經營科學
第16卷 第1號
1999年 5月

제품 개발 프로세스 관리를 위한 다층 통합 워크플로우 시스템 개발*

강석호** · 김영호** · 김동수** · 배준수** · 배혜림**

Development of a Multi-Layered Workflow Management System for Product Development Processes

S.H. Kang** · Y.H. Kim** · D.S. Kim** · J.S. Bae** · H.R. Bae**

Abstract

In this paper, we propose a multi-layered architecture of workflow management systems based on CORBA (Common Object Request Broker Architecture). The system aims to support product development processes in distributed environment. Many companies have started to adopt workflow management systems to manage and support their business processes. However, there are many problems in direct application of those systems to product development environments. These mainly resulted from the dynamic features of product development processes. It is strongly required to support dynamic processes as well as static and procedural ones in an integrated and consistent manner. To meet these requirements, a basic workflow management system has been developed as the core component of the integrated architecture. This performs the basic functions of workflow management system. Second, a dynamic workflow management system based on a bidding mechanism has been developed to manage processes that cannot be easily defined or are likely to be modified. Finally, an SGML workflow management system, which is the third layer in the architecture, has been developed to manage documents processing workflows by integrating SGML documents contents and process information into the structured SGML document.

* 이 논문은 1996년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음
** 서울대학교 산업공학과

1. 서 론

본 연구에서는 분산객체 관리 기술의 표준으로 자리잡고 있는 CORBA(Common Object Request Broker Architecture)를 이용하여 분산 환경에서 제품 개발 프로세스를 지원하기 위한 다층 워크플로우 관리 시스템(Workflow Management System) 아키텍처를 설계하고 개발하였다. 제품 수명 주기 단축, 소비자 구매 요구의 다양화, 경쟁의 심화 등으로 표현되고 있는 현재의 기업 환경 하에서 신속한 제품 개발 및 원가 절감은 제조 공정의 생산성, 유연성 제고 및 기업의 지속적 성장과 발전에 있어 필수 불가결한 요소로 인식되고 있다. 이러한 배경에서 출발하여 본 연구에서는 제품 개발 환경을 프로세스 관점에서 지원하여 업무의 흐름 및 정보에 대해 통제하고 관리할 수 있도록 하기 위한 다층 통합 워크플로우 관리 시스템을 개발하였다.

워크플로우 관리 시스템은 글로벌한 기업 환경에서 네트워크로 상호 연결된 컴퓨팅 기술을 이용하여 비즈니스 프로세스를 효율적으로 진행시키고 관리하는 도구로, 최근 관심이 집중되고 있는 PDM(Product Data Management)과 ERP(Enterprise Resource Planning)의 핵심 기능으로 자리 잡고 있다. 그러나, 보험회사 업무 프로세스, 병원 업무 관리, 문서 이미징 처리 프로세스 등과 같이 프로세스 정의가 용이하고, 정의된 프로세스에 대한 변경 요구가 거의 없는 프로세스의 관리에 적용하였던 기존의 워크플로우 시스템을 공학 설계 프로세스의 관리에 직접 적용하기는 쉽지 않다. 따라서, 기존의 정형화된 프로세스 관리뿐만이 아니라 명확한 프로세스 정의가 용이하지 않고, 실행 시 변경이 자주 발생하며, 프로세스 진행 시간이 비교적 긴 프로세스까지도 총괄해서 관리함으로써 제품 개발 환경에서 발생하는 프로세스들을 일관성 있게 관리하는 것이 필요하다.

본 연구에서는 우선 일차적으로 WfMC(Workflow Management Coalition)에서 제안한 워크플로우 참조 모델을 따라 워크플로우 개념을 파악하고, 기본 기능들을 구현하였다. 이 기본 워크플로우 관리 시스템은 다층 통합 워크플로우 관리 시스템 구조의 핵심적인 기능을 수행하는 부분으로, 프로세스 통제의 기본 기능을 수행하며, 정적인 프로세스를 관리한다. 동적인 워크플로우를 관리하기 위하여 비딩 메커니즘에 기반한 동적 워크플로우 관리 시스템을 개발하였다. 동적 워크플로우 관리 시스템은 중앙 집중식 엔진 중심의 구조에서 오는 문제점을 해결하기 위해 고안되었으며, 여기서 업무 처리 개체는 자치성을 갖고 있으며, 비딩을 통해 자율적으로 업무를 할당 받고 수행한다. 한편, 문서 관리 표준으로 정착되고 있는 SGML(Standard Generalized Markup Language) 문서와 공동 문서 작업 프로세스의 통합 관리를 위해 SGML을 이용하여 공동 문서 작업 프로세스를 정의하고, 처리할 수 있도록 하는 SGML 기반의 워크플로우 관리 시스템을 개발하였으며, 이는 다층 워크플로우 관리 시스템 아키텍처에서 SGML에 기반한 문서 데이터 작업을 관리하는 기능을 수행한다.

2장에서는 본 연구와 관련된 연구 현황과 배경이 되는 개념들을 소개하고, 3장에서는 제품 개발 환경의 특징과 제품 개발 프로세스 관리 요구사항을 설명한다. 4장에서는 본 연구에서 개발한 시스템의 각 서브컴포넌트들을 설명하고, 5장에서는 이들 서브컴포넌트들이 통합된 다층 워크플로우 관리 시스템 아키텍처를 설명한다. 6장에서는 구현한 시스템 내용을 제시하고, 7장에서 본 연구의 결론을 서술하였다.

2. 기존 연구 현황 및 관련 개념

2.1 워크플로우 시스템의 개념

컴퓨터를 이용하여 편리하고 자동적으로 수행되는 비즈니스 프로세스를 워크플로우라 하며, 워크플로우 관리 시스템이란 워크플로우 로직을 컴퓨

터로 표현하고 이 표현에 따라 생성된 순서를 따르는 소프트웨어의 실행을 통해 워크플로우를 완전하게 정의, 관리, 실행하는 시스템을 의미한다[4]. 즉, 워크플로우 관리 시스템은 업무 흐름의 자동화, 정보 및 문서 전달의 전자화, 그리고 일관적 데이터 접근 및 제어를 통해 업무 프로세스의 개선, 통제, 관리, 공동작업을 지원하는 소프트웨어를 일컫는다. 워크플로우 관리 시스템은 이러한 워크플로우 로직을 컴퓨터 표현에 의해 정해진 실행 순서를 따라 실행한다. 워크플로우를 관리한다는 것은 정보의 흐름과 관련된 부서 및 관련자들 간의 업무 연계를 돕고, 정보의 유연한 전달을 위한 기능들을 제공하는 것이다.

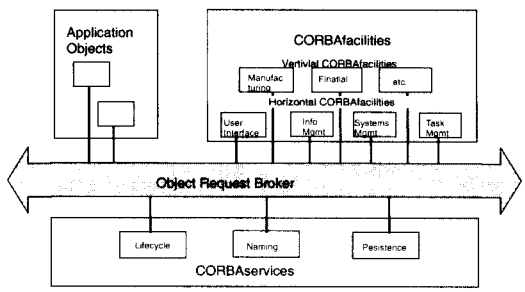
워크플로우 관리 기술이 업무 프로세스를 위한 모델을 제시해 주고, 업무의 실행과 관리를 지원하는 해결책을 세우는 기반을 제공하기 때문에 지난 몇 년 동안 크게 주목받아 왔다.

