경영정보학연구
제9권 제1호
1999년 3월

전사적 자원 관리 시스템을 위한 기업 리파지토리 구축*

이 희 석**, 이 제**, 이 충 석**, 조 창 래***, 손 주 찬****, 백 종 명****

Implementing an Enterprise Repository for Enterprise Resource Planning System

Lee, Hee Seok, Lee, Jay, Lee, Choong Seok, Cho, Chang Rae, Sohn, Joo Chan, Baik, Jong Myung

Currently, many companies are trying to be more competitive by implementing the most appropriate software package, called ERP (Enterprise Resource Planning). ERP can support the company's business process. A repository, which can store and retrieve enterprise-wide information, is a key component of such software package. This paper implements an enterprise repository for the ERP. A well-known repository standard, IRDS (Information Resource Dictionary System), is employed for the repository architecture. The proposed metadata schema can help develop database, business applications, models, and workflow. To illustrate the practical usefulness, a real-life ERP prototype is built within the framework of the proposed repository.

* 본 연구는 한국과학기술원 수탁연구과제 (98-17-P00540) 로 수행되었음.
** 한국과학기술원 테크노경영대학원
*** 삼성카드주식회사 마케팅팀
**** 한국전자통신연구원 전자상거래연구부

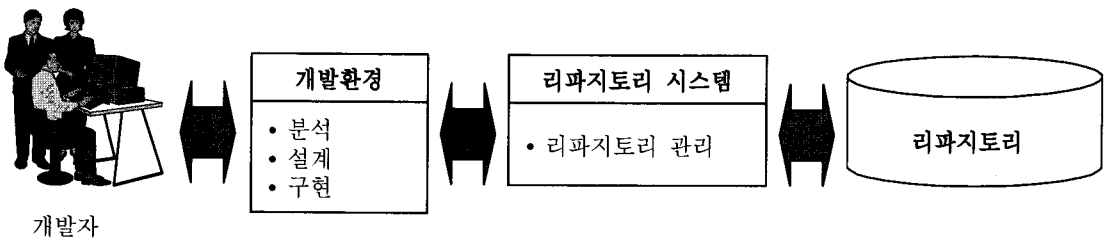
I. 서 론

급변하는 새로운 경영환경은 현재 가지고 있는 정보기술의 변화와 경영혁신을 끊임없이 요구한다. 이러한 혁신이 전사적 계획에 따라 이루어져야 함은 주지의 사실이다. 즉, 필요 시마다 개별적으로 구축되어진 정보시스템은 이질적인 모델로 인하여 생산성 향상과 정보 재사용성에 제약을 받게 된다. 이러한 문제점을 전략적으로 대처하기 위해서는 변화를 통합적으로 관리할 수 있는 기법이 필요하다. 지금까지 이질적인 시스템 환경에 저장되어 있는 다양한 기업 내 정보 자산을 통합적으로 관리하고 재사용하기 위해서는 CASE (Computer Aided Software Engineering), 시스템의 배치 (Configuration) 와 버전관리 (Version Control), 통합모델관리를 사용하였다. 이러한 기술의 기본은 메타데이터 (Metadata) 를 활용하여 이질적이거나 분산된 자원을 통합적으로 관리하려는 것이다.

메타데이터에 관한 기존 연구는 단일한 CASE 환경, 또는 데이터베이스의 기술 정보 만을 포함하는 협의의 메타데이터에 관한 것이었다. 이러한 한계를 극복하기 위한 하나의 방법으로 전사적 자원 관리 시스템 (ERP, Enterprise Resource Planning) 을 위한 메타데이터의 연구가 필요하다. ERP는 기업의 생산, 자재, 영업, 인사, 회계 등의 업무를 통합 관리해주는 대형 경영 관리용 소프트웨어 패키지이다. 다수의 기업들이 미래 시장에서 경쟁성을 유지하기 위하여 ERP를 도입 중이거나 도입을 심각히 고려 중이다. ERP 패키지 가운데 자신의 기업환경에 적합한 모듈만 부분적으로 선택하여 유연하게 개발하는 것도 가능하다. 이것이 최근의 소프트웨어 개발방법론으로 각광 받는 컴포넌트 기반 개발 (CBD, Component Based Development) 의 하나의 형식이다 [Kiely, 1998]. 이러한 시스템 통합성과 유연성이 ERP의 장점이다.

리파지토리 (Repository) 는 ERP, 나아가서 일반 정보시스템의 지식관리 기능을 수행하는 구성요소이다. 리파지토리는 기업의 경영전략에서 구현까지의 모든 정보를 통합적으로 관리하며, 특정 개발 환경에 독립적으로 존재할 수 있는 메타데이터를 관리한다 [MacGauhey, 1997; Philip, 1994]. 실제로 SAP (Systems, Applications and Products) [1998]와 같은 선진 ERP 업체도 리파지토리의 비중을 중요시 하고 있다. 즉, 비지니스를 비지니스 객체로 리파지토리에 저장하여 특정 업무 영역의 표준지식을 표현하고 있다.

리파지토리를 활용한 정보시스템의 개발은 <그림 1>과 같이 도식화 할 수 있다. 리파지토리 시스템은 분석, 설계, 구현을 위한 여러 도구들을 통합적으로 지원하며, 리파지토리에 접근하기 위한 데이베이스 관리시스템 (DBMS, Database Management System) 과의 인터페이스를 제공한다. 현재 대부분의 개발환경은 모델에 근거한 CASE에 기반 한다. 리파지토리를 위한 표준으로는 IRDS (Information Resource Dictionary System)가 가장 광범위하게 사용되고 있다. IRDS는 분석에서 구현까지의 일관성 유지 및 모델 독립성에



<그림 1> 리파지토리를 이용한 정보시스템 개발

타 표준에 비해 우수한 것으로 알려지고 있다. 본 논문의 목적은 ERP를 위한 통합 지식 저장소로서 IRDS 기반 리파지토리를 구현하는 것이다.

Ⅱ. 리파지토리 정의 및 현황

리파지토리에 관한 기존 정의가 <표 1>에 요약되어 있다. 본 연구에서는 Tannenbaum [1994]의 정의에 따라 리파지토리를 모델과 구현도구에 독립적으로 활용가능하고, 정보자원 공유가 가능한 저장구조 통합체로 정의한다. 즉, 리파지토리의 통합성과 독립성을 강조한다.

리파지토리 활용의 장점으로는 (i) 특정 개발 환경에 독립적이며, (ii) 이질적인 개발 도구들을 상호 유기적으로 연결하여 통합적인 개발 접근이 가능하며, (iii) 장기간 사용 시 기업 지식의 축적이 가능해진다는 것이다. 궁극적으로 리파지토리는 조직이 가지고 있는 모든 지식 자산을 통합적으로 저장 관리 할 수 있는 기반 기술이다.

