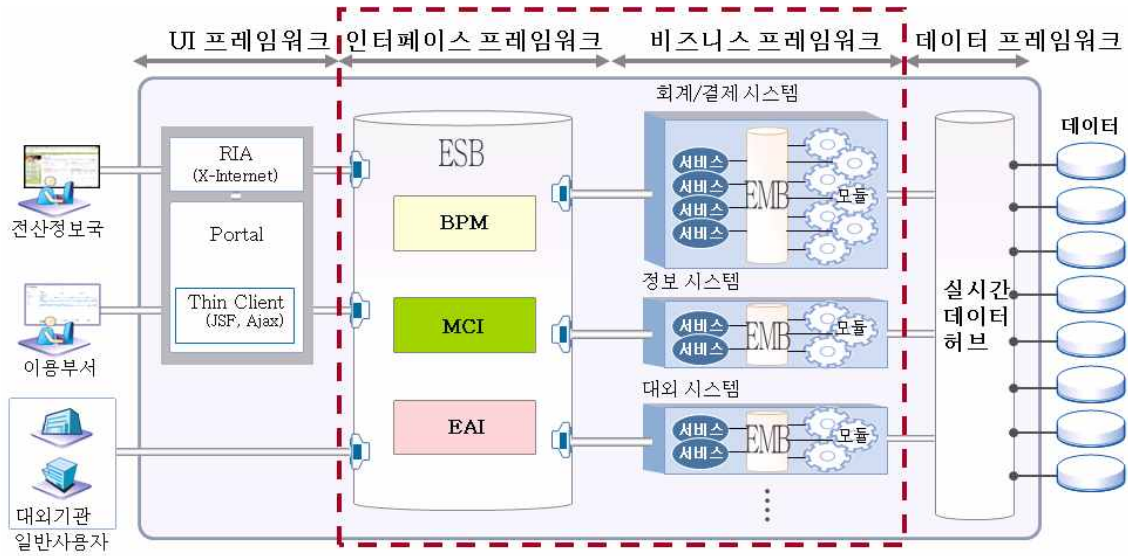
3. 구현 방법

SOA 구현이란 이기종 플랫폼들에 분산되어 있는 표준화된 서비스 중에 적절한 서비스를 골라 필요에 따라 느슨한 결합을 통해 원하는 업무를 새롭게 구성하여 수행할 수 있도록 해주는 어플리케이션 통합 기반을 구축하는 것이다. 이를 위한 전체적인 구성모델을 SOA 프레임워크라고 한다. 즉 SOA 프레임워크는 개념상의 SOA 모델을 바탕으로 한 실질적인 SOA 구현방법이라 할 수 있다 여기서는 각 프레임워크의 기능을 중심으로 설명하고자 한다.

가. SOA 프레임워크 체계

SOA 프레임워크는 프로세스나 특정 언어 및 플랫폼에 구애받지 않고 공유와 재사용이 가능한 서비스를 생성하고 만들어진 서비스를 필요한 곳에서 검색 및 사용이 가능하도록 게시 한다. 각각의 서비스를 조합하여 새로운 서비스를 생성할 수 있어야 하며, 서비스 흐름을 제어하고 재구성하여 새로운 비즈니스 흐름을 생성할 수 있어야한다. 또한 사용자에게 서비스를 활용하여 업무를 수행할 수 있는 비즈니스 도구 즉 사용자 인터페이스 도구를 제공해야한다.

SOA 프레임워크는 [그림 7]과 같이 그 적용범위에 따라 크게 UI(User Interface) 프레임워크, 인터페이스 프레임워크, 비즈니스 프레임워크, 데이터 프레임워크로 분류할 수 있다.



[그림 7] SOA 프레임워크

(1) UI 프레임워크(User Interface Framework)

UI 프레임워크는 시스템 사용자가 이용하는 화면을 제공하고 관리하는 기능을 수행한다. 어플리케이션의 표현 계층(Presentation layer)을 담당하는 영역으로 개발환경과 공통기능을 제공하며 기업 업무의 단일 접점 구축으로 다양한 업무를 통합한다.

기술적으로는 포털(Portal) 서비스와 씬 클라이언트(Thin Client) 혹은 리치 클라이언트(Rich Client)를 제공하는 Portal/Web 사용자 인터페이스 통합 개발환경을 제공하고, 개인화(Personalization), 통합인증(SSO : Single Sign On), 사용자 권한관리 등 공통서비스 기능을 제공한다.

(2) 인터페이스 프레임워크(Interface Framework)

인터페이스 프레임워크는 기업 내·외부 어플리케이션 간의 통합, 비즈니스 프로세스 자동화를 위한 기반을 제공하여 서비스 지향 아키텍처 구현에 있어 매우 중요한 기능을 담당한다. 다양한 기기와의 연결을 지원하는 통합채널(MCI: Multi Channel Integration), 어플리케이션 통합(EAI: Enterprise Application Integration), 비즈니스 프로세스 관리(BPM[3]), 대외중계처리(FEP: Front-End Processer) 등의 기능이다. 특히 SOA 프레임워크의 백본 역할을 하는 서비스 버스(ESB[4])를 기반으로 하면 통합된 인터페이스 환경을 자동으로 제공할 수 있다.

ESB는 비즈니스 프로세스를 구성하는 서비스간의 처리 흐름을 규칙화하여 서비스간 연계를 지원한다. 따라서 서비스는 순수 비즈니스 기능 로직만 처리하도록 설계되고, 서비스 간 의존성을 최소화하여(loosely coupling), 서비스의 조립 및 재구성을 가능하게 한다. 또한 서비스 버스는 다양한 프로토콜에 기반 한 동기/비동기 메시지를 처리하여 규약(Configuration rule)에 따라 목적지로 라우팅(Routing)하는 메시지 중개자(Broker) 기능과 메시지 변환(Transformation) 기능을 함께 제공한다.

(3) 비즈니스 프레임워크(Business Framework)

비즈니스 프레임워크는 비즈니스 기능을 구현하는 서비스의 개발환경과 운영환경을 제공한다. 느슨한 결합구조를 가지고 재사용이 가능한 비즈니스 프로세스를 구현할 수 있는 서비스를 설계하고 조립해야 한다. 이를 위해 EMB(Enterprise Module Bus) 아키텍처는 서비스간 호출을 규칙(Rule) 기반으로 처리하고, 비즈니스 모듈은 순수 비즈니스 로직만으로 구성하여, 모듈의 조합만으로 서비스를 개발 할 수 있도록 하는 새로운 방식의 SOA 어플리케이션 개발을 지원한다.

---

3) 본 논문 'Ⅱ. SOA에 대한 이론적 고찰, 2. SOA 관련 기술 표준, 나. BPM' 참고

4) 본 논문 'Ⅱ. SOA에 대한 이론적 고찰, 2. SOA 관련 기술 표준, 다. ESM' 참고

비즈니스 프레임워크에서 모든 비즈니스 로직, 계산식, DB접근은 반드시 서비스 컴포넌트를 통해서만 접근이 가능하다. 서비스 컴포넌트는 해당 서비스 컴포넌트에 포함되어 있는 서비스 컴포넌트, 비즈니스 모듈, 데이터 모듈의 흐름을 재구성하여 단위 비즈니스 로직 자체와 비즈니스 로직의 내부 흐름을 분리하여 관리하는 기능을 제공한다. 이러한 서비스 컴포넌트의 정의, 구성, 흐름을 구현하는 기능은 EMB 환경에서 담당하게 된다.

이와 같이 EMB는 서비스 내 비즈니스 처리 흐름을 가시화 하여 비즈니스 기능 모듈의 조립 및 재구성과 재사용을 용이하게 하며, 서비스 내의 모든 수행 흐름을 모듈 버스에서 제어하기 때문에 실시간 모니터링이 가능하다. 또한 EMB 기반 시스템에서는 단위서비스의 재사용 및 조립만으로도 단위서비스를 결합하는 복합(조립) 서비스(Composite Service)의 개발이 가능하다.

(4) 데이터 프레임워크(Data Framework)

데이터 프레임워크는 서비스 계층(Service Layer)에 추상화된 데이터 스키마(Data schema)를 제공하여 데이터 통합 효과를 제공하며, 전사 정보자원의 구성 방식에 유연성을 제공한다. 데이터 허브(Data Hub)로써 규칙기반의 데이터 동기화와 표준화된 데이터 접근 포맷 매핑(mapping), 대용량 데이터 이관기능을 제공하여 서비스 계층에서의 데이터 처리를 단순화하고 빠른 처리성능을 보장한다.

나. 구현 단계

SOA기반 정보시스템 구축을 위한 진입단계는 크게 4단계로 초기 도입단계, 어플리케이션 통합 단계, 리엔지니어링 단계 및 성숙 단계로 나눌 수 있다. SOA 준비상태, 조직의 전략 등을 고려하여 어플리케이션 통합단계 또는 리엔지니어링 단계에서 시작할 수도 있다.

(1) 초기 도입 단계(Application Interface)

특정 어플리케이션이나 신규 어플리케이션을 상호 연결하는 초기 단계에서는 시스템간 메시지 인터페이스를 표준화하고 향후 확장을 고려한 시스템 디자인이 필요하다. 이 기종 시스템 사이의 인터페이스를 확장하거나 외부기관 비즈니스 파트너와의 연결 등에서 SOA를 활용할 수 있다.

SOA 패키지를 적용하지 않고 업무 서비스간 종속성을 최소화하면서 중복되는 서비스를 제거하고 재사용을 극대화하는 SOA 사상에 충실하게 적용하는 단계이다.

(2) 어플리케이션 통합 단계(Service Integration)

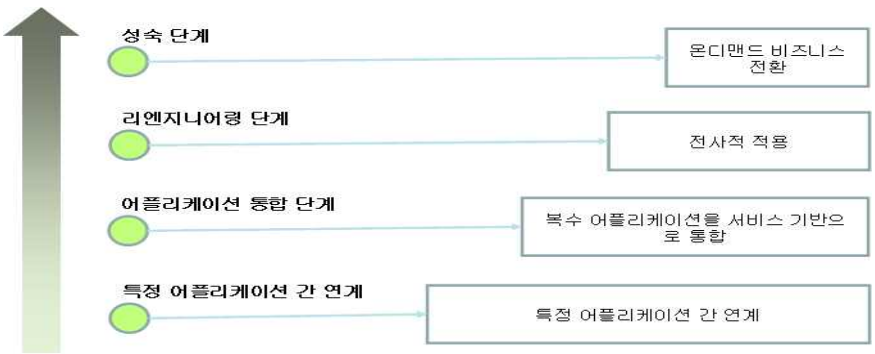
복수의 어플리케이션을 서비스 기반으로 통합하는 단계로 초기 도입 단계의 기술을 활용해 ESB를 구현하는 단계다. 기존 시스템과의 연계, BPEL을 사용한 비즈니스 프로세스 제어 등을 구현한다. ESB 구축을 통한 EAI 실현이나 비즈니스 프로세스 관리 및 어플리케이션 통합 등을 위해 적용될 수 있는 단계이다.

(3) 리엔지니어링 단계(Infra Integration)

SOA를 전사적으로 적용하는 리엔지니어링 단계에서는 시스템간 연계 뿐 아니라 IT를 이용한 전사적 비즈니스 프로세스의 횡적인 연계 검토가 수행되는 단계이다. 비즈니스 컴포넌트 모델링, 기업 내 데이터의 정규화, 일원화를 위한 분석, EA 접근방식에 의한 시스템 기본 계획의 책정, SOA 구현 방법론에 의한 서비스 컴포넌트의 도출과 컴포넌트 단위의 결정 등 작업을 진행할 수 있다. 이 단계에서 SOA를 시작한다면, 업무개발부터 서비스 개념을 도입하여 IT 인프라스트럭처에 연결함으로써 전사적인 SOA를 구현하게 된다.

(4) 성숙 단계(Business Intelligence)

성숙단계에서는 새로운 비즈니스 모델을 창출하고 EA 접근방식에 의한 시스템 확정 등의 작업을 하게 된다. 예측 불가능한 급격한 변화에 대응하기 위해서는, 고객의 요구에 대응해, 기업 내부 자원 뿐 아니라 외부에서 필요한 만큼의 자원을 조달해, 최종적인 가치를 제공하는 비즈니스 모델을 실천할 필요가 있고, 이를 통해 기업은 대응 능력이 뛰어난 유연한 비즈니스를 전개할 수 있다.



[그림 8] SOA 구현 방법

4. 구현에 따른 효과

가. 구현 장점

(1) 시스템 활용성 증대

SOA 내에서 구현된 서비스들은 공통의 디렉터리 내에서 존재하게 되는데, 이에 대한 목적, 위치 및 각각의 서비스 사용에 대한 정보가 동시에 제공되기 때문에 재사용이 훨씬 용이해진다.

