복합 이벤트를 이용한 패턴 기반 RFID 물류 프로세스 트리거링

Pattern-based RFID Logistic Process Triggering Using Complex Event

유영웅(Yeong-Woong Yu)*, 배혜림(Hyerim Bae)*
Sajal K. Das**, 구훈영(Hoonyoung Koo)***

초 록

물류 프로세스는 일반적인 비즈니스 프로세스와는 달리, 서로 이질적인 조직이 참여하여 프로세스를 진행한다는 특징을 보인다. 따라서 각 참여조직들 간의 상호작용을 제어하면서 프로세스를 수행하는 것이 필요하다. 프로세스의 실행이라는 측면에서 이를 제어하는 가장 좋은 대안으로 비즈니스 프로세스 관리(BPM) 시스템을 들 수 있으나, 참여조직 간의 프로세스 소유권과 접근 권한 관리 문제와 함께, 현재의 일반적인 상용 시스템들은 참여조직 간의 상호작용을 관리하기 위한 기능을 거의 제공하지 못한다는 문제점을 안고 있다. 또한 RFID 기술 기반의 물류 환경에서, RFID 이벤트의 적용과 처리를 지원하는 물류프로세스 관리에 대한 고려가 미흡한 실정이다. 따라서 본 연구에서는 물류 환경에서 발생하는 다조직 프로세스를 RFID 이벤트 기반의 통제를 위한 새로운 방법론을 제시하고자 한다. Inter-workflow(워크플로우 간) 패턴으로 표현되는 참여조직들 간의 프로세스 자동 트리거링(Auto-Triggering)을 위하여 ECA(Event-Condition-Action) 규칙에 복합이벤트(Complex Event)를 활용한 RFID 이벤트 기반의 RFID-based ECA 규칙을 제공한다.

ABSTRACT

In logistic environments, a process, in that it manages the flow of materials among partners, involves more than one organization. In this regard, a logistic process, as a combined process consisting of multiple sub processes, needs to be managed with controling interaction among partners. In achieving systematic management of a logistic process, traditional Business Process Management (BPM) cannot be used for the entire flow, since it lacks the ability to manage interactions among partners. Particularly in logistic environments where RFID technologies are used, how to deal with the connection between RFID event and logistic flow has not been properly addressed. To overcome this limitation, this paper proposes a new method of managing multi-organizational logistic processes based on RFID events. We define inter-workflow pattern, and suggest ECA(Event-Condition-Action) rules for auto triggering of logistic processes. To adjust the rules to RFID events, we invent RFID-based ECA rules using complex event. A prototype system has been developed for the purpose of demonstrating the effectiveness of our approach.

이 논문은 2007년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음 (KRF-20070528000).

* 부산대학교 산업공학과
** Dept. of Computer Science and Engineering, University of Texas at Arlington
*** 한국전자통신연구원(ETRI)

2009년 10월 21일 접수, 2009년 11월 02일 심사완료 후 2009년 11월 11일 게재확정.

키워드 : Inter-Workflow 패턴, RFID 이벤트, 복합이벤트, 물류, 자동트리거링, 비즈니스 프로세스관리
Inter-Workflow pattern, RFID Event, Complex Event, Logistic, Auto-triggering, BPM

1. 서 론

유비쿼터스 물류 환경이 도래함에 따라 급격하게 변화하는 물류 환경에서 현재의 바코드 시스템 등이 갖고 있는 구조적, 기능적 한계를 극복할 수 있는 새로운 기술이 필요하게 되었고, 그 해결 방안으로 RFID 기술에 초점이 맞추어 지고 있다. 물류 혁명을 이끌 대안으로 떠오르고 있는 RFID 기술의 도입으로 인해 기존의 물류 프로세스의 구조 및 관리 방법 역시 물류 환경의 변화를 수용 가능한 형태로의 변화가 요구되고 있다.

이런 물류 환경에서 기업의 경쟁력은 고객에게 제품이나 서비스가 전달되는 전 과정에 걸쳐서 결정되기 때문에 기업 간의 파트너쉽과 협력이 기업의 성공에 무엇보다도 중요한 요소로 자리 잡고 있다[11, 15]. 이러한 협력은 특히, 근본적으로 기업 간의 협력을 요구할 수 밖에 없는 물류 환경에서 더욱 중요하다. 또한, 협력에 의한 성공은 고객만족을 증대시키려는 비즈니스 파트너들 간의 유기적인 인터페이스에 의해 달성될 수 있다[7, 11, 15].

효율적인 공급사슬 관리를 위해 그 동안 공급사슬망관리(Supply Chain Management, SCM)가 소개되고 활용되어 파트너들 간의 협력을 계획, 구현, 통제해왔다[7]. 그러나 조직간 프로세스의 실행과 같은 실행적인 이슈들에 대해서는 비교적 연구가 활발하지 못했다. 이러한 프로세스 실행 이슈는 물류 프로세스를 관리하기 위한 체계적이고 시스템적인 방법론에 의해서 달성될 수 있다. BPM은 비즈니스 프로세스 관리와 실행에 효과적이고 통합적인 방법으로 널리 받아들여져 왔고[4, 17], BPM 시스템은 비즈니스 프로세스의 시스템적인 디자인, 관리, 통합 그리고 향상을 통해 기업의 생산성 증대를 위한 일반적인 방법론으로 고려되고 있다[4, 14, 15]. 그럼에도 불구하고, 공급사슬에서 발생하는 프로세스를 원활히 진행하고 통합관리하기 위해서는 기업 간 프로세스 공유 등 높은 수준의 공급사슬 통합 요구와[1], 효율적인 정보공유를 통한 기업 간의 최적화를 추구하는데 반해, BPM의 기본적인 기능에서는 이런 환경에서의 여러 이질적인 주체들 간 프로세스의 효율적인 관리에 대한 지원이 미흡한 실정이다. 이런 장애물을 극복하기 위해, 본 논문에서는 독립적인 주체 사이의 프로세스 간의 상호작용을 다루기 위해 패턴 기반 접근법을 사용하였다. 또한 RFID 기술 도입 및 적용이 활발한 현재 물류 환경에서 RFID 이벤트를 활용한 RFID 기반 물류프로세스에 대한 고려가 미흡한 실정이기 때문에, 물류 환경에서 inter-workflow 패턴을 통해 정의된 여러 참여 기업들 간의 프로세스를 자동

으로 트리거링하기 위하여 기존의 비즈니스 프로세스 이벤트와 RFID 이벤트를 결합한 복합 이벤트를 적용하여 기존의 ECA 규칙을 RFID 이벤트 기반의 ECA 규칙으로 확장한다. 이를 통하여 프로세스 간의 실행제어가 규칙 엔진이라는 참여조직들에 독립적이고 삼자적인 특성을 지니는 주체에 의해 제어 가능하도록 한다.

특히 본 연구는 RFID 환경을 고려함으로써, 다른 파트너의 프로세스를 트리거링하는 이벤트를 RFID 장비로 한정할 수 있기 때문에 프로세스 참여주체들 사이에서 발생할 수 있는 접근권한에 대한 문제도 일부 해결 가능하고, 이러한 패턴들은 분리된 프로세스 엔진 없이도 다른 프로세스의 액션을 트리거링하는 것에 의해 물류 프로세스의 수행을 가능하게 하는 ECA 규칙으로 변환된다.

