워크플로우 상호운용성 지원 프로토콜에 대한 연구 개발 동향

안 형 진* 김 광 훈**

◈ 목 차 ◈

1. 서 론
2. 워크플로우 & BPM 상호운용성
3. ASAP & Wf-XML
4. Wf-XML 2.0 리소스 모델
5. Wf-XML 2.0 개발 현황
6. Wf-XML 2.0 표준 동향 및 향후 추진 방향
7. 결 론

1. 서 론

현재의 비즈니스 환경은 단순한 기업 내부의 프로세스를 통한 업무 자동화를 지원하는 인트라 개념의 워크플로우 & BPM 시스템 뿐만 아니라, 서로 다른 기업 및 조직간에 보유한 프로세스 엔진들 간에 서비스를 상호교환하는 인터-워크플로우 & BPM 시스템의 개념으로 확대되어 비즈니스의 영역은 날로 거대화되어가고 있다. 각 개개의 기업 및 기관들이 사용하는 프로세스 엔진의 종류가 서로 다른 이기종 시스템이라 할 때, 이러한 이기종 프로세스 엔진들 간에 상호운용이 가능하게 하기 위해서 가장 적절하게 사용될 수 있는 방법이 데이터의 표준으로서 각광을 받고 있는 XML을 이용한 바인딩이다. XML은 현재 이슈가 되고 있는 웹 서비스의 근간이 되는 데이터 표준으로서 서로가 보유하고 있는 시스템의 환경이 다르더라도 XML을 이용하게 되면 상호 간의 통신이 가능해지게 된다.

비즈니스 프로세스 상호운용을 통한 서비스의 특성 상 업무 요청에 대해서 긴 시간의 응답 시간을 필요로 하는 업무들이 많기 때문에, 동기적인 업무 처리 방식 뿐만 아니라 비동기적인 처리 방식의 서비스 제공 또한 필요하다. 프로세스 상호운용 지원 환경에서의 비동기 서비스를 가능하게 하기 위하여 워크플로우 표준 단체인 WfMC(Workflow Management Coalition)에서는 OASIS 표준 단체에서 제안하는 비동기 웹 서비스 프로토콜인 ASAP (Asynchronous Service Access Proticol)을 기반으로 한 Wf-XML을 상호운용성 구현의 표준으로서 제시하고 있으며, 현재 2.0 버전이 배포된 상태이다.

본 고에서는 워크플로우 & BPM 상호운용성 및 상호운용성 모델에 대해 설명하고, 이러한 프로세스 엔진 간 상호운용성 지원이 가능하도록 하는 수단으로서 사용되는 Wf-XML이라는 프로토콜을 소개한다. Wf-XML 프로토콜의 기반이라 할 수 있는 ASAP라는 비동기 웹 서비스 프로토콜에 대해 설명하고, ASAP를 비즈니스 프로세스 영역에 적합하게 표준화시킨 Wf-XML 2.0 프로토콜을 구체적으로 살펴보도록 한다. 그리고, Wf-XML 2.0을 이용하여 동종 또는 이기종 프로세스 엔진 간 상호운용성 서비스가 어떻게 이루어지는 지에 대하여 기술하고자 한다.

2. 워크플로우 & BPM 상호운용성

워크플로우 & BPM 상호운용성은 각 프로세스 엔진들 간의 인엑트먼트 서비스에 대한 상호 작용

* 경기대학교 대학원 전자계산학과 석사과정
** 경기대학교 정보과학부 조교수, 부교수

을 의미한다. 워크플로우 & BPM 상호운용성은 둘 이상의 프로세스 엔진들 간의 서비스 교환을 통해 이루어지며, 각 프로세스 엔진들은 동종의 시스템일 수도 있고, 이기종의 시스템일 수도 있다. 또한 중첩 시스템의 형태로서, 여러 프로세스 엔진이 그룹화되어 인엑트먼트 서비스를 제공할 수도 있다. 동종의 프로세스 엔진들 또는 동종의 인엑트먼트 서비스를 지원하는 프로세스 엔진 환경일 경우에는 상호간의 통신에 아무 하자가 생기지 않는다. 직접적으로 메소드 호출을 통해 실제적인 인스턴스 데이터들이 네트워크를 통해 교환되어 서비스 요청에 대한 응답을 상호 간에 가능하게 할 수 있다. 다음의 그림 1은 동종의 인엑트먼트 서비스를 제공하는 중첩 프로세스 엔진의 상호운용을 나타내고 있다.

