SOAP을 이용한 이기종 워크플로우 관리시스템간 상호운용성

김승*, 이미숙*, 배혜림*, 김영호*, 박용태*
*서울대학교 산업공학과

Abstract

This paper describes the framework of integrating the business processes of different organizations through the interoperability of workflow management systems. In particular, this framework can be applied to developing B2B systems, in which each trade partner has its own heterogeneous workflow system on the Internet. We propose an approach to implementing an interoperable workflow system which can be integrated with other heterogeneous workflow systems through the four scenarios suggested by WfMC(Workflow Management Coalition), and we developed XML message-based prototype systems, which using SOAP(Simple Object Access Protocol), is applicable to standard e-business framework such as ebXML.

1. 서론

WFMS(Workflow Management System)은 기업이 처리해야 할 복잡한 프로세스를 정의하고, 관리하며, 자동으로 처리하여 주는 소프트웨어 시스템이다. WFMS을 도입함으로써 기업은 기업 내부의 비지니스 프로세스를 자동화하여 기업의 생산성을 향상시킬 수 있다.

최근에 들어, 많은 기업들은 여러 종류의 WFMS을 도입하여 왔으나 무분별한 WFMS의 도입은 분산, 이기종 환경하에서의 상호운용성 구현에 어려움을 가져왔다. 즉, 현재의 인터넷 환경에서는 이기종의 분산 시스템들이 산재하고 있으며, 이러한 분산, 이기종 시스템간의 상호운용성을 해결하지 못하면 비지니스 프로세스의 자동화는 단편적인 수준에서 그치게 된다.

워크플로우 관련 국제 표준 기관인 WfMC(Workflow Management Coalition) 및 여러 소프트웨어 업체에서는 이러한 문제점을 해결하고자 여러 가지 표준을 제정해오고 있다.

본 연구에서는 WfMC에서 제정한 표준의 하나인 Wf-XML binding[1] 및 MicroSoft가 제안한 SOAP(Simple Object Access Protocol)[2]을 이용하여 상이한 WFMS간에 상호 운용이 가능한 시스템을 개발하였다.

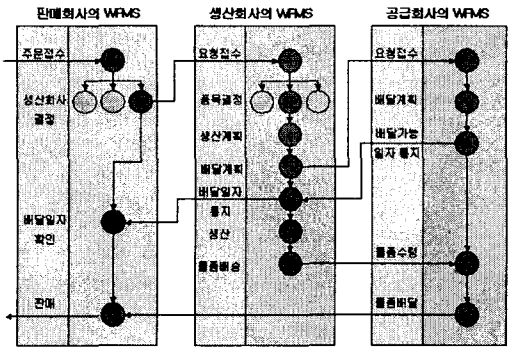
본 논문은 다음과 같이 구성되어 있다. 2장에서는 WFMS간 상호운용성 구현의 필요성에 대해서 논의하고 본 연구에서 사용된 Wf-XML binding 및 SOAP에 대하여 설명하였다. 3장에서는 이러한 두 표준을 활용하여 구현할 상호운용성의 전제 조건에 대해 논의하였으며, 4장에서는 프로세스 상호운용성의 시나리오를 기술하였다. 5장에서는 구현된 원형의 구조에 대하여 설명하였으며, 6장에서 결론을 맺는다.

2. 연구배경

2.1 상호운용성의 필요성

일반적인 기업의 비지니스 프로세스는 기업 내부의 비지니스 프로세스가 기업간에 연계되면서 가치사슬을 형성하게 된다. 과거에는 생산자가 가치사슬의 주체로서 물품의 생산 및 분배의 속도를 조절하였으나, 현재의 경쟁적인 환경에서 고객은 기업의 빠른 대응을 요구하게 되었고, 따라서 고객이 가치사슬의 주체가 되었다.

WFMS은 이러한 고객의 욕구 충족을 위한 기업의 대응 솔루션 역할을 하여왔으나 이는 개별 기업내의 업무프로세스의 자동화 수준에 그쳐 왔다. [그림 1]에서는 판매/생산/배달 회사 사이의 비지니스 프로세스를 도식하고 있다. 각 기업간의 WFMS가 이기종이라 하더라도 WFMS 간의 상호운용성이 확보된다면 보다 통합적인 수준에서 비지니스 프로세스의 자동화가 구축될 수 있다.



