The Journal of Korean
Institute of CALS/EC
Vol. 4, No. 1, June. 1999

분산객체 환경에서의 워크플로우 관리를 위한 정보저장소*

염태진**, 박재형**, 리 자**, 김기봉***, 진성일**

A Repository for Workflow Management on Distributed Object Environment

Tai-Jin Yeom, Jae-Hyung Park, Lizi, Ki-Bong Kim, Seong-Il Jin

Abstract

Workflow management system provides automation of job processes by maintaining shareability on information about various job process schedules and persons related to those schedules. Existing workflow management systems use file or database to store the information generated in those systems. However, file or database system could manage only non-complicated information for the workflow but not the information resources of an enterprise which is complicated and of various formats. Therefore, we need a data management system that could control those information resources. This system should manage the data which are distributed at several places geographically. Information Repository could meet those requirements. Information Repository may integrate, store and manage information resources requested by application systems. We have an international standard for the information repository, Information Resources Dictionary System(IRDS). The IRDS, however, does not support distributed environment. In this paper, we design and implement an information repository based on IRDS that may be operated in distributed environment. We verify that this information repository is more effective and is more effective than any other file or database system.

---

* 본 논문은 충남대학교 부설 소프트웨어연구센터와 국방과학연구소에 의해 지원되었음.

** 충남대학교 컴퓨터과학과

*** 대전보건대학 전산정보처리과

1. 서론

워크플로우 관리 시스템은 다양한 작업 활동의 순서와 활동의 순서에 관련된 사람간의 정보 기술 자원을 공유함으로써 업무 프로세스의 자동화를 제공하는 시스템이다[David, 1994]. 워크플로우 관리 시스템은 프로세스의 자동화를 위해 프로세스 및 관련 업무 데이터, 태스크 간의 전이를 나타내는 트랜지션 데이터, 프로세스가 사용하는 워크플로우 관련 데이터, 응용 프로그램 관련 데이터 등의 여러 정보들을 사용하며, 이들을 산출해 낸다. 이러한 시스템에서 사용되고 산출되는 정보를 저장하기 위해 기존의 시스템에서는 파일이나 데이터베이스 시스템을 사용하였다[David, 1994]. 그러나 파일이나 데이터베이스 시스템은 워크플로우를 위한 물리적인 데이터와 같은 일차적인 정보 외에는 관리하기 어려웠고, 이외의 각 데이터에 대한 스키마 데이터나 그들 사이의 관계, 전 조직체의 정보 자원과 같은 복잡하고 다양한 정보의 관리를 위한 시스템이 필요하게 되었다. 또한 이러한 데이터들이 지역적으로 분산되어 있는 경우 분산 환경에서 워크플로우와 연동하여 운영 가능한 정보 저장 시스템이 필요하게 되었는데, 이러한 역할을 할 수 있는 시스템이 바로 정보저장소다[김기봉 1998; 김기봉 1995]. 정보저장소는 복잡한 정보 처리 환경에서 보다 효율적인 정보자원의 관리를 위해, 각 응용 시스템들이 요구하고 사용하고 있는 모든 정보자원을 통합, 저장, 관리할 수 있는 시스템을 말한다[Mark, 1991; Mark, 1992].

현재 정보저장소의 구현을 위한 표준으로 IRDS(Information Resource Dictionary System)를 비롯하여 CDIF(Case Data Interchange Format), PCTE(Portable Common Tool Environment), ATIS(A Tool Integration Standard) 등이 있으나 이는 분산 객체 환경을 지원하지 않고 있고, 최근에는 OMG 등에서 분산객체 환경하에서의 정보저장소 표준에 대한 연구가 계속되고 있으나 이를 제품으로 만든 정보저장소는 거의 없으며, 또한 CORBA와 같은 진정한 분산 객체 환경을 지원하지 않고 단순한 통신 수단으로 분산 환경을 지원하는 제품정도이다. 따라서 본 논문에서는 워크플로우 관리를 위한 정보저장소 구현을 위하여 정보저장소의 대표적인 표준이라고 할 수 있는 IRDS 표준에 입각하여 이를 분산객체 환경에서 운영될 수 있도록 정보저장소를 설계 및 구현하였다.

이를 위해 워크플로우 관리를 위한 분산 환경에 맞는 정보저장소의 구조를 정립하고, 이에 적합하도록 각 서비스 인터페이스 및 사용자 인터페이스, 메타테이블을 설계 및 구현하였다. 정보저장소를 워크플로우 관리 시스템에서 활용하기 위해서 워크플로우 관련 데이터의 수집, 분석 및 메타모델링을 하였다. 또한 워크플로우 정보를 조직적으로 관리하기 위해서 기존의 관리 방식과는 달리 일차적인 정보에 대한 메타데이터들을 두 단계에 걸쳐 정보저장소의 표준에 준하여 저장하였다. 메타데이터 단계에서는 워크플로우의 물리적인 데이터에 대한 스키마를, 메타스키마 단계에서는 메타데이터에 대한 타입을 제공하는 데이터들을 저장하여 관리하도록 하였다. 또한 예제를 통하여 구현된 시스템이 워크플로우 관리를 위해 어떠한 식으로 정보를 관리하는 지와 그

장점에 대해 기술하고, 정보저장소를 이용할 경우 다른 시스템을 이용할 경우보다 더욱 효과적인 이유를 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서는 연구배경으로써 워크플로우 관리 기술과 분산 객체 기술, 정보저장소 기술에 대해 언급하고 3장에서는 워크플로우 관리를 위한 분산 환경에서의 정보저장소 구현에 대해 기술하고, 4장에서는 워크플로우 관리를 위해 구현된 시스템을 적용해 보고 그 활용도에 대해 기술하였다. 마지막으로 5장에서는 결론 및 향후 연구 방향에 대해 언급하였다.

2. 배경

워크플로우 관리를 위해 분산 환경에서 운용 가능한 정보저장소 구현을 위해서는 워크플로우 관리 기술을 포함하여 분산 객체 기술, 정보저장소 기술이 필요하다.

2.1 워크플로우

워크플로우란 전반적인 비즈니스 목표를 성취하기 위한 규칙의 집합을 규정함으로써 워크플로우의 참여자 사이의 전달되는 문서, 정보, 태스크의 진행의 자동화를 말하며, 워크플로우 관리 시스템이란 다양한 작업 활동의 순서와 활동의 순서에 관련된 사람 사이의 정보 기술 자원의 호출을 관리함으로써 업무 프로세스의 자동화를 제공하는 시스템을 말한다 [David, 1994; Nortel, 1997].

