2001 년 한국정보처리학회 추계 학술발표논문집 제 8 권 제 2 호

기업 통합을 위한 워크플로우 관리 시스템 개발

변보균*, 한현철, 윤청

*충남대학교 컴퓨터과학과
e-mail : {bbg, alex, cyoun}@cs.cnu.ac.kr

A Development of Workflow Management System For Enterprise Integration

Bo-gyun, Byoun*, Hyun-chul, Han, Cheong, Youn
*Dept. of Computer Science, Chungnam National University

요 약

워크플로우 관리 시스템은 비즈니스 프로세스를 자동화하여 효율적으로 관리하기 위해 등장한 개념이다. 초기에는 비교적 작은 규모의 업무현장에서 활용되었으나, 업무환경의 급속한 변화에 따라 점차 자동화해야 하는 프로세스의 규모가 확대되고 있다. 즉, 단순히 하나의 사업장에서의 프로세스 자동화가 아닌 좀더 복잡한 업무 환경에서도 활용 가능한 프로세스 자동화 개념이 요구되고 있으며, 이에 따라 기존의 워크플로우 관리 시스템의 개념을 확장해야 할 필요가 있다.

본 논문에서는 빠르게 변화하는 기업 환경에 효과적으로 대처하기 위하여 기업 내 업무시스템들을 손 쉽게 비즈니스 프로세스 관점에서 통합할 수 있는 워크플로우 관리 시스템을 설계 구현하였다. 이를 위한 기반 기술로 XML 을 사용하였으며 통합할 시스템들 간에 사용할 통신 프로토콜로는 HTTP 를 이용하였다.

1. 서론

워크플로우 관리 시스템은 비즈니스 프로세스를 자동화하여 효율적으로 관리하기 위해 등장한 개념이다. 초기에는 비교적 작은 규모의 업무현장에서 활용되었으나, 업무환경의 급속한 변화에 따라 점차 자동화해야 하는 프로세스의 규모가 확대되고 있다. 즉, 단순히 하나의 사업장에서의 프로세스 자동화가 아닌 좀더 복잡한 업무 환경에서도 활용 가능한 프로세스 자동화 개념이 요구되고 있으며, 이에 따라 기존의 워크플로우 관리 시스템의 개념을 확장해야 할 필요가 있다.

부서 중심의 소규모 비즈니스 프로세스 자동화 시스템과는 다르게 전사(Enterprise) 혹은 B2B(Business-to-Business) 비즈니스 프로세스 자동화의 경우 다양한 시스템들이 유기적으로 통합되어 운영되어야 하는 경우가 많으며[1], 업무환경의 변화를 예측하기 힘들기 때문에 이러한 시스템 상의 변경이 자주 요구된다. 이에 따라 워크플로우 관리시스템의 순수한 기능인 비즈니스 프로세스 자동화 기능을 넘어 이러한 주변 시스템들을 비즈니스 프로세스 관점에서의 유연하게 통합할 수 있게 해주기 위한 기능이 요구되기 시작하였다.

본 논문에서는 빠르게 변화하는 기업 환경에 효과적으로 대처하기 위하여 기업 내 업무시스템들을 손쉽게 비즈니스 프로세스 관점에서 통합할 수 있는 워크플로우 관리 시스템을 설계 구현해 보았다. 이를 위한 기반 기술로 XML 을 사용하였으며 통합할 시스템들 간에 사용할 통신 프로토콜로는 HTTP 를 이용하였다.

2. 워크플로우 관리 시스템

워크플로우 관리 시스템은 애초에 비즈니스 프로세

* 본 논문은 충남대학교 BK21 사업단의 지원을 받아 수행되었음

스를 관리하기 위해 탄생한 시스템이다. 기존 시스템들에서는 구현 코드 내에 함축되어 녹아있던 비즈니스 프로세스 관리 부분을 따로 떼어내어 독립적으로 관리하기 위한 시스템이며, 이를 통해 비즈니스 프로세스의 변경에도 시스템이 유연하게 대처할 수 있게 되는 것이다.

워크플로우 관련 비영리 국제 표준화 기구인 WfMC(Workflow Management Coalition)의 정의에 따르면 워크플로우 관리 시스템의 정의는 다음과 같다.