(2) 표준 준수

각각의 서비스들이 상호연동을 위한 표준들의 집합체로 구성되어 있기 때문에 운영 시스템, 플랫폼 및 벤더들의 특정 기술들과 무관하게 '플러그 앤 플레이' 방식의 상호 호환성을 제공한다.

(3) 모듈화와 독립성

모듈은 특정 목적에 사용되는 단위 부품이며, 서비스들이 단위 블록으로 모듈화되고 이를 다양한 방식으로 활용하여 전체 시스템을 모듈들의 조합으로 구성될 수 있으며, 많은 비즈니스 프로세스를 자연스럽게 구성 혹은 개발할 수 있다. 또한 각각의 서비스들은 어플리케이션에 대해서 독립적으로 변환되고 관리될 수 있다.

(4) 추출 용이성

각각의 서비스들이 모든 메타데이터들을 표현할 수 있는 표준 인터페이스를 사용하기 때문에 마치 '블랙박스'와 같은 역할을 자체적으로 할 수 있으며, 이는 개발자들의 개발 노력과 시간을 절감시켜준다.

나. 기대효과

(1) 변화에 적응하기 위한 비즈니스 유연성 강화

SOA를 통해 하나의 서비스가 다른 서비스들과 보다 쉽게 결합되고 조화롭게 통합됨으로써 새로운 비즈니스 프로세스를 신속하게 생성해 낼 수 있다. 예컨대 공유된 서로 다른 서비스 간의 조화로운 통합은 개별 서비스 객체 내에 대부분의 기능

이 이미 내재되어 있기 때문에 기존의 개발 단계에서 요구되어 온 동일한 수준의 핸드 프로그래밍을 요구하지 않는다. 가용 서비스 집합의 서비스 조합을 통해 비즈니스 변경 요구에 민첩하게 대응할 수 있다.

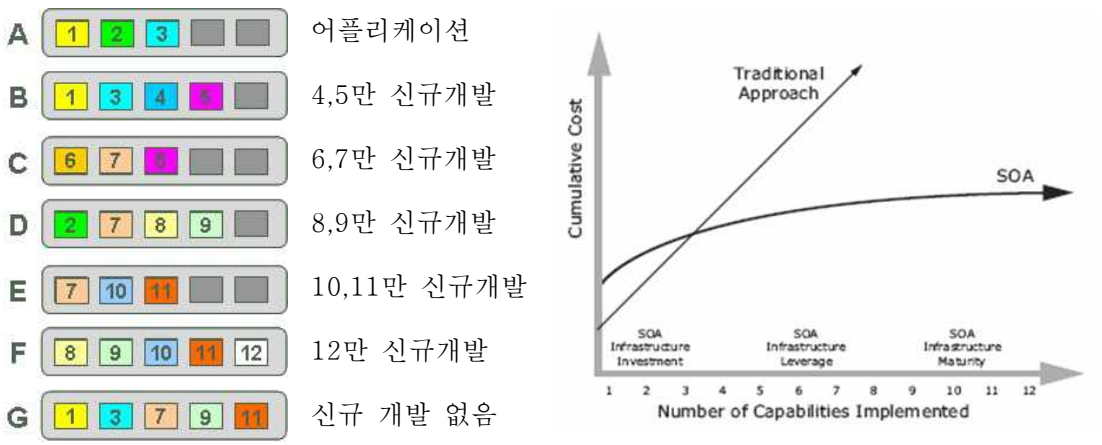
(2) 데이터 및 프로세서의 일관성 증대

반복 사용되는 기능을 서비스로 도출하여 구현함으로써 여러 장소, 여러 시스템에 분산 되어 있던 기능들을 가능한 일관되게 유지할 수 있다. 정보시스템의 경우 제공기능 및 정보를 타 정보시스템과 공유하는 경우가 있는데 과거의 컴포넌트는 동일시스템 내에서 동일한 도구를 사용해야만 재활용이 가능했으나 SOA기반 시스템은 다른 어떤 시스템과도 서비스 차원의 조합이 가능하다.

(3) 시간 절감 효과

SOA 구축은 완전히 새로운 프로세스라 할지라도 변화하는 비즈니스 여건에 맞추어 빠르고 정확하게 정의될 수 있으며, 서비스들은 전사적 차원의 다양한 비즈니스 기능을 갖춘 에코시스템을 구성하게 되어 기업 내 끊임없는 단위 프로젝트의 개발 비용과 ROI 창출에 이르는 시간 절감 효과를 거둘 수 있다.

SOA 시스템의 경우 초기에 SOA에 필요한 인프라의 구축(ESB, BPM, 모니터링, 보안 인증, UDDI 등)에 비용이 많이 소요되기 때문에 초기 투자비용이 기존의 IT시스템 보다 높아진다. 그러나 계속해서 업무를 추가하면서 새로운 업무가 완전히 새롭게 구성되는 것이 아니고 앞 단계에서 개발된 서비스들을 재사용하기 때문에 개발 비용은 지속적으로 감소된다.



[그림 9] SOA 시스템 개발과 비용관계

Ⅲ. 당행 정보시스템 현황

1. 정보시스템 개발 현황

1990년대에 들어서면서 컴퓨팅 환경이 급속도로 발전하여 메인프레임과 유닉스 시스템이 공존하고, 오픈 환경의 네트워크를 기반으로 인터넷이 보편화되고, PC 기반의 운영체제는 그래픽 처리 중심의 윈도우 환경으로 바뀌게 된다. 컴퓨터 언어는 COBOL, PL/I 등 텍스트 기반 언어에서 Delphi, Power Builder, Visual C, Java 등 비주얼 환경의 언어가 주류를 형성하게 된다. 이러한 정보통신기술 환경의 변화에 따라 당행의 정보시스템이 새로운 환경으로 재편되는 개발업무의 특징을 살펴보고자한다.

전산시스템 상에서 2000년 연도 표시 문제로 세계적인 이슈가 제기되면서 운영중인 기존 시스템을 확대 개편하거나 전면 재구축하는 붐이 조성되었다. 당행에서도 총괄계리, 여수신, 국고 증권, 국제금융, 발권 등 회계 관련 모든 업무시스템을 재설계 하고 주전산기를 기반으로 응용프로그램을 전면 재개발하면서 사용자에게는 업무수행이 편리하고 쉽게 접근할 수 있는 직관적인 GUI(Graphic User Interface)화면을 제공하기 위하여 클라이언트/서버 방식으로 구축하였다. 2000년 이후에는 당행의 대내외 역할 제고에 따른 새로운 개발 업무가 대폭 증가하면서 국고금실시간이체시스템, CLS연계시스템, 경제교육홈페이지 등이 새로이 구축되고, 경제통계시스템, 경영관리시스템, 문서종합관리시스템 등 기존 시스템도 통합 및 효율화를 위하여 UNIX기반의 Java를 사용한 웹 방식으로 재구축되었다.

정보시스템의 데이터는 기존에 VSAM[5], ISAM[6] 등 파일관리시스템으로 처리하던 방식을 모두 데이터베이스시스템으로 전환하여 사용하고 있으며 현재는 데이터베이스 중에서도 모든 데이터를 테이블 형식으로 구성하여 조작 및 관리가 편리한 관계형 데이터베이스를 사용하고 있다. UNIX 시스템에서는 DB2나 오라클 제품을 윈도우서버 이용 업무에서는 MS SQL을 사용하고 있다.

정보시스템 개발의 품질 향상과 개발의 투명성, 생산성 향상을 확보할 수 있도록 하기 위하여 소프트웨어 개발 방법론도 많은 발전을 이루었다. 소프트웨어 개발에

5) VSAM(Virtual Storage Access Method)은 IBM의 운영체제인 MVS(현재는 z/OS)하에서 사용되는 파일관리시스템이다. VSAM을 사용하면 기업은 파일내의 레코드를 들어갔던 순서대로 만들고 액세스할 수 있고 검색키를 이용하여 각 레코드를 검색하거나 정장할 수 있다.

6) ISAM(Indexed Sequential Access Method)은 레코드에 순차적으로 접근하거나 또는 색인을 통해 선택적으로 접근하는 방식 둘 모두를 제공하는 파일관리 시스템이다. 각 색인은 저장된 레코드들에 대해 순서를 다르게 정의한다. VSAM과 관계형 데이터베이스 이전에 주로 쓰였다.

관한 접근 방식에 따라 구조적 개발방법론, 정보공학 방법론, 컴포넌트 개발방법론 등으로 분류되고 있으며 회계시스템의 개발에서는 정보공학방법론이 기타 웹 기반의 정보계업무 개발에는 컴포넌트 개발방법론이 주로 사용되고 있다. 최근에는 콘텐츠 기반으로 구축된 경제교육홈페이지에서는 콘텐츠 기반 개발방법론이 새롭게 사용되기도 하였다.

정보시스템 개발업무의 또 하나의 특징은 개발 규모가 크고, 최신 IT기술을 적용하여 개발함에 따라 당행 자체인력만으로는 그 개발의 한계를 초과하는 경우가 대부분이어서 주로 외부 전문 업체를 통한 개발이 주류를 이루고 있다는 현실이다. 회계시스템의 경우에는 당행 직원과 시스템통합업체가 공동으로 참여하여 협업 방식으로 개발을 하였으나 나머지 대부분의 개발업무는 시스템통합업체에 의한 응용프로그램 개발과 인프라 도입까지 일괄 추진되고 있으며 당행 직원은 프로젝트 기획과 관리 및 현업 담당자와의 커뮤니케이션 등을 주로 담당하고 있다.

[표 2] 주요 정보시스템 개발현황

정보시스템	구축	SW, DB	HW	구현 기술	개발방식
회계온라인시스템 (회계 관련 전 업무)	1999	COBOL, Delphi, DB2	주전산기 (IBM)	C/S	공동개발
외환정보시스템	2000	Easy Base, Oracle	UNIX	Web	외주개발
한국은행정보시스템	2000	Java, Oracle	UNIX	Web	외주개발
국고금실시간시스템	2002	C, Delphi	UNIX	C/S	외주개발
감사정보시스템	2003	ASP, MS SQL	Windows	Web	외주개발
경제통계시스템	2004	Java, SAS, , Oracle	UNIX	Web	외주개발
보기화폐관리시스템	2004	Java, Oracle	UNIX	Web	외주개발
금융기관경영정보	2004	Java, Oracle	UNIX	Web,C/S	외주개발
CLS 연계시스템	2005	Java, Oracle	UNIX	Web	외주개발
한은금융망시스템	2005	Java, DB2	UNIX	Web	외주개발
경제교육홈페이지	2006	Java, Oracle	UNIX	Web	외주개발
경영관리시스템 (인사 등 8개 부문)	2006	Java, Oracle	UNIX	Web	외주개발
문서종합관리시스템	2006	Java, Oracle	UNIX	Web	외주개발
전자도서관시스템	2006	Java, Oracle	UNIX	Web	외주개발
채권정보시스템	2006	Java, Oracle	UNIX	Web,C/S	외주개발
사이버 화폐박물관	2007	Java, Oracle	UNIX	Web	외주개발

* SW 및 HW는 대표적인 요소 기술만 기술

2. 정보시스템 운영 현황과 과제

가. 정보시스템 운영 현황

앞에서 살펴보았듯이 1999년에 가동한 회계온라인시스템을 제외하고는 당행의 주요 정보시스템은 2000년 이후 새로운 IT기반 기술로 개발되었다. 시스템 구축에 참여한 당행 직원의 참여비율이 절대적으로 적은 상황에서 프로젝트 기획, 관리 등 행정적인 업무를 담당하고 있는 당행 직원이 용역개발업체가 구축한 프로그램 및 구축 내용을 파악하고 단시간 내에 운영관리를 전담하는 것은 현실적으로 불가능한 상황이다. 이러한 어려운 점을 고려하여 용역개발업체가 시스템 안정화 및 운영 역할까지 포함한 아웃소싱 형태의 계약을 체결하고 지속적인 개선을 수행하고 있다.

회계시스템의 경우는 개발규모가 방대하고 개발기간이 길었지만 사업 초기부터 당행 직원이 부문별로 직접 참여하여 개발완료 후에는 용역개발업체로부터 인수를 받고 독자 운영할 수 있는 기반을 마련하였다. 이후 개발이 진행된 대부분의 정보시스템은 운영에 따른 변경 등이 많이 발생하여 직접지원의 필요성이 높은 경우에는 상주용역인력을 활용하고, 패키지 소프트웨어를 당행 환경에 맞게 수정한 문서종합관리시스템이나 변경이 상대적으로 적고 긴급성을 요하는 정도가 낮은 경우는 비상주인력에 의한 운영지원을 하는 유지정비계약에 의하고 있다.

