워크플로우 상속을 이용한 확장적 협업 프로세스 구성+

Extensible Collaborative Process Composition using Workflow Inheritance

김훈태*, 정재윤**, 강석호**
*대진대학교 산업시스템공학과, **서울대학교 산업공학과

Abstract - e-비즈니스가 확산되면서 비즈니스 프로세스의 효과적인 관리가 요구되고 있다. 기업은 수많은 비즈니스 프로세스를 생성하고 변경, 삭제한다. 나아가 여러 거래 기업과의 관계 변화에 신속하고 유연하게 대응하기 위해서는 비즈니스 프로세스를 빠르게 정의하고 체계적으로 관리할 수 있는 방법이 요구된다. 본 연구에서는 워크플로우 상속을 이용하여 프로세스를 확장하는 방법을 제시하고, 기업간 환경에서 협력적 워크플로우를 작성하고 웹 서비스를 통해 협업을 수행하는 방안을 제안한다. 워크플로우 상속은 프로세스의 추상화와 다형성, 재사용성을 부여함으로써 프로세스를 효과적으로 확장하고 실행할 수 있게 한다. 그리고 기존에 보유하고 있던 검증된 워크플로우를 재사용함으로써 기업간 거래를 위한 유연한 워크플로우를 수행할 수 있도록 도와준다.

1. 서론

기업은 수많은 비즈니스 프로세스를 유지하고 있으며, 끊임없이 새로운 프로세스를 작성한다. 기업의 프로세스 설계자들은 보다 쉽고 효율적으로 프로세스를 설계하고, 효과적으로 유지 보수할 수 있는 프로세스를 보유하기를 원한다. 그러한 측면에서 이미 존재하는 프로세스를 조합하고 확장하는 것은 급변하는 기업간 거래 환경에 유연하게 적응할 수 있는 경쟁력을 제공할 것이다.

본 연구에서는 워크플로우 상속을 이용하여 프로세스를 확장하는 방법을 제시하고, 기업간 환경에서 이를 작성하고 수행하는 방안을 제안한다. 워크플로우 상속이란 빌드타임(build-time)이나 런타임(run-time)에 외부 프로세스를 사용하여 대상 프로세스의 일부를 상세화할 수 있도록 허용함으로써, 워크플로우를 확장하는 것이다. 본 논문에서는 워크플로우를 프로세스 수준에서 확장할 수 있도록 여섯 가지 기본적인 프로세스 상호운용성 패턴을 함께 제시하였다.

조직간 워크플로우(IOWF; Inter-Organizational Workflow)에 관한 연구는 워크플로우 상호운용성을 중심으로 많은 연구가 있었다. 프로세스의 액티비티 구성을 Perti-net으로 분석한 또다른 개념의 워크플로우 상속에 관한 연구가 있었다[1]. 또한 실행시간캡슐화를 이용한 워크플로우에 관한 연구는 본 논문의 기반을 제공한다[2,3].

본 논문의 접근방법은 가변적인 기업간 거래 환경에서 비즈니스 프로세스를 유연하게 설계하고 변형할 수 있도록 지원한다. 이와 같은 비즈니스 프로세스의 구성과 재사용에 관한 연구는 조직간, 기업간 협업을 위한 프로세스를 효과적으로 유지하고 관리하는 데 이바지할 것이다.

2. 기업간 비즈니스 프로세스

프로세스란 특정한 목적을 달성하는 데 필요한 정형화된 액티비티 구성이다. 비즈니스 프로세스를 정의하기 위한 여러 가지 접근으로는 기업내 워크플로우를 정의하기 위한 WfMC의 WPDL, XPDL이나, 웹 서비스를 이용한 비즈니스 프로세스 정의를 위한 WSCI, BPML, WSFL, XLANG, BPEL4WS 등이 있다. 또한, ebXML이나 RossetaNet과 같은 전자상거래 표준들도 나름대로 BPSS, PIP과 같은 비즈니스 프로세스 프로토콜을 제안하고 있다[4].