이러한 워크플로우 구현을 위한 표준의 작업을 하는 단체가 WfMC(Workflow Manage-ment Coalition)이다. 본 논문에서 기술하는 워크플로우 참조 모델은 WfMC의 참조 모델을 근간으로 하고 있다. WfMC에서 제시하고 있는 워크플로우를 위한 일반적인 구조는 워크플로우를 해결하는 워크플로우 엔진, 프로세스 정의(Process Definition)를 내리는 프로세스 정의 도구, 데이터를 저장하는 데이터 저장 시스템, 그리고 기타 사용자 인터페이스나 사용자 응용프로그램 등으로 구성되어 있다. 워크플로우 엔진은 프로세스를 해석하고, 프로세스들을 제어하며, 활동(activity)들의 순서를 정의하고, 정의된 활동들을 사용자 업무 요소(work item)에 추가 시켜 주며, 필요한 응용프로그램을 가동시켜준다. 워크플로우 엔진은 다양한 프로세스의 실행을 관리하는 하나 또는 그 이상의 워크플로우 엔진이 될 수 있다. 프로세스 정의 도구는 워크플로우 시스템이 사용하고 접근할 수 있는 프로세스 형태를 만드는데 사용된다. 사용자 인터페이스는 사용자와의 의사 소통과 제어를 담당하고, 사용자 응용프로그램은 워크플로우의 진행 중 필요한 메일 매니저나 워드 등과 같은 프로그램이다. 또한 워크플로우에 포함되는 정보의 형태로는 프로세스 정의 정보, 조직체(Organization)와 규칙(Role) 정보, 워크플로우 제어 데이터(Workflow Control Data), 워크플로우 관련 데이터(Workflow Relevant Data), 작업리스트(Work List) 등이 있다. 이러한 데이터들을 관리하기 위해 기존의 워크플로우 관리 시스템들은 파일이나 단순 데이터베이스를 이용하였다. 그러나 이러한 파일이나 데이터베이스를 이용하는 경우에는 단순히 응용 프로그램을 위한 단순 데이터의 관리 차원에 국한되어 있

어 워크플로우 운영 정책, 워크플로우 절차, 계획 등의 복합적인 정보자원을 관리 하기는 미흡하다. 따라서 워크플로우 관리 시스템에서 발생하는 데이터들을 정보저장소 내에 단계별 또는 계층적 구조화를 통해 효과적으로 관리를 할 수 있는 정보저장소를 사용하면 정보자원의 가용성을 높일 수 있으며, 다양한 정보의 형태를 저장할 수 있고, 다른 정보와의 연관성 등도 파악할 수 있다.

2.2 분산 객체 환경

이는 개발된 소프트웨어가 분산 환경에서 서로 다른 시스템들과 쉽게 통합될 수 있는 환경이라 할 수 있다. 즉, 분산 객체 환경이란 생산성 있는 각 응용 프로그램을 작성하여, 파일이나 데이터베이스와 같은 저장 매체를 통해 분산 저장하여 정보를 공유하고, 서로 다른 운영체제와 네트워크 환경 등의 이종의 환경에서 작동 가능한 클라이언트/서버 환경을 말한다[Randy, 1996; Thomas, 1995]. 이러한 분산 환경을 구축하기 위해서는 기존의 C 언어와 같이 논리(logic)의 흐름에 따라 프로그램을 작성하는 방식으로는 이와 같은 요구사항을 해결하기 어려우며, 또한 이종의 분산 환경에서 다양한 형태의 응용 프로그램을 통합하기 위해서는 일정한 결합 방법이 필요하다. 이런 문제를 해결하기 위한 표준 중의 하나가 CORBA(Common Object Request Broker)이다. CORBA는 OMG(Object Management Group)에서 제안한 표준으로 이기종간의 상이한 분산 환경에서의 상호 동작성을 제공하고 이질 객체 시스템간의 완전한 상호 연결을 위하여 제안된 것으로 객체간에 서비스를 요청하고 그것에 대한 응답에 대한 투명한 메커니즘을 제공한다. CORBA의 기본 구조는 클라이언트와 서버, ORB(Object Request Broker)로 구성되어 있다. 클라이언트는 동적 호출 API를 사용하여 컴파일된 IDL 스텁(stub)을 거치거나 처리 중에 서비스를 정적으로 불러낸다. 서버는 정적 골격(skeleton)을 사용하여 클라이언트 요구를 객체 수행 프로그램에 연결하거나, 새로운 동적 골격 인터페이스를 사용하여 런타임으로 보내기도 한다[Randy, 1996].

2.3 정보저장소

정보저장소는 소프트웨어 개발의 각 단계에서 사용되고 생산되어지는 모든 정보와 시스템 요소들을 저장할 뿐만 아니라 이러한 정보들의 상호관계와 이들 정보들의 유효성을 증명하고 사용하는 규칙도 포함하여, 시스템의 이해와 변화 관리를 지원하는 시스템으로 감사 관리, 컴퓨터 응용, 형상 관리, 자료 관리, 데이터베이스 관리, 통신망 관리, 프로젝트 관리, 시스템 공학, 워크플로우 관리(Workflow Management) 등에서 적용되어 사용될 수 있다[Mark, 1992; 김기봉, 1998].

국제 표준화 기구(ISO)에서 제정된 정보저장소 구현을 위한 표준안으로 IRDS가 있다. IRDS 표준안에는 IRDS 기본 기능, IRDS 모델링 규칙, IRDS 데이블, IRDS 추상 자료 구조 및 서비스 인터페이스 등의 IRDS에 대한 개념과 구현 방안을 위한 표준안을 제시하고 있다. 그러나 IRDS를 포함한 현재까지의 정보저장소는 중앙 집중적 방식으로 분산 환경에

서 여러 시스템들과 연계하여 사용하기가 어렵다.

따라서 분산 객체 환경하에서 발생하는 모든 정보자원을 효율적으로 관리하고 여러 시스템들이 연계되어 정보자원을 공유하며 사용하기 위해서는 분산 객체 정보저장소가 필요하다.

3. 분산환경에서의 워크플로우 관리를 위한 정보저장소 구현

ISO의 IRDS 표준안에는 정보의 자료 참조, 메타데이터/메타스키마 조작, 버전제어, 보안을 비롯한 기타 부가적인 기능이 많이 포함되어 있으나, 본 논문에서의 구현범위를 자료참조 및 메타데이터/메타스키마 조작에 필요한 핵심이 되는 서비스 인터페이스와 이의 원활한 사용을 위한 사용자 인터페이스만을 분산환경에서 운영 가능하도록 구현하였다.