워크플로우 관련 표준화 연구로는 WfMC의 연구가 대표적인데, 워크플로우의 5가지 인터페이스와 메타 모델 표준화를 주도하고 있다. PIF(Process Interchange Format) Working Group에서는 프로세스 정보 교환 형식에 대한 표준을 연구하고 있다[4]. 또한 NIST(National Institute of Standards and Technology)에서는 프로세스 표현 언어에 대한 요구 사항을 명세하였다[5]. 한편 대학 또는 연구기관의 연구는 데이터베이스 트랜잭션 기반 프로세스 모델링 및 분석이나[6], 특정 도메인을 적용대상으로 하는 워크플로우 관리 시스템 구현과 상호운영성[7][8][9][10], 프로세스 표현 시멘틱에 대한 수학적인 접근 및 분석[11], 프로세스 정의 언어 사양 및 시멘틱에 대한 연구[12][13], 시스템 구현 분산 환경 미들웨어에 대한 연구[14][15], 프로세스 라이브러리 구축에 대한 연구[16] 등이 있다. 이들 연구에서는 대부분 기존의 워크플로우 시스템의 성능 향상이나 프로세스 정의 언어의 표현력, 분산 환경에서의 상호운영성 등에 대한 연구를 수행하였으나, 정적인 프로세스와 동적 변경이 가능한 프로세스를 통합하여 일관되게 관리할 수 있도록 하는 시도는 없었으며, 특히 SGML 문서 표준으로 문서 작업 프로세스를 정의하여 관리할 수 있는 기능이 포함된 시스템은 보고된 바가 없다.

2.2 CORBA 기반의 워크플로우 시스템

분산객체 환경을 실현하기 위한 기술로 OMG (Object Management Group)에서 제시하고 있는 분산객체 환경 규약인 CORBA가 표준으로 자리잡고 있다. CORBA 규약을 추진하고 제시하고 있는 OMG는 현재 600개 이상의 분산객체 기술개발 관련업체로 구성되어 1991년에 설립된 표준규약제정 기구이다. CORBA를 사용하면 각기 상이한 프로그래밍 언어로 작성된 개별 실행 객체들이 IDL (Interface Definition Language)에 정의된 표준 인터페이스를 통해서 상호 서비스를 제공하거나 제공받을 수 있다. 따라서 코드의 중복을 막을 수 있고, 컴포넌트 기반의 소프트웨어를 구축함으로써 네트워크를 통해 산재해 있는 여러 객체가 제공하는 서비스를 이용할 수 있으며, 따라서 각 시스템 간의 상호운영성을 확보할 수 있다[18].

CORBA는 객체 단위의 작업수행을 위한 구조이므로 실제 애플리케이션 수준의 통합적 수행을 위해서는 객체관리 아키텍처(OMA : Object Management Architecture)가 필요하다. <그림 1>에서 보는 바와 같이 OMA는 CORBA를 기반으로 하여 여러 CORBA Facility 및 애플리케이션이 접근하



<그림 1> OMA(Object Management Architecture) [출처 : OMG]

는 또 하나의 상위 계층을 만들어 요소 기반 소프트웨어 환경을 구현하는 시작점으로서의 역할을 한다[18]. CORBA Facility는 두 가지로 나뉜다. 도메인에 관계없이 적용되는 기능에 대한 것이 수평적(Horizontal) CORBA Facility이고, 각 도메인별로 사용되는 인터페이스에 대한 표준을 제정해 가는 것이 수직적(Vertical) CORBA Facility이다.

최근 OMG 내의 워크플로우 관리 Facility 연구팀(Workflow Management Facility Taskforce)에서는 워크플로우 관리 시스템에 대한 CORBA Facility 제안요청서를 발행하였다. 이에 따라서 EDS, Newcastle 대학, jFlow 등 3개의 단체에서 제안요청서에 대한 사양 제안서를 제출하였으며, 표준안 심사 중에 있다[20][21][22]. 이들 사양 제안서는 다음과 같은 내용을 공통적으로 포함한다.

- OMG Common Facility의 활용
- 워크플로우 관리 시스템 개념 모델 제시
- 워크플로우 관리 시스템 객체 지향적 모델링
- 워크플로우 관리 시스템 IDL 정의
- 워크플로우 관리 시스템 개념 및 기술적 이슈

본 연구에서는 이러한 워크플로우와 관련한 OMG 그룹 내의 워크플로우 관리 Facility의 표준화 동향을 파악하고 분석하여 기본적인 워크플로우 관리 시스템 개발을 수행하였다.

2.3 SGML

SGML은 데이터의 영구성과 교환 가능성을 보장해 줄 수 있는 표준일 뿐만 아니라 정보의 구조를 표현하는데 적합한 수단으로서 주목을 받고 있다. SGML은 이기종 간의 효율적인 문서교환이나 임의 형태의 문서 및 그 응용에 대해 일반화된 마크업을 정의하기 위한 방법을 표준화한 메타언어로서 개괄적인 마크업 언어의 구문법을 정의하며, 문서의 복잡한 논리적 계층구조를 기술하는 수단을 제공한다[23]. 항공업체 등과 같은 제조업체 및 법률, 출판 등의 다양한 분야에서 대량의 문서들이 SGML 포맷에 따라 생성되고 있으며, 이러한 기술문서의 생성, 유지, 재생산 등을 위한 문서 관리 시스템에 대한 요구가 커지고 있다. 또한 이러한 대량의 SGML 문서는 곧 체계화된 정보 창고로서 구조적 문서 데이터를 최대한 활용한 정보 검색의 기틀을 제공한다. 즉, 기존의 정보 검색이 문서 내용을 포함한 키워드 형태의 검색인데 반하여 구조 정보 검색에서는 내용과 동시에 문서의 형태에 따라 제목, 목차, 각주 등과 같은 문서의 내부 구조를 지정하여 검색의 범위를 한정할 수 있다. 또한 검색 결과도 사용자의 편의에 따라 목차와 같이 문서 전체를 보거나 문서 구조에 대한 지식 없이 하이퍼텍스트 형태로 항해하는 프리젠테이션 등의 기능을 지원할 수 있다.

SGML 문서의 구성요소는 선언부(Declaration), 문서타입 정의부(DTD : Document Type Definition), 문서 인스턴스(Document Instance)로 나눌 수 있다.

첫째, SGML 선언부는 모든 SGML 문서에 공통으로 사용되며, 생략 가능하다. 문자 집합 및 분리자(Delimiter : <, >, /> 등)로 사용될 문자 정의, Capacity 정의, Syntax 재정의, 지원 가능한 Feature 정의한다. 둘째, 문서타입 정의부에서는 문서의 논리적인 구조를 기술하는데, 이는 문서 내부 또는 외부에 존재 가능하다. 한 단위로 인식될 수 있는 집합을 엔티티(Entity)라 하며, 태그 정의에 사용되는 요소를 엘리먼트(Element), 태그가 가지는 속성을 정의하는 것을 애트리뷰트(Attribute), 특정 문자열을 엔티티로 변환하는 규칙을 빠른 참조(Short Reference), 데이터의 종류를 알리는 기능을 표기법(Notation)으로 기술한다. 마지막으로 구성요소인 문서 인스턴스는 문서를 실제로 기술하는 부분으로 태그를 사용한 계층적인 형태를 띤다.

본 연구에서는 이러한 문서 관리의 표준인 SGML 문서를 이용하여 문서의 내용과 문서 작업 프로세스를 일관성 있게 관리할 수 있도록 하였다. 즉, 문서와 문서 작업 정보를 SGML 문서로 통합

하여 처리할 수 있도록 하기 위하여, 문서 내에 작업 정보를 삽입할 수 있도록 DTD를 설계하여 문서를 작성하도록 하였다.

3. 제품개발 환경 및 프로세스 관리 요구사항

일반적으로 제품 개발 업무에는 많은 참여자들이 공동으로 작업하며, 여러 분야의 전문 지식을 동원해서 문제를 해결해 나간다. 공동 작업이란 여러 작업자가 동일한 자원을 공유하며, 서로 작업 결과를 참조하여 하나의 목표를 향해 상호 작용하는 과정으로 다음과 같은 특징을 갖고 있다.

