CORBA에 기반한 워크플로우 엔진 프로토타입 개발*

김 동수, 김 영호, 강 석호
서울대학교 산업공학과

초록

본 연구에서는 공동 작업에 참여하는 작업자들의 업무흐름(workflow)을 관리하기 위한 워크플로우 엔진의 프로토타입을 개발하였다. 분산 객체관리의 표준으로 정착되어 가고 있는 CORBA(Common Object Request Broker Architecture)를 이용하여 네트워크 프로그래밍을 하였으며, JAVA를 사용하여 프로그램의 호환성과 이식성을 높였다. 또한, HAD(heterogeneous and autonomous and distributed) 환경에서 작동 가능하도록 엔진을 설계하였다. 따라서 공동작업에 참여하는 작업자들이 플랫폼에 상관없이 네트워크를 통해 엔진에 접속할 수 있으며, 워크플로우 엔진은 공동작업을 진행 및 감독할 수 있다. 본 연구에서 개발한 시스템을 설계 작업에서의 변경요청(ECO: Engineering Change Order)이라는 업무흐름에 적용하여 보았다.

1. 서론

오늘날 컴퓨팅 패러다임은 집중화에서 분산화로 변하고 있다. 또한, 기업 환경도 여러 부서의 사람들이 (혹은 같은 부서인 경우라도) 물리적으로 분산된 환경에서 공동의 목적을 가지고 협력하는 경우가 많다. 이러한 배경에서, 공동작업에 참여하는 사람들의 업무를 지원하기 위한 연구들이 수행되고 있다.

본 연구에서는 공동작업 비지니스 프로세스의 용이화 또는 자동화를 지원하기 위한 워크플로우 관리시스템 (Workflow Management System)을 구현하는 것을 목표로 한다. 워크플로우 관리시스템은 "업무활동의 순서와 그와 관련된 인적 자원, 정보 자원의 연계성을 고려함으로써 경영 활동의 절차적 자동화를 제공하는 것"이라고 정의된다 [1]. 워크플로우 관리시스템은 워크플로우의 실행을 정의하고 생성 및 관리하는 시스템으로 프로세스 정의를 해석할 수 있으며, 워크플로우 참여자들과 상호작용하고 정보기술 도구들과 응용 프로그램들을 호출할 수 있다.

WfMC (Workflow Management Coalition)의 참조 모델에 의하면 워크플로우 관리시스템은 프로세스 디자인 및 정의 부분, 워크플로우 시행(enactment) 서비스 부분, 업무리스트(worklist) 핸들러 부분으로 이루어진다 [1]. 워크플로우 프로세스 디자인 및 정의는 비즈니스 프로세스를 컴퓨터화해서 정의할 수 있도록 하는 것을 의미한다. 워크플로우 시행 부분은 정의된 프로세스를 현장에서 적용하여 가동시킬 때 모니터링 및 통제를 하는 부분으로 워크플로우 엔진이 그 핵심이 된다. 업무리스트 핸들러 부분은 업무 항목 리스트를 획득하기 위해서 워크플로우 시행 서버스로의 요청을 관리하고 정형화(formulate)하는 부분이다.

본 연구에서는 워크플로우 엔진 부분에 초점을 두고 워크플로우 관리시스템을 구현하여 네트워크로 연결된 작업자들의 업무를 관리할 수 있는 시스템을 개발하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 CORBA와 JAVA의 개요에 대해 설명하고, 3장에서는 시스템 아키텍처 및 워크플로우 엔진의 기능에 대해 설명한다. 4장에서는 구현에 대한

* 이 논문은 1996년도 한국학술진흥재단 공모과제 연구비에 의해 수행되었음.

상세한 내용을 알아보고, 5장에서는 결론 및 추후 발전방향에 대해 설명하였다.

2. CORBA와 JAVA 개요

CORBA는 차세대 컴퓨팅 패러다임이라 할 수 있는 분산화에 대처하기 위해 등장한 분산객체 관리 표준으로 OMG(Object Management Group)에서 제정하여 발전을 해오고 있다. 이기종 플랫폼과의 호환과 객체 공유를 목적으로 한 CORBA를 이용하여 네트워크 상의 분산된 작업자들이 유기적으로 연결되어 작업할 수 있다. 현재 CORBA 2.0 스펙이 나와 있으며, CORBA를 이용하므로써 얻는 장점은 다음과 같이 요약된다 [4].