본 논문의 제 2장에서 관련연구, 기본적인 프로세스 모델과 일반적인 ECA 규칙을 간략하게 알아보고, 제 3장에서 주요 inter-workflow 패턴과 그 중에서도 본 연구에서 중점적으로 다룰 연계 서비스 모델(Chained Service Model, CSM)에 대해 보다 상세하게 다루고, 실제 물류 환경에서 이 패턴이 발생하는 물류프로세스 예제를 소개한다. 제 4장에서는 BPM 시스템에서 관리되는 주요 개체인 프로세스와 태스크의 상태 전이 모델과 ECA 규칙, 그리고 RFID 이벤트 기반의 복합 이벤트에 대해 상세하게 다루고 제 5장에서는 복합 이벤트를 사용하여 inter-workflow 패턴을 처리하기 위한 RFID 기반 ECA 규칙에 대해 소개한다. 제 6장에서 결론과 향후 연구 방향을 제시하였다.

2. 연구의 배경

2.1 관련 연구

이전의 연구들은 대부분 워크플로우에 관한 연구 또는 규칙 기반 워크플로우 실행에 관한 연구들이다. 대부분의 연구들은 사전에 정의된 패턴들을 통해 프로세스를 모델링하기 위해 진행되어 왔다. 국제 워크플로우 표준 기구인 WfMC(Workflow Management Coalition)는 프로세스 분기와 병합에 대해, ‘AND’, ‘OR’, ‘LOOP’를 포함하는 여러 타입의 워크플로우 모델링 시맨틱을 정의하고 있다[17]. [16]은 20개의 향상된 워크플로우 패턴들을 확장하였고 워크플로우 및 BPM 연구자 그리고 대부분의 BPM 시스템에서 이러한 표준 패턴들을 채택하였고, 그리하여 워크플로우의 상호운용능력을 위한 기초가 만들어졌다.

규칙 기반 접근법을 비즈니스 프로세스 수행에 적용하기 위한 여러 연구들이 착수되었다[3, 5, 6, 10, 12, 13]. [3]은 ECA 규칙을 가능하게 하는 동적 데이터베이스로 워크플로우 엔진을 교체하는 것에 의해 자동화 된 비즈니스 프로세스 실행을 제안하였다. [12]는 차량 배치 프로세스의 모델링과 자동 실행에 대한 방법을 소개하였고, 이 방법에는 동적 프로세스의 편리한 모델링을 제어하는 규칙이 이용되었다. [13]역시 워크플로우의 실행에 규칙을 적용하였으나, 이전의 연구들과 달리 객체 다이어그램과 활동 다이어그램을 사용하였다. [6]은 ECA 규칙의 집합과 서비스 구성을 위해 요구된 실행을 위한 방법을 고안하였

고, [10]은 비즈니스 프로세스와 비즈니스 규칙을 분리하여 정의하고, 프로세스 단위(unit)를 사용한 프로세스 상태 기술과 프로세스 흐름으로부터 비즈니스 규칙을 분리하는 접근법을 통해 프로세스 유연성 강화와 이해용이성을 제공하는 등 서비스 지향적 비즈니스 프로세스에 대한 포괄적인 패러다임을 제시하였다.

RFID 환경을 고려한 관련 연구로는 정보시스템에서 존재하는 정보의 갭을 줄이기 위하여 다수의 RFID 이벤트를 복합이벤트로 정의하고 간단한 규칙을 적용한 복합이벤트 처리에 관한 연구[19]와, [18]의 RFID 데이터 관리를 위해 ECA 규칙을 사용한 연구가 있었다.

이전의 연구들이 총괄적으로 본 연구의 기초를 제공하기는 하지만, 본 연구의 방법은 물류 환경에서 적합한 RFID 이벤트 처리와 다조직 간 워크플로우 처리를 동시에 고려하여 이전의 연구와는 명백하게 다른 방법이라 할 수 있다. 그것은 이전 연구들은 단일 프로세스에 대한 프로세스 패턴과 ECA 규칙을 개발하는 것에 반해, 본 연구는 여러 기업 간의 복합 프로세스 사이의 관계와 인터페이스에 초점을 맞추고 있기 때문이다. 따라서 이미 비즈니스 프로세스 관리 시스템이 갖춰진 물류 환경에서 복합적인 조직들 사이의 프로세스 실행 그 자체 보다는 inter-workflow 간의 자동 트리거링을 수행하기 위하여 복합이벤트를 사용하는 RFID 기반의 ECA 규칙을 소개한다.

2.2 프로세스 모델

프로세스들 사이의 관계를 기술하기에 앞서, 기본 프로세스 모델 개념을 다룰 필요가 있다. 본 논문의 접근법에서 물류 프로세스에 참여하는 모든 파트너 기업들은 자신의 프로세스 관리 시스템이 있다고 가정한다. 이러한 가정을 토대로, 참여하는 기업의 프로세스를 표현하기 위한 프로세스 모델은 아래와 같이 간단하게 정의될 수 있다[2].

정의 1 : 프로세스 모델(Process Model)
프로세스 모델은 아래와 같이, 유방향 그래프 P = (T, L)와 분기 및 병합을 위한 라벨링 함수 $f(o)$로 정의된다.

- 단위업무 집합 T = $\{t_i \mid i = 1, \cdots, N\}$. 단, t_i는 i번째 단위업무이고 N은 단위업무의 수이다.
- 링크 집합 L = $\{(t_i, t_j) \mid t_i, t_j \in T$, 그리고 (t_i, t_j)는 단위업무 t_i는 단위업무 t_j보다 바로 앞서 실행되어야 함을 의미한다.
- 분기 단위업무 t_i(S = $\{t_k \mid (t_i, t_k) \in L\}$, $\mid S \mid > 1$)에 대하여, 만일 모든 t_k가 수행되어야 한다면 $f(t_i)$='AND'이고 그렇지 않으면 'OR'이다.
- 병합 단위업무 t_j(M = $\{t_k \mid (t_k, t_j) \in L\}$, $\mid M \mid > 1$)에 대하여, 만일 모든 t_k가 수행되어야 한다면 $f(t_j)$='AND'이고 그렇지 않으면 'OR'이다.

2.3 ECA(Event-Condition-Action) 규칙

기본적인 프로세스의 실행을 위하여, 대부분의 BPM 시스템이 제공하는 프로세스 실행 엔진이 사용될 수 있고 글로벌 물류 프로세스 역시 참여업체들 사이의 고른 상호운용을

위한 실행 메커니즘이 필요하다. 이 논문은 전체적인 프로세스의 실행보다는 inter-workflow 간의 트리거링에 초점을 맞춰 RFID 기반 ECA 규칙을 사용한다. 기존의 ECA 규칙을 사용하는 inter-workflow 간의 제어에 관한 이전의 연구에서는 RFID 이벤트 적용과 복합 이벤트에 대한 고려가 부족한 상황이며 일반적인 ECA 규칙은 다음과 같다.

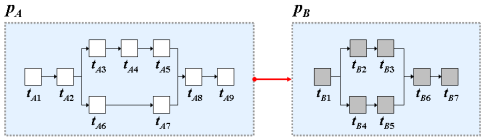
```
Rule (rule_name)
  ON (object).(event)
  IF conditionSet = {(condition_expression)}
  Do actionSet = {(object).(action)}
EndRule
```

3. Inter-Workflow pattern

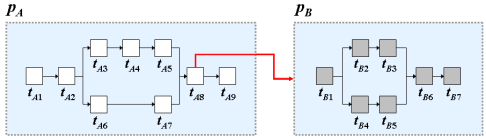
본 연구에서는 RFID 기반 물류 프로세스 간의 자동 트리거링을 위한 RFID 기반 ECA 규칙을 설명하기 위해 inter-workflow 패턴을 소개한다. 물류 프로세스 모델링에서 다양한 연구를 통해 발견된 여러 inter-workflow 패턴들을 중심으로 본 연구에서는 RFID 물류 환경에 초점을 맞춰 빈번하게 발생하는 패턴들을 활용한 물류 프로세스 예제를 소개한다.