첫째, 일련의 기준을 이용한 의사 결정 및 추론 작업을 한다. 둘째, 다양한 도구와 시스템을 이용한다. 셋째, 각 작업자는 다른 참여 작업자의 역할과 작업 능력에 대한 이해를 필요로 한다. 넷째, 전체 작업의 조율과 동기화는 각 참여자간의 상호 관계에 크게 의존한다. 마지막으로, 각 참여자는 아주 복잡한 문제를 해결할 수 있는 능력을 갖고 있지만, 다수의 참여자가 공동으로 작업해야만 문제를 해결할 수 있다.

이러한 공동 작업에서 참여자들이 성공적으로 의사결정을 수행하기 위해서는 다양한 공학적 분석 기법, 최적화 기법, 조립용이성 평가 모형, 제조용이성 평가 모형 등을 활용하고, 또한 이러한 기법을 효과적으로 지원하기 위해서 데이터베이스 및 지식베이스를 사용한다. 따라서 참여자들의 의사결정을 도와주고 상호 관계를 조정해 줌으로써 다양한 분야의 전문가들이 협력해서 공동작업을 할 수 있는 환경을 제공하는 것이 필요하다. 특히, 오늘날과 같이 기업이 글로벌화되어 가는 상황에서 같은 제품 개발 프로젝트에 참여하는 모든 참여자들을 동일한 시점에 동일한 장소에서 공동으로 작업하도록 하는 것은 현실적으로 실현 불가능한 경우가 많다. 이러한 공간적, 시간적 한계를 극복하기 위해서는 인터넷으로 대표되는 컴퓨터 네트워크를 이용하여 공동작업을 지원하는 시스템이 필수적이다.

Coopers & Lybrant 사의 설문 조사 자료에 따르면, 제품 개발 프로세스에 참여하는 엔지니어의 업무 중 상당한 부분(70% 이상)이 엔지니어 고유의 창의적인 업무라기 보다는 자료 찾기 및 전달, 재작업 등에 소요되고 있으며, 따라서 엔지니어의 업무 효율을 높이기 위해서 프로세스와 전자적 데이터의 관리가 필요하다는 인식이 점차 확산되고 있다. 이러한 배경에서 제품정보관리 시스템이 등장하여 각광을 받고 있다. PDM 시스템은 모든 공학 데이터(Engineering Data)와 공학 프로세스(Engineering Process)를 관리하는 시스템으로 정의할 수 있다. 이 PDM 시스템의 중심에 위치하여 제품 개발 프로세스를 관리하는 것이 바로 워크플로우 관리 시스템이라 할 수 있다.

워크플로우 관리 시스템은 본래 사무 자동화에서 출발하여 트랜잭션 응용시스템, 스캐닝 이미지 저장 및 처리, 전자문서 관리 등과 같은 업무의 자동화에서 출발한 시스템이다. 따라서 현재의 워크플로우 시스템들은 보험회사의 사고처리 업무, 금융회사의 융자 업무 등과 같이 절차적 정의가 쉬운 업무 프로세스를 관리하는데 많은 도움을 주고 있다. 기존의 워크플로우 시스템은 제품정보관리와 통합된 프로세스 관리에서 출발한 것이 아니기 때문에 이 시스템의 직접적인 도입을 통한 제품 개발 프로세스의 효율적인 관리는 어렵다. 이는 제품 개발 프로세스의 특징이 기존의 워크플로우 시스템에서 관리하던 프로세스의 속성과는 차별화되기 때문이다.

제품 개발 프로세스는 구조적으로 잘 정의하기 어렵고, 한 번 정의된 프로세스를 따라 그대로 프로세스가 진행되기보다는 실행 도중에 예외 상황 발생으로 인해 프로세스의 변경이 필요한 경우가 많다. 즉, 프로세스 정의 단계에서 제품 개발과 관련한 모든 프로세스를 정적으로 정의하여 그대로 따라 가는 것이 아니라, 실행시간에 잦은 변경이 발생한다는 것이다. 이러한 제품 개발 프로세스를 관리하기 위해서는 정적인 프로세스 뿐만이 아니라 동적인 프로세스 관리까지도 통합해서 일관되게 관리할 필요성이 있다.

한편, 최근 제품 수명 주기가 짧아지고 사용자의 요구가 다양해짐에 따라 신속한 제품 개발에 대한 요구가 증가되고 있으며, 설계 및 사무 작업에서 신속하고 정확한 문서작업이 요청되고 있다. 기존의 문서 관리 시스템에서는 문서의 내용과 문서작업과 관련된 정보를 독립적으로 관리하고 있어, 문서의 검색이나 문서작업의 흐름을 파악하여 적절한 업무 조치를 내리는데 많은 시간이 소요되고, 처리 프로세스가 복잡하다는 문제점이 있다. 따라서 문서와 문서 관련 정보를 통합하여 처리함으로써 문서 처리 프로세스의 효율성을 높이고 정보 활용도를 극대화하기 위한 노력으로 본 연구에서는 SGML 문서를 이용하여 이러한 문제점들을 해결하고자 하였다.

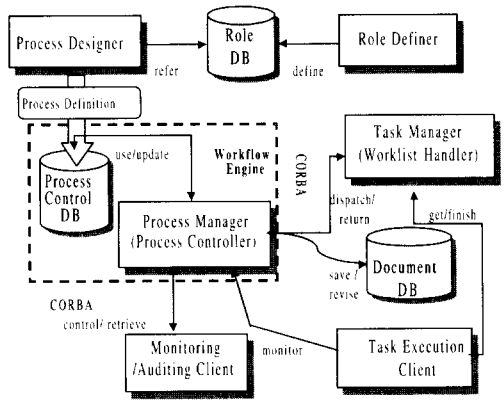
4. 통합 워크플로우 관리 서브컴포넌트

제품 개발 환경에서 발생하는 다양한 속성의 프로세스를 효율적으로 관리하기 위해서 세 가지 워크플로우 관리 시스템을 설계하고 구현하였다. 첫째가 기본 워크플로우 관리 시스템인데, 이것은 WfMC 참조 모델을 따라서 설계되었으며, 모든 워크플로우 관리의 토대가 되는 시스템이다. 둘째는 동적 환경에 대응하기 위한 동적 워크플로우 관리 시스템인데, 이는 바인딩 메커니즘에 근거하여 동적으로 변경 가능한 프로세스를 관리한다. 셋째는 SGML 문서 워크플로우 관리 시스템인데, SGML 문서를 이용하여 문서작업 워크플로우를 정의하고 통제할 수 있는 시스템이다. 이 세 가지 워크플로우 관리 시스템들은 V장에서 설명하게 될 다층 통합 워크플로우 시스템 아키텍처의 토대가 된다.

4.1 기본 WfMS 아키텍처

먼저 CORBA를 기반으로 하는 분산객체 환경에서 동작하는 기본 워크플로우 관리 시스템(Basic Workflow Management System)을 설명하였다.

<그림 2>와 같이 기본 워크플로우 관리 시스템은 프로세스 정의기(Process Designer), 워크플로우 엔진, 워크플로우 클라이언트, 워크플로우 모니터기의 네 가지 모듈로 구성된다. 엔진 모듈과 클라이언트 모듈 사이는 CORBA IDL로 인터페이스를 정의하여 모듈 간의 상호운영성을 구현하였다.



〈그림 2〉 기본 워크플로우 시스템 구조

프로세스 정의기는 그래픽 환경에서 프로세스를 모델링하는 모듈로, 이 모듈을 이용하여 사용자가 업무시스템 전체의 작업흐름을 설계하고, 그래픽하게 설계한 결과를 데이터베이스에 저장할 수 있다. 프로세스 정의기를 통해 정의된 프로세스는 편집 가능한 간단한 PDL(Process Definition Language) 파일로 출력할 수 있고, 이것을 다시 데이터베이스에 저장할 수 있다. PDL을 통하여 프로세스 정의를 분산 환경하에서 파일 단위로 쉽게 교환하거나 공유할 수 있다.