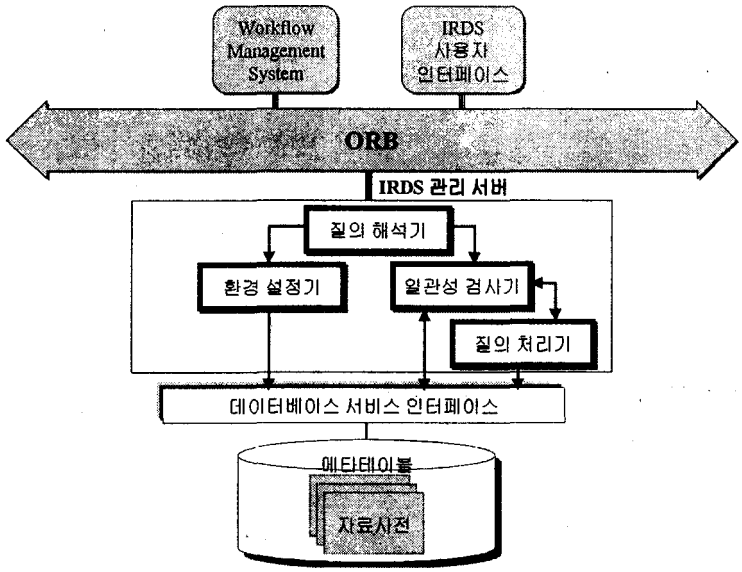
3.1 시스템 구현 환경

정보저장소의 구현을 위해서는 크게 데이터베이스, 클라이언트 시스템, 서버 시스템, 상용 ORB 제품과 구현을 위한 프로그래밍 툴이 필요하다. 클라이언트 시스템과 서버 시스템의 운영체제는 각각 Windows 95와 Windows NT 4.0을 사용하였다. 사용자 인터페이스의 구현 방식은 웹 기반 클라이언트, 윈도우 GUI 기반 클라이언트, 명령어 기반 클라이언트 등이 있는데 그 중 윈도우 GUI 기반 클라이언트로 구현하였다. ISO IRDS의 구현을 위한 데이터베이스 시스템은 객체 지향 데이터베이스가 적합하며, 이를 위해서는 IUS(Informix Universal Server)를 이용하였다[Informix, 1997a; Informix, 1997b; Informix, 1997c]. IUS는 객체-관계형 데이터베이스로 객체와 클래스, 상속 등의 객체 지향 개념을 지원하며 본 논문의 구현을 위한 요구사항을 만족한다. 서버와 클라이언트 프로그램을 구현하기 위한 윈도우 프로그램으로 Visual C++을 사용하였다[곽준기, 1997]. DBMS와 프로그램과의 연동을 위한 방법은 ODBC, ESQL, DBMS에 독립적인 연동 방식 등이 있는데, 여기서는 그 중 IUS에서 제공하는 COI(C++ Object Interface)를 이용하였다[Informix, 1997b]. COI는 IUS에 접근하기 위해 C++에서 사용하는 일종의 라이브러리이다. CORBA를 지원하는 제품으로 현재 널리 상용화 되어 있는 것은 IONA의 Orbix와 Visigenics의 VisiBroker [박재현, 1998]등이 있으며 그 중 Orbix 2.2를 이용하였다[IONA, 1995].

3.2 시스템 구조 및 구현

본 논문에서 분산 객체 환경에서 워크플로우를 위한 정보저장소를 위해 제안한 구조는 <그림 1>과 같다.

전체 구조의 중심은 ORB이다. ORB는 분산 환경을 지원하는 일종의 미들웨어로서 워크플로우 시스템이나 정보저장소 사용자 인터페이스와 같은 정보저장소 클라이언트와 정보저장소 서버와의 통신을 담당한다. 즉, 클라이언트의 질의를 서버에 전달해주는 역할을 담당한다. 워크플로우 관리 시스템은 단일의 워



<그림 1> 분산 환경에서의 워크플로우 관리를 위한 정보저장소 구조

크플로우 처리 엔진으로 자신이 가진 모든 메타데이터를 하나의 데이터베이스처럼 ORB를 통해 정보저장소에 넘겨준다.

정보저장소 관리 서버는 네 개의 부분 즉, 질의 해석기, 질의 처리기, 일관성 검사기, 환경 설정기로 구성되며 데이터베이스 서비스 인터페이스를 통해 메타테이블에 접근한다.

사용자 인터페이스를 통해 질의를 입력받은 클라이언트의 사용자 질의는 ORB를 통해 정보저장소 API를 사용해 서버로 보내진다. 서버는 질의해석기를 통해 API의 질의를 해석하며, 질의 처리기를 통해 질의를 처리한다. 처리 과정에 앞서 일관성 검사기를 통해 질의에 대한 일관성 및 무결성을 계속적으로 검사한다. 일관성 검사는 데이터베이스 서비스 인터페이스를 통해 메타테이블을 검색하면서 이루어진다. 명령어의 일관성과 무결성에 하나라도 결함이 있으면 오류 메시지를 반환하고 그렇지 않으면 주어진 명령어를 데이터베이스 서비스 인터페이스를 통해 메타테이블을 조작하면서 수행한다. 결국 서버는 질의에 대한 오류 번호나 성공 메시지를 사용자에게 반환하게 되는데 사용자 응용 프로그램은 오류번호이면 번호에 해당하는 적당한 메시지를 출력하고 성공적인 질의이면 성공 메시지를 출력한다.

정보저장소의 역할을 수행하기 위한 핵심 부분이 서비스 인터페이스이다. 서비스 인터페이스는 외부 시스템이나 사용자가 정보저장소 시스템에 접근하기 위한 명령어라 할 수 있으며 본 논문에서는 API의 형태로 구현하였다. ISO의 IRDS 표준안을 근간으로 설계한 API는 시스템 운영에 관한 API, 메타데이터 단계에 독립적인 API, 메타스키마 단계에 독립적인 API로 구분할 수 있다. 시스템 운영에 관한 API는 시스템의 활성화 및 종료 등 시스템

```
long IrdsOpen(login_info li, short CurrSessId)
long IrdsClose(short CurrSessId)
long CreateIRDDefinition(login_info li,short CurrSessId)
long IrdsRemoveIRDDefinition(short CurrSessId)
```

<그림 2> 시스템 운영에 관한 API

운영에 관한 API로 정보저장소 개방 API, 정보저장소 폐쇄 API, 메타스키마 생성 API, 메타스키마 제거 API가 있다. 메타데이터 단계에 독립적인 API는 메타데이터 단계의 자료를 조작할 때 사용되는 API로 메타데이터 객체/관계 추가 API, 메타데이터 객체/관계 삭제 API, 메타데이터 객체/관계 수정 API, 메타데이터 출력 API, 메타데이터 생성 API, 메타데이터 제거 API가 있다. 메타스키마 단계에 독립적인 API는 메타스키마 단계의 자료를 조작할 때 사용되는 API로 메타스키마 정의 객체/관계 추가 API, 메타스키마 정의 객체/관계 삭제 API, 메타스키마 정의 객체/관계 수정 API, 메타스키마 정의 출력 API가 있다.

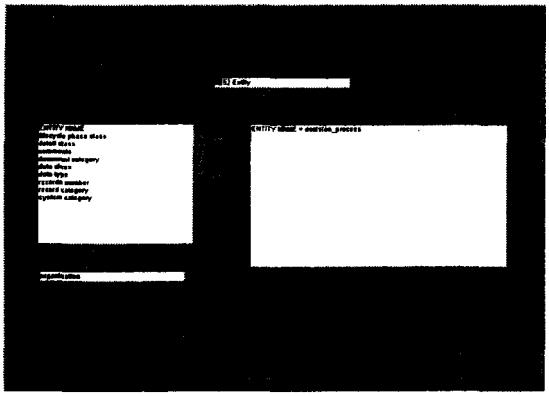
IDL은 정의만를 위한 것으로 직접적으로 사용할 수 있는 코드가 될 수 없기 때문에 API의 형태로 서비스 인터페이스를 구성하였다. API의 구현을 위해서는 CORBA IDL로 구성된 정의문을 컴파일한 코드에 부가적인 코드를 첨가하여 API를 구성한다. <그림 2>에서 제시하는 API는 시스템 운영에 관한 API 예이다.

분산 환경에서의 워크플로우를 위한 정보저장소의 각 구성요소의 설명은 다음과 같다.

● 정보저장소 사용자 인터페이스

정보저장소 사용자 인터페이스는 최종 사용자와 직접적으로 통신할 수 있는 사용자 단말 뷰로써 기본적으로 그래픽 사용자 인터페이스로 구현하였다. 이는 정보저장소 명령어에 익숙하지 않은 사용자에게 정보저장소 명령어를 숨겨 주며 정보저장소에 쉽게 접근할 수 있도록 정보저장소 표준의 기능을 갖춘 인터페이스이다.