리파지토리 이외에도 지식자원 통합을 목적으로 한 메타데이터 기술에는 자료사전 (Data Dictionary) [Hakala, 1996] 과 백과사전 (Encyclopedia) [ECMA, 1990] 이 있다 (<표 2> 참조). 자료사전은 데이터베이스 자료구조, 레코드 및 필드 등 단순한 데이터 관리용 메타데이터를 다룬다. 자료사전은 계획에서 분석, 설계, 그리고 구현에 이르기까지의 모든 정보를 통합 관리하는 백과사전으로 발전하였다. 그러나, 현재의 기업 전산환경은 대부분이 이질적 시스템의 통합이므로, 단일한 CASE 환경에서의 백과사전으로는 그 한계가 있다. 전사적 경영전략에서 구현까지의 모든 정보를 통합적으로 관리하며 특정 개발 환경에 독립적으로 존재할 수 있는 리파지토리가 필요한 것이다. 리파지토리는 기

<표 1> 리파지토리 정의

정의자	정 의
Martin [1989]	자료정보 뿐만 아니라 계획, 모델, 설계사양의 함축적 표현도 포함
Moriarty [1990]	각 공급업체로부터 제공되는 특수한 응용도구가 아니라, 기업의 전략적 관리를 위한 시스템
MacGaughey & Gibson [1993]	정보저장을 위한 빈 그릇으로 비유할 수 있으며, 기업모형의 컴포넌트들이 채워진다면 정보백과사전 이라고 할 수 있음
Tannenbaum [1994]	리파지토리 이외에 존재하는 이질적인 모델타입에 연계되어질 수 있는 하나의 통합된 저장 공간, 즉, 모델과 구현도구에 독립적으로 활용가능하고 정보 자원의 공유를 위해 조직화 되어진 저장 구조 통합체

<표 2> 메타데이터 기술의 발전

특성 \ 관리기술	자료사전	백과사전	리파지토리
자료범위	물리적 데이터베이스 메타데이터	시스템 분석설계의 결과	전사적 정보
구 조	비범용성	특정 비즈니스 영역에 제한적 사용	범용 구조 지향
공급업자와의 관계	종속적	종속적	독립적
표준화 작업	DBMS 업체별	CASE 도구 업체별	다양한 표준 기관 (IRDS/CDIF/PCTE/ATIS)
모델지원 여부	특정 모델만 지원	특정모델을 지원하나 모델 호환성 일부 가능	다양한 모델지원

업 내 존재하는 다양한 지식 자산을 통합 모형화 하여 관리하기 위한 일종의 메타데이터이다.

리파지토리 시스템은 리파지토리를 용이하게 관리할 수 있는 기능을 제공한다. Bordoloi [1998]는 리파지토리 시스템의 요구 기능으로 데이터 관리, 질의 관리, 버전 관리, 보안, 자료 변환 (Import/Export), 시스템 배치 관리, CORBA (Common Object Request Broker Architecture) 또는 DCOM (Distributed Component Object Model) [Orfali et al., 1996] 과의 인터페이스, 기타 API (Application Program Interface) 를 제시하고 있다. Microsoft, DEC, Unisys 등의 기업에서는 자체 표준안을 기반으로 리파지토리 상용화를 시도 중이며, SAP와 Oracle에서는 각자의 ERP에 일부 리파지토리 기능을 적용하고 있다. 그러나 현재 기업 지식의 통합적 접근을 시도하기 보다는 구현 단계에 집중된 초보적인 시도이다. 즉 리파지토리 기술의 핵심인 메타데이터 기술의 경우, 분석, 설계 및 구현의 전과정을 통합적으로 지원하기에는 미비한 데이터베이스시스템의 메타데이터 수준을 벗어나지 못하고 있다. 또한 실제 사례를 통한 구체적인 경험이 부재한 것도 문제로 지적될 수 있다. 특히, 국내의 경우는 그 경험이 거의 전무한 상황이며 리파지토리의 개념 자체도 명확하게 인식되어져 있지 않는 것이 현실이다.

Ⅲ. IRDS 기반 리파지토리

리파지토리 표준안으로는 가장 광범위하게 사용되는 IRDS 외에도 다음의 몇 가지가 있다. 우선 CASE 도구간의 모델 교환 표준 명세가 주목적인 CDIF (CASE Data Interchange Format) [1994] 가 있다. CDIF는 정보저장소의 자료 및 정보를 개별 CASE 도구 등에 보내거나 받는 경우 등에 상용된다. 그러나, 메타메타모델 계층, 메타모델 계층, 모델 계층, 데이터 계층 등의 계층간 정보공유와 상호참조는 불가능하다. PCTE (Portable Common Tools Environment) [ECMA, 1990] 의 주목적은 표준적인 소프트웨어 공학 환경 구축에 있다. 즉, PCTE는 구현 도구들간 통신 기반과 확장성을 제공한다. PCTE는 객체 관리 시스템 (OMS, Object Management System) 을 인터페이스로 사용하나, 개체 관계 모델 (ERM, Entity Relationship Model) 에 기반을 두고 있으며, LAN (Local Area Network) 기반의 분산 구현을 지향하고 있다 [Bernstein & Umeshwar 1994]. 모델 및 API 공유 표준명세인 ATIS (A Tool Integration Standard) [Tannenbaum, 1994] 는 통합 프로젝트 지원 환경을 위한 객체 지향적 관점의 정보 저장소 인터페이스를 정의한다. ATIS와 PCTE는 통합 프로젝트의 생명주기 전 단계를 지원하나 각 단계별로 상호 참조는 불가능하다 [Altomare et al., 1989][양해술, 1994].

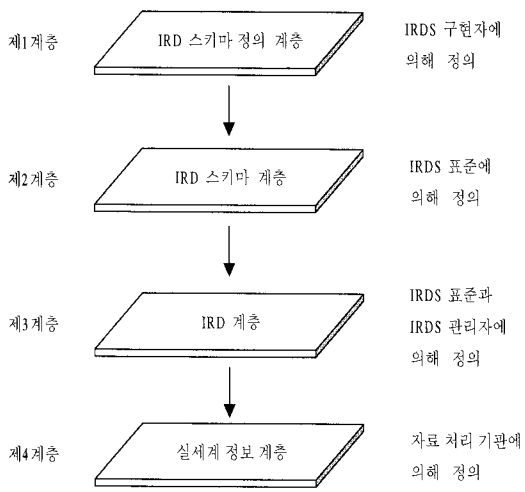
IRDS는 ANSI (American National Standard Institute) 와 NIST (National Institute of Standards and Technology) 에서 제시한 4계층 프레임워크를 따른다. IRDS는 기본적인 메타데이터 구조와 자료변환에 대한 정의 [ANSI, 1989] 를 제공하나 구체적인 설계와 구현에 대해서는 각 설계자의 자유에 맡기고 있다. 즉 상위층의 개념적 디자인 틀 제공에 중점이 주어지고 있다. <표 3>은 각 표준안을 비교하고 있다.

IRDS는 <그림 2>와 같은 4계층 구조를 기반으로 한다. 한 계층의 정보는 다음 하위계층에 저장된 정보를 설명하고 제어한다. IRDS 표준안은 계층 개념을 통해서 두 계층을 하나의 쌍으로 구성하며 상위층은 정의 (Definition), 하위층은 인스턴스 (Instance) 로 이루어진다. 즉, 다음 하위계층의 개체 타입을 그 상위계층에서 정의하게 되고, 하위계층은 상위계층에서 정의된 개체 타입에 관한 실제 값을 갖는다. IRDS는 다차원 모델하에서 계층간의 일관성을 유지함으로써, 계층간의 인터페이스 문제를 해결하고 분석에서 구현까지의 일관성과 각 모델의 독립성을 보장한다. 본 논문에서 집중 분석되어

<표 3> 리파지토리 표준안 비교

비교기준 \ 표준안	IRDS	CDIF	PCTE	ATIS
표준화 기관	ANSI, ISO	EIA, ANSI	ECMA	DEC, AT
특 성	공용적 리파지토리 자료구조제공으로 유연성 보장	CASE 메타 데이터 정의와 CASE 간 데이터 전송	-객체 관리 시스템 -소프트웨어 공학 환경의 기저제공 목적	객체지향 기반 리파지토리 인터페이스 정의
구 조	4계층 구조 1)IRD 스키마 정의 계층 2)IRD 스키마 계층 3)IRD 계층 4)실세계 정보 계층	4계층 구조 1)메타메타모델 계층 2)메타모델 계층 3)모델 계층 4)데이터 계층	2계층 구조 1)타입 (메타메타 데이터) 계층 2)스키마 정의 집합 타입 계층	미계층 구조
모델통합	개념적 가이드라인 제시	미지원	미지원	객체지향 관점에서만 지원
미 비 점	개념적 가이드 라인만을 제시	계층 간 정보공유와 상호참조 불가능	미세한 데이터 공유에 부적합	단계별 상호참조 불가능

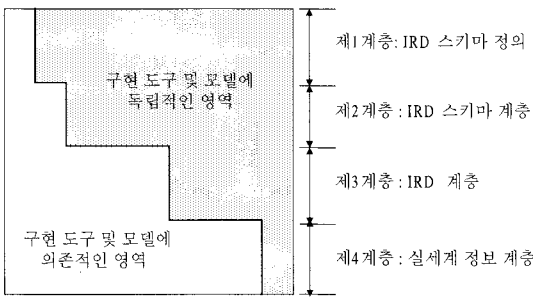
EIA : Electronic Industries Association
ECMA : European Computer Manufacturer's Association
DEC : Digital Equipment Corporation
AT : Atherton Technologies
CRUD : Create, Read, Update, Delete



<그림 2> IRDS 4계층 기본 모형

질 내용은 IRDS의 핵심인 IRD 스키마 계층과 IRD 계층 (제2, 제3계층) 이다. 따라서 이 부분에 대한 집중적인 연구와 설계가 수행되었다.