워크플로우 엔진 모듈은 프로세스 데이터베이스에 저장된 프로세스 정의를 이용하여 업무를 담당자에게 할당하고 전체 업무흐름을 감독 통제하는 모듈이다. 수행 대상 태스크를 프로세스 진행 상황에 따라 선택하여 특정 업무담당자의 태스크 관리자(Task Manager)에 할당한다. 태스크 관리자는 워크리스트 핸들러(Worklist Handler)라고도 하는데, 작업자별로 할당되는 태스크들의 내용을 영구적 저장 장치에 저장하여 관리하도록 하였다.

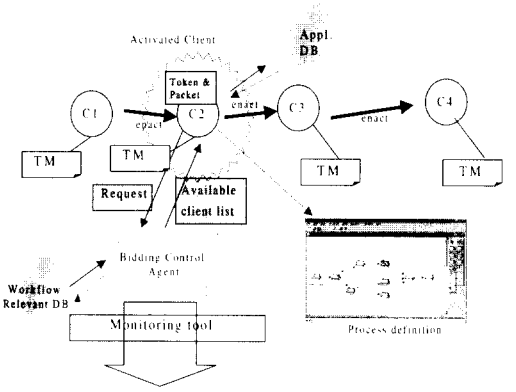
워크플로우 클라이언트 모듈은 모든 사용자에 구현 설치되어 담당자별로 할당된 업무리스트를 관리한다. 클라이언트 모듈은 할당된 태스크들을 관리하는 태스크 관리자와 태스크 정보를 이용하여 실제 업무 수행을 돕는 그래픽 인터페이스 프로그램인 태스크 실행 클라이언트로 구분한다.

워크플로우 모니터기 모듈은 프로세스의 진행 상황을 감독할 수 있게 하고, 수행된 프로세스의 통계 정보 조회 기능을 제공한다. 모니터기 모듈은 현재 진행중인 프로세스 인스턴스별 진행 상황을 그래픽 환경에서 열람할 수 있는 모니터링 기능과 프로세스 인스턴스의 생성 및 시작, 종료 및 통계 정보 열람 기능을 제공하는 감사(Auditing) 기능이 있다. 이 외에도 모니터링된 결과를 바탕으로 한 감독 명령(Supervisory Command) 기능이 있다.

이 기본 워크플로우 관리 시스템을 통해 전통적인 워크플로우 응용 분야의 모든 프로세스를 관리할 수 있다. 정적인 속성을 갖고, 절차적으로 정의가 가능한 프로세스들은 이 기본 시스템을 통해서 관리가 가능하다. 동적인 워크플로우 및 SGML 기반의 워크플로우에 대한 프로세스 통제 데이터의 관리와 모니터링 정보 제공 또한 이 기본 시스템에서 수행하는 기능이다. 이 기본 시스템을 이용하여 관리되는 대상 프로세스는 비교적 간단하면서 반복성이 있는 프로세스이다. 제품 개발 과정 또한 일상적인 업무 프로세스들이 발생하며, 일상적인 프로세스의 예에는 소요 물품의 조달 프로세스, 회의 소집 프로세스, 특정 사양의 승인 및 결재 프로세스 등이 있다. 이러한 프로세스는 한 번 정의되면 잘 변경되지 않고 반복적으로 프로세스 인스턴스가 발생하는 특징이 있다.

4.2 비딩 메커니즘 기반의 동적 워크플로우 관리 시스템

동적이며 적응성을 지닌 워크플로우 관리 시스템은 실행 시에 프로세스의 변화를 반영할 수 있어야 한다. 이를 위해서 태스크를 처리 개체에 재할당 할 수 있어야 하는데, 이전의 연구들은 이러한 유연성을 확보하기 위해서 대안의 경로(alternative routing)를 미리 설정해 두는 방법과 스케쥴을 담당하는 엔진을 분산하는 방식을 사용하였다[9]. 이러한 방식은 실행 시에 정확한 프로세스의 정의 및 스케쥴의 생성을 요구하는데, 프로세스 정의 시 기능에서 완벽한 스케쥴의 생성을 보장하지 못하는 경우가 많다. 비딩 메커니즘 기반의 동적 워크플로우 관리 시스템에서는 프로세스 정의를 바탕으로 실행 시에 자치성을 지니는 각 처리 개체(Processing Entity)들이 비딩(Bidding) 메커니즘을 이용하여 태스크를 다음 처리 개체로 할당하는 분산구조를 이용한다. 이를 그림으로 나타내면 다음 <그림 3>과 같다.



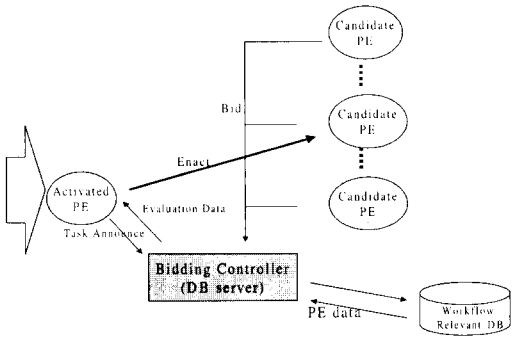
〈그림 3〉 비딩 메커니즘 기반의 동적 워크플로우 관리 시스템 구조

비딩 메커니즘 기반의 동적 워크플로우 관리 시스템의 가장 큰 특징은 처리 개체의 독립성이다. 즉, 태스크의 수행을 담당하는 처리 개체들이 자치적으로 프로세스를 해석해서 다음 작업을 처리할 개체를 선정하며 프로세스를 진행하는 것이다. 이를 위해서 프로세스 정의기를 사용하여 프로세스를 정의해 두고 각 처리 개체들이 실행 시에 이를 해석하며, 필요한 데이터와 활성화 정보들을 패킷으로 전달하도록 한다. 이에 따라 서버로서의 엔진의 역할은 비딩 통제기(Bidding Controller)로 바뀌게 되며, 처리 개체는 자치성을 지니면서 이후

시스템 확장성을 보장 받는 모듈이 된다. 비딩 통제기는 프로세스 진행의 모니터링, 다음 태스크를 처리할 개체 리스트의 전달, 워크플로우 관련 데이터의 유지, 처리 개체의 요청에 따른 태스크 인스턴스의 생성 및 전달 등의 역할을 수행한다. 처리 개체는 프로세스 정의의 해석, 다음 처리 개체의 선정 및 구동, 필요한 데이터의 전송, 워크플로우 통제 데이터의 수정, 필요한 응용 프로그램의 호출 등의 기능을 수행한다.

동적 워크플로우 관리기를 통해서 프로세스의 정의에 따른 워크플로우의 진행을 개체들이 실행 시에 수행하도록 함으로써 사실상 기존의 엔진 개념을 없앴으며, 처리 개체들이 자치적으로 프로세스를 진행함으로써 실행 시 프로세스 변경을 반영할 수 있도록 하였다.

워크플로우 엔진의 중요한 기능중의 하나는 태스크를 수행하기 위한 스케줄의 생성이며, 동적 워크플로우의 지원을 위해서는 실행 시에 스케줄이 생성되고 진행되는 동적 스케줄이 필요하다. 일반적으로 동적인 스케줄의 생성은 오프라인 스케줄보다 온라인 스케줄에 의해 효과적으로 달성될 수 있으며, 이는 워크플로우 관리 시스템에도 적용이 된다. 동적 워크플로우 관리 시스템에서 비딩에 의한 프로세스의 진행 과정은 <그림 4>와 같이 이루어진다.