비즈니스 프로세스를 효과적으로 작성하기 위해서는 빌드타임, 런타임, 분석타임의 세 가지 시점에서 고려해야 한다. 빌드타임에서는 기존의 프로세스들을 조합하거나 변형하여 재사용함으로써 손쉽게 실행 가능한 프로세스를 설계할 수 있어야 한다. 런타임에서는 설계된 여러 프로세스를 효과적으로 여러 조직의 작업자에게 할당되고, 체계적으로 수행 과정을 모니터링할 수 있어야 한다. 특히, 기업간 환경에서는 프로세스의 결합과 분리를 명백히 정의할 수 있어야 한다. 마지막으로 분석타임에서는, 수행이 완료된 프로세스 결과를 체계적으로 분석하고 개선할 수 있는 기반을 제공할 수 있어야 한다.

비즈니스 프로세스는 규모나 목표 범위에

+ 본 연구는 한국과학재단 목적기초연구(400-20020112) 지원으로 수행되었음.

따라서 세부적인 액티비티를 표현하는 하위 프로세스를 가지고 있거나, 외부의 또 다른 프로세스와 여러 가지 관계를 맺고 있을 수 있다. 본 논문에서는 워크플로우 상속을 위해 프로세스를 확장하기에 앞서, 위와 같은 상황에서 나타날 수 있는 프로세스 상호운용에 관한 기본적인 패턴을 먼저 파악할 것이다. 그리고 이 패턴들을 사용하여 워크플로우를 상속하여 협업 프로세스를 수행하는 방법을 제시할 것이다.

3. 프로세스 상호운용성 패턴

이 장에서는 비즈니스 프로세스 간에 발생할 수 있는 여러 가지 형태를 분석하여 여섯 가지 기본적인 상호운용성 패턴을 명시하였다. 이 기본 패턴은 WfMC이 제시한 세 가지 상호운용성 모델[5]을 프로세스 관리 측면에서 세분화한 것이다.

먼저, 연결 모델(chained model)에서는 하나의 프로세스가 다른 프로세스의 시작이나 실행을 발생시킨 후에, 그 프로세스와 별개로 실행된다. 이 모델을 다음 두 가지 패턴으로 구분하였다.

- ▷ 연결 대체(CS; Chained Substitutive) - 새로운 프로세스를 시작시킨 후에 종료되는 프로세스. 새로 시작된 프로세스가 요청 프로세스의 진행을 대신한다.
- ▷ 연결 추가(CA; Chained Additive) - 새로운 프로세스를 시작시킨 후에도 계속 수행되는 프로세스. 단, 그 이후에 두 프로세스는 다시 상호작용하지 않는다.

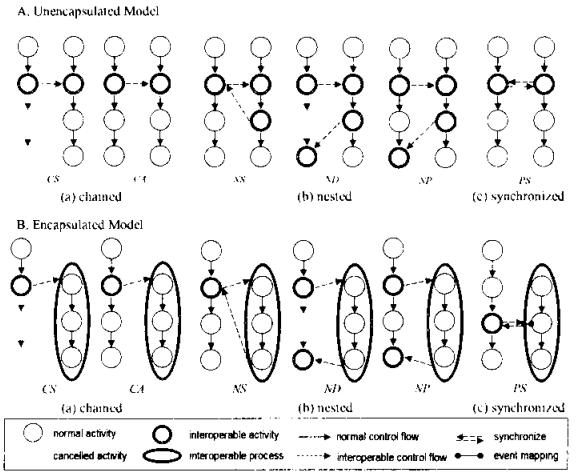
두 번째로, 중첩 모델(nested model)에서는 하나의 프로세스가 다른 프로세스를 호출한 후에, 특정 지점에서 실행 결과를 반환받는다. 이 모델을 다음과 같은 세 가지 패턴으로 구분하였다.