"소프트웨어를 사용하여 워크플로우(비즈니스 프로세스)를 정의, 생성하고 관리하는 시스템으로 하나 혹은 그 이상의 워크플로우 엔진에서 동작하며, 프로세스 정의를 해석할 수 있고, 워크플로우 참여자와 상호작용할 수 있고, 필요하다면 IT 도구나 어플리케이션을 실행시킬 수도 있다."[3]

여기에서 주지해야 할 사실은 워크플로우 관리시스템이 단순히 조직 내 사용자들 간의 비즈니스 프로세스 뿐만 아니라, IT 도구나 어플리케이션까지도 워크플로우 관리시스템을 통해 관리할 수 있어야 한다는 사실이다. 관리해야 할 비즈니스 프로세스의 규모와 복잡도가 증대되면서 사용자들이 담당자가 되어 처리하는 비즈니스 프로세스 뿐만 아니라 기업 내 다양한 시스템들을 유기적으로 통합하여 관리할 수 있는 기능의 중요성이 점점 커지고 있는 실정이기 때문에, 워크플로우의 핵심 기능인 비즈니스 프로세스 자동화 이외에도 기업 시스템들의 유연한 통합을 위한 추가적인 기능이 요구되는 것이다.

3. 관련연구

3.1 XML

XML(eXtensible Markup Language)은 "자기 기술적(self-describing)"인 언어이다. 데이터의 구조 자체를 기술하여 새로운 타입의 데이터 구조를 선언할 수 있는 기술로써, 대표적인 메타 언어인 SGML 의 부분집합이다.

XML 은 등장 초기부터 그 가능성을 인정 받아 현재 다양한 분야에서 활용되고 있다. 특히 데이터의 구조를 유연하게 기술할 수 있기 때문에, 상호 이질적인 환경 하에서 데이터를 주고받아야 하는 환경 하에서 데이터 교환 규약으로서 자주 활용되는 추세이다. 최근 관심이 집중되고 있는 기업간 전자상거래나 EDI 프로젝트 등에서 XML 은 더 이상 빠질 수 없는 기술로서 자리잡아 가고 있다.

3.2 HTTP

HTTP(Hyper Text Transfer Protocol)은 1990 년대 이래 인터넷의 폭발적인 발전으로 전세계적으로 가장 보편화된 통신 프로토콜이다. 현재 대부분의 정보 자원들은 HTTP 를 통해 접속할 수 있는 위치에 존재하고 있기 때문에, 다양한 시스템들의 통합을 위한 기본 프로토콜로의 활용이 기대되고 있다.

SOAP(Simple Object Access Protocol)은 기저에 이러한 HTTP 를 통신 프로토콜로 사용하여, 원격지의 객체를 호출할 수 있도록 고안된 기술이다. 이를 통해 다양한 객체들을 손쉽게 호출하여 사용할 수 있고, 특히 기존에 방화벽으로 가로막혀 있던 기업 내부의 시스템까지도 HTTP 를 통해 호출할 수 있다. SOAP 상에서는 XML 메시지를 통해 원격지의 객체를 실행시키며, 그 실행 결과 또한 XML 메시지로 되돌려받게 되어 있다.

본 논문에서는 SOAP 과 거의 유사한 형태의 HTTP 프로토콜 기반 XML 메시징 시스템을 설계하여, 통신 환경을 예측할 수 없는 분산 시스템 간의 상호작용을 가능케 하였다.

4. 기업 통합을 위한 워크플로우 관리 시스템 설계

워크플로우 관리 시스템을 기업 통합에 효율적으로 활용되기 위해서 기업 내 다양한 시스템 사이의 호출을 중재해주기 위한 추가적인 시스템을 고안하였는데, 그것이 통합 브로커이다. 일차적으로 통합 브로커는 워크플로우 관리 시스템 내의 비즈니스 프로세스 수행 중에 발생하는 외부 시스템 호출 요구 메시지를 해석하여 적절한 시스템으로 라우팅하여 결과적으로 외부 시스템이 원하는 형태로 실행되도록 해준다. 또한 외부 시스템과 내부 시스템 사이의 데이터 변환의 역할 또한 담당한다.

