

# 조직간 비즈니스 프로세스의 워크플로우 상호운용성 충돌패턴에 관한 연구

박진수

서울대학교 경영전문대학원  
(jinsoo@snu.ac.kr)

김보연

서울대학교 경영대학  
(kby@snu.ac.kr)

황유섭

서울시립대학교 경상대학 경영학부  
(yousub@uos.ac.kr)

정보기술이 발전함에 따라 최근 기업 환경이 매우 복잡해지고 분산화되고 있다. 이러한 시점에서 각 기업간의 그리고 부서 및 지점간의 프로세스 상호운용 및 협력이 중요한 이슈로 떠오르고 있다. 따라서 기업간 비즈니스 프로세스와 메시지 등을 교류할 수 있는 상호운용성이 기업들에게 필수적인 요소로 고려되고 있다. 지금까지 많은 기술자와 연구자들에 의해 기업간 워크플로우 상호운용성을 위한 개념적 프레임워크를 꾸준히 제안하여왔다. 그러나 이러한 개념적 프레임워크의 제안도 중요하지만 이보다 우선적으로 해결되어야 될 것은 비즈니스 프로세스 상호운용에 장애가 되는 충돌에 대한 개념적 모델링과 정확한 분석이다. 본 논문에서는 기존의 워크플로우 패턴들에 대한 연구들을 정리해 보여 줄 뿐만 아니라, 여러 기업간 비즈니스 프로세스가 상호운용시 발생할 수 있는 충돌패턴들을 제시할 것이다. 이러한 충돌패턴들에 관한 연구는 여러 기업들이 복잡한 비즈니스 프로세스를 문제없이 효율적으로 상호운용할 수 있도록 도움을 주는 기본적인 정보가 될 것이다. 또한 선행연구들에서 제안된 프레임워크들의 실질적 상호운용성 제공에 도움이 될 것이다.

논문접수일 : 2009년 01월 04일      게재확정일 : 2009년 03월 17일      교신저자 : 황유섭

## 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(Herman, 2002). 예를

들어 전세계 인터넷 관련 장비시장의 80% 이상을 장악하고 있는 시스코는 인터넷 네트워크 산업에서의 주도권을 유지하기 위해 마이크로소프트, 인텔 등 다양한 분야의 기업들과 전략적 제휴를 맺고 있다. 또한 시스코는 생산을 전량 아웃소싱에 의존하고 있는데 글로벌 네트워크를 통해 파트너, 공급자, 직원들을 직접 연결함으로써 보다 저렴하고 신속하게 고객이 원하는 제품과 서비스를 제공할 수 있게 되었다. 이러한 기업간 상호 협력 체제를 더욱 효율적으로 관리하며 신속하고 정확한 상호운용을 가능케하는 비즈니스 프로세스의 워크

\* 본 연구는 서울대학교 경영대학 경영연구소의 연구비 지원에 의해 수행되었음.

플로우 상호운용성이 기업들에게 요구되고 있는 실정이다.

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 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. 워크플로우 기술의 개발과 사용은 사람들의 작업을 라우팅하도록 지원하는 간단한 개념에서, 자원간에 작업을 수평적으로 또는 수직적으로 라우팅하는 것에 이르기까지 복잡화되고 있으며, 데이터가 프로세스들 간에 이동되고 이 데이터들이 시스템과 통합되면서 워크플로우는 기업 애플리케이션의 통합영역으로 확장되고 있기 때문에 기업간 비즈니스 프로세스의 통합은 매우 중요한 문제이다(Workflow Handbook 2003).

본 논문에서 주로 언급되는 몇 가지 용어에 대해 정리해 보면 다음과 같다. 우선 비즈니스 프로세스란 기업에서 제품을 생산하거나 서비스를 제공하기 위해 디자인된 연속적인 단계라고 정의할 수 있다(Rummler and Brache, 1995). 즉, 기업의 경영활동에서 목표를 달성해 가는 일련의 구조화된 단계를 의미한다. 워크플로우는 '컴퓨터에서 실행할 수 있도록 표현된 비즈니스 프로세스'로 정의되며, 워크플로우 관리시스템은 '이 워크플로우를 정의하고 정의된 워크플로우의 실행과 순서를 통제하며 프로세스 전체를 관리하는 소프트웨어 시스템'을 말한다(WfMC, 1999). 또한 WfMC에서는 워크플로우 상호운용성(Workflow Interoperability)을 '다수의 엔진간에 걸쳐서 수행되는 워크플

로우 프로세스 인스턴스들을 실행하기 위해 둘 또는 그 이상의 엔진들이 통신 및 상호운용할 수 있는 능력'이라고 정의하였다.

워크플로우 상호운용성은 현재 기업들에게 있어 중요하게 고려되고 있지만 아직까지는 해결되어야 할 문제들이 많다. 지금까지 몇몇 연구자와 기술자들에 의해 워크플로우간의 상호운용이 원활하게 진행될 수 있도록 연구가 진행되어 왔다. 하지만 대다수의 기존연구들은 워크플로우 상호운용성 제공에 필요한 기술적 문제의 해결을 위한 개념적 프레임워크에 중점을 두고 있다. 대표적인 노력으로 WfMC에서는 상호운용성 표준 및 WfXML 바인딩 등의 표준을 제정하여 워크플로우 관리 시스템의 상호운용성 확보에 노력을 가하고 있다. 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. 정확한 모델링은 시스템을 현재 또는 원하는 모습으로 가시화 시켜주며 시스템을 구축하는 기본형태를 제공하여 준다. 또한 상호운용시 모델링이 잘못되어 있다면 각 워크플로우 시스템이 서로 분산되어 있는 환경이기 때문에 수정에 많은 노력과 비용이 소모된다.

이에 따라 본 연구에서는 조직간 비즈니스 프로세스의 워크플로우 상호운용에 대한 개념적 접근을 시도한다. 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. 제 2장에서는 관련 연구로 워크플로우의 상호운용성을 제공하는 개념적인 프레임워크들을 분류하고 워크플로우 모델링 기법 중 본 논문에서 사용하고 있는 페트리 넷(Petri Nets)에 대해 알아보았다. 제 3장에서는 단일 워크플로우 패턴과 단일 워크플로우의 충돌패턴을 종합하였다. 제 4장에서는 상호운용성 패턴을 정리하였으며 또한 기존의 연구에는 없는 워크플로우 상호운용성 나타날 수 있는 충돌패턴들을 제안하였다. 마지막으로 제 5장에서는 결론과 본 논문의 의의와 향후 연구를 제시하였다.

