다중 워크플로우 충돌패턴에 관한 연구

김보연[a] and 박진수[b]

[a] 서울대학교 경영대학
151-916 서울시 관악구 관악로 599
Tel: +82-2-880-6957, Fax: +82-2-872-6366, E-mail: kby@snu.ac.kr

[b] 서울대학교 경영전문대학원
151-916 서울시 관악구 관악로 599
Tel: +82-2-880-9385, Fax: + 82-2-872-6366, E-mail:jinsoo@snu.ac.kr

Abstract

최근 기업 환경이 복잡해지고 분산화 되어감에 따라 각 기업간의 그리고 부서 및 지점간의 프로세스 상호운용 및 협력이 중요한 이슈로 떠오르고 있다. 이러한 시점에서 기업간 비즈니스 프로세스와 메시지 등을 교류할 수 있는 상호운용성이 기업들에게 필수적인 요소로 고려되고 있다. 지금까지 많은 기술자와 연구자들에 의해 기업간 워크플로우 상호운용성을 해결할 수 있는 기술적인 노력이 꾸준히 이루어져 왔다. 기술적 문제의 해결도 중요하지만 이보다 우선적으로 해결되어야 될 것은 비즈니스 프로세스 상호운용성에 대한 개념적 모델링과 정확한 분석이다. 본 논문에서는 기존의 워크플로우 패턴들에 대한 연구들을 정리해 보여 줄 뿐만 아니라, 여러 기업간 비즈니스 프로세스가 상호운용될 시 충돌을 유발시킬 수 있는 패턴들을 제시할 것이다. 이러한 패턴들에 관한 연구는 여러 기업들이 복잡한 비즈니스 프로세스를 문제없이 효율적으로 상호운용할 수 있도록 도움을 주는 기본적인 정보가 될 것이다.

Keywords:

워크플로우 관리 시스템, 상호운용성, 비즈니스 프로세스, 충돌패턴

1. 서론

최근 기업의 비즈니스 프로세스는 기업들의 가치 있는 핵심자산으로 그 중요성이 날로 부각되고 있다. 기업의 환경이 점차 복잡해지고 경쟁이 치열해지면서, 기업간 그리고 조직간에 서로 협력하지 않고서는 경쟁우위를 가지기 힘든 실정이다. 2002년 딜로이트 컨설팅에서 300여 개의 기업들에게 설문조사한 결과를 보면, 기업간 상호 협력을 하는 것이 비즈니스의 효율성을 무려 70% 이상 증가시키는 효과가 있다고 하였다[6].

현재 상용화된 워크플로우 관리 시스템 (Workflow Management System, WfMS)은 대략 200여 개 이상으로 그 종류와 기능이 다양하며, 많은 기업에서는 자신의 비즈니스 프로세스를 효율적으로 다루기 위해 이를 도입하였다. 지금까지 많은 기업들이 비즈니스 프로세스의 자동화를 위해 워크플로우 관리 시스템을 도입하였다면, 최근에는 여러 조직간에 걸쳐 비즈니스를 신속하고 효율적으로 수행하기 위해 워크플로우 관리 시스템을 활용하고 있다. Plesums(2003)은 " 워크플로우 기술의 개발과 사용은 사람들의 작업을 라우팅하도록 지원하는 간단한 개념에서, 자원간에 작업을 수평적으로 또는 수직적으로 라우팅하는 것에 이르기까지 복잡화되고 있으며, 데이터가 프로세스들간에 이동되고 이 데이터들이 시스템과 통합되면서 워크플로우는 기업 애플리케이션의 통합영역으로 확장되고 있다" 라고 하며 기업간 비즈니스 프로세스의 통합의 중요성을 강조하고 있다[18].

워크플로우 상호운용성은 현재 기업들에게 있어 중요하게 고려되고 있지만 아직까지는 해결되어야 할 문제들이 많다. 지금까지 몇몇 연구자, 기술자 등에 의해 워크플로우간의 상호운용이 원활하게 진행될 수 있도록 연구가 진행되어 왔다. 대표적인 노력으로 WfMC에서는 상호운용성 표준 및 Wf-XML 바인딩 등의 표준을 제정하여 워크플로우 관리 시스템의 상호운용성 확보에 노력을 가하고 있다. 또한 웹 서비스(Web Services) 상호운용성에 대한 연구[4, 13] 에이전트를 기반으로 상호운용성을 확보하고자 하는 연구[2, 10] 등 다양한 연구가 진행되고 있다.

