워크케이스 기반의 초대형 워크플로우 시스템 아키텍처

Workcase based Very Large Scale Workflow System Architecture

심 성 수 김 광 훈
Sung-Soo Shim Kwang-Hoon Kim
e-mail : {s3im,kwang}@kyonggi.ac.kr
Department of Computer Science Kyonggi University

요 약

워크플로우 관리 시스템은 정부나 기업과 같은 조직의 작업을 처리하기 위한 비즈니스 프로세스를 컴퓨터를 기반으로 자동화 함으로서 작업의 효율을 높이고 비용을 절감한다. 현재에 이르러 이런 워크플로우 시스템을 사용하는 조직들이 점차 거대화되어 가고 네트워크의 발달과 인터넷의 출현으로 인하여 워크플로우 시스템이 처리하여야 하는 작업의 수와 고객과 작업자 수 등이 빠른 속도로 증가하는 추세이다 이런 추세에서 워크플로우 시스템은 거대 조직 환경에 적합한 워크플로우 시스템 아키텍처를 필요하게 된다. 이에 본 논문은 거대 조직 환경을 관리할 수 있는 워크플로우 관리 시스템으로 워크케이스 기반의 초대형 워크플로우 시스템의 아키텍처를 설계 및 구현 하고자 한다. 그리고 워크플로우 시스템 아키텍처를 분류, 분석하여 장단점을 가려내어 이를 기반으로 워크플로우 시스템 아키텍처의 성능을 예측하여 워크케이스 기반 워크플로우 시스템 아키텍처가 본 논문에서 제안하는 초대형 워크플로우 시스템의 아키텍처라는 것을 예측하여 본다. 또한 초대형 워크플로우 시스템을 위한 하부 구조로 EJB(Enterprise Java Beans)를 사용하고 사용 이유를 기술한다. 본 논문에서는 이런 워크케이스 기반의 초대형 워크플로우 시스템 아키텍처를 위하여 개념적인 단계와 설계 단계,구현 단계로 나누어 설계 및 구현을 하며 개념적인 단계에서는 워크케이스 기반 워크플로우 시스템 아키텍처에 대하여 상세히 기술하고 설계 단계에서는 전체적인 기능 정의와 초대형 워크플로우 시스템의 구조를 설계한다. 그리고 구현 단계에서는 워크케이스 기반의 초대형 워크플로우 시스템 아키텍처를 실제 구현하기 위한 환경을 선택하고 구현 단계의 문제점들과 해결책을 기술한다.

1. 서 론

워크플로우는 조직에서 작업을 처리하는 프로세스인 비즈니스 프로세스를 컴퓨터 기반 위에서 자동화하여 처리하는 것이다. 이런 워크플로우를 처리하기 위하여 분산된 작업 환경 하에서 조직이 작업을 처리 할 수 있도록 협업과 작업 모니터, 그리고 다양한 작업들의 실행을 지원하기 위하여 개발된 시스템이 워크플로우 관리 시스템이다. 이런 워크플로우 관리 시스템은 시스템을 구성하는 서버의 수나 작업을 처리하는 사용자들의 클라이언트 컴퓨터, 시스템 내부에서 프로세스를 처리하는 방식과 데이터, 자동화된 프로그램을 처리하는 방식, 사용자들에게 모니터링을 제공하는 방식, 그리고 다른 워크플로우 시스템간의 협업에 이르기까지 여러 가지 복잡한 요소들을 가지게 되어 개발되는 워크플로우 시스템마다 다른 형태의 아키텍처를 가지게 된다. 특히 네트워크의 발달과 인터넷을 기반으로 하는 환경의 출현 그리고 워크플로우 관리 시스템을 사용하는 조직의 거대화는 조직의 워크플로우 시스템을 대량의 사용자와 서버, 데이터라는 환경에 들어가게 함으로서 워크플로우 관리 시스템이 관리할 수 있는 수 이상의 거대량의 작업들을 처리해야 하는 문제를 낳게 된다. 이런 문제를 해결하기 위하여 거대량의 작업들을 처리하기 위한 워크플로우 관리 시스템의 존재가 필요함에 따라서 많은 수의 작업들을 처리할 수 있는 초대형 워크플로우 시스템이 나타나게 된다. 이에 본 논문에서 초대형 워크플로우에 대하여 소개하고 워크플로우 시스템 구조를 분석하고 작업을 처리의 주체가 되는 구성 요소에 따라서 분류하고 각 구조의 장단점을 파악하여 본 논문에서 제안하는 초대형 워크플로우 시스템에 적합한 워크케이스 기반의 워크플로우 시스템 구조가 적합한지를 판단하고 하부구조로 사용되는 EJB 프레임워크의 적합한 이류를 기술한다. 또한 워크케이스 기반 워크플로우 시스템 구조를 구현하기 위한 요소들을 도출하여 초대형 워크플로우 시스템을 설계 및 구현한다. 특히 본 논문에서 제안하는 초대형 워크플로우 시스템은 워크플로우 시스템의 성능은 처리 가능한 작업 처리 수를 중심으로 하는 확장성(scalability)에 목적을 두고 개발된다.

2. 초대형 워크플로우

워크플로우를 사용하는 조직이 인터넷 환경에서 점차 거대화되고 높은 가치를 가지는 프로세스를 처리하게 됨에 따라서 워크플로우를 사용하는 조직들은 ' 트랜잭션 워크플로우' 와 ' 초대형 워크플로우' 라는 두 가지

요구사항을 요구하게 된다. 다음의 그림은 조직에서 요구하는 두 가지 요구 워크플로우를 나타낸 것이다.

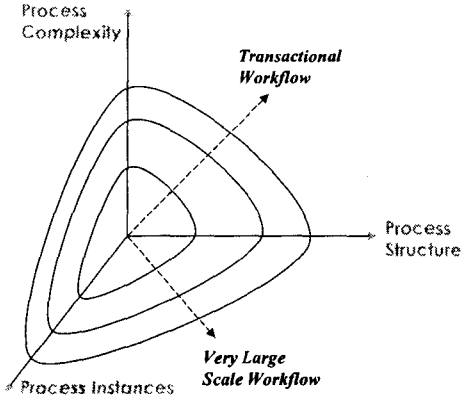

그림 1 워크플로우의 두 가지 동향

초대형 워크플로우는 워크플로우를 사용하는 조직이 점차 거대화되고 인터넷을 사용하게 되면서 작업의 수가 증가됨에 따라 이를 처리하는 것에 목적을 둔다. 그림 1 에서는 3 개의 영역이 선에 의하여 나누어져서 각 항목의 정도를 표시하여 초대형 워크플로우가 어떤 특성을 가지는 지를 나타낸다. 그림 1 에서 초대형 워크플로우는 거대 조직의 많은 수의 작업을 처리하기 때문에 프로세스 인스턴스 항목에서 일반적인 정도를 넘어서는 매우 많은 프로세스 인스턴스를 가지는 것을 보여준다. 또한 프로세스 구조 면에서는 다량의 작업을 효율적으로 처리할 구조를 가진다. 초대형 워크플로우는 많은 수의 프로세스 인스턴스, 많은 수의 작업 자, 지리적으로 분산되어 있는 많은 사이트들과 이종의 시스템에 기반을 둔다. 결과적으로 워크플로우의 기술적인 측면에서 중요 쟁점 사항은 초대형 워크플로우를 감당할 수 있는 워크플로우 시스템 아키텍처의 확장성(scalability)이다.

3. 워크플로우 시스템 구조 분류

워크플로우 시스템 구조의 분류는 워크플로우 시스템의 구성 요소 중에 워크플로우 시스템의 작업 처리 중심이 되는 구성 요소에 따라서 분류된다. 워크플로우 시스템은 워크플로우 구성 요소들의 구조를 통하여 어떤 방식으로 작업이 처리되는 지를 나타내고 전체적인 성능과 워크플로우 시스템의 특성을 나타낸다. 이 워크플로우 시스템의 구조의 분류에서 사용되는 워크플로우 구성 요소들은 일반적으로 워크플로우 분야에서 사용되는 개념들과 참조한 논문에서 사용되는 개념 등이 있기 때문에 다음의 개념 부분에서 본 논문의 워크플로우 분류에서 사용되는 개념에 대한 정의를 내린다. 그리고 워크플로우 시스템 구조를 분류하고 각각의 분류에 해당하는 워크플로우 시스템 구조를 분석한다.

3.1. 개념

워크플로우 분야는 여러 학문 분야가 포함되고 그에 따라 막대한 양의 발표 논문들이 존재한다. 이에 따라서 단어들과 그에 따르는 개념들의 충돌이 일어난다. 이에 본 논문에서는 ' 워크플로우 프로시저' 와 ' 액티비티' ' 워크케이스' 에 대한 ICN 의 기본 개념(Ellis and Morris, 1979)을 사용하여 워크플로우 시스템 구조를 분류한다. 또한 이 개념들은 워크플로우 표준화 기관인 WfMC(Workflow Management Coalition)의 표준과도 호환 가능하다.

