3

서비스 기반 통합, 마스터 데이터 통합 및 사례

LG CNS ▌장양자 · 김상수 · 조수형*

1. 서 론

최근 기업들은 다운사이징, 인수합병, 글로벌 경쟁과 불확실성 하에서 경쟁력 제고와 더불어 수익성 개선을 위한 방법을 모색하고 있다. 일례로, '매킨토시'로 명성을 얻은 애플(Apple)은 지난 1997년에 10억7천만 달러의 순손실로 파산 위기에 있었지만, '아이팟(iPod)'이라는 MP3 플레이어를 통하여 성공하고 있다[1].

이러한 기업 변화의 성공적인 모습을 한마디로 실시간 기업(Real Time Enterprise)또는 비즈니스 생태계(Business Ecosystem)라고 표현하고 있다[2].

실시간 기업이란 기업의 기회, 위협 요인을 경쟁사보다 먼저 파악하여 효율적인 의사결정을 통해 민첩하게 대응할 수 있도록 하는 개념이고, 비즈니스 생태계는 과거 독립적인 조직 및 프로세스에 의해 주도되는 수직적 협업에서 고객, 공급자, 파트너 등 다수와의 관계적 협업이 중시되는 환경에 유연하게 대처할 수 있는 기업을 의미한다.

실시간 기업과 비즈니스 생태계는 기업의 외부 환경의 변화에 따라 기업의 업무 흐름이 수시로 변화하고, 고객과 공급자 등 모두를 고려한 어플리케이션 통합 모델이 필요하다. 이러한 어플리케이션 통합을 위해서 가장 중요한 요소는 공동 이용 가능성이며, 이는 특정 업체나 플랫폼에 종속되지 않는 기술이어야 한다. 아울러 기존의 정보 시스템 투자를 보호할 수 있어야 하며, 개발자의 현재 보유 기술을 계속적으로 활용하고 응용해야 한다. 통합 기술로 대표되는 것이 바로 서비스 지향 아키텍처(Service Oriented Architecture, 이하 SOA로 함)이다.

또한 어플리케이션 통합 모델에서 중요한 것이 정보이다. 요즘 차세대 어플리케이션을 구축하면서 주요 어플리케이션들에서 공통으로 사용되고 있는 고객 및 제품 등 마스터 데이터 설계와 마스터 데이터의 값에 대한 중복성을 제거하는 마스터 데이터 통합(Master Data Integration, 이하 MDI로 함)이 주목 받고 있다. MDI는 주요 어플리케이션에게 동일한 마스터 데이터에 대한 서비스를 제공하며, 데이터 품질이나 중복 문제를 줄임으로써, 주요 비즈니스 업무의 재작업 및 정확성을 높일 수 있다.

2. SOA와 서비스 인프라

기업의 정보 시스템 분야에 SOA 개념을 적용하기 위해서는, 표준 기반 플랫폼과 기업 전반의 어플리케이션의 아키텍처를 위한 공용 기술 인프라를 제공할 수 있는 새로운 범주의 소프트웨어가 반드시 필요하게 된다. 이러한 범주 중 하나는 '업무 간 정보, 서비스 및 프로세스의 자유로운 흐름을 가능하게 하는, 독립적인 플랫폼'으로, "서비스 인프라"로 정의되고 있다[3].

2.1 아키텍처

SOA, 소프트웨어 아키텍처 등 여러 용어에서 사용되는 아키텍처는 여러 의미로 사용되고 있어, 아키텍처에 대한 정확한 개념을 설명하기는 쉬운 일이 아니다. 어플리케이션 개발 측면에서 아키텍처는 두 가지 모습을 가지고 있다. 아키텍처는 어플리케이션의 설계 및 구현을 위한 계획(Plan)을 지원하는 측면과 시스템의 복잡성을 관리하기 위해 추상화(Abstraction)를 제공하는 측면이 있다.

어플리케이션 개발에서의 아키텍처는 개발과 분리해서 작업이 이루어지는데, 이는 선행 작업으로 진행되어 어플리케이션의 설계 및 구현을 위한 계획으로 사용된다. 설계 및 구현 계획은 일반적인 S/W 프로젝트의 계획과는 다르다. 이는 어플리케이션을 S/W 측면에서 구성요소를 기술하고, 이들이 어떻게 협력하고, 또한 요구사항을 만족하는지를 확인하기 위한 구조적인 계획이다. 이것은 시스템 개발하는 과정에 청사진 역할을 한다.