현실적으로 IRDS의 독립성은 계층별 특성을 보인다. ATIS는 객체 지향적 관점에서 IRDS에 흡수되어지는 과정에 있는데 [Tannenbaum, 1994], 이는 IRDS의 통합적 특성을 시사하고 있다. 통합을 위한 이상적인 IRDS 4계층 구조는 각 계층별로 모델 및 구현 도구에 독립적인 것이다. 이론적으로, IRDS는 각 계층마다 구현도구 및 모델에 의존적인 영역이 없고 모두 독립적인 영역만으로 사상 될 수 있다. 그러나, IRDS가 제시하는 본연의 구조적 한계로 인해 현실적으로는 하위계층으로 갈수록 모델과 구현 도구에 의존적일 수 밖에 없다 <그림 3>. 특히 제4계층의 의존성이 큰 것은 현실적인 정보 처리 단계의 내용인 인스턴스를 담고 있는 것에 기인한



<그림 3> IRDS 표준안의 계층별 독립성

다. 반면, 4계층의 상위계층인 제3계층은 제4계층의 인스턴스에 관한 정의를 하고, 이를 제어하므로 제4계층보다 추상적이며 독립적인 영역이 커지게 된다. 이러한 2계층 쌍구조가 상위계층에도 연쇄적으로 적용되므로, <그림 3>에서와 같이 상위층으로 갈수록 모델 및 구현도구에 독립적인 영역이 커져 가는 구조로 볼 수 있게 된다. 완전한 독립성은 향후 객체지향 개념의 도입으로 가능할 것으로 판단된다.

Ⅳ. ERP리파지토리 구축

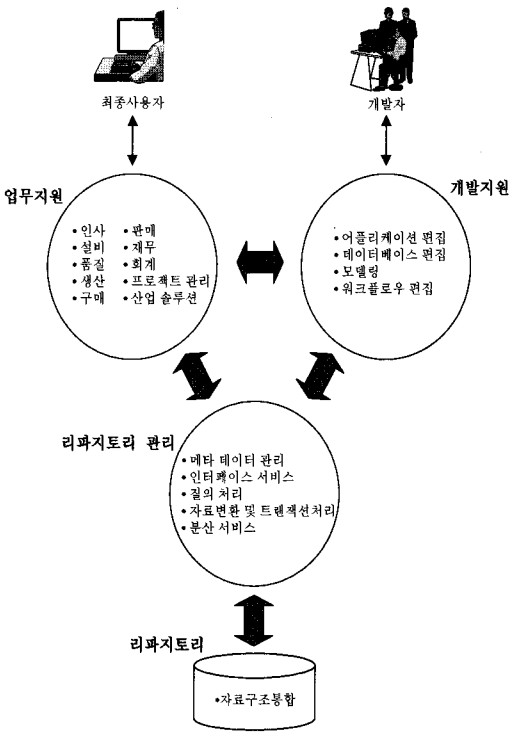
리파지토리 설계 및 시스템 관련 기술은 여러 국제 표준 기구 등에서 표준화를 시도하고 있으며 업계에서도 많은 관심을 갖고 있는 필수적인 분야이다. 그러나, 전사적 자원 관리에서는 표준화된 환경의 모델통합이라는 기술적 난이함에 의해 실용화에 많은 어려움이 있었다. 성공적인 ERP시스템을 위한 자원의 통합관리에는 리파지토리가 필수적이다. 또한 ERP 시스템의 도입 시 기업이 당면하는 비용문제 및 국내의 원천 기술 확보 측면에서 전략적 기술의 활용 기반 구축 및 실제 적용 사례 개발이 절실히 요구되어 왔다.

이런 취지에서 1997년 말부터 시작된 ERP 구축 프로젝트는 ㅎ 연구원, 국내 10여 개 시스템 개발업체와 한국과학기술원에서 수행 중이며, 약 50 여명의 개발, 설계 및 프로젝트 관리 인원이 참여하고 있다 [이희석, 1998]. 한국과학기술원은 특히 리파지토리의 개념적 설계 및 구축을 수행하고 있으며, 그 연구 범위는 <표 4>에 요약되어 있다.

ERP 시스템은 <그림 4>와 같이 업무지원, 개발지원, 리파지토리 관리, 리파지토리의 4개 서브시스템으로 구성된다. 최종사용자는 ERP시스템을 이용하여 현업을 수행하며, 개발자는 기업의 요구사항에 맞추어 ERP를 개발한다.

<표 4> 표준 리파지토리 구축 프로젝트 연구 개발 범위

연구개발내용	연 구 개 발 범 위
자료수집 및 문헌연구	-리파지토리 관련 연구 문헌 조사 -ERP 관련 연구 문헌 조사
메타데이타 표준안 분석	-IRDS, CDIF, ATIS, PCTE의 표준안 비교 -IRDS 4계층 분석
ERP 요구사항 분석	-ERP 기능 기본 요구사항 분석
리파지토리 메타데이터 설계	-ERP 분석설계용 리파지토리와 인터페이스 연구 -ERP 구현용 리파지토리 메타데이터 스키마 시제품 제작
리파지토리 구현	-데이터베이스 상의 실제 리파지토리 설치 -리파지토리 튜닝



<그림 4> ERP 시스템 아키텍쳐

각 서브시스템 세부 기능의 요약이 <표 5>이다. 이들 세부기능을 설명하면 다음과 같다.

<표 5> ERP시스템 세부 기능

서브시스템	세 부 기 능	설 명
업 무 지 원	인 사	인사계획, 조직 개발, 인사정보관리, 근태관리, 시간관리, 급여관리, 출장여비 관리 등의 종합적 인사관리
	설 비	보전일정관리, 예방보전, 보전 능력 계획, 보전정보관리
	품 질	생산된 상품의 불량 원인 관리 및 사무 간접 서비스 관리
	생 산	생산 계획 수립, 작업지시 및 작업실적 등록, 생산 모델 정의, 재고관리, 품질 관리, 발주관리
	구 매	구매요구관리, 견적관리, 거래계약관리, 청구서조회, 창고관리
	판 매	거래 조회, 견적, 수주, 출하, 청구 등의 판매/유통 관리, 판매분석
	재 무	간접비관리,제품원가계산 및 관리에 대한 다양한 원가관리, 업적관리, 수익성분석
	회 계	일반회계, 외상매출관리, 외상매입관리, 어음관리, 예산관리
	프로젝트 관리	프로젝트 일정관리, 프로젝트 정보관리
	산 업 솔 루 션	석유, 은행, 병원, 보험 등 업종 특유의 요구사항 충족을 위한 기능만을 별도로 모듈화하여 관리
개 발 지 원	테이터베이스 편집	데이터 타입 정보관리, 테이블과 관련한 속성 등의 정보 관리 기능 지원
	어플리케이션 편집	업무 분류, 사용자 관점에서의 화면 관리, 실행을 위한 트랜잭션 처리, 데이터베이스와의 연동되는 프로그램 구성 지원
	모 델 링	정보시스템 개발 시 모델의 생명주기 관리에 대한 정보, 모델 버전 관리 정보, 배치 관리, 모델간 정보 교환 기능 지원
	워크플로우 편집	비즈니스 프로세스 자동화, 조직의 사용자, 부서, 권한, 보안 관리를 위한 지원
리파지토리 관 리	메타데이터 관리	전사적 관점에서의 데이터 요소들간의 표준화, 무결성 유지, 자료값의 범위 제한, 자료 유효성 명시
	인터페이스 서비스	IRDS 계층 간의 인터페이스 및 사용자 인터페이스 처리상태 보고, 자료 입출력 결과 표현, 에러 메시지출력, 도움 기능, 출력에 필요한 리파지토리 관리 기능
	질 의 처 리	질의/답변 처리를 위한 리파지토리 관리
	자료변환 및 트랜잭션 처리	IRDS간 데이터 처리를 위한 자료변환과 트랜잭션 처리
	분 산 서 비 스	이질적인 데이터베이스 하에서의 정보 교환
리파지토리	자료구조 통합	IRDS 표준안에 근거하여 기업의 정보 및 자원을 전사적으로 관리하기 위한 자료 통합 구조