' 워크플로우 프로시저' 는 하나의 작업을 처리하기 위하여 미리 정의된 작업 단계들의 집합이며 이 작업단계를 부분적으로 순서를 가지도록 정렬한 것이다. 이는 경제 분야에서 이야기하는 비즈니스 프로세스와 개념이 유사하며 워크플로우 프로시저는 컴퓨터를 기반으로 하는 차이점을 가진다. 이런 워크플로우 프로시저를 구성하는 각각의 작업 단계를 ' 액티비티' 라 한다. 이 액티비티들은 서로 연결되며 때로는 정의된 조건에 의하여 분기되거나 선택, 결합된다. 이런 분기나 결합은 기본적으로 AND 나 OR 조건을 가지고 처리된다. 가령 액티비티 A, B, C, D가 존재한다고 할 때 액티비티 A의 조건이" A -> B이고 A -> C" 이라면 A라는 액티비티 후에 B와 C라는 액티비티로 분기되는 것을 나타낸다. 또 A의 조건이" A -> B이거나 A -> C" 이라면 A라는 액티비티 후에 B또는 C액티비티 하나를 결정하여 작업을 처리하는 것이 된다. 분기나 선택과 짝을 이루어 흐름을 합쳐주는 것이 결합이다

워크플로우 프로시저가 실행 환경으로 이동하였을 때 작업 처리 요청에 따라 워크플로우 프로시저의 여러 개의 복사본이 만들어지며 만들어진 복사본을 ' 워크케이스' 라고 한다. 워크케이스는 워크플로우 프로시저의

데이터 이외에도 작업 처리 요청에 의하여 실행 시간에 생성되는 워크케이스의 상태나 작업 데이터, 요청자, 작업 생성과 시작 일시 등의 워크케이스 인스턴스 데이터를 가진다. 예를 들어 워크플로우 프로시저를 클래스라 본다면 워크케이스는 실행 타임에서 만들어지는 인스턴스라 생각하면 된다.

3.2. 워크플로우 시스템 구조 분류

워크플로우 시스템 구조의 분류는 작업 처리의 중심이 되는 워크플로우 구성요소에 따라서 분류된다. 워크플로우 시스템 구조 분류에 영향을 주는 워크플로우의 구성 요소는 ' 시스템' 과 ' 워크플로우 프로시저' ,' 워크케이스' ,' 액티비티' 의 4 가지 구성 요소에 의하여 분류를 한다. 워크플로우 프로시저와 워크케이스, 액티비티는 ICN 의 개념을 따르며 시스템은 워크플로우 시스템을 가리킨다.다음의 그림은 워크플로우 시스템 구조의 분류를 나타낸 것이다.

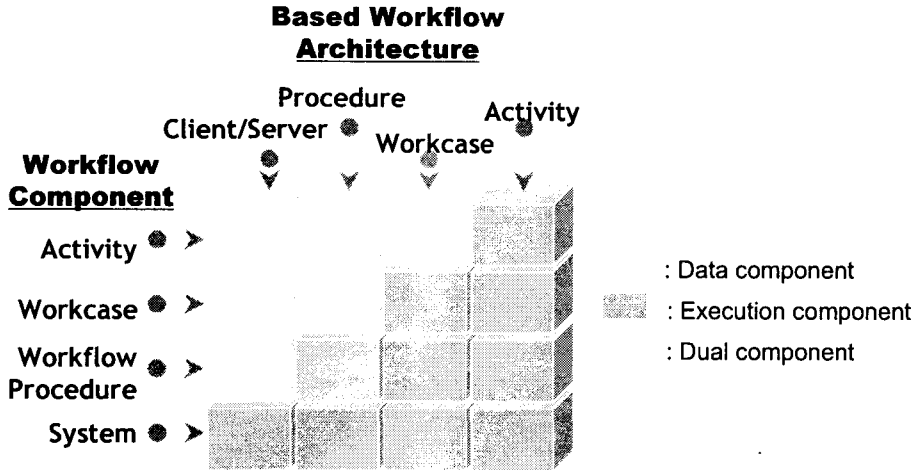

그림 2 워크플로우 시스템 구조 분류

그림의 워크플로우 시스템 구조 분류에서 워크플로우 시스템의 구조는 클라이언트/서버 기반의 워크플로우 시스템 아키텍쳐, 프로시저 기반의 워크플로우 시스템 아키텍쳐, 워크케이스 기반의 워크플로우 시스템 아키텍쳐, 액티비티 기반의 워크플로우 시스템 아키텍쳐의 4 가지로 분류된다. 그림에서는 각 워크플로우 시스템 구조에서 워크플로우의 4 가지 구성 요소들의 모습을 나타내어 워크플로우 시스템 구조를 나타낸다. 워크플로우의 구성 요소들은 데이터만을 가지고 있는 데이터 컴포넌트와 실행이 가능한 실행 컴포넌트, 그리고 데이터와 실행 능력을 가지고 있는 이중 컴포넌트로 나타내진다. 데이터 컴포넌트는 워크플로우의 실행과 관련된 데이터를 가지고 있는 컴포넌트이며 실행 컴포넌트는 내부에 데이터는 존재하지 않고 작업을 처리하기 위한 함수 또는 절차를 가진 컴포넌트이다. 이중 컴포넌트 데이터와 실행 컴포넌트의 기능을 함께 가지는 컴포넌트로서 데이터와 그에 따른 처리 함수를 같이 가진다. 실행 컴포넌트와 이중 컴포넌트는 처리의 주체로서 리소스를 사용하여 함수를 처리한다.

먼저 클라이언트/서버 기반의 워크플로우 시스템 구조는 그림 2 에서 보듯이 시스템이 실행 컴포넌트로 이루어지고 워크플로우 관련 컴포넌트는 데이터 컴포넌트로 구성된다. 이는 모든 작업의 실행을 시스템에서 하고 워크플로우 구성 요소들은 워크플로우의 데이터로 존재하는 구조로 시스템만이 작업을 처리하는 주체로 존재하고 워크플로우 구성 요소들은 처리 대상이 된다. 여기에서 시스템은 워크플로우에 직접적인 관련이 없는 실행 요소로 부가적으로 워크플로우 작업 처리를 한다. 다음의 그림은 클라이언트/서버 기반의 워크플로우 시스템 구조를 나타낸다.

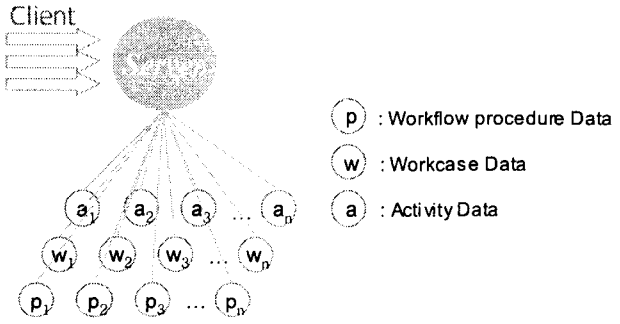

그림 3 클라이언트/서버 기반 워크플로우 구조

그림과 같은 구조를 가진 클라이언트/서버 기반의 워크플로우 시스템은 작업을 실행하는 처리 주체인 시스템이 한 개만 존재하는 정적인 구조이기에 여유 리소스에 비하여 작업의 처리 주체가 부족하여 시스템에 부하가 많이 생길 가능성이 있는 구조이다. 이 구조는 처리 작업이 약간 증가할 경우에도 시스템에 병목 현상이 일어날 가능성을 가진다. 워크플로우 시스템의 성능 측면으로 볼 때 적은 수의 작업 요청에도 시스템의 병목 현상이 일어나서 시스템 성능이 낮아질 가능성이 존재한다. 그리고 워크플로우 구성 요소들이 데이터 즉, 처리 대상으로만 존재하기 때문에 워크플로우 프로시저나 워크케이스가 증가되어 워크플로우 복잡도가 증가해도 시스템에는 영향을 주지 않아서 시스템의 복잡도와 워크플로우의 복잡도가 독립적인 모습이 된다.

프로시저 기반의 워크플로우 시스템 구조는 워크플로우 프로시저가 데이터와 작업을 처리하기 위한 실행 요소를 가지고 있는 형태의 이중 요소로서 작업 처리의 중심이 되는 구조이다. 즉, 워크플로우 프로시저가 작업 처리의 주체가 된다는 것을 말한다. 워크케이스와 액티비티는 시스템 단계의 구조와 마찬가지로 데이터로서 존재하고 시스템은 직접적인 작업 요청과는 거리가 있는 워크플로우 시스템의 하부 기능을 수행한다. 프로시저 기반의 워크플로우 시스템 구조는 다음의 그림 4 로 표현된다.

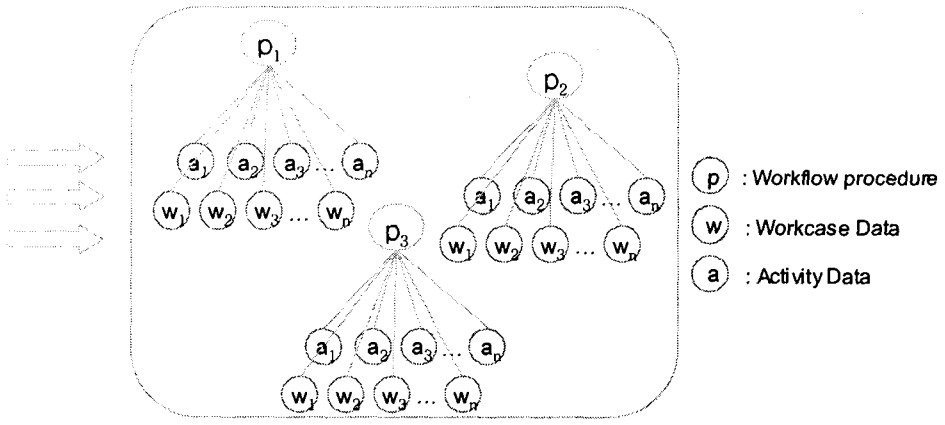

그림 4 프로시저 기반 워크플로우 구조

프로시저 기반의 워크플로우 시스템 구조에서 워크플로우 시스템에 작업 요청이 들어오면 요청 작업을 처리하는 주체는 작업의 워크플로우 프로시저의 종류와 같은 워크플로우 프로시저 컴포넌트이다. 이 구조는 앞의 클라이언트/서버 기반의 구조와는 다르게 워크플로우의 구성 요소 중 하나인 워크플로우 프로시저가 처리 주체로 등장하여 워크플로우의 복잡도가 증가할수록 시스템 복잡도가 완만하게 증가하는 형태를 취하는 데 이런 증가율을 보이는 것은 프로시저 기반의 워크플로우 시스템 구조는 작업의 요청이 증가해도 새로운 워크플로우 프로시저의 요청이 들어오기 전까지는 시스템의 복잡성이 증가되지 않기 때문이다. 그리고 프로시저 기반의 워크플로우 시스템 구조는 워크플로우 실행시간의 관점에서 볼 때 모델링을 한 시점에서 작업 처리의 주체인 워크플로우 프로시저가 정해져 정적인 구조를 가진다. 이는 작업 요청이 늘어나도 새로운 작업이 없을 경우에 워크플로우 프로시저가 생기지 않는 단점으로 나타나며 앞의 단계와 마찬가지로 작업의 처리 주체의 부족으로 시스템 부하의 원인이 되리라 예상된다. 이런 형태의 워크플로우 시스템 중에서 대표적인 시스템이 IBM 의 MQ series 로서 MQ 는 프로시저 기반의 워크플로우 시스템 구조를 가지며 작업을 처리하는 워크플로우 프로시저는 작업을 이루는 액티비티의 인스턴스들을 모델에서의 액티비티로 나누어 처리한다.