## 2. 이론적 고찰

### 2.1 기존 연구

워크플로우의 상호운용성을 제공하기 위하여 연구자들은 개념적 프레임워크들을 제안하였다. 기존의 연구들은 크게 세 가지 방향으로 분류될 수 있다. 첫 번째, 비즈니스 프로세스 관리와 데이터베이스 이론을 기반으로 한 연구들이다. van der Aalst와 Kumar는 워크플로우 상호운용성을 위한 순차적(sequential) 또는 병렬적(parallel) 라우팅 기법을 XML기반으로 한 워크플로우 의미 전달 표현법으로 표현하였다(van der Aalst and Kumar, 2003). Kumar와 Wainer는 기업간의 WfMS에서의 충돌 처리를 위한 조정과 통제의 수단으로 메타모델과 프로세스 설명을 활용하여 설명하였다(Kumar and Wainer, 2005). Chiu et al.은 연합 데이터베이스 시스템(federated database systems)의 '뷰(view)' 개념을 이용하여 기업간 협업을 위한 WfMS의 상호운용성을 제공하는 프레임워크를 제안하였다. 연합 데이터베이스 시스템은 이질적인 여러 개의 데이터베이스 시스템의 합집합

(i.e., federated view)을 만들어 여러 개의 데이터베이스들의 상호운용성을 가능하게 한다. 이와 유사하게 Chiu et al.은 가상 연합 워크플로우 뷰를 만들어 이질적인 여러 개의 워크플로우들의 상호운용성을 가능하게 하는 방법을 제안하였다(Chiu et al., 2004).

두 번째, 웹 서비스 기술을 기반으로 한 연구들이다. 웹 서비스 기술은 기존 비즈니스 프로세스 관리(business process management)의 가장 큰 한계점이었던 외부 응용프로그램과의 통합문제에 대한 해결책으로 대두됨으로써, 워크플로우 연구자들로부터 많은 관심을 받고 있다(Chen et al., 2005). 특히, 웹 서비스 조합(Composition)에 대한 연구에서 워크플로우 상호운용성과 관련하여 활발하게 연구되고 있다. Hamadi와 Benatallah는 워크플로우 모델 작성에 널리 사용되고 있는 페트리 넷을 이용하여 웹 서비스 조합에 필요한 형식의 미론(formal semantics)을 개발하였다(Hamadi and Benatallah, 2003). Shen et al.은 XML 기반 워크플로우 프로세스 명세언어(예, XPDL, BPEL)와 웹 서비스 온톨로지(예, Ontology Web Languages for Services, OWL-S)를 결합하여 웹 서비스 조합의 개념적 프레임워크를 제안하였다(Shen et al., 2006).

마지막으로 에이전트(agent) 기술을 적용한 연구들이다. 에이전트 기술은 분산환경에 있는 지식이나 정보, 그리고 기존 시스템의 상호운용성 제공을 위하여 활용되어 왔다(Jennings et al., 2000). 에이전트 기술을 활용한 WfMS 연구는 에이전트 기반 WfMS 접근방식(Agent-based WfMS Approach)과 에이전트 지원 WfMS 접근방식(Agent-supported WfMS Approach)으로 분류될 수 있다. 에이전트 기반 WfMS 접근방식은 워크플로우의 관리에 필요한 모든 기능들을 에이전트에 구현

하는 반면, 에이전트 지원 WfMS 접근방식은 기존의 워크플로우 시스템 위에 에이전트 층을 만들어 에이전트들이 워크플로우 관리자가 처리하던 장황한 업무를 대신 처리함으로써 워크플로우 관리자는 보다 중요한 업무에만 집중할 수 있게 한다. 대표적인 에이전트 기반 WfMS 접근방식은 Jennings et al.이 개발한 ADEPT 시스템이 있다(Jennings et al., 2000). Biegus와 Branki가 개발한 InDiA 시스템과 Gronemann et al.이 개발한 MOK-ASSIN 시스템 등이 에이전트 지원 WfMS 접근방식의 대표적인 연구이다(Biegus and Branki, 2004; Gronemann et al., 1999).

위에서 정리하였듯이 워크플로우 상호운용성의 필요에 따라 여러 기관 및 연구자들이 상호운용성 문제를 해결하고자 노력해 왔지만, 지금까지의 노력들은 대부분 비즈니스 프로세스 상호운용시 발생하는 기술적인 문제의 해결을 위한 개념적 프레임워크를 개발하는데 중점을 두어 왔다는 것을 기존의 연구에서 찾을 수 있었다.

## 2.2 워크플로우 모델링-페트리 넷트(Petri Nets)

워크플로우를 분석하기 위해서는 조직의 비즈니스 프로세스의 동적인 구조를 정확히 표현하는 워크플로우 모델을 작성하여야 한다. 지금까지 워크플로우를 모델링하는데 주로 사용된 도구로는 IDEF3(Mayer et al., 1995), ARIS EPC(Sheer, 1999), 페트리 넷트(van der Aalst, 1998) 등을 뽑을 수 있다. 이 중에서도 본 연구에서는 Carl Adam Petri (1962)에 의해 고안된 페트리 넷트를 이용하여 워크플로우 시스템을 분석하고 모델링할 것이다. 페트리 넷트를 이용한 워크플로우 모델링은 많은 장점을 가지고 있다(van der Aalst, 1998; Ellis and

Nutt, 1993). 페트리 넷트는 비즈니스 프로세스를 정확하고 용이하게 모형화할 수 있으며 논리적 특성의 규명을 가능하게 한다. 뿐만 아니라 프로세스를 알아보기 쉽게 도식화할 수 있으며, 수리적 성능 분석도 수행할 수 있다.

페트리 넷트는 트랜지션(transition)과 플레이스(place)라는 두 개의 노드로 구성된 이진 그래프의 형태이다(Murata, 1989). 페트리 넷트에서는 일반적으로 태스크(task)를 트랜지션으로 본다. 트랜지션이 발생하기 위해서는 시스템의 여러 조건이 만족되어야 된다. 이러한 조건에 대한 정보가 플레이스에 저장된다. 페트리 넷트에서 어떤 플레이스는 한 트랜지션의 입력 조건이 되고, 어떤 플레이스는 한 트랜지션의 출력 조건이 된다. 그리고 트랜지션과 플레이스 사이의 상호 연관관계를 아크(arc)로 표현한다. 이러한 관계를 통하여 페트리 넷트는 다음과 같이 정의될 수 있다.