상호운용성의 필요에 따라 여러 기관 및 연구자들이 상호운용성 문제를 해결하고자 노력해왔지만, 지금까지의 노력들은 대부분 비즈니스 프로세스 상호운용시 발생하는 기술적인 문제의 해결에 중점을 두고 있다는 것을 기존의 연구에서 찾을 수 있었다. 물론 기술적 문제의 해결도 중요하지만 우선적으로 워크플로우간

상호운용에 대한 올바른 개념적 모델링이 선행되어야 한다. 복잡한 비즈니스 프로세스의 정확한 분석과 설계를 위해 올바른 모델링은 필수적인 요소이다.

이에 따라 본 연구에서는 조직간 비즈니스 프로세스의 워크플로우 상호운용에 대한 개념적 접근을 하겠다. 그 중에서도 특히 기업들이 잘못된 방향으로 워크플로우를 모델링 하는 것을 피할 수 있도록 기업간 비즈니스 프로세스의 상호운용시 발생할 수 있는 충돌패턴(conflict pattern)들에 대해 자세히 살펴보도록 하겠다. 이러한 패턴들에 관한 연구는 여러 기업들이 복잡한 비즈니스 프로세스를 문제없이 효율적으로 상호운용할 수 있게 하는데 도움이 되는 핵심 정보가 될 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 워크플로우 모델링 기법 중 본 논문에서 사용하고 있는 페트리 네트에 대해 알아보았으며, 3장에서는 단일 워크플로우의 패턴을 소개하였다. 4장에서는 상호운용성 패턴을 정리하였으며 또한 기존의 연구에는 없는 워크플로우 상호운용시 나타날 수 있는 충돌패턴들을 제안하였다. 마지막으로 5장에서는 결론과 본 논문의 의의와 향후 연구를 제시하였다.

2. 워크플로우 모델링 - 페트리 네트(Petri Nets)

워크플로우를 분석하기 위해서는 조직의 비즈니스 프로세스의 동적인 구조를 정확히 표현하는 워크플로우 모델을 작성하여야 한다. 지금까지 워크플로우를 모델링하는데 주로 사용된 도구로는 IDEF3[7], ARIS EPC[12], 페트리 네트[17] 등을 뽑을 수 있다. 이 중에서도 본 연구에서는 Carl Adam Petri [9]에 의해 고안된 페트리 네트를 이용하여 워크플로우 시스템을 분석하고 모델링할 것이다.

페트리 네트는 트랜지션(transition)과 플레이스(place)라는 두 개의 노드로 구성된 이진 그래프의 형태이다[8]. 페트리 네트에서는 일반적으로 태스크(task)를 트랜지션으로 본다. 트랜지션이 발생하기 위해서는 시스템의 여러 조건이 만족되어야 된다. 이러한 조건에 대한 정보가 플레이스에 저장된다. 페트리 네트에서 어떠한 플레이스들은 한 트랜지션의 입력 조건이 되고, 어떠한 플레이스들은 한 트랜지션의 출력 조건이 된다. 그리고 트랜지션과 플레이스 사이의 상호 연관관계를 아크(arc)로 표현한다. 이러한 관계를 통하여 페트리 네트는 다음과 같이 정의할 수 있다.

페트리 네트는 다음의 플레이스, 트랜지션, 아크의 튜플(tuple)로 구성된다.

(P, T, A)

여기서,

- P: $\{p_1, p_2, \ldots, p_n\}$, 플레이스의 유한집합;
- T: $\{t_1, t_2, \ldots, t_n\}$, 트랜지션의 유한집합, $(P \cap T = \varnothing)$;

$A \subseteq (T \times P) \cup (P \times T)$는 아크의 집합이다;

A_i : 입력 접속 행렬(Input Incidence Matrix), $(T \times P) \rightarrow \{0, 1, 2, 3, \ldots\}$;

A_o : 출력 접속 행렬(Output Incidence Matrix), $(P \times T) \rightarrow \{0, 1, 2, 3, \ldots\}$.