워크케이스 기반의 워크플로우 시스템 구조는 직접적인 작업의 처리가 워크케이스에 일어난다. 워크케이스는 워크플로우 프로시저로 작업 요청이 들어오면 워크플로우 프로시저의 인스턴스로 생성되어 프로시저 데이터와 액티비티 데이터, 처리 함수와 작업에 대한 인스턴스 데이터(상태, 결과값, 요청자, 시작 일시 등)를 가지고 작업을 처리한다. 워크플로우 프로시저는 데이터와 워크케이스의 생명 주기를 담당하는 역할을 한다. 아래의 그림 5 는 워크케이스 단계의 워크플로우 구조를 나타낸 것이다.

워크케이스 기반의 워크플로우 시스템 구조는 워크케이스가 작업의 처리 주체로서 존재하여 작업의 처리는 모두 워크케이스를 통하여 이루어진다. 워크케이스 기반의 워크플로우 시스템 구조의 특성은 요청한 작업의 수와 같은 수의 동적인 워크케이스 인스턴스 증가이다. 워크케이스 기반의 워크플로우 시스템 구조에서 워크케이스 인스턴스는 워크플로우의 실행시간에 동적으로 증가되는 데 이런 방식은 워크플로우 시스템에서 작업을 처리하는 처리 주체의 부족으로 리소스의 여유에도 불구하고 병목 현상이 일어나는 것을 예방한다. 워크케이스 기반의 워크플로우 시스템 구조에서 작업의 처리는 작업의 요청에 따라서 워크케이스가 생성되어 워크케이스가 워크케이스의 정보와 액티비티의 데이터를 통하여 작업을 처리하는 방식을 가진다. 이런 워크케이스 기반 방식은 작업의 증가에 따른 워크플로우 시스템의 구성 요소의 부하가 처리 주체의 동적인 증가로 정적인 구조에 비하여 상당량 감소하게 된다. 또한 이 구조는 워크플로우의 복잡도가 실질적인 작업 객체의 증가로

나타나는 시스템의 복잡도와 비례하여 증가하게 된다.

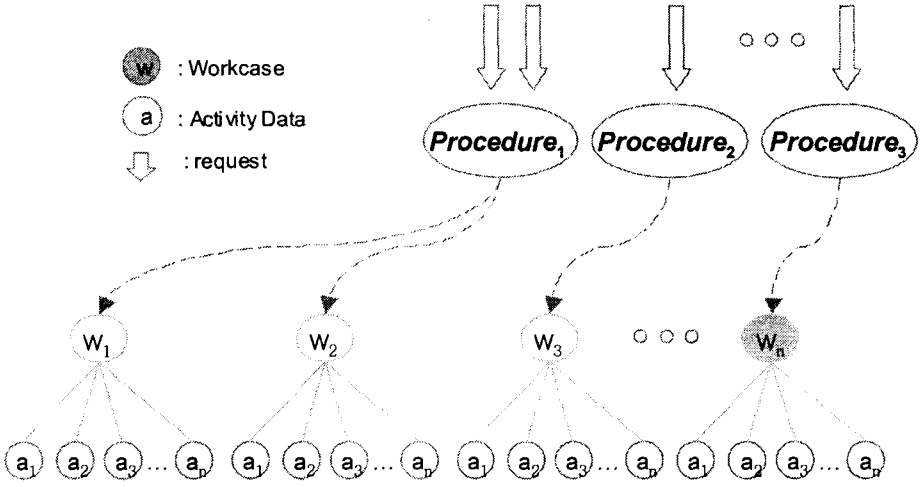

그림 5 워크케이스 기반 워크플로우 구조

액티비티 기반의 워크플로우 시스템 구조는 액티비티가 작업을 처리하는 주체로서 존재한다. 전체적인 작업 요청에 대한 관리는 워크케이스가 하지만 작업이 일어나는 세부적인 작업 단계는 액티비티에서 처리된다. 앞 단계의 워크케이스 기반 구조와 마찬가지로 워크케이스와 액티비티가 동적으로 생성된다. 시스템과 워크플로우 프로시저, 워크케이스, 액티비티에 이르는 4 가지 구성요소가 모두 처리 주체가 되는 형태를 가진다. 다음의 그림은 액티비티 단계의 워크플로우 시스템의 표준적인 구조로서 객체 표준화 그룹인 OMG(Object Management Group)에서 제안 Joint workflow(이하 조인트 플로우) 구조로서 액티비티 기반의 워크플로우 시스템 구조를 나타낸다.

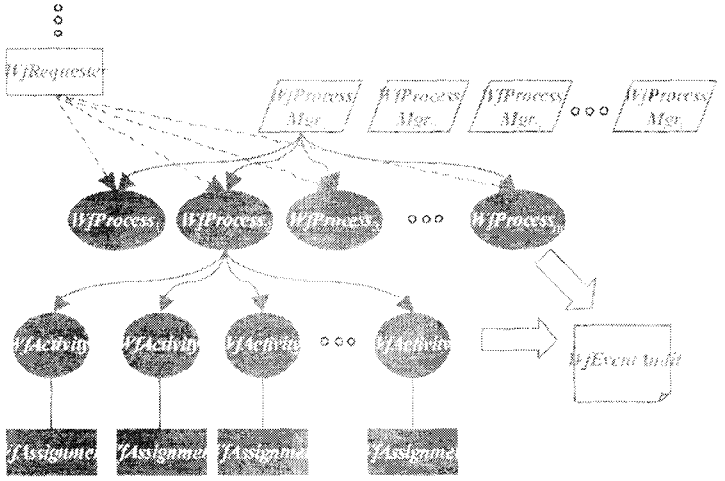

그림 6 OMG 조인트 플로우 모델

조인트 플로우의 wfRequester, wfProcessMgr, wfProcess, wfActivity 는 각각 작업에 대한 요청, 워크플로우 프로시저, 워크케이스, 액티비티를 나타낸다. 그리고 작업 주체인 액티비티에서 세부 작업이 이루어지며 실제적인 작업에 관련된 워크케이스나 액티비티는 오딧 데이터를 남겨 마이닝이나 시스템 복구 등의 여러 분야에 사용한다. 위와 같은 구조의 워크플로우 시스템은 워크케이스에 이어 액티비티도 동적인 생성이 되어 작업의 증가에 따라서 급격히 인스턴스들이 증가하게 되어 워크플로우의 복잡도가 증가함에 따라서 실제적인 인스턴스의 개수와 깊은 연관을 가지는 시스템의 복잡도가 급격히 증가하게 된다. 예를 들어 만약 n 개의 액티비티를 가진 워크플로우 프로시저 P_1이 있다고 했을 때 P_1의 작업 요청이 m 번이 들어왔다고 하면 총 인스턴스의 개수는 m+(m * n)개가 된다. 예를 들어서 6 개의 액티비티를 가진 워크케이스를 1000 개를 처리하기 위해서는 7000 개의 인스턴스가 필요하게 된다. 앞의 워크케이스 기반 워크플로우 시스템 구조와 비교하여 워크케이스 기반 구조가 m 개 즉 1000 개의 인스턴스가 필요한 것에 비하여 너무 많은 인스턴스를 필요로 한다. 이런 이유로 요청 작업이 증가하게 됨에 따라 시스템에 과다한 처리 주체가 존재하여 처리를 위해 사용되는 리소스가

부족해져서 워크플로우 시스템의 병목 현상이 일어날 가능성이 존재한다.

3.3. 성능 예측

앞장에서 워크플로우 시스템 구조의 분류를 통하여 구조를 이해하고 각각의 특성과 장단점을 알아낸 것은 초대형 워크플로우 시스템에 워크케이스 기반의 워크플로우 시스템을 선택해야 하는 이유를 제시하기 위하여 각 워크플로우 시스템 구조의 성능 예측을 증거로 삼기 위해서 이다. 이를 위한 참고 자료로 워크플로우 아키텍쳐를 위한 성능 분석 모델과 분석(Kim and Ellis, 2001)에서 이론적인 방법으로 분석한 워크플로우 아키텍쳐 비교를 사용한다. 다음의 그래프는 성능 평가 비교 그래프이다.