3.1 패턴 1 : 연계 서비스 모델

연계 서비스 모델(CSM)은 WfMC의 표준 스펙에 의해 소개된 간단한 사례이다[17]. CSM에서, 앞선 프로세스의 그 실행이 완료되면, 뒤따르는 프로세스가 트리거링되고 실행이 시작된다. CSM은 트리거링을 발생시키는 객체에 따라 두 개의 서브 패턴으로 분류할 수 있다. 만약 이후에 트리거링되는 프로세스가 이전의 프로세스에 의해 트리거링되면, 그 패턴은 CSM-I이라 부른다. 반면, 뒤따르는 프로세스가 그 이전의 프로세스 내에 존재하는 하나의 태스크에 의해 트리거링되면, CSM-II라 부른다. <그림 1>(a)는 P_A가 완료된 이후에 P_B가 P_A에 의해 트리거링되며, 초기화되는 것을 나타내고 <그림 1>(b)에서 P_B는 t_{A8}에 의해 트리거링 된다.



(a) Chained Service Model triggered by process (CSM-I)



(b) Chained Service Model triggered by task (CSM-Ⅱ)

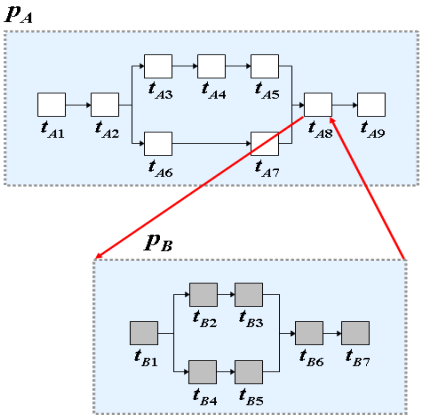
〈그림 1〉 연계 서비스 모델

만약, CSM-II에서 프로세스를 구성하는 마지막 태스크인 t_{A9}가 P_B를 트리거링하는 경우에는 패턴의 구조가 CSM-I과 약간 유사해 보일 수 있다. 그러나 이 경우에 뒤에 나오는 <그림 8>의 프로세스와 태스크의 상태 전이 모델에서 명확한 차이점을 찾을 수 있다. CSM-I의 프로세스 P_A가 시작될 때, P_A의 상태는 'READY'에서 'DOING'으로 바뀌게 되고, 이 때 첫 번째 태스크의 실행이 시작될 수 있다. 모든 태스크들은 프로세스의 상태가 'DOING'인 상황 하에서 실행을 시작하고 완료

해야 한다. 그리고 마지막 태스크는 프로세스 P_A의 상태가 'DOING' 상태일 때 종료되어야 하며, 그 이후에 프로세스의 상태는 'DOING'에서 'FINISHED' 상태로 바뀌게 된다.

3.2 패턴 2 : 중첩 서브프로세스 모델

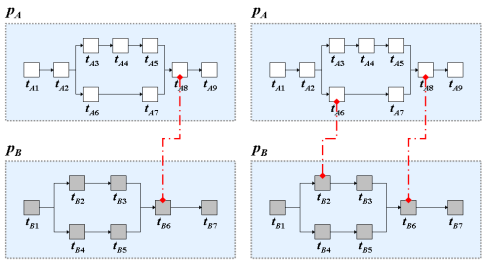
<그림 2>처럼 중첩 서브프로세스 모델(Nested Sub-process Model, NSpM)에서, 하나의 프로세스는 부모-자식 관계를 형성하는 다른 프로세스의 단위 태스크가 된다. 부모 프로세스에 해당하는 P_A를 구성하는 태스크의 하나인 t_{A8}은 자식 프로세스에 해당하는 P_B를 포함하고 있다. P_A의 관점에서, P_B의 실행은 t_{A8}의 실행과 유사하게 다루어질 수 있다.



〈그림 2〉 중첩 서브프로세스 모델

3.3 패턴 3 : 병렬 동기화 모델

병렬 동기화 모델(Parallel Synchronization Model, PSM)은 두 개의 다른 프로세스들이 병렬로 실행되고, 특정 지점에서 동기화되는 관계 패턴을 포함한다. 다시 말하면, 하나의 프로세스는 다른 프로세스가 사전 정의된 동기화 지점에 도달할 때까지 정확한 지점에서 기다려야 한다.



(a) PSM-I (b) PSM-II

〈그림 3〉 병렬 동기화 모델

PSM은 동기화 지점의 개수에 따라 한 지점에서 동기화되는 모델인 One-Point 동기화 모델(PSM-I)과 여러 지점을 가지는 Multi-Point 동기화 모델(PSM-II)로 분류될 수 있고, <그림 3>(a)와 (b)에 각각 표현되어 있다.

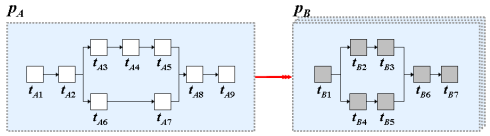
PSM-I <그림 3>(a)에서, 만약 하나의 프로세스 P_A내에 하나의 태스크 t_{A8}이 P_B의 t_{B6}의 완료 없이 먼저 완료된다면, P_B가 t_{B6}에 도달할 때까지 P_A의 수행은 멈춰있어야 한다. PSM-II <그림 3)(b)의 작동은 동기화 지점이 여러 곳인 것을 제외하면 PSM-I과 유사하다.

3.4 패턴 4&5 : 다중 인스턴스 모델과 인스턴스 병합 모델

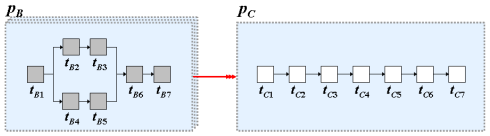
물류 프로세스에서, 제품은 트럭을 통해 제조업체에서 고객으로 간다. 전체 프로세스의 어떤 단계 동안, 때로는 박스에 함께 담기거나 때로는 따로 분리된다. 다중 인스턴스 모델(Multiple Instance Model, MIM)은 프로세

스를 다른 프로세스의 복합 인스턴스를 트리거하는 것을 가능하게 하기 때문에 이러한 물류 프로세스 환경에서 유용하게 활용될 수 있다.

가끔 인스턴스들의 정확한 수는 결정될 수 있고 결정되지 않을 수도 있다. 만약, 그 수를 결정할 수 있다면, 정적 인스턴스 모델(Static Instance Model), MIM-I이라 불리고, 만약 결정할 수 없다면, 동적 인스턴스 모델(Dynamic Instance Model), MIM-II라고 불린다.



(a) Multiple Instance Model (MIM)



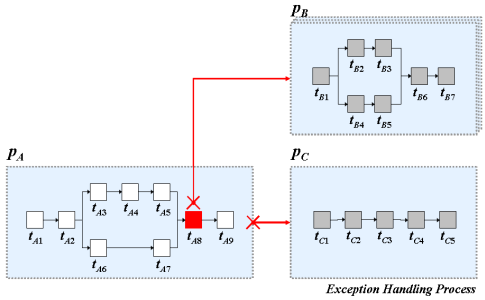
(b) Instance Merge Model (IMM)

〈그림 4〉 다중 인스턴스 모델과 인스턴스 병합 모델

MIM이 <그림 4>(a)에 나타나 있다. 프로세스 P_A가 끝나자마자, 다음에 뒤따르는 프로세스 P_B의 다중 인스턴스들은 자신들의 실행을 시작한다. MIM과 대조적으로, IMM에서 다중 프로세스 인스턴스들은 단일 인스턴스를 트리거링 할 수 있다. <그림 4>(b)에서, P_A의 다중 인스턴스들이 완료된 이후에, P_B의 단일 인스턴스의 실행이 시작된다.

