복잡한 엔터프라이즈 응용 개발을 위한 ISIS 아키텍처

(ISIS Architecture for Developing Complex Enterprise Applications)

조 은 환 † (Eun Hwan Jo) 이 갑 훈 † (Kap Hoon Lee) 이 민 수 † (Min Soo Lee) 이 봉 † (Bong Lee)

요 약 최근 비즈니스 프로세스와 IT 시스템은 점점 더 복잡해져 가고 있다. 특히 엔터프라이즈 어플리케이션은 복잡도를 제어하기가 힘들어지면서 관리비용도 계속 증가해가는 추세다. 따라서 복잡도는 소프트웨어 개발에 있어서 방심해선 안될 중요한 문제가 되었으며, 이와 같은 문제를 효과적으로 해결할 수 있는 방법이 절실히 필요한 실정이다. 본 논문에서는 엔터프라이즈 어플리케이션 개발 복잡도 문제를 해결하기 위한 ISIS(Integrated System of Independent Subsystems) 아키텍처를 제안하고자 한다. ISIS는 대규모 엔터프라이즈 어플리케이션의 복잡도를 줄이고자 하는 노력에서 연구개발 되었으며, 시스템 개발의 복잡도를 줄이고 컴포지트 어플리케이션 개발이 가능한 아키텍처 모델이다. 엔터프라이즈 어플리케이션은 상호연관성 및 ISIS 분해방법에 따라 독립적인 서브시스템(sub-system)으로 나뉘게 된다. 그리고 이 기종 분산 플랫폼에 위치한 각 서브시스템의 상호연동을 위해서 ISIS 지원 미들웨어를 사용한다. 본 논문에서는 이와 같은 ISIS 기술을 검증하고자 ITSM(IT Service Management) 시스템에 ISIS 아키텍처를 적용 및 구현하였다. 결론적으로 ISIS 아키텍처는 개발 복잡도를 줄임으로써 비즈니스 요건이 변경되거나 기존 시스템을 업그레이드 할 경우 구조유연성 및 개발생산성을 향상시킬 수 있다.

키워드 : ISIS 아키텍처, 복잡도 감소, 엔터프라이즈 어플리케이션, 컴포지트 어플리케이션, SOA, 미들웨어

Abstract Recently, as both business processes and IT systems become ever more complex. Especially, enterprise applications tend to become unmanageably complex and increasingly costly to maintain. Therefore complexity is the insidious enemy of software development. It is critical to have a methodology that recognizes and manages this enemy effectively. In this paper, we propose ISIS (Integrated System of Independent Subsystems) - the architectural style needed to develop the complex enterprise applications. The ISIS was developed to meet the challenge of reducing the complexity of a larger enterprise application today. It gives us architecture models for reducing development complexity and composite application. The enterprise application is partitioned into a collection of independent subsystems using ISIS decomposition schemes and equivalence relations. We use middleware named ISIS engine that provides a service for subsystems interoperability by enabling the integration of distributed, cross-platform subsystems. We have implemented an ITSM system that achieves our objectives, reducing development complexity, using the ISIS architecture. Finally, ISIS architecture provides greater flexibility and productivity when an organization needs either to change its business processes, or to update the underlying systems.

Key words : ISIS architecture, complexity reducing, enterprise application, composite application, SOA, middleware

† 정 회 원 : 동부CNI
joeh@dongbu.com
leekaphoon@dongbu.com
mslee422@dongbu.com
bonglee@dongbu.com
논문접수 : 2009년 11월 23일
심사완료 : 2010년 1월 3일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제4호(2010.4)

1. 서 론

오늘날의 엔터프라이즈 어플리케이션은 계속 증가하는 복잡도와 새로운 프로세스의 추가로 인해서 개발 및 유지보수에 대한 비용과 위험요소가 증가하고 있다[1, 2]. 또한 최근 엔터프라이즈 IT환경은 변화와 속도라는 도전에 직면하게 되면서 호환성 없는 시스템과 아키텍처로 인해 비즈니스 프로세스가 경직되고, 그 결과 시장에 민첩하게 반응할 수 없는 상황이 되고 있다. 또한 IT 부서들은 효율성과 유연성을 높여서 급변하는 비즈니스 요구에 대처할 수 있고 IT 복잡성을 극복할 수 있는 혁신적 방법을 필요로 하게 되었다. 이와 관련하여 최신 어플리케이션 개발 패러다임과 아키텍처들은 기존의 객체(object) 및 컴포넌트(component) 기술에서 SOA (Service Oriented Architecture)처럼 프로세스를 서비스로 모듈화하는 방식으로 진화하고 있다. 따라서, 본 논문에서는 엔터프라이즈 어플리케이션 개발이 가지고 있는 복잡성 문제를 해결하기 위한 방안으로 ISIS 개발 아키텍처를 제안하고, 개발사례를 소개한다. 본 논문의 내용을 간단히 살펴보면 다음과 같다. 우선 2장에서는 관련연구에 대해서 살펴보고 3장에서는 ISIS 개념 및 아키텍처를 설명한다. 그리고 4장에서는 ISIS 개발을 지원하기 위한 미들웨어에 대해서 기술하고, 5장에서는 ISIS 기반의 PMS(Project Management System)와 ITSM 시스템 개발결과에 대해서 설명한다. 마지막으로 6장에서는 본 논문의 결론과 향후 과제를 제시한다.

2. 관련연구

2.1 소프트웨어 시스템 복잡도

지난 수십 년간 많은 기업의 IT 조직은 발전된 기술과 많은 리소스를 투입하여 효율적인 업무지원을 위한 시스템을 개발 및 구축해왔다. 그러나 필요에 따라 개별적으로 개발된 레거시(legacy) 시스템들은 IT 아키텍처를 점점 더 복잡하게 만들었고 유지보수 및 통합에 굉장히 많은 어려움과 비용지출을 초래하게 되었다. 특히, 그림 1에서와 같이 엔터프라이즈 어플리케이션이 점점 더 대형화되고 복잡해지면서 빠르게 변하는 시장 요구사항을 따라가지 못하고 있는 실정이다. 그래서 개발 복잡성을 줄이고 비즈니스 지원 속도를 가속화하기 위한 IT 개발기술이 필요하게 되었다.

시스템 복잡도가 매우 중요한 문제로 인식되고 있지만 "complexity"란 용어 정의가 명확하지 않고 표준적 정의 및 측정방법이 구체적이지 못한 실정이기 때문에 다양한 형태의 복잡도가 존재하며, 대상에 따라서 서로 다른 의미를 가지기도 한다.

본 관련연구에서는 분산 시스템의 복잡도를 task-structure complexity, unpredictability, size complexity, chaotic complexity 그리고 algorithmic complexity 등 5가지로 구분하고 있다. 각각에 대한 설명은 다음과 같다[4,5].

- Task-Structure Complexity: 소프트웨어 구조적인 복잡도로서 분산 시스템에서 하나의 태스크를 수행하거나 이해하는 정도에 대한 복잡도이다.
- Unpredictability: 분산 시스템에서 어떤 행위에 대한 영향을 예상할 수 있는 정도의 복잡도로서 시스템 내의 엔트로피의 양이 중요한 측정 요소 중 하나이다.
- Size Complexity: 어떤 코드의 라인 수(Line of code)와 같은 분산 시스템의 크기의 복잡도이다.
- Chaotic Complexity: 분산 시스템의 서로 다른 구성요소들간의 의존성 또는 영향도에 대한 복잡도로써 컴포넌트 간의 coupling 정도를 의미하며, data coupling, control coupling, common coupling 그리고 content coupling 등으로 구분할 수 있다.
- Algorithmic Complexity: 복잡도에 대한 전통적인 정의로써 분산 시스템이 처리하는 알고리즘의 이해 정도에 따른 복잡도다.

본 논문에서 제안하는 ISIS는 5개의 복잡도 중에서도 task-structure, size complexity 그리고 chaotic complexity와 관련된 아키텍처다.