페트리 넷트는 다음의 플레이스, 트랜지션, 아크의 튜플(tuple)로 구성된다(Murata, 1989).

$$(P, T, A)$$

여기서,

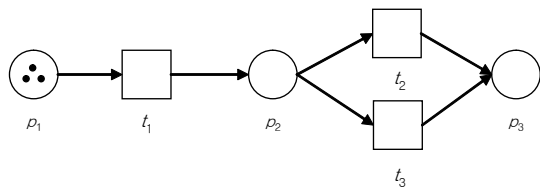
- $P$  : 플레이스의 유한집합;
- $T$  : 트랜지션의 유한집합,  $(P \cap T = \emptyset)$ ;
- $A \subseteq (T \times P) \cup (P \times T)$ 는 아크의 집합이다;
- $A_i$  : 입력 접속 행렬(Input Incidence Matrix) :  $(T \times P) \rightarrow \{0, 1, 2, 3, \dots\}$ ;
- $A_o$  : 출력 접속 행렬(Output Incidence Matrix) :  $(P \times T) \rightarrow \{0, 1, 2, 3, \dots\}$ .

플레이스 집합은  $P = \{p_1, p_2, \dots, p_n\}$ 으로 표현하고, 트랜지션 집합은  $T = \{t_1, t_2, \dots, t_m\}$ 으로 표현한다.  $A_i$ 는 플레이스로 들어오는 트랜지션의 연결 정보를 나타내며, 이를 입력접속행렬(Input Incidence Matrix)이라고 부르고, 각 원소의 기호는  $a(t, p)$ 로 쓴다.  $A_o$ 는 플레이스에서 나가는 트랜지

선 연결정보를 나타내고 이를 출력접속행렬(Output Incidence Matrix)이라 하며 각 원소의 기호는  $a(p, t)$ 로 표현한다. 여기서  $p \in P, t \in T$  이다.  $a(t, p) = a(p, t) = 0$ 은 플레이스  $p$ 와 트랜지션  $t$  사이에 입·출력 관계가 없다는 것을 의미한다.  $a(t, p)$ 가 0이 아닐 때에는 트랜지션  $t$ 가 플레이스  $p$ 의 입력이 되고,  $a(p, t)$ 가 0이 아닐 때에는 플레이스  $p$ 가 트랜지션  $t$ 의 입력이 됨을 의미한다. 플레이스와 트랜지션의 입·출력 관계는 아래의 기호를 사용하여 표현한다.

- $\bullet p = \{t \mid a(t, p) \neq 0, t \in T\}, p \bullet = \{t \mid a(p, t) \neq 0, t \in T\}$
- $\bullet t = \{p \mid a(p, t) \neq 0, p \in P\}, t \bullet = \{p \mid a(t, p) \neq 0, p \in P\}$

$\bullet p$ 는 플레이스  $p$ 의 입력집합이고,  $p \bullet$ 는 플레이스  $p$ 의 출력집합이라 부른다. 그리고  $\bullet t$ 는 트랜지션  $t$ 의 입력집합이고,  $t \bullet$ 는 트랜지션  $t$ 의 출력집합이라 부른다. 페트리 넷에서는 아래의 <그림 1>과 같이 플레이스는 타원으로 트랜지션은 사각형으로 표현하며, 아크는 플레이스와 타원 사이에 화살표로 표현된다. 또한 <그림 1>에서  $p_1$ 을 보면 세 개의 검정색 점들이 있다. 이것들은 토큰(token)이라 부르며, 플레이스는 토큰을 담고 있을 수 있다. 페트리 넷은 이산현상 시스템의 동작을 표현하기 위하여 '토큰'의 개념을 도입한다. 토큰은 각 플레이스에 부여된 상태의 값이다. 플레이스는 시스템의 이벤트를 발생시키기 위한 조건 정보



<그림 1> 페트리 넷 예

를 담고 있고 그 조건 정보가 토큰의 개수로 표현된다. 그러므로 각 플레이스에 부여된 토큰 개수의 정보로 시스템의 상태를 나타낸다. 각 토큰은 하나의 플레이스에 고정되어 있지 않고 상태의 흐름에 따라 위치가 변화된다.

### 3. 단일(single) 워크플로우<sup>1)</sup>

일반적으로 시스템 디자인에서 패턴(pattern)이란 어떠한 비임의적인(non-arbitrary) 상황에서 계속적으로 반복을 하는 추상적인 것들에 대한 명확한 형태를 말한다(Riehle and Zullighoven, 1996). 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.

워크플로우 분야에서도 패턴의 중요성을 인식하고 이에 대한 몇몇 연구가 진행되어 왔다(van der Aalst et al., 2003; Dumas et al., 2005; Barros and Varas, 2004). 지금까지 진행되어온 워크플로우 패턴에 대한 대표적인 연구로 van der Aalst et al.(2003)의 연구를 뽑을 수 있다. van der Aalst et al.(2003)은 워크플로우에서 발생할 수 있는 여러 가지 패턴을 제안하였다. 그들이 정리한 패턴은 순차(sequential), 분기(split), 병합(join) 등 총 21가지의 패턴을 제시하였다. 그러나 그들의 연구에서는 본 연구에서 고려하고자 하는 조직간 비즈니스

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

프로세스 상호운용시 발생할 수 있는 충돌적 패턴들에 대해서는 언급하지 않고 있다. 이에 따라 본 연구에서는 워크플로우간 상호운용시 발생할 수 있는 충돌패턴들을 제시하고자 한다.

지금까지 비록 많은 연구가 되어오지는 않았지만 몇몇 연구자들에 의해 워크플로우간 상호운용과 그에 따라 발생할 수 있는 패턴들에 대한 연구가 이루어져왔다. 워크플로우 관리 시스템의 기술 표준화를 위한 국제 기구인 WfMC에서는 워크플로우간 상호운용성에 대한 참고모델만을 제시하였을 뿐 각 패턴들에 대한 자세한 언급이 없었다. Casati et al.(1996)은 워크플로우의 통합 관점에서 상호운용성을 다루었다. 그들의 연구를 보면 여러 워크플로우가 상호운용할 때 발생할 수 있는 속성들을 정리하였다. 상호운용의 속성을 조건 의존적 실행과 동일한 선행조건을 가지고 있는 태스크로 나누었고, 프로세스 인스턴스 상호운용을 협력(cooperation), 경쟁(competition), 방해(interference) 관계로 분류하였으며, 워크플로우의 종료를 고려한 속성들로 충분(sufficence), 기아(starvation) 그리고 데드락(deadlock) 상태의 세 가지 경우를 제시하였다(제 4.1절 참고). 하지만 그들의 논문에서는 충돌상황에 대한 구체적인 패턴들을 제시하여 주지는 못하고 있다. 또한 van der Aalst (2000)의 연구에서도 워크플로우 상호운용성 패턴들을 다루고 있지만 체계적으로 패턴들을 정리해 주지 못하고 있다. 다음 절에서는 단일 워크플로우의 패턴들과 단일 워크플로우 충돌패턴을 정리하겠다.