3.5 패턴 6 : 예외 처리 모델

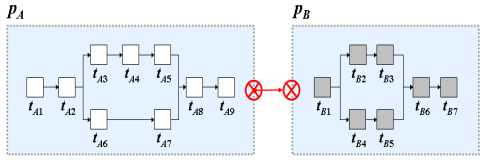
이미 많은 연구자들에 의해 예외처리 문제들이 다루어져 왔고[8, 9], 그들의 접근법은 워크플로우의 태스크-레벨 핸들링을 위해서 제공되었다. 그러나 물류 환경에서 예외처리는 분리된 프로세스에 의해 제공되어야 한다. 예외처리프로세스(Exception Handling Process)는 예외를 발생시키는 다른 프로세스를 위해 예외를 처리하는 역할을 해낸다. <그림 5>는 P_B가 P_A를 위한 예외 처리기를 나타낸다. P_A에서 t_{A8}은 예외를 발생시킬 잠재적인 가능성을 가진다. 만약 예상치 못한 문제가 태스크의 실행 동안에 발생하게 되면 프로세스의 제어는 P_C에게 넘어간다.



〈그림 5〉 예외 처리 모델

3.6 패턴 7 : 취소 전파 모델

취소 전파 모델(Cancellation Propagation Model, CPM)은 두 프로세스들 사이의 관계 패턴이다.

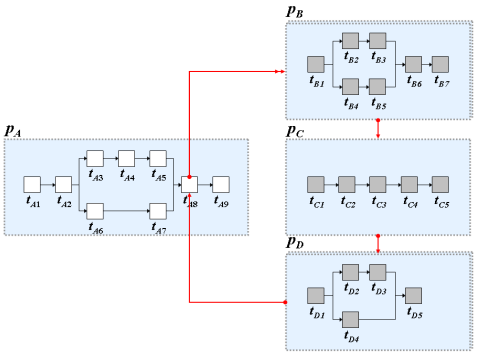


〈그림 6〉 취소 전파 모델

이것에 의해 프로세스 실패는 다른 프로세스에게로 전파된다. <그림 6>처럼, 만약 프로세스 P_A가 실패하면, 그 다음에 프로세스 P_B 역시 수행을 멈추게 된다.

3.7 복합 관계 모델

가끔 프로세스들 사이의 관계들은 다른 모델들과의 조합에 의해 설명될 필요가 있다. <그림 7>에서, 복합관계모델(Complex Relationship Model, CRM)은 네 개의 다른 프로세스들 사이에서 여러 관계들을 표현하는데 사용된다.



〈그림 7〉 복합 관계 모델

만약 P_D의 모든 인스턴스들이 종료되면, 마지막으로 P_A가 t_{A9}보다 앞서 먼저 실행된다.

4. 이벤트 정의

본 연구에서는 프로세스 간의 자동 트리거링을 위하여 크게 두 가지 이벤트 타입인 RFID 이벤트(RFID Event, RFE)와 비즈니스 프로세스 이벤트(Business Process Event, BPE)로 구분한다. 특히, RFID 기술 기반의 물류 환경에서의 inter-workflow 패턴을 고려한 RF-ECA 규칙을 생성하기 위해 RFE와 BPE의 여러 다양한 조합으로 발생하는 복합이벤트(Complex Event)를 활용한다.

4.1 RFID 이벤트

정의 2. ***RFID Event, RFE***

e_{RF} = <*tagID*, *readerID*, *timestamp*>

RFE는 RFID 태그 및 리더와 같은 RFID 장비들로부터 발생하는 이벤트 집합으로 정의 2에 따르면, RFID 이벤트 e_1(∈RFE) = <*tag_001*, *reader_001*, *2009.10.15.18 : 30 : 42*>는 2009년 10월 15일 18시 30분 42초에 리더 *reader_001*이 태그 *tag_001*의 데이터를 읽었다는 것을 의미한다.

4.2 비즈니스 프로세스 이벤트

정의 3. ***Business Process Event, BPE***

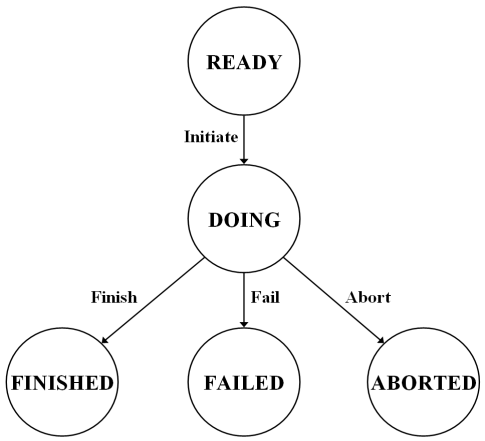
e_{BP} = <*objectID*, *eventType*, *timestamp*>

비즈니스 프로세스 이벤트(BPE)는 비즈니스 프로세스 실행과정에서 발생하는 이벤트들의 집합으로 나타내고, object 식별자, event type 그리고 time stamp 이 3가지 구성요소로 이루어진다. 본 연구에서, 객체는 프로세스와 태스크 두 가지 타입으로 분류된다. 다음에 나오는 상태 전이 모델에서 두 객체 타입 프로세스와 태스크에서 발생 가능한 모든 이벤트 타입을 설명한다.

4.2.1 객체 상태 전이 모델

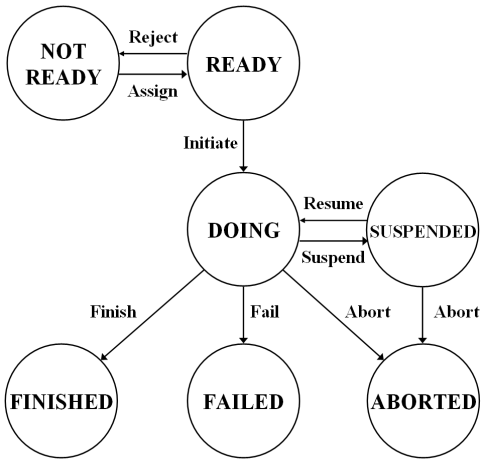
BPM 시스템이 관리하는 주요 개체인 프로세스와 태스크는 비즈니스 프로세스가 실행되는 동안 그 상태가 변하게 된다. <그림 8> (a)와 (b)는 각각 프로세스와 태스크의 상태의 전이를 나타내고 있다.

프로세스가 모델링되고 실행 단계에서, <그림 8>(a)와 같이 프로세스는 'READY' 상태를 가지게 되고, 그 프로세스에 대해 시작 권한을 가진 사용자는 'Initiate' 이벤트를 발생시키는 것으로 프로세스를 'READY'에서 'DOING' 상태로 바꾸는 실행을 할 수 있다. 프로세스의 실행이 완료되면, 'FINISHED' 상태가 되거나 오류 등의 결과나 사용자에 의해 'FAILED' 또는 'ABORTED' 상태가 될 수 있다.



〈그림 8〉(a) 프로세스 상태 전이 모델

태스크의 경우 프로세스의 상태 전이 모델보다 조금 더 복잡하다. 프로세스 내에 하나의 태스크를 고려해보자. <그림 8>(b)와 같이 해당 태스크를 앞서는 이전의 모든 태스크들이 완료하게 되면, 그 태스크가 사용자에게 할당되며, 그에 대응하는 'Assign' 이벤트가 발생되고 'NOT READY' 상태에서 'READY' 상태로 바뀌게 된다.