[그림 1] 판매/생산/배달 회사간의 워크플로우

2.2 Wf-XML binding

Wf-XML 규약은 WfMC에서 1999년 4월 초안을 제출한 이후 2000년 5월에 표준으로 확정되었다. Wf-XML은 이기종 워크플로우 시스템간의 상호운용성을 위해 XML[4] 메시지를 사용한다. XML 형태로 전송되는 데이터는 텍스트이므로 플랫폼에 독립적이다.

2.3 SOAP(Simple Object Access Protocol)

SOAP은 분산환경에서의 정보교환을 위한 경량(lightweight)의 프로토콜로써 MicroSoft에서 먼저 제안하였고, 2001년 7월 현재 버전 1.2가 웹 관련 기술 표준 단체인 W3C(World Wide Web Consortium)의 Working Draft 상태에 있다.

SOAP은 XML 기반의 프로토콜로써 플랫폼에 독립적이고, 분산객체들간의 통신을 어렵게 해왔던 방화벽 문제를 HTTP기반의 메시지 전송방식을 사용하여 해결하였다. 또한, 이전에 분산객체들간의 통신을 위해 존재하였던 CORBA, DCOM, Bean등의 표준들은 이기종 시스템간의 통신을 허용하지 않았으나 SOAP을 이용하면 이를 쉽게 구현할 수 있다.

본 연구에서는 이기종 워크플로우 시스템간의 통신을 위해 교환하는 메시지는 Wf-XML에서 규정한 메시지를 이용하고 하부 프로토콜로서 SOAP을 이용하였다. 대상으로 삼은 이기종 시스템은 국내 H사의 Bizflow와 SNUFlow(Seoul National University workFlow management system)를 이용하였다.

3. 워크플로우 상호운용성의 기본 전제

상호 이질적인 워크플로우 시스템 간의 상호운용성을 보장하기 위한 기본 전제는 다음과 같다.

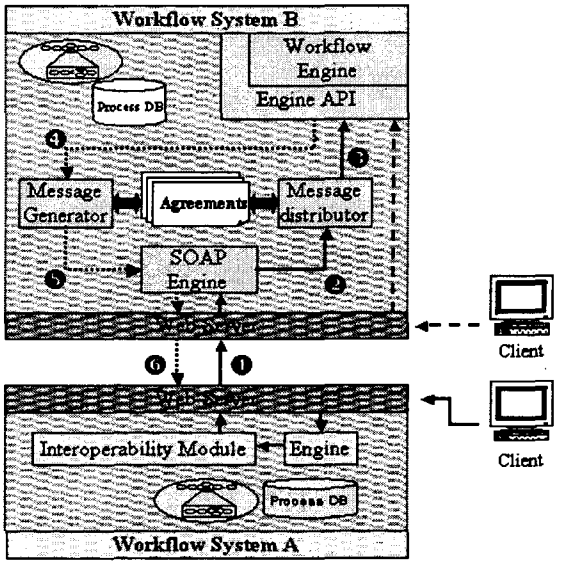
- 외부 워크플로우 시스템과의 독립성
- 내부 프로세스의 은닉
- 메시지 기반의 통합
- 계층적 구조의 프로세스

즉, 상호운용성을 구현하는 데 있어, 각 워크플로우 시스템은 다른 워크플로우 시스템의 구조에 구애받지 않아야 하며(외부 독립성), 또한 내부 프로세스의 구체적인 구조에 대해서는 보안이 이루어질 수 있어야 한다.(내부 은닉성) 이렇게 서로 간에 내부적인 구조를 알 필요 없이 두 시스템이 통합될 때, 상호운용성은 메시지 기반으로 이루어지게 된다. 즉, 각 시스템은 메시지를 통해 상대방에게 상호운용에 대한 요청 및 응답을 하게 된다.(메시지 기반)

메시지 기반의 통합이 이루어질 때, 전체적인 프로세스 구조는 위의 [그림 1]과 같은 계층적인 모습을 띠게 된다.(계층적 구조) [그림 1]에서 판매회사의 프로세스가 상위 프로세스가 되며, 생산회사 결정이란 단위 업무가 실행될 때 생산회사에 프로세스 실행을 요청하게 된다. 결국, 상위 프로세스의 단위업무가 다른 시스템의 전체 프로세스와 대응되는 구조를 통해 각 시스템의 비즈니스 프로세스가 통합될 수 있다.