- ▷ 중첩 동기(NS; Nested Synchronous) - 새로운 프로세스를 시작시킨 지점에서 기다린 후에 수행 결과를 다시 반환받는 프로세스. 새로운 프로세스는 호출한 프로세스의 특정 액티비티의 하위 프로세스 역할을 한다.
- ▷ 중첩 대기(ND; Nested Deferred) - 새로운 프로세스를 시작시킨 후에 대기하고 있다가 다른 지점에서 수행 결과를 반환받는 프로세스. 새로운 프로세스는 호출한 프로세스에서 두 지점 사이의 액티비티들을 재정의하는 역할을 한다.
- ▷ 중첩 병렬(NP; Nested Parallel) - 호출한 프로세스의 두 지점 사이의 프로세스가 계속 실행된다는 점을 제외하고는 ND와 동일하다.

마지막으로 동기화 모델(synchronized model)은 다음과 같다.

- ▷ 병렬 동기(PS; Parallel Synchronized) - 두 프로세스가 특정 지점에서 동기화된다. 두 프로세스가 모두 특정 지점에 도착해야만 계속 실행될 수 있다.

[그림 1]은 두 프로세스 사이에서 나타날 수 있는 프로세스 상호운용성 패턴을 보여주고 있다.



[그림 1] 기본적인 프로세스 상호운용성 패턴

본 논문에서는 제시된 프로세스 상호운용성 패턴과 함께 워크플로우의 실행시간캡슐화(run-time encapsulation)를 고려하고 있다[2,3]. 실행시간캡슐화는 외부 시스템에게 프로세스의 세부 정보를 은닉하는 정책이다. 다시 말해서 한 프로세스가 캡슐화되지 않았다면, 그 프로세스의 액티비티 정보를 획득하고 직접 호출할 수 있다. 그러나 캡슐화된 프로세스를 호출하기 위해서는 정해진 이벤트를 발생시키는 메시지를 통하여 상호운용할 수 있다.

[그림 1]의 상단부 A는 캡슐화되지 않은 상태에서의 상호운용 패턴을, 하단부 B는 그렇지 않은 경우의 패턴을 보여주고 있다.

4. 워크플로우 상속

이 장에서는 비즈니스 프로세스를 보다 동적이고 확장적으로 활용하기 위하여 프로세스 상속이라는 개념을 제안한다. 워크플로우 상속이란 빌드타임이나 런타임에 외부 프로세스를 통하여 대상 프로세스의 일부를 상세화(specialization)할 수 있도록 허용함으로써, 프로세스 정의를 확장하는 것이다. 이 때 상세화하는 방법은 2장에서 언급한 프로세스 상호운용성 모델을 사용하여 외부 프로세스와 통신하게 된다.

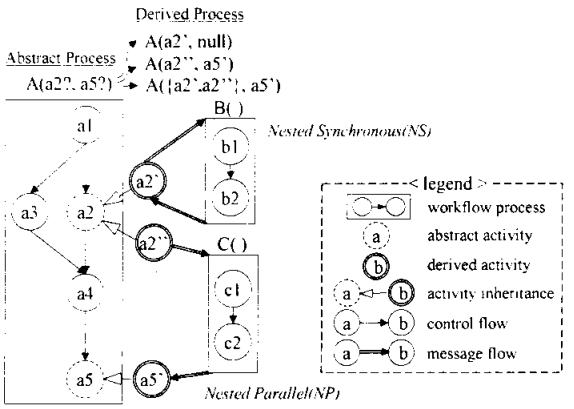
프로세스 상속은 상속 시기(빌드타임과 런타임)에 따라 프로세스 정의 상속와 프로세스 인스턴스 상속로 구분된다.

- ▷ **프로세스 정의 상속(IPD)**

비즈니스 프로세스 설계자는 빌드타임에

프로세스 정의 중에서 다양한 형태로 확장되거나 아직 불확실한 영역을 추상 액티비티(abstract activity)로 지정함으로써 추상 프로세스 정의(APD; Abstract Process Definition)를 생성할 수 있다. 프로세스 상호운용 패턴을 사용하여 그 미완성된 추상 액티비티를 상세화하는 것을 프로세스 정의 상속(IPD; Inheritance of Process Definition)라고 부른다.