〈그림 8〉(b) 태스크 상태 전이 모델

만약 사용자가 태스크를 수락하지 않으면, 태스크의 상태는 이전 상태로 되돌아가게 되고, 사용자가 태스크를 수락한다면, 'DOING' 상태로 들어가게 된다. 실행 이후에, 태스크 상태 모델은 프로세스의 경우와 같이 세 가지 완료 상태 중 하나의 상태로 들어가게 된다. 비즈니스 프로세스 이벤트의 발생 주체가 되는 '프로세스'와 '태스크'에서 발생하는 주요 이벤트는 <표 1>과 같이 정리할 수 있다.

4.3 복합이벤트와 이벤트 연산자

본 연구에서는 RFID 이벤트와 비즈니스 프로세스 이벤트의 결합 관계를 나타내는 복합이벤트를 표현하기 위해, 네 가지의 이벤트 연산자를 이용하여 이벤트 간의 결합 관계를 나타내며 각 연산자의 의미는 아래와 같다.

〈표 1〉 프로세스 이벤트와 태스크 이벤트

Object	Event		
	Event Name	Corresponding Function	Description
Process (Instance)	Create	*createInstance()*	When a process instance is created.
	Initiate	*initiate()*	When a process begins.
	Finish	*finish()*	When a process completes.
	Fail	*fail()*	When a process fails.
	Abort	*abort()*	When a process is aborted by a user.
Task (Instance)	Assign	*assign()*	When a task is assigned to a user.
	Reject	*reject()*	When a user rejects a task
	Begin	*begin()*	When a task begins
	Suspend	*suspend()*	When execution of a task is suspended.
	Resume	*resume()*	When a suspended task resumes its execution.
	Finish	*finish()*	When a task completes normally.
	Fail	*fail()*	When a task fails.
	Abort	*abort()*	When a task is aborted by a user.

- **And(∧)** : $e_1 \wedge e_2$는 두 이벤트 e_1과 e_2의 논리곱의 관계를 나타낸다.
- **Or(∨)** : $e_1 \vee e_2$는 두 이벤트 e_1과 e_2의 논리합의 관계를 나타낸다.
- **Precedence(⇒)** : $e_1 \Rightarrow e_2$는 두 이벤트 e_1과 e_2가 모두 발생했다면, e_1이 e_2에 앞서 발생됨을 나타낸다.
- **Negation(¬)** : $\neg e_1$은 이벤트 e_1이 발생하지 않았음을 의미한다.

정의 4. *Complex Event, CPE*

복합 이벤트(CPE) $e_C = <e_t, start_time, end_time>$으로 정의될 수 있다. e_t는 이벤트 시멘틱, *start_time*은 첫 번째 이벤트의 발생 시각, *end_time*은 마지막 이벤트의 발생 시각을 나타낸다. 위에서 정의한 네 가지 연산자를 사용하여, e_C의 네 가지 인스턴스를 설명한다.

- AND(e_1, e_2, …, e_i, …, e_n) : 어떤 이벤트 $e_i \in$ (BPE∪RFE∪CPE)이고, n이 고려되는 이벤트의 개수일 때, $e_1 \wedge e_2 \wedge \cdots \wedge e_i \wedge \cdots \wedge e_n$을 나타내는 복합이벤트이다.
- OR(e_1, e_2, …, e_i, …, e_n) : 어떤 이벤트 $e_i \in$(BPE∪RFE∪CPE)이고, n이 고려되는 이벤트의 개수일 때, $e_1 \vee e_2 \vee \cdots \vee e_i \vee \cdots \vee e_n$을 의미하는 복합이벤트이다.
- SEQ(e_1, e_2, …, e_i, …, e_n) : 모든 이벤트에 대해서, $e_i \in$(BPE∪RFE∪CPE)이면, $e_i \Rightarrow e_{i+1}$은 순서를 나타내는 복합이벤트이다. RFID 이벤트와 비즈니스 프로세스 이벤트는 순간적으로 발생하지만 복합 이벤트는 시간의 범위를 포함하고 있기 때문에 time stamp의 표현이 서로 다르다. 따라서 복합이벤트는 time duration을 고려해야 한다. 복합이벤트 e_C의 *start_time*과 *end_time*은 아래와 같이 계산될 수 있다.
  - $e_C.start\_time = \min(e_i.timestamp)$
  - $e_C.end\_time = \max(e_i.timestamp)$

(i = 1, 2, ⋯, n)

- NOT(e_1, e_2, ⋯, e_i, ⋯, e_n) : 어떤 이벤트 $e_i \in$(BPE∪RFE∪CPE)이고, n이 고려되는 이벤트의 개수일 때, $\neg e_1 \wedge \neg e_2 \wedge \cdots \wedge \neg e_i \wedge \cdots \wedge \neg e_n$을 의미하는 복합이벤트이다.

본 연구에서 RF-ECA 규칙을 적용하여 inter-workflow 간의 자동 트리거링을 위해 정의 4의 복합이벤트를 활용한다. 기업 내부에서 실행되는 프로세스 관리를 위해서도 복합이벤트가 필요하지만, 여러 업체가 참여하고 동시에 여러 프로세스가 실행되는 환경에서는 RFID 이벤트와 비즈니스 프로세스 이벤트 둘 다를 활용한 복합 이벤트의 사용이 더욱 정확하고 신뢰성 있는 프로세스 실행을 가능하게 하고 본 연구의 초점이 되는 RFID 환경에서 다른 참여 업체 간의 관계인 inter-workflow 간의 정확한 트리거링이 가능하다.

5. Inter-Workflow 패턴 제어

RF-ECA 룰에 따라, 액션은 시스템에 의해 조건이 만족한 이후에 수행된다. 그러므로 조건은 컴퓨터 시스템이 해석할 수 있는 형태로 표현되어야 하고, 시스템은 사전에 정의된 개체들 사이에서 연산을 수행하고 평가한다. 조건이 평가된 이후에, RF-ECA 규칙에 정의된 해당 액션은 그 평가 결과를 토대로 호출된다. 본 연구에서 이 액션들은 inter-workflow 간에 매핑된 해당 프로세스에 대한 트리거링 기능으로 초점을 맞춘다.

5.1 패턴제어를 위한 RFID 기반 ECA 규칙

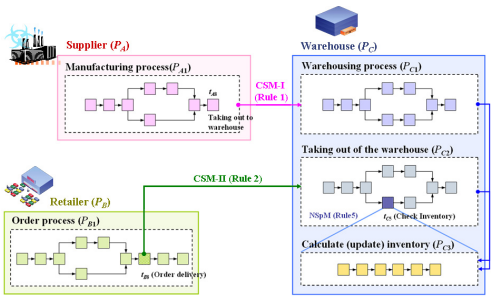
프로세스들 사이의 관계가 기술되고 나면, 그 관계들은 실행 단계에서 트리거링 제어를 위해 RF-ECA 규칙으로 변환될 수 있다. 본 연구의 접근법에서 RF-ECA 규칙은 '프로세스', '태스크'와 같은 기본 객체들에서 발생하는 비즈니스 프로세스 이벤트와 추가로 RFID 이벤트를 사용한다. 대부분의 상용 BPM 시스템들은 객체에 대한 이벤트와 기능들은 사전 정의하고 제공한다. RF-ECA 규칙 내에서 이벤트, 조건 그리고, 액션 부분을 상세하게 기술하기 위해 그 기능들을 사용하기로 하며, 복합 이벤트를 활용한 inter-workflow 간의 자동 트리거링을 위한 RF-ECA 규칙은 아래와 같이 정의될 수 있다.