업무지원 서브시스템은 인사, 설비, 품질, 생산, 구매, 판매, 재무, 회계, 산업 솔루션 관리, 프로젝트 관리의 업무를 지원한다. 리파지토리 관리 서브시스템은 메타 데이터 관리, 인터페이스 서비스, 질의 처리, 자료변환 및 트랜잭션 처리, 분산 서비스 기능을 제공한다. 업무지원과 리파지토리 관리 기능은 개발지원 기능과 연계되어 있으므로 개발지원 서브시스템의 상세한 설명으로 대신하고자 한다.

개발지원 서브시스템은 크게 어플리케이션 편집, 데이터베이스 편집, 모델링, 워크플로우 편집의 4가지 기능을 제공한다. 어플리케이션 편집 분야는 각 업무를 대중소로 분류한 후, 응용프로그램들을 해당되는 각 분류체계에 할당하는 방

법을 사용한다. 이는 기존 ERP 시스템에서 가장 많이 사용되는 방법이다. 또한 객체지향 개념을 사용하여 실행 가능한 최소의 단위인 메소드 (Method) 를 정의하여 재사용성을 증대시키며 유지보수를 용이하게 한다. 사용자 인터페이스 역시 객체지향 개념을 이용하여 관리하며, 모든 어플리케이션들이 표준 인터페이스를 사용할 수 있도록 한다.

데이터베이스 편집 분야의 경우, 이질적인 DBMS를 사용하는 분산환경에도 적용 가능한 리파지토리 설계가 요구된다. 이를 위해서 Microsoft SQL Server 6.5와 Oracle Workgroup Server7.3의 메타데이터 스키마를 분석하고, DBMS가 데이터를 관리하는 방법들을 조사하여 공통 기능을 정의하였다.

모델링 지원분야의 경우, 모델간의 호환성이 관건이다. 본 연구에서는 재사용성을 목적으로 UML (Unified Modeling Language) [OMG, 1997]을 사용하였다. 워크플로우 분야는 조직의 정보와 밀접한 관계를 갖고 있다. 조직과 관련하여 사용자, 부서, 업무, 권한 등에 대한 내용을 관리하는 것이므로 데이터베이스나 어플리케이션의 보안도 중요하다. 예를 들어 조직 내 인력과 데이터베이스 및 어플리케이션의 사용자는 동일하게 관리되어야 하는 경우가 그것이다. 이외에도 버전 관리와 관련된 기본적인 요구사항도 포함된다.

V. 리파지토리 스키마 사례

본 연구에서는 IRDS 표준안에 따라 4계층의 리파지토리가 구현되었다. 이 중 실세계 정보를 관리하기 위한 계층이 바로 IRD계층 (제3계층)이다. 이 IRD계층을 추상화한 계층이 IRD스키마계층 (제2계층) 이다. 제2계층을 관리하는 IRD 스키마 정의계층으로 제1계층이 되는 것이다. 1계층은 2계층의 내용을 추상화한 개체, 관계, 속성의 3종류의 개체가 산출되어있다. 향후 추가적으로 보완될 개체가 프로젝트의 진행과정에서 산출될 예정이다. 특히 객체지향 개념의 도입에 따라 상위 개념의 변경이 예상된다. 제3계층은 제2계층에 기반한 개체 인스턴스 집합이라고 할 수 있다. 본 연구결과 제3계층은 인사, 설비, 품질, 생산, 판매, 재무, 회계, 프로젝트 관리와 산업 솔루션 등을 지원하기 위한 개체로 500여 개가 산출되었다. 예를 들어, 생산의 경우 제품, 제품별 자재명세 (BOM, Bill Of Material), 자재, 제품재고, 자재재고 등의 개체가 산출되었다.

본 절에서는 설명의 편의상 IRDS의 중요 계층인 제2계층을 중심으로 내용을 요약하겠다. 적용 현실성을 고려하여 IRDS 기반 메타스키마 모델링에 개체 관계도 (ERD, Entity Relationship Diagram) 를 이용하였다. 제2계층의 스키마를 개체 관계도로 도식화한 내용이 <그림 5>이다. 60여 개의 개체가 도출되었으며, 이들에 대한 상세한 설명은 부록에서 참조할 수 있다. 제 2계층 스키마를 좀 더 설명하면 다음과 같다.

우선 어플리케이션 편집부분 (<그림 5>에서 실선으로 표시) 에 대한 리파지토리 스키마는 크게 5부문으로 대별된다. 즉, (i) 기업의 업무분야를 크게 대중소로 구분하는 업무 분류 관련 스키마, (ii) 화면 관련 개체 및 어플리케이션 실행을 위한 트랜잭션 관련 스키마, (iii) 트랜잭션을 실행하는데 필요한 메뉴관련 스키마, (iv) 각 화면의 사용자 인터페이스를 통합적으로 관리하기 위해 사용자 컨트롤을 직접 리파지토리에서 관리하게 되는 화면 구성 관련 스키마, (v) 실제 각 메소드들이 데이터베이스와 연동되는 프로그램 구성 관련 스키마가 그것이다.

업무 분류 스키마의 리파지토리 대분류 정보 개체 (rep_gcode) 는 업무의 대분류를 의미하는 것이며 rep_mcode 개체는 업무의 중분류를 의미한다. rep_class는 업무의 소분류를 의미하는데 모든 어플리케이션들은 이 소분류에 속하게 된다. 이 소분류는 각 화면 관련 개체들과 트랜

잭션 (rep_transaction), 프로그램 (rep_program) 등의 개체 등과 관계를 갖게 된다.

소분류는 폼 (rep_form) 을 가지고 있으며 각 폼은 rep_formpbo (FormPBO, Form Program Before Output) 와 rep_formpai (FormPAI, Form Program After Input) 로 구성된 화면 관련 제반 개체들과 관계가 있다. 예를 들어, 특정 부서 정보 등록 시 입력결과를 보여주는 화면이 FormPAI이며, 프린트 하기 전에 프린터 방식설정을 위하여 화면에 보여주는 화면이 FormPBO이다. 특정 화면은 다음 화 면이 어떤 화면인지를 알고 있어야 하므로 일진관계 (Unary Relationship)를 가지고 있다. 모든 어플리케이션의 실행은 트랜잭션을 통해서만 가능하다. 따라서 특정 트랜잭션은 모두 해당 어플리케이션이 시작되는 초기 화면을 가지고 있다. 트랜잭션 및 메뉴 관련 스키마에서는 트랜잭션이 실행되기 위해 메뉴 (rep_menu) 와 트랜잭션 간의 관계가 필요함을 보인다. 단 메뉴는 하위메뉴 (rep_submenu) 를 가지고 있으므로 일진관계를 가지고 있다.