프로세스 정의 상속은 미완성된 프로세스 정의의 추상성을 보장하고, 이를 상세화할 수 있도록 하여 다형성(polymorphism)을 부여하고 재사용성(reusbility)을 높이기 위한 것이다. 하나의 추상 프로세스 정의의 일부를 외부 기업에 맞게 확장할 수 있도록 개방하고, 변형된 프로세스를 일관되게 처리하고 분석할 수 있다.



[그림 2] 프로세스 정의 상속의 예

[그림 2]는 프로세스 정의 상속을 한 예를 보여준다. 추상 프로세스 정의 A(a2?,a5?)는 추상 액티비티의 상속을 통하여 세 가지 상속된 프로세스 정의로 확장될 수 있다. 세 가지 상속 프로세스는 각각 NS 상속, NP 상속, NS와 NP 상속으로 확장된 예이다.

이러한 프로세스 정의 상속을 XML로 표현하기 위해서는 다음과 같은 구성요소가 필요하다.

```
<WorkflowInheritance>
  <AbstractProcessDefinition Package=...>
    <AbstractActivities> ... </AbstractActivities>
    <DataFields> ... </DataFields>
  </AbstractProcessDefinition>
  <ActivityInheritances> ... </ActivityInheritances>
  <Interoperations> ... </Interoperations>
</WorkflowInheritance>
```

최상위 엘리먼트 <WorkflowInheritance>는 세 개의 하위 엘리먼트를 가진다. 첫 번째 하위 엘리먼트인 <AbstractProcessDefinition>는 확장 대상이 되는 추상 프로세스 정의에 대한 정보로서, 추상 액티비티와 관련 데이터 정보를 지닌다. 두 번째 하위 엘리먼트 <Activity Inheritances>는 추상 액티비티의 확장 정보를 나타내며, 마지막 엘리먼트인 <Interoperations>는 프로세스 상호운용성 패턴을 사용하여 어떤 거래자에게 어떤 데이터를 주고받는 지 정의하게 된다.

▷ **프로세스 인스턴스 상속(IPI)**

한편, 일반적인 프로세스 정의나 상속된 프로세스 정의에 대응되는 특정 인스턴스가 실행하기 직전에, 또는 진행 중에 아직 진행되지 않은 영역을 임의적으로(ad hoc) 확장하거나 상세화하는 것을 프로세스 인스턴스 상속(IPI; Inheritance of Process instance)이라고 정의하였다. IPI도 IPD와 마찬가지로 프로세스 상호운용 패턴을 사용하여 프로세스를 확장한다.

IPD는 상속 받은 프로세스 정의를 유지관리하기 때문에 앞으로도 동일한 프로세스 정의에 변경 내용이 계속 적용되는 반면에, IPI는 특정 프로세스 인스턴스에 한하여 상속이 적용되기 때문에 일회적이다.

프로세스 인스턴스 상속은 완성된 프로세스 정의나 추상 프로세스 정의에 대해 모두 적용할 수 있다. 프로세스 정의 상속 대신에 프로세스 인스턴스 상속이 요구되는 경우는 다음과 같다.

- 워크플로우로 실행하기 직전에 일회적인 추가 활동이나 수정이 필요한 경우
- 워크플로우가 실행하는 도중에 동적으로 임의적인 변경이 필요한 경우
- 외부의 특정 워크플로우 인스턴스와의 관계를 지정해야 하는 경우

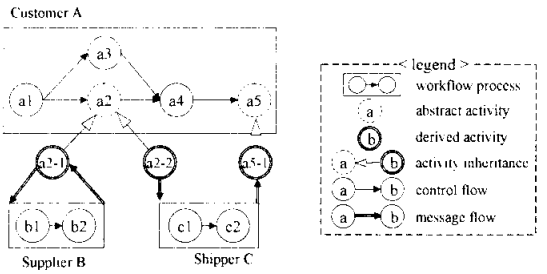
프로세스 인스턴스 상속을 위한 XML 스키마는 아래와 같다. 프로세스 정의 상속과 유사하지만, 프로세스 정의 대신에 워크플로우 벤더-특화된(vendor-specific) 프로세스 인스턴스 키를 사용하는 것이 다르다.

```
<WorkflowInstanceInheritance>
  <ProcessInstance InstanceKey="...">
    <ProcessDefinition Package="..."/>
    <TargetActivities> ... </TargetActivities>
    <DataFields> ... </DataFields>
  </ProcessInstance>
  <ActivityInheritances> ... </ActivityInheritances>
  <Interoperations> ... </Interoperations>
</WorkflowInstanceInheritance>
```