페트리 네트에서는 아래의 [그림 1]과 같이 플레이스는 타원으로 트랜지션은 사각형으로 표현하며, 아크는 플레이스와 타원 사이에 화살표로 표현된다. 또한 [그림 1]에서 p_1을 보면 세 개의 검정색 점들이 있다. 이것들은 토큰(token)이라 부르며, 플레이스는 토큰을 담고 있을 수 있다. 페트리 네트는 이산현상 시스템의 동작을 표현하기 위하여 '토큰'의 개념을 도입한다. 토큰은 각 플레이스에 부여된 상태의 값이다. 플레이스는 시스템의 이벤트를 발생시키기 위한 조건 정보를 담고 있고 그 조건 정보가 토큰의 개수로 표현된다. 그러므로 각 플레이스에 부여된 토큰 개수들의 정보로 시스템의 상태를 나타낸다. 각 토큰은 하나의 플레이스에 고정되어 있지 않고 상태의 흐름에 따라 위치가 변화된다.

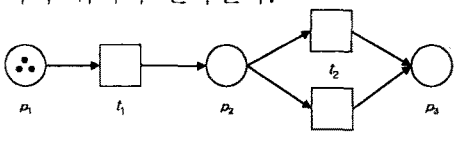


그림 1- 페트리 네트 예

3. 단일(single) 워크플로우[1]

일반적으로 시스템 디자인에서 패턴(pattern)이란 어떠한 비임의적인(non-arbitrary) 상황에서 계속적으로 반복을 하는 추상적인 것들에 대한 명확한 형태를 말한다[11]. 비즈니스 프로세스도 마찬가지로 일정한 패턴으로 구분할 수 있다. 본 연구에서 말하는 워크플로우 패턴이란 디자인 패턴의 특수한 형태로 기업의 비즈니스 프로세스상에서 지속적으로 반복되는 상황이나 문제들 그리고 문제들을 처리하는 방법들을 워크플로우 관리 시스템 개발을 위하여 명확한 형태로 정의하는 것을 말한다.

패턴의 중요성을 인식하고 워크플로우 분야에서도 이에 대한 몇몇 연구가 진행되어 왔다[1, 5, 15]. 지금까지 진행되어온 워크플로우 패턴에 대한 대표적인 연구로 van der Aalst et al. [15]의 연구를 뽑을 수 있다. 그들은 워크플로우에서 발생할 수 있는 여러 가지 패턴을 제안하였다. 그들이 정리한 패턴은 순차(sequential),

[1] 본 논문에서는 다음과 같은 기호를 사용할 것이다.
워크플로우의 집합은 $Wf = \{wf_1, wf_2, \ldots, wf_n\}$으로, 플레이스 집합은 $P = \{p_1, p_2, \ldots, p_n\}$으로 표현하고, 트랜지션 집합은 $T = \{t_1, t_2, \ldots, t_n\}$으로 표현한다.

분기(split), 병합(join) 등 총 21가지의 패턴을 제시하였다. 그러나 그들의 연구에서는 본 연구에서 고려하고자 하는 조직간 비즈니스 프로세스 상호운용시 발생할 수 있는 충돌적 패턴들에 대해서는 언급하지 않고 있다. 이에 따라 본 연구에서는 워크플로우간 상호운용시 발생할 수 있는 충돌패턴들을 제시하고자 한다.

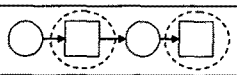
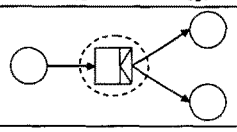
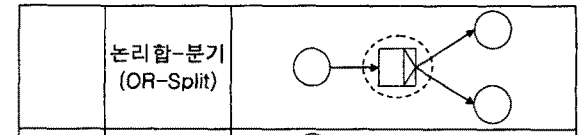
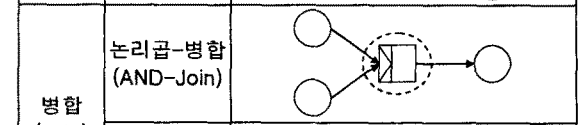
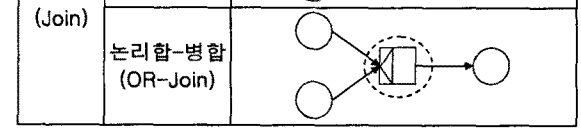
3.1 단일 워크플로우 패턴

van der Aalst et al. [15]이 제안한 단일 워크플로우의 기본적인 패턴들을 정리하면 [표 1]과 같다. 대표적으로 순차, 분기, 병합이 있으며, 분기는 논리곱-분기(AND- Split)와 논리합-분기(OR- Split)로 나누어지며 병합은 논리곱-병합(AND-Join)과 논리합-병합 (OR-Join)으로 구성된다. 이들은 [표 1]에서 점선으로 표시되어있는 부분과 같이 표현한다.