[표 3] 주요 정보시스템 운영계약 현황(2007.9말 현재)

정보시스템	계약업체	상주용역	유지정비(개발부문)	비고
경영관리시스템	동양시스템	4명	-	인사 등 8부문
경제통계시스템	동양시스템	2명	-	
경제교육홈페이지	SK C&C	1명	-	
채권시장정보시스템	코스콤	1명	-	
한은금융망시스템	SK C&C	1명	-	
문서종합관리시스템	핸디소프트	-	○	패키지SW
한국은행정보시스템	탑앤와이즈	-	○	
감사정보시스템	위키소프트	-	○	
보기화폐관리시스템	탑앤와이즈	-	○	
금융기관경영분석시스템	코오롱베니트	-	○	

* 전산정보국에서 운영 중인 정보시스템을 중심으로 작성

나. 정보시스템 과제

현재 운영 중인 정보시스템을 중심에서 개발과 운영에 관한 현황을 살펴보았다. 여기서는 IT 서비스지향 관점에서 개선이 필요한 과제를 다음과 같이 제시해보고자 한다.

(1) 정보시스템 운영업무 통합 관리

경영관리시스템, 경제통계시스템, 경제교육홈페이지 등 각각의 정보시스템은 해당 시스템의 업무영역 내에 있는 세부업무까지 전산화되면서 시스템의 개발 규모가 계속 증대되고 있으며 독립적으로 개발이 되고 있다. 통상 입찰에 의해 개발을 수행하는 업체가 선정이 되고 동 업체는 개발의 편의성을 위하여 자체적으로 사용하고 있는 개발 프레임워크에 당행에서 제시하는 전산기기(인프라 환경, 개발 언어 등)의 구성 조건을 적용하여 개발을 진행한다.

이렇게 진행된 개발 프로젝트는 운영환경에서 그대로 적용돼 개발에 참여한 직원만이 원활하게 유지보수 및 운영업무도 맡을 수 있는 제약조건이 되고 있다. 개발언어나 데이터베이스를 당행 환경에 맞게 운영을 하더라도 각각 개발 업체별로 상이한 개발 프레임워크가 존재하게 되어 운영업무 자체가 개발업체에 종속되는 현상을 낳고 있으며 심지어 동일한 개발업체의 경우에도 개발 시점에 따라 시스템별로 버전이 상이한 경우도 있어 유지정비를 독립적으로 수행할 수밖에 없다.

정보시스템의 운영업무를 외주용역업체에 의존하는 경우에도 정보시스템 별로 분리되지 않고 통합운영이 가능하도록 표준 개발 프레임워크를 통일하거나 국제적인 표준에 부합하는 제품이 들어올 수 있도록 개선할 필요가 있다.

(2) 개발 컴포넌트 재활용

현재 운영 중인 시스템은 대부분 컴포넌트개발방법론(CBD)을 활용하여 개발했지만 컴포넌트의 재사용은 동일한 개발 플랫폼에서만 사용되고, 다른 정보시스템에서는 전혀 재활용되지 못하고 있다. 컴포넌트개발방법론은 업무기능이나 서비스 단위로 모듈, 즉 컴포넌트를 만들고 동 컴포넌트를 재사용함으로써 개발의 생산성과 중복투자에 따른 낭비를 제거하는데 필요한 것이다. 이러한 효과를 위해서는 모든 정보시스템 구축을 한번에 통합개발하거나 컴포넌트개발방법론으로 구축한 개발환경을 다음번 개발에 그대로 사용하는 경우에는 가능하지만 정보시스템 개발이 별도의 환경에서 독립적으로 이루어지는 현실에서는 실현되기 곤란한 방법이다.

동일한 업체의 개발 프레임워크를 사용한다고 해도 정보시스템마다 별도로 구성된 개발환경에서는 컴포넌트를 공유하기 위해서 각각 별도의 환경에 존재해야 한다. 이렇게 사용된 컴포넌트는 변경이 필요해서 수정할 경우에도 각각 분산된 컴포넌트를 모두 수정하고 각 시스템별 상이한 영향평가를 해야만 동일한 결과가 보장될 수 있다.

지금까지의 컴포넌트 개발은 단위업무별로 개발업체의 관점에서 이루어졌으나 앞으로는 정보시스템 전반에 걸쳐 재활용이 가능하고 이기종 플랫폼에도 실행될 수 있도록 표준에 맞추어 개발하는 것이 필요하다.

(3) 정보시스템 인프라 통합

당행 뿐 아니라 대부분의 기관에서 정보시스템 개발업무는 최신의 정보통신기술을 기반으로 시스템통합업체를 활용하여 구축하고 있다. 또한 응용프로그램 개발과 더불어 정보인프라에 해당하는 전산기기의 도입까지 통합 발주하는 형태를 취하고 있다. 기존에는 응용프로그램 개발과 기기도입을 분리하여 추진한 경우도 있었으나 분리발주를 할 경우 전산기기 도입 단계에서는 응용프로그램 개발업체에서 제안하는 기기의 사양을 수용해야하는 등의 문제점이 있어 통합 발주를 주로 활용한다.
이러한 상황에서 당행의 정보시스템 확충과 함께 정보계용 서버의 양적인 증가도 함께 수반되어 서버운영업무 부담도 가중되고 있는 현실이다.

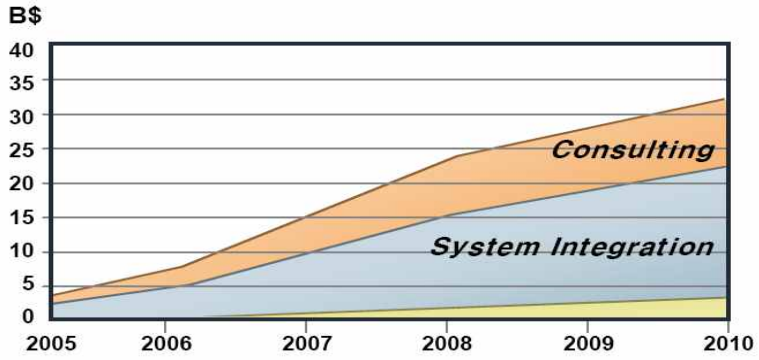
개발당시에는 서버용량의 정확한 측정이 불가능할 수 있으나 어느 정도 개발이 완료된 상태에서 노후 기기에 대한 교체 수요도 발생하고 현재 운영 중인 시스템의 용량이나 사용도 분석을 통한 정확한 규모 측정이 가능하므로 전산 인프라에 대한 통합을 고려할 필요가 있다.

Ⅳ. SOA 도입 사례 조사

1. 국내외 시장 동향

가. 국외 SOA 시장 동향

IDC의 조사에 의하면 SOA관련 시장은 지난해 전 세계적으로 36억 달러 규모를 형성한 것으로 나타났다. 2010년에는 338억 달러 시장으로 성장할 것이라고 전망했는데, 이는 연평균 167%에 달하는 높은 성장률이다. 특히 컨설팅 부문과 시스템 통합 서비스가 주로 SOA 초기 시장을 형성하여 전체 SOA 기반 서비스 시장의 95% 이상 차지할 것으로 전망했다.



[그림 10] SOA 기반 IT서비스 시장 규모(IDC 2006. 4)

가트너는 SOA가 과도한 관심에서 벗어나 주류 기술 트렌드로 자리를 잡기 위한 과도기에 있으며, 2년 ~ 5년 내 안정기로 접어들 것으로 전망했다. 또한 '앞으로 킬러 어플리케이션은 없을 것이며, 킬러 환경만 존재할 것이다'라며, 급변하는 변화에 빠르게 대처할 수 있는 환경을 갖춰야 한다고 강조했다. SOA는 제품이 아니라 설계 개념으로, 기존에 구축된 다양한 IT영역에 영향을 미치고 있다. 컨설팅 및 SI를 중심으로 하는 IT 서비스 영역과 어플리케이션 서버, 메시징, EAI 및 채널통합을 포함한 어플리케이션 미들웨어, 사용자 인터페이스 및 패키지 솔루션 등 소프트웨어 영역까지 빠르게 침투했다.

나. 국내 SOA 시장 동향

한편 IDC는 국내 시장이 현재 초기 시장에서 지나 본격적인 진행 단계를 밟고 있다고 정의하고 있다. 지난해 하반기를 기점으로 국내 시장에서 SOA의 개념 및

구현 기술에 대한 논의 수준을 벗어나 파일럿 프로젝트 등을 통한 실제 구현의 모습이 나타났다. 통신, 금융, 제조, 공공 등 다양한 업종에서 SOA 파일럿 프로젝트가 진행되고 있으며, 차세대 시스템과 결합돼 전사 규모의 도입도 검토되고 있다.

국내 사용자들의 SOA에 대한 인식은 벤더들의 홍보와 파일럿 프로젝트 등을 통해 널리 확산됐다. 전산시스템 담당자라면 누구나 한번쯤 SOA에 대해 들어봤을 것이다. 특히 금융기관에서 차세대시스템에 SOA를 도입하려고 하는 것은 기업 자체가 실시간 기업이 되지 않으면 생존할 수 없다는 생각에 따른 것이다. 어떻게 하면 위험을 줄이고 최적의 효과를 낼 수 있을까 하는 고민의 대안으로 SOA가 주목 받고 있다. SOA는 한 번에 전사적으로 도입하는 것은 현실적으로 불가능하다고 인식되고 있다. 대부분 프로젝트가 일부 점진적인 수준으로 확대되고 있다.

2. 구축 사례 분석

가. KTF

KTF는 모바일 콘텐츠 제공 사업자 CP(Content Provider)가 KTF 내부 인프라의 기능을 사용해서 서비스를 개발할 때, 반복되는 복잡한 시스템간 연동 문제를 해소하고 KTF 인프라의 모든 기능을 용이하게 사용할 수 있는 플랫폼을 구축했다.
기존의 시스템은 전용 API를 제공하는 강결합(tightly-coupled) 방식이었기 때문에 서비스 개발 및 제공 시에 각 시스템에 대한 개별적인 의존성이 비교적 높은 편이었다. 따라서 품질이 개선된 서비스 제공과 비즈니스 경쟁력 향상을 위해서 KTF는 기존의 CP시스템과 내부 인프라간의 연동방식을 느슨한 결합(loosely-coupled) 기반으로 하는 SOA 기반 형태로 바꾸기로 결정했다.

이를 위해 KTF는 무선 인터넷 인프라 기능을 통합하여 제공하는 개방형 무선인터넷 플랫폼인 통합 비즈니스 플랫폼(KTF Hub for oUr Business, KHUB)을 구축했다. 궁극적으로 이번 KHUB 프로젝트의 가장 핵심은 바로 '통합'과 'SOA 기반의 소결합 인터페이스' 구축이다. KTF는 SOA를 기반으로 모든 통합 요구에 대해 단일 인터페이스로 접근하기 위해 SOA 라이프사이클을 지원하는 Oracle 솔루션 도입을 통해 KHUB 시스템을 가동했다. 기존의 단순 기능연동의 API 방식과는 달리 KTF 내의 모든 인프라의 기능을 단 한번의 호출로 쉽게 제공받을 수 있는 통합 플랫폼을 구축함으로써 서비스 개발 시 시간과 비용 절감이 이뤄지게 됐다.

개발의 편리성 이외에 부가적인 이점으로는 SOA기반의 신규 API 시스템을 통해 개별 인프라 시스템이 제공하던 단위기능을 서비스 관점에서 모듈화 하여 기존에는

매우 어렵게 여겨지던 융·복합 서비스 개발도 용이해졌다. 이를 통해 복잡한 비즈니스 모델에 대한 지원방법도 다양해져 향후 서비스의 경쟁력 향상으로 이어질 것으로 내다보고 있다. 궁극적으로는 개발 환경이 다양한 서비스 및 비즈니스 개발이 가능하도록 서비스 개발자 중심으로 개선된 것이며 고객 만족 중심의 고도화된 서비스 제공이 가능해지는 것을 의미하기 때문이다.

나. CJ

CJ는 ERP, SCM, CRM 등 여러 시스템에 각각 접속해야만 원하는 정보를 얻고 업무를 처리할 수 있는 불편함을 해소하기 위해 EKP(Enterprise Knowledge Portal) 프로젝트에 SOA를 도입했다. SAP 넷위버의 엔터프라이즈 포털을 적용해 직무 역할별 업무 프로세스를 정의하고 각 직무별로 과제 목록(to do list)을 포함한 워크벤치(Workbench)와 개인별 정보 조회 및 업무 처리를 위한 개인 포탈을 구현했다. 새로운 보직을 배정 받은 직원은 업무 첫날 바로 자신의 엔터프라이즈 포털 사이트에서 새로 습득해야 할 업무 관련 지식과 정보, 프로세스 등을 즉시 파악할 수 있다. 이를 통해 신규 업무에 대한 적응 시간을 대폭 단축하는 효과를 거뒀다.