5. 기업간 프로세스의 확장적 설계

본 논문에서는 워크플로우 상속을 이용하여 기업간 협업을 수행하는 방안을 제시한다. 이를 위해서는 비즈니스 협업에 참여하는 기업들은 Wf-XML 표준을 통해 상호운용할 수 있는 워크플로우 시스템이 웹 서비스로 제공되어야 한다. 이러한 웹 서비스 정보는 WSDL (Web Sevice Description Language)로 작성되어 사설 UDDI 레지스트리(private UDDI registry)나 ebXML 레지스트리로 검색될 수도 있을 것이다.

공급 사슬에 참여하는 기업간 협업을 예로 들어보자. 공급 업체는 사설 레지스트리를 통하여 검색된 고객 업체의 워크플로우 서비스를 통하여 추상 프로세스 정의를 제공받는다. 이와 상호운용할 수 있는 자신의 워크플로우를 작성한 다음, 고객 업체와 상호운용성 계약(interoperability contracts)를 맺는다. 필요에 따라서는 운송 업체도 동시에 참여하여 그 기업의 워크플로우와 상호운용하는 자신의 워크플로우를 작성할 수 있다. 이들은 서로 제공하는 워크플로우 서비스를 통하여 Wf-XML 메시지를 SOAP 프로토콜로 교환하면서, 워크플로우를 상호운용하거나 해당 정보를 교환할 수 있다. [그림 3]은 고객 업체 A의 워크플로우와 공급 업체 B, 운송 업체 C의 워크플로우가 각각 NS 상속, NP 상속을 통하여 상호운용하는 예제를 도시하고 있다.



[그림 3] 공급 사슬 프로세스의 예제

5.1 프로세스 정의 상속의 XML 스키마

4.1절에서 언급한 바와 같이 프로세스 정의 상속을 위해서는 크게 세 가지 구성요소(추상 프로세스 정의 참조, 상속 액티비티 선언, 상호운용성 구현)가 필요하다. [그림 3]의 고객 업체를 중심으로 각각에 대한 스키마를 제시한다.

▷ **추상 프로세스 정의 참조**

추상 프로세스 정의는 하나 이상의 추상 액티비티를 하나 이상 가지고 있는 미완성 프로세스이다. 아래는 [그림 3]의 추상 프로세스 A의 예제를 보여준다. WfMC의 XPDL로 정의된 원본 프로세스 정의를 참조하고 있다[6].

```
<AbstractProcessDefinition
   Package="http://ara.snu.ac.kr/pdl/wf1.xpdl"
   Id="A" Name="EOrder" AccessLevel="PUBLIC">
   <AbstractActivities>
      <AbstractActivity Id="a2" Name="sendPO">
      <AbstractActivity Id="a5" Name="receiveJudg"/>
   </AbstractActivities>
   <DataFields>
      <DataField Id="orderInfo" Type="orderType">
         ...
   </DataFields>
</AbstractProcessDefinition>
```

이후에 워크플로우를 상속할 때 두 개의 액티비티를 사용할 수 있으며, 그 때 사용되는 데이터필드 리스트를 보여준다. 이 <DataField>는 참조하고 있는 XPDL의 프로세스 내에서 사용되는 데이터들을 지정한다.