최근 소프트웨어 개발 복잡도 감소와 관련된 연구사

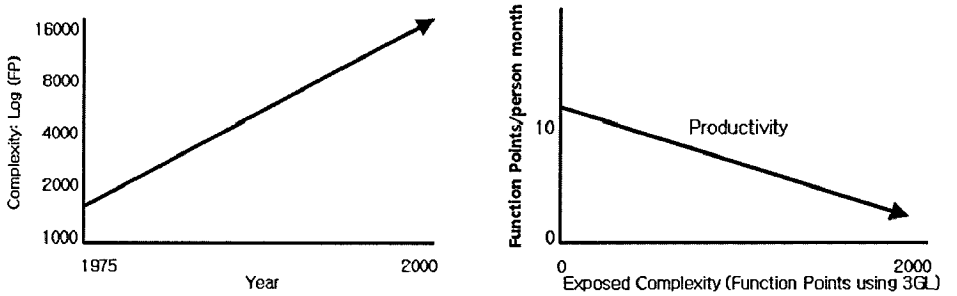

그림 1 소프트웨어 복잡도[3]

례 중 SIP(Simple Iterative Partitions)은 수학적 모델에 기반한 엔터프라이즈 아키텍처 방법론이다. SIP은 일반 방법론과는 달리 엔터프라이즈 아키텍처의 복잡도 제어에 초점을 맞추고 있다. SIP의 접근방법은 크게 두 단계로 나뉘는데, 우선 복잡도를 줄이기 위해서 대상을 분해한다. 그리고 다음으로 단순화 알고리즘을 사용하여 비즈니스 요구사항과 IT 기능 사이에 매핑 관계를 정의한다. 이때 비즈니스 요구사항에 매핑되지 않는 IT 기능들은 제거되고 중복되는 부분은 통합된다. 결국 이와 같은 두 단계를 엔터프라이즈 어플리케이션 개발에 반복적으로 적용함으로써 엔터프라이즈 아키텍처의 복잡도를 줄이고자 하는 것이 SIP의 목적이다[2].

2.2 컴포지트 어플리케이션 개발기술

최근 엔터프라이즈 IT 환경은 사용자들의 다양한 요구사항 증가로 인해서 제품인도소요시간(delivery cycle time)이 빨라지고, 개별 구축된 이질적인 시스템들로 인해 인프라스트럭처가 점점 더 복잡해지고 있다. 그리고 이에 따른 통합노력 증가, 변화관리 어려움, 그리고 신규 시스템 규모 및 복잡도 증가로 인해 전체 개발기간과 TCO(Total Cost of Ownership)가 증가하고 있다. 따라서, 이와 같은 문제점들을 해결하고자 컴포지트 어플리케이션(composite application) 개발기술들이 제안되고 있다[6]. 특히, 최근 컴포지트 어플리케이션 개발기술 중에서 가장 각광을 받고 있는 것이 SOA이다[7].

1996년 Gartner의 "Service Oriented Architectures, Part1" 보고서에 의하면 SOA는 표준 인터페이스의 개념으로 전체 어플리케이션을 구축하는 소프트웨어 아키텍처라는 개념[8,9]으로 최초로 등장하였다.

SOA는 비즈니스 프로세스 및 이를 지원하는 IT 인프라를 표준화시켜서 변화하는 비즈니스의 우선순위에 따라 재사용 및 결합 가능한 컴포넌트(서비스)로 통합하는 프레임워크로 정의할 수 있다. 여기서 서비스(service)란 반복사용 가능한 비즈니스 태스크이며, 서비스 지향성(service orientation)이란 비즈니스 프로세스를 연결된 서비스와 그 결과물로 통합하는 방법을 의미한다. 그리고 서비스 지향 아키텍처(Service Oriented Architecture)란 서비스 지향을 지원하는 IT 아키텍처 스타일이다[10]. 그림 2에서처럼 SOA 구조 위에 구축된 비즈니스 프로세스를 지원하는 통합된 서비스들의 집합을 컴포지트 어플리케이션이라 한다.

즉, SOA는 IT 분야에서 이 기종환경에서 시스템의 상호 운용성을 해결하기 위한 방법으로서 컴포넌트 재사용, 안정성 향상, 개발 및 운용비용 절감과 같은 기대효과를 가지고 있다.

SOA는 컴포넌트 기반 아키텍처 또는 객체지향 아키텍처와 같은 이전 아키텍처 프레임워크를 대체하는 것

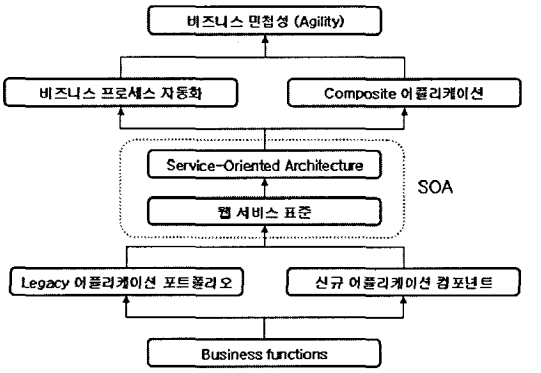

그림 2 SOA와 컴포지트 어플리케이션 [Source: IDC, 2003 Dec]

이 아니라 기술적 구성요소를 비즈니스 프로세스와 결합시키기 위한 또 하나의 아키텍처 계층을 추가한 것이다.

SOA의 서비스 특징들은 다음과 같다.

- 플랫폼 또는 구현 기술에 대해 독립적이고, 손쉽게 사용, 재사용할 수 있고 동적 상호 운용성을 보장하는 표준 기반의 동일한 인터페이스 사용
- 서비스가 어떻게 사용되는지 또는 누가 사용하는지에 대해 사전에 정해진 개념 없이 그리고 서비스 구현 변경 사항이 사용자에게 영향을 미치는 것에 대한 우려 없이 서비스가 생성될 수 있는 느슨한 결합
- 각 서비스를 비즈니스 어플리케이션 기능의 개별 단위로 나타내는 모듈성

SOA는 어떤 기술이나 제품이 아니라 아키텍처 방식 중 하나로서 SOA를 구현하기 위해서는 ESB(Enterprise Service Bus)와 같은 인프라 지원이 필요하다. ESB는 엔터프라이즈 어플리케이션 및 서비스 통합을 위한 유연한 연결 인터페이스를 제공하는 미들웨어로서 다음과 같은 기능들을 제공한다.

- 서비스 간에 메시지를 전달(route)
- 요청자와 서비스 사이에서 전송 프로토콜을 변환(convert)
- 요청자와 서비스 사이에서 메시지 형식을 변환(transform)
- 출처가 각기 다른 비즈니스 이벤트를 처리(handle)

이와 같은 기능들을 통해서 SOA는 서비스 및 프로세스 최적화를 통한 어플리케이션 차원의 유연성 확보와 소프트웨어 재활용을 통한 비용절감 및 구축시간을 단축하고자 한다. 다음 장에서 설명할 ISIS역시 SOA와 마찬가지로 엔터프라이즈 어플리케이션의 개발 용이성 및 생산성을 높이고자 한다.

3. ISIS 개발 아키텍처

3.1 ISIS 개념

엔터프라이즈 어플리케이션들은 그림 3에서처럼 전통

표 1 CBD, SOA 그리고 ISIS 비교[12,13]

구분	CBD	SOA	ISIS
목표	특정 시스템 관점에서의 컴포넌트 구축 및 재사용	내×외부 공통 비즈니스 서비스 구축 통한 재사용	독자적 서브시스템 구축 통한 개발복잡도 감소
조합 단위	컴포넌트	서비스	서브시스템
모듈 크기	비교적 작음 - 추상화 정도가 낮음	업무에 따라 다름 - 추상화 정도가 높음	업무에 따라 다름 - 추상화 정도가 높음
모듈 특성	기능중심 - 의존적/부분적 기능수행	비즈니스 중심 - 독립적 기능 수행	비즈니스 중심 - 독립적 업무 수행 - 개별 DB 가짐
표준화	독자표준	Open Standard	Open Standard
연결성	Tight coupling	Loosely coupling	Loosely coupling
인터페이스	개별 인터페이스방식 활용 - 특정 플랫폼 기반	미들웨어 기반 - Two-way 메시징 - 이기종 간 통합 연계	미들웨어 기반 - One-way 메시징 - 이 기종 간 통합 연계
재사용성	컴포넌트 수준	서비스 수준 - 상대적 재사용성 높음	서브시스템 수준 - 기존 레거시 재사용성 높음
구현	컴포넌트 별 분리 구현	각 서비스 독립적 구현	각 서브시스템 독립적 구현
활용기술	WAS 기반 Web 서비스 기술	ESB, Web 서비스 기술	ESB, Web 서비스 기술, 메시지 큐

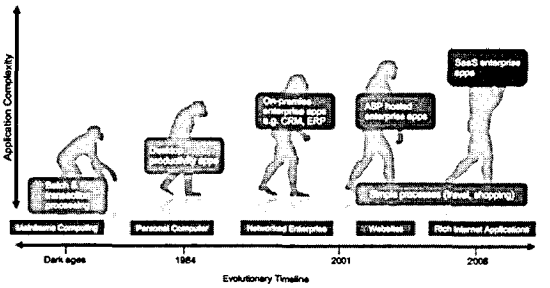

그림 3 어플리케이션 복잡도[11]

적인 데스크톱 어플리케이션 기능과 특징을 웹 어플리케이션으로 구현한 리치 인터넷 어플리케이션(Rich Internet Application: RIA) 형태로 진화하고 있다. 이와 같은 엔터프라이즈 어플리케이션들은 이질적인 인프라 환경에서 다양한 디바이스를 인터페이스로 하는 대규모 분산 시스템이기 때문에 복잡도와 크기가 상당히 높아져 가고 있다.