워크플로우 관리 시스템과 통합 브로커는 서로 밀접한 관계에 놓여있기 때문에 EJB의 RMI를 이용하여 통신하며, 전달되는 데이터의 형식은 XML 메시지를 이용한다. 성능이나 향후 분산 환경 지원을 위해 EJB 를 사용하였으나 실제로 두 서버 간의 결합도는 높다.

통합 브로커와 외부 시스템 사이의 호출은 HTTP를 이용하여 이루어진다. 외부 시스템의 실제 API를 직접 호출하는 기존의 방식이 아니라 외부 시스템 호출을 XML 메시지로 작성하여 전달하는 방식을 사용한다. 때문에, 외부 시스템 측에는 HTTP 요청을 처리하고 전송한 XML을 해석하여 실제 API를 호출할 수 있는 기능을 수행하는 부분이 추가되어야 한다. 그러한 기능을 수행하는 부분을 일반적으로 어댑터(Adapter)라고 칭한다.

이러한 구조는 일반적으로 기업통합 솔루션들에서 일반적으로 채택하고 있는 구조 패턴(Architecture Pattern)으로서, 본 시스템 설계 시 그림1 과 같은 구조 패턴을 적용하였다.

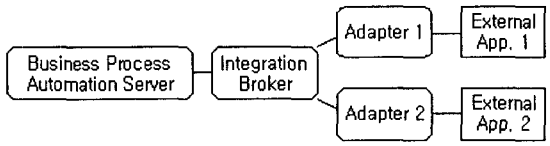

그림 1 EAI Architecture Pattern[2]

통합 브로커의 호출이 실제적으로 이루어지기 위해

서는 호출될 대상 외부 시스템에 대한 기능에 대해 정의가 이루어져있어야 하며, 그 정의 정보를 바탕으로 비즈니스 프로세스를 정의하여야 한다. 이러한 정의 작업이 끝나면 정의정보를 바탕으로 비즈니스 프로세스가 실행되면서 외부 시스템이 실제 호출되게 되는 것이다.

정의 시점과 수행 시점에 발생하는 일들을 좀더 상세히 기술해 보면 아래와 같다.

4.1 정의 시점에서의 정보

4.1.1 호출하려고 하는 외부 시스템의 기능 정의

통합 브로커에서는 일반적인 객체지향 언어에서 클래스를 정의하는 방식과 유사하게 외부 시스템 기능을 정의할 수 있도록 하고 있다. 외부 시스템 기능 정의 정보는 객체와 그에 속하는 메소드, 그리고 각 메소드에 속하는 파라미터 정보들로 구성된다. 이 정의 정보들은 XML로 작성되어 통합 브로커의 정의 정보 관리 서버에 의해 ExecutableObject라는 이름으로 저장된다.

외부 시스템을 객체지향 관점에서 객체 단위로 정의함으로써, 기존에 구조적 프로그래밍 방법 등의 비객체지향 시스템들의 통합에 대해서도 객체지향 관점의 통합이 가능해질 수 있다는 장점이 있다.

외부 시스템의 정의를 위한 정의 XML DTD는 아래와 같다.

```
<!ELEMENT Integration (MetaExecutableObject+)>
<!ELEMENT MetaExecutableObject (ObjName, Description, Category, MetaMethod+, MetaAdapterInfo)>
<!ELEMENT ObjName (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT MetaMethod (MethodName, Description, ReturnValue, MetaParameter*)>
<!ELEMENT MehodName (#PCDATA)>
<!ELEMENT ReturnType (#PCDATA)>
<!ELEMENT MetaParameter (ParamName, Description, DataType)>
<!ELEMENT ParamName (#PCDATA)>
<!ELEMENT DataType (#PCDATA)>
<!ELEMENT MetaAdapterInfo (URL, RequestMethod)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT RequestMethod (#PCDATA)>
```

MetaExecutableObject 라는 것은 하나의 외부 시스템 전체를 지칭하기 위한 요소(element)이며, 그 하위에 외부 시스템이 제공하는 기능들을 MetaMethod 로 기술하고 있다. MetaMethod 는 다시 기능 수행을 위해 전달하여야하는 파라미터(MetaParamenter) 정보와 수행의 결과로 돌아올 반환값의 타입(ReturnType)으로 구성되어 있다. 또한, MetaAdapterInfo 에는 통합하려는 외부 시스템의 호출을 중재하는 Adapter 의 URL 과 RequestMethod 를 기술할 수 있도록 하였다. 이를 통해 실제 외부 시스템의 물리적 인터넷 주소가 변경되어도 정의 정보만 변경하면 그 시스템 호출에 있어서의 위치 투명성을 확보할 수 있다.