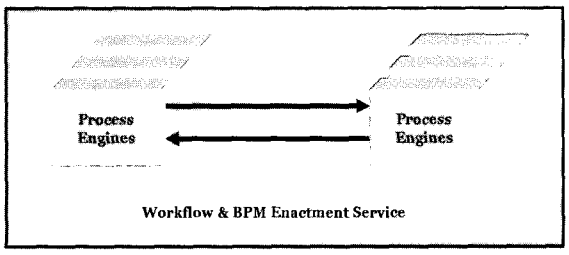
그러나, 비즈니스 프로세스 환경에서 각 개개의 기업들이 모두 동일한 프로세스 엔진을 사용할 리는 만무하다. 여러 벤더들이 제공하는 프로세스 엔진이 존재하고 사용하는 회사들은 다양한 프로세스 엔진들을 사용하게 되며, 한 기업 내에서도 사용되는 프로세스 엔진들이 각각 다른 기종의 엔진일 수도 있다. 아래의 그림 2는 이기종 간의 인엑트먼트 서비스를 수행하는 중첩 프로세스 엔진의 상호운용을 나타낸다.

이기종의 프로세스 엔진들 또는 중첩 프로세스 엔진들 간의 상호운용일 경우에는 동종의 시스템들에서 이루어지는 상호운용의 통신과는 다른 메커니즘으로 수행된다. 서로 다른 시스템들이 의사 소통이 가능하기 위해서는 각각의 시스템들 사이에 교량 역할을 할 수 있는 매개체가 필요하며 이러한 상황을 만족시킬 수 있는 환경이 XML 기반의 웹 서비스 환경이라 할 수 있다.

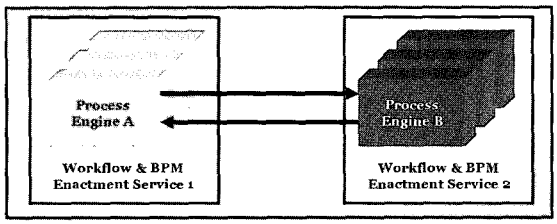
3. ASAP & Wf-XML

웹 서비스 프로토콜은 크게 두 가지로서 존재하는데 그것은 바로 동기적 웹 서비스프로토콜(Synchronous Web Services Protocol)와 비동기적 웹 서비스(Asynchronous Web Services Protocol)이다. 이 두 가지 프로토콜의 수행 처리 방식은 요청 및 응답 처리 방식에 의해서 구분되어진다. 동기적 웹 서비스 프로토콜은 클라이언트가 서비스에 요청을 보내고 그에 대한 응답이 올 때까지 대기하여 처리하는 방식이며, 비동기적 웹 서비스는 클라이언트가 서비스에 대한 요청을 보낸 뒤 응답이 올 때까지 기다리지 않고 다른 작업을 계속 수행해나갈 수 있게끔 지원해주는 방식이다. 동기적 웹 서비스 프로토콜은 클라이언트가 서비스를 호출한 뒤 요청에 대한 응답을 받을 때까지 대기하는 것이 그 특성이다. 이에 따라 클라이언트는 요청을 보낸 뒤 잠시 처리를 중지하므로, 동기적 웹 서비스 프로토콜은 서비스가 짧은 시간 내에 처리될 수 있는 곳에서 유리하게 사용될 수 있는 프로토콜이다. 또한 동기적 웹 서비스 프로토콜은 애플리케이션이 요청에 대한 즉각적인 처리를 필요로 할 때 유용하게 사용될 수 있다. 아래의 그림 3은 동기적 웹 서비스 프로토콜을 이용한 서비스 처리에 대한 예제를 그림으로 나타낸 것이다.

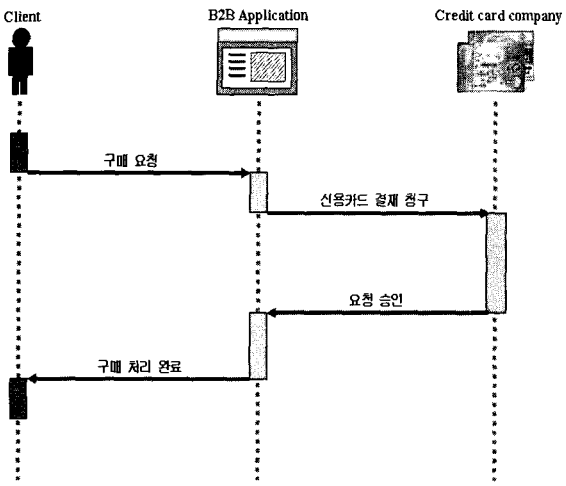
그림 3에서 보는 바와 같이, 쇼핑몰에서 고객이 어떠한 물건을 구매하고자 주문을 하고 결재를 신용카드로 한다고 가정할 경우, 고객의 정보를 인증하고 카드 결재에 대한 서비스 처리가 완료될 때까지 고객은 대기하게 되며 쇼핑몰 측에서는 처리가 완료되면 최종적으로 구매하고자 한 물건에 대한



(그림 1) 동종의 인엑트먼트 서비스를 통한 상호운용



(그림 2) 이기종의 중첩 워크플로우들 간의 인엑트먼트 서비스를 통한 상호운용



(그림 3) 동기적 웹 서비스를 이용한 전자상거래 예제

처리가 성공적으로 완료되었음을 고객에게 알리게 된다. 이러한 일련의 과정은 동기적인 서비스 방식을 통해 이루어지게 됨을 알 수 있다.