사용자 인터페이스 메뉴는 크게 시스템 운영, 메타스키마 조작, 메타데이터 조작, 메타데이터 뷰, 유틸리티, 도움말로 구성된다. 시스템 운영 메뉴를 선택하면 시스템 개방, 시스템 종료가 나타난다. 시스템 개방을 선택하면 사용자 로그인 정보를 입력하기 위한 창이 나타난다. 시스템 종료를 선택하면 확인 메시지 후 시스템을 종료한다. 메타스키마 조작 메뉴를 선택하면 메타스키마 추가, 삭제, 수정 메뉴가 나타나며, 메타데이터 조작 메뉴를 선택하면 메타데이터 추가, 삭제, 수정, 메뉴가 나타난다. 메타데이터 뷰 메뉴는 메타데이터/메타스키마의 내용을 관련된 관계 정보와 함께 보여주고, 유틸리티는 메타데이터 생성과 삭제를 위한 메뉴이다. 이 중 메타데이터 추가를 위한 입력 창의 예는 <그림 3>과 같다.



<그림 3> 사용자 인터페이스 예

메타데이터 추가 인터페이스 화면은 테이블 선택을 위한 콤보 박스, 선택된 테이블의 추가할 속성을 입력 받기 위한 리스트 박스, 추가한 결과를 보여주는 리스트 박스, 추가할 속성 값을 입력 받는 콤보 박스로 구성된다. 다른 사용자 인터페이스도 각각의 특성에 맞게 구성하였다.

● 질의 해석기 (Request Interpreter)

ORB로 전달된 질의를 해석하여 정보저장소에 해당되는 함수로 변환한다. ORB로 전달된 질의는 일반적으로 워크플로우에서 사용되는 질의로 이를 직접적인 정보저장소에 해당하는 함수로 변환하기 위해서는 한 개 이상의 정보저장소 함수가 조합된 형태가 될 수 있으며 이를 하나의 트랜잭션으로 구성한다.

예를 들어, 클라이언트에서 새로운 사용자를 등록하라는 질의인 IrdsCreateIRDDefinition을 호출하면 서버의 IrdsCreateIRDDefinition은 공통 테이블인 MAT, MET, MRT를 생성하는 함수를, 메타스키마 단계의 테이블인 IRD_DEF_TABLE, IRD_DEF_COLUMN, IRD_DEF_REL을 생성하는 함수를, 메타데이터 단계의 테이블인 IRD_TABLE, IRD_COLUMN, IRD_REL을 생성하는 함수를, 내부 테이블인 IRD_OBJECT를 생성하는 함수를 차례로 호출하며, 일관성 검사기에 질의를 전달한다.

● 환경 설정기 (Environment Seter)

초기 시스템 환경을 설정하고 시스템을 이용하고 있는 사용자의 등록 및 관리를 담당한다. 환경 설정기는 처음 사용자 정보를 확인하고 사용자 등록일 경우 사용자에 맞는 환경을 설정한다. 환경 설정 과정은 사용자에 해당하는 메타테이블을 구성하고 각각에 해당하는 기본적인 메타데이터들을 로드시키는 과정이다.

환경 설정기는 처음 사용자가 로그인 해서 사용자 등록이 되어 있지 않을 경우, 사용자 등록을 원할 때 실행된다. 사용자 처음 로그인하는 함수는 IrdsOpen이고, 이 경우 사용자가 등록되었으면 사용자의 세션 ID를 반환하고, 그렇지 않으면 오류 값을 반환한다. 만약 오류 값이 반환되면 사용자 등록을 수행한다. 사용자 등록은 사용자에게 새로운 세션 ID를 부여하고 사용자 정보와 함께 DICT_NAME 테이블에 등록한다. 이 후 메타스키마 단계의 테이블인 IRD_DEF_TABLE, IRD_DEF_COLUMN, IRD_DEF_REL과 메타데이터 단계의 테이블인 IRD_TABLE, IRD_COLUMN, IRD_REL과 내부 테이블인 IRD_OBJECT를 생성하고 기본적인 데이터를 로드시킨다.

● 일관성 검사기 (Consistency Checker)

전달된 명령어의 종류에 따라 메타데이터

의 내용이 정당한지에 대한 일관성 및 무결성을 검사한다. 개체 이름의 길이, 조작하려는 항목의 IRDS 내의 존재 여부, 시스템이 관할하는 사항의 수정 의도 등이 이 모듈에서 수행하는 사항이다. 연속적인 일관성 검사의 실행에 따라 오류 코드가 생성되며 하나의 오류라도 발생되면 질의의 수행을 중단하고 롤백한다. 검사 항목의 예는 다음과 같다.

— 매개변수 항목이 제대로 들어있는지를 검사한다. IRD 정의 단계의 메타스키마 관계의 추가일 경우 매개변수 중 하나인 ColList의 ColName에 me1_name, me2_name, me_rel_name과 이에 해당하는 값(ColValue)은 반드시 있어야 한다.

— 매개변수 항목이 정당한지를 검사한다. IRD 정의 단계의 메타스키마 관계의 추가일 경우 me1_name과 me2_name는 IRD_DEF_TABLE에 존재하는 값 중의 하나이어야 한다. 존재하면 일관성 검사가 정당한 것이고 그렇지 않으면 오류이며, 이때는 오류 번호를 반환하고 실행을 즉각적으로 중단하고 롤백한다.

● 질의 처리기 (Request Processor)

질의 해석기에 의해 변환된 정보저장소 명령을 처리하는 핵심 부분으로 각각의 질의를 다시 몇 개의 질의로 분할하여 일관성 검사를 거친 후 질의가 타당하다면 질의 처리를 시작한다. 질의 해석기에 의해 적당한 함수를 구성하고 일관성 검사기에 의해 질의가 정당하다면 질의 처리를 시작한다. 질의 처리는 해당 명령어에 맞는 SQL 문을 생성하고 이를 적절한 변수에 저장한다. 저장된 변수는 COI에서 제공하는 SQL 질의 처리 함수의 매개변수로 사용되어 이를 실행한다. 실행이 성공적으로 끝나면 SQL 질의 처리 함수는 질의의 결과를 반환하며, 질의 결과에 대해 오류이면 오류 번호를, 성공이면 결과 값을 반환한다.

● 데이터베이스 서비스 인터페이스 (DBMS Service Interface)

명령어들이 필요로 하는 모든 정보자원을 DBMS를 이용하여 필요한 서비스를 수행한다. 이는 일관성 및 무결성 검사에 대한 오류가 존재하지 않으면 질의를 수행하기 위해 필요한 SQL 명령어를 구성한다. SQL 명령어는 연속하는 몇 개의 SQL 명령어 집합이 될 수 있다.

● 메타테이블(Meta-table)

메타데이터 정보를 내용별, 단계별로 체계화하여 저장하여 놓은 내부 정보 테이블로 단계별로 구분 가능하며 내부 테이블, 환경 테이블, 공통 테이블, 한정 테이블로 구성된다. 메타테이블 설계 시 고려할 점은 두 단계의 데이터를 단계별 또는 단계간의 쌍으로 묶어 표현하는 것이다. 여기서 말하는 두 단계란 실세계 객체의 스키마인 메타데이터와 메타데이터의 스키마인 메타스키마를 말한다. 본 논문에서는 메타테이블을 두 단계 쌍들에 대해 최대한의 병렬성을 유지하면서 테이블의 내용에 따라 네 종류의 테이블로 구분하였다. 본 논문에서는 ISO IRDS 표준에 준하여 <표 1>과 같이 네 부류의 메타테이블을 구현하였다.