따라서 개발자들은 이러한 시스템을 위한 서비스나 프로그램을 개발하기가 어렵고, 관리자들은 유지보수가 힘들다. 또한 기존 레거시 시스템에 새로운 비즈니스 요구사항이 생기면 이에 대한 민첩한 대응이 어렵거나 많은 비용을 들여서 신규로 시스템을 재개발하는 경우가 많다.

따라서 본 논문에서는 이와 같은 문제를 개선하기 위해서 엔터프라이즈 시스템 개발 아키텍처 스타일인 ISIS를 제안하고자 한다. ISIS란 복잡도 높은 엔터프라이즈 어플리케이션을 효과적으로 개발 및 운영하기 위해서 연구개발된 아키텍처 모델로서, ISIS의 기본사상은 엔터프라이즈 시스템이 필요로 하는 기능을 점차적으로 하나씩(piecemeal) 개발함으로써 시스템의 전체적인 생명주기(life cycle)를 연장하는 것과 신규 엔터프라이즈 시스템을 개발할 때 복잡도를 줄임으로써 개발 용이성 및 개발 생산성을 향상시키는데 있다.

이와 같은 ISIS는 다음과 같이 정의될 수 있다.

"ISIS(Integrated System of Independent Sub-systems)란 표준 인터페이스 기술과 메시징 방식을 사용하여 복잡도 높은 엔터프라이즈 어플리케이션을 소규모 비즈니스 트랜잭션 단위인 서브시스템으로 분해하고 이를 독립적이고 점진적으로 구현 및 결합하는 방식의 IT Architecture style이자 개발철학"

다시 말해서, ISIS란 대규모 엔터프라이즈 어플리케이션을 소규모 업무시스템 단위인 서브시스템으로 모듈화하여 개발복잡도를 줄이고, 각각의 서브시스템들을 큐(queue) 기반의 메시징(messaging)과 표준 지향적인 인터페이스 기술을 사용하여 점차적으로 유연한 구조의 통합 어플리케이션을 완성할 수 있도록 지원하는 개발 아키텍처이다. 이때, ISIS의 서브시스템이란 하나의 비즈니스 프로세스를 독립적으로 처리 가능한 정도의 소규모 업무 시스템이다.

이와 같은 ISIS의 주요 개념은 다음 두 가지로 요약할 수 있다.

- Sub-system Oriented: 비즈니스 프로세스를 서브시스템 단위로 개발 및 점차적으로 하나씩 통합하는 개발방식
- Independent Development: 자체 DB를 가진 독립 시스템 성격의 서브시스템을 개별적으로 설계 및 구현하는 방식

우선 서브시스템 중심적인 개발특성은 엔터프라이즈 어플리케이션의 복잡도를 줄이기 위해서 대규모의 시스템을 분해하여 내부적으로 처리되는 알고리즘을 단순화

시키는 것이다. 최근 컴포지트 어플리케이션 개발기술 및 패러다임은 대부분이 이와 같은 분할과 정복(divide & conquer) 원리를 기본으로 따르고 있다. 다음으로 독자적 개발특성은 ISIS의 가장 큰 특성으로서 개발자들은 전체 시스템을 모두 이해할 필요 없이 해당 서브시스템을 개별적으로 개발 및 통합할 수 있게 된다. 이를 위해서 각 서브시스템은 개별적인 데이터베이스를 가지고 있어야 한다. 이에 대해서는 다음 절에서 자세히 다루기로 한다.

컴포지트 어플리케이션 개발의 대표 기술인 CBD 및 SOA와 ISIS를 비교한 결과는 표 1과 같다.

3.2 ISIS 어플리케이션 모델

ISIS 어플리케이션 모델은 서브시스템 중심의 분산 컴포지트 어플리케이션으로서, SOA와 마찬가지로 비즈니스 프로세스를 모듈화 해 놓은 서브시스템을 조합하는 형태로 개발된다. 특히 복잡도가 높은 엔터프라이즈 어플리케이션을 개발할 때 전체 시스템에 대한 이해 없이도 각 서브시스템을 독립적으로 설계 및 구현 가능한 구조를 가진다.

ISIS 어플리케이션은 그림 4에서처럼 presentation layer, process layer, sub-system layer 그리고 component layer로 구성된다. Presentation layer는 통일된 사용자 인터페이스를 제공하게 되며, process layer는 해당 업무 시스템이 처리해야 할 메가 비즈니스 프로세스에 대한 뷰를 제공한다. 그리고 sub-system layer는 ISIS 어플리케이션의 핵심 구성요소로서, 메가 비즈니스 프로세스를 실제로 구성하는 서브 구현 모듈로서 하위의 component layer에 위치한 관련 컴포넌트를 사용하여 구현이 가능하다. 이때 각 서브시스템들은 ISIS Support Middleware를 통해서 상위 process layer에 서비스를 제공하게 된다. 즉, 관련 서브시스템은 ISIS Support Middleware를 통해서 하나의 엔터프라이즈 어플리케이션으로 조합된다.

다시 말해서, ISIS 어플리케이션은 한 개 이상의 응집력 높은 서브시스템들을 독자적으로 개발하고 ISIS Support Middleware가 제공하는 표준 인터페이스와 메시징 방식을 통해서 연동함으로써 하나의 엔터프라이즈 어플리케이션을 구성하기 때문에 개발용이성이 향상되고, 유연성 있는 어플리케이션 구조를 갖추게 된다.

ISIS 어플리케이션 모델이 기존 어플리케이션 구조와 다른 점은 기존 전통적인 S/W개발에서는 객체 또는 컴포넌트 모듈로 구성된 어플리케이션이 하나의 통합 DB를 사용하여 개발된 반면, ISIS에서는 개별 DB를 가진 서브시스템들이 ISIS 지원 미들웨어를 통해 상호연동되는 분산시스템 구조로 개발된다는 점이다. 이에 대해서는 ISIS 서브시스템 모델에서 자세히 다루기로 한다.

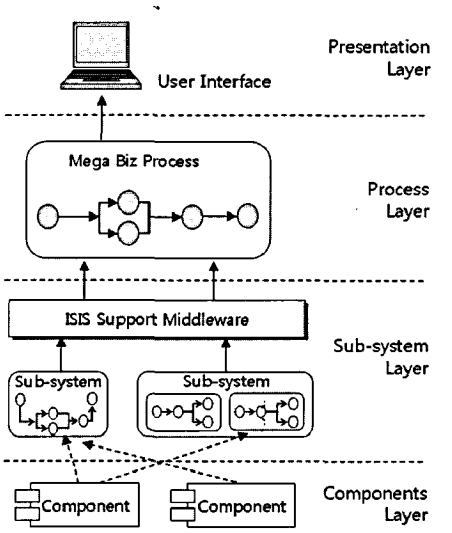

그림 4 ISIS 어플리케이션 모델

그리고 ISIS Support Middleware는 본 논문의 4장에서 설명한다.