아키텍트는 아키텍처의 구조라는 면에서 빈번하게 발생하는 문제와 해결책을 위해 패턴(Pattern)을 정의

* 정회원

하고 있다. 이러한 패턴은 비즈니스 업무(Operation)에서, 소프트웨어 구조, 설계 구조, 코드 구조까지 여러 단계로 적용 가능하다. 소프트웨어 구조에서 사용되는 아키텍처 패턴은 컴포넌트(Component) 혹은 서브 컴포넌트사이의 상호 관계를 다룬다. 현재 어플리케이션에서 가장 많이 사용되는 패턴은 계층 패턴(Layered Pattern)으로 계층(각 계층은 그 상위 계층에 서비스를 제공하고, 하위 계층에게는 클라이언트 역할을 함)이라는 컴포넌트와 프로토콜이라는 커넥터로 이루어진다. 일례로, 이는 재사용(Reuse)과 이식성(Portability)이 좋으며, 추상화 레벨에 기반을 둔 설계를 보다 쉽게 지원한다.

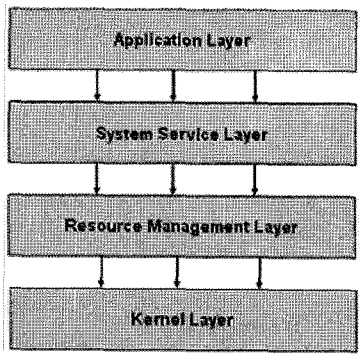


그림 1 계층 패턴

아키텍처의 추상화 측면은 복잡성을 관리하는 것이다. 아키텍처는 시스템의 구성요소로 분해하는 것 이외에도, 각 구성요소에 대해 인터페이스와 구조 자체에 대해서도 고려해야 한다. 아키텍처에서 가장 고려되는 것은 확장성, 재사용성과 성능 등이다. 그러므로 아키텍트(Architect)는 어플리케이션의 기능이 확장할 수 있도록 설계해야 하고, 어플리케이션의 변경과 이식성(Portability)을 보장해야 한다. 예를 들어, 어플리케이션에서 사용되는 EJB(Enterprise Java Bean)에서의 데이터 액세스 부분이 현재 가능한 기술이 JDBC(Java Database Connectivity)이고, 조직의 기술 원칙에 의해 EJB의 CMP(Container Managed Persistence) 기능으로 정해져 있다면, 아키텍트는 데이터 액세스 부분이 언제든지 CMP 기능을 이용할 수 있도록 설계해야 한다[4].

2.2 SOA

전통적인 어플리케이션 개발 접근방법은 각 소프트웨어 구성요소를 구현하는데 적합하였지만, 기업의 실시간 기업과 비즈니스 생태계 환경은 각 어플리케이션의 구성요소를 개방된 환경으로 변화시키고 있다. 그러므로 SOA에서 사용하는 어플리케이션의 아키텍처는 각 구성요소의 자율성(Autonomy)과 이질성(Heterogeneity)이 중요시되고 있다. 가트너(Gartner Research)에 의하면, SOA로의 변화는 과거 단말기 기반 아키텍처에서 클라이언트/서버 아키텍처로의 변화에 견줄만한 사고의 틀을 변화시키는 것으로 예견하고 있다[5].

구체적인 의미에서의 SOA는 변화에 적절히 대응할 수 있는 어플리케이션을 구축하는 개념으로, 어플리케이션을 구성하는 내부 업무 프로세스, 데이터와 기능을 각각 '서비스'라는 기본적인 기능 단위로 나누고, 이들 '서비스'를 연결하여 원하는 기능을 제공하는 것이다.

또한 SOA는 내부 어플리케이션과 서비스를 통합하는 것은 물론이고 협력사와 하도급 업체 등의 외부 어플리케이션까지 연계한다.

2.3 서비스 인프라

어플리케이션의 모든 기술을 하나의 업체 기술로 제한하지 않는, 통합된 가상(Virtual) 인프라에서 각 어플리케이션의 구성요소를 조합(Composition) 및 연계(Integration)할 수 있는 모듈화(Modularity)와 유연성(Flexibility)에 대한 관심이 높아지고 있다[3].