CJ의 프로젝트가 성공을 거둘 수 있었던 데에는 포털에 대한 요구사항과 도입한 SOA 솔루션의 기본 개념이 유사했다는 점을 들 수 있다. 또한 개념 검증을 위해 파일럿 프로젝트를 추진한 것도 중요한 성공 요인이다. 무엇보다 사용자의 요구사항이 무엇인지를 먼저 면밀히 검토했다는 점이 관건이었다. 결국 엔터프라이즈 포털을 통해 기업 내에 존재하는 다양한 정보를 통합된 관점에서 조회하고 이를 실제 업무 활동에 적극 활용하기 위한 토대를 마련했다는 사실이 이번 프로젝트를 성공으로 이끄는데 주효했다.

한편 IT운영의 초점이 사용자 중심으로 변화했고 이에 따라 EKP 내의 프로세스 정의뿐 아니라 IT체계 전반에 걸친 프로세스 정립을 완성했다. 특히 IT부서 내에도 현업의 기능 요구사항을 분석하고 직무 역할과 프로세스에 맞게 재배치하는 일을 담당하는 업무 프로세스 분석가(Business Process Analyst)를 구성했다. 이들 인력은 업무 프로세스와 직무 역할 정의는 물론 기업용 패키지 소프트웨어 평가 분석까지 담당한다. IT부서는 엔터프라이즈 포털을 도입하는 과정에서 현업 사용자의 업무 프로세스를 중심으로 사고하고 지원하는 체계로 일대 전환을 꾀할 수 있었다.

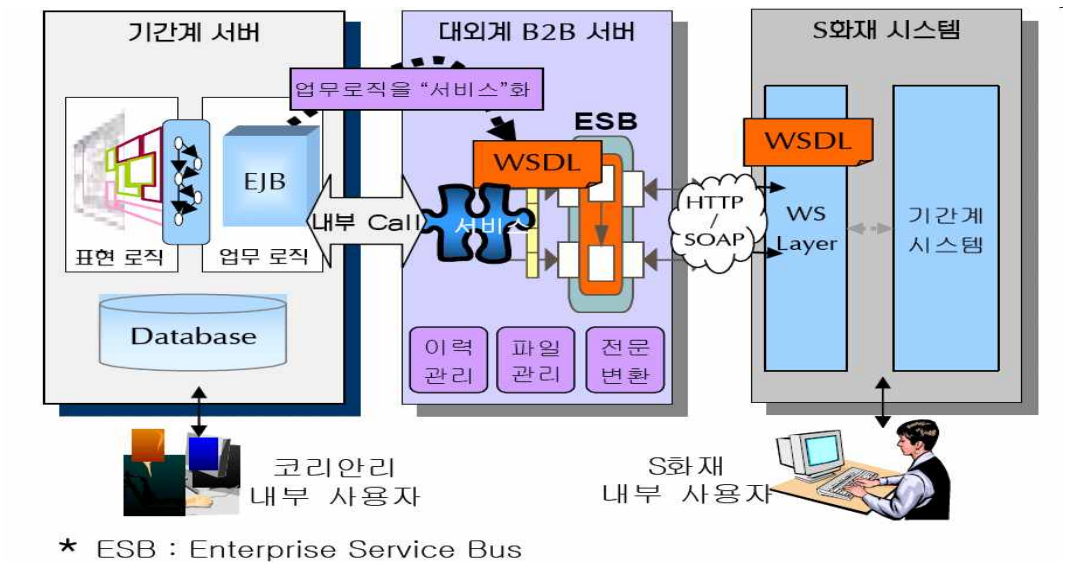
다. S화재

코리안리재보험과 S화재 간의 대외계 연결을 웹서비스 방식을 통해 구축함으로써 S화재 내부 사용자는 코리안리 시스템의 일부를 마치 S화재 시스템과 같은 느낌으

로 사용할 수 있고, 코리안리재보험은 재보험 관련 서비스를 시스템 간 연동을 통해 자동으로 S화재에 제공할 수 있게 되었다. 보험사간에 재보험 처리 프로세스를 SOA기반으로 개발하여 재보험 관련 B2B 업무를 양 사 시스템 간에 자동으로 처리될 수 있도록 신 대외계 시스템을 구축하였다.

웹서비스 기술을 적용하여 이기종 시스템 혹은 기업 외부의 다양한 파트너 들 간의 서비스 연계가 용이하고, 공개 표준인 웹 서비스 기술을 사용하므로 웹서비스 기술을 지원하는 다양한 채널 시스템과의 연계가 가능하여 기업 측면에서 볼 때 파트너나 고객사와의 가치 체인을 구축하여 기업 경쟁력을 강화시킨다. XML 기반의 메시지 포맷을 사용하기 때문에 첨부 파일 처리나 중점 테이블 형식의 전문 처리 등 복잡한 업무요건을 수용할 수 있고 ESB 구축을 통해 자원 관리의 효율성을 증가시킨다.

각각의 어플리케이션은 비즈니스 로직에만 충실하고 그 외의 인터페이스 관련 기능은 ESB가 담당하고, 기업 내의 웹 서비스 뿐만 아니라 방화벽 바깥의 웹서비스까지 사용자의 접근을 중앙에서 관리할 수 있어 IT 자원 재사용이 증가하여 TCO 절감 효과를 기대할 수 있다. 또한 선도적인 IT 표준 기술을 습득하고, 빠른 비즈니스 요건 변경에도 빠르게 적용할 수 있는 유연한 IT 운영환경을 구축함으로써 다른 보험사에서도 재사용 가능한 표준기반 공통 인프라를 구축하여 보험 산업의 중추 역할을 제공할 수 있는 허브 시스템을 기대하고 있다.



[그림 11] 웹서비스 기반 B2B 서버 구축 프로젝트

Ⅴ. SOA 기반 정보시스템 구축 방안

1. 추진 전략

가. 로드맵

SOA는 사실 어떤 특정 시스템이나 표준 방법론을 말하는 것이 아니고 시스템을 구현할 때 저변에 기반이 되는 철학과 같은 것이다. 따라서 모든 IT시스템 환경에 대해서 다양한 접근방법으로 SOA를 적용할 수 있다. 본 논문에서는 당행의 SOA 도입 적용을 검토하기 위해 SOA 구축 단계에 따른 로드맵을 제시하고 단계별 SOA 도입을 위한 과제와 추진전략을 수립해 보고자 한다.

[표4] 당행의 단계별[7] SOA 도입 로드맵

구분	1단계	2단계	3단계	4단계
	초기 도입 단계	어플리케이션 통합 단계	리엔지니어링 단계	성숙 단계
	2008년	2009년	2010년	2011년 이후
목표	SOA 전략 수립 (인터페이스 구현)	SOA 구현 (서비스 통합)	SOA 확산 (인프라 통합)	SOA 최적화
추진 내용	공동서비스 목록도출	공동 서비스 검증	통합 사업 추진	BPM, RTE[8], BI[9] 구현
	조립 서비스 식별	적용 범위 확정		
	파일럿시스템 구축	ESB, EMB 도입 검토		
	인터페이스 표준수립	인터페이스 프레임워크		
	SOA 문화 도입	SOA 문화 확립		
추진 결과	범위, 우선순위 선정	CSF[10] 도출	KPI[11] 자료	지능적인 시스템
평가 방법	인터페이스 효과분석	정성적 평가	정량적 평가 (IT ROI 효과분석)	IT-BSC[12]

7) 본 논문 'Ⅱ. SOA에 대한 이론적 고찰, 3. 구현 방법, 나. 구현 단계' 참고

8) 실시간기업(RTE:Real Time Enterprise)은 가트너가 제시한 개념으로 조직의 하부조직에서부터 최고 의사결정권자까지 모든 정보와 지식이 공유되는 환경에서 기업 내·외부 업무처리를 실시간으로 할 수 있도록 지속적인 프로세스 개선을 통해 경쟁력을 극대화한 기업

9) 신속하고 정확한 비즈니스 의사결정을 지원하는 지능적인 데이터 접근, 수집, 보관, 분석 등의 기술 체계로 사용자의 IT관련 지식 정도와 상관없이 용이하고 직관적으로 정보환경을 제공하며 이를 통해 생성되는 정보 역시 누구나 직관적으로 사용함으로써 조직의 경영전략 목표 수립에 지원

10) 핵심성공요인(CSF:Critical Success Factors): 조직경영 목적(Goal) 및 목표(Objective) 성취를 보장하는 주요 영역

11) 핵심평가지표(KPI:Key Performance Indicator): 매출이나 이익처럼 조직의 과거 재무실적만을 나타내는 지표가 아닌 미래 성과에 영향을 주는 여러가지 핵심지표를 묶은 평가기준

12) IT BSC(IT Balanced ScoreCard) : 비즈니스와 IT전략을 일치시켜서 재무효과, 내부프로세스, 고객관리, 학습 및 성장의 BSC 평가 기반에 대한 IT 성과측정

나. 추진 목표

본 장에서는 앞에서 살펴본 당행 정보시스템의 과제[13]를 기반으로 SOA 도입을 위한 추진 목표를 수립하고자 한다.

(1) 표준화

향후 SOA 기반 체제를 확립하기 위해서는 기존의 정보시스템의 표준화가 제일 중요한 과제이다. 비즈니스 프로세스를 서비스로 식별하여 공유 및 재사용 할 때 서비스 제공자와 사용자의 작업이 재작업 수준으로 많이 발생한다면 이는 SOA의 근본적인 목표와는 거리가 발생한 결과이기 때문이다. 요컨대 당행의 표준 개발 프레임워크를 구축하여 프로젝트 개발 시에 사용함으로써 표준기반을 마련해야 한다. 운영중이 정보시스템의 경우는 공동 서비스 도출 작업 과정에서 점진적인 표준화 작업이 수행되어야 한다.

(2) 통합 기능 관리

당행의 각 시스템에서 필요로 하는 공통 기능에 대해 통합 서비스로 구성하여 공동으로 사용할 수 있는 기반을 마련해야 한다. 예컨대 SSO[14] 연동과 관련해서 각 시스템별로 SSO Agent를 별도로 설치하는 방식은 시스템별로 비용과 유지보수가 각각 발생할 뿐 아니라, 접근 이외의 통합 권한 관리는 불가능하다는 문제가 있다. 실질적인 당행 전체의 통합 권한관리인 EAM[15]을 위해서는 논리적인 통합 단말 체제가 선행되어야 한다. 메뉴 상의 조합과 링크로 구성된 단순한 포탈이 아닌 SOA를 기반으로 Enterprise Portal이 필요한 것이다. 시스템별 각 업무(기능 및 화면) 단위로 RACI chart[16]를 구성하여 권한 메트릭스 테이블을 생성해 두고, 시스템별 SSO Agent가 아닌 통합단말에서의 SSO Agent 호출로 접근권한 이외에 기능별 권한 테이블을 체크하여 각 시스템에 결과를 분기하는 방식을 고려해 볼 수 있을 것이다.

---

13) 본 논문 'III. 당행 정보시스템 현황, 2. 정보시스템 운영현황과 과제, 나. 정보시스템 과제' 참고

14) SSO(Single Sign On): 한번의 로그인만으로 기업의 각종 시스템이나 인터넷 서비스에 접속하게 해 주는 보안 응용 솔루션. 각각의 시스템마다 인증 절차를 밟지 않고도 1개의 계정만으로 다양한 시스템에 접근할 수 있어 ID, 패스워드에 대한 보안 위험 예방과 사용자 편의 증진, 인증 관리 비용의 절감 효과가 있다.

15) EAM(Enterprise Access Management): SSO 기능과 관리자에게 다양한 접근 제어 기능을 동시에 제공하는 통합 인증 및 권한 관리 체계. 기업 자원과 정보에 대한 단일화된 인터페이스는 물론 개별화 기능, 정보와 지식에 대한 체계적 관리, 그룹웨어, 지식 관리 시스템(KMS), 경영진 정보 시스템(EIS), 공급망 관리(SCM), 기업 자원 관리(ERP) 등에 대한 단일화 창구 제공과 같은 도입 효과를 얻을 수 있다.

16) RACI는 Responsible(수행책임)의 R, Accountable(전결권한)의 A, Consulted(협의)의 C, Informed(통보)의 I 약자를 모아 만든 용어로, 각 시스템과 시스템 내부 각 기능 직무에 따른 역할과 책임을 총괄적으로 나타낸 테이블

(3) 인프라 통합

본 프로젝트에서는 테스트 DB서버만을 공유했으나, 당행의 최근 1~4년치 서버별 CPU 사용율 데이터를 분석해 보았을 때 최대값(peak)과 평균값을 고려하여 통합 관리가 가능할 것으로 예상된다. 서버를 공유하게 되면 해당 장비의 유지보수 및 패치관리와 관련 보안, WAS, DBMS 뿐만 아니라 각종 Tool 등의 소프트웨어를 공유할 수 있게 된다. 각각의 시스템은 더 이상 별개의 어플리케이션이 아닌 하나의 어플리케이션 내에서 구분되는 서비스로만 관리될 수 있을 것이다.