- 코바는 플랫폼에 상관없이 작동가능한 상호운용성(interoperability)을 제공한다.
- 개발 과정의 전 단계에서 객체지향 패러다임을 지원한다.
- 프로그램의 생산성을 높인다.
- 코드의 재활용이 가능하다.

한편, JAVA는 '간단하며, 객체지향적이고, 인터프리터 방식이고, 견고하며, 보안적이고, 중립적 아키텍처를 가지고 있으면서 이식성이 좋고, 고성능의 멀티 스레드를 지원하는 동적인 언어'로 정의할 수 있다 [5].

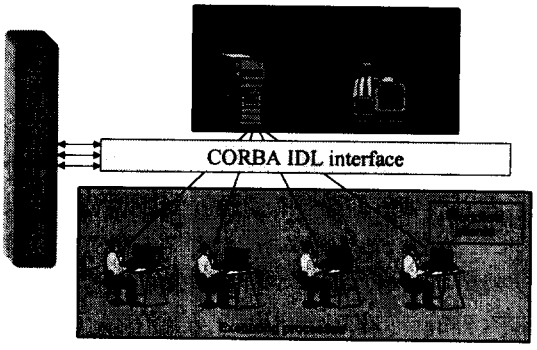
JAVA로 작성된 프로그램은 운영체제에 상관없이 사용할 수 있다. JAVA 프로그램을 컴파일하면 JAVA 바이트코드라는 일종의 수행 가능한 중립적 코드가 생기는데 이러한 코드는 기종이나 운영체제에 상관없이 수행할 수 있다. 이러한 JAVA 바이트코드를 해석할 수 있는 해석기(interpreter)를 기계마다 두는데, 기계별 해석기는 기계 종속적인 그래픽 처리나 입출력 관련 부분을 처리하게 된다.

따라서 본 연구에서는 JAVA와 CORBA의 장점들을 결합하여 프로그램의 호환성과 이식성을 높일 수 있도록 하였다.

3. 시스템 아키텍처 및 엔진의 기능

3.1 시스템 아키텍처

본 연구에서는 CORBA를 통신 하부구조로 하고, JAVA 언어를 이용하여 응용프로그램을 개발하였다. 전체적인 시스템 아키텍처는 [그림 1]과 같다.



[그림 1] 시스템 아키텍처

워크플로우 서버는 엔진이라고도 하는데 특정한 프로세스를 수행하는 워크플로우 클라이언트들이 엔진에 연결되어 업무를 할당받고 결과를 전달해 준다. 업무담당자(Client)와 엔진은 CORBA의 클라이언트 서버 구조와 잘 부합된다. 즉, CORBA의 IDL(Interface Definition Language)로 정의된 인터페이스를 바탕으로 객체 구현(object implementation)과 클라이언트가 연결되어 작동하게 되는데, 객체 구현 부분을 서버에서 담당하고, 클라이언트인 업무담당자 쪽의 프로그램에서는 객체 구현에 있는 코드를 호출하여 사용한다.

메시지를 전달하는 모든 과정은 CORBA의 IDL 인터페이스를 통해서 이루어진다.

3.2 워크플로우 엔진의 기능

워크플로우 엔진은 런타임시에 프로세스의 실행을 스케쥴링하고, 이를 가동시키며, 실행 과정을 통제하고 모니터링 한다. 일반적인 워크플로우 관리 시스템에서 엔진이 수행하는 기능은 다음과 같다.

