폼 기반의 워크플로우 런타임 클라언트 응용 프로그램 생성기 설계

류재광*, 원은태*, 김광훈*, 백수기*
*경기대학교 전자계산학과
e-mail : {jkyou, etwon, kwang, skpaik}@kyonggi.ac.kr

Form-Oriented Workflow Runtime Client Application Generator Design

Jae-Kwang Ryu*, Eun-Tai Won, Kwang-Hoon Kim*, Su-Ki Paik*
*Dept. of Computer Science, Kyonggi University

요 약

기업의 워크플로우 시스템 개발과 도입에 관한 관심이 증가하고 있다. 이에 시스템 관련 개발 기업에서는 많은 예산과 인원을 투입하여 워크플로우 시스템 개발에 총력을 기울이고 있는 실정이다. 이러한 워크플로우 시스템에 있어서 폼이 가지는 중요성은 매우 크다. 폼은 런타임 클라이언트가 작업을 직접적으로 수행하고 수행된 결과를 되돌려야 하는 역할을 하기 때문이다. 따라서 데이터를 주고 받는 메커니즘을 얼마나 잘 정의를 하는가에 따라 폼에 대한 설계 방법론이 달라 지고, 이를 바탕으로 폼을 생성해야 하기 때문에 폼 생성기에 대한 새로운 설계가 필요하다. 본 논문에서는 워크플로우 시스템의 한 축을 담당하는 런타임 클라이언트 작업 도구인 폼(Form)에 관하여, 기존의 폼의 문제점을 분석하여 보다 나은 모델을 제시하고, 보다 강력한 기능을 제공하는 폼 생성기의 설계를 제안하고자 한다.

1. 서론

변화하는 기업환경에 기업들은 보다 나은 시스템을 도입함으로써 대처를 하려하고 있고, 그 중심에는 워크플로우 시스템(Workflow System)이 있다. 이는 대학과 같은 연구 기관은 물론 워크플로우 시스템을 개발하는 기관에서 기업의 환경에 맞는 워크플로우 시스템을 개발해야 한다는 기본적인 과제를 던져주게 된다. 그렇지만 현재 개발되고 있는 많은 시스템들이 아직 시스템과 시스템을 둘러싼 작업 환경간의 최적의 구현이 이루어지지 못하고 있는 실정이다. 특히 런타임 클라이언트(Runtime Client)가 작업을 수행하는 폼(Form)의 경우 폼 생성기(Form Generator)에 의해 만들어지는데, 이렇게 생성되는 기존의 폼의 경우 폼의 런타임 클라이언트의 작업, 즉 엔진 또는 어플리케이션 데이터베이스로부터 데이터를 가져온다거나 데이터를 저장하는 방식에만 초점을 맞추고 있다. 따라서 이렇게 설계된 폼은, 폼 생성기로부터 폼을 만들어내는 폼 디자이너(Form Designer)가 폼을 생성하는데 큰 어려움을 주게 된다. 본 논문에서는 보다 일관된, 즉 현재 개발되어지고 있는 워크플로우 시스템이 대부분 자바 개발 환경을 가지고 있는 점을 고려하여 폼의 구현에 관한 새로운 패턴을 제시하고 이를 바탕으로 폼이 보다 강력한 기능을 갖고, 폼을 생성하는 디자이너가 폼을 보다 쉽게 생성할 수 있도록 하기 위해 폼 생성기에 대한 설계를 제안하고자 한다.

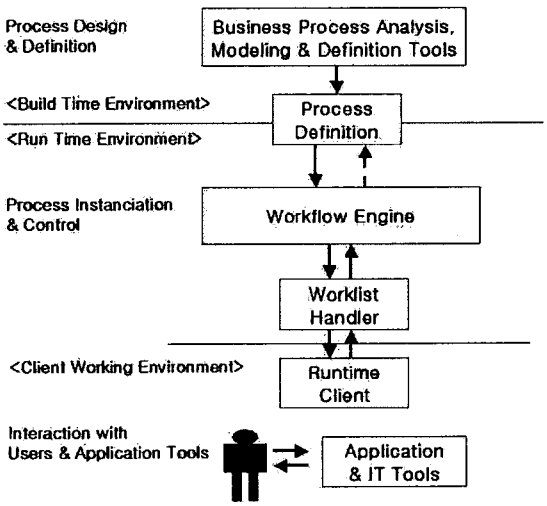
2. 런타임 클라이언트(Runtime Client)