비동기 웹 서비스 프로토콜은 타 비즈니스 영역간의 서비스 통신을 완료하는데 오랜 시간이 걸리는 속성의 서비스에 접근하는데 이용될 수 있는 웹 서비스 프로토콜이다. 우리가 흔히 알고 있는 웹 서비스 프로토콜은 서비스 요청에 대한 응답에 걸리는 시간이 1분 내외에서 적어도 몇 분 내이다. 이에 반해서, 비동기 웹 서비스 프로토콜은 응답 시간이 앞서 얘기한 일반적인 웹 서비스와는 다르게 응답 시간이 훨씬 더 걸리는 경우, 즉 응답 시간이 적어도 수 분에서 한 달 또는 그 이상의 시간이 필요한 경우에 효율적으로 사용될 수 있는 프로토콜이다. 현재의 비즈니스 환경은 기업 내부 뿐만 아니라 여타의 기업들과 서비스를 상호 교환하며 글로벌하게 운영되는 가운데, 각 기업들 간에 일어나는 서비스에 대한 응답은 반드시 요청이 생기게 되면 바로 응답을 하는 방식의 서비스들도 많이 존재하나, 장시간의 트랜잭션을 유지해야 하는 비즈니스 서비스도 무수히 존재하므로 이러한 긴 시간을 요하는 업무 처리에는 비동기 웹 서비스 프로토콜을 사용하는 것이 좋은 해결 방법이라 할 수 있다.

ASAP(Asynchronous Service Access Protocol)은 국제 표준 단체인 OASIS에서 제안하는 비동기 웹 서비스 프로토콜에 대한 표준이며, Wf-XML은 워크플로우 표준 단체인 WfMC(Workflow Management Coalition)에서 OASIS의 ASAP 비동기 웹 서비스 프로토콜을 기반으로 비즈니스 환경에서 일어나게 되는 상호운용성을 위한 구현에 적합하도록 제안된 워크플로우 및 BPM에 대한 상호운용성 표준이다. Wf-XML은 ASAP가 가지는 웹 서비스를 통한 비동기 서비스 구현의 모습을 그대로 가지고 있기 때문에, 비즈니스 기업 조직들 간에 서로 다른 프로세스 엔진을 가지고 있다 하더라도 ASAP를 통한 웹 서비스 메세징 기술을 통하여 상호 통신을 통한 서비스 요청 및 응답이 가능하게 된다. 이기종 워크플로우 & BPM 관리 시스템들 간의 상호운용성을 위한 WfMC 표준인 Wf-XML은 현재 2.0버전이 제작되어 있는 상태이다.

Wf-XML 2.0의 특징들을 나열해보면 다음과 같다.

- Wf-XML 2.0은 ASAP에서 비즈니스 프로세스와 관련없는 불필요한 부분을 제외시키고 워크플로우 & BPM 상호운용성 지원에 적합한 형태로 재구성한 프로토콜로서, 기존의 ASAP가 가지는 XML 및 SOAP의 특징들을 모두 포함한다.
- Wf-XML 2.0 프로토콜은 어떠한 기종의 프로세스 엔진이라도 구현 가능할 수 있도록 하는 표준이다.
- Wf-XML 2.0은 상호운용성과 관련된 서비스를 지속적으로 확장 가능할 수 있게끔 해주는 프로토콜이다.
- Wf-XML 2.0은 웹 서비스 기반의 메시징 기술을 기반으로 구현되기 때문에 어떠한 플랫폼에서도 사용 가능한 프로토콜이다.

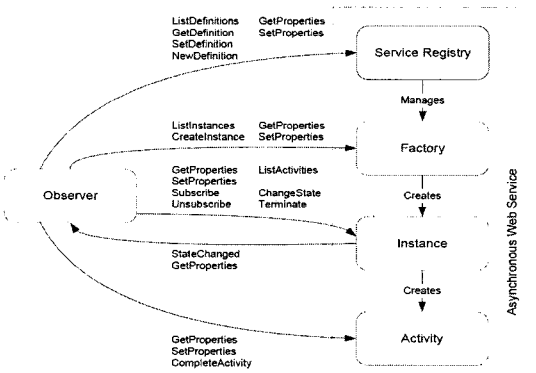
위에서 설명한 바와 같이 Wf-XML 2.0은 XML과 SOAP을 기반으로 구성됨으로서 확장성 및 유연성이 뛰어나며, 특정 플랫폼을 구분하지 않기 때문에 임의의 플랫폼에 위치한 프로세스 엔진들 간의 상호운용성에 대한 처리가 가능하고, 비동기 웹 서비스를 지원하기 때문에 장시간의 트랜잭션이 필

요한 비즈니스 서비스 수행에 적합한 프로토콜이라고 할 수 있다.