다. 추진 원칙

(1) 아키텍처 분할

기존의 시스템을 구성하는 세부 어플리케이션 레벨의 변경을 통한 통합이 아닌, 아키텍처 범위에서 분할을 통한 통합을 구현해야 한다. 복잡한 대형 시스템을 독립적인 기능을 가진 서비스로 분할해 전체 시스템을 보다 유연하게 만들 수 있는 요소로 다시 묶는 것이 필요하다. 이렇게 하면 종속관계를 갖지 않고 서로 독립적으로 존재하게 되어 한 부분에서의 변화가 시스템 전체적으로 영향을 미치지 않는다.

(2) 점진적 개발 및 적용

한꺼번에 모든 시스템을 SOA로 구현할 수도 있지만 한 부분부터 점진적으로 SOA를 적용함으로써 내부적인 혼란을 최소화 할 수 있다. SOA의 점진적 이행 능력은 SOA가 기존의 것을 뜯어내고 바꾸는 방식이 아니라 새로운 기술과 기존에 보유하고 있는 기술에 모두 적용할 수 있는 방식임을 의미하며, 여러 장소, 여러 시스템에 분산되어 있는 기능들을 통합하여 가능한 한 일관되고 정확한 시스템으로 통합시킬 수 있는 기회를 가질 수 있다.

(3) 재사용과 공유

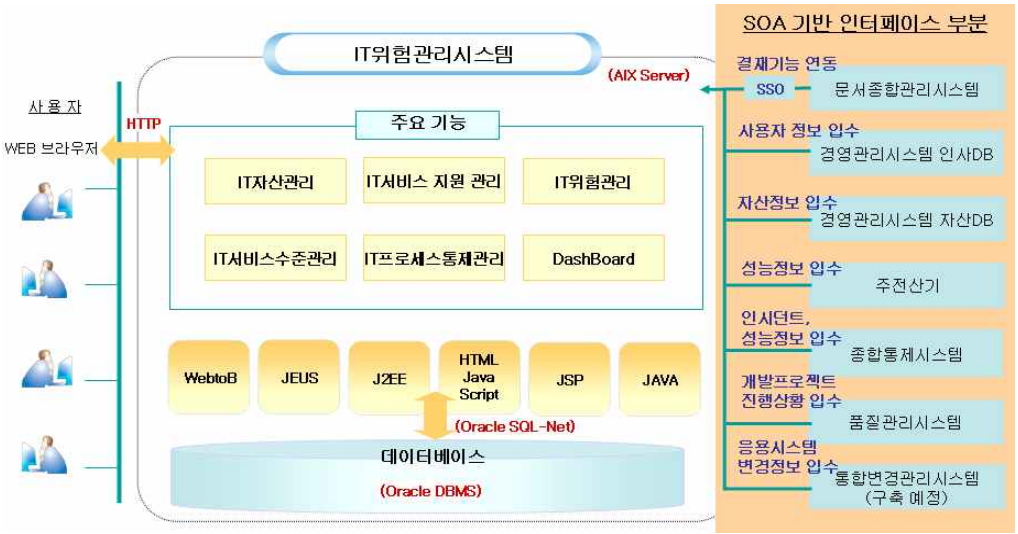
SOA 개념에서 재사용이 진정으로 의미하는 것은 공유다. 즉, 공유 정보 생성, 공유 프로세스 및 서브 프로세스 생성, 트랜잭션 공유, 데이터 공유 능력을 말한다. 이 모두가 SOA의 잠재적 효과의 매우 큰 부분을 차지하고 있다. 이와 같은 공유를 하게 되면 비용절감의 효과가 나타난다. 여러 번 개발하는 것이 아닌, 한 곳에서만 개발하고 이를 공통으로 사용하면 되기 때문이다. 그 한번을 정확하게 하는데 모든 역량을 집중할 수 있으면, 여러 번의 분산 활동으로 추진할 때 보다 실제로 더 좋은 결과를 가져 올 수 있다.

2. 파일럿시스템 구축

가. 파일럿시스템 대상 업무

당행의 SOA 도입을 검토하기 위해 올해 개발을 수행한 「IT위험관리시스템」개편 프로젝트에 SOA의 기본 사상을 적용해서 결과를 분석해 보고자 한다. 현재 시중에 나와 있는 SOA 엔진을 탑재한 솔루션을 사용하여 실행파일 공유수준의 완벽한 서비스 공유를 구현하지는 않았지만, 개발 범위내 충분히 소화할 수 있는 영역에서 SOA 개념을 활용해 보았다.

[그림 12] 파일럿시스템 구성도(SOA 기반 인터페이스부분 포함)



[표 5] 파일럿시스템 적용 대상

분야	내용
재사용 서비스	SSO 권한체크, 결재, 성능 및 장애정보 수집, 프로젝트 진척률 계산, 자산 및 인사정보 수집 등 서비스 호출 및 데이터 연동
개발 컴포넌트	AJAX기반의 Grid와 Chart 컴포넌트를 공통 사용하고, Tree와 FileUp/DownLoad 컴포넌트 재사용
통합 인프라 환경	별도의 테스트 DB서버를 구축하지 않고 문종관 및 경영관리시스템 등의 DBMS에 사용자 계정과 테이블스페이스만을 추가하여 공동 이용

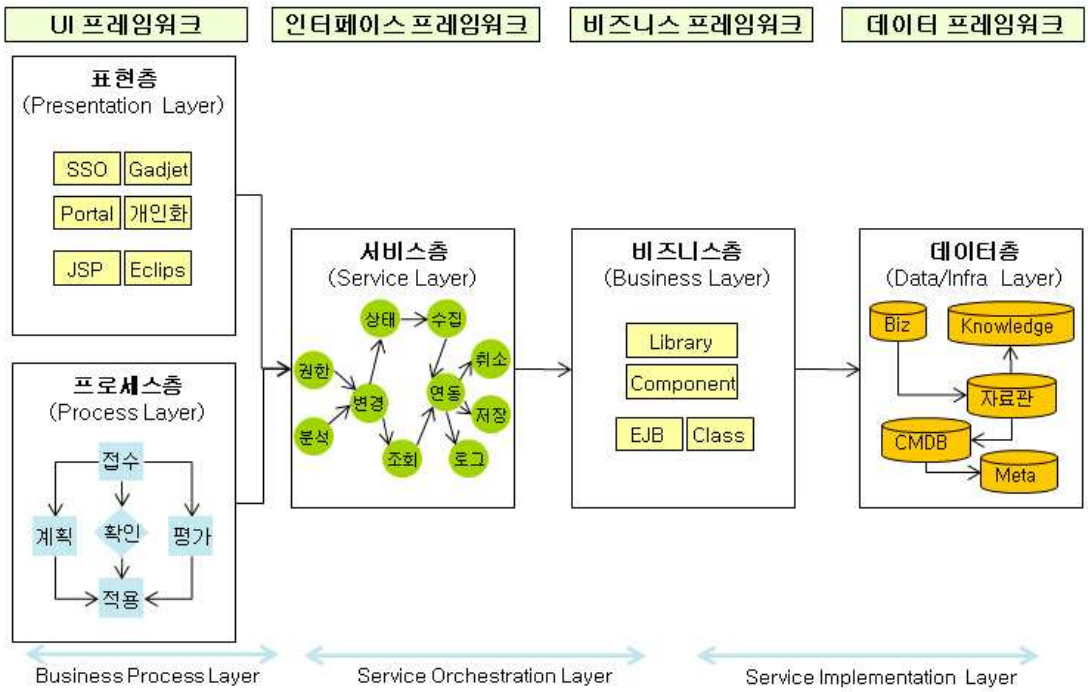
나. 시나리오 구성

SOA 개념을 도입한 파일럿시스템의 분석결과를 검토하기 위해서 본 논문에서는 사례를 들어서 설명하고자 한다. IT운영 업무를 수행함에 있어서 외부의 변경 발생으로 인한 관련 IT시스템의 변경 프로세스를 비즈니스 시나리오로 구성했다.

위의 사례에 대한 업무 처리는 세 가지로 구분 된다. '금융기관 공동코드 변경안내' 프로세스는 수작업 환경에서 우편 수신으로 처리되고, '테스트 및 운영 환경 변경적용' 프로세스처럼 직접 해당 시스템에서 발생하는 경우를 제외한 나머지 프로세스는「문서종합관리시스템」에서 실행된다. 어플리케이션 및 데이터에 대한 변경 프로세스는「변경영향평가시스템」을 참조하여「변경관리시스템」에서 수행하며,「IT위험관리시스템」의 DashBoard를 통해 서비스 전체의 상태를 조회할 수 있다.

다. 프레임워크 구성

본 파일럿시스템의 아키텍처는 분석된 기능에 대하여 UI, 인터페이스, 비즈니스, 데이터 4개의 프레임워크에 대응하는 표현층, 프로세스층, 서비스층, 비즈니스층, 데이터층으로 구분하여 구성했다.



[그림 13] 파일럿시스템 구축을 위한 SOA 프레임워크 구성도

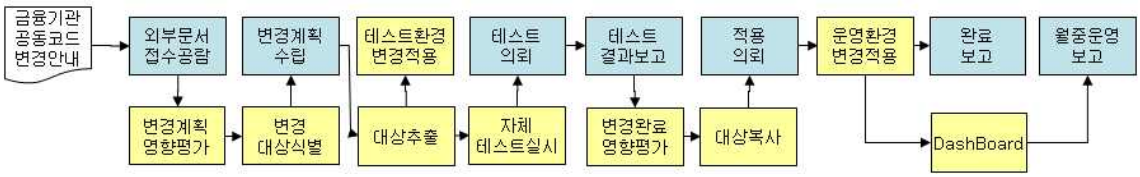
라. 세부 구축내용

파일럿시스템에서 구성한 아키텍처 각 계층을 SOA 통합 프레임워크(UI 프레임워크, 인터페이스 프레임워크, 비즈니스 프레임워크, 데이터 프레임워크)[17]에 매핑하여 세부 구축내용을 분석해 보고자 한다.

17) 본 논문 'Ⅱ. SOA에 대한 이론적 고찰, 3. 구현 방법, 가. SOA 프레임워크 체계' 참고

(1) UI 프레임워크

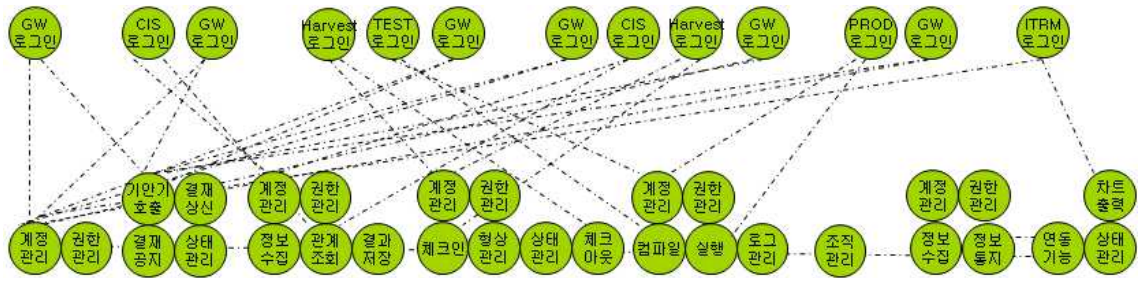
기존의 Client/Server 방식으로 개발된「IT위험관리시스템」의 UI를 Web 방식으로 개편하여 Presentation 영역에서는 Eclips를 사용하여 JSP로 구현하였다. 따라서 당행 그룹웨어인 문종관 연동을 손쉽게 가젯(gadjet) 소스를 구성하여 Iframe 전송 방식으로 포탈에 진입 메뉴를 구성하였고, SSO 연동도 최초 진입 포인트 4곳에 대한 SSO Agent의 필터링 부분만을 추가하는 것으로 구현이 가능했다.



[그림 14] 비즈니스 프로세스 사례

(2) 인터페이스 프레임워크

정의된 비즈니스 각 프로세스를 실행하기 위해서 수행할 다음 단계는 서비스를 식별하는 과정이다. 본고의 사례로 식별한 서비스 레벨은 매우 상위 수준이라고 할 수 있다. 보다 성숙한 SOA 기반에서는 더 작은 모듈 단위로 재사용 및 공유할 수 있는 서비스 단위를 세분화 시킬 수 있을 것이다.



[그림 15] 서비스 공동 사용 사례

위와 같이 사용자 인터페이스(UI)를 구성하기 위해서는 각 서비스 시스템 간의 연동이 필요했으며, 본 프로젝트에서는 서비스를 연동하기 위한 별도의 Adapter나 Gateway 구축 없이 EAI(Enterprise Application Integration) 방식으로 구현했다.