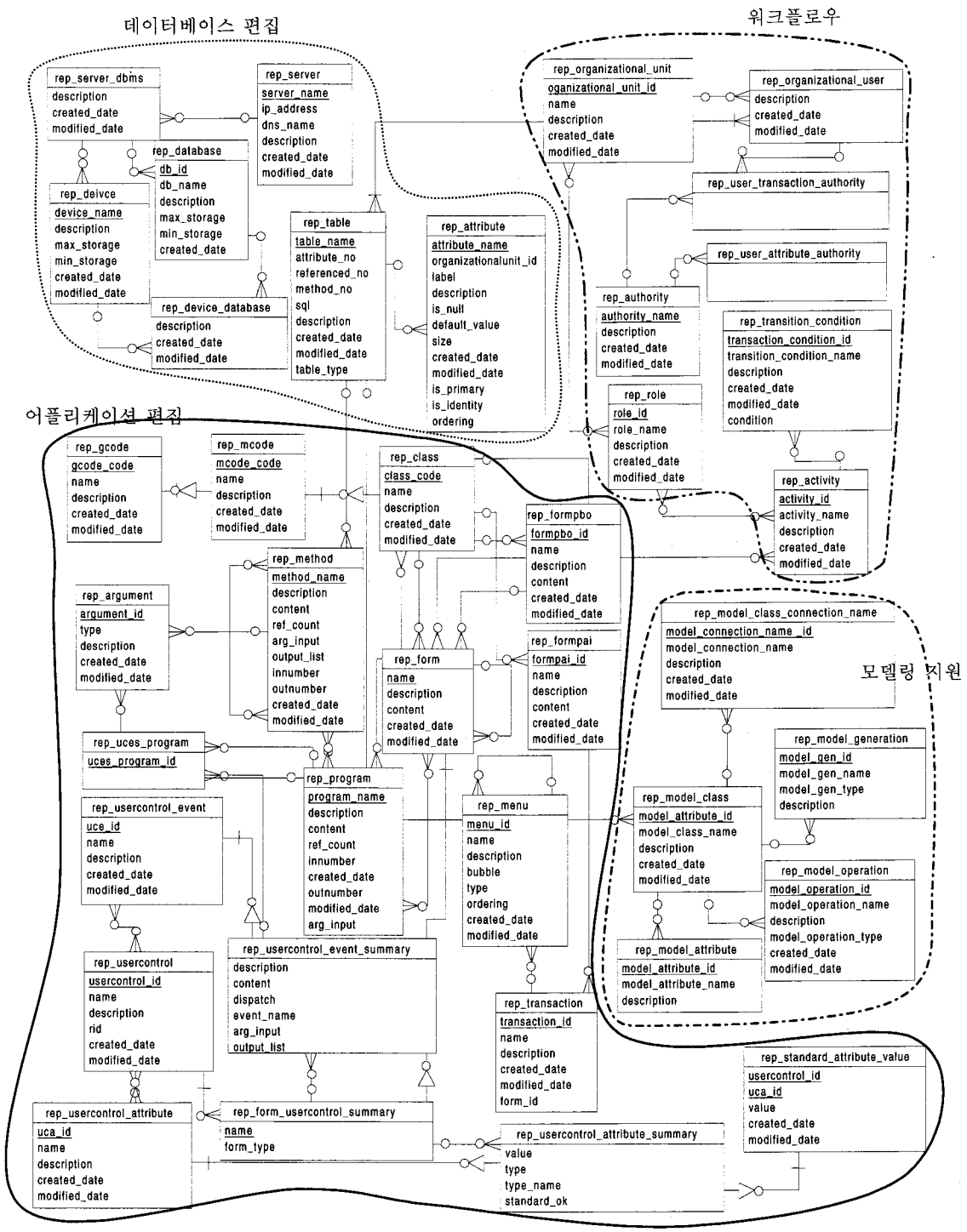
화면 구성 스키마에서는 각 화면의 사용자 인터페이스를 통합적으로 관리하기 위해 사용자 컨트롤을 직접 리파지토리에서 관리한다. 각 사용자 컨트롤 및 해당 사건 (Event) 과 속성 (Attribute)을 리파지토리에서 관리하며 특정 화면에서 사용되는 사용자 컨트롤 인스턴스 역시 관리하고 있다. 사용자 컨트롤의 표준정보화를 위해서는 별도의 테이블이 요구된다. 특정 컨트롤의 특정 사건이 호출하는 프로그램 정보도 관리한다. 예를 들어, 화면에서 자주 사용되는 버튼 (Button)을 사용자 컨트롤이라고 했을 때, 버튼을 누르는 것이 하나의 사건이며, 버튼의 색깔, 위치, 크기, 이름, 글자의 크기, 글자의 색상 등이 속성이다.

프로그램 구성관련 스키마에서 각 메소드 (rep_method) 는 테이블을 사용하는 것과 그렇지 않은 것으로 양분된다. 각 메소드는 여러 개의 메소드의 조합으로 구성할 수 도 있다. 프로그램은 실제 실행되는 응용 논리와 관련 있는 부분이다. 프로그램 역시 다양한 메소드로 구성될 수 있다. 변수 개체 (rep_argument) 는 각 프로그램과 메소드가 사용하는 입출력에 관련된 변수들을 저장한다. 예를 들어, '사번에 의한 사원검색' 메소드는 사번을 입력 변수로 하여 사원 테이블을 사용하여 사원명을 출력 변수로 사용하는 메소드이며, 사원검색 프로그램에서는 이 메소드에 변수를 통하여 호출하여 사용할 수 있다.

다음으로 데이터베이스 편집 분야 (<그림 5>의 좌상부) 는 3부문으로 대별된다. 즉, (i) 해당 데이터 타입 정보를 관리하기 위한 데이터 타입 관련 스키마, (ii) 이기종 데이터베이스를 위한 분산 환경 관련 스키마, (iii) 해당 테이블과 관련 속성 등의 정보를 관리하는 테이블 관련 스키마이다. 데이터의 타입정보를 관리하기 위해 필수 적인 것은 해당 데이터 타입의 구성 방식을 어떻게 하느냐 하는 것이다. 여기서는 추가적인 데이터 타입은 기본 데이터 타입 정보들의 조합으로 가능하게 함으로써 다양한 데이터 타입을 추가할 수 있게 하였다. 특정 DBMS마다 데이터 타입의 정의가 다르므로 이를 위해 DBMS의 데이터 타입 (rep_dbms_data_type)이라는 개체를 사용하여 내부적으로 특정 DBMS에 맞게 변환될 수 있게 하였다.

분산환경 관련 스키마는 분산 데이터베이스, 이기종 DBMS, 그리고 이기종 데이터베이스를 위한 메타데이터이다. 서버 (rep_server)는 해당 서버의 IP 주소 (Internet Protocol Address) 및 도메인 이름 (Domain Name)을 관리하며, 디바이스 (rep_device) 와 데이터베이스 (rep_database)는 해당 개체의 최대 용량 등의 정보를 보여준다.

테이블 관련 스키마는 테이블과 관련한 개체 (rep_table) 를 보여 주고 있다. 테이블과 관련한 정보로는 해당 테이블에 대한 SQL, 인텍스, 키, 속성의 개수, 테이블의 유형 등의 정보가 저장된다. rep_reference는 외래키에 해당하는 정보를 관리하며 키와 관련한 정보는 rep_attribute와



<그림 5> 리파지토리 제 2 계층 스키마

rep_table 모두와 관계를 갖는다. rep_attribute 개체에는 기본값 여부, 널 (Null) 값 허용여부, 크기 등의 기본적인 제약조건을 정의한다. 그러나, 속성이 복잡하거나 특수한 제약조건을 가질 수 있어 rep_attribute_constraint 라는 제약조건 정보 관리를 별도로 한다.