4. Wf-XML 2.0 리소스 모델

워크플로우 & BPM 상호운용성을 위한 비동기 웹 서비스 인터페이스를 구현하는데 필요한 5가지 기본 리소스들은 옵저버(Observer), 서비스 레지스트리(Service Registry), 팩토리(Factory), 인스턴스(Instance), 액티비티(Activity)이다. 이 5개의 리소스들 중에서 옵저버, 팩토리, 인스턴스 이렇게 3개의 리소스들은 ASAP의 기본 구성 리소스들이며, ASAP를 워크플로우 상호운용성에 적절하게 표준화시킨 Wf-XML에 서비스 레지스트리와 액티비티 2개의 리소스가 개념적으로 추가되었다. Wf-XML 2.0은 이와 같은 5가지 리소스들을 기반으로 상호운용성 모델을 구성하고 있다. 다음의 그림 4는 Wf-XML 2.0 기반 프로세스 엔진의 리소스 모델을 나타낸다.

"옵저버(Observer)"는 비즈니스 프로세스와 관련된 서비스가 수행되는 동안에 발생되는 이벤트 및 외부로부터 워크플로우 & BPM 시스템의 서비스를 이용하기 위해 요청되는 서비스에 대해 감지하고 해석하는 역할을 하는 리소스이다. 서비스에 대한 응답을 요구하는 측에서는 비동기 웹 서비스 프로토콜 기반의 Wf-XML 2.0을 이용하여 해당 서비스에 대한 내용을 메시지로서 구성하여 전송하게



(그림 4) Wf-XML2.0 기반 프로세스 엔진의 리소스 모델

되고, 옵저버는 Wf-XML 메시지를 받아들여 해석하고 이에 상응하는 서비스를 수행하여 결과를 돌려주게 된다. "서비스 레지스트리(Service Registry)"는 서비스 제공을 위한 모체가 되는 프로세스 정의 데이터의 저장소 역할을 하는 리소스이다. 이 서비스 레지스트리에는 기존에 존재하는 프로세스 정의뿐만 아니라, 모델링 툴을 이용하여 새로운 프로세스 정의를 추가할 수도 있다. 서비스 레지스트리로부터 관리되는 프로세스 정의가 워크플로우 & BPM 시스템에 의해 가용 데이터(모델 데이터)로서 존재하게 되는데, 이것을 "팩토리(Factory)" 리소스라고 한다. 워크플로우 & BPM 시스템 엔진은 팩토리 리소스인 모델 데이터를 통해 프로세스 인스턴스를 생성하여 실질적인 비즈니스 서비스를 수행하게 되는데 이 프로세스 인스턴스를 "인스턴스(Instance)" 리소스라고 한다. 인스턴스 리소스는 서비스의 처리 주체가 되어 비즈니스 프로세스의 제어 흐름에 의해 업무가 처리되는데, 이러한 업무는 액티비티 인스턴스인 "액티비티(Activity)" 리소스로서 존재하여 해당 업무에 대한 수행을 진행하게 된다. 이와 같은 리소스 타입 모델은 Wf-XML 2.0 이라는 비동기 메시지 프로토콜을 이용하여 이기종의 워크플로우 & BPM 시스템 간에 상호운용성 지원이 가능하다는 것을 보여준다.

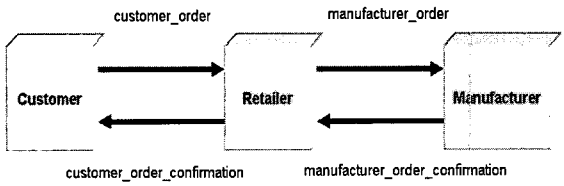
5. Wf-XML 2.0 개발 현황

2004년 6월 미국 샌프란시스코에서 워크플로우 & BPM 상호운용성에 대한 구현 관련 데모가 있었다. 이 데모에는 여러 워크플로우 및 BPM 업체들이 참여하였고 이 분야에 관계하는 사람들은 많은 관심을 나타내었다. 샌프란시스코에서의 데모는 Wf-XML 2.0을 이용해서 각 여러 벤더들끼리 비즈니스 서비스에 대한 상호운용성이 원활히 이루어질 수 있으며 또한 워크플로우 & BPM 상호운용성이 손쉽게 구현 가능하다는 걸 직접 확인시켜줌으로서 앞으로의 워크플로우 & BPM 시장에 더욱 박차를 가할 수 있는 계기를 마련하게 되었다고 할 수 있다. 샌프란시스코에서의 상호운용성 데모의