순차란 워크플로우 흐름상 선행 작업이 완료된 후 다음 작업을 진행하는 기본적인 패턴이다. 즉, 순서에 따라 각 태스크가 차례대로 실행되는 것을 말한다. 분기란 하나의 태스크로부터 다수의 태스크로 분할되어 실행되는 것을 말한다. 논리곱-분기란 분기점으로의 입력은 하나이지만 출력이 다수인 경우, 병렬적으로 실행되는 다음 태스크들이 동시적으로 실행될 수도 있으며 순서적으로 실행될 수도 있다. 논리합-분기는 의사결정이나 데이터에 의해서 뒤에 실행될 병렬적인 태스크 중 하나만 선택하는 경우를 말한다. 논리합-분기시 실행될 병렬적인 태스크의 결정은 [표 1]에서와 같이 가능한 전제조건이 만족되는 태스크로 실행되거나 다음 실행될 수 있는 병렬적인 태스크 중 결정 법칙에 만족하는 태스크가 진행하게 된다. 이는 워크플로우 내에서 하나의 태스크가 복수의 대안 진행 태스크가 있을 경우 분기에 대한 결정을 하는 역할을 한다.

병합이란 다수의 병렬적인 태스크들이 하나의 태스크로 합쳐지는 것을 말한다. 병합 중 논리곱-병합은 워크플로우 내에서 적어도 2개의 병렬 활동이 하나의 태스크로 모이게 되는 지점이며, 두 태스크의 조건이 모두 만족되어야만 다음 태스크가 실행되는 경우를 말한다. 논리합-병합은 워크플로우 내에서 적어도 2개의 대안 활동 태스크가 다음 단계로 재병합되는 것으로 둘 중 어느 하나만 만족하면 다음 태스크가 실행 된다.

표 1 - 단일 워크플로우 패턴

패턴 유형		표기(Notation)
순차 (Sequential)		
분기 (Split)	논리곱-분기 (AND -Split)	
	논리합-분기 (OR-Split)	
병합 (Join)	논리곱-병합 (AND-Join)	
	논리합-병합 (OR-Join)	

4. 다중(Multi) 워크플로우 - 상호운용성

본 장에서는 상호운용성 패턴들을 살펴볼 것이며, 기존의 연구에서는 찾을 수 없었던 다수의 워크플로우가 상호운용할 경우 발생할 수 있는 패턴들 중 충돌이 발생할 수 있는 패턴을 제안하겠다.

4.1 워크플로우 상호운용성 패턴

Casti et al. [3]과 van der Aalst [16]의 연구에서는 기업간 상호운용되는 워크플로우들에 대한 패턴들에 대해 연구하였다. 이들의 연구에서는 상호운용시 발생할 수 있는 속성들을 중심으로 정리하였을 뿐, 충돌패턴에 대한 내용을 구체적으로 제안하지 않았다.

1) 조건 의존적 실행

두 개의 상호운용하는 워크플로우 wf_a와 wf_b가 있다고 하자. 이 패턴은 두 개의 워크플로우가 상호운용시 $wf_a.t_1$과 $wf_b.t_2$사이의 실행 순서의 관계에서 태스크 $wf_b.t_2$가 임의의 태스크로부터 실행되기 이전에 태스크 $wf_a.t_1$이 임의의 태스크로부터의 실행이 종료되어야 한다. 즉, 조건 의존적 실행이란 상호운용하는 워크플로우의 선행 태스크가 먼저 실행되어야만 다른 워크플로우의 다음 태스크가 실행할 수 있는 경우이다. 이러한 관계를 실행 하기 위해서는 조건의존적인 관계를 가진 태스크를 서로 연결해주어야 된다.

2) 동일한 선행조건을 가지고 있는 태스크

두 개 이상의 워크플로우가 서로 교류할 때, 각 워크플로우의 태스크가 동일한 선행조건을 가지고 있는 경우가 있다. 즉, 두 기업의 태스크인 $wf_a.t_1$과 $wf_b.t_2$의 관계가 $wf_a.t_1 < wf_b.t_2$ 이고 $wf_b.t_2 < wf_a.t_1$이면서 상호배타적일 경우이다. 이 의미는 $wf_a.t_1$은 $wf_b.t_2$보다 먼저 실행되어야 되고 마찬가지로 $wf_a.t_2$는 $wf_b.t_1$보다 먼저 진행되어야 된다는 것으로 이들은 동일하다는 의미이다. 이는 서로 다른 워크플로우에서 각 태스크가 진행되고 있지만 결국은 하나의 동일한 태스크를 수행하고 있는 경우를 뜻한다. 그러므로 이들은 워크플로우

통합시 하나의 태스크로 묶을 필요가 있다.