2.1 정의

워크플로우 표준화 기관인 WfMC 에서 제정한 워크플로우 참조 모델(Workflow Reference Model)에 따르면, 워크플로우 관리 시스템(Workflow Mangerment System)은 크게 세 부분으로 나누어 볼 수 있다.[1]

그림 1 에서 보듯 프로세스(process)를 디자인 하는 빌드타임(build-time)영역과 프로세스를 구성하고 있는 각각의 액티비티(activity)들을 순차적으로 실행시키고

관리하는 런타임(run-time)부분, 그리고 유저(user)와 IT(Information Technoloty) Tool 등의 기업내의 자원(resource)간의 상호작용(interaction)이 일어나는 클라이언트 작업환경 등 크게 세부분으로 나뉘어 진다.
따라서 런타임 클라이언트는 워크플로우 정의기에서 정의되어진 프로세스가 진행될 때, 각각의 액티비티들의 수행을 담당하는 클라이언트라고 정의 할 수 있다.



[그림 1] Workflow System Characteristics

2.2 런타임 클라이언트 작업환경의 중요성

워크플로우 관리 시스템에 있어 가장 중요한 부분은 정의되어진 프로세스를 실행하고 관리하는 엔진(Engine)부분이라 할 수 있다. 엔진 내부에서는 정의되어진 프로세스에 따라 프로세스 인스턴스를 발생시키고, 이렇게 생성된 프로세스는 각각의 액티비티들을 생성시키게 되고, 생성 되어진 액티비티들을 관리하게 된다. 이러한 액티비티들은 정의되어진 대로 워크리스트 핸들러(Worklist Handler)로 하여금 런타임 클라이언트에게 작업을 할당하게 한다.[1] 런타임 클라이언트는 작업을 할당받으면서 작업에 필요한 데이터를 엔진 내부의 저장소(repository)나 어플리케이션 데이터베이스 서버(Application Database Server)로부터 데이터를 받게 되고, 작업을 수행하게 된다. 그리고 작업을 수행한 결과를 워크리스트 핸들러로 보내면 워크리스트 핸들러는 결과를 해당 저장소에 저장을 하게 된다. 어플리케인션 데이터베이스 서버에 저장해야 하는 데이터들 역시 어플리케이션 데이터베이스 서버에 데이터를 저장하게 된다. 따라서 폼은 런타임 클라이언트가 워크리스트 핸들러나 어플리케이션 데이터베이스 서버간의 데이터를 주고 받을 수 있는 메커니즘을 제공받아야 한다.

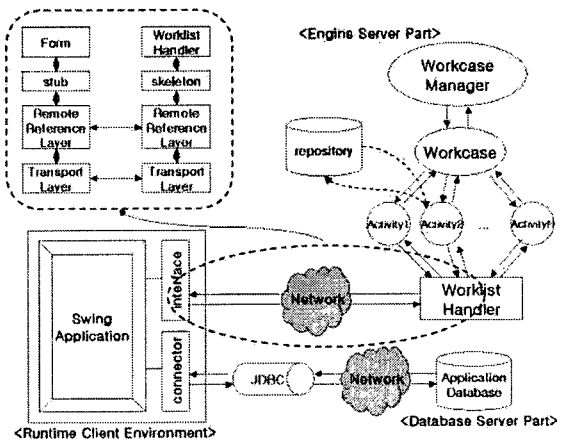
2.3 기존 런타임 클라이언트 폼의 한계

위에서 런타임 클라이언트의 작업 환경의 중요성에 관하여 언급을 하였다. 이렇듯 폼은 엔진, 데이터베이스와의 연동관계로 인하여 폼을 디자인 할 때부터 매우 어려움을 겪게 된다. 수 많은 데이터들에 관하여 폼상의 각각의 컬럼(column)들과 각각의 데이터들이 일대일 대응이 되어야 하기 때문에 디자인 부분에서부터 어려움이 따르는 이유이다. 그리고 폼이 자바 환경으로 개발되지 않을 경우 어플리케이션 데이터베이스에 데이터를 저장하는 데는 별 어려움이 없으나 엔진내의 저장소에 데이터를 저장하기 위한 폼의 구현에 어려움이 따르게 된다. 따라서 프로그램에 대한 지식과 경험이 없는 디자이너가 폼을 쉽게 디자인하게 하려면, 폼 생성기 자체의 구현이 매우 어렵게 된다. 따라서 폼에 대한 모델을 보다 잘 정의를 함으로써, 보다 강력한 기능을 지원하는 폼 생성기의 설계가 가능하다.