〈그림 4〉 비딩에 의한 프로세스 진행과정

최초의 태스크를 수행할 처리 개체가 활성화되고 나면 이 처리 개체는 작업의 완료시점에서 비딩 통제기에 작업완료에 따른 다음 태스크 선정을 요청한다. 비딩 통제기는 데이터베이스에 있는 각 처리 개체들의 상태를 파악하고, 이에 따라 가능한 개체들에 수행되어야 할 태스크를 통보한다. 통보를 받은 후보 처리 개체들이 비드(Bid)를 보내고, 이를 현재의 처리 개체가 받아서 평가를 한 후 다음 태스크를 수행할 처리 개체를 선정하여 태스크를 이양한 뒤 작업을 끝낸다.

활성화된 하나의 처리 개체가 주어진 태스크를 끝낸 상태로부터 다음 태스크를 처리할 처리 개체를 선정하여 구동하기까지의 단계를 요약하여 정리하면 다음과 같다.

[단계 1] 후행 태스크의 확인
처리 개체가 주어진 태스크 수행을 완료하고, 프로세스인스턴스 데이터베이스를 검색하여 다음 태스크를 확인한다. 태스크의 처리 프로세스명을 확인한다.

[단계 2] 태스크 비딩 요청
워크플로우 데이터베이스에서 주어진 태스크를 처리할 수 있는 처리 개체의 리스트를 넘겨받는다. 리스트에 있는 처리 개체들에게 주어진 태스크에 대한 비드를 요청한다.

[단계 3] 후보 처리 개체들의 비딩
요청을 받은 후보 처리 개체는 주어진 태스크에 대해 요구된 자신의 데이터를 선행 처리 개체로 보냄으로써 비드를 낸다.

[단계 4] 비드에 대한 선호도 평가
현재의 처리 개체가 넘겨진 데이터들을 취합하고 이를 평가한다. 선호도 평가는 다음에 설명되는 선호도 평가 함수를 이용하여 수행된다.

[단계 5] 다음 처리 개체의 구동
평가 결과를 바탕으로 다음 처리 개체에 태스크 실행 명령을 전달하여 구동시킨다.

[단계 4]에서 이루어지는 비딩 통제기에서 후보

처리 개체들에 대한 선호도는 현재 후보 처리 개체에 할당되어 있는 태스크들의 처리 시간의 합, 대상 태스크의 작업 기한, 두 처리 개체간의 거리의 함수로 표현되며, 이 함수는 워크플로우의 특성과 각 태스크별로 다르게 설정될 수 있다. 이를 수식으로 나타내면 다음과 같이 표현된다. 아래 수식 설명에서 PE는 처리 개체를 의미한다.

$PR_i = f(DD, PT_i, D_i)$

DD : 대상 태스크의 $DueDate$

P_i : i번째 후보 PE에 있는 태스크들의 처리 시간의 합

D_i : 현재 태스크를 수행중인 PE와 대상태스크간의 거리

여기서 D_i는 현재 태스크를 수행중인 처리 개체와 비드에 참여한 태스크 처리 개체와의 거리를 의미하는데 이 거리는 물리적인 거리를 의미할 수도 있고, 또는 처리 개체 간의 작업 연관성을 나타내는 값을 의미할 수도 있다. 실제 현장의 요구사항에 따라 D_i 값을 상황에 맞게 정의할 수 있다. 또한 함수 자체도 환경에 따라 여러 가지로 정의될 수 있으며, 실행시간에 수행도 분석을 통해 수정이 가능하다.

비딩 메커니즘에 기반한 워크플로우 관리 시스템은 앞에서 설명한 바와 같이 미리 정의하기 어렵거나 전체 프로세스를 시작하기 이전에 완벽하게 설계하기 어려운 프로세스의 워크플로우 관리를 가능하도록 해 준다. 제품 개발 환경에서 발생하는 많은 프로세스들은 이러한 특성이 있다. 제품 개발 환경에서 발생하는 많은 프로세스들에서 프로세스의 진행 시 변경 요구가 발생한다. 예를 들면, 특정 부품에 대한 설계 변경 요청 작업과 변경 사항 검토 및 변경안 제출, 승인 등으로 구성되는 프로세스를 디자인 단계에서 정의하여 실행시킬 때, 실행 시 변경 요구가 발생하여 프로세스 정의를 변경해야 하는 경우가 종종 있다. 이러한 프로세스는 자주 발생하기는 하지만 동일한 과정을 거쳐 진행되지 않는 경우가 많고, 수행 작업자도 상황에 맞게 변경되는 경우가 많다.

실행 시간에 발생하는 모든 변화 가능성을 미리 예측하여 프로세스 모델링을 수행하는 것은 현실적으로 불가능하며, 만일 예상 가능한 모든 경우를 반영하여 프로세스를 모델링할 경우 프로세스 정의 자체가 너무 복잡해져 이해하기 힘들고 실행 시 통제하고 관리하는 것 또한 어렵게 된다. 제품 개발 환경과 같이 프로세스 변경 요구가 많은 영역에서 기존의 정적인 워크플로우 관리 시스템을 이용하여 프로세스를 관리하는 경우 이상에서 설명한 바와 같이 프로세스의 모델링이 어렵고, 모델링된 프로세스의 실행 및 통제가 어렵다.

비딩 메커니즘을 사용하는 동적 워크플로우 관리 시스템을 적용하면, 동적 변경 요구가 발생할 때 프로세스 실행 중간에 프로세스 정의만 변경해 주면 된다. 태스크 실행 클라이언트는 자치적 처리 개체이기 때문에 영향을 받지 않고 계속해서 프로세스의 진행이 가능하다. 따라서, 비딩을 이용한 워크플로우 관리기는 기존의 워크플로우 시스템에 비해 동적인 측면과 적응성의 측면에서 다음과 같은 이점을 가진다.

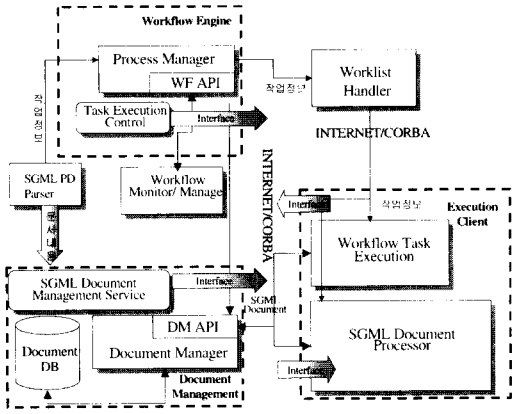
- 프로세스 자체의 변경이 가능하므로 시스템 오동작에 대처할 수 있다.
- 태스크의 수행 경로를 유연하게 설정할 수 있다.
- 처리 개체들의 추가에 의한 시스템 확장이 용이하다.

4.3 SGML 문서 워크플로우 관리 시스템

본 연구에서는 문서 작업 프로세스와 문서 작업 내용을 SGML 문서로 통합 관리하고자 SGML 문서에 기반한 워크플로우 관리 시스템을 개발하였다. SGML 문서에 기반한 워크플로우 관리 시스템은 문서와 문서 관련 정보를 통합하여 처리함으로써 문서 처리 프로세스의 효율성을 높이고 정보 활용도를 극대화하기 위해 고안되었다.

4.3.1 SGML 문서 관리 시스템 구조

SGML 문서에 기반한 워크플로우 관리 시스템은 <그림 5>와 같이 CORBA 기반의 클라이언트/서버 구조로 이루어져 있다. 이 시스템은 크게 SGML 정의 워크플로우 관리 엔진과, 문서 관리 시스템 서버, 태스크 실행 클라이언트로 구성된다.