3.3 ISIS 서브시스템 모델

ISIS에서 서브시스템이란 ISIS 어플리케이션을 구성하는 기본구성 요소로서 독립적인 비즈니스 처리 및 개발단위이다. 즉, 하나 이상의 비즈니스 프로세스를 자체적으로 처리할 수 있는 소규모의 독립된 시스템으로서 전체 시스템 설계를 최소화 하면서 서브시스템 별로 독자적인 설계, 구현 그리고 실행이 가능하다. 이를 위해서 서브시스템은 자신이 처리하는 비즈니스 프로세스의 관련 데이터를 모두 로컬 DB에 가지고 있어야만 한다. 그리고 이질적인 분산 환경에 위치한 각 서브시스템들이 상호 연동되기 위해서는 표준 인터페이스를 제공해야 한다. 그 밖에 기존의 레거시 시스템도 논리적으로 하나의 서브시스템이 되어 타 시스템과 연동될 수 있다.

그림 5는 이와 같은 ISIS 서브시스템의 메타 모델을 보여주고 있다.

ISIS 서브시스템 모델에 의하면 서브시스템은 기본적으로 처리해야 할 비즈니스 트랜잭션을 포함하고, 해당 비즈니스 트랜잭션은 다수의 객체 컴포넌트로 구성된다. 그리고 서브시스템은 sender와 receiver에 연결된 Message EndPoint를 통해서 타 서브시스템들에게 메시지를 송수신한다. 특히, ISIS의 서브시스템은 자신만의 데이터베이스를 가지고 있어야 한다. 이처럼 설계된 각 서브시스템의 설계 명세에서는 서브시스템의 비즈니스 로직 명세, 시스템의 구조 명세 및 행위 명세, 그리고 처리 데이터를 명세하게 된다. 그리고 서브시스템에서 처리하는 일련의 비 동기전송 메시지를 함께 명세한다. 이와 같은 설계요소는 독립된 서브시스템 구현자 사이의

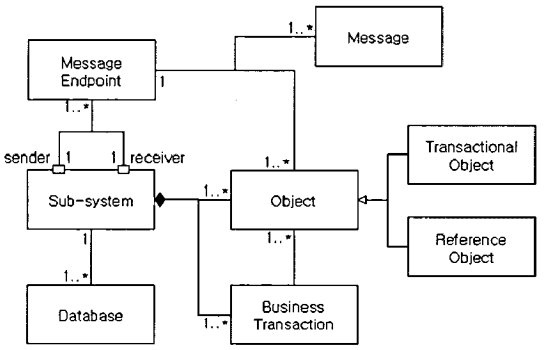

그림 5 ISIS 서브시스템 모델

계약으로서의 의미를 갖는다. 즉, 서브시스템 개발자는 타 서브시스템과 어떠한 상호작용을 해야 할지와 필요한 데이터를 어디서 공급받아야 할지에 대한 최소한의 정보를 제공하게 된다.

ISIS의 주요 개발 목표인 복잡도 감소 및 개발 생산성 향상을 위해서는 무엇보다 실제 비즈니스 프로세스를 적당한 수준의 서브시스템으로 명확하게 분석 및 분해하는 것이 가장 중요하다. 왜냐하면 서브시스템 분해 수는 개발 복잡도에 직접적으로 영향을 줄 수 있는 요소이기 때문이다.

그림 6에서 보듯이 서브시스템을 x1개로 분해할 경우 서브시스템간의 인터페이스는 줄어드나 서브시스템 내부가 tightly-coupled한 구조로 설계되어 개발복잡도가 y3로 커질 수 있고 이에 따라 유연성 및 민첩성이 저하된다. 반면 서브시스템을 x3개로 분해할 경우 각 서브시스템간의 인터페이스가 증가되고 하나의 업무 트랜잭션이 타 서브시스템에 영향을 미쳐서 개발복잡도가 y2로 증가될 수 있다. 결국 서브시스템의 독립성이 떨어지고 운영관리가 어렵게 된다. 따라서, 최적의 서브시스템 분해라 함은 그림 6에서 개발 복잡도를 y1으로 할 수 있는 x2개의 서브시스템 도출을 의미한다.

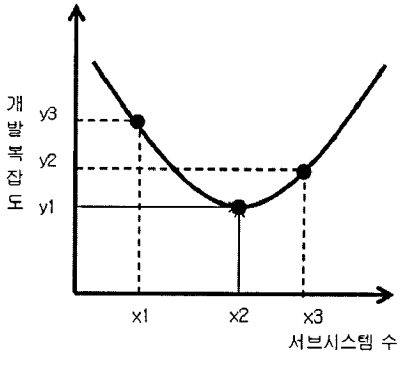

그림 6 서브시스템 분해와 복잡도

D사 구매관리 업무

L1	L2	L3
계약관리 (Contract) CRT	구매요청	구매요청
	견적	견적요청
		견적제출
		견적확정
	발주	발주
		수주확인
	계약	계약의뢰
		계약합의
		계약확정
지불관리 (Payment) PMT	검수	검수요청
		검수
	지급	세금계산서요청
		세금계산서발행
		세금계산서확정
		대금지급요청
		대금지급
공급업체관리 (Supply Company) SCM	공급업체등록관리	업체등록요청
		업체등록확정
	평가관리	업체등록평가
		계약평가
		검수평가

큰 입자 Top-down 분석 작은 입자

그림 7 ISIS 서브시스템 크기도

최적의 서브시스템 분해를 위해서는 ISIS 서브시스템의 크기를 정확히 이해할 수 있어야 한다. 그림 7은 컴포넌트, 서비스 그리고 서브시스템의 크기를 상대적으로 비교할 수 있도록 D사의 구매관리 업무를 하향식(top-down)으로 분석한 내용을 예로 보여주고 있다. 우선 L3 수준의 크기를 갖는 비즈니스 프로세스는 객체 또는 컴포넌트 단위 수준이며, L1 정도의 비즈니스 프로세스가 ISIS의 서브시스템으로 분해될 수 있는 정도의 단위이다. 즉, ISIS의 서브시스템 크기 수준은 SOA에서 말하는 서비스 수준과 같거나 상위에 있는 비즈니스 프로세스이다.

서브시스템 수는 개발 대상 업무의 특성 및 분석가에 따라 달라질 수 있기 때문에 최적의 서브시스템 분해를 위해서 다음과 같은 사항들을 고려해야 한다.

- 서브시스템 간 트랜잭션 처리가 독립적이어야 한다.
- 서브시스템 간 데이터 중복을 최소화 할 수 있어야 한다.
- 독자적인 DB 보유가 의미가 있는 수준이어야 한다.
- 3~6개월 정도의 개발 기간이 필요한 정도의 규모이어야 한다.

그리고 시스템 분석 시에 다음과 같은 원칙에 따라 ISIS 서브시스템을 식별할 수 있게 된다.

- ISIS 서브시스템은 사용자가 요청한 하나의 비즈니스

태스크를 시작해서 완료할 때까지의 모든 트랜잭션을 포함해야 한다.

- ISIS 서브시스템은 자신의 업무 트랜잭션에 대한 roll-back이 자체적으로 가능해야 한다.
- ISIS 서브시스템은 타 서브시스템 데이터의 생성, 갱신 그리고 삭제 요청 없이도 업무 트랜잭션 처리가 가능해야 한다.

3.3 ISIS 메시징 모델

서브시스템들이 독립적으로 개발 및 통합되기 위해서는 3.2절에서 설명한 것처럼 각 서브시스템들은 개별적인 DB 구축 및 자신이 필요한 모든 데이터를 가져야 한다. 즉, 타 서브시스템에게 필요한 데이터를 요청할 필요 없이 최신의 데이터를 유지해야 한다. 이를 위해서 각 서브시스템은 자신의 비즈니스 트랜잭션 처리 결과인 데이터를 변경 시점에서 관련 서브시스템에게 push 해야 할 의무가 있다. 하나의 엔터프라이즈 어플리케이션을 구성하는 각 서브시스템들은 이와 같은 데이터 push 메커니즘을 통해서 데이터 동기화가 가능해 진다. 그러나 이와 같은 데이터 동기화 및 메시징 모델 때문에 ISIS는 금융 비즈니스와 같은 실시간 시스템 분야 개발에는 적용이 힘들고 일반적인 비실시간 업무시스템 개발에 적당하다.

그림 8은 서브시스템이 비동기 전송해야 할 데이터의 유형과 push 메커니즘에 대해서 설명하고 있다.