▷ **상속 액티비티 선언**

<ActivityInheritances>에서는 외부 프로세스와의 인터페이스가 될 상속 액티비티들을 지정한다. 하나의 액티비티는 여러 개로 상속할 수 있어서 각각 다른 상호운용 패턴으로 사용된다.

```
<ActivityInheritances>
   <ActivityInheritance
         AbstractActivity="a2" DerivedActivity="a2-1"/>
   <ActivityInheritance
         AbstractActivity="a2" DerivedActivity="a2-2"/>
   <ActivityInheritance
         AbstractActivity="a5" DerivedActivity="a5-1"/>
</ActivityInheritances>
```

▷ **상호운용성 구현**

본 논문에서 제시하는 스키마는 상호운용 모델을 구현하기 위하여 워크플로우 시스템간의 Wf-XML 바인딩을 가정한 SOAP 메시지를 사용한다[7]. Wf-XML 표준을 처리하는 웹서비스 구현에 관한 연구는 이미 진행된 바 있다[8].

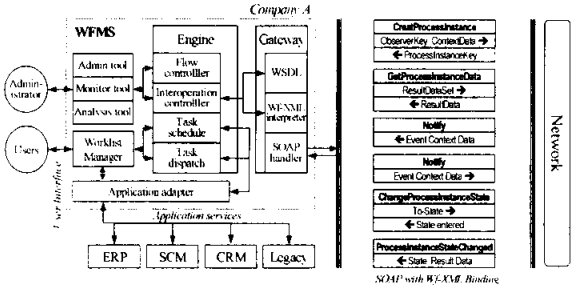
선언된 추상 액티비티들을 이용한 상호운용 패턴을 통하여 외부 프로세스와 인터페이스할 수 있다. 아래의 예는 위에서 지정한 상속 액티비티 *(a2-1)*을 사용하여 "중첩 동기화(NS)"를 구현하는 예이다.

```
<Interoperations>
   <Interoperation Pattern="NestedSynchronous">
      <InternalActivity DerivedActivity="a2-1"/>
      <ExternalProcess
            portType="cst:Wf-XMLPortType"
            operation="CreateProcessInstance">
         <Input Parameter="orderInfo"
               Element="cst:purchaseOrder"
               Type="cst:purchaseOrderType"/>
         <Output Parameter="judgementInfo"
               Element="cst:judgement"
               Type="cst:judgementType"/>
      </ExternalProcess>
   </Interoperation>
   ...
</Interoperations>
```

위의 예는 외부의 프로세스를 시작시키기 위하여 Wf-XML 표준 오퍼레이션 중에서 CreateProcessInstance 메시지를 전송하고 있다. SOAP 메시지에는 Wf-XML 표준에서 지정한 대로 CreateProcessInstance에 필요한 입력 데이터(ContextData, ObserverKey, Name?, Subject?, Description?)를 포함해야 한다(?는 optional을 의미한다). 그리고 이에 대응되는 응답 메시지에는 ProcessInstanceKey, Name?과 같은 출력 데이터가 포함된다. 이러한 정보는 쌍방간의 프로세스 상태 질의나 상태 변경을 위해 필수적인 데이터들이다.

5.2 상호운용 메시지의 웹 서비스 구현

5.1에서 기술된 워크플로우 상속에 대한 스키마를 해석하고, Wf-XML 표준을 웹 서비스를 통하여 구현하기 위한 시스템 아키텍처는 [그림 3]과 같다. Wf-XML 표준은 워크플로우 상호운용에 필요한 여섯 가지 오퍼레이션과 그에 해당하는 전달인자를 [그림 4]의 우측 부분과 같이 제공한다.



[그림 4] Wf-XML을 이용한 워크플로우 상호운용

워크플로우 시스템에서 비즈니스 협업을 구성하기 위한 상호운용 처리기(Interoperation controller)는 5.1에서 제시한 스키마를 파싱한 후, 워크플로우 상속 패턴에 따라서 [표 1]과 같은 Wf-XML 오퍼레이션 메시지를 생성하여 외부와 교환할 것이다.