워크플로우 (<그림 5>의 우상부) 에 관련한 스키마는 전체 조직을 구성하는 조직 단위인 rep_organizational_unit, 데이터베이스의 사용자로 조직의 구성원인 rep_organizational_db_user, 보안 및 권한 정의에 대한 정보를 가지고 있는 rep_authority, 조직에 속한 사용자 정보인 rep_actor 등을 이용한 보안 및 조직 관련 부분을 포함한다. 일반 데이터베이스의 메타데이터 스키마와는 상이하며 사용 권한 및 보안을 위해 각 사용자 마다 rep_authority 권한을 부여한다. 특정 조직에 속한 사용자가 특정한 행위를 허용 받을 수 있게 하는 기능도 가능하다.

모델링 지원 스키마 (<그림 5>의 우하부) 는 재사용성을 목적으로 UML의 기본 모델인 객체 모델을 중심으로 작성되었다. 객체지향 설계에서 사용되는 클래스를 어플리케이션 분야에서 사용되는 소분류 (rep_class) 와 차이를 두기 위해서 rep_model_class라는 이름으로 정의하였다. 각 rep_model_class는 기본적으로 rep_model_attribute와 rep_model_operation을 갖는다. 그리고 각 모델 클래스 간의 관계를 위해 rep_model_class_connection과 rep_model_generalization 개체를 두었다. rep_model_class_connection은 rep_model_generalization을 제외한 여타의 일반적인 관계를 정의한 것이고 rep_model_generalization은 상속 관계를 정의하는 개체이다.

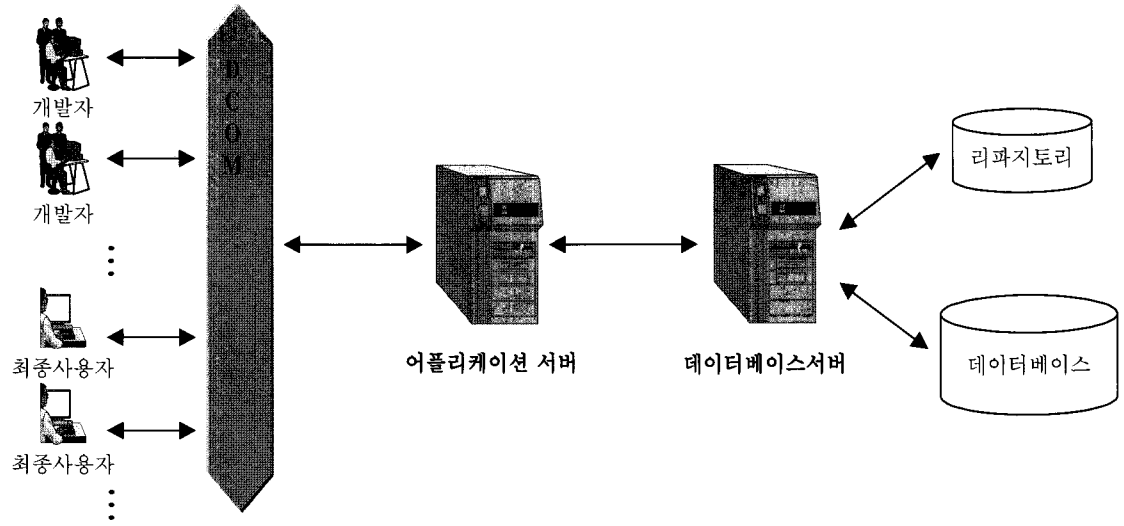
이상 본 논문에서 제안된 리파지토리 스키마와 비교할 때, SAP R/3 리파지토리 스키마는 다국화를 지원하는 모듈의 구성으로 스키마가 부분적으로 중복되었을 것이다. SAP R/3의 테이블 수는 8000여개로 알려져 있다 [Bancroft et al., 1997]. 이에 비해 본 리파지토리는 국내 기업의 표준 정보 시스템 구축이라는 특수 목적을 지향하므로 중복의 제거가 가능하였다.

Ⅵ. ERP 프로토타입 시스템

본 절에서는 위에서 제안된 리파지토리 스키마를 근간으로 한 ERP 시스템 프로토타입 구성을 설명한다. 3단계 (3-Tier) 클라이언트 서버 방식으로 구성된 ERP 시스템의 클라이언트측 운영시스템은 Windows 95를 이용하고, 서버에는 Windows NT를 이용하였다. 클라이언트와 서버의 분산 미들웨어로 양자가 기본적으로 지원하는 DCOM을 활용하였다. ERP 시스템의 구성도는 <그림 6>과 같으며, 시스템 구성요소별 이용 제품 목록은 <표 6>과 같다.

클라이언트는 개발자들이 사용하는 개발지원 기능과 최종사용자들이 사용하는 업무지원 기능을 제공한다. 개발지원 기능은 어플리케이션을 설계하는 어플리케이션 편집기 (<그림 7>), 데이터베이스를 설계하는 테이터베이스 편집기 (<그림 8>), 그리고 프로세스를 설계하는 워크플로우 편집기를 가지고 있으며 C++를 이용하여 개발되었다. 그림에서 볼 수 있듯이, 화면의 왼쪽에 탐색기능을 제공하는 브라우져 기능이 있고, 오른쪽에는 실화면이 나오도록 구성되어 있다. 업무지원 부문은 어플리케이션 서버에서 런-타임 (Run-Time) 시 제공하는 프로그램을 클라이언트에서 실행할 수 있도록 하기 위하여 VB 스크립트 (Visual Basic Scripts)를 사용하여 개발되었다.

DCOM은 클라이언트에서 실행되는 어플리케이션이 서버에서 실행되는 컴포넌트에 접근할 수 있는 분산 기능을 지원한다. DCOM을 사용함으로써 클라이언트와 서버의 어플리케이션을 분산할 수 있다. 어플리케이션 서버는 클라이언트의 요구 처리를 위한 리파지토리 관리 기능과 어플리케이션 관리 기능을 제공하며, C++를 이용하여 개발하였다. 데이터베이스 서버는 리파지토리와 어플리케이션용 데이터베이스를 저



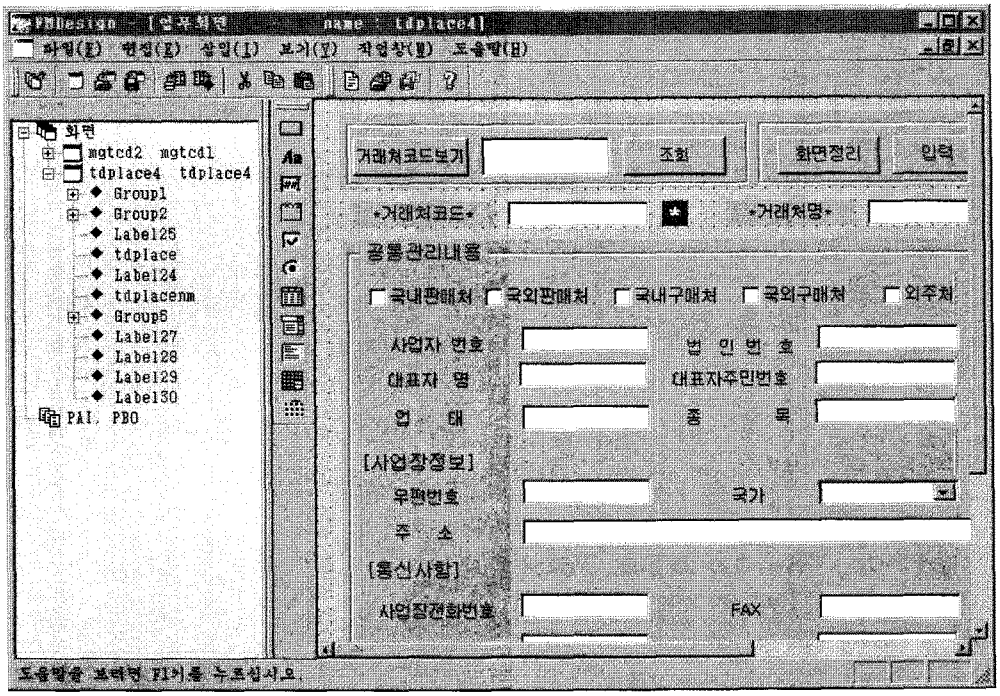
<그림 6> ERP 시스템 구성도

장한다. DBMS로는 Oracle Workgroup Server 7.3과 Microsoft SQL Server 6.5를 이용하였으며, 데이터베이스용 미들웨어로 Microsoft Transaction Server 2.0 [Mills, 1998] 을 활용하였다. Transaction Server는 서버 응용 프로그램을 개발, 배치, 관리하기 위한 트랜잭션 처리 시스템이다.