우선 push 대상이 되는 데이터는 사용자 화면을 통해서 입력되는 A유형의 데이터와 내부 비즈니스 로직 처리에 필요한 정보성 데이터인 B유형 그리고 단순 조회용 데이터인 C유형으로 구분할 수 있다. 각 서브시스템은 A유형과 B유형의 데이터가 생성, 갱신 그리고 삭제되었을 경우 해당 데이터를 ISIS 지원 미들웨어로 push 해야 한다. 즉, 서브시스템A의 데이터 b가 변경되었을 경우 그 시점에서 자신의 DB 내용을 수정하고 동시에 ISIS 지원 미들웨어에게 데이터 b를 push한다. ISIS 지원 미들웨어는 데이터 b를 서브시스템B에게 전달하게 되고 서브시스템B는 서브시스템A에게 데이터 요청 없이도 자신의 DB에서 바로 데이터 b를 사용할 수 있게 된다. 따라서 로컬 DB에 있는 데이터는 항상 최신의 상태를 유지할 수 있고 중복성에 따른 데이터 동기화도 만족시킬 수 있다.

이처럼 분산 시스템에서 신뢰할 수 있는 데이터 전달을 통해 어플리케이션 간 고속의 비 동기 방식의 통신을 가능하게 하기 위해서는 메시징(messaging) 기술을 사용해야 한다. 특히 이러한 통신 메커니즘을 제공하는 미들웨어를 일반적으로 MOM(Messaging Oriented Middleware)이라 하고 분산 환경에서 이 기종 어플리케이션 간에 비동기 메시지를 라우팅하는 지능적인 교환기 역할을 담당한다. MOM은 일반적으로 Point-to-Point방식과 Publish/subscribe방식의 메시지 전송 패턴을 사용하는데, ISIS에서는 이 기종 분산 서브시스템간에 큐 기반의 비동기 데이터 전송이 가능해야 하기 때문에 publish/subscribe 방식의 메시지 전송 메커니즘을 기본으로 사용한다. 그림 9는 이와 같은 메시징 단계를 세부적으로 보여주고 있다.

① Producer의 adapter가 ISIS 지원 미들웨어에게 메시지 전달
② ISIS 지원 미들웨어는 전달받은 메시지를 대상 큐에 저장
③ ISIS 지원 미들웨어는 메시지를 로그 저장소에 저장
④ ISIS 지원 미들웨어는 producer에게 메시지 수신 확인 전달
⑤ ISIS 지원 미들웨어는 메시지 경로를 결정
⑥ ISIS 지원 미들웨어는 대상 메시지를 consumer의 adapter에게 전달
⑦ Consumer adapter는 메시지를 consumer 어플리케이션에게 전달
⑧ Consumer adapter가 메시지 사용에 대한 확인을 엔진에게 전달

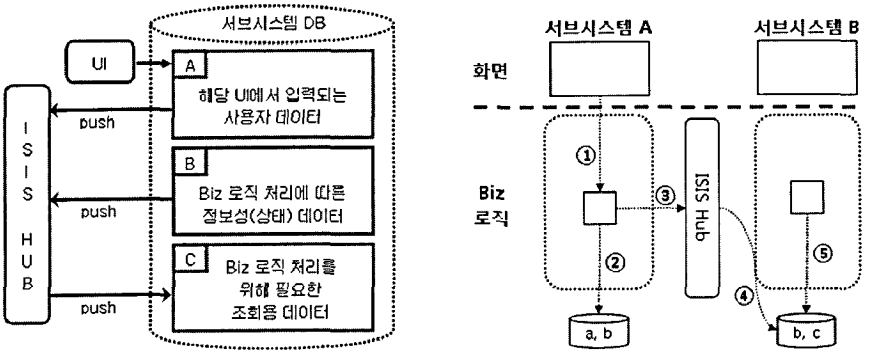

그림 8 데이터 유형과 push 방식

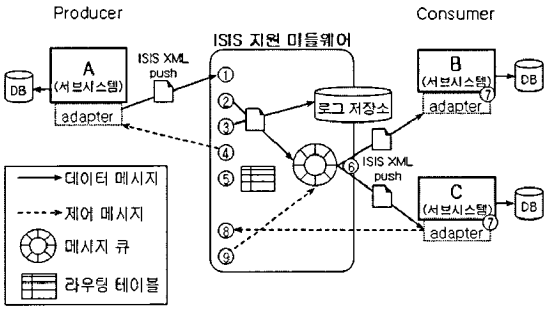

그림 9 메시징 메커니즘

⑨ ISIS 지원 미들웨어가 메시지 처리를 확인하고 해당 메시지를 삭제

ISIS에서는 서브시스템간 전송 메시지 구조를 XML 표준에 따라 정의하여 사용한다.

표 2는 ISIS XML Spec1.0의 구조를 보여주고 있다.

기본적으로 ISIS XML Spec1.0은 Header와 Body로 구성된다. Header는 메시지 전송에 필요한 정보들을 가지고 있는 부분으로서 ISIS Support Middleware에서 사용되는 정보들을 포함하고 있다. 그리고 Body는 소스 시스템에서 타깃 시스템으로 보내지는 데이터 부분으로서 Object, List 그리고 RList와 같은 3가지 형태의 데이터타입을 가지고 있다.

우선 Header의 주요 요소는 다음과 같다.

- SOURCE_ID: 메시지를 전달하는 소스 시스템의 ID 정보로서 ISIS에서 관리되는 ID이다.
- SOURCE_IP: 메시지를 전달하는 소스 시스템의 IP정보로서 물리적인 IP를 나타내며 하나의 시스템이 2대 이상의 서버로 구성될 경우 각각의 개별 IP가 들어간다.
- MSG_ID: 전송되는 메시지의 고유 ID로서 ISIS 기반으로 하는 시스템을 통틀어 유일한 메시지를 갖는다.
- MSG_TYPE: 메시지 전송 방식에 대한 정보로서, 해당 메시지가 1-way방식의 메시지인지 2-way방식의 메시지인지 나타낸다.
- MSG_ORDER: 메시지의 전송 우선순위로서, 일반/지연/긴급 3가지 타입이 있다.
- USER_ID: 소스 시스템을 통해 메시지를 발생시키는 주체로서 일반적으로 해당 메시지를 발생시키는 행위를 하는 시스템 사용자 ID나 사원번호 등이 들어가며, 배치작업 같은 시스템이 발생시키는 메시지에는 시스템 ID가 들어가게 된다.
- TIME: 소스 시스템에서 메시지가 발생할 때의 시간 정보다.
- MSG_VER: 전송되는 메시지 버전으로서 실제 데이터인 Body 부분의 구조가 변경될 경우 버전을 관리한다.
- TEMP1, TEMP2: 메시지 전송에 필요한 정보가 추가 될 경우 사용되는 임시 공간이다.

그리고 Body의 데이터 종류는 다음과 같다

- OBJECT: 개별형 데이터로서 전달하려는 메시지가 반복되지 않는 구조의 데이터 형태를 가질 때 사용된다.
- LIST: 반복형 데이터로서 전달하려는 메시지가 동일한 구조의 데이터가 반복되는 형태를 가질 때 사용된다.
- RLIST: 관계형 데이터로서 전달하려는 메시지가 2개 이상의 서로 다른 종류의 List형 데이터를 포함하며, 서로간의 관계(상하관계)가 나타날 경우 사용된다.

이와 같은 ISIS 메시징 모델은 4장에서 설명할 ISIS Support Middleware의 주요 서비스 기능으로 구현되어 제공된다.