3) 프로세스 인스턴스시 발생 가능한 패턴

Casati et al. [3]은 서로 다른 워크플로우들간의 프로세스 인스턴스 교류에 대한 관계에 대해 간략하게 다루고 있다. 크게 프로세스 인스턴스 상호운용에 대해 협력, 경쟁, 그리고 방해관계로 분류하였다. 첫째, 협력관계란 어떠한 태스크 A가 어떠한 인스턴스를 생성하면 그것을 다른 시점에서 B라는 태스크가 A의 인스턴스를 사용하는 경우이다.

둘째로 경쟁관계는 제한된 자원 때문에 경쟁이 일어나는 경우이다. A라는 태스크에서 발생된 a라는 인스턴스를 태스크 B와 태스크 C가 모두 필요로 할 경우가 발생할 수 있다. 만약 이때 인스턴스가 한 개뿐이라면 태스크 B와 C는 서로 경쟁이 필요한 관계이다.

마지막으로 방해관계란 잘못된 디자인으로 인해 서로의 진행을 방해하는 경우를 말한다. 이 경우는 한 쪽이 인스턴스를 사용하면, 다른 편의 인스턴스에 영향을 끼쳐 서로의 진행을 방해하는 경우이다.

4) 종료를 고려한 속성

이외에도 종료를 고려한 속성들로 충분, 기아, 데드락 관계를 정의하였다. 한 워크플로우의 모든 태스크가 상호운용하는 다른편의 워크플로우 프로세스상에 존재할 경우 이를 충분관계라고 한다. 기아상태란 워크플로우가 상호운용시 어떠한 태스크가 실행되지 못하고 무한히 실행을 대기하고 있는 경우이다. 한 워크플로우의 태스크가 실행화되어 다음 조건을 받아야 다른 워크플로우에서 태스크가 진행될 수 있는데, 그렇지 못해 무조건 기다리고 있는 상태이다. 마지막으로 워크플로우간의 데드락이란 상호운용하는 두 태스크가 수행할 수 없는 어떤 특정 태스크를 기다리고 있는 상태를 말한다. 즉, 두 개 이상의 워크플로우가 상호운용할 때 실행되지 못하고 있는 태스크 때문에 계속 다음 프로세스를 진행하지 못하고 무한히 기다리는 상황이 발생할 수 있는데, 이런 상태를 데드락이라고 한다. 일반적으로 기아상태를 데드락의 일부로 볼 수 있다.

Aalst [16]의 연구에서는 워크플로우가 상호운용할 경우 발생할 수 있는 충돌패턴으로 데드락과 무한반복의 경우를 보여주고 있으나 구체적인 패턴들을 제시하지 못하고 있다. 다음 절에서는 워크플로우 상호운용시 발생할 수 있는 충돌패턴들을 제시하도록 하겠다.

4.2 워크플로우 상호운용성 충돌패턴

여러 개의 워크플로우가 상호운용할 경우 발생할 수 있는 패턴들 중 충돌이 발생할 수 있는 패턴을 제안하겠다. 상호운용하기 전 독립적으로 운용되었던 워크플로우에서는 문제없이 진행되었지만, 조직의 필요에 의한 여러 이유로 상호운용하면서 비즈니스 프로세스가 원활하게 진행되지 못하는 경우가 발생할 수 있다. 서로 다른 두 개 이상의 워크플로우가 충돌 없이 상호운용하기 위해서 우선적으로 갖추어야 되는 전제조건으로는 상호운용에 참여하는 각각의 워크플로우가 충돌 없이 무결성을 가져야 한다는 것이다. 그러므로 본 연구에서는 상호운용하는 두 개의 워크플로우 wf_a와 wf_b가 상호운용하기 이전 독립적으로 운용되었던 워크플로우 자체에는 어떠한 충돌도 없이 무결하다는 가정하에 워크플로우간 상호운용시 발생할 수 있는 충돌패턴을 제안하겠다.