<표 1> 메타데이터 테이블

테이블 분류	테이블 설명	관계형 테이블
내부 테이블	한정 테이블의 각 객체에 대한 관리 및 버전 제어를 위해 설계된 테이블	IRD_OBJECT
환경 테이블	IRDS 사용자, 구현 시 한계 값 등을 정의	DICT_NAME
한정 테이블	자료 모델링 기능의 구조화 규칙을 표현하며 IRD 한정 테이블은 메타데이터를, IRD 정의 한정 테이블은 메타스키마를 관리	IRD_DEF_TABLE, IRD_DEF_COLUMN IRD_DEF_REL, IRD_TABLE IRD_COLUMN, IRD_REL
공통 테이블	모든 단계에서 공통으로 사용되는 테이블	MAT, MET, MRT

4. 워크플로우를 위한 정보저장소 활용

분산환경에서의 워크플로우 관리를 위한 정보저장소로의 활용 가능한지에 대한 검증을 위해서 계층적인 워크플로우 데이터 수집 및 분석이 필요하다. 계층화된 단계의 가장 상위 단계인 IRD 정의 스키마(IRD Definition Schema) 단계에서는 정보저장소의 가장 존재론적인 의미를 가지는 기본적인 객체를 관리한다. 따라서 이 단계의 데이터는 특정 응용분야와 상관없이, 처음 시스템이 구현되면 한번 정의되어 이후에 수정이 가해지지 않는다. 이 단계에서는 MET 테이블에 object, property, association의 세 객체와 MRT 테이블에 그들 사이의 관계인 object has property, object has association의 두 객체, 그리고 MAT 테이블에서 메타스키마 단계의 테이블에 존재하는 모든 컬럼들을 관리한다.

워크플로우 데이터는 응용데이터의 스키마인 메타데이터와 메타데이터의 스키마인 메타스키마로 나누어 분석되어야 한다. 메타데이터는 IRDS 표준에서는 IRD라고 명명하고 있으며, 응용데이터베이스에 대한 스키마이면서 메타스키마가 가지는 타입의 인스탄스이며 응용 데이터베이스 정보를 나타내기 위한 타입을 제공하는 데이터이다. 메타스키마는 IRDS 표준에서는 IRD 정의라고 명명하고 있으며, 메타데이터 단계에 있는 정보를 나타내기 위한 수단을 제공하며 메타데이터 단계의 정보의 타입을 결정하는 데이터이다. 이와 같이 분석된 결과를 가지고 IRDS에서 제공하는 모델링 방법에 따라 통합적인 메타모델링이 이루어져야 하는데, 모델링 또한 메타모델링과 메타스키마모델링이 필요하며 두 단계의 데이터는 타입과 인스탄스의 관계를 가지고 있어야 한다.

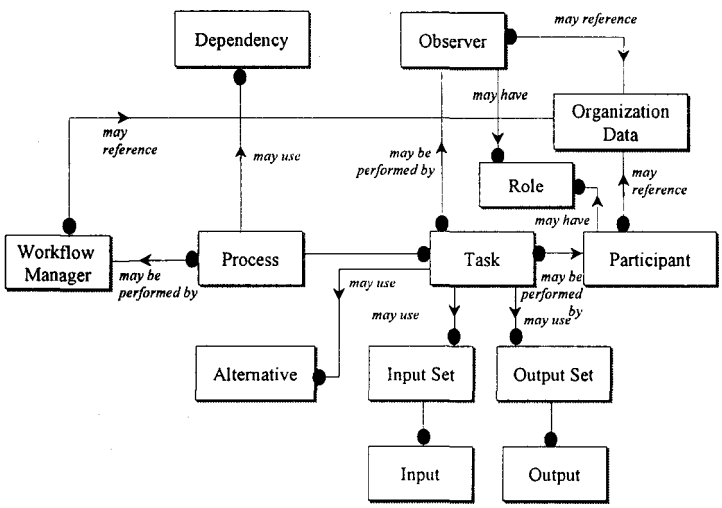
4.1 메타데이터 단계의 메타모델링

메타데이터 단계에서는 메타데이터들이 저장되며 메타데이터내에 저장되는 데이터의 타입은 메타스키마 단계에서 반드시 기술되어 있어야 한다. 즉, 메타데이터 단계의 데이터는 일반 데이터베이스에 저장되는 인스탄스에 대한 스키마이며 메타스키마 단계의 인스탄스를 타입으로 가지는 데이터들이다. 이 단계에서의 메타모델링은 <그림 4>와 같으며 이 모델링은 OMT[James, 1991] 방법론을 따른다.

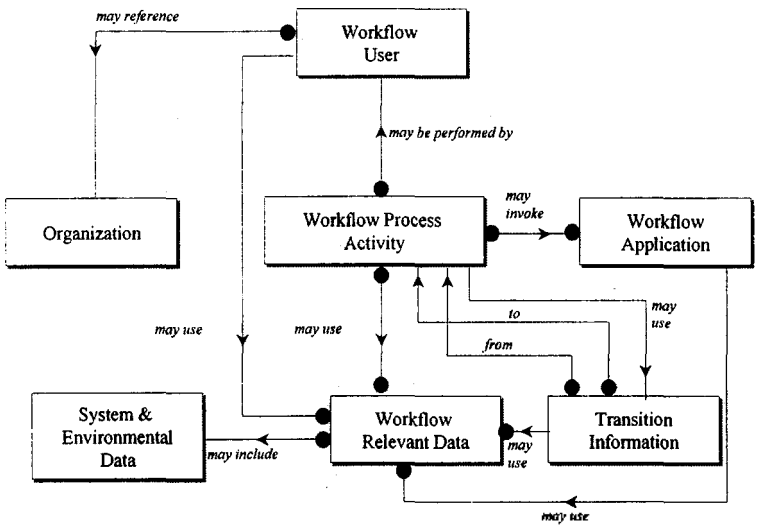
<그림 4>의 모델의 중심은 Process라 할 수 있다. Process는 워크플로우의 작업 단위로 여러 개의 작은 작업 단위인 복수 개의 Task로 구성된다. Task는 Task의 참여자(participant)와 Task 간의 의존 관계를 나타내는 Alternative, Task의 입출력 집합을 나타내는 Input Set, Output Set으로 구성된다. 워크플로우를 위한 사용자는 전체 워크플로우를 관장하는 Workflow Manager, 특정 Task에 참여하는 Participant, 워크플로우의 흐름을 감시하는 Observer가 있다. 이 외에 조직체 정보를 나타내는 Organization Data 등이 있다.