4. ISIS Support Middleware

ISIS Support Middleware란 각 서브시스템을 연동하기 위해서 필요한 지원 기능을 제공하는 미들웨어 시스템이다. ISIS Support Middleware는 기본적으로 ISIS를 구성하는 각 서브시스템들 간에 큐 기반의 비동기 메시지 전송이 가능해야 한다. 이를 위해서 ISIS Sup-

표 2 ISIS XML Spec1.0

<HEADER>…</HEADER> <BODY> <OBJECT>…</OBJECT> <LIST>…</LIST> <RLIST>…</RLIST> </BODY>	<HEADER> <SOURCE_ID>...</SOURCE_ID> <SOURCE_IP>...</SOURCE_IP> <MSG_ID>...</MSG_ID> <MSG_TYPE>...</MSG_TYPE> <MSG_ORDER>...</MSG_ORDER> <USER_ID>...</USER_ID> <TIME>...</TIME> <MSG_VER>...</MSG_VER> <TEMP1/> <TEMP2/> </HEADER>

port Middleware는 최근 각광을 받고 있는 엔터프라이즈 어플리케이션 통합 기능과 SOA 지원 기능을 포함하는 ESB 미들웨어 제품을 기반으로 구축하였다. 그림 10은 ISIS 지원 미들웨어의 역할을 보여주고 있다. 그림 10에서처럼 ISIS Support Middleware는 ISIS의 각 서브시스템간에 데이터 전송뿐만이 아니라 기존 레거시 시스템간의 연동도 지원하도록 설계되었으며, 웹 서비스 및 개방형 표준 인터페이스 기술을 사용한다. 또한 HTTP, SOAP, JMS 이외에도 다양한 프로토콜을 지원한다. 예외적으로 SAP의 경우에는 RFC를 자체적으로 구현한 어댑터를 사용하고, DB 및 특정 패키지 어플리케이션인 경우에는 해당 어댑터를 추가적으로 구입 또는 개발하여 사용한다.

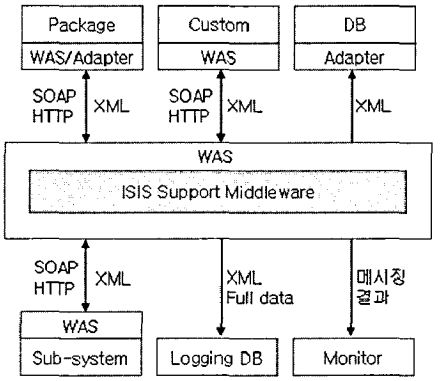

그림 10 ISIS 지원 미들웨어

ISIS Support Middleware가 제공하는 주요기능은 표 3과 같다.

그 동안의 연구개발을 통해서 ISIS Support Middleware를 다음과 같이 3가지 종류로 구축 및 개발하여 사용하고 있다.

- Enterprise ISIS Hub1.0: BEA WLI 제품을 커스터마이징한 ISIS Support Middleware
- Enterprise ISIS Hub2.0: IBM WPS 제품을 커스터마이징한 ISIS Support Middleware
- Open ISIS Hub: 오픈 소스를 사용하여 자체적으로 개발중인 ISIS Support Middleware

Enterprise ISIS Hub1.0 및 Enterprise ISIS Hub2.0의 경우 대규모 업무시스템 및 다수의 레거시 시스템 연동에 효율적인 반면 중소규모의 ISIS 시스템 개발에는 자체적으로 개발 중인 Open ISIS Hub가 TCO 측면에서 효과적이다.

Enterprise ISIS Hub1.0의 경우 ISIS 연구개발 초기 단계에 구축 및 파일럿 테스트에 사용하였고, 이후 Enterprise ISIS Hub2.0으로 개선되어 ISIS 신규시스템 개발에 적용하였다. Enterprise ISIS Hub2.0의 성능을 테스트한 결과 평균 3,600byte 정도의 XML 데이터를 내부적으로 처리하는데 걸리는 시간이 평균 0.19sec로 측정되었다. 이와 같은 처리성능은 일반적인 비실시간 업무시스템에 적용해도 지장이 없을 정도이다.

이와 같은 ISIS Support Middleware는 현재 D사의 기존 업무시스템에 적용하여 운영하고 있으며, 향후 모든 신규시스템 개발에도 적용할 예정이다.

5. ISIS 기반 어플리케이션 개발사례

5.1 ISIS 개발방식과 기대효과

ISIS의 개발방식은 다음과 같이 크게 2가지로 구분될 수 있다.

- Inter-system composition: 신규 엔터프라이즈 어플리케이션을 개발할 때 처음부터 ISIS 기반으로 개발하거나 확장해 나가는 형태의 개발방식이다.

  [case1: ISIS 어플리케이션 = 서브시스템1 + 서브시스템2 + 서브시스템3 + ……]
- 이 경우에는 개발 대상의 비즈니스 프로세스를 분석하여 서브시스템으로 분해 및 구현한 후에 ISIS Support Middleware를 통해 연동함으로써 ISIS 어플리케이션을 완성한다.

[case2: 개선 및 확장 ISIS 어플리케이션 = 기존 ISIS 어플리케이션 + 신규 서브시스템]

이 경우에는 이미 ISIS 기반으로 개발된 어플리케이션에 새로운 요구사항이 생기거나 점진적인 개발이 필요할 경우 신규 서브시스템을 추가함으로써 ISIS 어플리케이션을 완성한다.

표 3 ISIS 지원 미들웨어 기능

기능	내용
서브시스템 연결	표준 웹 서비스 방식(HTTP, SOAP, WSDL, XML 등)을 사용하여 서브시스템 및 레거시 시스템 연결
메시지 전송	One-way 전송 메커니즘 사용하여 XML 데이터 전송
메시지 큐잉	전송 신뢰성 보장 위해서 push된 메시지 큐에 저장관리
타깃 라우팅	1:N 메시지 전송을 위한 target 서브시스템 라우팅
메시지 로깅	Push된 XML 메시지 저장
전송 모니터링	메시지 전송 유무 및 성공여부에 대한 모니터링
장애처리	장애발생 시 메시지 전송보장을 위한 장애처리

- Intra-system composition: 기존 레거시 시스템에서 필요로 하는 신규 요구기능을 ISIS 서브시스템으로 개발하여 연동하는 형태의 개발방식이다.
  [case3: 개선 레거시 어플리케이션 = 기존 레거시 어플리케이션 + 서브시스템1 + 서브시스템2 + ……]
- 이 경우에는 ISIS 기반이 아닌 기존 레거시 어플리케이션, 특히 패키지 형태의 어플리케이션일 경우 신규 요구사항들을 서브시스템으로 개발하여 ISIS Support Middleware를 통해 연동함으로써 개선된 레거시 어플리케이션을 완성한다. 단, 새롭게 요구되는 기능이 ISIS의 서브시스템 정도의 단위일 경우 적용하는 것이 바람직하다.

이와 같은 방식으로 개발된 엔터프라이즈 어플리케이션이 ISIS 사상에 충실하고 서브시스템들이 매우 잘 설계되었을 경우에 표 4와 같은 이점들을 기대할 수 있다.

물론, 서브시스템별 DB운영으로 인한 데이터 스토리지 증가 및 실시간 데이터 처리문제와 분산 서브시스템 연동 및 처리로 인한 성능저하 등의 문제 등이 우려될 수 있다. 따라서 이에 대한 대처방안으로는 우선 최근 하드웨어 기술의 발달로 인해서 데이터 스토리지(storage)에 대한 부담은 적을 것으로 예상되며, ISIS 적용 분야를 비실시간(non real-time) 엔터프라이즈 어플리케이션으로 제한함으로써 실시간 처리에 대한 문제를 극복하고 ISIS 지원 미들웨어 기능 및 기술개선을 통해서 성능문제는 극복될 것으로 예상한다.

5.2 ISIS 기반 PMS 개발사례

PMS는 소프트웨어 개발 및 구축 프로젝트 관리 어플리케이션으로서 D사의 PMS에 새로운 비즈니스 요구사항이 발생하였으나, 해당 PMS는 S공급업체에서 개발한 패키지 솔루션이고 더 이상 기술지원이 불가능한 상황이었기 때문에 D사 자체적으로 내부 구조를 변경하는 것이 어려웠다. 따라서, D사는 ISIS 기술에 대한 파일럿 차원에서 개발방식 중 하나인 Intra-system composition 방식으로 개선 PMS를 개발하였다.