1) 순차충돌패턴

워크플로우 관리 시스템은 기업의 원활한 업무흐름을 위하여 일정한 작업절차를 정해 놓는다. 만약, 워크플로우 상호운용을 모델링할 시 기업의 전체적인 비즈니스 프로세스의 순차적 흐름을 고려하지 않는다면 충돌이 발생할 수 있다. 순차충돌패턴이란 상호운용하는 각각의 워크플로우의 순서에는 이상이 없었지만, 상호운용을 함에 따라 잘못된 모델링으로 인해 진행순서가 엇갈려 올바른 진행을 하지 못하게 되는 상황을 말한다. [그림 3]의 경우를 보면 $wf_b.t_x$는 $wf_a.t_d$가 실행되어야지만 실행할 수 있다. 하지만 $wf_a.t_d$는 상대편의 워크플로우의 $wf_b.t_y$가 실행되어야지만 논리곱-병합으로 모델링된 $wf_b.t_a$가 실행되어 작업을 수행할 수 있다. 즉, wf_a와 wf_b는 서로의 태스크가 실행되기만을 기다리고 있는 상황으로 충돌이 발생하였다고 할 수 있다.

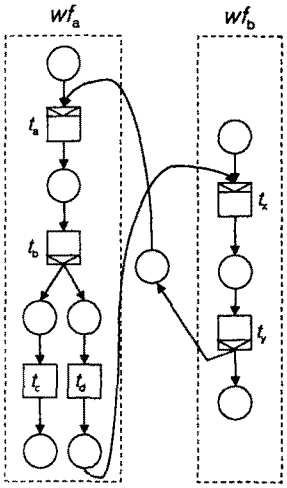


그림 3 - 순차충돌패턴

2) 논리곱-분기(AND-Split)시 순차충돌패턴

van der Aaslt & Hee [14]에 따르면 워크플로우를 모델링할 때 주의할 점으로

논리곱-분기와 논리곱-병합 그리고 논리합-분기와 논리합-병합을 유의하여 사용해야 된다고 했다. 비즈니스 프로세스의 순차적인 진행에 의해 발생할 수 있는 또 다른 충돌패턴으로는 논리곱-분기시 진행순서에 따른 충돌이 있다. 논리곱-분기시의 순차충돌패턴이란 논리곱-분기를 하는 특정 태스크의 분기 순서가 상대편 워크플로우 진행순서와 일치하지 않아 충돌이 생기는 경우를 말한다.

[그림 4]와 같이 논리곱-분기를 하는 태스크 $wf_a.t_b$의 실행순서가 [그림 4]에서 표시해 놓은 것과 같이 ①과 ②의 순서로 실행될 경우 상호운용하는 상대편의 워크플로우인 wf_b의 진행순서와 일치되지 않아 충돌이 발생할 수 있다. 만약 $wf_b.t_y$로 먼저 실행된 후 $wf_b.t_x$로 실행된다면 wf_b의 워크플로우의 진행순서는 $wf_b.t_x$가 실행된 후 $wf_b.t_y$가 실행되는데, $wf_b.t_x$가 논리곱-병합으로 모델링되어 있으므로 실행되지 못하고 있으며 $wf_b.t_y$ 또한 수행하지 못하는 상태가 된다. 즉, 태스크들이 수행할 수 없는 특정 태스크를 계속 기다리고 있는 상태에 빠지므로 충돌이 발생한다.

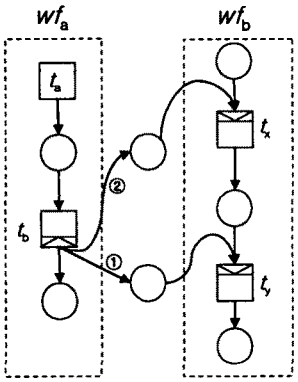


그림 4 - 논리곱-분기(AND-Split)시 순차충돌패턴

3) 논리합-분기와 논리곱-병합에 따른 충돌패턴

두 개 이상의 워크플로우가 상호운용될 시 발생할 수 있는 충돌패턴으로 논리합-분기와 논리곱-병합으로 발생할 수 있는 충돌이 있다. 이는 한 쪽의 워크플로우에서는 논리합-분기로 두 개의 태스크 중 한 개만 실행되는데, 다른 한 쪽의 워크플로우에서는 논리곱-병합으로 되어있어 반드시 두 개의 태스크의 값을 받아야 실행할 수 있는 경우이다. 이 때에는 논리곱-병합되는 태스크가 선행 태스크의 모든 값을 받아야지만 실행될 수 있으므로 비즈니스 프로세스가 원활히 진행되지 못한다.