성공에 힘입어 올 2005년 2월 28일에는 이기종의 프로세스 엔진 간에 각각 다른 워크플로우 & BPM 모델링 도구를 통해 정의된 프로세스들에 대한 정보를 주고받는 상호운용성 데모를 성공적으로 마침으로서 갈수록 더 나은 비전을 제시해가는 중이다. 본 고에서는 2004년 6월의 상호운용성 데모 예제를 토대로 Wf-XML 2.0을 이용해 어떻게 워크플로우 & BPM 상호운용성을 구현하는지에 관해 설명하고자 하며, 시나리오 구현의 일련의 과정 중 주문 처리 프로세스의 생성 및 시작이 어떠한 방식을 통해 이루어지는지에 대해 구체적으로 살펴보겠다.

데모에 대한 시나리오는 다음과 같다. 물건을 구매하려는 고객이 자신의 마음에 드는 제품을 주문하기 위해 해당 물품에 대한 정보 및 고객 정보를 소매 기업의 웹 사이트를 통해 입력하게 되면 입력된 정보가 소매 기업에 전달되고, 전송된 제품과 고객 정보에 대한 확인 절차가 끝나게 되면 주문에 대한 처리가 이어지게 되며, 만약 주문된 제품이 대량으로 생산되어야 하는 제품일 경우엔 제품 제조 회사에 대량 생산을 위한 주문 제품의 정보를 보내어 업무를 수행하게 하며, 이러한 일련의 절차들은 워크플로우 & BPM 시스템에 의해 시뮬레이션이 가능하게 된다. 지금까지의 시나리오를 간단히 그림으로 표현한 것이 아래와 같으며, 이와 같은 주문 거래 과정에 대한 처리가 워크플로우 & BPM 상호운용성을 지원하는 Wf-XML 2.0 메시지 프로토콜을 통해 수행된다.

고객의 주문을 통한 제품 구입은 그에 대한 프로세스 정의가 존재하며 이러한 프로세스 정의는 워크플로우 & BPM 시스템에 의해 실질적으로 프로세스 인스턴스화 되어 실제적인 주문 처리 작업을 시작하게 된다.



(그림 5) Wf-XML 2.0 데모 시나리오도

첫번째 메시지는 고객이 제품에 대한 정보를 얻기 위해 소매 기업에 위치한 팩토리에게 컨텍스트 데이터를 요청하는 것으로서 이에 대한 메시지의 스키마는 다음과 같다.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://harness/customer</as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Retailer
   </as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:GetPropertiesRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"/>
 </soap:Body>
</soap:Envelope>
```

소매 기업 측의 팩토리에서는 고객으로부터 요청받은 작업을 수행하고 제품 정보에 대한 컨텍스트 데이터인 프로세스 정의의 URI 및 응답 데이터 메시지를 구성하여 다음과 같이 돌려주게 된다.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <as:Response>
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Retailer</as:SenderKey>
   <as:ReceiverKey>http://harness/customer</as:ReceiverKey>
  </as:Response>
 </env:Header>
 <env:Body>
  <as:GetPropertiesRs>
   <as:Key>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=Retailer
   </as:Key>
   <as:Name>Retailer</as:Name>
   <as:Subject>Retailer process</as:Subject>
   <as:Description>Business process to accept a customer order and fulfill it,
    placing a manufacturer order if necessary.
   </as:Description>
   <as:ContextDataSchema>
    <xsd:schema
     targetNamespace="http:SAMEERP:9004/iflowjsp/jsp/cds.jsp?planName=Retailer"
     xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/cds.jsp?planName=Retailer"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified">
     <xsd:complexType name="ContextDataType">
      <xsd:sequence>
       <xsd:element name="customer first name" type="xsd:string" minOccurs="0"/>
       <xsd:element name="customer surname" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address first line" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_second_line" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_city" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_zipcode" type="xsd:string" minOccurs="0"/>
       <xsd:element name="phone number" type="xsd:string" minOccurs="0"/>
       <xsd:element name="product code" type="xsd:int" minOccurs="0"/>
       <xsd:element name="product description" type="xsd:string" minOccurs="0"/>
       <xsd:element name="product quantity" type="xsd:int" minOccurs="0"/>
       <xsd:element name="order date" type="xsd:date" minOccurs="0"/>
        minOccurs="0"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>
   </as:ContextDataSchema>
   <as:ResultDataSchema>
    <xsd:schema
     targetNamespace="http:SAMEERP:9004/iflowjsp/jsp/rds.jsp?planName=Retailer"
     xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/rds.jsp?planName=Retailer"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified">
     <xsd:complexType name="ResultDataType">
      <xsd:sequence>
       <xsd:element name="customer first name" type="xsd:string" minOccurs="0"/>
       <xsd:element name="customer surname" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address first line" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address second line" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_city" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_zipcode" type="xsd:string" minOccurs="0"/>
       <xsd:element name="phone number" type="xsd:string" minOccurs="0"/>
       <xsd:element name="order date" type="xsd:date" minOccurs="0"/>
       <xsd:element name="retailer name" type="xsd:string" minOccurs="0"/>
       <xsd:element name="order number" type="xsd:int" minOccurs="0"/>
       <xsd:element name="est delivery date" type="xsd:date" minOccurs="0"/>
       <xsd:element name="product code" type="xsd:int" minOccurs="0"/>
       <xsd:element name="product description" type="xsd:string" minOccurs="0"/>
       <xsd:element name="product quantity" type="xsd:int" minOccurs="0"/>
       <xsd:element name="price_per_unit" type="xsd:float" minOccurs="0"/>
       <xsd:element name="order_price" type="xsd:float" minOccurs="0"/>
       <xsd:element name="in stock" type="xsd:boolean" minOccurs="0"/>
       <xsd:element name="manufactured by" type="xsd:string" minOccurs="0"/>
       <xsd:element name="manufacturer code" type="xsd:int" minOccurs="0"/>
       <xsd:element name="manufacturer receipt date" type="xsd:date"
        minOccurs="0"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>
   </as:ResultDataSchema>
   <as:Expiration>P126Dc/as:Expiration>
  </as:GetPropertiesRs>
 </env:Body>
</env:Envelope>
```