또한 어플리케이션 차원에서 IT를 바라보던 관점이 '서비스 전달(Service Delivery)'차원으로 변화하고 있다. 이에 따라 IT 부서는 임직원, 고객, 파트너 및 공급자가 신속하게 사용할 수 있는 새로운 서비스를 생성, 조합 및 전달할 수 있는 구조를 제공해야 한다.

현재 대부분의 기업은 초기 SOA 프로젝트의 구축 및 배치를 위해 어플리케이션 서버, 통합 서버, 개발 도구 및 포털 소프트웨어와 같은 각자의 어플리케이션 인프라 소프트웨어를 사용하고 있다. 하지만 기업은 이러한 서비스를 더욱 신속하게 작성, 전달, 구성 및 관리할 수 있는 새로운 종류의 소프트웨어 인프라가 점점 요구되고 있다. 기업은 많은 서비스를 추가 구축 및 배치할 때 이러한 요구를 경험하는데, 이런 경우 지속적인 통합이 필요하며, 확장이 어려운 "서비스의 불규칙 확산"이라는 상황에 빠진다. 또한 예를 들어 "자동차 부품 제작"과 같은 전통적인 코딩 도구 외에도 자동차 제작을 위한 "조립 라인"에 더 가까운 새로운 복합 도구 모음이 필요하다[3].

이러한 요구를 해결하기 위해 미들웨어 플랫폼 벤더들은 '업무 간 정보, 서비스 및 프로세스의 자유로운 흐름을 가능하게 하는 새로운 범주의 서비스 인프라 소프트웨어'를 지원하고 있다. 서비스 지향 아키텍처를 기반으로 하는 이 소프트웨어에는 COLA(Composed Once and Leveraged Anywhere) 방식의 서비스를 가능하게 하는 도구를 포함하고 있다[3].

3. 서비스 기반 어플리케이션 통합

많은 조직들은 분산된 어플리케이션을 통합하려는 노력을 계속해 왔다. 이러한 노력의 대부분은 단순한 정보 단위의 전달이었고, 인터페이스와 정보 수준에서 여러 개의 정보 시스템을 동시에 개발하는 전략적인 접근 방법이었다. 그러나 SOA는 어플리케이션 통합에도 하나의 메커니즘(Mechanism)으로 자리 잡고 있다. 서비스 기반의 어플리케이션 통합은 어플리케이션을 서비스 단위로 연결한다. 그러므로 기존의 어플리케이션 통합 패턴인 정보 기반 통합, 업무 프로세스 기반의 통합, 인터페이스 처리 기반 통합, 포탈 지향 통합 등에서 서비스 단위로 연계 또는 통합이 적용되어 진다[6].

3.1 업무 프로세스 기반의 통합과 BPM

실시간 기업의 초기 개념에 가장 먼저 주목되었던 정보 기술은 업무 프로세스 기반의 통합이고, 이보다 BPM(Business Process Management)으로 더욱 많이 알려져 있다. 이 접근 방안은 변화에 민첩하게 대처하면서 기업의 가치를 극대화한다는 목적 아래에, 비즈니스 자산과 프로세스를 통합하고 최적화하는 방안으로 구축과 동시에 가시적인 성과를 제공하였다.

기업들은 현재 고객들의 새로운 요구에 대응하면서 관련된 조직간의 협업이 크게 늘어나고 있으며, 기업 자산 및 규모의 확장과 법, 제도의 변화로 인하여 시스템이 복잡해 졌으나, 변화에 빠르고 유연하게 응답하는 정보 기술을 요구하고 있다.

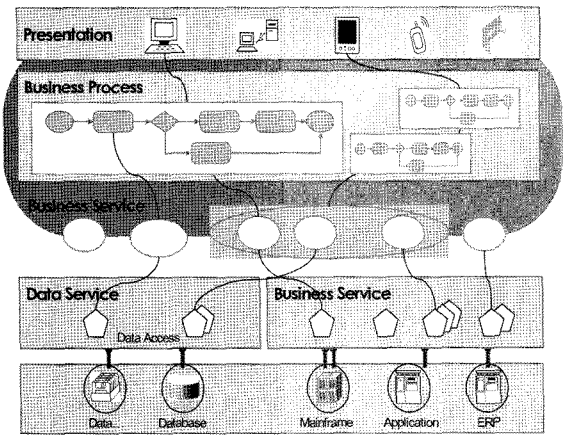


그림 2 서비스 기반 BPM

최근 BPM은 '서비스 기반 BPM'로 언급되며, 서비스 기반의 통합을 기술적 기반으로 고려되고 있다. 실제로 여러 BPM 프로젝트에서 프로세스를 가시화하는 노력뿐만 아니라 다양한 기술 플랫폼으로 구현된 레거시 시스템을 프로세스로 통합하기 위한 노력이 상당하고, 또한 많은 이슈가 발생하고 있다. 이를 해결하기 위해서 많은 어댑터를 구입하여 사용하고 EAI(Enterprise Application Integration) 또는 ESB(Enterprise Service Bus)를 도입하기도 한다. 향간에는 BPM과 ESB의 통합을 예측하고 있다.