[그림 5]를 보면 논리합-분기하는 태스크인 $wf_a.t_a$가 $wf_a.t_c$로 분기하는 경우에는 문제가 발생하지 않지만, 만약 $wf_a.t_b$로 분기한다면 상호운용하는 상대편 워크플로우의 태스크 $wf_b.t_y$는 실행할 수 없다. 왜냐하면 $wf_b.t_y$는 논리곱-병합으로 모델링되어있어 $wf_b.t_x$에서 보낸 값과 $wf_a.t_c$가 실행된 값을 모두 받아야지만 실행될 수 있기 때문이다.

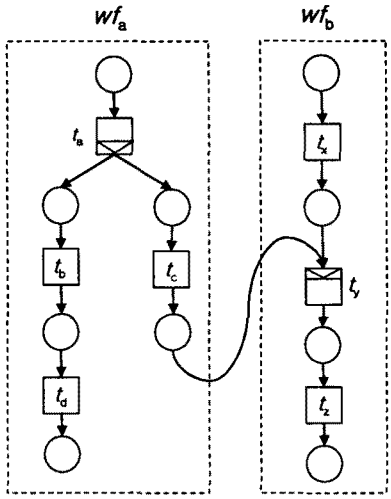


그림 5 - 논리합-분기와 논리곱-병합에 따른 충돌패턴

4) 순환충돌패턴

두 개 이상의 워크플로우가 상호운용시 발생할 수 있는 패턴으로 무한루프(infinite loop)가 있다. 프로세스 수행 중 끝없이 동일과정을 반복하는 경우로 부정확한 정보나 자기가 자기를 호출하는 프로세스가 반복되는 경우에 일어난다. [그림 6]을 보면 논리곱-분기로 모델링된 $wf_a.t_b$는 다음 태스크로도 실행이 되며 동시에 $wf_b.t_x$로 실행된다. $wf_b.t_x$는 순차적으로 $wf_b.t_y$로 실행되며 다시 논리곱-병합으로 모델링된 $wf_a.t_b$로 실행되면서 동일한 프로세스를 계속 반복하므로 충돌이 발생하게 된다. 즉, $wf_a.t_b$와 $wf_b.t_x$ 그리고 $wf_b.t_y$가 순환적으로 계속 실행하게 된다.

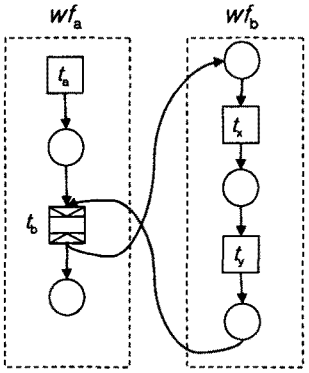


그림 6 - 순환충돌 패턴

5. 결론

많은 기업들은 기업간 상호운용의 중요성을 인식하고 있다. 본 연구는 워크플로우가 상호운용시 발생할 수 있는 여러 가지 충돌패턴들을 정리하였다. 상호운용시 발생할 수 있는 충돌패턴으로 크게

순차충돌, 논리곱-분기시 순차충돌, 논리합-분기와 논리곱-병합시 충돌, 그리고 순환충돌패턴을 제시하였다.

비즈니스 프로세스의 정확한 모델링을 위해 페트리 네트를 사용하여 더욱 정교하고 분석적으로 프로세스의 상호운용을 분석하였다. 지금까지의 비즈니스 프로세스의 워크플로우 상호운용의 중요성을 인식하고 있었지만 기술적인 문제해결에 중점이 되어 왔을 뿐, 단지 몇몇 연구에서만 상호운용에 따른 패턴들을 연구해 왔다. 특히 워크플로우 상호운용에 따른 충돌패턴을 정리한 연구는 찾기 힘들다.

만약 조직들이 사전에 올바른 모델링을 하지 않고 기업간 프로세스의 상호운용을 위해 주먹구구식으로 워크플로우간 상호운용을 한다면 많은 문제가 발생할 수 있다. 그러므로 이러한 문제상황을 미리 방지할 수 있도록 충돌패턴들을 정리하였다. 이미 구축된 시스템을 사용하다가 디자인 단계에서 잘못된 점이 발견된다면 상호운용하는 각 워크플로우 시스템은 서로 분산되어 있는 환경이기 때문에 문제를 해결하기 힘들뿐만 아니라, 많은 비용과 시간이 소모될 것이다. 이러한 충돌패턴에 대한 이해는 복잡한 비즈니스 프로세스에서 효율적으로 문제없이 여러 기업이 상호운용할 수 있는데 중요한 정보가 될 것이다.