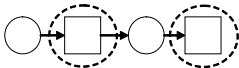
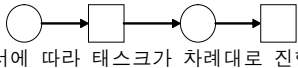
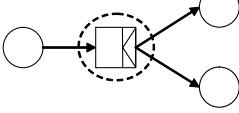
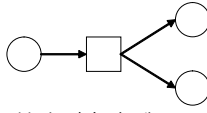
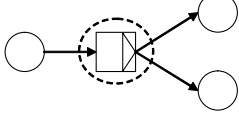
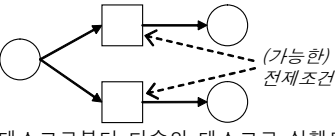
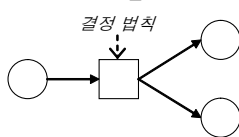
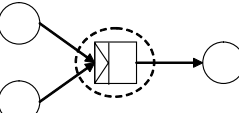
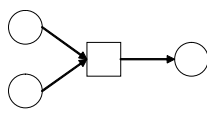
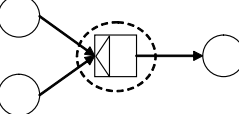
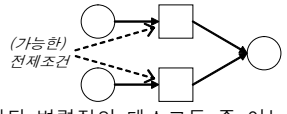
### 3.1 단일 워크플로우 운영패턴

van der Aalst et al.(2003)이 제안한 단일 워크플로우의 기본적인 운영패턴들을 정리하면 <그림

2>와 같다. 대표적으로 순차(Sequential), 분기(Split), 병합(Join)이 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 <그림 2>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 분기점에서의 입력은 하나이지만 출력이 다수인 경우, 병렬적으로 실행되는 다음 태스크들이 동시적으로 실행될 수도 있으며 순서적으로 실행될 수도 있다. 논리합-분기는 의사 결정이나 데이터에 의해서 뒤에 실행될 병렬적인 태스크 중 하나만 선택하는 경우를 말한다. 논리합-분기시 실행될 병렬적인 태스크의 결정은 <그림 2>의 “논리합-분기”에서 보여주듯이 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.

패턴 유형	표기(Notation)	의미
순차(Sequential)		 순서에 따라 태스크가 차례대로 진행됨
분기(Split)	논리곱-분기 (AND-Split) 	 하나의 태스크로부터 다수의 태스크로 분리되어 실행되는 병렬적인 태스크들이 동시에 또는 순차적으로 실행됨
	논리합-분기 (OR-Split) 	 하나의 태스크로부터 다수의 태스크로 실행되는 병렬적인 태스크들 중 전제조건이 만족된 태스크만 실행됨  또는 결정 법칙  하나의 태스크로부터 다수의 태스크로 실행되는 병렬적인 태스크들 중 결정법칙이 만족되는 경우로 실행됨
병합(Join)	논리곱-병합 (AND-Join) 	 이전에 실행된 병렬적인 태스크들 모두가 다음 진행될 하나의 태스크로 모이게 됨
	논리합-병합 (OR-Join) 	 이전에 실행된 병렬적인 태스크들 중 어느 하나만 다음 진행될 하나의 태스크로 모이게 됨

<그림 2> 단일 워크플로우 운영패턴

### 3.2 단일 워크플로우 충돌패턴

워크플로우의 잘못된 모델링은 정상적인 프로세스를 수행하지 못하게 한다. <그림 3>의 예시를 통해 단일 워크플로우상에서 잘못된 모델링으로 어

떤 문제가 발생하는지를 살펴보자(van der Aalst and Hee, 2002). 이 프로세스는 보험금 청구에 대한 과정 중 일부를 보여주고 있다. 보험금 청구에 대한 요구가 들어오는 것으로 프로세스가 시작(start)되며 이를 수락(accept)한다. 이후 두 가지

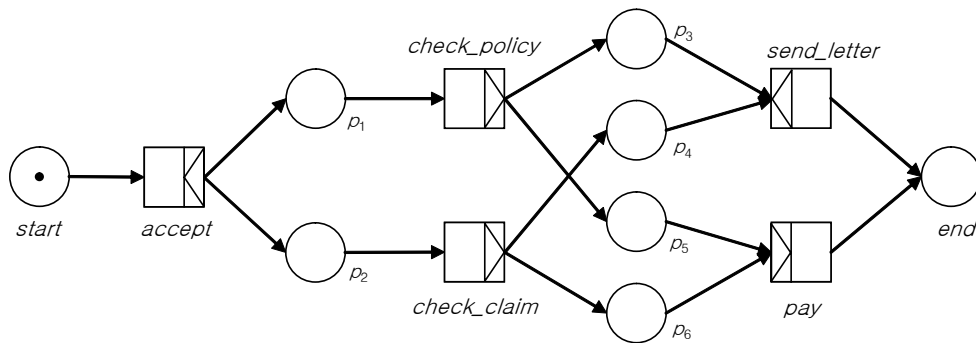
사항에 대해 병렬적으로 검토한다. 보험금 방침을 검토(*check\_policy*)하며 고객이 요구한 보험금 청구내용을 검토(*check\_claim*)한다. 보험금 방침을 검토한 후 방침에 부합하지 않으면 보험금을 지급할 수 없다는 내용을 통보(*send\_letter*)해주며 또는 보험금 청구내용을 검토한 후 적절하지 않은 경우 거절되었다는 내용을 통보(*send\_letter*)해준다. 보험금 지불의 경우 회사의 방침에도 맞으며 청구 내용을 검토한 결과도 적합하면 보험금을 지급(*pay*)해 준다. 즉, 두 가지 조건이 모두 만족되어야 보험금이 지불된다. 이로써 프로세스는 종료(*end*)된다.

본 프로세스를 얼핏 보면 문제가 없어 보이지만, 잘못된 모델링으로 인하여 여러 상황에서 프로세스가 정상적으로 수행되지 못할 수 있다. 만약 *check\_policy*가  $p_5$ 로 실행되고 *check\_claim*이  $p_6$ 으로 진행된다면 보험금이 지급(*pay*)되는 것으로 프로세스가 종료된다. 이 경우는 논리곱-분기로 모델링된 *accept*가 병렬적으로 분기하여 *check\_policy*와 *check\_claim*이 실행된다. 이때 논리곱-분기로 모델링된 *pay*는 위의 두 개의 실행결과를 받아야 진행되는데, *check\_policy*와 *check\_claim*의 실행 결과를 모두 받음으로 본 프로세스는 문제없이 잘 진행된다. 즉, 이 경우는 <그림 3>의 프로세스

가 문제없이 정상적으로 진행되는 최적의 시나리오이다.