RF-ECARule (rule_name)
ON an event e ($\in$\{$BPE \cup RFE \cup CPE$\})
IF conditionSet = \{c_i | c_i is a condition\}
Do actionSet = \{ac_i | ac_i is an action\}
EndRF-ECARule

RFID 이벤트와 비즈니스 프로세스 이벤트, 또는 이들의 결합을 표현하는 복합 이벤트가 발생하면 conditionSet에 정의된 해당 조건을 체크하게 되고, 만약 이 대응되는 조건을 만족하면 패턴에 정의된 해당 업체의 프로세스를 자동으로 트리거링함을 정의하고 있다. 위의 RF-ECA 규칙을 토대로 본 연구의 범위에서 초점을 맞추고 있는 CSM-I과 CSM-II에 대한 처리 예제와 RF-ECA 규칙을 생성한다.

5.2 RFID 기반 물류 프로세스 예제

Inter-workflow 패턴들을 사용하는 것으로, 프로세스들 사이의 관계들을 수립할 수 있다. 아래 <그림 9>의 예제 프로세스를 고려해보자. 그림에는 세 개의 조직이 있고, 각 조직들은 조직 자신의 프로세스를 가지고 있다. 제조업체-고객 사이의 물류를 처리하기 위해서, 모든 파트너들은 시스템적인 상호운영이 필요하다. 프로세스들 사이의 관계를 모델링하고, 파트너들 사이에서 그들의 상호운영을 자동화하기 위해 inter-workflow 패턴을 사용한다.



〈그림 9〉 물류 프로세스 예제

위의 물류 프로세스 예제 <그림 9>에서 Rule 1과 Rule 2로 표현된 CSM-I과 CSM-II 패턴을 위한 RFID 기반의 ECA 규칙의 예제는 다음과 같이 정리할 수 있다. 그림처럼, 다섯 개의 프로세스들은 inter-workflow 패턴들에 의해 상호관계를 가지게 된다. 예를 들면, Supplier의 'Manufacturing process(P_{A1})'과 'Warehousing process(P_{C1})'은 CSM-I 패턴에 의해 상호관계를 가지게 된다. CSM-I 패턴의 정의에 따라, 'Rule 1'은 Supplier의 P_{A1}이 완료되면, Warehouse의 P_{C1}이 트리거링된다. 그리고 CSM-II의 정의에 의해, 'Rule 2'로 매핑된 Retailer의 P_{B1}의 태스크 t_{B8}에 의해 Warehouse의 P_{C2}가 트리거링된다.

예제 1은 'Warehouse #1' 위치에서 100개의 수량에 대한 RFID 이벤트를 e_1으로 정의하고 CSM-I에 따라 비즈니스 프로세스 이벤트 e_2에서 프로세스 P_{A1}에서 발생한 이벤트가 'Finish' 이벤트임을 나타내는 두 이벤트의 논리곱인 $e_1 \wedge e_2$를 복합이벤트 $e_3 = <(e_1 \wedge e_2), t_2, t_3>$으로 정의하고, tag_1의 *quantity*가 '100'이고 P_{A1}에서 발생한 이벤트가 'Finish'이면, 매핑된 액션인 프로세스 P_{C1}의 트리거링을 수행하라는 것을 의미한다.

〈예제 1〉 Rule-1 : CSM-I

이벤트 정의 :
RFE = {e_1}, BPE = {e_2}, CPE = {e_3}

e_1 ($\in$RFE) =
<tag_1, reader_101, 2009.05.01.13.30.24>

tag_1.*quantity* == '100'
reader_101.*location* == 'WH #1'

e_2 ($\in$BPE) =
<P_{A1}, Finish, 2009.05.01.13.30.30>

e_3 ($\in$CPE) = <($e_1 \wedge e_2$), t_1, t_2>

RF-ECARule (R 1 : CSM-1)
ON { e_3 }
IF { (e_1.tag_1.*quantity* == '100' AND
e_1.reader_101.*location* == 'WH #1')
AND (e_2.*eventType* == Finish) }
DO { P_{C1}.*initiate*() }
EndRF-ECARule

다음으로 프로세스가 앞선 프로세스의 특정 태스크에 의해 트리거링되는 CSM-II 패턴에서는, Retailer의 'Order process(P_{B1})'를 구성하는 태스크 중 하나가 Warehouse의 'Taking

out of the warehouse process(P_{C2})'와 관계를 가지게 되며, CSM-I 패턴의 정의에 따라, t_{B8}이 시작되면, P_{C2}가 트리거링된다.

예제 2를 살펴보면 'Counter #4'의 위치에서 'Monitor_001'이 판매가 되고, 'Backroom #5' 위치에서 'Monitor_001'의 현재 재고 '49'에 대한 RFID 이벤트 e_4와 e_5를 두 이벤트의 논리곱인 $e_4 \wedge e_5$로 정의하고, CSM-II에 따라 P_{C1}의 태스크 t_8이 순차적으로 시작됨을 나타내는 세 개의 이벤트의 복합이벤트인 $e_7 = <(e_4 \wedge e_5) \Rightarrow e_6, t_4, t_5>$을 정의하였다.

〈예제 2〉 Rule-2 : CSM-II

이벤트 정의 :
RFE = {e_4, e_5}, BPE = {e_6}, CPE = {e_7}

e_4 (∈RFE) =
<tag_4, reader_401, 2009.05.02.14.40.10>

tag_4.*productName* == 'Monitor_001'
tag_4.*status* == 'sold out'
reader_401.*location* == 'Counter #4'

e_5 (∈RFE) =
<tag_5, reader_501, 2009.05.02.14.40.11>

tag_5.*productName* == 'Monitor_001'
tag_5.*currentInventory* == '49'
reader_501.*location* == 'Backroom #5'

e_6 (∈BPE) =
$<P_{C1}.t_{B8}$, Begin, 2009.05.02.14.40.25>

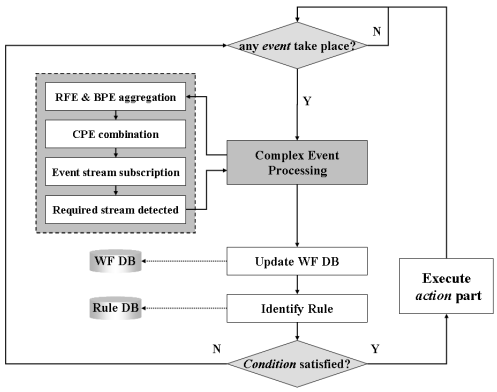
e_7 (∈CPE) = $<(e_4 \wedge e_5) \Rightarrow e_6\ t_4, t_5>$

RF-ECARule (R 2 : CSM-II)
ON { e_7 }
IF { (e_4.tag_4.*productName* == 'Monitor_001' AND e_5.tag_5.*currentInventory* < '50') AND (e_4.tag_4.*productName* == e_5.tag_5.*productName*) AND (e_7.*eventType* == Begin) }
DO { P_{C2}.*initiate*() }
EndRF-ECARule

e_4의 *productName*은 'Monitor_001'이고 e_5의 현재 재고 *currentInventory*가 '50'보다 적고 두 RFID 이벤트 e_4와 e_5의 *productName*이 같으며, 프로세스 P_{C1}의 태스크 t_8에 의해 'Begin' 이벤트가 발생되면, 프로세스 P_{C2}의 P_{C2}.*initiate*() 액션의 실행을 의미한다.