〈그림 5〉 SGML 기반 워크플로우 관리 시스템 아키텍처

이 시스템에서 워크플로우 엔진은 SGML로 정의된 프로세스를 해석하여, 문서 내용은 SGML 문서 관리 시스템을 통해 관리하고, 작업 정보는 엔진의 프로세스 관리자를 통해 클라이언트 쪽의 SGML 워크리스트 핸들러에 넘겨 준다. 태스크 실행 클라이언트는 작업 관련 정보와 작업 문서 내용이 통합된 SGML 문서를 가지고 작업을 수행한다. 작업이 완료되면 작업한 SGML 문서를 첨부하여 워크플로우 엔진에게 작업 결과를 통보한다. 문서 내용을 다루는 문서 관리 시스템은 저장 및 수정 모듈, 검색 모듈, 문서화(documentation) 모듈, 태그 제공 모듈로 구성된다. 파싱된 문서 내용은 저장 및 수정 모듈을 통해 데이터베이스에 저장되고, 재작업 시에는 이 모듈을 이용해 기존의 내용을 변경한다. 검색 모듈은 수정, 또는 문서화 작업 시 저장되어 있는 객체를 검색하는 기능을 하며, 문서화 모듈은 데이터베이스 내에 산재하는 문서 내용을 모아 DTD에 맞도록 문서를 만들어 제공하는 역할을 수행한다. 태그 제공 모듈은 작업 명령 시 각 구조별로 할당된 부분의 엘리먼트를 나타내는 태그를 제공한다.

4.3.2 SGML 문서와 문서 처리 작업 프로세스의 통합

SGML 표준을 따라서 작성하는 문서 작업의 경우 작업 프로세스 자체를 SGML을 이용하여 정의하고, 엔진이 SGML 프로세스 정의를 파싱하여 작업 순서를 스케쥴링하고 작업을 할당한다. SGML 문서 작업 프로세스를 관리하는 전체 처리 절차는 다음과 같다.

[단계 1] 클라이언트 작업 결과 모니터링

클라이언트 작업 결과 모니터링 모듈을 통해 실시간으로 클라이언트로부터의 작업 결과를 점검하면서 대기하다가 결과가 도착하면 엔진이 SGML 문서를 파서에 보낸다.

[단계 2] 문서 파싱

도착된 SGML 문서에서 문서내용과 작업정보를 분리하여 API를 통해 워크플로우 통제 시스템과 문서관리 시스템에 보낸다.

[단계 3] 파싱된 정보 처리

파싱된 작업 정보는 워크플로우 통제 시스템에서, 문서내용은 문서관리 시스템에서 동시에 처리된다.

첫째, 작업 정보는 다음과 같이 처리된다. 작업이 완료된 경우에는 작업 리스트를 갱신하고 후행 작업 정보를 검색하여 작업명령을 통지하도록 관련 정보를 넘겨준다. 선행 작업에 대한 승인정보에 따라 선행작업의 완료처리, 또는 재작업 명령을 통지하도록 하고 재작업의 경우 작업 리스트를 갱신한다.

둘째, 문서 내용 처리 방법은 다음과 같다. 각 엘리먼트별로 파싱된 문서 내용은 정의된 문서 스키마에 따라 구조별로 저장

된다. 객체로 관리되는 문서 내용은 문서 요구 시 DTD에 맞게 문서화되어 제공된다.

[단계 4] 작업명령 통지

작업내역과 관련 작업에 대한 태그를 넘겨받아 작업자에게 발송하여 작업명령을 수행하도록 한다.

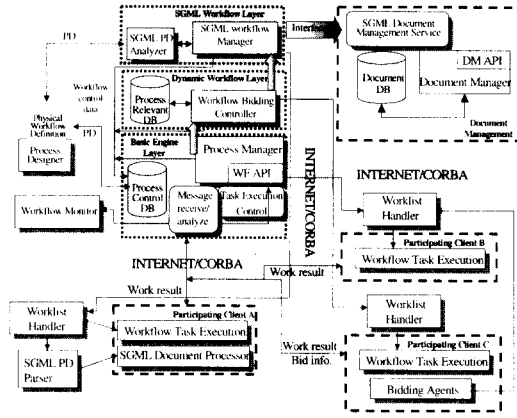
SGML 문서 기반 워크플로우 관리 시스템은 SGML 문서를 공동 작업을 통해 작성해야 하는 경우에 적용이 가능하다. 예를 들면, 제품 정보 회보(Product Advisory) 문서를 여러 작업자가 문서의 일부분을 맡아서 공동 문서 작업을 통해 최종 문서를 완성하는 프로세스에 이 시스템을 적용할 수 있다. 제품 정보 회보는 본래 회보 번호, 타입, 발간 날짜, 수정 날짜, 제품명으로 구성된 헤더와 제목, 서브섹션, 승인여부로 구성되어 있다. 본 연구에서는 이러한 제품 정보 회보를 대상으로 문서 내용과 워크플로우 작업 정보를 통합해서 SGML 문서로 작성할 수 있도록 DTD를 설계하였고, 워크플로우 관리 시스템을 통해 문서를 완성할 수 있도록 하였다.

위와 같은 SGML 문서 작업을 위한 프로세스를 관리하기 위해서 기존의 워크플로우 관리 시스템과 SGML 문서 관리 시스템을 각기 따로 도입하여 운영할 경우에 가장 큰 제약점은 작업 결과로 생기는 문서와 문서 작업 프로세스의 일관된 관리가 불가능하다는 것이다. 즉, 문서 작업 처리를 위한 프로세스 정의를 워크플로우 관리 시스템의 프로세스 정의기를 이용하여 작성하여야 하고, 작업 결과로 생기는 응용 데이터(Application Data)로서의 SGML 문서는 별도로 문서 관리 시스템에서 저장 및 관리해야 한다. 본 연구에서 개발한 시스템을 이용하여 문서와 작업 정보를 SGML 문서로 일관되게 관리함으로써 관리의 효율성이 증대되고, 전체 문서 작업을 완료하는데 걸리는 시간이 단축된다. 또한 구조화된 SGML 문서를 이용함으로써 문서 검색의 효율이 높아지고, 수정 및 유지 보수가 용이하다.

5. 다층 아키텍처를 통한 시스템 통합

본 절에서는 앞 절에서 설명한 세 가지 워크플로우 관리 시스템 즉, 기본 워크플로우 관리 시스템과 비딩 메커니즘에 기반한 동적 워크플로우 관리 시스템 및 SGML 워크플로우 관리 시스템을 각각 층(Layer)으로 통합하는 아키텍처에 대하여 설명하고 이들 시스템들이 통합되는 메커니즘을 제시하고자 한다.

<그림 6>은 다층 통합 워크플로우 시스템 아키텍처를 보여 주고 있다.



〈그림 6〉 다층 통합 워크플로우 시스템 아키텍처

<그림 6>에서 보는 바와 같이 전술한 세 가지 워크플로우 관리 시스템이 각각 하나의 층으로 자리 잡고 있다. 세 가지 층 중에서 가장 하부에 있고 근간이 되는 층이 기본 워크플로우 층이며, 이 층의 기능은 4.1 절에서 상술한 바와 같이 프로세스 통제 데이터를 관리하고 태스크 실행 클라이언트에게 작업을 할당하기 위해 워크리스트 핸들러에게 작업을 할당하는 기능을 수행하며, 또한 프로세스 정의 해석 기능과 워크플로우 모니터기와의 인터페이스 기능을 제공한다. 미리 정의하기 어렵거나, 잦은 변경이 발생하는 프로세스들의 경우 4.2 절에서 상술한 바와 같이 두 번째 층인 동적

워크플로우 관리 시스템 층에서 비딩 메커니즘을 통해 관리한다. 이 경우 참여 클라이언트 C에서 보는 것처럼 클라이언트는 비딩에 참가하는 에이전트가 되고, 비딩에서 성공할 경우 해당 태스크를 수행하게 된다. 다층 통합 워크플로우 시스템 아키텍처에서 최상위에 있는 층이 SGML 문서 관리 층인데, 이는 4.3 절에서 설명한 바와 마찬가지로 문서의 내용과 문서 작업 프로세스를 SGML 문서를 통해 일관되게 관리하기 위한 층이다. 이 층에서는 SGML로 정의된 프로세스를 해석하여 태스크 수행 클라이언트를 결정하고, 작업 정보와 작업 문서 내용이 통합된 SGML 문서를 전달해 준다. 이 때 파서를 이용하여 SGML 문서에서 작업 정보를 분리하여 문서 작업을 통제하고, 문서 내용은 SGML 문서 관리 시스템을 통해 관리한다.