그림 13에서와 같이 개선 PMS는 기존 PMS에 손익분석과 비용전망이라는 새로운 요구사항을 서브시스템으로 개발하여 통합하였고, 이를 위해서 필요한 ERP 시스템인 SAP을 PMS의 논리적인 서브시스템으로 연동하였다. 이와 같은 개발방식은 기존 시스템의 수정을 최소화하면서 신규 요구기능을 추가적으로 개발하는데 적당하다. 각 서브시스템의 데이터포맷은 ISIS XML spec1.0을 사용하였으며, HTTP 및 SOAP 프로토콜을

표 4 ISIS 이점

기술 관점	비즈니스 관점
• 개발복잡도(Task-Structure Complexity, Chaotic Complexity) 감소로 개발기간 단축 • 전체 시스템 이해 없이 개발이 가능하여, 개발자 교육 및 투입이 용이 • Loosely-coupled한 구조에 따른 신규 비즈니스 요구사항 개발에 민첩 • 시스템의 품질관리 및 개선이 용이	• 개발기간 단축, S/W 라이프사이클 연장으로 전체 개발비용 절감 • 맞춤형 패키지 어플리케이션 제공 가능 • 제품인도시간 단축 • 새로운 고객 요구사항에 민첩한 대응이 가능 • 기반 IT 인프라 개선을 통해서 고객 서비스 향상

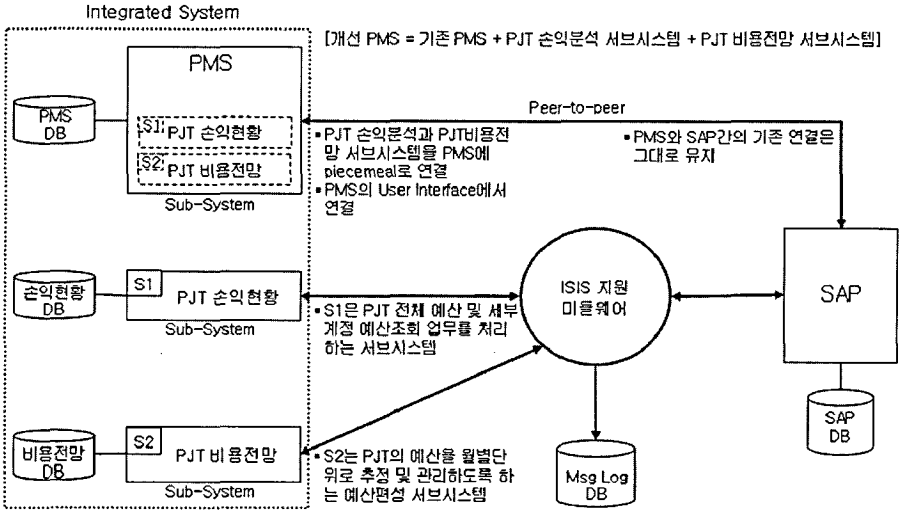

그림 13 ISIS 기반 개선 PMS 시스템

사용하여 인터페이스를 구현하였다. 특히, SAP는 RFC 구현을 통해서 Enterprise ISIS Hub1.0과 연동된다. 참고로 D사의 PMS 모든 데이터는 SAP에서 이중적으로 관리되고 있기 때문에 SAP만을 ISIS 지원 미들웨어에 연결하더라도 PMS의 데이터 동기화가 가능해진다

그 결과 기존 PMS의 수정이 거의 없이 새로운 요구사항을 민첩하게 추가할 수 있었고, 이를 토대로 기존 PMS의 생명주기(life-cycle)를 연장할 수 있었다.

5.3 ISIS 기반 ITSM 개발사례

ITSM은 단순히 IT 내부의 인프라스트럭처 운영 중심, 기술 중심의 관리 기능에서 벗어나 IT 서비스를 비즈니스 중심으로 재설계함으로써 협의된 서비스 수준에 맞는 서비스를 제공하고 효율적인 프로세스 운영을 통한 비용 절감을 목표로 하는 서비스 관리 시스템이다. 즉, ITSM 시스템은 ITIL 기반으로 IT 서비스 운영관리를 자동화 및 지원하는 시스템을 말한다. 여기서 ITIL은 IT 서비스 관리를 위한 대표 성공사례(Best Practice)를 제공하는 'de-facto' 표준으로써 IT 서비스 관리 프로세스를 정의하고 있으며 ITSM을 실현하기 위한 참조모델을 제공한다. ITIL의 구성은 고객과의 서비스 수준 계약 및 서비스 수준 모니터링에 필요한 제반 프로세스를 정의한 'Service Delivery' 영역과 이 프로세스를 운영 및 지원하기 위한 'Service Support' 영역으로 구분돼 있다.

D사가 운영해오던 ITSM은 H사의 외산 제품으로써 국내사용 환경에 맞도록 커스터마이징하는데 상당한 노력과 비용이 투자되었고 지속적으로 운영 및 유지보수 하는데 어려움이 있었다. 이에 따라 D사는 ISIS의 개발 방식인 Inter-system composition 방식으로 유연성 있는 국산 ITSM을 자체적으로 개발하게 되었다.

이를 위해서 우선 ITIL 버전2 기반의 IT 서비스 관리 프로세스를 분석하고 그림 14처럼 총 12개의 ISIS 서브시스템으로 분해하였다. 각각의 서브시스템은 개발자들이 독립적으로 설계 및 구현하였고, 이는 다시 Enterprise ISIS Hub2.0을 통해서 신규 ITSM으로 개발되었다. ISIS 기반의 ITSM 개발환경은 다음과 같다.

- 개발 언어: JAVA
- 운영체제: Windows
- DBMS: ORCLE
- 미들웨어: Enterprise ISIS Hub2.0
- 프레임워크: DAFrame
- 개발 툴: eclipse, WID
- WAS: WebSphere

표 5는 ISIS 기반의 ITSM 개발결과를 오프소스 ITSM 어플리케이션인 OTRS::ITSM[14]과 비교 분석한 내용이다. 본 결과분석에서 참조한 OTRS::ITSM는 ITIL 표준 프로세스를 따르고 있기 때문에 본 논문의 개발결과와 상대적 비교자료로 사용했지만, 두 시스템간의 업무처리 프로세스의 규모가 다를 수 있기 때문에 비교분석 결과에 차이가 있을 수 있다.

우선, LOC에서는 상대적으로 ISIS 기반 ITSM이 약 7.0% 정도 줄어든 것으로 파악되었으나 프로그래밍 습

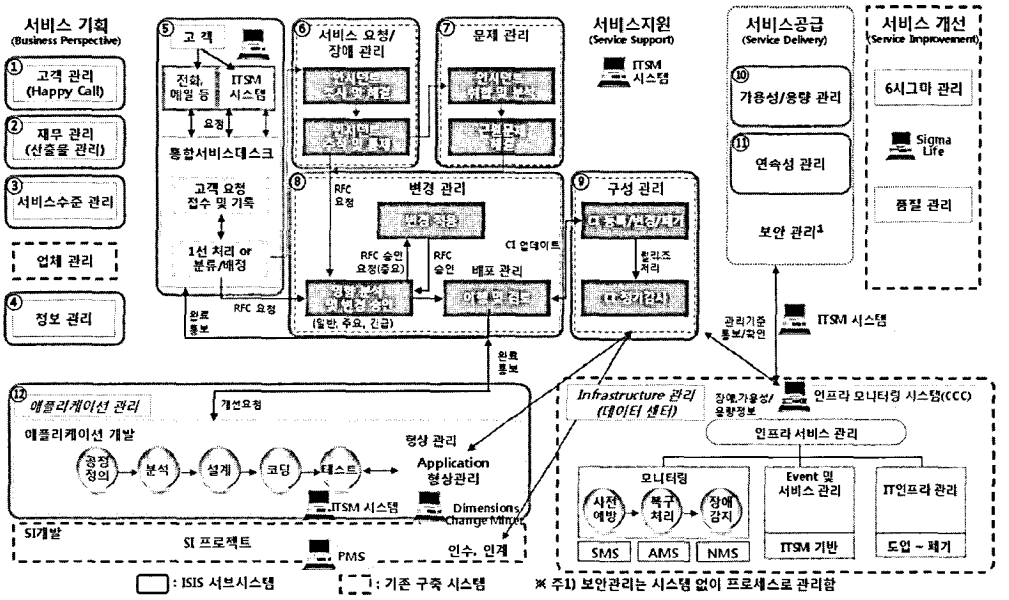

그림 14 ITSM 서브시스템 분해

표 5 개발결과 비교내용

구분	OTRS::ITSM	ISIS 기반 ITSM	비교결과
LOC	52,237(Perl)	114,279(Java)	약 7.0% 감소
투입인력	168 M/M	128 M/M	약 24.0% 절감
개발기간	확인 안됨	8개월	비교 안됨
개발비용	약 $ 747,088	약 $ 541,677	약 27.5% 절감