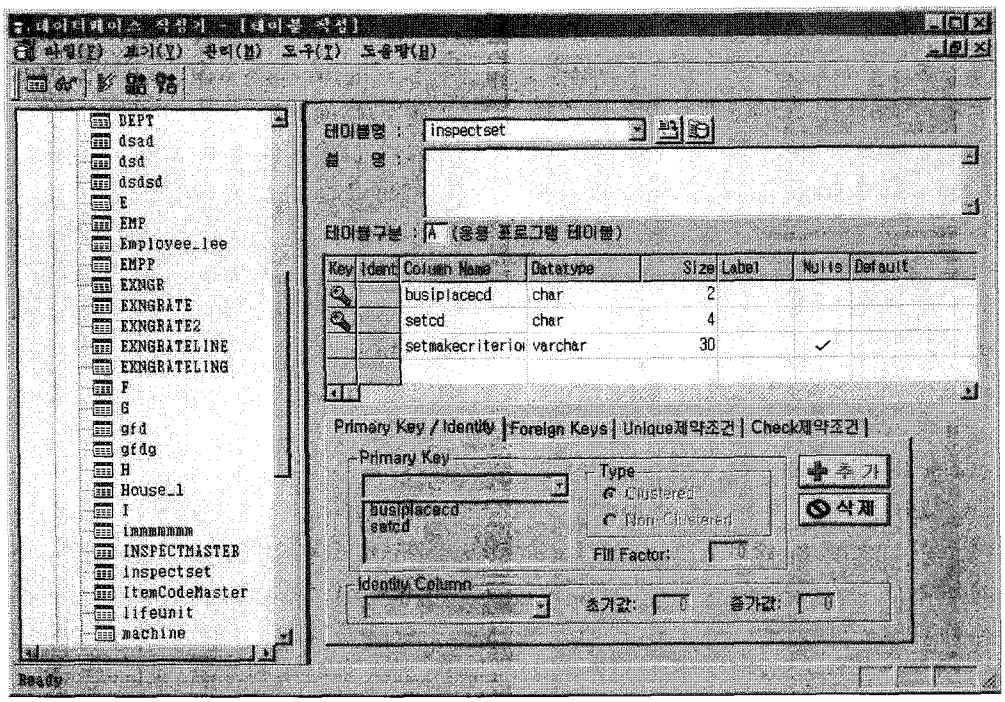
클라이언트의 기능이 수행되기 위해서는 서버와 DCOM을 이용한 통신이 요구되며, 그 과정은 다음과 같다. 클라이언트의 사용자는 어플리케이션 편집기, 데이터베이스 편집기, 워크플로우 편집기 등을 선택하여 작업을 수행하며, 각 편집기는 DCOM을 통하여 어플리케이션 서버에 서비스를 요청한다. 그 후 어플리케이션 서버는 데이터베이스 서버에 있는 리파지토리를 미들웨어인 Transaction Server를 통하여 접근한다. 이에 따라 필요한 메타 데이터를 참조하여 요구사항을 처리한다. 본 연구에서 제안된 리파지토리에 근거하여 통합적인 ERP 시스템이 구축되었으며, 이를 통하여 기업 표준정보시스템의 개발과 유지 보수가 가능하다.

<표 6> 시스템 구성요소별 이용 제품 목록

구성요소	기 능	구성소프트웨어	운영 체계	비 고
클라이언트	개발지원	어플리케이션 편집기 워크플로우 편집기 데이베이스 편집기	Windows 95	C++ 사용
	업무지원	비즈니스 어플리케이션	Windows 95	VB 스크립트 사용
미들웨어	분산 컴포넌트 지원	DCOM	Windows 95 Windows NT	—
어플리케이션 서버	리파지토리 관리 어플리케이션 관리	ERP 서버 모듈	Windows NT	C++ 사용
데이터베이스서버	트랜잭션 관리	MS Transaction Server 2.0	Windows NT	—
	리파지토리 데이터베이스	Oracle Workgroup Server 7.3 MS Sql Server 6.5	Windows NT	—



<그림 7> 어플리케이션 편집기 화면



<그림 8> 데이터베이스 편집기 화면

Ⅶ. 결 론

리파지토리를 이용한 시스템 개발의 장점은 특정 개발 환경과 개발 도구에 독립적이며 유연하여 통합적인 개발이 가능하다는 점이다. 본 논문은 ERP로 대표되는 전사적 자원 관리를 국내기업에 적용하기 위해 IRDS표준에 따라 리파지토리를 설계하였다. 본 리파지토리는 기업 지식의 구조화된 틀로서 기업 자산의 기본 인프라라고 할 수 있겠다. 또한, 본 연구의 성과를 지식 경영측면에서 지식 리파지토리로 확장이 가능하다 [이희석 외, 1998].

본 연구를 기반으로 향후 수행되어야 할 연구로는 다음과 같다.

첫째, 독립성을 좀 더 보장하기 위하여 객체 기반 인프라로 본 리파지토리를 확장하는 것이다. 우선, IRDS기반의 객체 리파지토리 가능성 여부를 판단하여야 할 것이다. 둘째, ERP 리파지토리를 기업 지식 리파지토리로 발전시키는 연구이다. ERP 리파지토리에 내재된 지식은 주로 베스트 프랙티스 (Best Practice) 에 관련된 지식으로 이외의 지식 자산 (Intellectual Capital) [Edvinsson & Malone, 1997] 의 포함이 관건이다.

〈참 고 문 헌〉

[1] 이희석, 중소기업형 표준 정보 시스템 개발 리파지토리 구조화 연구 중간 연구 보고서, 한국전자통신연구원, 1998.

[2] 이희석, 최병구, 장재경, 홍순근, "창조적 지식경영 아키텍쳐," *한국정보전략학회지*, 1998.

[3] 양해술, "I-CASE의 도구 통합과 방법," *정보과학회지*, 12(2), 1994.

[4] Altomare, D., Carulli, M., Casanova, P. and Limongelli, D., "A Prototype for the Integration of Information Resource Dictionary System and PCTE," *Computer Standards & Interface*, 2,1989.

[5] Asrafi, N. and Kuilboer, J., "The Information Repository: A Tool for Metadata Management," *Journal of Database Management*, Spring, 1995, pp. 3-11.

[6] American National Standards Institute, Inc., *Information Resource Dictionary System (IRDS) Standard x13.138-1988*, New York: ANSI, 1989.

[7] Bancroft, N.H., Seip, H. and Sprengel, A., *Implementing SAP R/3, Manning Publications* Co., 1997.

[8] Bernstein, P.A. and Umeshwar D., "An Overview of Repository Technology," *International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers, San Francisco, 1994, pp. 705-713.

[9] Bragen, M.A., "Repository in Action," *Database Management*, 2(9), 1992.

[10] Bodoloi, B., Sircar, S. and Lakhanpal, B., "Desirable Characteristics of Information Resource Dictionary Systems," *Journal of Database Management*, Spring, 1998, pp. 3-15.

[11] Edvinsson, L. and Malone, M.S., *Intellectual Capital*, HarperCollins Publishers, Inc., 1997.