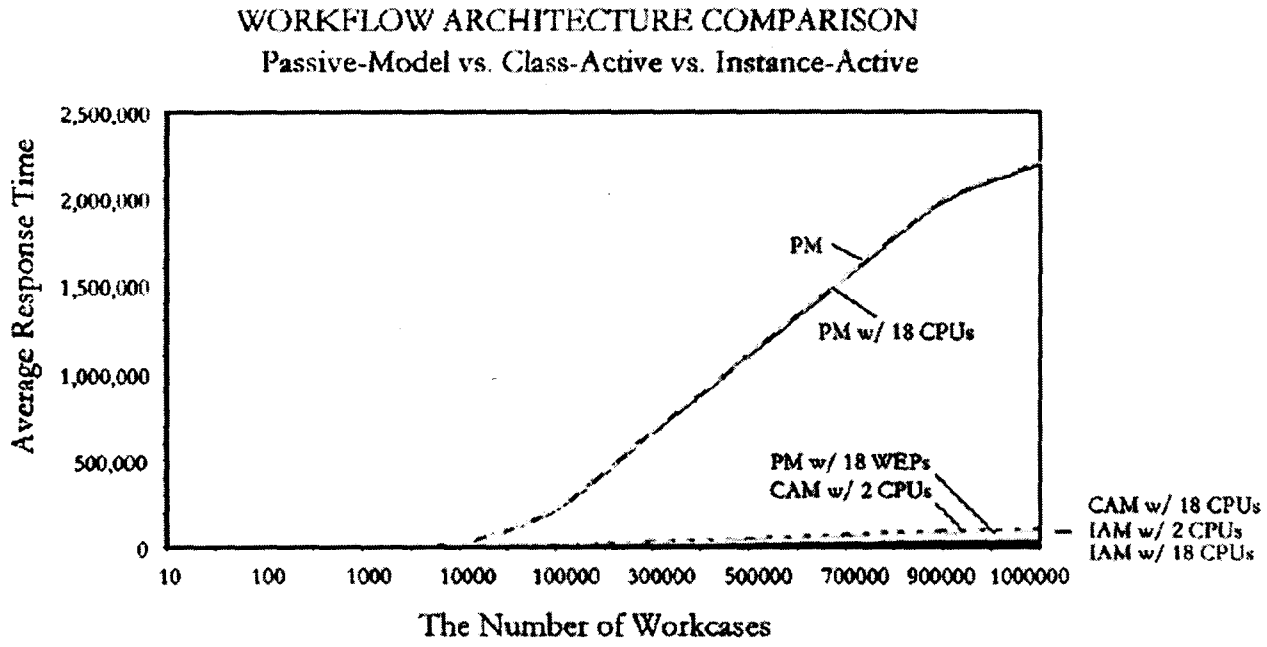

그림 7 성능 평가 비교 (Kim and Ellis, 2001)

성능 평가 비교는 PM (Passive-Model)과 CAM (Class-Active-Model), IAM (Instance-Active-Model)의 세 가지 아키텍쳐를 비교한다. PM 은 수동적인 모델로서 워크플로우의 모델이나 인스턴스에 처리 주체를 주지 않는 모델이다. 본 논문의 클라이언트/서버 기반의 워크플로우 시스템이 이 모델에 속한다. CAM 은 워크플로우의 정의된 모델마다 처리 주체를 주는 모델로서 프로시저 기반의 워크플로우 시스템 구조가 여기에 속한다. 마지막으로 IAM 은 워크플로우 시스템에서 실행 시간에 발생하는 인스턴스에 처리 주체를 주는 모델로서 워크케이스 기반의 워크플로우 시스템 구조와 액티비티 기반의 워크플로우 시스템 구조가 여기에 속한다.

위 그림 7 의 성능 평가 비교 그래프에서 세 가지 사항을 발견한다. 첫번째 사항은 같은 아키텍쳐 모델을 사용하며 2 개의 CPU 를 사용하는 CAM w/ 2 CPUs 와 18 개의 CPU 를 사용 하는 CAM w/ 18 CPUs 의 평 균 응답 시간에서 차이가 나는 것으로 하드웨어의 보충을 통하여 시스템의 성능 향상을 시킬 수 있다는 것이다. 두 번째 사항은 18 개의 CPU 위에서 PM 을 사용하 는 PM w/ 18 CPUs 와 18 개의 워크플 로우 엔진 위에 서 PM 을 사용하는 PM w/ 18 WEPs 는 평 균 응 답 시간에 큰 차이 가 나는 것에서 알 수 있듯이 시 스템의 리소 스가 남는 다 하더라 도 처리 주체가 없으면 병목현상이 일어난다는 것이다. 세 번째는 PM 과 CAM, 그리고 IAM 의 성능 분석 평가 비교에서 IAM 이 우수하게 나타난다는 것이다. 이 사항들은 본 논문의 워크플로우 시스템 구조 분류 중에서 워크케이스 기 반 구조 나 액 티비티 기반 구조가 클 라이언트/서버 기반 구조 나 프로시저 기 반 구조 보다 초대형 워크플로우에 적합함 을 보여 준다.

이 사항들과 워크플로우 시스템 구조 분류를 통하여 얻은 장단점을 바탕으로 워크플로우 시스템 구조 별 성능의 예측을 그래프로 나타낸다. 처음의 그래프는 워크플로우 시스템 별로 워크플로우 복잡도와 시스템의 복잡도의 관계를 나타낸 그래프이고 다음의 그래프는 워크케이스 즉, 작업의 증가에 따르는 시스템에 부하를 보여주는 그래프이다.

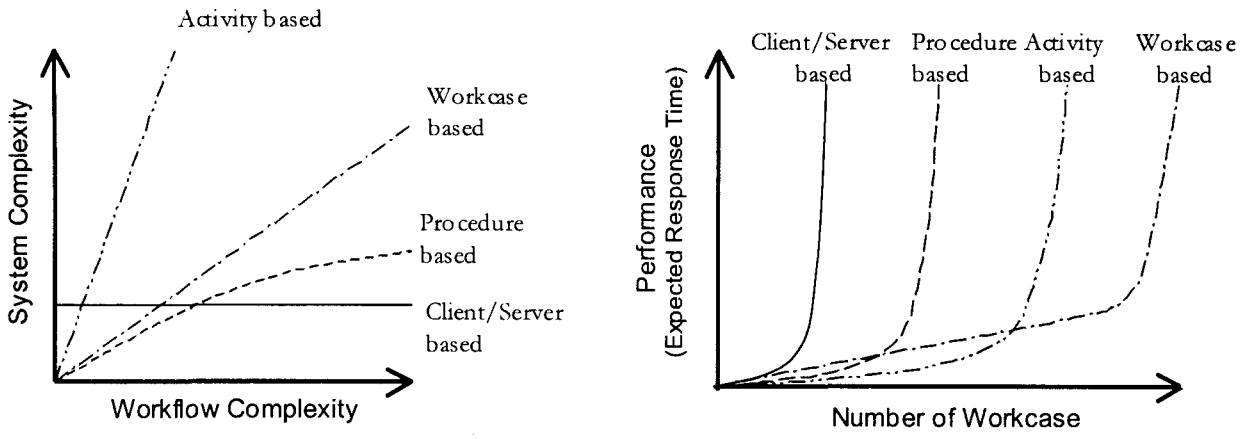

그림 7 워크플로우 구조 성능 예측

워크케이스 수에 따라서 시스템의 부하가 걸리는 그래프에서 보듯이 일단 모든 워크플로우 아키텍쳐는 하드웨어 적으로나 소프트웨어 적으로 처리 범위를 넘어서 작업 요청을 받으면 병목 현상이 생기게 된다. 초대형 워크플로우의 아키텍쳐를 선택하는 것은 이런 병목 현상이 늦게 발생하는 즉, 더 많은 수의 작업을 처리하는 구조를 선택한다. 앞 부분의 분석 결과를 바탕으로 클라이언트/서버 기반의 구조나 프로시저 기반의 구조보다 액티비티 기반의 구조나 워크케이스 기반의 구조를 선택한다. 액티비티 기반의 구조는 처음에는 시스템의 부하가 적지만 작업 요청이 많아지면서 작업에 따르는 액티비티 인스턴스를 생성시켜 많은 수의 작업을 처리하는 데 적합하지 않고 이에 반해 워크케이스 기반의 구조는 비록 액티비티 보다는 부하가 더 걸리지만 병목 현상이 일어나는 시간까지 더 많은 작업을 처리할 것으로 예상된다. 이에 본 논문은 워크케이스 기반의 워크플로우 시스템 구조를 초대형 워크플로우 시스템의 아키텍쳐로 선택한다.

4. EJB 가반의 워크플로우

워크플로우 시스템을 개발하는 많은 개발자들은 워크플로우를 독립적인 시스템으로서 개발하려 하고 그에 따라 시장에는 독립적인 솔루션으로 많은 워크플로우 시스템이 발매되는 중이다. 그러나 근래에 이르러 워크플로우 시스템을 프레임워크에 들어가는 하나의 컴포넌트로 개발하는 방법이 기존 개발 방식의 문제점에 대한 해결책으로 각광을 받는 실정이다. 그런 워크플로우 시스템이 들어가는 프레임워크 중 대표적인 것이 EJB 프레임 워크이다. 여기에서는 프레임워크로서 EJB 의 개념과 프레임워크를 기반으로 워크플로우 시스템을 개발할 때의 이익에 대하여 기술한다. 다음은 EJB 프레임워크의 구조이다.