서비스 기반 BPM은 비즈니스 업무 규칙과 데이터를 표준화된 인터페이스로 포장된 서비스로 정의하고 서비스 조합을 통해서 비즈니스 프로세스를 구성하게 된다. 또한, 기존의 서비스들을 재사용함으로써 새로운 프로세스 구성을 빠르게 한다. 이는 비즈니스 프로세스와 어플리케이션의 변경에 따른 영향을 최소화해주며, 빠른 속도와 낮은 운영비용을 가능하게 해준다는 것이다[7].

SOA 기반 프로세스 구현의 핵심 기술로 웹서비스는 우선적으로 평가받고 있으며, 프로세스 통합을 위한 표준은 워크플로우(Workflow) 기반의 XPDL(XML Process Definition Language)에서부터 BPEL4WS(Business Process Execution Lanaguage for Web Services), BPML(Business Process Markup Lanaguage) 등이 있으나 BPEL4WS이 가장 유력한 대안으로 떠오르고 있다. IBM, BEA 등 대형 벤더들이 BPM 통합 솔루션에 BPEL4WS을 적용하면서 더욱 힘이 실리고 있다[8].

3.2 정보 기반의 통합과 데이터 서비스

기업이나 공공조직은 복잡하고 다양한 업무가 혼재하고 있었고, 이를 위해 어플리케이션과 정보 시스템은 독립적으로 구현되었다. 그러나 점점 더 이들 업무는 상호 긴밀한 연관을 가지게 되었고, 서로 다른 업무간 정보의 공유와 통신이 가능해야 전체적인 업무가 진행되었다. 이러한 업무간 정보의 공유는 데이터베이스 또는 데이터를 처리하는 API(Application Program Interface)사이에서의 통합이고, 즉 정보 기반 통합 방법이라 한다.

또한 어플리케이션의 수가 증가할수록 P2P(Point-to-point) 인터페이스의 메시지 전달에 치중해서, 정보기반 통합은 데이터 무결성 및 완전성, 실시간 데이터 확보 및 접근은 어플리케이션 통합 프로젝트의 성공적인 구축에 필수적인 보완재로 인식되고 있다.

즉, 데이터 통합은 산재되어 있는 다양한 데이터 소스들과 어플리케이션으로부터 데이터를 추출하고 통합하여 전사 차원의 데이터 표준을 제공하는 것이다.

가트너에서는 데이터 통합이 전체 어플리케이션 통합의 2/3를 차지할 정도로 중요하며, 성공적인 어플리케이션 통합 전략 속에 메시지 기반의 데이터 모델링 및 데이터 통합 전략이 포함되어야 한다고 지적하고 있다[9].

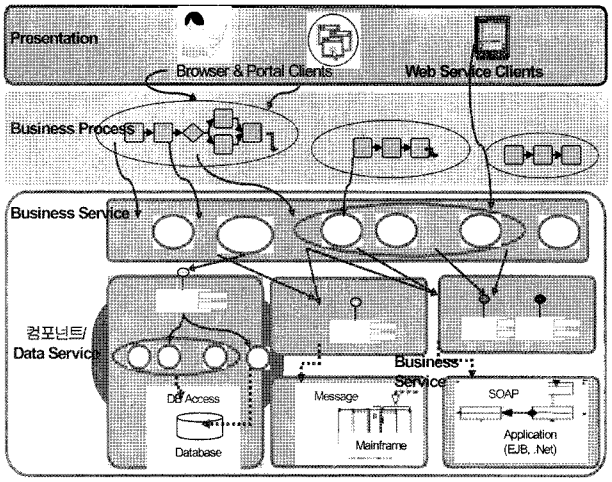


그림 3 데이터 서비스

또한 SOA에 대한 관심이 높아질수록 데이터의 역할에 대해 다시 고려되고 있다. SOA 프로젝트를 성공적으로 구현하기 위한 주요 성공요소는 데이터 무결성 및 실시간 데이터 접근이다. 이를 지원하기 위한 SOA의 개념이 데이터 서비스(Data Service)이다.