하지만 만약 *check\_policy*가  $p_3$ 으로 진행된 후 *check\_claim*이  $p_4$ 로 진행된다면  $p_3$ 에 의해 *send\_letter*가 한번 실행되고 또다시  $p_4$ 에 의해 실행되므로 작업이 두 번이나 실행되며, 최종적으로 *end*에는 두 개의 토큰이 남겨지게 된다. 즉, 고객은 보험금 지급이 거절되었다는 통보를 두 번이나 중복해서 받게 된다. 또한 만약 *check\_policy*가  $p_3$ 으로 실행되고 *check\_claim*이  $p_6$ 으로 진행된다면 *send\_letter*는 한 번만 실행되지만,  $p_6$ 에는 한 개의 토큰이 계속 남아 있을 것이다. 왜냐하면 *pay*는 논리곱-병합으로 모델링 되어있어  $p_5$ 의 값을 계속 기다리고 있는 상태이기 때문이다. 즉, 이 경우에는 고객에게 거절통보를 보냈지만 계속 보험금을 지급하기를 기다리고 있다. 만약 *check\_policy*가  $p_5$ 로, *check\_claim*이  $p_4$ 로 진행될 경우에도 마찬가지로 오류가 발생하게 된다.

위의 예를 통하여 단일 워크플로우에서 잘못된 모델링으로 인하여 발생할 수 있는 상황을 살펴 보았다. 지금부터는 van der Aalst et al.(2003)이 제시한 내용을 바탕으로 충돌패턴을 <그림 4>를 통해 단일 워크플로우에서 발생 가능한 6가지의 경우로 분류하여 설명하도록 하겠다.



<그림 3> 잘못된 모델링의 예



분 류	충돌패턴
(1) 입력(input) 또는 출력(output)이 없는 태스크	
(2) 데드 태스크(dead task)	
(3) 데드락(deadlock)	
(4) 종료된 후에도 토큰이 남아있는 경우	
(5) 종료된 후에도 계속 실행되고 있는 경우	
(6) 라이브락(livelock)	

<그림 4> 단일 워크플로우에서 발생 가능한 충돌패턴

### 3.2.1 입력(input) 또는 출력(output)이 없는 태스크

어떤 태스크에 입력조건이 없다면 그 태스크는 언제 수행해야 될지 모르는 상황에 빠진다. 또한 어떤 태스크가 출력조건이 없다면 프로세스는 성

공적으로 완료하지 못할 것이다. 예를 들어 <그림 4>의 (1)을 보면  $t_d$ 는 아무런 입력조건을 가지고 있지 않아 실행될 수 없는 상태에 있으며,  $t_e$ 는 출력조건을 가지고 있지 않아 더 이상 프로세스를 진행할 수 없다.

### 3.2.2 데드 태스크(dead task)

프로세스 중 한번도 실행되지 못하는 태스크가 있을 경우 이를 데드 태스크라 한다. 예를 들어 <그림 4>의 (2)를 보면  $t_b$ 는 절대로 수행될 수 없다. 왜냐하면  $t_a$ 는 논리합-분기로 모델링되어 있으며  $t_b$ 는 논리곱-병합으로 모델링되어 있어 실행조건을 계속 만족하지 못해 프로세스 중 결코 실행될 수 없는 상태에 빠지게 된다. 이처럼 정의해 놓은 태스크가 프로세스 상에서 한번도 실행되지 못하는 경우는 잘못된 모델링으로 지양해야 된다.

### 3.2.3 데드락(deadlock)

데드락이란 태스크가 수행하지 못할 어떤 특정 태스크를 기다리고 있는 상태를 말한다. 예를 들어 <그림 4>의 (3)을 보면  $t_a$ 는 논리합-분기이기 때문에 한쪽으로 밖에 실행될 수 없는데  $t_c$ 는 논리곱-병합으로 양쪽의 값을 받아야 진행될 수 있다. 그러므로  $t_c$ 는 계속 두 개의 실행결과값을 무한정 기다리는 상태에 빠지게 되므로 프로세스가 정상적으로 진행되지 못한다.

### 3.2.4 종료된 후에도 토큰이 남아있는 경우

토큰이  $end$ 로 도달하였다면 이는 프로세스가 종료된 것으로 다른 플레이스에는 토큰이 남아 있으면 안 된다. 예를 들어 <그림 4>의 (4)의 경우 논리곱-분기로 모델링된  $t_b$ 가 먼저  $end$ 로 실행될 경우  $t_c$  이전의 플레이스에는 토큰이 실행되지 못한 채 남겨져 있어 완벽히 종료되지 못한 프로세스라고 볼 수 있다.

### 3.2.5 종료된 후에도 계속 실행되고 있는 경우

만약 프로세스가 종료에 도달하였다면, 더 이상

다른 태스크들이 수행되어서는 안 된다. 종료된 후에도 계속 실행되고 있는 경우로는 <그림 4>의 (5)와 같이 논리곱-분기로 모델링된  $t_a$ 에 의해  $end$ 에 도달된 후에도,  $t_b$ 와  $t_c$ 가 계속 진행을 하고 있는 경우이다. 이것은 프로세스가 종료된 후에도 프로세스의 일부가 진행되고 있는 경우로 디자인이 잘못된 경우이다.

### 3.2.6 라이브락(livelock)

라이브락이란 프로세스가 종료된 후에도 일부가 빠져나가지 못하고 끊임없이 순환하고 있는 경우를 말한다. 예를 들어 <그림 4>의 (6)를 보면  $t_c$ 가 논리곱-분기로 모델링되어 있어  $end$ 와  $t_b$ 로 실행되는데,  $end$ 로 태스크가 진행되어 프로세스가 종료된 후에도 여전히  $t_b$ 와  $t_c$ 는 실행하고 있으며 결코 벗어날 수 없는 상태에 빠져 동일한 작업을 반복하고 있다. 라이브락은 앞서 살펴본 종료된 후에도 계속 실행되고 있는 경우 중 특수한 경우로, 무한반복에 의해 프로세스가 벗어나지 못하고 계속 실행되고 있다는 특징이 있다.