[12] Electronic Industries Association, CDIF Transfer Format General Rules for Syntaxes and Encoding, Washington, DC: EDA, 1994.

[13] European Computer Manufacturers Association, *Portable Common Tool Environment (PCTE) Abstract* Specification, Standard ECMA-149, Geneva, 1990.

[14] Hazzah, A., "Data Dictionaries: Paths to Standard," *Database Programming & Design*, 2(8), 1989.

[15] Kiely, D., "Are Components the Future of Software?" *Computer*, 3(2), February 1998, pp. 10-11.

[16] Hakala, J., Husby, O. and Warwick, T.K., Framework and Dublin Core Set Provide a Comprehensive Infrastructure for Network Resource Description, <http://www.bibsys.no/warwick.html>, 1996.

[17] MacGauhey, R. E. and Gibson, M., "The Repository/Encyclopedia: Essential to Information Engineering and Fully integrated CASE," *Journal of Systems Management*, 44(3), 1993.

[18] Martin, J., Information Engineering: A Trilogy, Vol.1-3.Englewood Cliffs, NJ: Prentice-Hall, 1989.

[19] Mills A., *Using Microsoft Transaction Server 2.0 with COM Transaction Integrator 1.0*, Microsoft Corporation, March 1998.

[20] Moriarty, T., "Are You Ready for a Repository?" Database Programming & Design, 1990.

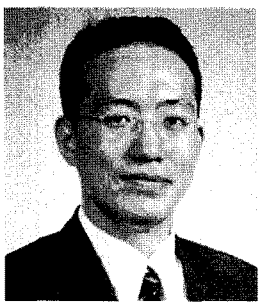
[21] Object Management Group, "Unified Modeling Language Version1.1," <http://www.rational.com>, September 1997.

[22] Orfali, R., Harkey, D. and Edwards, J., *The Essential Client/Server Survival Guide*, John Wiley & Sons, Inc., 1996.

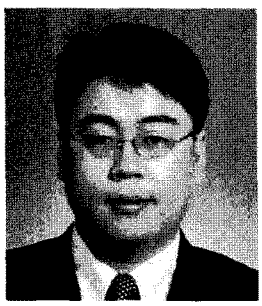
[23] Tannenbaum, A., *Implementing a Corporate Repository: The Model Meets Reality*, John Wiley & Sons, Inc., 1994.

◈ 이 논문은 1998년 9월 21일 접수하여 1999년 1월 25일 게재확정되었습니다.

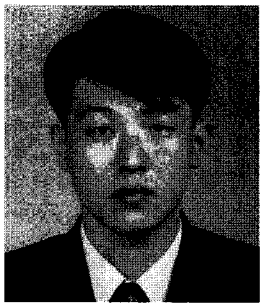
◆ 저자소개 ◆



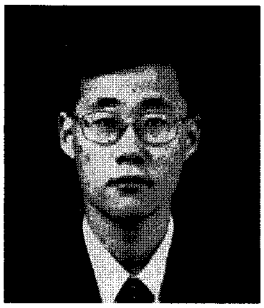
이희석 (Heeseok Lee)
현재 KAIST 테크노경영대학원 부교수로 재직 중이다. 서울대학교 공과대학을 졸업하고 KAIST에서 석사학위를 University of Arizona에서 경영정보공학 박사 학위를 취득하였다. 주요관심분야는 ERP, 지식관리, 데이터웨어하우스, 리파지토리, 비지니스 엔지니어링 등이다.



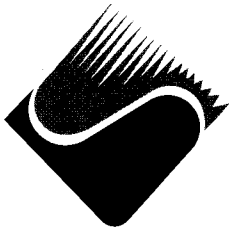
이 제 (Jay Lee)
현재 KAIST 박사과정에 재학 중이다. 서강대학교 경영학과를 졸업하고, KAIST 경영공학 석사학위를 취득하였다. 주요관심분야는 객체지향 설계, 리파지토리 등이다.



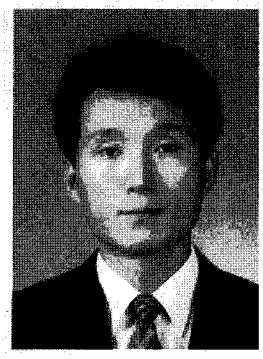
이충석 (Choongseok Lee)
현재 KAIST 박사과정에 재학 중이다. KAIST 경영과학과를 졸업하고, KAIST 경영공학 석사학위를 취득하였다. 주요관심분야는 가상기업, 인트라넷, 리파지토리 등이다.



조창래 (Changrae Cho)
현재 삼성카드주식회사 마케팅팀 데이터웨어하우스 파트에서 데이터마이닝 담당자로 재직 중이다. 고려대학교 통계학과를 졸업하고, KAIST 경영공학 석사확위를 취득하였다. 주요관심분야는 지식경영시스템, 전자상거래구현, ERP, 데이터마이닝, 데이터웨어하우스 등이다.



손주찬 (Joochan Sohn)
현재 한국전자통신연구원 전자상거래연구부 선임연구원으로 재직 중이다. 한국외국어대학교 서반아어과를 졸업하고, 한국외국어대학교 경영정보대학원에서 경영정보학 석사를 취득하였다. 주요관심분야는 ERP, 데이터 모델링, 소프트웨어 엔지니어링 등이다.



백종명 (Jongmyung Baik)
현재 한국전자통신연구원 전자상거래연구부 정보통합연구팀장으로 재직 중이다. 고려대학교 산업공학과를 졸업하였다. 주요관심분야는 ERP, CALS/EC, CIM 등이다.

부록 1: ERP용 리파지토리 시스템의 IRDS 제 2 계층 (IRD Schema) 개체 목록

개체	설명	개체	설명
rep_gcode	대분류	rep_table	테이블과 뷰
rep_mcode	중분류	rep_database	데이터베이스
rep_class	일종의 소분류. 대분류와 중분류 하부에 존재	rep_attribute	데이블 속성
rep_usercontrol_event	사용자 컨트롤 사건	rep_server	서버
rep_usercontrol_event _ summary	사용자 컨트롤 사건의 실제 인스턴스 값	rep_server _dbms	서버와 DBMS와의 관계
rep_aregument	프로그램과 메소드에서 사용되는 변수	rep_device	데이터베이스에 기반이 되는 디바이스
rep_uces_program	사용자 컨트롤과 프로그램의 관계	rep_role	프로세스에서 사용자의 역할
rep_transaction	트랜잭션	rep_authority	보안 및 권한
rep_form	어플리케이션 편집기에서 생성되는 폼 (화면)	rep_organizational_unit	조직 부서
rep_form_usecontrol_ summary	폼과 사용자 컨트롤과의 관계	rep_activity	프로세스의 단위 업무
rep_formpai	입력 후 처리 폼	rep_user_transaction_ authority	트랜잭션에 대한 사용자 권한
rep_formpbo	출력 전 처리 폼	rep_user_attribute_ authority	테이블 속성에 대한 사용자 권한
rep_menu	프로그램을 실행시키기 위한 메뉴	rep_transaction_ condition	단위 업무의 트랜잭션 조건
rep_method	프로그램의 가장 기초적인 단위인 메소드	rep_organizational_user	부서의 사용자
rep_program	클래스에서 실제로 사용되는 프로그램 개념	rep__model_class	객체지향 모델의 클래스
rep_standard_attribut e_ value	표준 속성 값	rep_model_operation	클래스의 오퍼레이션
rep_usercontrol	사용자 컨트롤	rep_model_generation	클래스의 일반화
rep_usercontrol_ attribute	사용자 컨트롤의 속성	rep_model_class_ connection	클래스간의 관계
rep_usercontrol_ attribute_summary	사용자 컨트롤 속성 값	rep_model_attribute	클래스의 속성
rep_device_ database	물리적 디바이스와 데이터베이스의 관계		