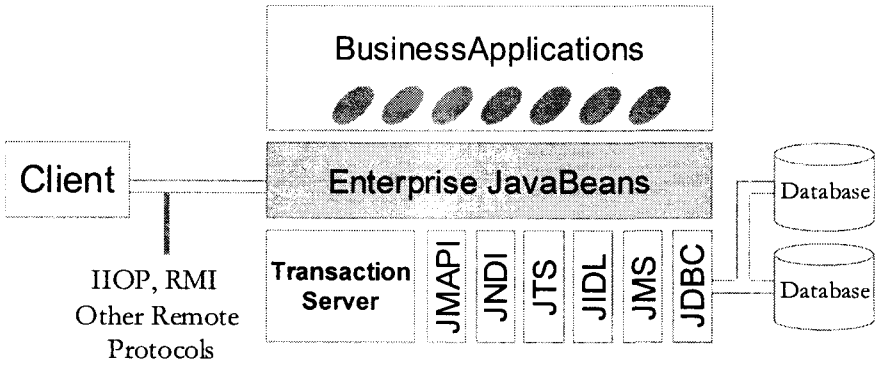

그림 8 EJB 프레임워크 구조

EJB 프레임워크는 기존의 통신 인프라로서 사용되던 분산 객체 기술에 기업이 필요로 하는 하부 시스템 레벨 기술들을 흡수하여 개발의 용이성과 신뢰성을 제공받는 시스템이다. EJB 프레임워크는 EJB 에서 시스템 레벨 서비스들과 비즈니스 로직 개발자들이 개발한 컴포넌트들이 돌아가는 EJB 컨테이너로 이루어진다. EJB 는 전자의 시스템 레벨 기술들 위에 비즈니스 로직이 들어있는 컴포넌트가 들어가는 EJB 컨테이너가 들어있는 구조를 가진다. EJB 에서 제공하는 시스템 레벨 서비스들에는 트랜잭션 서비스, 네이밍 서비스, 데이터베이스 서비스, 메시지 서비스, 그리고 시스템 기술인 클러스터링과 로드-밸런싱 등이 존재한다. 그리고 EJB 컨테이너에는 비즈니스 로직 컴포넌트가 들어가는 데 컴포넌트는 비즈니스 로직을 가지는 빈과 생성 주기를 관리하는

홈 인터페이스, 그리고 외부에서 빈으로 연결을 할 수 있는 통로인 리모트 인터페이스가 존재한다. 컨테이너는 이런 컴포넌트가 들어오면 그에 따른 내부 시스템들과 서비스들을 연결하도록 클래스를 자동으로 생성한다.
워크플로우 시스템에서 EJB 프레임워크를 사용할 경우와 하지 않을 경우의 차이점은 개발, 성능, 비용, 신뢰도의 4 가지 측면으로 나누어진다.

- 개발 : 워크플로우 시스템을 개발하기 위하여 프레임워크 접근 방식을 사용하지 않을 경우는 워크플로우 시스템을 위한 하부 구조를 개발하게 되어 시간을 소비하고 워크플로우 시스템 구조의 범위도 증가하여 시스템 개발에 많은 인력과 시간이 소비되며 또한 하부 시스템을 개발할 경우에 시스템 서비스를 개발하기 위한 고급 개발자 인력을 투입해야 한다. 이에 반하여 EJB 프레임워크 구조에서는 비즈니스 로직을 개발할 개발자만 있으면 내부의 시스템 레벨 기술들은 프레임워크에서 제고을 하여준다.
- 성능 : 워크플로우 시스템이 EJB 프레임워크를 기반으로 개발되면 프레임워크의 검증된 시스템 레벨 서비스를 사용하기 때문에 성능을 증가시키고 EJB 프레임워크가 발전할 경우에 워크플로우 시스템의 성능도 자동으로 개선된다.
- 비용 : 성능이 우수한 상용화된 EJB 프레임워크를 사용하기 위해서는 CPU 당 가격으로 많은 비용이 들고 워크플로우 시스템의 가격과 합쳐지면 EJB 기반 워크플로우 시스템은 높은 비용이 든다. 그러나 EJB 프레임워크를 사용하게 되어 시스템 자원이 효율적으로 사용되면 하드웨어나 소프트웨어의 설비 투자를 줄여서 워크플로우 시스템을 사용하는 전체적으로 보면 이익을 안겨준다.
- 신뢰성 : EJB 는 신뢰성을 검증 받은 프레임워크이다. 이런 EJB 프레임워크를 기반으로 개발되는 워크플로우 시스템은 신규로 전체 워크플로우 시스템을 개발하는 것에 비하여 신뢰성이 높고 시스템의 실패가 줄어든다.

위의 4 가지 측면으로 볼 때 워크플로우 시스템은 EJB 프레임워크를 기반으로 하는 시스템에 프레임워크 접근방식을 통하여 개발하는 것이 효율적으로 보여진다. 그럼으로 본 논문에서 개발되는 초대형 워크플로우 시스템도 EJB 프레임워크를 기반으로 하여 개발한다.

5. 아키텍쳐

초대형 워크플로우 시스템의 아키텍쳐는 개념적인 단계의 모델과 설계 단계 그리고 물리적인 구현의 세 단계로 나누어진다. 첫번째 단계인 개념적인 단계는 워크플로우 시스템의 개념적인 모습을 나타내는 단계로 워크케이스 단계의 구조를 중심으로 그에 대한 세부 사항과 전체적인 구조를 그린다. 두 번째 단계인 설계 단계에서는 첫번째 단계의 워크플로우 구조를 기반으로 하여 초대형 워크플로우 시스템에 필요한 요소들을 결합하여 초대형 워크플로우 시스템을 설계하는 단계이다. 이 단계에서는 초대형 워크플로우를 위해 필요한 비즈니스 메소드나 리소스들을 정의하고 전체적인 시스템의 구조를 설계한다. 세 번째 단계인 구현 단계는 워크플로우 시스템을 직접적으로 구현하는 단계로서 어떤 환경에서 구현 할 것인가와 어떤 기술을 사용해서 구현할 것인가 등의 여러 가지를 결정하고 구현하는 단계이다. 본 논문에서는 초대형 워크플로우 시스템을 구현하기 위하여 시스템의 부하를 줄이고 트랜잭션의 처리를 원활하게 하고 그것을 분산 시키는 등의 여러 가지 방법을 사용한다. 위의 세 단계로 나누어진 초대형 워크플로우 시스템의 아키텍쳐를 각각의 단계로서 자세히 설명한다.

5.1. 개념적인 단계

워크플로우 시스템의 개념적인 구조는 사용되는 워크플로우의 특성에 맞게 워크플로우 시스템 구조 분류 중에 하나의 구조를 선택한다. 본 논문에서 제안하는 초대형 워크플로우 시스템의 개념적인 구조는 워크케이스를 기반으로 하는 구조이다. 이 구조는 막대한 양의 작업을 처리하는 초대형 워크플로우의 특성을 고려하여 워크플로우의 복잡도에 따라서 시스템에 복잡도에 영향 관계와 작업 요청의 증가에 따른 워크플로우 시스템의 부하라는 두 가지항목으로 워크플로우 시스템 구조 분류를 분석한 결과로 워크케이스 기반 구조가 초대형 워크플로우에 적절하여 선택된다. 초대형 워크플로우 시스템의 작업 처리의 중심이 되는 워크케이스를 다음과 같이 나타낸다.

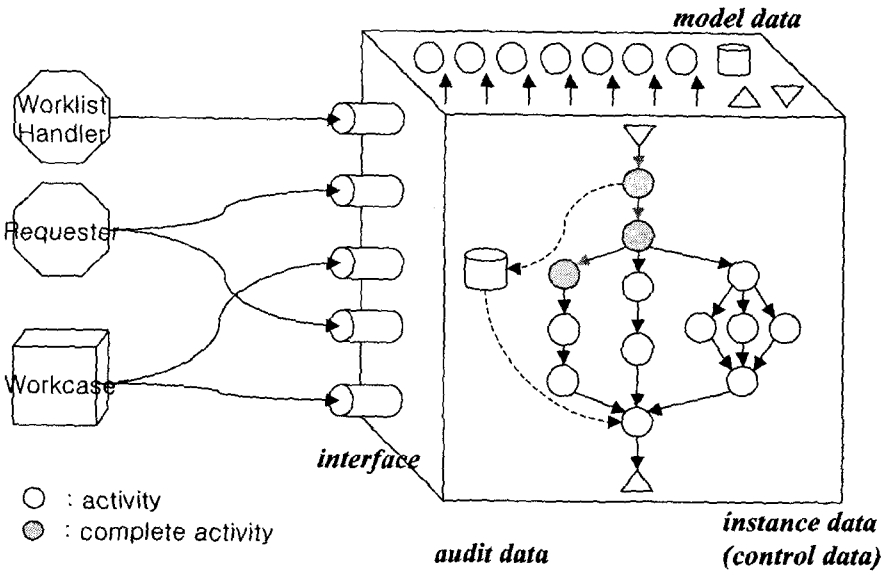

그림 9 워크케이스 개념 구조

초대형 워크플로우 시스템의 워크케이스는 인스턴스 데이터와 모델 데이터, 그리고 인터페이스의 3 가지 부분과 작업의 결과로 산출되는 오딧 데이터로 이루어진다. 워크케이스의 모델 데이터는 워크플로우 모델에서 정의된 비즈니스 프로세스를 출력한 데이터로서 워크플로우 프로시저 데이터로 표현된다. 인스턴스 데이터는 워크플로우 프로시저의 인스턴스인 워크케이스가 나타내는 상태나 관련 데이터, 작업 요청자, 시작 시간 등의 실행 시간에 나타나는 데이터이다. 인터페이스는 작업의 시작이나 상태 관리 등을 위한 외부와의 접점으로 작업 단계인 액티비티를 포함하는 특성에 따라 액티비티를 위한 인터페이스도 포함한다. 워크케이스 그림 8 과 같이 작업을 요청하는 요청자(requester) 와 액티비티를 처리하는 수행자와 접점인 워크리스트 핸들러, 그리고 다른 워크케이스와 관계를 가진다. 위와 같은 워크케이스가 초대형 워크플로우에서 주는 장점은

- 작업을 처리하기 위한 인스턴스의 감소
- 데이터 통합으로 인한 워크케이스 처리 속도 증가
- 워크케이스 처리의 복잡성 감소

작업을 처리하기 위한 인스턴스의 감소는 워크케이스를 처리하기 위해서 워크케이스만을 생성하면 되기 때문에 작업을 처리하기 위해서는 서브-플로우 등의 예외 상황이 일어나지 않는다면 하나의 작업 인스턴스만 존재하면 되어서 시스템이 유지하는 인스턴스를 줄여서 더 많은 작업을 처리 가능함을 말하고 데이터 통합으로 인한 워크케이스 처리속도 증가는 액티비티가 워크케이스에 데이터 요소로 통합되기 때문에 액티비티의 작업을 처리할 경우에 통합된 데이터를 통하여 보다 빠른 속도로 데이터를 검색 처리하여 작업 처리 시간을 줄여서 동시간 안에 더 많은 작업을 처리한다. 워크케이스의 처리 복잡성 감소는 워크케이스의 처리를 위한 프로세스의 분기나 결합 액티비티 간의 데이터 이동이나 처리 순서 등을 워크케이스에서 조정하기 때문에 액티비티 단계에서 작업을 처리하는 것에 비하여 타 액티비티를 참조하여 작업을 처리하는 등의 워크케이스를 처리하기 위한 어려운 상황이 발생할 경우에 복잡성이 감소된다.