4. 워크플로우 상호운용성의 시나리오

다음 [그림2]에서는 상호 운용 가능한 시스템의 구조 및 메시지 교환 시나리오를 보여주고 있다.



[그림 2] 시스템 구조 및 메시지 교환 시나리오

메시지 교환의 핵심은 SOAP 엔진, 메시지

해석/분배기(Message Generator), 메시지 생성기의 세부분으로 구성된 Interoperability Module이다. 메시지 해석/분배기는 외부로 부터의 요청(request) 메시지를 해석하여 워크플로우 엔진에 요청하는 역할을 하고, 메시지 생성기는 내부 워크플로우 시스템의 연산 결과에 대한 응답 메시지를 생성하는 역할을 한다.

위 [그림 2]에서 나타난 순서는 WFMS A가 WFMS B에 대해 프로세스 생성을 요구하는 연산 (CreateProcessInstance)에 대한 흐름를 도시한 것이다. 세부적인 내용은 아래와 같다.

① 프로세스 생성 요청 메시지

② 메시지 분배기(Message Distributer)에 메시지 전달

③ 실제 프로세스 생성 API를 호출

④ 생성 요청에 대한 응답

⑤ 응답에 대한 XML 메시지 생성/전달

⑥ 응답 메시지 전달

위에서 ①번 과정에서 두 WFMS간에 전달되는 XML 메시지는 다음 [그림 3]과 같다.

```
<?xml version="1.0"?>
<WfMessage Version="1.0">
  <WfTransport/>
  <WfMessageHeader>
    <Request ResponseRequired="Yes">
    </Request>
    <Key>http://www.WMFSB.com/Wfengine?id=
        1178203
    </Key>
  </WfMessageHeader>
  <MessageBody>
    <CreateProcessInstance.Request
      StartImmediately="true">
    <ObserverKey>http://www.WFMSA.com/wfx456
    </ObserverKey>
    <ContextData>
     ....
    <ContextData>
    </CreateProcessInstance.Request>
  </MessageBody>
</WfMessage>
```

[그림 3] 프로세스 생성 요청에 대한 XML 메시지

실제 WFMS의 상호 운용에는 위에서 사용된 CreateProcessInstance 외에도 여러 연산이 사용된다. 본 원형에서 사용된 연산을 정리하면 다음과 같다.

연산이름	파라메터	설 명
CreateProcessInstance. Request	Key, ContextData, Subject, Name, Description, StartImmediately	프로세스 인스턴스를 생성하라는 연산
CreateProcessInstance. Response	ProcessInstanceKey	프로세스 인스턴스를 생성한 후 Key를 보낸다
GetProcessInstanceData. Request	Key, ResultDataAttributes	Key에 해당하는 프로세스 인스턴스의 정보를 요청, 요청 데이터의 종류는 ResultDataAttribute에 명시
GetProcessInstanceData. Response	ProcessInstanceData	프로세스 인스턴스에 대한 정보를 요청 받았을때 해당 데이터를 보낸다.
ChangeProcessInstance State.Request	Key, State	프로세스 인스턴스의 상태 변경
ChangeProcessInstance State.Response	State	프로세스 인스턴스의 상태 변경 후 결과 통지
ProcessInstanceState Changed.Request	Key, ProcessInstanceKey, State, ResultData, LastModified	프로세스 인스턴스의 상태가 바뀌었는지 바뀐 상태에 대한 데이터를 요청
ProcessInstanceState Changed.Response		프로세스 인스턴스의 상태 변경을 통지

[표 2] 프로세스 정의 연산

5. 시스템 구조 및 구현

5.1 시스템의 구조

메시지 기반의 상호운용성을 구현하는 데 있어, 각 워크플로우 시스템은 상호 규약(Agrement), 메시지, 메시지 처리 기능을 필요로 한다.