5.3 Inter-Workflow 제어 절차

본 연구는 RF-ECA 룰들로부터 룰을 해석하고, 관리하고, 추론할 수 있는 룰 엔진을 사용하여 inter-workflow의 실행을 제어한다. 제어 절차는 두 저장소인 워크플로우 데이터베이스(WF DB)와 규칙 데이터베이스(Rule DB)를 사용한다. 이벤트가 발생할 때, 시스템은 먼저 WF DB에 기록하고, 그 다음 Rule DB에서 규칙의 식별을 시도한다. 시스템은 식별된 RF-ECA 규칙의 이벤트(Event)를 발생된 이벤트와 매치시킨다. 그리고 WF DB에 존재하는 사실(fact)들로부터 직접적으로 추출되거나 추론된 그 사실에 대한 참조를 통해 시스템은 규칙의 조건(Condition)들을 평가한다. 그리고 만약 모든 조건들이 만족되고 나

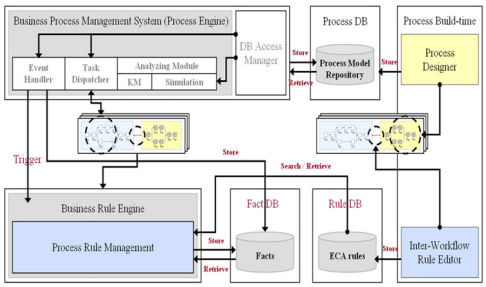


〈그림 10〉 규칙 기반 컨트롤 프로시저

면, 액션(Action)이 수행된다. 액션의 수행 역시 이벤트를 발생시킬 수 있으며 전체 프로세저가 <그림 10>에 나타나 있다.

5.4 프로토타입 시스템 구현

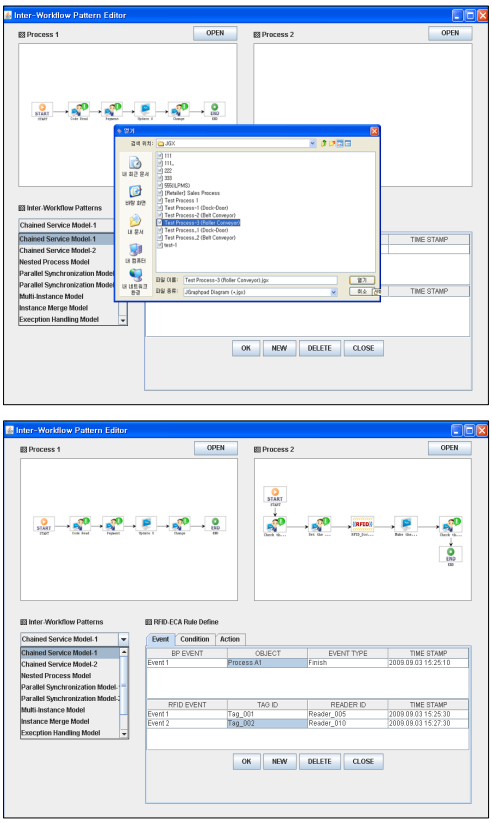
아래의 <그림 11>과 같이 inter-workflow 관리를 위한 프로토타입 시스템을 개발하였다. 프로세스 정의 단계에서, 프로세스들을 호출한 이후에, 'Inter-Workflow Rule Editor'를 사용하여 프로세스들 사이의 inter-workflow 관계들이 매핑 된다. 물류 프로세스가 실행되는 동안에, 각 기본 프로세스들은 자신들의 BPM 시스템에 의해 관리되고, 그들 사이의 관계들은 'Business Rule Engine'에 의해 관리된다.



〈그림 11〉 프로토타입 시스템 구조

<그림 12>는 inter-workflow 간의 편집을 위해 사용된 사용자 인터페이스를 보여준다.

사용자는 설계되어 있는 두 프로세스를 열고, inter-workflow 패턴 리스트에서 원하는 모델을 선택하는 것으로 관계들을 만들 수 있다. 이 시스템은 ECA 규칙을 자동으로 생성하고, 사용자는 약간의 파라미터 값을 수정하는 것으로 규칙을 간단하게 조절할 수 있다.



〈그림 12〉 프로토타입 시스템

6. 결 론

물류 환경에서, 자재나 제품은 공급업자로부터 시작하여 최종적으로 고객에게 배달된다. 그런 환경에서, 서로 다른 참여자들에 의해 관리되는 여러 프로세스들을 만나게 된다. 그러한 흐름을 관리하기 위해, 독립된 프로세스 주체들 사이의 관계를 표현하는 inter-workflow 패턴을 사용한다. 이러한 패턴들이 정의 단계에서 기술될 때 ECA 규칙은 이후 실행 단계에서의 제어를 위해 자동적으로 생성된다.

본 연구의 의의는 여러 조직들에 의해 운영되는 프로세스들을 통합하는 것에서 찾을 수 있고, 그것에 의해 프로세스들 사이의 연결을 효과적이고 효율적으로 수행되게 하는 것을 가능하게 한다. inter-workflow 패턴과 프로세스 사이의 상호작용을 룰 기반으로 관리하는 것은 물류 환경에서 여러 가지 장점을 준다. 첫 째, 그것은 조직 간의 물류 프로세스들의 모델링과 수행을 가능하게 한다. 전통적인 프로세스 패턴들은 단일 프로세스 내에서 간단한 의미만을 지원하고 물류 흐름 지원하지 않는데 반해, inter-workflow 패턴은 물류 프로세스 내에 참여하는 파트너들 사이의 관계의 표현을 가능하게 한다. 둘 째, 다른 프로세스로부터의 프로세스 트리거링이 프로세스 실행 로직으로부터 분리될 수 있다. inter-workflow 패턴과 관련된 적절한 규칙이 준비된 이후에, 물류 흐름을 자동으로 트리거링 할 수 있는 제삼자 (third-party) 규칙 엔진에 의해 모든 프로세스의 상호운영이 실행되고 관리된다. 셋 째, 사용자에 의해 Rule DB가 확장되기 때문에 새로운 타입의 프로세스 관계가 추가될 수 있다. 만약 새로운 관계의 이벤트, 컨디션 그리고 액션이 표현될 수 있으면, 그것은 시스템에 의해 모델링되고 실행될 수 있다.

추후 연구로는 첫째, 본 연구에서는 두 프로세스 사이에서 발생하는 관계 패턴들 위에서만 초점을 맞췄다. 그것을 셋 또는 그 이상의 프로세스로 확장하는 것이 필요할 것이다. 이는 실제 여러 기업들이 복잡하게 얽혀있는 물류 환경에서의 프로세스를 효율적으로 관리하기 위한 방안을 제공할 수 있다.

참 고 문 헌

[1] 배혜림, 서용원, 최용선, 장진영, “BPM을 이용한 웹서비스 기반의 SCM 프로세스 실행”, 한국전자거래학회지, 제9권, 제4호, 2004, 65-83.

[2] Bae, H., Hur, W., Yoo, W., Kwak, B., Kim, Y., and Park, Y., “Document Configuration Control Processes Captured in a Workflow,” Computers in Industry, Vol. 53, No. 2, 2004, pp. 117-131.

[3] Bae, J., Bae, H., Kang, S.-H., and Kim, Y., “Automatic Control of Workflow Processes Using ECA Rules,” IEEE transactions on knowledge and data engineering, Vol. 16, No. 8, 2004, pp. 1010-1023.

[4] Basu, A. and Kumar, A., “Research Commentary : Workflow Management Systems in e-Business,” Information Systems Research, Vol. 13, No. 1, 2002, pp. 1-14.

[5] Castai, F., Ceri, S., Pernici, B., and Pozzi, G., “Deriving Active Rules for Workflow Enactment,” Proceedings of 17 Int’l Conference on Database and Expert Systems Applications, 1996, pp. 94-110.