- 프로세스 정의를 해석하는 기능
- 프로세스 인스턴스를 생성하고 그들의 실행을 관리(시작, 종결, 일시 정지, 재

시작)하는 기능
- 태스크가 종결되고 룰이 평가되었을 때 각 프로세스의 상태를 수정하는 기능
- 태스크를 수행할 수 있는 사람에게 주어진 업무를 전달하는 기능
- 프로세스 과정상 적합한 일의 항목을 생성하고, 태스크간의 이동을 규정짓는 룰에 따라 이동을 제어하는 기능
- 태스크나 프로세스가 데드라인을 넘겼을 때의 조치
- 감독, 관리 기능

본 연구에서는 WfMC의 참조 모델에서 제안하는 기본 기능을 구현하였다. 엔진의 구현에 초점을 맞추었으므로 프로세스 정의 모듈과의 인터페이스는 현재는 구현되지 않았다.

워크플로우가 초기화되어 시작되면 현재의 상태를 체크하고 업무담당자에게 수행해야할 업무를 알리는 메시지를 전달한다. 메시지를 전달받은 업무담당자는 자신이 수행해야 할 업무를 확인하여 수행하고, 그 결과를 다시 워크플로우 엔진에게 전달하게 된다. 각 업무담당자가 수행해야 할 과업의 리스트를 화면에 나타내 주면 편리하게 자신의 업무를 수행할 수 있다.

워크플로우 엔진은 정의된 워크플로우를 진행시켜 나가면서 다수의 업무담당자를 관리하고 조정하는 역할을 하는데 워크플로우 인스턴스 별로 자신의 상태를 업무의 진행에 따라 변경시켜 가면서 워크플로우의 수행을 책임진다. 업무담당자가 자신의 업무를 수행했다는 메시지를 보내면 메시지를 받았다는 이벤트에 대한 반응으로 자신의 상태를 갱신하고 다음 업무 담당자에게 업무 수행 명령 메시지를 전달한다. 이때 워크플로우 엔진은 그 업무의 진행 상황에 따라 업무담당자에게 독촉 메시지를 전달할 수도 있다.

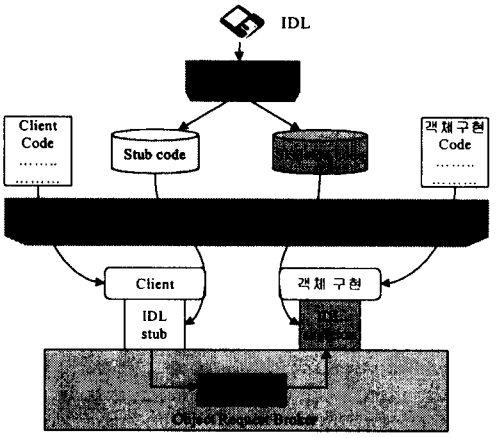
4. 구현

본 연구에서 사용한 CORBA 패키지는 IONA 社의 Orbix이며, 매핑 언어로 JAVA를 지원해 주는 Orbixweb 2.0을 사용하였다.

워크플로우 엔진 구현 과정을 상세히 설명해 보면 다음과 같다.

① IDL 인터페이스를 정의하고, 컴파일한다. 컴파일 결과로 JAVA 클래스 화일들이 생성된다.
② JAVA 클래스로 인터페이스를 구현한다.
③ 객체 구현을 통하여 JAVA 서버를 작성한다.
④ 구현저장소(Implementation Repository)에 서버를 등록시킨다.
⑤ 서버에 바인딩하여 서버의 객체 구현을 사용하는 클라이언트 프로그램을 작성한다.

이 과정을 그림으로 나타내면 [그림 2]와 같다.



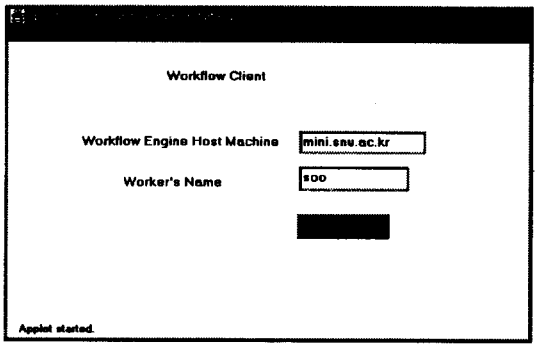
[그림 2] 시스템 개발 과정

IDL에 정의된 인터페이스를 이용하여 클라이언트가 객체 구현을 호출하며, 이 과정을 ORB가 책임진다.