즉, [그림 2]에서 보는 것과 같이 상위 프로세스를 구현하고 있는 시스템 A에서 시스템 B에 요청 메시지를 보내게 되면 시스템 B에서는 메시지를 받아 처리하게 된다. 이 때, 메시지를 전달하고 이를 받아 해석하는 부분에 있어 두 시스템 간의 상호 규약이 필요하다.

5.2 상호 규약

워크플로우 시스템 간에 메시지를 주고 받기 위해서는 서로 간에 규약이 사전에 정의되어야 한다. 즉, 상대방 시스템의 ID, URL, 프로세스 이름, 메시지 이름, 교환되는 XML 데이터의 구조 등과 같은 규약을 통해 메시지

가 서로 교환된다.

5.3 메시지

Wf-XML 표준 메시지는 메시지 헤더(Message Header), 본문(Body), 통신(tranport)으로 구성된다.

메시지 헤더는 다시 다음의 요소를 갖게된다.

- 메시지의 종류: 요청(Request) 메시지 인지 응답(Response) 메시지인지의 여부
- 회신이 필요한 메시지인지의 여부
- 키(Key): 시스템의 식별자(identifier)

메시지 본문 부분엔 [표 1]에서 보는 것과 같은 프로세스 연산과 각 연산 부분을 수행하는 데 있어서의 입출력 데이터(Context Data)가 들어가게 된다.

5.4 메시지 처리

메시지를 처리하는 부분은 크게 SOAP 엔진, 메시지 해석/분배기(Message Generator), 메시지 생성기(Message 의 세 부분으로 나뉜다.

메시지 해석/분배기는 외부로부터 들어온 요청메시지를 해석해서 요청에 해당하는 연산을 워크플로우 엔진에 요청하는 역할을 한다.

메시지 생성기는 내부 워크플로우 시스템의 연산 결과에 대한 응답 메시지를 생성하는 부분이다. 즉, 외부 요청에 대한 처리 결과를 응답 메시지로 만들어 전달해주는 역할을 한다.

SOAP 엔진은 모든 입출력 메시지를 전달하는 부분으로, 외부로부터 오는 메시지를 메시지 해석기에 보내주거나, 메시지 생성기로부터 메시지를 받아 외부 시스템에 전달하는 역할을 한다.

6. 결론

본 연구에서는 WfMC에서 제안한 표준인 Wf-XML binding 및 MicroSoft에서 제안한 SOAP을 이용하여 이기종의 워크플로우 시스템간의 상호운용성을 확보한 원형을 구현하였다.

이기종의 워크플로우 시스템간에 전달되는 메시지는 Wf-XML binding에서 제안한 명세를 따름으로써 이후의 확장성을 확보하였으며 메시지의 전달 프로토콜은 SOAP을 이용하여 네트워크의 방화벽 문제를 해결함과 동시에 이기종 시스템간에 효율적인 통신이 가능하도록 하였다.

본 연구의 의의는 새롭게 제안된 SOAP 표준을 통해 시스템의 상호운용에 필요한 XML 메시지들이 분산, 이기종간의 워크플로우 시스템간에 효과적으로 전달됨으로서 워크플로우를 이용한 B2B 및 B2C의 실현이 가능함을 보였고, 이러한 개념을 확장하여 META-flow를 제안한데에 있다.

참고 문헌

1. WfMC, Interoperability Wf-XML Binding, WfMC standards, WFMC-TC-1023, 11-Jan-00, http://www.wfmc.org

2. W3C, SOAP Version 1.2, 9-July-2001, http://www.w3.org/TR/soap12/

3. Extensible Markup Language(XML) 1.0, W3C Recommendation 10-Februrary 1998, http://www.w3c.org/TR/REC-xml.html