일반적으로 데이터 통합 이슈를 해결하기위해, 여러 가지 도구와 접근 방안이 적용되고 있다. 데이터 복제, 이벤트 방식의 Publish-Subscribe, 데이터 추출, 변환과 전환, 다양한 데이터 소스로부터의 연합 뷰 생성 등이 대표적인 방법이다.

데이터 서비스(SOA와 정보 기반 통합의 융합)는 SOA에서 요구하고 있는 상호운용성(Interoperability)과 의미론적 통합을 지원하기 위한 표준 기반의 접근방법이다[10].

대부분의 복합 어플리케이션(Composite Application)은 하나의 통합 계층을 필요로 한다. 이는 파일 이름, 데이터 형식, 메시지와 데이터 구조를 이동, 변환, 조정하는 것이다. 또한 데이터 서비스는 데이터 이동, 데이터 변환, 데이터 접근, 데이터 품질도 제공해야 한다.

데이터 서비스는 데이터의 물리적인 구현에서 어플리케이션을 분리한다. 그러므로 개발자는 어플리케이션에 대해 이해할 필요는 없다.

3.3 인터페이스 기반의 통합과 ESB

어플리케이션 통합은 매우 잘 정의되어 있는 어플리케이션 인터페이스를 사용하는 것으로 인터페이스 기반 통합이라 한다. 분산된 인터페이스 기반 통합에 사용되는 것이 일반적으로 미들웨어(Middleware)이다.

복잡하고 분산된 기업 IT 환경에서 서로 다른 운영체제와 어플리케이션 플랫폼들 사이의 서로 이해할 수 있는 포맷과 내용으로 표현된 비즈니스 정보를 교환하는 것뿐만 아니라 이종의 통신 프로토콜을 사용하는 네트워크 간의 접속, 네트워크 자원에 대한 접근, 그리고 시스템 연결을 위해 인터페이스 프로세싱 솔루션들이 사용된다.

지금까지 EAI(Enterprise Application Integration)와 같은 미들웨어가 이러한 역할을 하였으나 서비스 기반으로 변화하면서 웹서비스를 기반으로 하는 ESB로 변화하고 있다.

기업 전반에 서비스가 급증하게 되면 복합 어플리케이션이 개발되고 배치되면서 사용되는 P2P 연결도 관리가 어렵게 될 것입니다. 게다가 비즈니스 프로세스 정의가 어플리케이션 로직에 포함되어 비즈니스 환경의 변화에 신속하게 대응하기 어려울 것이다. 이와 함께 각 서비스에서 서로 다른 전송, 포털, 상호 작용 스타일이 사용됨으로써 발생하는 비호환성 문제까지 서비스 소비자가 해결해야 한다는 점을 고려하면 실패할 수밖에 없는 상황이 된다. 여기서 필요한 것은 개발자에게서 이런 문제를 덜어줄 수 있는 수단이다.

ESB는 "Bus"라는 단어에서도 알 수 있듯이 서비스 사용자와 서비스 제공자 사이에 위치해서 서비스의 유통을 담당한다[11]. 즉, 다양한 프로토콜을 사용하는 통합과 정보 교환이 필요한 이기종 기업 어플리케이션들 사이에서 표준화된 중개자로서 통합의 백본 역할을 한다[12].

서비스 제공자와 서비스 사용자는 직접 연결되지 않고 ESB를 통하여 연결되어 사용자가 제공자의 정확한 위치를 알지 못해도, 사용자가 요구하는 서비스를 연결하고 메시지 변환을 통한 통역자의 역할도 수행하며 송수신 메시지를 안전하게 전달하는 역할을 한다.

ESB는 명시적인 서비스 호출 뿐만 아니라 특정 이벤트를 통해서도 실행되어 이벤트 중심 방식(Event Driven Architecture)과 서비스 지향 아키텍처를 결합하여 비즈니스 단위들의 통합을 수월하게 한다[10].

대부분의 ESB 솔루션들은 표준 인터페이스 기술로 웹서비스를 사용하고 있으며, 이외 JMS와 같은 표준 기반의 통신 인프라, XSLT(EXtensible Stylesheet Language Transformations)나 XQuery와 같은 표준 기반의 문서 변환, LDAP(Lightweight Directory Access Protocol)와 SSL(secure sockets layer)을 이용한 표준 기반의 보안 등을 구현하고 있다[13].