[그림 16] 서비스 공동 사용을 위한 인터페이스 사례

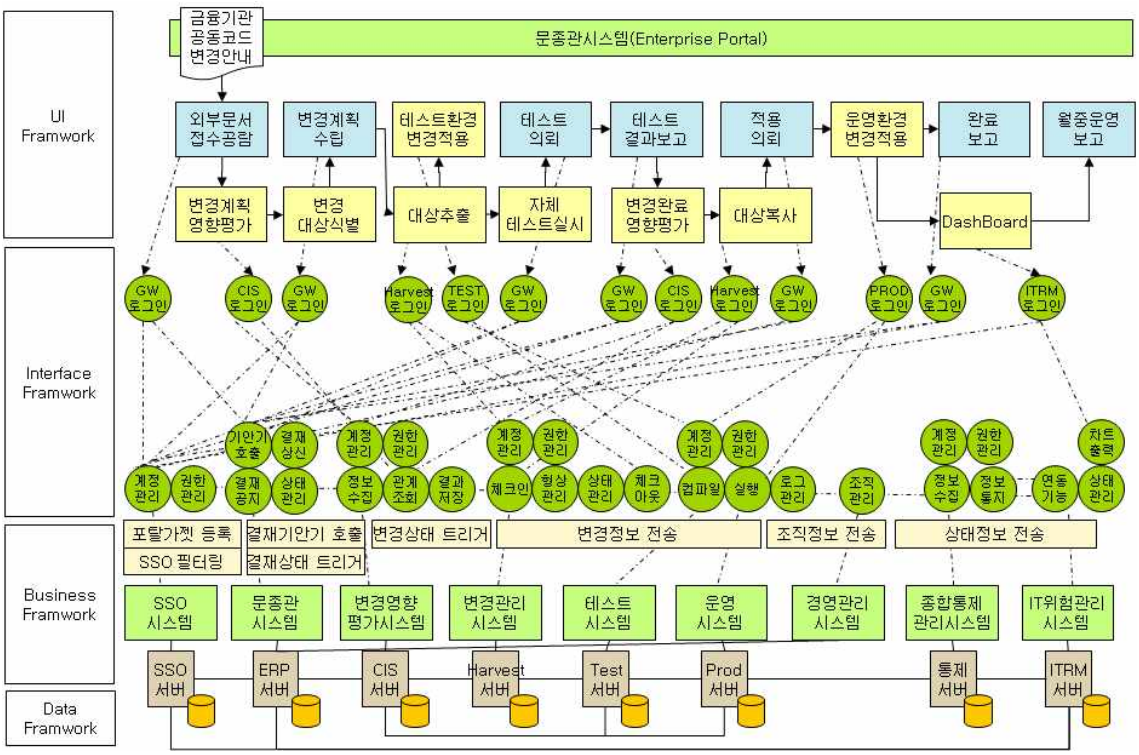
(3) 비즈니스 프레임워크

특히 본 개편 프로젝트에서 가장 주요한 점은 기존에 별도로 구성된 결재기능을 제거하고「문서종합관리시스템」의 결재 서비스를 호출하여 사용한 것이다. 이 방식은 향후시스템(통합변경관리시스템 KMS 등) 구축시 전자결재 연동에 적용하는데 있어 표준을 제공하게 될 것으로 기대한다.

IT정보자산에 대한 변경계획 수립 및 변경결과 적용 전 과정의 상태정보 연동에는 기본적으로 IT정보자산과 담당자 정보가 현행화 되어 있어야 하는데 이는「경영관리시스템」의 자산 및 인사 데이터 수집 서비스 실행으로 해결했다[18].

(4) 데이터 프레임워크

다음은 비즈니스부터 서비스와 각 응용시스템 및 데이터 인프라까지의 전체 구성이다. 데이터 프레임워크는 Technology Layer와 Data Layer로 구분하여 식별했다.



[그림 17] 파일럿시스템의 전체 프레임워크 구현 사례

18) 본 논문의 비즈니스 사례는 응용프로그램이나 데이터 변경과 관련한 변경과 관련 장애 발생정보를 입수하여 DashBoard에 보여주는 영역만을 식별했는데, 그 외에도 프로젝트에는 局內에서 관리하는 IT정보자산에 대한 용량정보, 장애처리 정보, 진행중인 프로젝트 정보를 각 시스템과 연동 및 데이터베이스 구축이 포함된다. 용량과 장애처리 정보는「종합통제시스템」의 용량 및 장애정보 수집서비스를 통해 구축한 데이터 연동으로 구현했고, 프로젝트 정보는「품질관리시스템」의 진척률 계산 서비스를 호출하여 진행중인 프로젝트 정보를 DashBoard로 구성했다.

마. 효과 분석

SOA 구축 초기단계 수준의 EAI기반 파일럿시스템을 수행한 결과를 바탕으로 ITROI 측면에서 효과를 평가하고, 향후 당행의 SOA 도입시 고려해야 할 사항을 제시하고자 한다.

(1) 효과분석 항목

본 논문에서는 파일럿시스템 전체 비용 중에 SOA 기반 적용에 해당하는 부분만을 아래와 같이 식별하여 효과분석에 사용했다.

[표 6] 파일럿시스템의 SOA 적용 목록

구분			비용효과
개발 및 운영측면	응용프로그램 연동	SSO 연동	소프트웨어 개발비 (데이터베이스 구축비용은 별도 산정하지 않음)
		결재 연동	
		인사,자산 연동	
		장애 연동	
		성능 연동	
		QMS 연동	
		변경 연동	
	유지보수	각 응용프로그램 정비	소프트웨어 개발비 × 유지보수 난이도(%) (재개발비는 별도 산정하지 않음)
	환경	테스트서버공용	시스템 라이센스 및 운영비
이용측면	중복작업 축소	문서 작성	문서 작업시간 단축
		결재 처리	결재 작업시간 단축
	시스템 안정성	신속한 장애복구	다운타임 축소
	사용자 만족도	상태 정보 제공	만족도 조사점수 향상(정성적 평가)

(2) 효과분석 세부 내용

소프트웨어 개발 및 운영 측면에서의 비용을 계산하기 위해서는 「소프트웨어산업진흥법」 제22조(사업대가기준) 규정에 의한 "소프트웨어사업 대가의 기준(정보통신부 고시 제2007-20호)"을 기반으로 했다. 소프트웨어 개발비 산정을 위한 개발 원가계산 산정은 기능점수[19]방식을 우선 적용하고 투입인력과 투입기간에 대한 산정방법으로 최종 비용을 검증했다.

19) '기능점수(function point)'란 업무 요구기능에 대해 개발관점에서 특정기술을 적용하여 개발한 소프트웨어 양을 의미하는 것이 아니고, 사용자 관점에서 소프트웨어 크기를 결정하는 소프트웨어 기능 유형별 수량과 성능 및 품질요인들의 영향도를 고려하여 소프트웨어의 양을 측정하는 방법

[표7] 파일럿시스템의 SOA 기반 개발비 산정 공식(개발 규모 기반)

소프트웨어 개발비 = 소프트웨어 개발원가 + 직접경비[20] + 이윤[21]
= (보정전개발원가 × 보정계수) + 직접경비 + 이윤

- ㅇ 보정전개발원가 = 데이터기능점수 + 트랜잭션기능점수
- ㅇ 데이터기능점수 = ∑(내부논리파일별 가중치) + ∑(외부연계파일별 가중치)
- ㅇ 트랜잭션기능점수 = ∑(외부입력별가중치) + ∑(외부출력별가중치) + ∑(외부조회별가중치)
- ㅇ 보정계수 = 규모보정계수×어플리케이션유형보정계수×개발언어보정계수×품질특성보정계수

유지보수비 = 소프트웨어 개발비 × 유지보수 난이도(%)

- ㅇ 유지보수 난이도 = 10 + 5 × (총점수 / 100)
- ㅇ 총점수(TMP) = 연간 유지보수횟수 12회초과(35점) + 연간 자료처리건수 10만미만(0점) + 타시스템 3개이상연계(10점) + 전문실무능력필요(10점) + 통합하의 분산처리(10점) = 65점[22]

[표 8] 파일럿시스템의 SOA 대상 신규개발 기능 목록

구분		서비스	데이터기능		트랜잭션기능			개발원가
			내부 논리	외부 연계	외부 입력	외부 출력	외부 조회	
		평균복잡도 가중치[23]	7.4	5.5	3.9	5.1	3.8	
응용 프로그램 개발	SSO 연동	로그인 및 세션 관리	0	0	0	0	1	보정전 개발원가 =(총기능×가중치) ×기능점수당단가[24] = 147.5 × 572,933 × 0.6 = 50,704,571
	결재 연동	결재문서 정보등록	1	0	0	0	0	
		기록물철 관리	1	0	0	0	0	
		결재선 선택 및 저장	1	0	0	0	0	
		결재상신,취소,확인	1	0	0	0	0	
		파일첨부 및 저장 기능	0	0	1	0	0	
		진행상태 관리	0	1	0	0	0	
	인사 연동	조직관계 등록	1	0	0	0	0	
		인사변경 관리	1	0	0	1	0	
	자산 연동	감가상각 관리	0	0	1	0	0	
		자산변경(이관,폐기)관리	1	0	0	0	0	
	장애 연동	인시던트 자동감지	0	1	0	0	0	
		인시던트 접수등록	1	0	0	1	0	
	성능 연동	성능정보 수집(Server)	0	1	0	0	0	
		성능정보 수집(M/F)	0	1	0	0	0	
		성능정보 등록	1	0	1	0	0	
	QMS연동	프로젝트 정보관리	1	0	1	0	0	
		진척률 계산	0	1	0	0	0	
	변경 연동	코드 관리	0	0	1	0	0	
		변경의뢰/계획/결과 등록	1	0	0	1	0	

20) '직접경비'로는 컴퓨터시스템 사용료, 소프트웨어 도구 사용료, 지급이자, 전문가 비용, 여비, 특수자료비, 인쇄 및 청사진비, 자료조사비, 기자재시험비, 위탁비 및 현장운영비, 모형비, 그 외의 소프트웨어 개발사업에 소요되는 직접비용이 있다.

21) '이윤'은 개발원가의 10% 범위 내에서 반영

22) 본 파일럿시스템 개발비 계산에 유지보수 및 재개발비는 합산하지 않음

23) 「소프트웨어산업진흥법」 제22조(사업대가기준) [별표7] 평균복잡도 가중치를 적용함

24) 「소프트웨어산업진흥법」 제22조(사업대가기준) [별표8] 단계별 기능점수당 단가 572,933원은 108,857(분석단계), 137,504(설계단계), 183,339(구현단계), 143,233(시험단계)의 합이며, 본 계산에서는 총금액의 60%를 적용함

[표 9] 파일럿시스템의 SOA 대상 신규 개발비

보정계수	규모[25]	300 기능점수 미만 → 0.65	총보정계수= 0.65×1.0×1.2×1.1 = 1.88
	어플리케이션유형	업무처리용 → 1.0	
	개발언어[26]	Java → 1.2	
	품질 및 특성	0.025 × 총 영향도[27] + 1 = 1.1	
소프트웨어 개발비 보정결과		소프트웨어 개발원가×총보정계수	50,704,571×1.88=**95,324,593(원)**

위의 계산 결과는 외부 시스템으로부터 연동하지 않고 순수하게 전체를 신규로 개발할 경우이다. 이중에는 연동을 하더라도 인터페이스에 소요되는 비용만을 산정하여 그 차액을 비교하는 것이 실질적인 비용효과를 측정할 수 있다.

[표 10] 파일럿시스템의 SOA 대상 인터페이스목록과 개발비

구분		서비스	데이터기능		트랜잭션기능			개발원가
			내부 논리	외부 연계	외부 입력	외부 출력	외부 조회	
		평균복잡도 가중치	7.4	5.5	3.9	5.1	3.8	
응용 프로그램 개발	SSO 연동	로그인 필터링	0	0	0	0	1	보정전 개발원가 =(총기능×가중치) ×기능점수당단가 = 86.7 × 572,933 × 0.6 = 29,803,975
	결재 연동	결재 기안기 호출	0	1	0	0	0	
	인사 연동	인사 DB 수집	1	1	0	0	0	
	자산 연동	자산 DB 수집	1	1	0	0	0	
	장애 연동	장애 DB 수집	1	1	0	0	0	
	성능 연동	성능 DB 수집	1	1	0	0	0	
	QMS연동	QMS DB 수집	1	1	0	0	0	
	변경 연동	변경 DB 수집	1	1	0	0	0	
소프트웨어 개발비 보정 결과			29,803,975 × 1.88 = **56,031,472(원)**					

따라서 신규개발을 하지 않고 공동서비스의 인터페이스를 통해 얻을 수 있는 소프트웨어 개발 비용 절감효과가 41.22%나 발생함을 알 수 있었다. 이는 유지보수 비용에도 절감의 효과가 있다. 동일 기능에 대한 오류 수정 및 개선을 한군데에서만 실행해도 해결할 수 있기 때문이다.