위에서 정리한 6가지 패턴들은 상호 배타적인 관계가 아니며, 서로 중복되어 발생할 수 있다. 예를 들어 데드락일 경우 데드 태스크나 종료된 후에도 토큰이 남아있는 경우가 생길 수 있으며, 또한 라이브락이 발생해 종료된 후에도 계속 실행될 경우가 생길 수도 있다.

## 4. 다중(Multi) 워크플로우의 상호운용성

과거에는 업무의 효율성을 높이기 위해 비즈니스 프로세스를 도입하였다면 최근에는 기업간 교류 및 협력을 위하여 비즈니스 프로세스의 상호운

용을 필요로 하고 있다. 본 장에서는 기존의 상호 운용성 패턴들을 살펴본 후, 여러 워크플로우가 상호 운용할 경우 발생할 수 있는 운영패턴들 중 충돌이 발생할 수 있는 패턴을 제안하겠다.

#### 4.1 다중 워크플로우 운영패턴

Casti et al.(1996)와 van der Aalst(2000)의 연구에서는 기업간 상호운용되는 워크플로우들에 대한 패턴들을 연구하였다. 이들의 연구에서 정리된 다중 워크플로우 운영패턴들은 다음과 같다.

##### 4.1.1 조건 의존적 실행

두 개의 상호운용하는 워크플로우  $wf_a$ 와  $wf_b$ 가 있다고 하자. 이 패턴은 두 개의 워크플로우가 상호운용시  $wf_a \cdot t_1$ 과  $wf_b \cdot t_2$ 사이의 실행 순서의 관계에서 태스크  $wf_b \cdot t_2$ 가 임의의 태스크로부터 실행되기 이전에 태스크  $wf_a \cdot t_1$ 이 임의의 태스크로부터의 실행이 종료되어야 한다. 즉, 조건 의존적 실행이란 한 워크플로우의 선행 태스크가 먼저 실행되어야만 다른 워크플로우의 다음 태스크가 실행할 수 있는 경우이다. 이러한 관계를 실행하기 위해서는 조건의존적인 관계를 가진 태스크를 서로 연결해주어야 된다. 예를 들어 고객으로부터 주문을 받아 물품을 배송해 줄 경우, 재고를 보유하고 있는 회사로부터 재고 확인을 거쳐야 물품을 배송해 줄 수 있다. 즉, 재고관리회사의 태스크인 '재고확인'이 먼저 실행되어야 물품배송업체의 다음 작업인 '물품배송'이 가능해진다.

##### 4.1.2 가상의 동등하게 실행(pseudo-simultaneous execution) 되는 태스크

두 개의 워크플로우가 각기 서로 다른 프로세스를 정의해 놓았지만, 두 개의 태스크가 동등하게

만족되어야만 다음 태스크가 실행되는 경우가 있다. Casti et al.(1996)은 이를 가상의 동등하게 실행(pseudo-simultaneous execution) 되는 태스크라고 정의하였다. 이들은 워크플로우에서 각각 실행되고 있지만, 상호의존적으로 함께 실행되는 태스크로 워크플로우 통합시 하나의 태스크로 묶을 필요가 있다. 예를 들어 어떠한 물건을 수리할 경우 두 전문업체가 관여하여 수리를 해야 될 경우가 있다. 수리 시 전문성을 요구하는 비행기의 경우 전자정비업체와 기계정비업체의 전문가들의 지식이 필요한데, 각 기업에서는 수리라는 업무를 각자 기업의 프로세스에 따로 정의해 놓고 교류를 하고 있다. 이와 같은 경우에는 두 기업에서 수리라는 업무를 각각 정의해 놓고 있지만, 이들은 상호의존적인 업무로 결국은 동등하게 실행되어야 다음 태스크가 진행 될 수 있다.

##### 4.1.3 프로세스 인스턴스시 발생 가능한 운영패턴

Casati et al.(1996)은 서로 다른 워크플로우들간의 상호작용에 대한 관계를 협력(cooperation), 경쟁(competition) 그리고 방해(interference)관계로 분류하였다. 첫째, 협력관계란 두 개의 워크플로우들 간의 상호작용이 예상될 뿐만 아니라 바람직할 경우에 발생된다. 예를 들면, 워크플로우 A가 어떤 데이터를 생성하면 이것을 다른 시점에서 워크플로우 B가 A의 데이터를 사용하는 경우이다. 예를 들어 A라는 기업에서 어떠한 부품을 생산하면 그것을 B라는 기업에서 조립하여 완성된 제품을 생산하는 경우이다.

둘째로 경쟁관계는 두 개 이상의 워크플로우들 간의 상호작용이 예상은 되지만 바람직하지 않은 경우에 발생된다. 이 경우 워크플로우들 간에 제한된 자원을 두고 경쟁이 일어난다. 예를 들면, 워크

플로우 A와 B가 동시에 같은 자원을 필요로 할 경우가 발생할 수 있다. 만약 이때 자원이 한 개뿐이라면 태스크 A와 B는 서로 경쟁이 필요한 관계이다. 예를 들어 두 회사가 A라는 부품업체에서 생산한 부품을 받아 사용하는데 A회사에서 제공하는 부품이 한 개밖에 없을 경우 두 회사는 이 부품을 받기 위해 서로 경쟁해야 된다.

마지막으로 방해관계란 두 개의 워크플로우들 간의 상호작용이 예상도 되지 않고 바람직하지도 않는 상황에서 잘못된 디자인으로 인해 서로의 진행을 방해하는 경우를 말한다. 이 경우, 워크플로우 A와 워크플로우 B는 독립적으로는 정상적으로 디자인 되어 있지만, 두 워크플로우 간의 동시 진행 협조는 부정확하게 디자인 된 경우이다.

#### 4.1.4 종료를 고려한 속성

이외에도 종료를 고려한 속성들로 충분, 기아, 데드락 관계를 정의하였다. 한 워크플로우의 모든 태스크가 상호운용하는 다른 편의 워크플로우 프로세스상에 존재할 경우 이를 충분관계라고 한다. 기아상태란 다른 편의 사용이 끝나기만을 기다리며 무한히 실행을 대기하고 있는 경우를 말한다. 워크플로우  $wf_a$ 와  $wf_b$ 가 모두  $t_1$ 라는 태스크의 실행을 기다리고 있는데,  $wf_a$ 가 계속 독점하여 사용하고 있어  $wf_b$ 는  $t_1$ 의 실행을 계속 기다리는 상황에 빠지게 되므로  $wf_b$ 는 기아상태에 빠졌다고 할 수 있다. 마지막으로 워크플로우간의 데드락이란 하나 또는 그 이상의 태스크가 수행할 수 없는 어떤 특정 태스크를 기다리고 있는 상태를 말한다. 즉, 두 개 이상의 워크플로우의 상호운용시 어떠한 태스크가 실행되지 못할 태스크를 무한히 기다리는 교착상태에 빠져 다음 프로세스를 진행하지 못하는 상태를 일컫는다. 일반적으로 기아상태를 데드락의 일부로 본다.

van der Aalst(2000)의 연구에서는 워크플로우가 상호운용할 경우 발생할 수 있는 충돌패턴으로 데드락과 라이브락의 경우를 보여주고 있으나 구체적인 패턴들을 제시하지 못하고 있다. 국내외 선행 연구들에 대한 포괄적 문헌연구를 통해 다중 워크플로우들 간의 충돌패턴에 중점을 둔 연구는 찾을 수 없었다. 다음 절에서는 다중 워크플로우 상호운용 시 발생할 수 있는 충돌패턴들을 제시하도록 하겠다.