제품 개발 환경에서 구동되는 모든 물리적 워크플로우가 프로세스 정의기를 통해서 정의된다. 즉, 기본 워크플로우 층에서 관리되는 정적 워크플로우와 비딩에 기반한 동적 워크플로우 층에서 관리되는 워크플로우, SGML 문서로 정의되는 SGML 워크플로우 등 모든 워크플로우는 프로세스 정의기를 통해서 정의된다.

다층 통합 워크플로우 시스템에 참여하는 클라이언트는 크게 세 가지 클라이언트 기능을 갖고 있으며, 자신에게 할당되는 작업의 내용에 따라 해당되는 클라이언트 인터페이스를 통해 엔진과 상호작용한다. 첫째가 정적 프로세스에 속하는 태스크를 처리하는 일반 클라이언트 인터페이스이며, 둘째는 비딩 에이전트를 통해 비딩에 참여하며, 동적 워크플로우에 참여하는 비딩 클라이언트 인터페이스이고, 마지막으로 워크리스트 핸들러를 통해 SGML 문서를 받아서 작업하는 SGML 문서 처리 클라이언트 인터페이스가 있다.

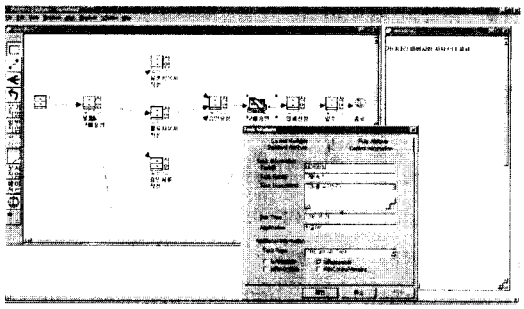
작업에 참여하는 클라이언트들이 작업 결과를 기본 엔진을 통해 전달하면, 기본 엔진의 메시지 수신 및 해석기가 결과 메시지를 수신한다. 프로세스 통제 데이터베이스 내용을 갱신한 후, 도착한 메시지 내용에 따라 기본 엔진을 통해 다음 태스크를 진행하거나, 상위 층인 동적 워크플로우 관리 시스템 혹은 SGML 워크플로우 관리 시스템을 통해서 다음 태스크를 진행하게 된다.

6. 구현 및 적용

본 연구에서 제안한 다층 통합 워크플로우 시스템은 이기종의 하드웨어 및 운영체제 상에서 여러 종류의 소프트웨어 개발 도구들을 CORBA를 통하여 통합하였다. 기본 워크플로우 엔진 모듈과 데이터베이스는 Solaris 2.5 운영체제가 동작하는 SUN 워크스테이션에서 구현하였다. Microsoft Windows 95/98, 또는 Windows NT 운영체제에서 운영되는 클라이언트들에는 프로세스 정의기, 워크플로우 모니터기, 태스크 실행 클라이언트가 있다. 프로세스 정의기는 Microsoft Visual C++를 사용하여 구현하였다. 기본 워크플로우 엔진 모듈과 동적 워크플로우 비딩 통제기, SGML 워크플로우 관리 시스템은 객체지향 데이터베이스인 Objectivity/DB와 IONA Orbix MT2.2를 통합하여 C++로 구현하였다. 태스크 실행 클라이언트는 PC 상에서 IONA Orbix Desktop2.2와 Visual C++을 사용하여 구현하였으며, 모니터 모듈은 PC 상에서 Orbix Desktop2.2와 Visual C++을 사용하여 구현하였다. SGML 문서를 관리하는 모듈은 Objectivity/DB와 IONA Orbix MT2.2를 이용하여 구현하였다.

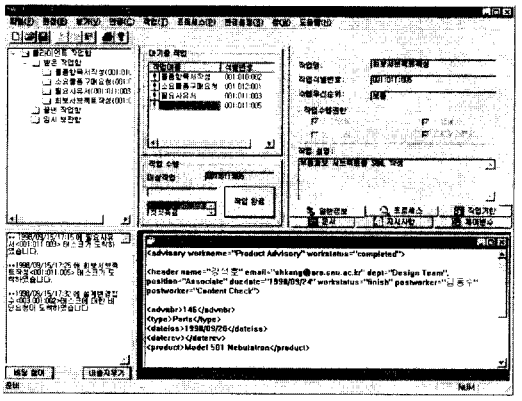
본 연구에서 개발한 다층 통합 워크플로우 시스템의 구현을 통해 기존의 정형적인 프로세스 관리뿐 아니라 동적인 프로세스 관리가 통합되어 구현 가능하였으며, SGML에 기반한 문서 처리 프로세스 등과 같이 제품 개발 환경에서 발생하는 모든 타입의 프로세스를 일관되게 관리하는 것이 가능하였다.

다음의 그림들은 실행 화면 중에 주요한 기능을 보여 주는 일부 화면들을 보여 주고 있다. 주로 Windows 기반의 클라이언트 프로그램 위주로 나타내었으며, 나머지 모듈들은 유닉스 플랫폼에서 서버로서 작동하고 있다.



〈그림 7〉 프로세스 정의기 실행 화면

<그림 7>은 프로세스 정의기를 이용하여 제품 개발 환경에서 발생하는 물품 구매 프로세스를 설계하는 과정을 보여 주고 있다. 태스크를 나타내는 노드와 컨트롤의 흐름을 나타내는 아크로 구성되며, 노드와 아크 상에서 워크플로우 구동에 필요한 다양한 애트리뷰트를 정의할 수 있다. 위 그림의 예에서는 태스크 노드의 애트리뷰트들을 입력하는 것을 보여 주고 있다. 모든 프로세스 설계는 이 프로세스 정의기를 통해서 이루어진다.



〈그림 8〉 태스크 실행 클라이언트 실행 화면

<그림 8>은 업무담당자가 컴퓨터와의 인터페이스를 통해 직접 업무를 수행하는 화면이다. 이 화면에서 보는 바와 같이 태스크 실행 클라이언트에는 기본 클라이언트 기능에 덧붙여 SGML 문서 편집 기능과 비딩 참여 기능이 포함되어 있다. 비딩에 참여하기 위한 인터페이스가 좌측 하단에 제공되고 있으며, 클라이언트에게 도착한 작업 정보와 작업 수행을 위한 각종 인터페이스가 제공되고 있다. 또한, 클라이언트 프로그램 내의 문서 편집기를 이용하여 SGML 문서 처리 작업인 경우 문서를 직접 편집할 수 있다.

이 이외에도 워크플로우 모니터기를 통해 작업 수행에 관한 정보를 모니터링할 수 있다. 다양한 유형의 프로세스들이 통합되어 진행되고 있는 프로세스에 대한 일관적인 모니터링 정보 획득 및 관리를 위해 제공되는 모듈로서 프로세스와 태스크, 작업자 수준의 모니터링 기능과 관리자 차원의 감독 기능으로 구성되어 있다. 프로세스 진행상황을 그래픽하게 모니터링할 수 있으며, 프로세스 통계 정보 및 지체 프로세스 검색, 태스크 상태 정보 및 통계 정보, 작업자 관리 정보 등을 열람하는 기능이 있다.