ESB는 표준을 따르지 않는 기 구축된 어플리케이션을 표준 기반 인터페이스로 전환하고, 메시지의 라우

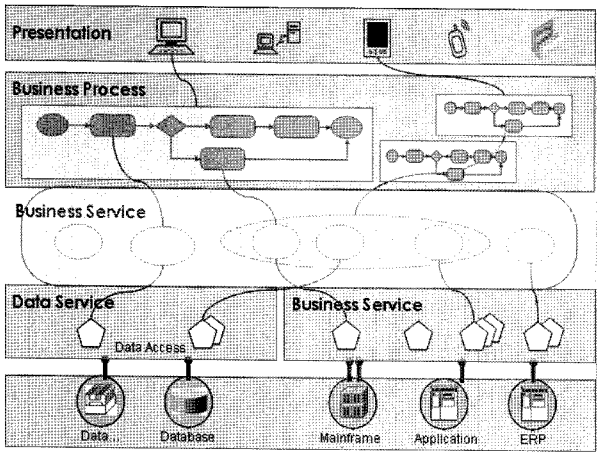


그림 4 ESB

팅(Routing), 보안, 품질, 비즈니스 규칙(Rule) 등의 서비스 연결에 필요한 서비스 명세 정보를 어플리케이션과 분리하여 코드의 수정 없이도 간편하게 서비스 연결을 변경할 수 있게 함으로써 변화에 유연하게 대응한다.

4. MDI

기업의 많은 시스템은 고객, 제품 등 마스터 데이터를 사용하고 있다. 이는 종종 서로 다른 데이터베이스에 나뉘어 저장되어 있거나, 서로 다른 버전의 일관되지 않은 데이터 값으로 저장되어 있다.

마스터 데이터에 대한 단일 뷰가 없다면 고객이나 제품과 관련된 가치나 고객 이탈 등의 위험을 판단하기가 매우 어려울 것이다. 또한 마스터 데이터 통합이 부실하면 CRM(Customer Relationship Management), ERP(Enterprise Resource Planning), SCM(Supply Chain Management) 등의 기업용 어플리케이션 프로젝트를 성공적으로 진행하는데 방해가 되거나 원하는 ROI를 낼 수가 없다. IDC에 따르면 글로벌 기업의 어플리케이션들은 다중 인스턴스를 가지고 있기 때문에 마스터 데이터에 대한 단일 접근 포인트를 필요로 하고 있다. MDI을 활용하면 데이터 품질이나 중복 문제를 줄일 수 있고 공급자 협업, 군살없는 생산 방식(lean manufacturing), e-Commerce도 지원 가능하다[14].

4.1 마스터 데이타

마스터 데이터는 기업의 비즈니스를 고유하게 정의하기 위해 필요한 주요 데이터이다. 각 비즈니스 객체는 비즈니스 프로세스에서 공통으로 사용되는 일관되고 일정한 아이디와 확장 속성들을 가지고 있다. 마스터 데이터에는 관계자(고객, 잠재 고객, 주민, 시민, 직원, 벤더, 공급자, 거래 상대자), 장소(위치, 사무실, 지역, 지리), 사물(계정, 자산, 정책, 제품, 서비스) 등이 포함된다.

4.2 MDI

MDI는 기업이 데이터 공유라는 오랜 목표에 한발짝 접근할 수 있게 하였다. 조직이 전사 수준에서 마스터 데이터를 사용할 때 부서마다 또는 분리해서 저장하고 있던 다양한 버전의 데이터들이 줄어들게 된다. 이렇게 해서 마스터 데이터 프로그램은 부서간 장벽을 허물고 기업의 유연성과 통합 노력을 단순하게 해준다. 마스터 데이터가 기록 시스템으로 점차 성숙해지면서 시스템간 복잡한 인터페이스들이 점차 해소될 수 있다.

MDI는 마스터 데이터에 대한 단일 뷰를 저장, 유지, 업데이트해주고 여러 시스템이 트랜잭션을 처리할 때 마스터 데이터를 참조할 수 있도록 해준다.