3. 폼 생성기의 설계

3.1 자바 vs. 자바 환경의 이용

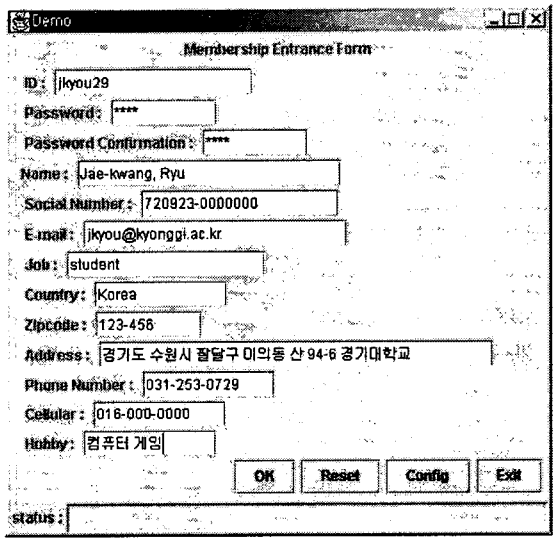
현재 대학이나 연구기관 등에서 개발하고 있는 워크플로우 관리 시스템은 대체로 자바 환경에서 개발이 되어지고 있다. 이는 이기종간의 호환성을 고려한 플랫폼 독립적인 측면을 강화하기 위함이다. 그리고 자바는 분산 객체 환경에 있어서도 보다 완벽한 환경을 제공해준다. 물론 윈도우에서의 분산 객체 환경 역시 그 자체로 매우 강력한 기능을 가지고는 있으나 플랫폼 독립성이 매우 떨어진다는 단점을 가지고 있다. 이러한 이유 때문에 워크플로우 시스템은 대부분 자바 환경에서 구현이 되어지고 있다. 그러므로 폼 생성기에서 생성된 폼이 다른 언어로 구현이 되어있을 때, 자바 대 자바환경에서 보다 더 어려운 통신 메커니즘에 대한 구현이 이루어지게 된다. 더구나 앞에서 언급한대로 단순한 데이터의 저장이 목적이 아닌 데이터 간의 연산이 필요한 경우 폼 내부에서는 프로그램이 또한 구현 되어져 있어야 한다. 이는 프로그램에 대한 지식과 경험이 없는 디자이너라면 폼의 디자인을 매우 어려워지게 하는 결과를 초래하게 된다.



[그림 2] 개선된 폼의 설계 구조

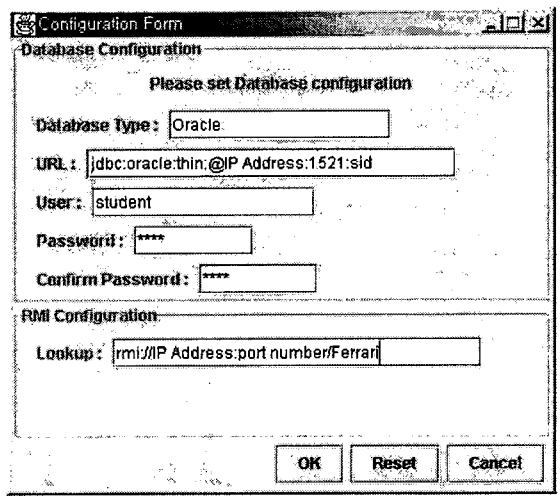
3.2 RMI & JDBC 를 이용한 폼 인터페이스

그림 2 를 보면 폼 작업자가 처리해야 할 데이터를 주고 받는 메커니즘에 크게 두 가지 방식을 사용함을 알 수 있다. 먼저 어플리케이션 데이터베이스와 폼의 통신은 자바에서 제공하는 JDBC 드라이버를 사용하게 된다. 그리고 워크리스트 핸들러와 폼의 통신은 RMI 방식을 이용하여 하게 된다. 그림 2 의 방식으로 간단한 폼 예제를 구현한 것이 그림 3 과 그림 4 이다.



[그림 3] 폼 예제

그림 3 은 런타임 클라이언트가 작업을 수행하기 위한 폼이다. 그리고 그림 4 는 런타임 클라이언트가 수행한 작업을 어플리케이션 데이터베이스에 저장하고 그리고 워크리스트 핸들러로 데이터를 보내주기 위한 환경을 설정해주는 부분이다.