4.2 메타스키마 단계의 메타모델링

이 단계는 메타스키마를 위한 단계로 하나 이상의 메타스키마를 기술하고, 하나 이상의 메타스키마는 하나 이상의 메타데이터를 포함한다. 또한 메타데이터 단계에 있는 정보를 나타내는 수단을 제공하기 위해 메타데이터에 대한 타입을 정의하여 관리한다. 이러한 타입



<그림 4> 워크플로우를 위한 메타데이터 단계의 모델



<그림 5> 워크플로우를 위한 메타스키마 단계의 모델

은 메타스키마 단계 테이블의 인스탄스로 관리된다. 예를 들어, process, task라는 메타데이터 단계의 데이터의 타입은 Workflow Process Activity가 되고, Workflow Manager, Observer, Participant라는 데이터의 타입은 Workflow User가 될 수 있다. 이처럼 메타데이터 단계에 나타나는 인스탄스의 타입을 하나의 인스탄스로 고려하여 관리하는 것이 메타스키마 단계의 테이블이다. 메타스키마 단계에 저장될 데이터들은 메타모델링에 따라 각 테이블에 저장된다.

메타스키마 단계에서 워크플로우를 위해 관리되는 메타스키마들은 <그림 5>와 같이 프로세스 정의를 위한 Workflow Process Activity, 워크플로우 관련 데이터인 Workflow Relevant Data, 조직체 정보를 위한 Organization, 워크플로우 사용자를 위한 Workflow User, 워크플로우에서 호출되는 응용프로그램을 위한 Workflow Application, 워크플로우 프로세스나 태스크 간의 의존 관계를 나타내는 Transition Information, 전체 시스템과 환경 등의 정보를 가지고 있는 System & Environment Data가 있다.

메타스키마 단계와 메타데이터 단계는 타입/인스탄스 관계가 있다. <표 2>는 메타데이터 단계와 메타스키마 단계의 데이터에 대한 타입/인스탄스의 관계이다.

4.3 워크플로우를 위한 정보저장소 검증

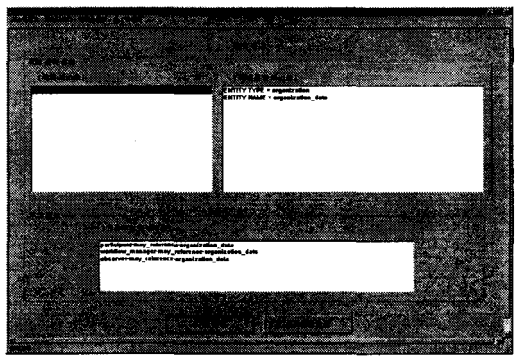
정보저장소가 워크플로우 응용을 위해서 어떠한 식으로 데이터를 관리하는지를 예제를 통해 서비스 인터페이스와 관련된 테이블을 가지고 설명하며, 그 결과가 사용자 인터페이스로 어떻게 보여지는지 기술한다. 예제는 메타데이터와 관련된 예제와 메타스키마와 관련된 예제로 보인다.

<표 2> IRD 단계와 IRD 정의 단계의 타입/인스탄스 관계

	[illegible] 인스탄스	IRD 단계 인스탄스
IRD_DEF_TABLE /IRD_TABLE	Workflow Process Activity	Process, Task
	Transition Information	Alternative, Dependency
	Workflow User	Workflow Manager, Observer, Participant
	Workflow Relevant Data	Input Set, Output Set, Input, Output
IRD_DEF_REL/ IRD_REL	Organization	Organization Data
	Workflow Process Activity *may be performed by* Workflow User	Process *may be performed by* Workflow Manager, Task *may be performed by* Observer, Task *may be performed by* Participant

[예제 1] "Organization Data 테이블을 삭제할 경우 이 테이블에 연관되어 변경이 초래되는 테이블을 찾아라"

메타데이터 단계의 일부의 테이블이 더 이상 사용되지 않거나 아니면 변경을 할 경우가 생길 경우, 예를 들어 Organization Data 테이블에 변경이 초래될 때 정보저장소 관리자는 이 테이블에 관한 정보를 얻고 이 테이블과 연관된 모든 관계들의 정보를 파악하여야 테이블 변경에 따른 파급효과를 미리 감지할 수 있다. 이러한 경우에는 정보저장소 사용자는 사용자 인터페이스를 통해 테이블 입력 창에서 IRD_TABLE을 선택하고, 출력할 객체 이름인 Organization을 입력하고, 출력하여 보고자 하는 속성을 선택하고 확인 버튼을 누르면 된다. 시스템은 Output이라는 서비스 인터페이스 함수를 통해 입력한 매개변수를 넘겨주고, 서버의 질의 해석 및 일관성 검사, 질의 처리를 통해 관련된 모든 정보를 얻는다. <그림 6>은 [예제1]의 결과 화면이다.



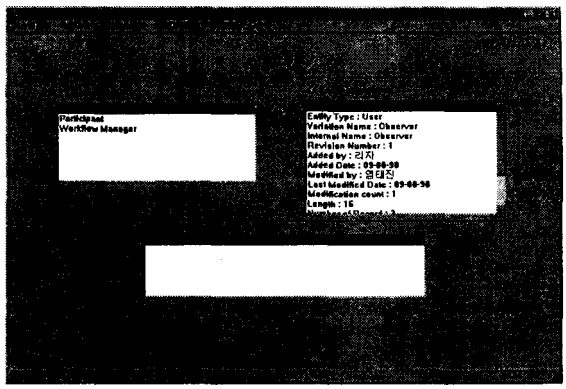
<그림 6> [예제1]의 결과

Output 서비스를 수행하여 결과를 보여주는 화면은 크게 세 부분으로 나누어진다. 하나는 화면 왼쪽 위 부분의 엔티티를 표시하는 부분이고 또 하나는 화면 오른 쪽 위 부분의 엔티티에 관한 일반적인 정보를 표시하는 부분이고, 또 하나는 화면 아래 쪽의 엔티티와 관련된 모든 관계를 보여주는 부분이다. Organization Data의 엔티티에 관한 정보는 이

엔티티의 타입이며 IRD 정의 단계의 한 인스탄스인 Organization, 이 엔티티의 이름인 Organization Data이다. 이 엔티티와 관련된 관계는 Participant *may reference* Organization Data, Workflow Manager *may reference* Organization Data, Observer *may reference* Organization Data 이며 결국 이 테이블을 변경했을 경우, 관련된 세 개의 관계와 세 개의 엔티티(Observer, Participant, Workflow Manager)에 변경이 가해짐을 알 수 있다.

[예 2] "워크플로우 사용자와 관련된 모든 테이블을 찾아라"

또한 워크플로우를 사용하는 모든 사용자들을 확인해야 할 경우가 있다. 데이터베이스의 경우, 사용자와 관련된 테이블이 여러 개의 테이블에 나누어져 있다면 어떠한 테이블이 사용자를 관리하는 테이블인지 찾아내기가 어렵다. 이러한 경우, 메타스키마 단계의 검색을 통해 사용자와 관련된 테이블이 무엇인지 알 수 있다. 즉 사용자는 사용자 인터페이스를 통해 테이블 입력 창에 사용자에 관련된 테이블을 인스탄스로 가지고 있는 IRD_DEF_TABLE을 선택하고, 출력할 객체인 Workflow User를 선택하고, 출력할 속성을 선택하고 확인 버튼을 누르면 시스템은 Output이라는 서비스 인터페이스 함수를 통해 입력한 매개변수를 넘겨주고, 서버의 질의 해석 및 일관성 검사, 질의 처리를 통해 관련된 모든 정보를 넘겨 받는다. <그림 7>은 [예제 2]의 결과 화면이다.