빅뱅식 프로젝트가 아닌 경우라면 SOA는 프로젝트를 완료한다고 곧바로 ROI가 나오는 것이 아니다. 초기에는 오히려 SOA 구축에 대한 비용이 증가하겠지만, 장기적으로 TCO 절감 효과가 점진적으로 높아질 것이다. SOA 프로젝트는 효과를 극대화할 수 있는 업무부터 시작해 프로젝트를 늘려가면서 보다 가시적인 효과를 볼 수 있을 것으로 기대된다.

25) 한 사업에서 여러개의 어플리케이션을 통합 구축시 재산정하는 [규모보정계수 = 0.108 × $\log_e$(기능점수) + 0.2229]으로 계산하되, 300 기능점수 미만인 경우는 0.65를 적용한다.

26) 본 프로젝트 비용 산정시 적용한 기능점수당 단가 572,933원은 108,857(분석단계), 137,504(설계단계), 183,339(구현단계), 143,233(시험단계)의 합으로 분석과 설계단계에는 언어보정을 적용하지 않아야 하는데, 전체 보정계수에 대해 각 개발단계를 분해하지 않고 총합으로 계산을 실시함

27) 「소프트웨어산업진흥법」 제22조(사업대가기준) [별표13] 품질 및 특성 보정계수를 적용함
- ○ 품질 및 특성 보정계수 = 0.025 × 총 영향도 + 1
- ○ 총 영향도 = 분산처리 영향도 + 성능 영향도 + 신뢰성 영향도 + 다중사이트 영향도

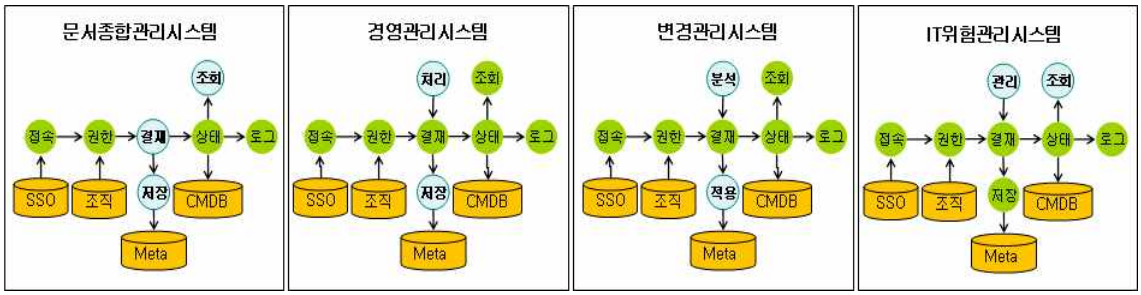
3. 향후 추진 방안

가. 단계별 추진 과제

파일럿시스템 결과분석을 통해 서비스 및 인프라 환경의 공동사용에 대해 상당한 효과가 있음을 살펴보았다. 아울러 당행의 SOA 도입 추진에 대한 방안을 SOA 구현 단개별[28] 과제를 도출해 보고자 한다.

(1) 초기도입 단계

당행의 SOA 구축에 대한 초기도입 단계를 완성하기 위해 기존 시스템간에 공동사용가능한 서비스 목록을 도출하여 최대한 중복 사용을 배제한 어플리케이션 표준 연동(Interface) 환경을 구축해야 한다.



[그림 18] 서비스 중복사용 현황

위의 프로세스 예는 현재 당행의 시스템별로 동일한 기능을 중복으로 개발해서 사용하는 모습을 보여주고 있다. 각 시스템별 동일 기능에 대해 인터페이스 표준을 수립하여 상호 연동할 수 있도록 하기 위한 기반 마련이 필요하다. 예컨대 각 시스템에서 별도로 구성된 결재기능 대신에 통합결재 서비스를 사용하고자 한다면 호출해야 하는 결재서비스 위치와 명칭, 필요한 접근 권한, Input/Output 파리미터 수와 유형, 관련 데이터(Triggering) 작업, 예외조건 등에 대한 인터페이스를 정의한 명세 작성 및 공유가 있어야 한다.

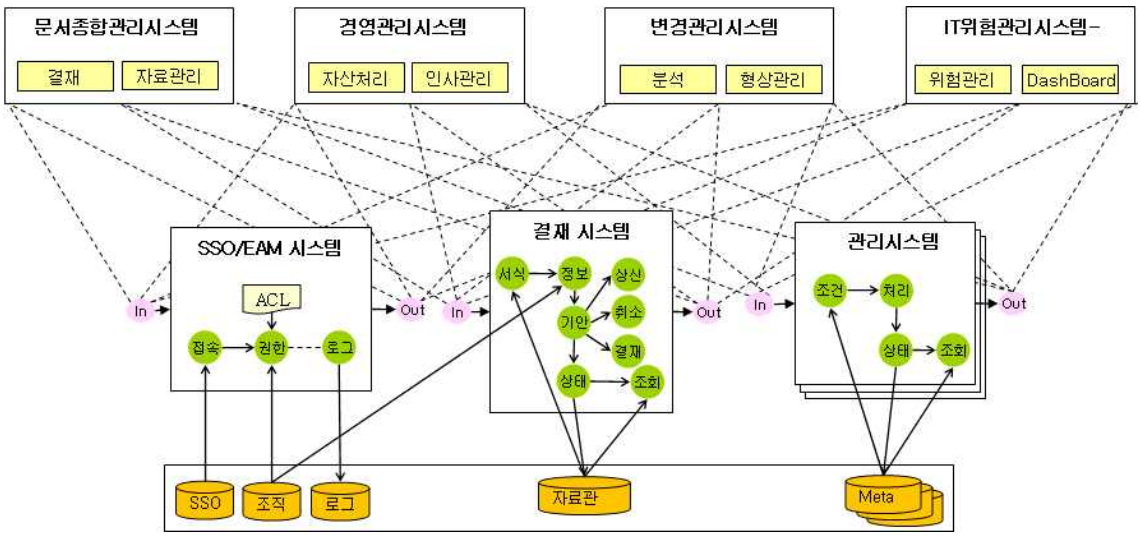
각 시스템별 공동 사용(Request/Provide) 가능한 단위 서비스 목록을 도출하고, 각 시스템 간의 인터페이스 정도에 따라 단위 서비스들의 조립(Composite) 정도를 결정하여 점진적으로 시스템 간 연동을 실행할 수 있을 것이다.

---

28) 본 논문 'Ⅱ. SOA에 대한 이론적 고찰, 3. SOA 구현 방법, 나. SOA 구축방법' 참고

(2) 어플리케이션 통합 단계

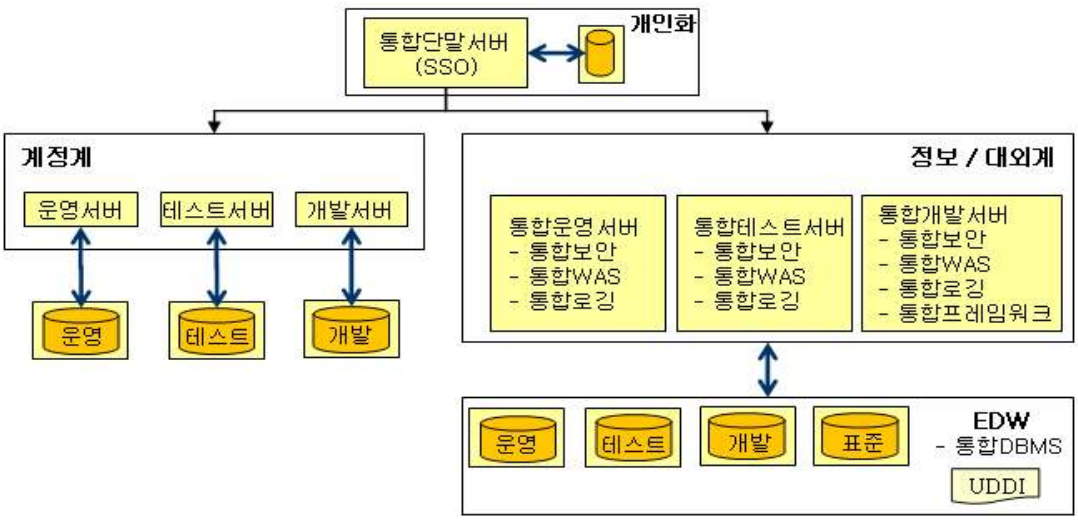
각 시스템간 개별적인 연동이 확산되면 공동서비스를 독립된 서비스로 구성하여 연계(Interface)를 넘어서 통합(Integration)의 개념으로 시스템 환경을 구축해야 한다.



[그림 19] 서비스 공동사용 모형

(3) 리엔지니어링 단계

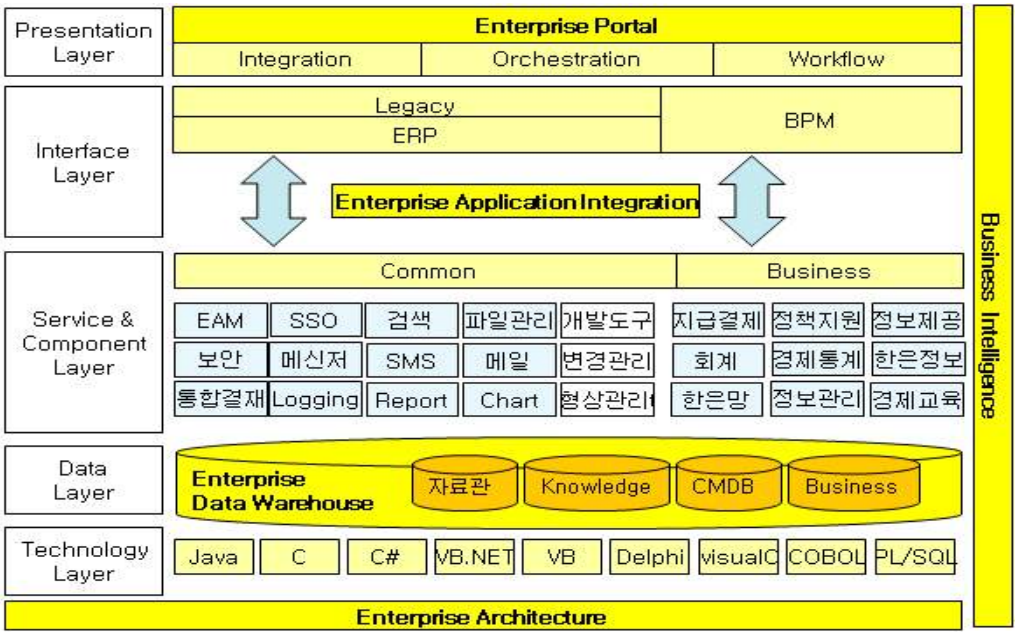
개별 비즈니스를 기본 단위로 하는 응용시스템과 서비스가 철저하게 분리된다면 분리 각 서비스들을 지원하는 응용시스템의 인프라 구성은 논리적 또는 물리적으로 통합하여 사용할 수 있을 것이다.



[그림 20] 인프라/데이터 통합 모형

(4) 성숙 단계

SOA 성숙 단계에서는 당행 IT 청사진인 EA(Enterprise Architecture)를 기반으로 IT자산 각 요소들이 SOA 사상을 기반으로 연결되면 안정적인 IT정보 운영과 효과적인 의사결정 지원을 이루는 BI(Business Intelligence)를 실현할 수 있게 될 것으로 기대한다.



[그림 21] SOA 기반의 Enterprise Architecture 모형

나. 당행의 SOA 도입 분야

앞에서 수립한 로드맵에 의거하여 당행의 SOA 초기 도입 단계에서 SOA 도입 프로젝트 조직을 구성해야 하다. SOA 프로젝트 팀 구성에 있어서는 전행 차원의 공동 서비스 기능을 도출하여 표준을 수립해야 하므로 일반적인 분석·설계·구현·품질 조직 외에 인터페이스 표준을 전담하는 조직이 추가로 요청된다. 또한 SOA 프로젝트는 파일럿시스템을 통해 도출된 공동 서비스에 대한 인터페이스를 검증하여 점진적으로 반복·확산하는 방법으로 구현하는 것이 성공의 관건이다.

특히 도입 분야를 결정할 때는 대상 시스템의 현재의 운영환경 및 향후 확장성을 고려해야 한다. 본 논문에서는 회계·결제 등 동기식 처리를 기반으로 하는 시스템은 기 구축된 안정된 플랫폼을 그대로 사용하는 것이 효율적이라고 판단하여, 비동기식의 정보계 시스템 중에서 타 시스템과의 연관성이 크고 재사용성이 높은 서비스를 중심으로 SOA 초기단계 모형으로 구축할 수 있는 대상을 도출해 보았다.