팩토리로부터 보내진 주문 제품에 대한 정보를 토대로 고객은 제품을 주문하게 되는데, 이것은 주문 프로세스에 대한 인스턴스를 생성하여 시작 구동시킴으로서 실제적인 서비스를 시작시킴을 의미한다. 프로세스 인스턴스를 생성하기 위한 요청 오퍼레이션이 메시지의 바디에 태그로서 구성되며, 인스턴스 발생에 필요한 매개변수와 컨텍스트 데이터들이 오퍼레이션의 하위 태그들로서 구성되어진다. 주문 프로세스 인스턴스의 시작에 대한 메시지는 다음과 같이 구성된다.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://harness/customer</as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Retailer
   </as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:CreateInstanceRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:StartImmediately>true</as:StartImmediately>
   <as:ObserverKey>http://harness/customer.observer</as:ObserverKey>
   <as:Name>TestRetailer</as:Name>
   <as:Subject>Test retailer process</as:Subject>
   <as:Description>Test retailer process with proper context data</as:Description>
   <as:ContextData
    xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/cds.jsp?planName=Retailer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pd:customer first name>Bill</pd:customer first name>
    <pd:customer surname>Smith</pd:customer surname>
    <pd:address first line>285 Garnet Grove</pd:address first line>
    <pd:address second line>Suite 48</pd:address second line>
    <pd:address city>San Jose</pd:address city>
    <pd:address zipcode>95134</pd:address zipcode>
    <pd:phone number>(408)-555-2476</pd:phone number>
    <pd:product_code>245</pd:product_code>
    <pd:product_description>Lawn mower</pd:product_description>
    <pd:product quantity>1</pd:product quantity>
    <pd:order date>2004-06-01</pd:order date>
   </as:ContextData>
  </as:CreateInstanceRq>
 </soap:Body>
</soap:Envelope>
```

워크플로우 & BPM 시스템은 고객으로부터 제품 주문에 대한 요청을 통해서 전달된 주문 프로세스 인스턴스의 생성을 수행하고 그에 대한 응답 메시지를 다음과 같이 구성하여 고객 측에 돌려주게 된다.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <as:Response>
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=Retailer
   </as:SenderKey>
   <as:ReceiverKey>http://harness/customer</as:ReceiverKey>
  </as:Response>
 </env:Header>
 <env:Body>
  <as:CreateInstanceRs>
   <as:InstanceKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:InstanceKey>
  </as:CreateInstanceRs>
 </env:Body>
</env:Envelope>
```