<그림 7> [예제2]의 결과

화면은 세 부분으로 구성되는데 화면의 왼쪽 위 부분은 사용자와 관련된 테이블을 보여주고 화면의 오른쪽 위 부분은 선택된 테이블에 대한 상세한 설명을 보여준다. 또한 화면의 아래 부분은 엔티티와 관련된 관계를 보여준다. 사용자와 관련된 테이블은 모두 Observer, Participant, Workflow Manager이고 Observer일 경우 엔티티 이름, 변형 이름, 내부 이름이 Observer이고 엔티티 타입은 Workflow User임을 알 수 있다. 또한 Observer 엔티티를 처음 생성 시간(09-08-98)과 생성한 사람(리자), 마지막으로 수정한 시간(09-08-98)과 수정한 사람(염태진), Observer 엔티티의 내부 길이(16), 레코드 수(4) 등의 정보를 알 수 있다.

4.4 워크플로우를 위한 정보저장소의 효과

본 논문에서 구현한 정보저장소를 워크플로우에 적용할 때 파일 시스템이나 데이터베이스 시스템을 이용하는 것보다 다음과 같은 효과가 있다.

● 메타데이터에 대한 모델링 기능을 지원

메타스키마 단계는 메타데이터 단계에서 산출되는 모든 정보, 메타데이터 단계는 응용 데이터베이스에서 산출되는 모든 정보의 구조 및 의미 등을 정의하기 위한 통일된 규칙을 제공한다. 특히 워크플로우에 의해 산출되는 정보는 상당히 복잡한 데이터이며, 이러한 정보자원들을 통일된 규칙에 의해 시스템 사용자는 메타데이터 모델링에 의해 얻어진 데이터를 손쉽게 저장 시스템에 저장하여 관리할 수 있다. 또한 이미 정의된 모델링된 결과를 사용자가 손쉽게 확인하여 볼 수 있다. 기존의 데이터베이스 시스템에서도 스키마 정의 기능을 일부의 테이블로 부분적으로 가지고 있으나 상당히 제한적이고 각 정보 객체간의 복잡한 메타데이터에 대한 저장은 어렵다.

● 스키마의 유연성과 확장성 제공

응용 데이터의 제어를 위해서 메타데이터가 필요한 것과 마찬가지로 메타데이터의 변경 등의 제어를 하기 위해서는 메타데이터 단계뿐만 아니라 메타스키마 정보가 필요하다. 정보저장소에서 이러한 역할을 하는 것이 바로 메타스키마 단계와 메타데이터 단계이다. 메타스키마 단계는 각 메타데이터, 즉 응용 데이터에 대한 메타데이터를 하나의 인스탄스로 취급하고 그에 대한 타입을 정의함으로써 메타데이터를 조직적으로 저장, 관리하여 메타데이터의 변경에 쉽게 대처하고 유연성을 제공한다.

예를 들어 워크플로우만을 관리하는 모델로 이루어진 시스템이 소프트웨어나 전 조직체에 관련된 정보를 취급하는 모델까지 포함한 큰 시스템이 될 경우 기존의 시스템을 잘 활용하여 시스템을 유연하게 확장하며 재사용할 수 있도록 하는 것이 필요하다. 그러나 일반적인 데이터베이스처럼 메타데이터 단계만을 가지고 스키마를 확장하려고 할 경우 스키마로 확장으로 인한 그 복잡도는 매우 크다. 이러한 복잡성을 줄이고 스키마 확장을 가능하게 하는 것은 메타스키마 단계에서 먼저 스키마를 확장하고 관련된 메타데이터만을 확장하면 스키마 확장으로 인한 복잡도는 줄어 들 수 있다.

● 스키마 통합이 용이

메타스키마 단계의 데이터는 메타데이터들의 리스트라 할 수 있다. 즉 지역적으로 분산되어 있는 메타데이터들의 리스트를 재구성하며, 메타데이터들의 타입을 정의하고, 그들 간의 관계를 타입으로 구성하여 저장한다. 예를 들어 지역적으로 분산되어 있는 두 스키마가 이름은 다르지만, 근본적으로 같은 내용을 나타내는 스키마라면 정보저장소에서 관리하는 두 단계의 타입을 비교해 하나의 스키마로 통합할 수 있다. 따라서 같은 성격을 가진 이 두 테이블을 통합하여 사용하고자 할 때, 메타스키마 단계의 정보를 이용하여 같은 속성을 가진 정보만을 이용하여 두 스키마를 통합하여 하나의 가상 스키마를 구성할 수 있다.

5. 성능 분석

본 논문에서 제안하고 구현한 시스템과 기존의 정보저장소인 UNISYS의 UREP, IBM의 Repository Manager/MVS와 비교 분석한다.

UREP는 OMG에서 제안한 분산 환경에서의 정보저장소를 기반으로 UNISYS사에서 상용 제품으로 만든 정보저장소다. UREP은 3단계의 저장 구조에 데이터를 저장하고 이를 제한된 분산 환경에서 운용 가능하도록 되어 있다. UREP은 분산 환경에서 동적으로 작동하며, 여러 가지 도구와 통합 가능하며, 다양한 사용자 인터페이스, 응용 프로그램 등을 지원한다. 그러나 UREP는 현재 진정한 분산 환경이라 볼 수 없고 제한적으로 CORBA가 아닌 RPC로 작동한다[진성일, 1998; Ovum, 1993].

Repository Manager/MVS는 IBM이 개발한 응용 시스템개발 환경인 AD/Cycle의 모든 생명주기 단계를 지원하는 소프트웨어 도구들과 서비스를 제공하는 핵심 부분이다. 정보저장소 내에 저장된 정보는 임의의 도구에 의해 접근되어지고 Repository Manager/MVS에 의해 표현되는 형식에 의해 조작된다. SAA (System Application Architecture)의 한 요소인 Repository Manager/MVS는 SAA 환경 전체에 걸쳐 포괄적이고 통합된 응용시스템 개발의 틀을 제공하는 기반이 된다[진성일, 1998; Ovum, 1993].

본 논문에서 구현한 시스템은 ISO IRDS에

<표 3> 정보저장소 시스템 비교

데이터관리방법	3 layer	3 layer	3 layer
모델링 방법	객체지향 모델링	객체지향 모델링	객체지향 모델링
메타데이터 관리 기능	가능	가능	가능
생명주기 관리 기능	불가능	가능	가능
버전관리기능	불가능	가능	가능
분산 환경	가능	제한적	불가능
사용자 인터페이스	GUI	GUI	GUI
타 시스템과의 연계 가능성	가능	불가능	불가능
이질 환경	가능	불가능	불가능

기반하여 구현된 시스템으로 분산 환경에서 자유롭게 운영될 수 있도록 CORBA 기반으로 되어 있다. 따라서 분산 환경에서 서로 다른 이질적인 시스템들이 정보저장소를 공유하며, 시스템들이 서로 연계되어 사용될 수 있다. 또한 본 시스템은 사용자 편의를 위해 윈도우 기반의 GUI를 지향한다.

<표 3>은 본 논문에서 구현한 분산 환경에서의 정보저장소와 UNISYS의 UREP, IBM의 Repository Manager/MVS 세 시스템의 비교한 결과이다.