[6] Chen, L., Li, M., and Cao, J., “ECA Rule-Based Workflow Modeling and Implementation for Service Composition,” IEICE Transactions on Information and Systems, Vol. E89-D, No. 2, 2006, pp. 624-630.

[7] Gunasekaran, A. and Ngai, E. W. T., "Information systems in supply chain integration and management," European Journal of Operational Research, Vol. 159, No. 2, 2004, pp. 269-295.

[8] Hagen, C. and Alonso, G., "Exception Handling in Workflow Management Systems," IEEE Transactions on software engineering, Vol. 26, No. 10, 2000, pp. 943-958.

[9] Klein, M. and Dellarocas, C., "A knowledge-based approach to handling exceptions in workflow systems," Computer Supported Cooperative Work, Vol. 9, No. 3, 2000, pp. 399-412.

[10] Lee, S., Kim, T., Kang, D., Kim, K., and Lee, J., "Composition of Executable Business Process Models by Combining Business Rules and Process Flows," Expert Systems With Applications, Vol. 33, No. 1, 2007, pp. 221-229.

[11] Liu, R., Kumar, A., and Aalst, W., "A Formal Modeling Approach for Supply Chain Event Management," Decision Support Systems, Vol. 43, No. 3, 2007, pp. 761-778.

[12] Liu, R., and Kumar, A., "An Analysis and Taxonomy of Unstructured Workflows," 3rd International Conference on Business Process Management (BPM 2005) Nancy, France, Springer-Verlag, Lecture Notes in Computer Science, Vol. 3649, 2005, pp. 268-284.

[13] Lucia, A. D., Francese R., and Tortora, G., "Deriving workflow enactment rules from UML activity diagrams : a case study," Proceedings of IEEE Symposium on Human Centric Computing Languages and Environments, 2003, pp. 211-218.

[14] Rhee, S.-H., Cho, N., and Bae, H., "A More Comprehensive Approach for Enhancing Business Process Efficiency," Lecture Notes in Computer Science, 4558(HCI International 2007), 2007, pp. 955-964.

[15] Rhee, S. -H., Bae, H., and Choi, Y., "Enhancing the Efficiency of Supply Chain Processes through Web Services," Information Systems Frontier : Special Issue on from Web Services to Services Computing, Vol. 9, No. 1, 2007, pp. 103-118.

[16] van der Aalst W. M. P., ter Hofstede A. H. M., Kiepuszewski B., and Barros A., "Advanced workflow patterns," Lecture notes in Computer Science, Vol. 1901, 2000, pp. 18-19.

[17] WfMC-TC00-1003, "The Workflow Reference Model," Workflow Management Coalition, Lighthouse Point, FL, 1995.

[18] Wu, J., Wang, D., and Sheng, H., "ECA Rule-based RFID Data Management," RFID Eurasia (2007 1st Annual), 2007, pp. 1-5.

[19] Zang, C., Fan, Y., and Liu, R., "Architecture, implementation and application of complex event processing in enterprise information systems basedon RFID," Information Systems Frontier, Vol. 10, 2008, pp. 543-553.

〈부 록〉

* 물류 환경에서 활용 가능성이 높은 여러 패턴에 대한 RFID-based ECA 규칙 예제

이벤트 정의 : RFE = {e_{RF}}, BPE = {e_{BP}}, CPE = {e_C}
e_{RF} = < *tagID*, *readerID*, *timestamp* >
e_{BP} = < *objectID*, *eventType*, *timestamp* >
e_C = < e_t, *start_time*, *end_time* >

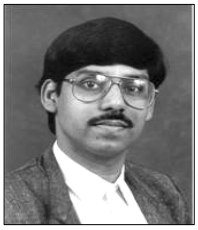
Pattern	RFID-based ECA (Event - Condition - Action) rules
CSM	**Rule** (R 1-1 : CSM-I) **ON** { e_C } (단, e_{BP} = < P_A, Finish, *timestamp* > 일 때,) **IF** { (e_{RF}.conditionSet = satisfied) ⊗ (e_{BP}.conditionSet = satisfied) } **DO** { P_B.*initiate*() } **EndRule**
	Rule (R 1-2 : CSM-II) **ON** { e_C } (단, e_{BP} = < $P_B.t_i$, Finish, *timestamp* > 일 때,) **IF** { (e_{RF}.conditionSet = satisfied) ⊗ (e_{BP}.conditionSet = satisfied) } **DO** { P_B.*initiate*() } **EndRule**
NSpM	**Rule** (R 2-1 : NSpM) **ON** { e_C } (단, e_{BP} = < $P_A.t_i$, Begin, *timestamp* > 일 때,) **IF** { (e_{RF}.conditionSet = satisfied) ⊗ (e_{BP}.conditionSet = satisfied) } **DO** { P_B.*initiate*() } **EndRule**
	Rule (R 2-2 : NSpM) **ON** { e_C } (단, e_{BP} = < P_B, Finish, *timestamp* > 일 때,) **IF** { (e_{RF}.conditionSet = satisfied) ⊗ (e_{BP}.conditionSet = satisfied) } **DO** { $P_A.t_i$.*finish*() } **EndRule**
PSM	**Rule** (R 3-1 : PSM-I) **ON** { e_C } (단, e_{BP} = < $P_B.t_i$, Finish, *timestamp* > 일 때,) **IF** { (e_{RF}.conditionSet = satisfied) ⊗ ($P_A.t_i$.*state* == 'SUSPENDED') } **DO** { $P_A.t_i$.*resume*(), $P_A.t_i$.*finish*() } **EndRule**
EHM	**Rule** (R 6-1 : EHM-1) **ON** { e_C } (단, e_{BP} = < $P_A.t_i$, Fail, *timestamp* > 일 때,) **IF** { (e_{RF}.conditionSet = satisfied) ⊗ (e_{BP}.conditionSet = satisfied) } **DO** { P_{EHP}.*initiate*(), $P_A.t_i$.*suspend*() } **EndRule**
	Rule (R 6-2 : EHM-2) **ON** { e_C } (단, e_{BP} = < P_{EHP}, Finish, *timestamp* > 일 때,) **IF** { (e_{RF}.conditionSet = satisfied) ⊗ $P_A.t_i$.*state* == 'SUSPENDED') } **DO** { $P_A.t_i$.*resume*() } **EndRule**

($\boldsymbol{P_A}$ = preceding process, $\boldsymbol{P_B}$ = succeeding process, $\boldsymbol{P_{EHP}}$ = exception handling process, ⊗ = { ∧, ∨, ⇒, ¬ }, **conditionSet** = { c_i | i = 1, ⋯, N })

저 자 소 개

유영웅 (E-mail : hero@pusan.ac.kr)
2007년~현재 부산대학교 산업공학과 박사과정 수료
관심분야 RFID 기반 물류프로세스 관리, Rule-based System

배혜림 (E-mail : hrbae@pusan.ac.kr)
2002년 서울대학교 산업공학과 공학박사
2003년 삼성카드, 정보기획팀
2003년~2004년 동의대학교 인터넷비즈니스학과
2004년~현재 부산대학교 산업공학과 부교수
관심분야 BPM, 웹서비스, 물류프로세스 관리



Sajal K. Das (E-mail : das@cse.uta.edu)
현재 Professor, Dept. of Computer Science and Engineering, University of Texas at Arlington

구훈영 (E-mail : koohy@etri.re.kr)
2002년 서울대학교 산업공학과 공학박사
2002년~2005년 LG CNS
2005년~현재 한국전자통신연구원(ETRI) 선임연구원
관심분야 확률모형, 물류, RFID