주문 프로세스 인스턴스가 생성된 후 임의의 시간이 지난 뒤, 해당 인스턴스가 시작되었음을 알리는 메시지를 고객에게 응답 메시지를 통해 알리게 되며 이러한 응답은 비동기식으로 이루어지게 된다.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:SenderKey>
   <as:ReceiverKey>http://harness/customer.observer</as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:CompletedRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:InstanceKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:InstanceKey>
   <as:ResultData
    xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/rds.jsp?planName=Retailer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pd:customer first name>Bill</pd:customer first name>
    <pd:customer surname>Smith</pd:customer surname>
    <pd:address first line>285 Garnet Grove</pd:address first line>
    <pd:address second line>Suite 48</pd:address second line>
    <pd:address city>San Jose</pd:address city>
    <pd:address_zipcode>95134</pd:address_zipcode>
    <pd:phone_number>(408)-555-2476</pd:phone_number>
    <pd:order date>2004-06-01</pd:order date>
    <pd:retailer name>ACME Systems</pd:retailer name>
    <pd:order number>87404584</pd:order number>
    <pd:est delivery date>2004-06-15</pd:est delivery date>
    <pd:product code>245</pd:product code>
    <pd:product description>Lawn mower</pd:product description>
    <pd:product quantity>1</pd:product quantity>
    <pd:price_per_unit>59.99</pd:price_per_unit>
    <pd:order_price>59.99</pd:order_price>
    <pd:in_stock>false</pd:in_stock>
    <pd:manufactured by>ALPHA Manufacturing</pd:manufactured by>
    <pd:manufacturer code>2734</pd:manufacturer code>
    <pd:manufacturer receipt date>2004-06-10</pd:manufacturer receipt date>
   </as:ResultData>
  </as:CompletedRq>
 </soap:Body>
</soap:Envelope>
```

지금까지 주문 처리 프로세스에 대한 서비스 인스턴스를 생성 및 시작하는 프로세스를 Wf-XML 2.0 프로토콜을 이용해 어떻게 구현하는지에 대해서 살펴보았다. 제품 주문 처리 과정에 대한 다른 과정들 또한 Wf-XML 2.0 메시지를 이용한 상호운용을 통해 서비스 구현을 할 수 있다.

워크플로우 & BPM 상호운용성 구현 데모에 대해 자세히 알고자 한다면 WfMC 공식 웹 사이트를 방문하면 보다 자세한 내용을 확인할 수 있을 것이다.

6. Wf-XML 2.0 표준 동향 및 향후 추진 방향

ASAP와 Wf-XML은 Netscape, Oracle 및 여러 다른 기관들에 의하여 SWAP(Simple Workflow Access Protocol)이라는 명명 하에 추진되었으며, 현재의 ASAP와 Wf-XML은 1997년에 IETF(Internet Engineering Task Force)에서 추진한 성과이다. SWAP은 다수의 상업적 제품들에 적용되었는데 WfMC는 이러한 SWAP에서 아이디어를 가져

와 그 적용 범위를 상당히 축소하여 Wf-XML을 정의하게 되었다. SWAP은 SOAP 이전에 개발되어졌기 때문에 SOAP의 메시지 구조를 따르지 않고 있지만, ASAP는 SOAP의 메시지 구조를 기반으로 시도되었었다. 이에 Wf-XML은 기존의 Wf-XML을 ASAP에 적용시키기 위하여 시도되었으며, SOAP을 기반으로 하고 있다. 현재까지 대략 20개 이상의 개인 회사가 ASAP 표준화에 관여하고 있으며, 기술위원회에는 Amberpoint, British Telecom, Cisco, Computer Associates, Fujitsu, Iway, Lockheed Martin, Research in Motion 그리고 The University of Hong Kong 등이 포함되어있다. 현재 다수의 정부, 기관, 대학에서 ASAP에 관한 사항 및 표준화 그리고 상품화를 추진하고 있으며, 이에 따른 국내에서의 표준화 및 상품화가 추진되어야 할 것으로 기대된다.