관이나 작성방법에 따라 이 정도의 차이를 있을 수 있다고 본다. 즉, LOC 측면에서는 거의 차이가 없다. 참고로, OTRS::ITSM의 LOC가 52,237라인이지만 Perl 언어를 사용했기 때문에 Java가 Perl에 비해서 약 2.4배의 코드길이가 늘어난다는 것[15]을 전제로 한다면 OTRS::ITSM를 Java로 개발했을 경우 133,188라인 정도 된다. 다음으로 투입인력 부분에 있어서는 약 24% 정도의 절감효과를 가져왔다. 이는 ISIS 기반으로 개발했을 때 서브시스템으로 분할 개발함으로써 개발 복잡도가 감소되었기에 가능한 것으로 해석할 수 있다. 이와 같은 분석결과는 ISIS 기반 ITSM 개발자들을 대상으로 설문조사를 한 결과, 실제로 기존의 개발방식에 비해서 개발이 용이했다는 응답결과가 뒷받침하고 있다. 즉, 개발자들은 전체 시스템을 이해할 필요가 없었고 고급 수준의 개발자들이 아니더라도 개발이 충분히 가능했었다. 그리고 개발기간 역시 OTRS::ITSM의 정확한 개발기간이 확인되지 않았기 때문에 상대비교가 어렵지만 LOC 및 투입인력에 대한 비교결과를 봤을 때 ISIS 기반 ITSM 개발기간이 단축되었을 것으로 예상된다. 마지막으로 개발비용은 현재 환율로 계산했을 때 ISIS 기반 ITSM 개발비용이 약 27.5% 절감된 것으로 분석되었다. 단, 환율변동, 임금차이 그리고 기타비용에 따라 차이가 있을 수 있고, 향후 운영 및 유지보수 비용까지 포함한 TCO 측면에서의 정확한 비용분석이 요구된다.

ISIS 기반 ITSM 개발사례에 대한 분석결과, ISIS 개발 아키텍처는 새로운 기술습득을 위한 학습곡선(learning curve)을 감안하더라도 엔터프라이즈 어플리케이션의 개발 복잡도를 줄임으로써 개발용이성 및 개발생산성을 향상하는데 기여한 것으로 분석된다. 그러나, 개발과정에서 ITSM의 서브시스템의 분해단위가 작아서 상대적으로 DB 중복 증가 및 상호간에 프로세스 독립적이지 못한 부분이 발생하였다. 따라서, 이와 같은 문제점들을 해결하기 위해서 ITIL 버전3을 토대로 5개의 서브시스템으로 리모델링하고 있으며 ISIS 방법론을 통해서 명확한 가이드를 제시하고자 한다. 만약 최적의 서브시스템 분해가 된다면 DB 중복이 최소화될 것이며 이로 인한 복잡도 역시 줄어들 것이다.

현재 ISIS 기반 ITSM은 개발 완료 후 운영 중에 있으며, 그 동안의 피드백을 토대로 지속적인 개선을 진행중에 있다.

6. 결론 및 향후 과제

수십 년 동안 많은 소프트웨어 개발 기술들이 해결하고자 했던 가장 중요한 문제는 바로 시스템의 복잡도를 줄임으로써 개발 생산성을 향상시키고 유연성 있는 시스템을 개발하는 것이었다. 특히, 오늘날의 대규모 엔터프라이즈 어플리케이션을 성공적으로 개발하기 위해서는 주요 컴포넌트 및 아키텍처가 관리 및 제어 가능한 정도의 복잡도를 가져야 한다. 이를 위해서 본 논문에서는 대규모 엔터프라이즈 어플리케이션의 개발 복잡도를 줄임으로써 개발 생산성을 향상시키고 독립적이고 유연성 있는 시스템 개발이 가능한 ISIS 개발 아키텍처를 제안하였다. ISIS는 개발 영역의 문제를 동시에 해결하지 않아도 될 정도의 서브시스템으로 분해하고 이를 독립적으로 개발함으로써 복잡도 문제를 해결할 수 있는 아키텍처 모델로서, 개발자들이 전체 시스템에 대한 이해 없이도 해당 비즈니스 프로세스를 담당하는 서브시스템 개발에 집중할 수 있도록 해준다.

본 논문에서는 ISIS 아키텍처를 적용 및 검증하기 위해서 엔터프라이즈 어플리케이션인 PMS와 ITSM를 ISIS 아키텍처 기반으로 개발하였다.

결론적으로 ISIS 개발 아키텍처를 적용하게 되면 기존 레거시 시스템에 대한 새로운 비즈니스 요구사항이 발생할 경우에 전체 시스템을 새롭게 개발하지 않더라도 민첩한 추가기능 개발이 가능하다. 그리고 신규 엔터프라이즈 시스템을 개발할 경우에는 개발 복잡도를 줄일 수 있어서 개발의 용이성 및 구조 유연성을 제공할 수 있다. 결국 이와 같은 ISIS 기술은 본 연구결과에서 알 수 있듯이 엔터프라이즈 시스템의 생명주기를 연장하고 개발생산성을 향상시킬 수 있을 뿐 아니라, 무엇보다 차세대시스템 개발이 빅뱅 방식에서 점진적 방식으로 전환하는데 기여할 수 있게 된다.

향후에는 ISIS 개발 아키텍처의 단점으로 지적되고 있는 데이터 중복문제, 데이터 정합성 문제, 장애처리 문제 그리고 통합테스트 및 관리 지원 도구에 대한 해결방안을 연구할 계획이다.

참 고 문 헌

[1] Roger Sessions, "Comparison of the Top Four Enterprise Architecture Methodologies," ObjectWatch, Inc., May 2007.
[2] Roger Sessions, "Controlling Complexity in Enterprise Architectures," June 2007.
[3] James Martin, Rapid Application Development, Macmillan Publishing Company, 1991.
[4] Anand Ranganathan, Roy H. Campbell, "What is the Complexity of a Distributed System?," 2005.
[5] Thomas J. McCabe, Charles W. Butler, "Design

Complexity Measurement and Testing," December 1989.
[6] Tim Jennings, Rob Hailstone, "Developing Composite Applications," December 2006.
[7] Herb VanHook, "Reduce IT Costs and Complexity with Effective Application Problem Management," November 2006.
[8] Roy W. Schulte, Yefim V. Natis, "Service Oriented Architectures, Part1," Gartner, 12 April 1996.
[9] Roy W. Schulte, "Service Oriented Architectures, Part2," Gartner, 12 April 1996.
[10] IBM, "Service Oriented Architecture 백서," March 2006.
[11] Paul Giurata, "Enterprise RIAs close the performance gap between on-premise software and SaaS, 24 September 2008.
[12] http://mm.sookmyung.ac.kr/%7Ek920/hw1/CBD.htm
[13] Mika Koskela, Mikko Rahikainen, Tao Wan, "Software development methods: SOA vs. CBD, OO and AOP," 2008.
[14] http://www.ohloh.net/
[15] http://servicexen.wordpress.com/2008/07/25/common-problems-with-it-project-cost-estimation/

조 은 환

1997년 건국대학교 전자계산학(학사). 1999년 건국대학교 대학원 컴퓨터정보통신학(석사). 2005년 건국대학교 대학원 컴퓨터정보통신학(박사). 2003년~2005년 건국대학교 컴퓨터공학부 강의전담 교수 2006년~현재 동부CNI 연구소 정보화추진팀. 관심분야는 엔터프라이즈 아키텍처, 소프트웨어 공학, 실시간 미들웨어 시스템

이 갑 훈

1982년 서울대학교 조선공학과 졸업(학사). 1984년 서울대학교 조선공학과 졸업(석사). 2005년 서울대학교 산업공학과 졸업(박사). 2004년~현재 동부CNI SD 사업부장. 관심분야는 프로젝트 관리, 구매관리

이 민 수

1990년 한양대학교 수학과 졸업(학사) 1990년~2002년 창현정보기획 개발 실장. 2002년~현재 동부CNI 연구소 정보화추진팀장. 관심분야는 IT 아웃소싱, 엔터프라이즈 아키텍처

이 봉

1973년 서울대학교 금속공학(학사). 1978년 미국뉴욕대학교대학원 컴퓨터공학(석사). 1979년 미국뉴욕대학교대학원 컴퓨터공학(박사수료). 1983년 미국컬럼비아대학교 경영학(석사). 2005년 한국외국대학교대학원 경영학(박사). 2007년~현재 동부CNI 대표이사. 관심분야는 IT 아웃소싱, 시스템개발 방법론, 소프트웨어 패키징 방법론