3장에서 살펴 본 바와 같이 제품 개발 환경에서 발생하는 프로세스의 종류와 그 특성은 매우 다양하다. 즉, 일반적으로 흔히 워크플로우화해서 자동화할 수 있는 정적 프로세스, 실행 시에 잦은 변경이 발생하는 동적인 프로세스, SGML 표준을 이용한 문서 작업 프로세스 등이 그 예이다. 이와 같이 다양한 워크플로우 유형을 단일 시스템으로 관리하기 보다는 각 유형의 프로세스 관리를 위한 서브컴포넌트를 층으로 만들고 이들을 통합할 수 있는 시스템을 통해 제품 개발 환경에서 발생하는 프로세스를 일관되고 통합된 관점에서 관리할 수 있다.

7. 결 론

본 연구에서는 제품 개발 프로세스를 효율적으로 관리하고 지원하기 위한 다층 워크플로우관리 시스템 아키텍처를 구성하는 서브 컴포넌트들을 설계하고 구현하였으며, 이들을 통합하여 운영하기 위한 통합 아키텍처 운용방안을 제시하였다. 제품 개발 환경을 프로세스 관점에서 지원하여 업무의 흐름 및 정보에 대해 통제하고 관리할 수 있도록 하기 위해서 기존의 워크플로우 관리 시스템에

서 추구하는 기능을 근간으로 하였으며, 여기에 덧붙여 중앙 집중식 엔진 중심의 구조에서 오는 문제점을 해결하고, 동적인 워크플로우를 관리하기 위하여 비딩 메커니즘에 기반한 워크플로우 처리를 가능하도록 하였다. 또한, 문서 관리 표준으로 정착되고 있는 SGML 문서와 공동 문서 작업 프로세스의 통합 관리를 위해 SGML 문서로 공동 문서 작업 프로세스를 정의하고, 처리할 수 있도록 하는 SGML 기반의 워크플로우 관리 시스템을 SGML에 기반한 문서 데이터 작업을 관리할 수 있도록 하였다.

본 연구의 결과로 얻어진 다층 워크플로우 관리 시스템의 적용을 통해 얻을 수 있는 장점은 다음과 같다.

첫째, 제품 개발 과정에서 발생하는 여러 가지 프로세스를 일관된 관리 시스템을 통해서 관리할 수 있다. 정형적인 프로세스 관리뿐만 아니라, 비정형적이며, 동적인 프로세스들까지도 일관성 있게 관리가 가능하다.

둘째, 오동작에 대한 대처가 가능하다. 비딩 메커니즘에 기반한 동적 워크플로우 관리 시스템을 통해 프로세스 자체의 변경이 가능하므로 프로세스의 변경을 통해 대처할 수 있다.

셋째, 비딩 메커니즘에 기반한 동적 워크플로우 관리 시스템을 통해 유연한 경로 설정이 가능하다. 워크플로우의 프로세스에서 태스크들이 거쳐야 할 처리 개체들이 미리 설정되지 않고 실행 시에 자치적으로 결정되므로 경로의 유연성이 최대한 보장된다.

넷째, 문서 관리 작업 프로세스와 문서 작업 내용을 SGML 문서로 통합 관리할 수 있다. 즉, 문서 작업을 SGML 표준을 이용하여 정의할 수 있고, 작업 내용 또한 이 정의와 함께 일관성 있게 관리할 수 있다.

다섯째, 모든 서브컴포넌트들을 분산객체관리의 표준인 CORBA를 이용하였기 때문에 상호운영성과 확장성이 보장된다. 본 연구에서 개발한 모든 요소들은 이질 분산 환경의 다양한 정보 시스템 간의 통합이 용이하다.

참 고 문 헌

[1] L. Fischer, The Workflow Paradigm - The Impact of Information Technology on Business Process Reengineering, Future Strategies, *Inc., Alameda*, CA, 2nd. edition, 1995.

[2] C.Mohan, Tutorial : State of the Art in Workflow Management System Research and Products, *IBM Almaden Research Center, Workflow Systems and Interoperability*, Istanbul. Turkey, August, 1997.

[3] David Hollingsworth, Workflow Management Coalition Specification : The Workflow Reference Model, *WfMC specification*, 1994.

[4] Jintae Lee, Michael Gruninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost, The PIF Process Interchage Format and Framework v1.1, *PIF Working Group*, May 24, 1996.

[5] Graig Schlenoff, Amy Juntilla, Steven Ray, Unified Process Specification Language : Requirements for Modeling Process, *NIST*, 1996.

[6] C.Mohan, G.Alonso, R. Günthör, M.Kamath, Exotica : A Research Perspective on Workflow Management Systems, *Data Engineering*, Vol.18, No.1(1995).

[7] Amit Sheth, Decashish Worah, Krys Kechut, John Miller, Ke Zheng, Devanand Palaniswami, Souvik Das, The METEOR Workflow Management System and Its Use in Prototyping Significant Healthcare Applications, *Technical Report, Large Scale Distributed information Systems Lab*, The University of Georgia, 1995.

[8] John A, Miller, Devanand Palaniswami, Amit P. Sheth, Krys J, Kochut and Harvinder Singh, WebWork : METEOR2's Web-based Workflow Management System, *Technical Report, Large Scale Distributed information Systems Lab*, The University of Georgia, 1997.

[9] S. Das, K. Kochut, J. Miller, A. Sheth, D. Worah, ORBWork : A Reliable Distributed CORBA-based Workflow Enactment System for METEOR2, *Technical Report, Large Scale Distributed information Systems Lab*, The University of Georgia, 1997.

[10] Walt Scacchi, John Noll, Process Driven Intranets : Life-Cycle Support for Process Reengineering, *IEEE Internet Computing 97/Sep,Oct*, 1997.

[11] Munindar P. Singh, Formal Sementics for Workflow Computations, *Proc of the 5th Intl. Workshop of DBPL*(1995).

[12] Dirk Wodtke, Jeanine Weissenfels, Gerhard Weikum, Angelica Kots Dittrich, The Mentor Project : Steps Towards Enterprise Wide Workflow Management, *Technical Report*, 1996.

[13] Esin Gokkoca, Mehmet Altinel, Ibrahim Cingil, E. Nesime Tatbul, et al., Design and Implementation of a Distributed Workflow Enactment Service, *Proc. of Intl. Conf on Cooperative Information Systems*(1997).

[14] Stefano Ceri, Paul Grefen, Gabriel Sahchez, WIDE-A Distributed Architecture for Workflow Management, *Technical Report*, 1997.

[15] Gail E. Kaiser, George T. Heineman, Peter D. Skopp, Jack J. Yang, Incremental Process Support for Component-based Software Engineering, *Technical Report*, Columbia University, 1997.

[16] Tohmas W. Malone, Kevin Crowston, Jintae Lee, Brian Pentland, Toward a Handbook of Organizational Process, *Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative*(1993).

[17] Sape Mllender, *Distibuted Systems*, second edition, Addison Wesley, 1995.

[18] Jon Seigel, CORBA Fundamentals and Programming, *OMG*, 1996.

[19] OMG, The Common Object Request Broker : Architecture and Specification, *OMG*, 1996.

[20] Electronic Data Systems Corporation, EDS Workflow Management Facility, *OMG Document Number bom/97-08-06*, OMG, 1997.

[21] University of Newcastle upon Tyne, Workflow Management Facility Specification, *OMG document number 98-01-11*, OMG, 1998.

[22] jFlow, Workflow Management Facility, *OMG*, 1997.

[23] Charles F. Goldfarb, *The SGML Handbook*, Oxford Univ. Press, 1990.