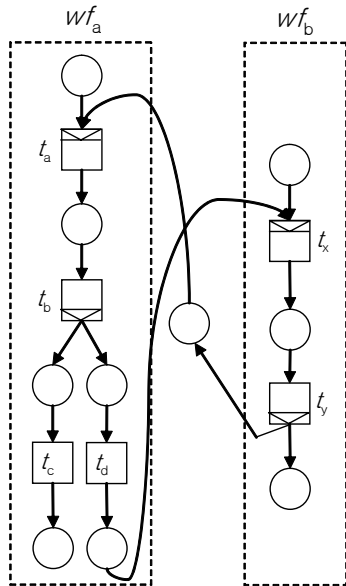
## 4.2 다중 워크플로우 충돌패턴

여러 개의 워크플로우가 상호운용할 경우 충돌이 발생할 수 있는데 본 절에서는 이러한 충돌패턴을 제안하겠다. 상호운용하기 전 독립적으로 운용되었던 각각의 워크플로우에서는 문제없이 진행되었지만, 기업간 인수·합병, 해외지사와의 교류, 부서간의 협력, 협력업체와의 교류 등 조직의 필요에 의한 여러 이유로 상호운용하면서 비즈니스 프로세스가 원활하게 진행되지 못하는 경우가 발생할 수 있다. 원만하지 못한 비즈니스 프로세스의 흐름은 기업의 업무효율성을 저하시키는 원인으로 지양되어야 된다. 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  $wf_a$ 와  $wf_b$ 가 상호운용하기 이전 독립적으로 운용되었던 워크플로우 자체에는 어떠한 충돌도 없이 무결하다는 가정하에 워크플로우간 상호운용 시 발생할 수 있는 충돌패턴들을 분석한다.

### 4.2.1 순차충돌패턴

워크플로우 관리 시스템은 기업의 원활한 업무

흐름을 위하여 일정한 작업절차를 정해 놓는다. 만약 상호운용성을 높이기 위해 여러 워크플로우를 모델링 할 경우 기업의 전체적인 비즈니스 프로세스의 순차적 흐름을 고려하지 않는다면 충돌이 발생할 수 있다. 순차충돌패턴이란 개별 워크플로우가 독립적으로는 이상이 없었지만, 상호운용을 함에 따라 잘못된 모델링으로 인해 진행순서가 엇갈려 올바른 진행을 하지 못하게 되는 상황을 말한다. <그림 5>의 경우를 보면  $wf_b \cdot t_x$ 는  $wf_a \cdot t_d$ 가 실행되어야지만 실행할 수 있다. 하지만  $wf_a \cdot t_a$ 는 상대방의 워크플로우의  $wf_b \cdot t_y$ 가 실행되어야지만 논리곱-병합으로 모델링된  $wf_b \cdot t_a$ 가 실행되어 작업을 수행할 수 있게 모델링되어 있다. 즉,  $wf_a$ 와  $wf_b$ 는 서로의 태스크가 실행되기만을 기다리고 있는 상황으로 충돌이 발생하였다고 할 수 있다.



<그림 5> 순차충돌패턴

이러한 상황은 실제 조직들이 여러 개의 비즈니스 프로세스를 함께 운용할 경우 자주 발생할 수

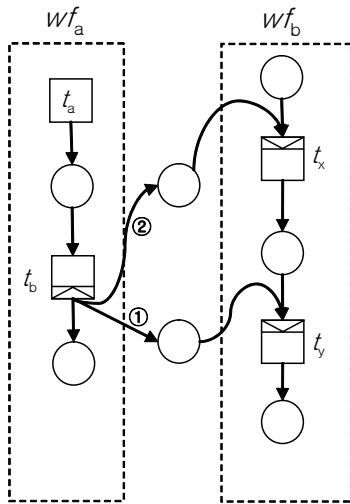
있는 경우이다. 예를 들어 두 회사가 사업을 협의할 경우 한편에서는 사업계획서를 받아야 사업에 대한 예산을 작성하여 제출해 준다고 생각하고 있으며, 다른 한편의 회사에서는 사업에 대한 예산을 알아야 사업계획서를 작성하여 보내줄 수 있다고 생각하는 경우가 있을 수 있다. 이러한 경우에는 서로의 작업이 진행되기만을 기다리고 있어 충돌이 발생할 수 있다.

#### 4.2.2 논리곱-분기(AND-Split)시 순차충돌패턴

van der Aalst and Hee(2002)에 따르면 워크플로우를 모델링할 때 주의할 점으로 논리곱-분기와 논리곱-병합 그리고 논리합-분기와 논리합-병합을 유의하여 사용해야 한다고 했다. 비즈니스 프로세스의 순차적인 진행에 의해 발생할 수 있는 또 다른 충돌패턴으로는 논리곱-분기시 진행순서에 따른 충돌이다. 논리곱-분기시의 순차충돌패턴이란 논리곱-분기를 하는 특정 태스크의 분기 순서가 상대방 워크플로우 진행순서와 일치하지 않아 충돌이 생기는 경우를 말한다. 이 경우에는 상대방의 워크플로우가 논리곱-병합인 경우와 상대방의 워크플로우가 논리합-병합인 경우의 두 가지 경우로 나누어 생각해 볼 수 있다.