5.2. 설계 단계

초대형 워크플로우의 설계는 거대 조직 환경에서 수십에서 수백만 건의 이른 작업들을 처리할 수 있는 처리 능력을 가지는 워크플로우 시스템 엔진을 설계하는데 목적을 둔다. 이를 위하여 워크플로우 표준화 기관인 WfMC 에서 제안한 참조 모델에서 정의하는 워크플로우 엔진 시스템의 기능을 수행하는 엔진을 기반으로 하여 거대 조직 환경에서 워크플로우 시스템을 운영할 수 있도록 초대형 워크플로우 시스템 엔진을 설계한다.
워크플로우 엔진이 만족시켜야 하는 기능들을 WfMC 에서는 다음과 같이 제안한다.

- 프로세스 모델의 정의를 엔진의 가용 데이터로 변환
  워크플로우의 모델을 정의하는 시점인 빌드타임에서는 워크플로우를 모델링하는 모델러를 통하여 만들어진 워크플로우 모델을 데이터베이스나 wf-XML, WPDL ,XPDL 과 같은 파일 또는 기타 저장 방식을 통하여 모델을 저장하는 데 이 데이터를 실제적으로 워크플로우를 실행시키는 런타임의 엔진 시스템이

엔진의 입력방식에 맞게 변환 하는 기능을 이야기한다.

- 프로세스 인스턴스 조정
  프로세스 인스턴스는 워크케이스 즉, 작업을 이야기하는 것으로 워크케이스가 생성된 후에 작업의 진행에 따라 워크케이스를 정지시키거나 완료 시키는 등의 상태를 관리하는 것을 이야기한다.
- 워크플로우 액티비티들 간의 처리 이동
  작업을 처리하기 위해선 작업 단계 즉, 액티비티들을 흐름에 맞게 순차적으로 처리해야 한다. 엔진은 액티비티들의 흐름을 관리하는 데이터를 이용하여 이 흐름을 관리하고 분기나 선택 등의 사건이 생기면 조건에 따라서 흐름을 관리하여 작업을 처리하는 기능을 수행한다.
- 특정 사용자의 시작과 종료
  워크플로우 시스템의 엔진은 워크플로우 시스템의 대부분을 담당하며 사용자와의 상호 작용을 통하여 액티비티를 처리하여 작업을 처리한다. 이에 따라 사용자의 구분과 사용자의 로깅 정보를 관리하는 기능을 수행하여 필요 사건에 제공하는 기능을 수행한다.
- 유저를 위한 워크아이템의 식별과 유저 상호작용을 위한 인터페이스 제공
  워크플로우 엔진 시스템에서 수행자들에게 할당되는 작업을 워크아이템이라고 한다. 워크아이템은 수행자가 정해져 있는데 유저와 수행자를 맞추어서 워크아이템을 제공하고 그에 대한 처리 기능을 제공한다.
- 워크플로우의 컨트롤 데이터와 관련 데이터 의 관리 유지
  워크플로우 엔진 시스템에서는 작업이 런타임에 발생할 때 작업 인스턴스에 대한 요청자와 생성 일시, 시작 일시, 상태 등과 같은 인스턴스 데이터와 작업 처리 결과로 액티비티들간의 흐름의 조건으로 사용되는 관련 데이터를 유지하여야 한다.
- 외부 어플리케이션 호출 인터페이스와 워크플로우 관련 데이터 연결
  워크플로우 시스템에서 어플리케이션을 통하여 처리하는 자동 액티비티를 처리할 때 워크플로우 시스템에서 외부 어플리케이션을 호출하고 처리할 관련 데이터를 넘기는 기능을 워크플로우 시스템 엔진에서 수행한다.
- 컨트롤과 관리, 감시를 위한 관리 행동
  워크플로우 시스템에서 작업이 처리되거나 수행 중인 경우 또는 작업이 끝난 후더라도 그에 대한 감시나 관리를 위한 기능을 엔진에서 제공한다.

위의 기능을 바탕으로 워크케이스 기반의 워크플로우 시스템 구조를 이용하여 기본적인 초대형 워크플로우 시스템의 구조를 설계한다. 다음은 초대형 워크플로우 시스템 구조를 그림으로 나타낸 것이다.

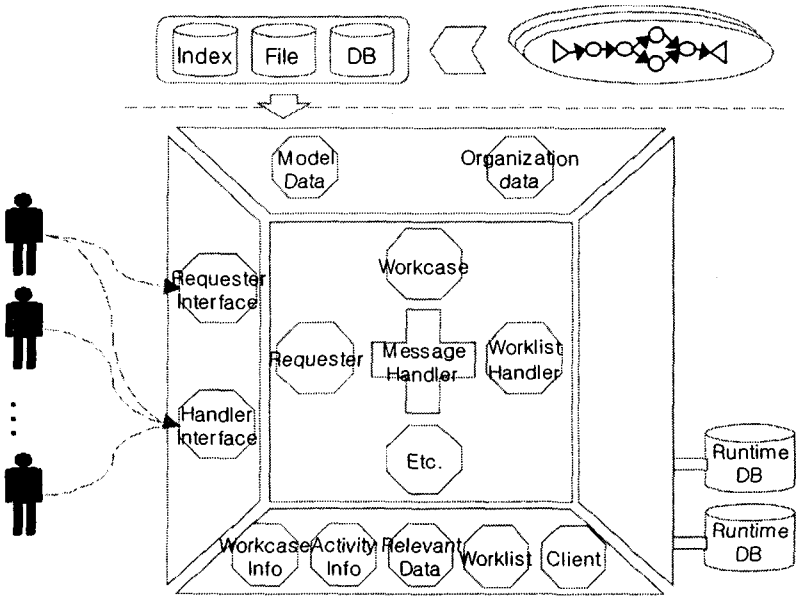

그림 10 초대형 워크플로우 시스템 구조

초대형 워크플로우 시스템 구조에서 주요 구성 요소는 시스템 안에 존재하는 Workcase(이하 워크케이스)와 Requester(이하 리퀘스터), Worklist Handler(이하 핸들러)이다. 세 개의 컴포넌트들은 EJB 의 컴포넌트로 설계되고 그 중에서 외부와 상호작용이 있는 리퀘스터와 핸들러는 클라이언트에게 인터페이스를 제공한다. 그리고 워크케이스를 관리하는 워크플로우 프로시저 컴포넌트는 EJB 의 홈 인터페이스가 워크케이스의 생명주기

를 관장함으로 설계에서 제외한다. 그림에서 주위에 존재하는 객체들은 데이터 컴포넌트로 워크플로우 시스템을 위한 데이터를 제공한다. 각각 객체에 대하여 설명하면 리퀘스터 객체는 클라이언트들과 상호 작용을 하는 객체로서 클라이언트로부터 작업 요청을 받아들여 워크케이스를 생성하거나 생성된 작업을 실행시키고 그에 대한 모니터링 정보를 제공한다. 또한 관리 기능의 한 가지로 엔진의 워크플로우 모델 데이터를 워크플로우 엔진 데이터로 입력하는 기능을 가진다. 워크케이스 객체는 모델 데이터와 관련 데이터를 이용하여 작업 처리를 하는 객체로 모델에서 정의된 워크플로우 프로시저에 따라 작업을 처리하며 작업의 상태 관리 등의 일을 한다. 액티비티 작업의 처리도 워크케이스 내부에서 일어나기 때문에 액티비티에서 수행자와 작업을 할 경우에 워크리스트 핸들러를 이용하여 작업을 처리한다. 핸들러 객체는 작업을 이루는 액티비티 중에서 작업 수행자가 필요한 수동 액티비티에서 발생하는 워크아이템을 작업 수행자의 클라이언트와 연결하며 작업을 처리한다. 작업 수행자 또는 클라이언트를 관리하는 역할도 한다. 그리고 워크플로우 시스템은 메시지 핸들러를 통하여 메시지를 통하여 비동기적으로 작업을 처리하여 작업 객체 간의 커플링을 최소화하고 엔진 시스템에 추가 객체가 생길 경우 메시지 핸들러에 연결하여 추가를 용이하게 한다. 위의 초대형 워크플로우 시스템 구조를 세부적인 인터페이스와 속성을 첨가하여 다음의 클래스 다이어그램을 통하여 나타낸다.