4.1.2 워크플로우 관리 시스템에서의 외부 시스템 호출 정의

4.1.1에서 정의한 외부 시스템의 기능 정의 정보를 바탕으로 비즈니스 프로세스를 정의할 때, 프로세스 흐름 내에서 외부 시스템을 호출하도록 정의하기 위한 추가적인 정보가 요구된다.

외부 시스템 호출을 위한 추가적인 정의정보 중 가장 중요한 것은 프로세스 자동화 서버 내에서 관리되고 있는 데이터를 외부 시스템에 전달하기 위한 방법이다. 본 논문에서 설계 구현한 프로세스 자동화 서버에서는 프로세스 데이터와 산출 데이터, 이렇게 두가지 정보를 외부 시스템으로 전달할 수 있도록 하였으며 외부로의 전달에 대한 추가적인 프로세스 정의 정보에 대한 것은 아래의 추가 DTD에 정의되어 있다 (전체 프로세스 정의 DTD는 너무 방대하기 때문에 추가 DTD만 첨부하였음).

```
<!ELEMENT MetaBusinessRlue(SelectedExcutableObject, SelectedMethod,ReturnValue,Description, MetaParameter+)>
<!ELEMENT SelectedExecutableObject (#PCDATA)>
<!ELEMENT SelectedMethod (#PCDATA)>
<!ELEMENT ReturnValue (#PCDATA)>
<!ELEMENT MetaParameter (Name, Type, DataType, ProcessDataName?, OutObjectName?, SrcActivityKey?, Value?)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT SrcActivityKey (#PCDATA)>
<!ELEMENT Value (#PCDATA)>
```

4.2 수행 시점 정보

1) 프로세스 자동화 서버와 통합 브로커 간의 호출 방법

프로세스 자동화 서버는 원칙적으로 프로세스 정의정보를 바탕으로 수행된다. 프로세스 정의정보에는 앞서 언급한 바와 같이 외부 시스템 호출을 위한 정의 정보가 포함되어 있는데, 이 정보가 수행시점에는 수행상태에 맞게 실제 값들이 첨부되어 통합 브로커로 전달되어야 한다. 이에 대한 DTD는 아래와 같다.

```
<!ELEMENT BusinessRule (SelectedExecutableObjectKey, SelectedExecutableObjectName, SelectedExecutableObjectVersion, SelectedMethod, ReturnType, ReturnValue, Description, Parameter*)>
<!ELEMENT SelectedExecutableObjectKey (#PCDATA)>
<!ELEMENT SelectedExecutableObjectName (#PCDATA)>
<!ELEMENT SelectedExecutableObjectVersion (#PCDATA)>
<!ELEMENT SelectedMethod (#PCDATA)>
<!ELEMENT ReturnType (#PCDATA)>
<!ELEMENT ReturnValue (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Parameter (Name, Type, DataType, Value)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT DataType (#PCDATA)>
```

<!ELEMENT Value (#PCDATA)>

selectedExecutableObjectKey, selectedExecutable-ObjectName, selectedExecutable-ObjectVersion은 선택한 Object의 정보이다. 또한 selectedMethod는 executableObject의 메소드들 중, 선택한 메소드의 이름이다. Parameter 정보는 선택한 메소드의 파라미터로 넘어가는 값을 전달하기 위한 정보이다. 일반적인 메소드의 파라미터 정의처럼 Name, DataType, Value 등의 정보를 포함하고 있다.

2) 통합 브로커와 어뎁터 간 호출 방법

통합 브로커와 어뎁터 간 호출을 위해서는 프로세스 자동화 서버로부터 전달된 XML 메시지 중, 필수적으로 요구되는 데이터만 전달한다.