향후 연구과제로는 다음과 같은 것이 있다. 앞서 정리한 다섯 가지 패턴 이외의 충돌패턴들에 대해 생각해 볼 필요가 있다. 비즈니스 프로세스 상호운용시 발생할 수 있는 데이터 흐름까지 고려한 보다 폭넓은 연구가 필요할 것이다. 또한 페트리 네트의 분석적인 장점을 활용하여 도달성 분석(reachability analysis)를 수행하면 좋을 것이다. 그리고 충돌패턴을 사전에 예방할 수 있도록 이를 자동적으로 발견할 수 있는 알고리즘의 개발도 필요할 것이라고 사료된다.

참고문헌

[1] Barros, O., Varas, S., "Frameworks derived from business process patterns," Technical Report 56, Industrial Engineering Department, University of Chile, 2004, (Available at http://www.obarros.cl).

[2] Biegus, L., Branki, C., "InDiA: a framework for workflow interoperability support by means of multi-agent systems," *Engineering Applications of Artificial Intelligence*, Vol. 17, No. 4, October 2004, pp. 825–839.

[3] Casati, F., Ceri,S., Pernici, B., and Pozzi,G., "Semantic workflow interoperability," *In Proceeding of the 5th Conference on Extending Database Technology (EDBT)*, Lecture Notes In Computer Science, Vol. 1057, 1996, pp. 443-462.

[4] Chen, M., Zhang, D., Zhou, L., "Empowering collaborative commerce with Web services enabled business process management systems," *Decision Support Systems*, Vol. 43, 2007, pp. 530– 546.

[5] Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.(editors), "Process-Aware Information Systems: Bridging People and Software Through Process Technology," John Wiley and Sons, 2005.

[6] Herman, J., "Making Collaborative Commerce Happen," 2002, (Available at http://www. bcr.com/ bcrmag/2002/10/p18.asp)

[7] Mayer, R., Menzel, C., Painter, M., Perakath, B., de Witte P. and Blinn T., "Information Integration For Concurrent Engineering (IICE) - IDEF3 Process Description Capture Method Report," Technical Report, September 1995, (Available at http://www.idef.com/pdf/idef3_fn.pdf).

[8] Murata, T., "Petri nets - properties, analysis, and applications," *Proceedings of the IEEE*, Vol. 77, No. 4, April 1989, pp. 541 – 580.

[9] Petri, C.A.., "Communication with automata," *In Schriften des IIM Nr. 3*, Institut für Instrumentelle Mathematik, Bonn, 1962, pp.16-27.

[10] Raghu, T. S., Jayaraman, B., "Toward an Integration of Agent- and Activity-Centric Approaches in Organizational Process Modeling: Incorporating Incentive Mechanisms," *Information Systems Research*, December 2004, Vol.15, No. 4, pp. 316 – 335.

[11] Riehle, D., Zullighoven, H., "Understanding and Using. Patterns in Software Development," *Theory and Practice of Object System*, Vol. 2, No. 1, 1996, pp. 3-13.

[12] Sheer, A., "ARIS-Business Process. Modeling," Springer, Berlin-Heidelberg, 1999.

[13] Shen, J., Grossmann, G., Yang, Y., Stumptner, M., and Schrefl, M., "Analysis of business process integration in Web service context," *Future Generation Computer Systems*, Vol. 23 , No. 3, March 2007, pp. 283-294.

[14] van der Aalst, W. M. P., van Hee, K. M., Workflow Management: Models, Methods, and Systems, MIT Press, 2002.

[15] van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A.P., "Workflow Patterns," *Distributed and Parallel Databases*, Vol. 14 No. 1, July 2003, pp. 5-51.

[16] van der Aalst, W.M.P., "Loosely. Coupled. Interorganizational Workflows: Modeling and Analyzing. Workflows. Crossing. Organizational. Boundaries,". *Information and Management*, Vol. 37, No. 2, 2000, pp. 67-75.

[17] van der Aalst, W.M.P., "The application of Petri Nets to workflow management," *The Journal of Circuits, Systems and Computers*, Vol. 8, No.1, 1998, pp.21-66.

[18] Workflow Handbook 2003, Published in association with the Workflow Management Coalition (WfMC). Edited by Layna Fischer, 2003.