[표 1] Wf-XML 메시지를 이용한 상호운용

Pattern		Wf-XML operations
Chained	*CS*	*CreateProcessInstance/Notify*
	CA	*CreateProcessInstance/Notify*
Nested	*NS*	*CreateProcessInstance/Notify →ChangeProcessInstanceState*
	ND	*CreateProcessInstance/Notify →ChangeProcessInstanceState*
	NP	*CreateProcessInstance/Notify →Notify*
Synchro-nized	*SP*	*Notify*

아래의 예는 거래 파트너의 워크플로우를 시작시키기 위한 SOAP 메시지의 예를 보여준다. Wf-XML 표준을 이용한 웹 서비스를 제공하기 위한 WSDL과 SOAP의 자세한 구성은 Rossi[8]의 연구를 참조하였다. 아래의 예는 Wf-XML을 구현한 SOAP 메시지를 보여준다.

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
   SOAP-ENV:encodingStyle=
      "http://www.wfmc.org/standards/docs/wf-xml"
   xmlns:SOAP-ENV=
      "http://schemas.xmlsoap.org/soap/envelope"
   xmlns:wf="http://www.wfmc.org/standards/docs/wf-xml">
   <SOAP-ENV:Header>
      <wf:Request wf:ResponseRequired="Yes"
            SOAP-ENV:mustUnderstand="1"/>
      <wf:Key SOAP-ENV:mustUnderstand="1">
            http://jumong.snu.ac.kr/purchase/027
      </wf:Key>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <wf:WfMessage
      xmlns="http://www.wfmc.org/standards/docs/wf-xml">
            <CreateProcessInstance.Request>
               <ObserverKey>
                  http://jumong.snu.ac.kr/customer/163
               </ObserverKey>
               <ContextData>
                  <PurchaseOrder/>
               </ContextData>
            </CreateProcessInstance.Request>
      </wf:WfMessage>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5. 결론

기업간 통합(B2Bi)에 있어서 비즈니스 프로세스 통합 및 자동화에 관한 연구는 매우 중요한 이슈이다. 본 연구는 비즈니스 프로세스 통합을 위한 여러 접근 방법 중에서 워크플로우 시스템을 강하게 결합한(tightly-coupled) 방법을 제안하였다. 워크플로우를 기업간 거래 환경에 적용하기 위하여 프로세스간의 상호운용성 패턴을 제시하고, 이를 이용하여 워크플로우 확장을 시도한다.

WfMC의 Wf-XML 표준은 워크플로우 상호운용성을 위한 기본적인 기능들을 제공한다. 하지만, Wf-XML은 상호운용을 위한 오퍼레이션과 전달인자를 제공하여 정적인 메시지 교환을 지원할 뿐이다. 본 연구의 워크플로우 상속은 프로세스간의 지속적인 상태 정보를 유지하며 메시지 교환 모델과 스키마를 구체화하고 있다. 이와 같은 워크플로우의 구성과 재사용에 관한 연구는 기업간 협업을 위한 비즈니스 프로세스를 효과적으로 유지하고 관리하는 데 이바지할 것이다.

참 고 문 헌

[1] Aalst, v., "Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce," *Electronic Commerce Research*, vol.2, no.3, 2002, pp.195-231.

[2] Y. Kim et al., "WW-Flow: Web-Based Workflow Management with Runtime Encapsulation," *IEEE Internet Computing*, vol. 4, no. 3, May/June 2000, pp.55-64.

[3] 정재윤, 김동수, 김영호, 강석호, "실행시간 캡슐화를 통한 워크플로우관리시스템의

구축", 한국경영과학회 추계학술대회, 2000, pp.215-218.
[4] 한국전산원, 기업간 워크플로우 통합 기술 표준 연구, 한국전산원 연구보고서, 2002.
[5] *WFMC-TC-1012, Workflow Standard-Interoperability Abstract Specification,* Workflow Management Coalition, Winchester, UK, 1999.
[6] *WFMC-TC-1025, Workflow Process Definition Interface - XML Process Definition Language,* Workflow Management Coalition, Lighthouse Point, Fla., 2002.
[7] *WFMC-TC-1023, Workflow Standard-Interoperability Wf-XML Binding,* Workflow Management Coalition, Lighthouse Point, Fla., 2001.
[8] Rossi, M., "Process Management: A Fundamental Component of Successful Web Service Execution". *Workflow Handbook 2002*, Future Strategies Inc. 2002, pp.95-116.