[표 11] SOA 도입 대상 서비스 목록

구분	기능	제공 서비스	활용 시스템
운영	계정관리	통합 로그인(SSO)	정보시스템 공통
	권한관리	EAM	
	보안관리	DBMS, OS, Network 보안 SW	
	성능관리	성능 모니터링	
	변경관리	형상관리(백업 및 이력관리) 버전관리(CheckOut/CheckIn[29])	
	전자결재	그룹웨어(문서종합관리시스템)	경영관리시스템, IT위험관리시스템, 변경관리시스템, 지식관리시스템 등 결재기능을 보유한 정보시스템
	검색	행내각종 DB검색(시스템 Agent)	모니터링시스템, IT위험관리시스템
		통합검색엔진	문서종합관리시스템, 자료관, 경영관리시스템, IT위험관리시스템, 기타 행내시스템과 홈페이지, KMS
		서지정보검색	전자도서관
		법률정보검색	법규정보시스템
		외부포탈지식검색	지식관리시스템
개발	영향평가	변경영향평가[30]	IT위험관리시스템, 변경관리시스템, 품질관리시스템
	테스트	소스검사도구(Resort for Java/C[31])	변경관리시스템, 품질관리시스템
		기능테스트	개발 프로젝트
	컴포넌트	통합개발 프레임워크(디자인패턴[32])	
		데이터베이스 연결	
		게시판, 자료실, 공지사항, 메일	
		팝업(사용자, 부서, 코드 검색 등)	
		차트그래프, 보고서 등	
		그리드(엑셀), 파일저장 기능 등	
		입력값 체크	
인프라 환경	SSO서버	통합 로그인	정보시스템 공통
	DB서버	통합 DBMS	시스템별로 시간대별 CPU, Memory, File I/O 등의 사용율을 분석하여 통합대상 선별
	WAS서버	통합 웹서비스	

29) 프로그램 변경이 필요한 경우, 최종 소스버전을 실제 운영환경으로부터 반출(Checkout)해서 이를 대상으로 테스트 환경에서 변경 작업을 수행하며, 작업이 완료된 버전은 통제된 환경에서 변경절차에 의해 운영환경에 반입(CheckIn)하여 적용하는 과정

30) 변경계획 수립 및 적용 전에, 프로그램 및 데이터의 변경 대상 설정과 적용 확인을 위한 소스대 소스, 소스대 데이터 간의 영향(포함, 참조, 연계 등)을 평가하는 도구

31) 프로그램 소스에 대하여 개발언어(Java, C)별로 사전에 정의된 개발 표준 준수율을 체크하는 도구

32) 프로그래머가 소프트웨어 개발시 사용하는 data Model(프로그램 실제 계산), View(사용자 인터페이스 표현), Controller(사용자와 View 간의 상호작용) 집합으로 프로그램 재활용 구성 단위가 됨

VI. 결론

1996년 SOA 개념을 처음 소개한 가트너 그룹은 2008년까지 신규 개발 프로젝트의 80%이상이 SOA를 기반으로 개발될 것으로 전망하고 있다. 아울러 SOA가 소프트웨어 시장 뿐만 아니라, 기업환경 전반에 걸쳐 최고의 화두가 되고 있는 것은 SOA야말로 급변하는 비즈니스 환경변화에 대응할 수 있는 최적의 대안이라고 보고 있기 때문이다.

비즈니스 환경변화에 대한 신속한 대응

최근의 가장 두드러진 비즈니스 환경변화는 비즈니스 생태계(ecosystem)의 개념과 실시간 기업(Real-time Enterprise)의 등장으로 볼 수 있다. 비즈니스 생태계란 비즈니스 환경이 과거 독립적인 조직 및 프로세스에 의해 주도되는 수직적 통합에서 고객, 공급자, 파트너 등 다수 기업과의 협업관계가 중시되는 환경에서 유기체로서의 기업을 의미한다. 그리고 실시간 기업이란 기업의 기회, 위협요인을 경쟁사보다 먼저 파악해 효율적인 의사결정을 하고 민첩하게 대응할 수 있도록 하는 개념이다.

이러한 비즈니스 환경에서는 경쟁력 있는 기업은 급변하는 비즈니스 환경변화와 시장요구에 민첩하게 대응할 수 있어야 한다. 이를 위해서는 실시간 정보공유가 가능하도록 적재적소에서 데이터를 수집하고 접근하여, 업무 프로세스의 중단이 없도록 하는 것이 필수적이다. 즉 기업내 모든 어플리케이션과 데이터가 필요할 때 곧바로 접근할 수 있는 통합 환경이 되어야 한다. 현재 수준에서 이러한 실시간 기업에 대한 요구에 가장 적합한 것이 바로 SOA인 것이다.

IT자원의 상호 운용성 및 재활용성 증대를 통한 비용절감

최근 들어 기업의 IT 투자는 비용절감과 효율성 증대를 중시하는 경향이 있다. 그동안 IT부서는 복잡한 전산시스템 구조를 단순화시키기 위해 시스템과 어플리케이션을 통합하는데 여념이 없었다. 대규모 투자로 통합한 뒤의 고민은 과연 기존 IT자원을 어떻게 재활용할 것인가로 귀결된다. 그리고 기업의 기존 시스템들은 ERP, CRM, SCM 등 다른 시스템과 통합, 연계가 부족하여 실제적으로 고객서비스를 위한 프로세스가 단절되고 여러 시스템에 로그인해야 하는 불편함이 있었다. 이렇듯 프로그램 중심에서 프로세스 중심으로 설계하고 재사용할 수 있는 적정한 서비스 단위를 개발하는 것이 변화에 민첩하게 대응할 수 있는 관건이 되고 있다.

SOA 기반 웹서비스는 서비스 단위로 재활용 가능하므로 특정 프로세스나 서비스 변경 시 프로그램을 재개발하지 않고 새로운 기능을 제공하는 서비스로 교체하여

사용할 수 있다. 이러한 SOA의 강점을 활용하여 기업이나 공공기관이 내부 또는 외부시스템과의 비즈니스 통합 시 적용되거나, 서비스의 조합을 통해 새로운 서비스를 창출하여 새로운 비즈니스 기회를 창출하는데 활용되고 있다.

당행에서도 시행착오를 줄이고 SOA를 구축하기 위해서는 실질적인 구현 능력을 보유하고 SOA 방법론도 반드시 필요하다. 당행의 다양한 업무추진 과제는 결국 IT 시스템의 유연성에 대한 요구를 의미한다. 사용자가 늘어나고 사용자의 유형이 다양해지면서 자연히 시스템의 규모가 커지고 복잡해지게 됐다. 따라서 다양한 메커니즘이 구축되면서 유연성의 문제는 해결이 더욱 힘들어지고 있다.

SOA는 점점 복잡해지고 있는 시스템을 다시 유연하게 만들고, 비즈니스의 필요사항에 대응해 사업전략을 구현할 수 있는 방식이다. 또한 SOA는 어플리케이션 통합 문제에 대한 해답이기도 하다. 다양한 어플리케이션 포트폴리오를 보유한 조직은 기존의 어플리케이션들을 연계해서 보다 통합된 업무 프로세스를 창출할 수 있다. 뿐만 아니라 다양한 환경의 어플리케이션 개발 문제도 SOA를 도입하도록 하는데 하나의 요인이 되고 있다. 기존에 개발된 것을 재사용하여 쓰기 쉬운 IT시스템을 만들기 위해서 IT조직이 보유한 기술을 가장 효과적인 방법으로 활용할 수 있어야 하기 때문이다.

당행도 SOA 기반으로 정보시스템을 구현하면 앞에서 언급한 효과분석 사례[33]와 같이 시스템의 유연성 확보와 일관성 있는 기능 제공 및 재사용 증대 등이 결과를 얻을 수 있다. 이는 또한 담당직원의 개발 및 운영에 필요한 정보습득 시간을 단축시킴으로써 각 시스템별로 별도의 운영용역 인력을 사용하고 있는 현실에서 통합 유지정비 체제로 전환도 가능할 것으로 판단된다. SOA의 초기 도입단계를 적용하는 경우에도 이러한 효과를 볼 수 있으며, 향후 추진되는 정보시스템의 개발업무는 SOA 기반에 의할 경우 중장기적으로 전체적인 IT인프라에 대한 효율성을 제고할 것으로 기대된다.

33) 본 논문 'V. SOA 기반 정보시스템 구축방안, 2. 파일럿시스템 구축, 마. 효과분석' 참고

< 참고문헌 >

경영과컴퓨터, “SOA방법론과 시장진단”, 경영과컴퓨터, 2006.7

고만석, “기업과 컨택센터 환경에서의 인텔리전트 커뮤니케이션”, IDC Conference, 2007

고원규, “SOA 시장, 어디쯤 와 있나”, DBguide.net, 2007

권윤태, “ESB/BPM을 통한 SOA Service Layering 구현전략”, IDC Conference, 2007

김성익・박정일, “SOA 프레임워크 아키텍처”, 정보과학회지 제25권 제1호, 2007.1

김영주, “사례중심의 SOA 구축전략”, 2006 SOA/U-city/RFID/USN/VSE 성공사례 구축 Solution Forum, 2006

마이크로소프트웨어, “화려하고 새로운 아키텍쳐의 유혹에서 벗어나라”, 2002.12

박대연, “4대 Framework를 통한 4세대 컴퓨팅의 구현”, CIO, 2007

박범순, “엔터프라이즈 SOA & 신속한 혁신”, IDC Conference, 2007

백종현・김형석・김영호・한상인, “SOA 플랫폼 분석과 시장전망”, 정보과학회지 제25권 제1호, 2007.1

서경기, “유연한 비즈니스를 위한 유연한 IT환경SOA”, 정보과학회지 제25권 제1호, 2007.1

신수미, “SOA 기반의 정보시스템 설계 및 구현”, 한국과학기술정보연구원, 2005

양희정, “3I(Interface, Integration, Intelligence)의 중요성”, 경영과컴퓨터, 2005.2

양희정, “IT 투자성과 평가모델 개발 연구”, 행내논문, 2006

유영석, “EA와 SOA는 참조(Reference)가능한 아키텍쳐 모델인가?”, 2007

이경하・이규철, “Service Oriented Architecture와 웹서비스”, 정보과학회지 제22권 제10호, 2004.10

이상민, “Oracle SOA Suite 적용사례”, Oracle, 2006

이석진, “SOA 적용 전략과 적용 방안”, Oracle Technology Summit, 2006

이성규・진찬욱・김태석, “서비스 지향 아키텍처를 기반으로 한 웹서비스 시스템 모델링”, 한국시뮬레이션학회 논문지 제16권 제1호, 2007.3

이순환, “SOA개념 및 도입전략”, LGCNS, 2007

이재경・박성환・방제성・이한민, “서비스 지향구조 개념을 적용한 통합 설계 시스템 개발에 관한 연구”, 韓國情密工學會 2005년도 春季學術大會論文集, 2005

이재원, “SOA 환경에서의 프로세스 유연성 극대화를 위한 솔루션 및 방안”, IDC Conference, 2007

이주영, “SOA 구현을 위한 BPM 적용방안 및 구축사례”, TmaxSoft, 2007

이진권, “Beyond Next Generation System, Beyond SOA”, IDC Conference, 2007

이헌중・신신애・안현수・변현진, “SOA 기반 전사적아키텍쳐 접근방안”, 정보과학회지 제22권 제10호, 2006

임철홍, “서비스 컴포넌트 아키텍처”, 정보과학회지 제25권 제1호, 2007.1

임철홍・홍도석・최정순, “ESB기반 SOA Application에 대한 S/W Architecture 관점의 평가와 개선방안에 대한 연구”, 韓國IT서비스學會誌 제5권 제2호, 2006.8

정보통신부, “2007 소프트웨어사업 대가의 기준해설”, 2007.6

정해영, “SOA와 정보통합”, IBM, 2005

조재훈・최우용・이상완, “Implementing Service Oriented Architecture as an upcomming IT standard for Business Integration”, IBM, 2006

최은만, “Service Oriented Architecture와 재사용”, 정보과학회지 제24권 제11호, 2006.11

한국정보산업연합회, “Enterprise Archtecture”, 2007.6

한상우・박선희・노재호, “Service Oriented Architecture 적용을 위한 서비스 식별 기법”, 정보과학회지 제24권 제11호, 2006.11

홍정기, “SOA에 기반한 차세대 채널 인티그레이션 로드맵과 거버넌스”, IDC Conference, 2007

Chase Hacker, “Trends, Tales & Truths In Enterprise Intelligence”, IDC Conference, 2007

HandySoft Consulting team, “情報시스템의 開發 및 運營의 最適化를 위한 方案 考察”, HandySoft, 2007.9

IDC, “IDC's Asia/Pacific Business Optimization : BI, BPM & SOA Conference 2007”, IDC Conference, 2007

Kim Yong Jin, “Service Oriented Architecture Under the Magnifying Glass”, Gartner, 2006

Soonyul Chang, “Business Innovation beyond SOA”, IDC Conference, 2007

TmaxSoft, “SOA 기반의 금융차세대시스템 구축방향과 전략”, TmaxSoft, 2006.12

TmaxSoft, “SOA Guide”, 경영과컴퓨터, 2007