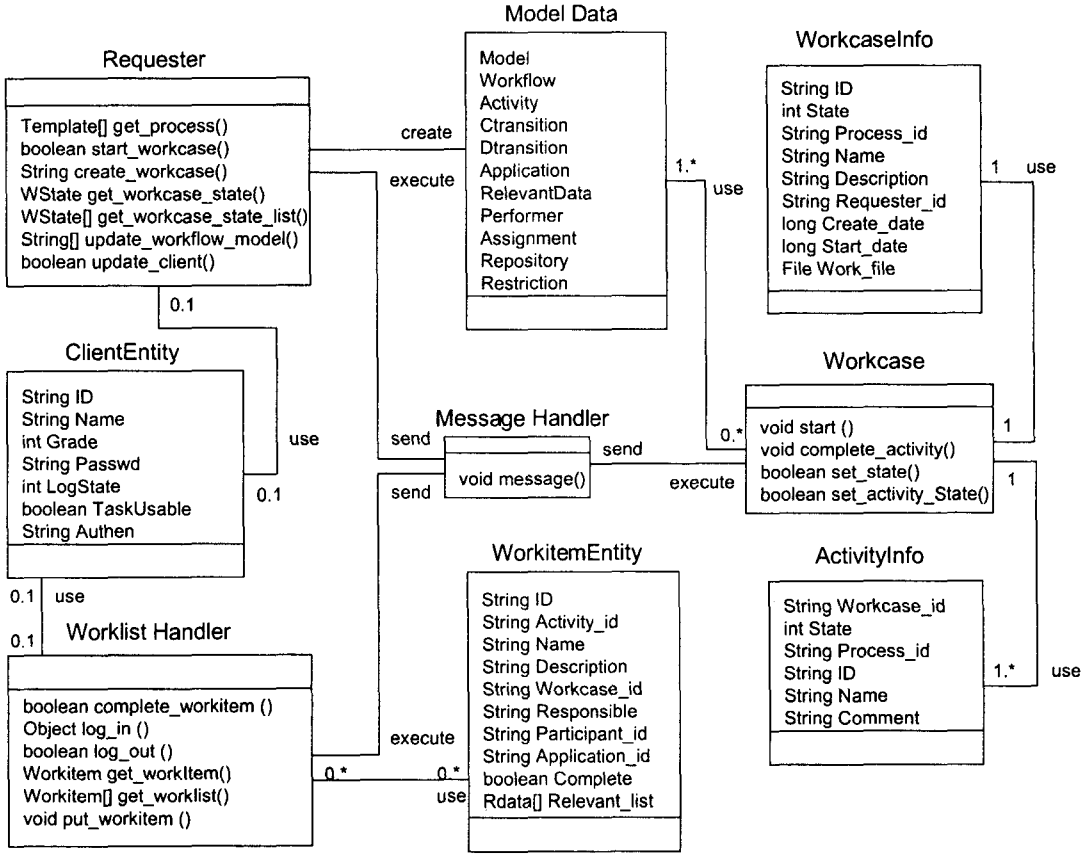

그림 11 클래스 다이어그램

클래스 다이어그램을 통하여 2 가지 부분으로 나누어 설계된 초대형 워크플로우 시스템에 대하여 설명한다. 초대형 워크플로우 시스템은 데이터와 이를 처리하는 로직이 나누어진다.

먼저 데이터 부분에서 데이터는 크게 3 가지 부분으로 다시 나누어진다. 첫번째 데이터는 빌드타임에서 정의된 모델 데이터가 변환된 데이터로 작업을 처리하기 위한 기본 데이터가 되며 정적인 데이터이다. 이 데이터는 클래스 다이어그램에서 Model Data 로 표현이 되며 세부적으로 모델과 워크플로우, 액티비티, 트랜지션, 데이터 트랜지션 등의 11 개 데이터 객체로 구성된다. 이 데이터는 전적으로 작업을 처리하기 위해 필요한 정보만이 존재하며 런타임 환경에서는 갱신 등의 변경 행위를 할 수 없다. 두 번째 데이터 부분은 작업을 런타임 환경에서 처리하면서 생기는 인스턴스 데이터 부분이다. 이 부분은 유지하는 인스턴스 데이터는 워크케이스와 액티비티에 관련된 데이타이다. 워크케이스 데이터는 요청자와 생성 일시, 시작 일시, 런타임 시에 정해지는 이름, 아이디, 워크플로우 프로시저의 아이디를 유지한다. 액티비티 데이터는 워크케이스의 아이디와 워크플로우 프로시저의 액티비티 아이디와 동일한 자신의 아이디, 워크플로우 프로시저 아이디, 상태와 수행자를 위한 코멘트를 유지한다. 세 번째 데이터 부분은 워크리스트 핸들러가 사용하는 데이터 부분으로서 워크플로우 시스템의 클라이언트 정보와 워크케이스에서 전달된 수행자들이 처리할 작업들의 정보와 작업 데이터이다. 클라

이언트의 정보는 리퀘스터도 클라이언트와 작업을 처리하기 때문에 사용한다. 이 중 클라이언트 데이터 중 Authen 속성은 세션이 열려졌을 때 생성시켜 세션을 관리하기 위하여 쓰여지는 데이터이고 워크아이템 데이타의 Relevant_list 속성은 작업 데이터를 나타내는 속성이다.

로직 부분은 앞의 워크플로우 시스템 구조에서 설명했듯이 리퀘스터와 핸들러, 워크케이스 그리고 이 객체들간의 메시지 방식의 통신을 담당하는 메시지 핸들러 부분으로 구성된다. 리퀘스터는 관리 인터페이스로 엔진의 모델데이타를 갱신하는 인터페이스와 사용자의 등급에 따라 사용 가능한 워크플로우 프로시저를 보여주고 워크케이스를 생성하고 실행시키고 모니터링 하는 인터페이스를 가진다. 워크케이스는 실행 명령을 통하여 작업 처리를 시작하고 수동 액티비티일 경우에 워크리스트 핸들러로 작업을 전달하여 전체적인 작업을 처리하는 인터페이스를 가진다. 워크리스트 핸들러는 클라이언트들에 관한 인터페이스를 제공하며 이를 통한 클라이언트 데이터를 리퀘스터와 공유하여 클라이언트 관리를 한다. 그리고 워크아이템에 대한 인터페이스를 제공하여 워크아이템을 가져오거나 처리된 워크아이템을 도려주는 것이 가능하다.

초대형 워크플로우 시스템의 구성 요소들은 서로 연계하여 작업을 수행하거나 처리하는 기능들과 워크플로우 시스템을 관리하는 기능을 외부에 제공한다. 다음 초대형 워크플로우 시스템의 실행 시나리오를 나타낸 시퀀스 다이어그램이다.

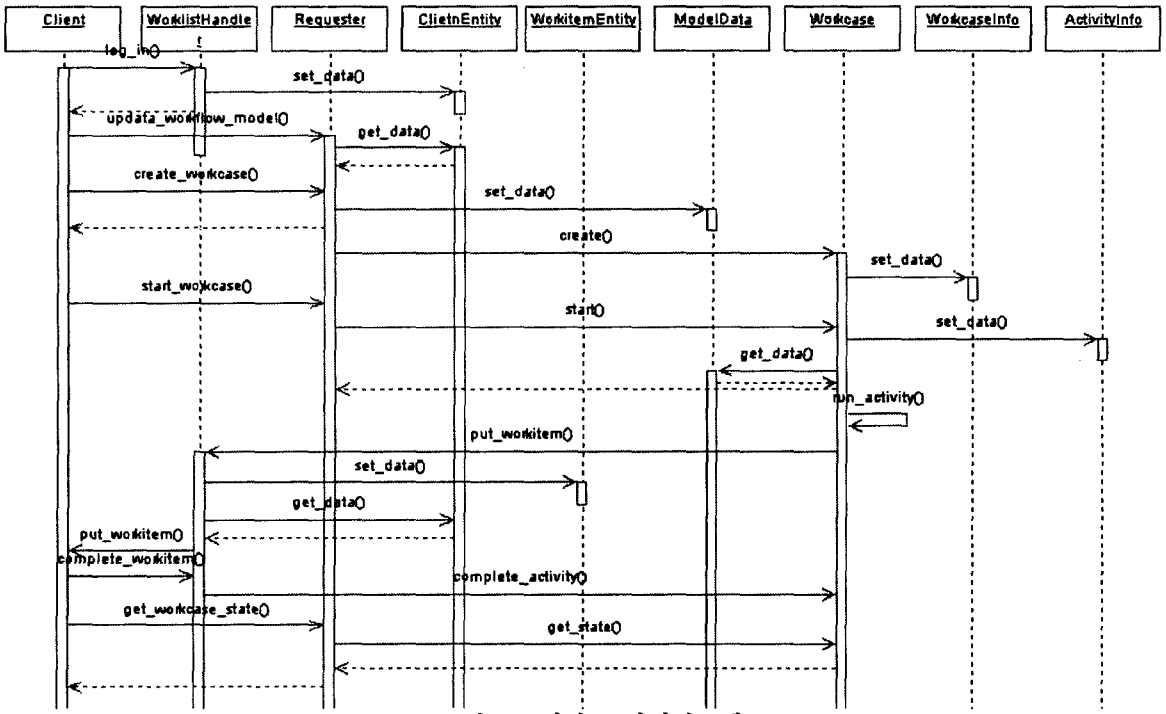

그림 12 시퀀스 다이어그램

그림의 시퀀스 다이어그램으로 표현된 초대형 워크플로우 시스템의 시나리오는 워크플로우 시스템에 로그인을 하는 경우와 워크플로우 엔진의 모델 데이터를 갱신하는 경우와 워크케이스를 생성 및 실행하는 경우, 할당된 워크아이템을 처리하는 경우, 워크케이스를 모니터링하는 경우의 5 가지 경우로 나누어진다.