MDI는 전사가 공유해야 하는 공통 정보 자산을 일치시키고 정리하고 보고하기 위해 비즈니스와 IT 인력이 워크플로우(workflow) 기반으로 상호 협업해야 하는 프로세스를 요구하고 있다. 또한 비즈니스에서도 마스터 데이터의 생성, 변경, 삭제 등을 위해 검토, 승인, 표준 개발, 롤업, 데이터 품질 관리, 변경 관리를 포함하는 사무장 기능(stewardship)을 지원해야 한다. 이를 통해 누가 가지고 있는 데이터가 맞는건지, 누가 정의한 판매 정의를 사용할 것인지 등에 대한 지루하고 끝없는 논쟁을 종식시킬 수 있을 것이다.

5. 사례 연구

SOA와 MDI가 산업 전 분야에 적용되고 있지만, 관심이 집중되고 있는 통신 산업과 금융 산업에서의 적용 사례를 분석해 구체적인 적용 유형을 제안한다.

90년대 초에 활성화된 무선통신 서비스의 폭발적인 수요에 따라 2000년에는 무선통신 가입자의 수가 유선 통신 가입자의 수와 대등한 수준까지 성장하였다. 이와 동시에 유무선 사업자들 간의 경쟁이 심화되고 있으며 이로 인해 각 사업자들은 보유하고 있는 가입자의 수를 통하여서는 수익성을 보장 받을 수 없는 상황에 직면하고 있다. 한편 통신 규제의 완화 경향에 맞추어 유선 통신 사업자와 무선 통신 사업자간의 장벽이 없어짐에 따라 유무선 사업자들의 사업 전략도 다른 서비스 영역으로의 확장을 통해 경쟁력을 강화하고 수익 증대를 꾀하는 방향으로 변화하고 있다.

이러한 상황에서 고객 서비스 강화와 복합 서비스를 위한 차세대 고객 및 빌링 시스템을 개발하고 있다. 고객 및 빌링 시스템에서 중요한 것은 '고객 및 상품에 대한 어플리케이션 통합 모델'이고, 외국 통신 사례

에서도 '고객에 대한 통합'이 우선적으로 진행되었다.

뉴질랜드 텔레콤은 P2P 인터페이스로 인해, 다른 고객 데이터에서 고객 정보를 이용하고 있어서, 통합 시 고비용이 발생하기 때문에, 고객 중심의 뷰(View)를 구축하고자 하였다. 즉 고객 접점의 여러채널에서 사용하는 표준화된 고객 및 관계 데이터를 생성하고 제공하는 프로젝트를 진행하였다. 이로 인하여 빈번하게 발생하는 통합 요구사항에 적은 비용으로 대처하게 되었다[15].

MetLife는 개인과 법인 고객에게 보험과 기타 금융 서비스를 제공하는 선두 업체로서 1,300만 세대의 개인과 3,700만 직장인들에게 연금을 제공하고 2백만 명의 자동차와 주택소유자 보험 고객을 가지고 있다. 2004년말 기준으로 199억달러의 수입과 3,860억달러의 자산을 보유하고 있으며 LOB(Line of Business)가 매우 많은 특징을 가지고 있다.

MetLife는 거래 당사자인 고객에 대한 기록 시스템을 만들고, 상품 중심 회사에서 고객 중심 회사로 변신을 꾀하고 있었다. 고객 중심 금융 서비스 회사가 되기 위해 중앙화된 고객 정보가 필요했으며 이에 대한 대안으로 고객 접점 솔루션, 데이터웨어하우징 솔루션, 운영 레코드 시스템 등을 고려하였다. 그 결과 모든 상품과 서비스에 대한 모든 관계자 정보 레코드를 갖는 시스템, 관계자 중심 비즈니스 어플리케이션을 가능케 하는 새로운 솔루션 구현을 결정했다.

MetLife는 이를 통해서 고객에 대한 지식을 증가시키고 관계의 가치를 이해할 수 있으며 교차 판매의 기회를 높여 차별화된 서비스를 제공할 수 있었다. 또한 합리화된 비즈니스 프로세스로 효율을 증가시키고 신상품을 차별화하거나 개인화할 수 있었다. 현재는 IBM에 인수된 DWL Customer Center 제품을 기반으로 새로운 시스템을 만들었고 이러한 접근법은 고객 접점 어플리케이션 통합 같은 다른 대안들보다 비용이 높고 통합을 위한 더 큰 노력이 필요했으나 통합의 결과로 더 정확하고 유지관리가 용이한 시스템을 가질 수 있었다[16].