4.1 설계 변경 업무 적용

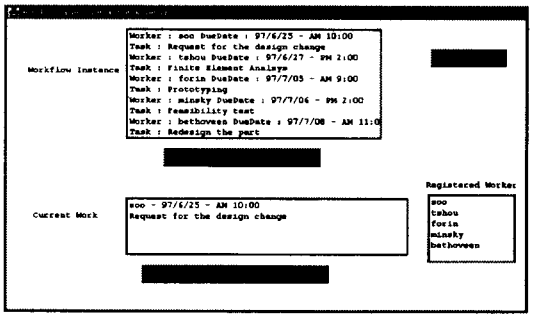
본 연구에서 개발한 엔진 프로토타입을 설계 변경 요청 업무에 적용시켜 보았다.

워크플로우 엔진이 구현되어 있는 호스트에 클라이언트가 접속하기 위한 애플릿을 구동한 화면이 [그림 3]에 나타나 있다. 'connect' 버튼을 누르면 엔진에 접속된다.



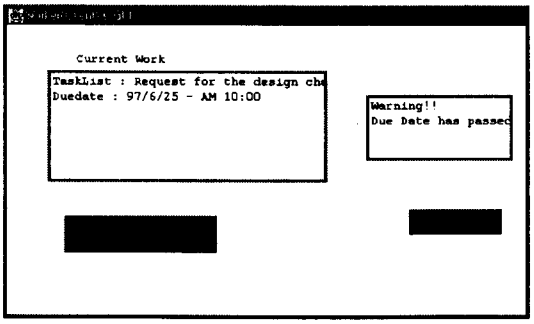
[그림 3] 클라이언트 등록 프로그램 GUI

[그림 4]는 워크플로우 엔진 쪽의 GUI 화면을 나타내고 있다. 설계 변경 요청(ECO) 워크플로우를 워크플로우 인스턴스 항목에 나타내고 있다.



[그림 4] 서버 쪽의 GUI

서버에 등록해 온 작업자를 등록 작업자 항목에 나타내었고, 현재 진행되고 있는 워크플로우의 상태를 나타내고 있으며, 진행 상황에 따라 작업담당자에게 독촉 명령을 내릴 수 있다.



[그림 5] 워크플로우 클라이언트 GUI

[그림 5]에는 워크플로우 진행에 참여하는 클라이언트 쪽의 GUI 화면을 보여 주고 있다. 자신이 수행해야 할 작업 리스트와 작업 기한이 나타나 있으며, 서버 쪽에서 보낸 독촉 메시지가 나타나 있다.

5. 결론

본 연구에서는 워크플로우 관리 시스템을 위한 워크플로우 엔진을 CORBA와 JAVA를 이용하여 구현하였다. CORBA 인터페이스와 서버 프로그램을 작성하였다. 개발한 워크플로우 엔진을 설계 변경 요청 프로세스에 적용시켜 보았다. CORBA의 장점인 이기종 시스템과의 통합의 편리함을 확인할 수 있었다.

JAVA와 CORBA를 이용하여 구현된 워크플로우 엔진은 보다 향상된 모습의 시스템으로의 확장이 가능하다. 즉, 프로세스 정의 모듈을 따로 개발하여 통합시킬 수도 있을 것이며, 업무담당자들의 업무 진행시 각종 어플리케이션과의 연동 기능 등을 제공할 수도 있을 것이다.

참고문헌

[1] Amit Sheth, Narayanan Krishnakumar, Managing Heterogeneous Multi-system Tasks to Support Enterprisewide Operations, D&PDB, 1995

[2] David Hollingworth, Workflow Management Coalition : The Workflow Reference Model, 1994

[3] J. A. Miller, A. P. Sheth, K. J. Kochut, X. Wang, CORBA-Based Run-Time Architectures for Workflow Management Systems, technical papers, URL - http://www.cs.uga.edu/LSDIS.

[4] Jn Siegel, CORBA Fundamentals and Programming, John Wiley & Sons, Inc., 1996.

[5] Mary Campione and Kathy Walrath, The Java Tutorial Object-Oriented Programming for the Internet - a practical, online guide to writing programs in the Java language, 1997.