먼저 워크플로우 엔진에 로그인을 하는 시나리오는 클라이언트가 워크리스트 핸들러로 로그인을 하여 조직 데이터를 참조하여 인증 과정을 거친다. 그리고 인증이 성공하면 인증 코드를 생성하여 세션 관리에 사용하며 클라이언트 데이터를 저장하고 인증키를 생성하여 클라이언트에게 반환한다. 클라이언트가 받은 인증키는 엔진의 클라이언트 데이터를 핸들러와 리퀘스터가 공유하여 클라이언트는 보안 등급이 허락하면 어느 쪽으로도 작업이 가능하다.

다음은 엔진의 모델 데이터를 갱신하는 시나리오이다. 엔진의 모델 데이터를 갱신하기 위하여 클라이언트는 먼저 관리자 등급의 인증을 받고 리퀘스터에게 엔진의 모델 데이터 갱신을 위한 함수를 호출하면 내부 메소들를 통하여 모델 데이터를 엔진에서 사용할 수 있게 변화한 후에 엔진의 모델 데이터 입력한다.

워크케이스의 생성 및 실행의 시나리오는 인증을 받은 클라이언트가 사용 가능한 워크플로우 프로세스를 리퀘스터를 통하여 받아서 생성시키려는 워크플로우 프로세스의 아이디와 함께 리퀘스터에게 요청을 하면 리퀘스터는 워크케이스를 생성시키고 워크케이스는 인스턴스 데이터들을 초기화한다. 그리고 리퀘스터에게 생성되 워크케이스 아이디를 반환하고 리퀘스터는 워크케이스 아이디를 다시 클라이언트에게 반환을 하게 된다. 클라이언트가 워크케이스 아이디를 통하여 리퀘스터에 워크케이스의 시작을 요청하게 되면 리퀘스터는 워크케이스에 시적을 요청하고 워크케이스는 모델 데이터를 참조하여 작업을 처리해 나간다. 워크케이스 내부의 액티비

티 처리는 내부 처리 함수를 불러 처리한다. 액티비티의 작업이 수동 작업이면 워크리스트 핸들러에게 작업을 위한 워크아이템을 전달한다.

할당된 워크아이템을 처리하는 시나리오는 인증된 클라이언트가 워크리스트 핸들러에게 접속한 상태에서 워크리스크 핸들러에게 워크아이템이 할당되면 워크리스트 핸들러는 조직 데이터베이스를 참조하여 수행자를 선정하고 수행자의 클라이언트에게 워크아이템을 전달한다. 클라이언트는 워크아이템을 처리한 후에 워크리스트 핸들러에게 보내고 워크리스트 핸들러는 처리가 끝난 워크아이템을 워크케이스에게 전달한다. 워크케이스는 내부 처리를 거쳐 다음 액티비티 작업을 수행한다.

워크케이스를 모니터링 하는 시나리오는 인증된 클라이언트가 리퀘스터에게 워크케이스의 상태를 요청하고 리퀘스터는 워크케이스에게 요청하여 워크케이스의 상태 정보를 클라이언트에게 전달한다. 워크케이스의 상태 정보는 클라이언트의 수행자에 관련된 모든 워크케이스의 상태를 검색하는 방법과 모든 워크케이스의 상태를 검색하는 방법이 존재한다.

5.3. 구현 단계

본 논문의 구현 단계는 사용자 인터페이스가 없는 엔진 시스템의 특성으로 예제로 사용할 ' 채용 프로세스' 의 ICN 모습과 ' 채용 프로세스' 를 초대형 워크플로우 시스템과 연계하여 처리하는 런타임 클라이언트와 모델러의 작업 화면을 구현 결과로서 보인다.

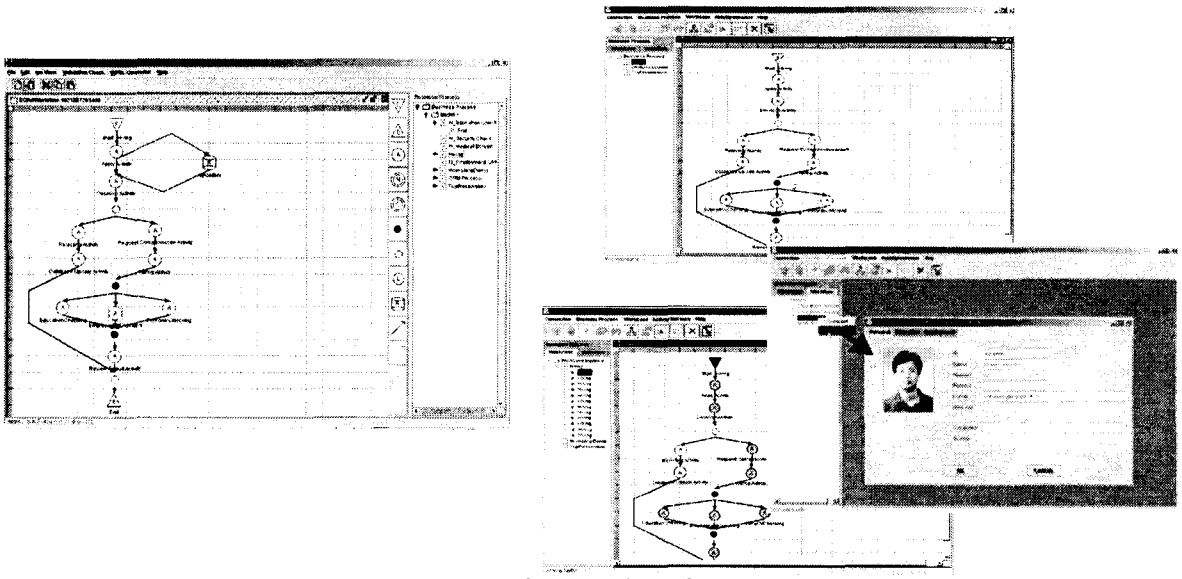

그림 13 구현 예제

좌측 부분의 그림이 ' 채용 프로세스' 를 모델링 하고 있는 모델러이다. 모델링이 끝난 모델 데이터는 워크플로우 엔진으로 입력되어 클라이언트가 작업을 처리하는 모델을 선택하고 시작하면 엔진에서 작업이 처리되기 시작한다. 우측 상단의 그림은 앞에서 이야기한 작업을 위한 모델을 선택하는 그림이다. 엔진에서 작업을 처리하기 위하여 첫번째 수행자에게 워크아이템을 할당하여 클라이언트에서 처리하는 그림이 우측 중단의 그림이다. 우측 하단의 그림은 작업이 처리 상태를 모니터링하는 화면이다.

6. 결론 및 향후 과제

본 논문에서는 초대형 워크플로우의 상황을 분석하고 이를 통하여 초대형 워크플로우에 적합한 시스템을 개발하고자 워크플로우 시스템들의 구조를 분류, 분석하여 장단점을 찾아내고 참고 자료를 바탕으로 성능 평가를 비교를 하여 본 논문에서 선택한 워크케이스 기반의 워크플로우 시스템 구조가 적합하다는 것이 예측하고 또한 이를 위하여 시스템 레벨의 하부 구조에서는 EJB 를 사용하여 컴포넌트 개발 시간을 절약하고 시스템 레벨의 문제점에 대하여 워크플로우 시스템이 관여할 필요가 없게 하였고 EJB 프레임워크의 발전에 따라 워크플로우 시스템의 능력도 향상되게 한다. 그러나 본 논문에서 개발한 초대형 워크플로우는 아직 여러 대의 하드웨어 위에 올려놓아 클러스터링을 사용하여 작업을 처리하는 등의 다중 서버 환경에 대한 설계가 미흡하다. 그리고 초대형 워크플로우 시스템에 대한 성능 예측을 하여서 초대형 워크플로우 시스템을 개발한 이유로 개발된 초대형 워크플로우 시스템의 성능을 측정하고 측정 자료를 바탕으로 초대형 워크플로우 시스템을 개선하

기 위하여 개발된 워크플로우 시스템에 대한 측정이 필요하다.

Acknowledgement
본 연구는 한국과학재단 목적기초연구(R05-2002-000-01431-0)지원으로 수행되었음.

참 고 문 헌

[1] Kim K, Ellis C. Performance Analytic Models and Analyses. *Information Systems Frontiers* 3:3, 339-355, 2001

[2] Ellis C, Morris P. The Information control nets model. *Performance Evaluation Review* 1979;8(3).

[3] Ellis C, Kim K. A framework and taxonomy for workflow architectures. In: *The Fourth International Conference on Design for Cooperative Systems*, May 2000

[4] Kim K. *Architecture for Very large scale workflow management systems*. PhD Thesis, Computer Science Department, University of Colorado at Boulder, May 1998

[5] Kim K, Han D. Performance and Scalability Analysis On Client/Server Workflow Architecture. *Proceedings of the 8th ICPADS*, June 2001

[6] Kim K. Instance-Active Workflow : Framework, Architecture, and Application. *Journal of Applied Systems Studied*, 2000

[7] Kim K. A Framework and Taxonomy for Distributed Workflow Architectures. *Telecommunications Review*, October 2001.

[8] 오동근, 김광훈. EJB 기반의 워크플로우 정의 데이터베이스 에이전트 설계 및 구현. 한국인터넷정보학회 논문지, 2001.12

[9] 김광훈. 차세대 워크플로우 런타임 작업 환경 아키텍쳐. 기초과학논문집, 2001.12

[10] 심성수, 김광훈. 대규모 워크플로우 시스템을 위한 EJB기반 워크리스트 핸들러의 설계 및 구현. 한국인터넷정보학회 추계학술발표논문집, 2001.11

[11] 오동근, 박웅, 문기동, 김광훈, 백수기. EJB기반의 워크플로우 정의 컴포넌트 설계 및 구현. 한국인터넷정보학회 추계학술발표논문집, 2001.11

[12] Workflow Management Facility Specification, V1.2 – OMG Document formal/00-05-02 – Contact:: *Mr. Juergen Boldt*

[13] EnterpriseJavaBeansProgramming SL-351 – SUN microsystems