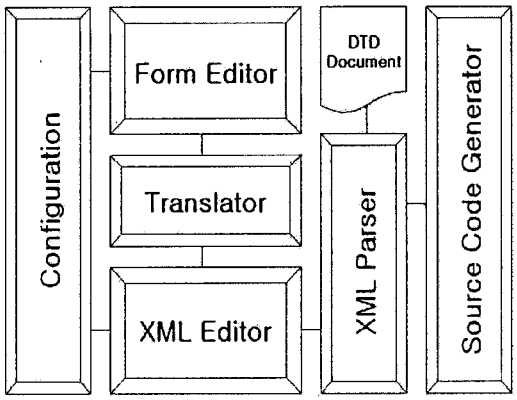
[그림 4] 데이터베이스 및 RMI 환경 설정

그림 4 에서 보면 어플리케이션 데이터베이스에 데이터를 저장하기위해 데이터베이스에 대한 환경을 설정해 주고 있다. 또한 워크리스트 핸들러와의 통신을 위해 RMI(Remote Method Invocation)에 관한 환경을 설정하고 있다.

즉 폼에서 처리되는 데이터는 각각 정의 되어진 어플리케이션 데이터베이스 또는 워크리스트 핸들러로 보내어지게 된다. 분산 객체간의 통신을 RMI 를 이용함으로써 간결하게 작업을 처리 한 결과를 전달 할 수 있게 된다. 또한 데이터의 연산이 필요한 부분은 폼 내부의 자바 프로그래밍 부분에서 처리를 해주게 된다. 폼에 프로그래밍에 관한 기능을 갖게 하기 위해서는 폼 생성기에서 폼을 생성해낼 때 연산이 필요한 부분을 정의를 해주어야 한다. 지금 구현된 폼은 매우 단순한 기능을 가지고 있는, 그리고 간단한 연산 기능을 가지고 있는 어플리케이션에 불과하다. 그렇지만 이런 단순한 어플리케이션이 폼 생성기에 의해 자동 생성이 되어져야 하기 때문에 폼 생성기가 보다 강력한 기능을 가질 수 있도록 설계가 이루어져야 한다.

3.3 폼 생성기의 설계

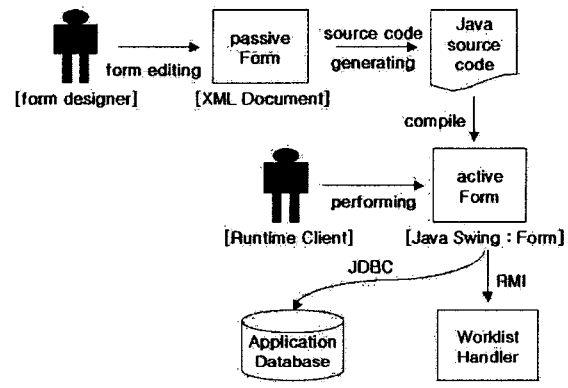
폼 생성기의 구조는 그림 5 와 같다. 폼 생성기가 가져야 하는 모듈은 크게 여섯개의 모듈로 나눌 수 있다. 첫번째는 환경을 설정해 주는 환경 설정 모듈(Configuration Module), 두번째는 GUI 환경에서 폼을 제작 하는 폼 편집기 모듈(Form Editor Module), 그리고 세번째는 폼 편집 모듈과 동시에 폼을 디자인하는 XML 편집기 모듈(XML Editor Module), 네번째는 폼 편집기에서 편집과 동시에 XML 편집을, 그리고 XML 편집기에서 편집과 동시에 폼 편집을 자동으로 변환하여 주는 변환기 모듈(Translator Module)이 있다.



[그림 5] 폼 생성기의 구조

그리고 다섯번째는 생성된 XML 문서를 파싱하는 XML 파서 모듈(XML Parser Module), 마지막으로 파싱된 결과를 바탕으로 자바 소스코드를 생성하고 컴파일 하는 소스코드 생성기 모듈(Source Code Generator Module)등 여섯 개의 모듈로 나눌 수 있다. 그리고 XML 태그에 관한 정보는 DTD 문서에 저장이 되어

있고, 이를 바탕으로 XML 파서는 파싱을 수행하게 된다.[3] 앞에서도 언급했듯이 워크플로우 시스템에서 폼이 자바환경으로 이루어졌을 때 가장 최적의 작업 환경을 구성할 수 있다고 생각이 된다. 그러기 위해서는 폼 생성기가 보다 자바의 여러 가지 기능들을 지원하는 폼의 생성이 필요하다고 본다. 그림 3 과 그림 4 에서 보여준 예제는 폼 생성기에 소스 코드를 생성시키기 위한 탬플릿(template)으로 제공이 되어진다. 즉 이미 검증된 폼을 기반으로 새로운 폼을 만들어 내는 것이다. 그리고 보다 효율적으로 소스 코드를 생성하기 위해 폼의 각 컬럼에 해당되는 부분을 모두 객체화 시켜 XML 문서로 만들게 되고 이를 파싱하여 폼을 구성하는 자바 패널(Panel) 컴포넌트를 생성하게 되는 것이 폼 생성기 설계부분에 있어 가장 핵심적인 부분이다.