6. 결론

워크플로우에 대한 관심 및 워크플로우가 사용하는 모든 정보 자원을 효율적으로 저장, 관리, 통합하여 공유 가능하게 지원하는 시스템에 대한 요구는 날로 증가하고 있다. 그러나 워크플로우 시스템에서 사용되고 산출되는 정보를 저장하기 위해 기존의 시스템에서는 파일이나 데이터베이스 시스템을 사용하였으나, 이들이 갖는 정보는 일차적인 정보만을 표현하고 있어 모든 정보자원을 효율적으로 관리하기 위해서는 정보저장소가 필요하다. 따라서 본 논문에서는 워크플로우 관리를 위한 정보저장소를 설계 및 구현하였으며, 또한 워크플로우와 연동하여 사용할 수 있도록 분산 환경에서 운용 가능하도록 하였다. 이를 위해 분산 객체 환경에서 운영이 가능한 워크플로우 관리를 위한 정보저장소 프로토타입을 설계 및 구현하였다. 워크플로우 관리를 위한 정보저장소의 핵심인 서비스 인터페이스를 위한 DBMS로는 Informix Universal Server 객체관계용 데이터베이스를 사용하였고, 분산 객체 환경을 지원하기 위해 IONA의 Orbix 2.2를 사용하였으며, Visual C++을 이용한 윈도우 기반 사용자 인터페이스도 구현하였다. 또한 이를 워크플로우 시스템에 적용하기 위해 통합메타모델을 제안하고 이의 활용도를 검증하였으며, 분산객체 환경에서 구현된 정보저장소를 이용시 효과에 대해 기술하였다.

그러나, 원래가 정보저장소의 프로토타입을 구현할 목적이었으므로, 버전 관리 기능, 보안 기능 등은 제외한 정보저장소의 핵심 기능만을 구현하였다. 또한 워크플로우의 모든 데이터를 관리하는 것이 아니라 워크플로우에서 생성되는 워크플로우 관련 데이터와 프로세스 정의 데이터 등의 스태틱한 데이터 만을 관리할 수 있도록 구현하였다. 향후 연구 방향으로 데이터의 변경에 대해 능동적으로 대처할 수 있는 버전 관리 기능, 인가된 사용자만이 정보자원을 접근할 수 있고 정보에 대한 보호를 위한 보안 기능 등이 요구되며, 또한 다양한 워크플로우 응용을 위해서 워크플로우 제어 데이터와 응용 데이터까지 관리할 수 있도록 하는 것이 바람직하며, 다수의 워크플로우 엔진을 사용하면서 그들간의 협력 체제가 이루어 질 수 있도록 정보 자원을 관리할 수 있는 기능이 요구된다. 이를 위해서는 자원에 대한 독립적인 유일한 위치를 보장하면서 중복된 목록을 제공해 줄 수 있는 기능 등의 목록 관리 기능이 필요하다.

참고 문헌

[David, 1994] David Hollingsworth, Workflow Management Coalition The Workflow Reference Model, Workflow Management Coalition, 1994

[Informix, 1997a] Informix, Informix Error Message, Informix Inc., 1997

[Informix, 1997b]Informix, INFORMIX FastStart for University, Informix Inc., 1997

[Informix, 1997c]Informix, Informix-Universal Server Informix Guide to SQL, Informix Inc., 1997

[IONA, 1995] IONA Inc., Orbix 2 Programming Guide, IONA Inc., 1995

[James, 1991] James Rumbaugh 외, Object-Oriented Modeling and Design, Prentice Hall International Inc., 1991

[Mark, 1991] Mark Jones, Brave New World : A Vision of IRDS, Database Programming and Design, pp. 43-49, Nov, 1991

[Mark, 1992] Mark R. Jones. "Unveiling Repository Technology", Database Programming and Design, pp. 28-35, April, 1992

[Nortel, 1997] Nortel, Workflow Management Facility Specification, OMG, 1997

[Randy, 1996] Randy Otte, Understanding CORBA, Prentice Hall International Inc., 1996

[Ovum, 1993] Repositories and Frameworks : a Detailed Product Evaluation Volume 2, Ovum, 1993

[Shunsuke, 1997] Shunsuke Akifuji, Workflow Management Facility, OMG, 1997

[Thomas, 1995] Thomas J. Mowbray, The Essential CORBA : Systems Integration Using Distributed Objects, John Wiley & Sons Inc., 1995

[곽준기, 1997] 곽준기, 백정렬, Inside Secret Visual C++ 5.0, 삼각형, 1997

[김기봉, 1998] 김기봉, IRDS 기반 정보저장소와 CASE 도구 통합에의 활용, 박사 학위 논문, 충남대학교, 1998

[김기봉, 1995] 김기봉, 박중기, 조유희, 진성일, 정보자원 관리를 위한 IRDS 서비스 사용자 인터페이스의 설계 및 구현, 정보과학회논문지, 제22권 제11호, pp. 1499-1509, 11월 1995

[박재현, 1998] 박재현, Core CORBA 코아 코바, 영한출판사, 1998

[진성일, 1998] 진성일 외, 통합자료관리체계 시험구현연구 최종보고서, 국방과학연구소, 1998

[전산원, 1995] 한국전산원, 정보자원사전시스템(IRDS) 서비스 표준연구, 한국전산원, 1995

저자 소개

염태진 (e-mail : tjyeom@sorec.chungnam.ac.kr)
1997년 충남대학교 컴퓨터과학과 졸업 (학사)
1999년 충남대학교 대학원 컴퓨터과학과 졸업(이학석사)
1999년~현재 충남대학교 소프트웨어연구센터 전임연구원
관심분야 : 통합데이터베이스, CALS/EC, 정보저장소, 객체지향, Workflow 등

박재형 (e-mail : jhpark@cs.chungnam.ac.kr)
1998년 충남대학교 컴퓨터과학과 졸업 (학사)
1998년~현재 충남대학교 대학원 컴퓨터과학과 석사과정
관심분야 : 데이터베이스, 통합데이터베이스, 멀티미디어시스템, 분산객체시스템, 공동작업 시스템 등

리 자 (e-mail : lizi@sorec.chungnam.ac.kr)
1992년 중국 길림대학교 컴퓨터과학과 (학사)
1999년 현재 충남대학교 컴퓨터과학과 대학원 석사과정
관심분야 : 데이터베이스, 통합데이터베이스, 분산객체시스템, 버전제어시스템 등

김기봉 (e-mail : kbkim@tiger.tjhealth.ac.kr)
1991년 충남대학교 컴퓨터과학과 졸업 (학사)
1993년 충남대학교 대학원 컴퓨터과학과 졸업 (이학석사)
1998년 8월 충남대학교 대학원 컴퓨터과학과 (이학박사)
1994년 9월~1997년 2월 혜전대학 전산정보처리과 교수
1997년 3월~현재 대전보건대학 전산정보처리과 교수
관심분야 : 통합데이터베이스, 정보저장소, CALS/EC 등

진성일 (e-mail : sijin@cs.chunanam.ac.kr)
1978년 서울대학교 계산통계학과 (학사)
1980년 한국과학기술원 전산학과 (이학석사)
1994년 한국과학기술원 전산학과 (이학박사)
1983년~현재 충남대학교 컴퓨터과학과 교수
관심분야 : 데이터베이스, 멀티미디어시스템, 소프트웨어공학, 시뮬레이션 모델링 등