<!ELEMENT BusinessRule (selectedMethod, returnType, parameter*)>
<!ELEMENT selectedMethod (#PCDATA)>
<!ELEMENT returnType (#PCDATA)>

<!ELEMENT paramter (Name, DataType, Value)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT DataType (#PCDATA)>
<!ELEMENT Value (#PCDATA)>

위의 과정을 그림으로 도식화하면 그림2와 같다.

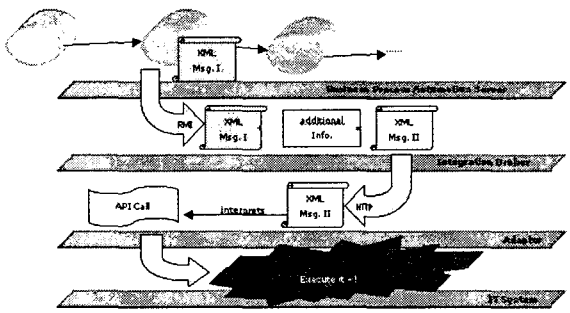

그림 2. XML 메시지의 변환 과정

5. 구 현

통합 브로커와 프로세스 자동화 서버는 EJB(Enterprise JavaBeans) 기반 환경 하에 작성되었으며, EJB 컨테이너로는 WebLogic 5.1 서버를 사용하였다.

통합 브로커의 통합 가능성 테스트를 위해 MS 측 분산 객체 기술인 DCOM(Distributed COM)으로 구현된 서버와 Java측 분산 객체 기술인 EJB로 구현된 서버를 외부 시스템으로 하여 통합하였다.

각각의 DCOM, EJB 기반 서버를 HTTP 기반으로 호출하고 통합 브로커로부터 전달한 XML 메시지를 해석하는 Adapter가 각각의 외부 시스템에 부착되어 있어야 한다.

XML 해석을 위해 Apache의 공개 소프트웨어인 Xerces 1.3.1을 사용하였으며 각각의 서버들에서 데이터를 저장하여야 하는 경우에는 Oracle 8i 데이터 베이스를 사용하였다.

6. 결 론

본 논문에서는 기존 워크플로우 관리 시스템에 기업 통합을 위한 기능을 추가함으로써, 비즈니스 프로세스 관리 관점에서 기업 내 존재하는 다양한 시스템들을 통합할 수 있는 솔루션을 설계 구현하였다. 각 시스템들간 데이터 교환을 위하여 XML 메시지를 활용함으로써, 각 시스템의 기능과 데이터의 구조를 좀 더 알기 쉬운 형태로 정의할 수 있었다. 또한 통합 브로커의 주요 기능인 외부 시스템 기능 호출 시에도 XML 로 구성한 메시지를 주고받고 그를 해석하여 실제 API 를 호출하는 방식을 취함으로써, 통합 브로커와 외부 시스템 간 결합도를 낮출 수 있었다.

XML 메시지 송수신을 위해서 가장 보편적인 통신 프로토콜인 HTTP 를 이용함으로써, 외부 시스템들의 각기 예상하기 힘든 환경에 대하여 대처할 수 있게 되었다.

그러나, 동기식(Synchronous) 메시지 교환 프로토콜을 사용함으로써, 시간이 오래 걸리는 외부 시스템 호출의 경우, 프로세스 자동화 서버에서 불필요한 지체 현상이 발생할 가능성이 있다. 이에 대한 해결책으로 제시되고 있는 비동기식(Asynchronous) 프로토콜 등의 안정적이고 효율적인 프로토콜의 적용에 대한 추가적인 연구가 요구된다.

참고문헌

[1] Accenture, "EAI 시장동향", 2001 EAI Solution Fair, 2001. 6
[2] Jeffrey C. Lutz, "EAI Architecture Patterns", EAI Journal, March 2000
[3] David Hollingsworth, "The Workflow Reference Model", WfMC, 1994
[4] Thomas Puschmann, Rainer Alt, "Enterprise Application Integration-The case of the Rebert Bosch Group", Proceedings of the 34th Hawaii International Conference on System Sciences 2001, 2001