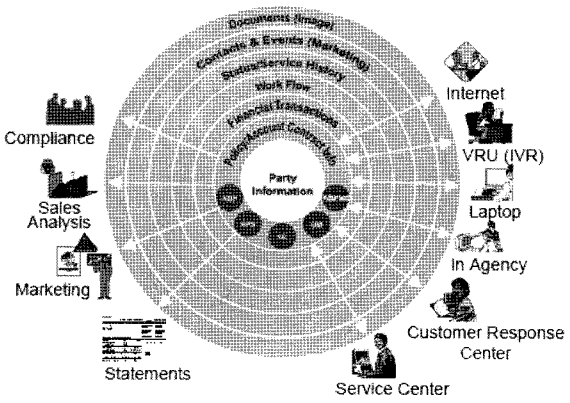


그림 5 MetLife의 MDI 구현 사례

6. 결 론

SOA의 구현은 궁극적으로 기업 전반의 어플리케이션에 영향을 미치고, 기업의 정보 기술 부서에도 큰 변화를 줄 것으로 예측되고 있다. 그러나 초기의 SOA는 주로 BPM, 인터페이스 기반 통합 및 정보 공동 이용의 모습으로 나타나고 있다.

사례 연구에서 살펴본 바와 같이, 기업이나 공공기관은 조직 전체 차원에서의 데이터 및 정보를 실시간(Real-time)으로 접근하기를 원하고 있다. 이로 인하여 데이터와 인터페이스 기반 통합 프로젝트가 주를 이루고 있다. 정보 기반 통합에서 하나 주목받고 있는 것이 MDI이다.

참고문헌

[1] 김국태, "애플 부활을 통해 본 혁신 키워드", LG주간경제, Aug. 2005

[2] Service Architecture, LG CNS Technology Inside, pp.21-27, June. 2006

[3] The Advent of a New Service Infrastructure BEA, http://www.bea.com/content/news_events/white_papers/BEA_new_category_wp.pdf

[4] Christine Hofmeister, Robert Nord, Dilip Soni, Applied Software Architecture, Addison-Wesley Professional, 1999

[5] 조재훈, 이상환, 서비스 지향 아키텍처의 기반기술과 구축사례에 대한 연구, 한국정보시스템학회 2005년 추계학술대회 발표 논문집, pp.455-462, 2005

[6] David Linthicum, Next Generation Application Integration, Addison-Wesley, 2004

[7] Introduction to SOA based BPM, LG CNS 연구개발센터, pp2, June. 2006

[8] Business Process Management, LG CNS Technology Inside, pp.210-211, Jan, 2005

[9] David Newman, Ted Friedman, "Data Integration Is Key to Successful Service-Oriented Architecture Implementations", Gartner Research, Oct. 2005

[10] SOA Reference Architecture Model, LG CNS 구개발센터, pp.7-9, June, 2006

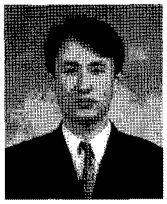
[11] Introduction to SOA Technical Reference Model, LG CNS 연구개발센터, pp.7, pp.41, June, 2006

[12] Mike Gilpin, "What is an Enterprise Service Bus",

Forrester Research, Aug. 2004
[13] SOA 기술 분석 ESB, LG CNS Technology Inside, pp.106-107, Jan, 2005
[14] Rasmus Andsbjerg, 'Telenor Case Study: How to Master Data Management', IDC, 2006.05
[15] Oracle Datahub 사례, Oracle Korea, 2005
[16] John Radcliffe, "Creating the Single Customer View with Customer Data Integration", Gartner, 2006

장 양 자

1993 서울대학교 지구과학교육학과 학사
1996 서울산업대학교 정보산업공학과 석사
2002 서울대학교 산업공학과 박사
2002~2005 ETRI 우정기술연구센터
2006~현재 LG CNS 정보기술연구소
관심분야 : SOA, SaaS
E-mail : yjjang@lgcns.com



김 상 수

1991 서강대학교 전자계산학과 학사
1993 서강대학교 전자계산학과 석사
1996~현재 LG CNS 정보기술연구소
관심분야 : MDI, SOA, EA
E-mail : sookim@lgcns.com

조 수 형

1995 동국대학교 대학원 경영정보학과 석사
1998~현재 LG CNS 사업이행본부 금융부문 전문위원
관심분야 : EA, BPM, 금융산업 규제/제도
E-mail : soohcho@lgcns.com