[그림 6] 실행 환경

그림 6 은 폼의 디자인 과정에서부터 실행되는, 그리고 실행된 결과를 저장하는 전반적인 실행환경을 보여주는 그림이다. 그림을 보면 폼 디자이너에 의해 디자인된 폼은 소스 코드 생성기에 의해 소스 코드로 변환된 뒤 컴파일 되어 자바 실행 파일로 변환되게 된다. 이 실행 파일은 런타임 클라이언트가 워크리스트 핸들러로부터 작업을 할당 받아 작업을 시작할 때 실행이 된다. 프로그램이 실행이 되면 작업에 필요한 데이터를 워크리스트 핸들러와 어플리케이션 데이터베이스로부터 읽어 오게 되고, 이때 이용되는 인터페이스는 앞에서 언급했듯이 RMI 와 JDBC 방식이다. 새롭게 설계된 폼 생성기는 폼 디자이너가 보다 합리적인 방식의 폼 생성을 할 수 있도록 도와줄 것이며, 생성된 폼은 워크플로우 엔진과 어플리케이션 데이터베이스와의 연동에 가장 최적의 환경을 제공하게 된다.

4. 결론

본 논문에서는 런타임 클라이언트의 작업도구인 폼에 대한 새로운 모델을 제시하였고, 모델을 바탕으로 새로운 폼을 만들어 내는 폼 생성기의 설계를 제안하였다. 런타임 클라이언트 작업 환경의 중요성을 고려해 봤을 때 폼 생성기의 설계가 얼마나 큰 비중을 차지하는가를 알 수 있다. 결국 폼 생성기는 프로그램을 모르는 폼 디자이너가 프로그램에 신경을 쓰지 않고 폼을 디자인할 수 있어야 한다. 그러기 위해서는 폼 생성기가 워크플로우 엔진 즉, 워크리스트 핸들러와의 완벽한 통신을 지원해야 하고 데이터베이스와의 연동 역시 지원 하는 폼을 디자이너의 의도대로 자동 생성해 줄 수 있어야 한다. 폼 생성기의 기능이 강력할수록 폼 디자이너가 폼을 디자인 하는 것이 한결 쉬워질 수 있기 때문이다. 결론적으로 워크플로우 환경이 대부분 자바환경으로 이루어져있고, 최근에는 EJB(Enterprise Java Beans)의 환경의 WAS(Web Application Server)가 많이 개발이 되고 있으므로 결국 폼 역시 분산 객체 환경에 완벽한 지원을 필요로 한다면 결국은 자바 분산 객체 환경으로 가야 된다고 생각된다.

그러나 한가지 고려해야 될 점은 네트워크상의 모든 프로그램이 다 그러하듯 분산 객체 환경의 데이터 전달 역시 보완에 관한 완벽한 지원이 이루어 져야 한다. 따라서 추후 연구 과제는 폼 생성기 설계에 이어 폼 생성기에 의해서 생성된 폼이 네트워크 통신을 할 때 고려되어야 할 보안에 관련된 연구가 필요하다고 생각된다. 그리고 폼을 실행하는 환경에 있어서 BOS(Business Operating System)의 개념을 도입, 작업 환경에 있어 런타임 클라이언트가 보다 향상된 환경에서 작업을 할 수 있도록 지원하는 것 역시 추후 연구 과제로 남아있다.

참고문헌

[1] WfMC. "The Workflow Reference Model", Document Number TC00-1003, 19-Jan-95

[2] Layna Fischer. "Workflow Handbook 2001", ISBN 0-97033509-0-2, 2000

[3] Alexander Nakimovsky, Tom Myers. "Professional Java SML Programming", ISBN 1-861002-85-8, 2000

[4] WfMC. "Workflow Management Application programming Interface(Interface 2&3) Specification", Document Number WFMC-TC-1009, July-98