워크플로우 시스템 간의 상호운용성에 대한 표준화는 많은 다른 종류의 시스템 및 서비스와의 연계에 초점을 맞추고 있다. 이에 워크플로우 상호운용성을 위한 ASAP 표준화는 단순 연결 워크플로우와 중첩 워크플로우의 지원, 동기 및 비동기 상호작용 제공, 그리고 구현에 대한 독립성의 유지 등과 같은 특징뿐만 아니라, XML을 활용한 정보의 교환 및 비동기 정보 통신을 요구하는 다른 SOAP 기반의 프로토콜과의 상호운영을 지원할 것이다. 워크플로우 시스템의 이기종 간 상호 운용성 및 사용자의 서비스 요청에 대해서, 비동기 웹 서비스인 ASAP 기반 Wf-XML 2.0 이라는 비동기 메시지 프로토콜을 통하여 장시간의 트랜잭션을 요하는 프로세스 관련 작업 수행에 대한 비동기식 처리에 의해 보다 효율적인 비즈니스 업무 처리를 할 수 있다. 이러한 비즈니스 프로세스 관련 비동기식 서비스 처리는 Wf-XML 2.0 표준에서 제안하는 워크플로우 시스템 리소스 모델을 기반으로 하여 워크플로우 시스템을 구성하는 주요 컴포넌트들이 리소스 모델의 각 리소스들과 연계되어 비동기적 서비스를 제공할 수 있게 된다. ASAP 및 ASAP 관련 기술의 표준화는 국내의 워크플로우 솔루션 업체들과 정보기술 관련 기업체들에게 비동기 웹 서비스 접근 프로토콜 기술에 대한 방향을 제시할 것이며, 또한 국내 솔루션 업체들 간의 상호교류 그리고, 국내 표준규격화 활동을 통하여 국내 전자정부, 공기업정보화, 대기업 및 중소기업, 교육기관 등의 산업분야별 업무처리 프로세스 표준화 및 국내 정보기술의 효율적 정보화 자동화의 전개에 기여할 것으로 기대된다.

7. 결 론

급변하고 거대화되어가는 비즈니스 프로세스 환경은 기업 내부에서의 업무 처리뿐만 아니라, 더 나아가 기업과 기업간의 상호교환을 통한 업무 수행을 요구하게 되었고 이러한 요구 사항에 적절하게 대응할 수 있는 전략으로서의 개념이 워크플로우 & BPM 상호운용성이다. 워크플로우 & BPM 상호운용성은 둘 이상의 프로세스 엔진들 간의 서비스 상호 교환을 의미한다. 이기종의 프로세스 엔진 간에 서비스 교환이 원활하게 이루어질 수 있는 방법은 현재 가장 널리 사용되고 있는 웹을 이용하는 방법이며, 이를 바탕으로 상호운용성의 기반이 되는 기술로서 웹 서비스가 접목된다. 오늘날의 비즈니스 관련 서비스는 짧은 기간 내에 요청에 대한 응답이 가능한 업무들도 많지만 또한 장시간의 트랜잭션을 요하는 업무들 또한 증가하는 추세이기 때문에 이에 따라 비동기 웹 서비스를 이용하여 비즈니스 영역 간의 상호운용성 구현이 이슈가 되고 있다.

따라서, 워크플로우 & BPM 분야에서는 비동기 웹 서비스 프로토콜인 ASAP를 기반으로 비즈니스 환경에 더욱 적합하도록 만들어진 Wf-XML이라는 표준 메시지 프로토콜을 제안하여 워크플로우 및 BPM 상호운용성을 구현할 수 있도록 하고 있다. 본 고에서는 Wf-XML 2.0에 대한 리소스 모델 및 Wf-XML 2.0을 이용한 구현 데모에 대한 예제를 살펴보면서 Wf-XML 2.0을 이용하여 워크플로우 & BPM 상호운용성에 대한 구현을 어떻게 하는지에 대해서 살펴보았다.

현재 워크플로우 & BPM 해외 시장은 여러 정부 및 기관에서 표준화 및 상품화를 활발하게 추진 중

이나 국내에서는 아직 표준화와 상용화가 추진되어져야 하는 시작 단계에 있다. 그러므로 국내에서의 발빠른 표준화 작업과 국내 벤더들의 상용화를 위한 활발한 움직임과 관심이 필요한 때라고 할 수 있다. ASAP & Wf-XML과 같은 비동기 서비스 처리 방식을 통한 비즈니스 상호운용성의 발전은 앞으로 국내의 비즈니스 프로세스 업계의 더 나은 발전뿐만 아니라 국제적으로도 기업의 자동화 및 정보화 발전에 큰 기여를 할 수 있을 것으로 전망한다.

참 고 문 헌

[1] "Workflow Standard - Interoperability Abstract Specification", The Workflow Management Coalition

[2] "Simple Workflow Access Protocol (SWAP)", Keith D Swenson

[3] "Wf-XML 2.0 - XML Based Protocol for Run-Time Integration of Process Engines", Keith D Swenson, Sameer Predhan

[4] "ASAP / Wf-XML 2.0 Cookbook", Keith D Swenson, Fujitsu Software Corporation

[5] "Workflow Standard - Interoperability Internet e-mail MIME Binding", The Workflow Management Coalition

[6] "The Workflow Reference Model", The Workflow Management Coalition

[7] "Simple Object Access Protocol(SOAP) version 1.2 Part 1 : Messaging Framework", W3C Working Draft 17

◐ 저 자 소 개 ◑

안 형 진

2004년 경기대학교 전자계산학과 학사
2004년~현재 경기대학교 대학원 전자계산학과 석사과정

김 광 훈

1984년 경기대학교 전자계산학과 학사
1986년 중앙대학교 대학원 전자계산학과 석사
1994년 University of Colorado at Boulder, Computer Science, MS
1998년 University of Colorado at Boulder, Computer Science, Ph.D
1986년 2월~1991년 8월 한국전자통신연구원
1993년 5월~1994년 8월 American Educational Products, Inc., Professional DB Consultant
1994년 9월~1995년 8월 Colorado Advanced Software Institute, Research Assistant
1995년 9월~1997년 2월 Aztek Engineering, Inc., Software Engineer
1998년 3월~현재 경기대학교 정보과학부 조교수, 부교수