<그림 6>는 상대방의 워크플로우가 논리곱-병합인 경우 논리곱-분기시 진행순서에 따른 충돌이다. 어떠한 태스크가 논리곱-분기를 할 경우 다음 번에 실행될 수 있는 병행 태스크 중 분기의 결정은 전제조건이 만족 여부 또는 결정 법칙에 의해서 임의의 각기 다른 순서로 진행될 수 있다. 태스크  $wf_a \cdot t_b$ 의 실행순서가 <그림 6>에서 표시해 놓은 것과 같이 ①과 ②의 순서로 실행될 경우 상호 운용하는 상대방의 워크플로우인  $wf_b$ 의 진행순서와 일치되지 않아 충돌이 발생할 수 있다. 만약  $wf_b \cdot t_y$ 로 먼저 실행된 후  $wf_b \cdot t_x$ 로 실행된다면  $wf_b$

의 워크플로우의 진행순서는  $wf_b \cdot t_x$ 가 실행된 후  $wf_b \cdot t_y$ 가 실행되는데,  $wf_b \cdot t_x$ 가 논리곱-병합으로 모델링되어 있으므로 실행되지 못하고 있으며  $wf_b \cdot t_y$  또한 수행하지 못하는 상태가 된다. 즉, 태스크들이 수행할 수 없는 특정 태스크를 계속 기다리고 있는 상태에 빠지게 된다.

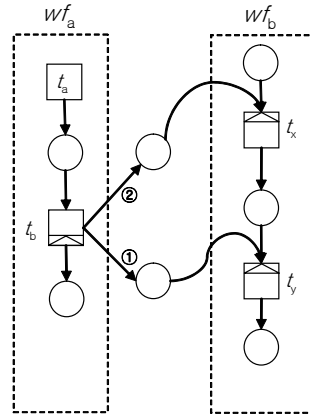


<그림 6> 논리곱-분기(AND-Split)시 순차충돌패턴-상대편의 워크플로우가 논리곱-병합인 경우

예를 들어  $wf_b$ 는 신입 사원을 채용하고자 하는 A라는 회사의 워크플로우이며,  $wf_a$ 는 A회사의 채용에 관련된 작업들을 보조하기 위해 아웃소싱을 한 회사의 워크플로우이다.  $wf_a$ 에서는 A회사의 지원자들의 서류를 검토하고 거짓으로 기재한 사항이 있는지를 체크하며, 각 지원자들의 신용상태가 불량인지를 체크하여 A회사에게 전달해 준다. 이 경우 만약 <그림 6>와 같이 모델링되어 있다면,  $wf_a$ 에서 지원자들의 서류심사의 거짓여부의 결과 보낸 후 신용상태에 대한 결과를 보내야지만  $wf_b$ 가 정상적으로 실행되나 만약 반대의 순서로 실행된다면 잘못된 모델링으로 인하여  $wf_b$ 는 서류심사

의 거짓여부에 대한 결과값을 계속 기다리고 있어 다음 작업을 진행하지 못한 채 무작정 기다리고 있는 상황에 빠지게 된다.

<그림 7>은 앞의 경우와 마찬가지로 논리곱-분기시 진행순서에 따른 충돌이 발생하는 경우이나 다른 점은 상대편의 워크플로우가 논리합-병합으로 모델링되어 있다는 것이다. 이 상황도 <그림 6>과 마찬가지로 논리곱-분기를 하는 태스크  $wf_a \cdot t_b$ 의 실행순서가 <그림 7>에서 표시해 놓은 것과 같이 ①과 ②의 순서에 따라 실행될 경우 상호 운용하는 상대편의 워크플로우인  $wf_b$ 의 진행순서와 일치되지 않아 충돌이 발생할 수 있다.  $wf_b \cdot t_y$ 로 먼저 실행된 후  $wf_b \cdot t_x$ 로 실행된다면  $wf_b \cdot t_x$ 가 논리합-병합으로 디자인되어 있어 앞서 살펴본 <그림 6>과 같이 계속적으로 기다리는 상황은 발생하지는 않지만 ①이 진행된 후 ②가 진행된다면  $wf_b \cdot t_y$ 가 두 번 중복 실행되는 문제가 발생한다.



<그림 7> 논리곱-분기(AND-Split)시 순차충돌패턴-상대편의 워크플로우가 논리합-병합인 경우

예를 들어  $wf_a$ 는 개인 신상정보를 체크하는 부서의 워크플로우이며  $wf_b$ 는 신용카드를 발급해주는 부서의 워크플로우라고 하자. 이 경우  $wf_a$ 에서

는 신용카드 신청자가 정기적인 수입원이 있는지 직장상태를 체크하고, 또한 신용불량자인지를 체크하여  $wf_b$ 에게 정보를 전달한다.  $wf_b$ 에서는 이러한 정보를 받아 신용카드를 발급( $wf_b \cdot t_y$ )해주는데 만약 <그림 7>과 같이 모델링되어 있다면, 논리곱-분기로 모델링 되어있는  $wf_a \cdot t_b$ 가 ①로 분기하여 신용불량자가 아니라는 정보를 받고 신용카드를 발급해주고 ②에 의해 또다시 안정적인 직장상태라는 정보를 받고 신용카드를 중복해서 발급해주게 된다.

#### 4.2.3 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

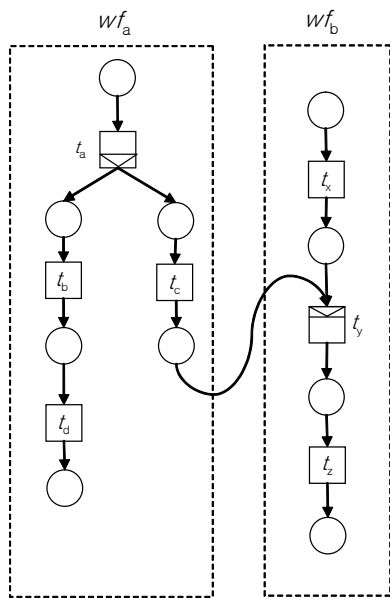
중 한 개만 실행되는데, 다른 한 쪽의 워크플로우에서는 논리곱-병합으로 되어있어 반드시 두 개의 태스크의 값을 받아야 실행할 수 있는 경우이다. 이 때에는 논리곱-병합되는 태스크가 선행 태스크의 모든 값을 받아야지만 실행될 수 있으므로 비즈니스 프로세스가 원활히 진행되지 못한다.

<그림 8>을 보면 논리합-분기하는 태스크인  $wf_a \cdot t_a$ 가  $wf_a \cdot t_c$ 로 분기하는 경우에는 문제가 발생하지 않지만, 만약  $wf_a \cdot t_b$ 로 분기한다면 상호운용하는 상대편 워크플로우의 태스크  $wf_b \cdot t_y$ 는 실행할 수 없다. 왜냐하면  $wf_b \cdot t_y$ 는 논리곱-병합으로 모델링되어있어  $wf_b \cdot t_x$ 에서 보낸 값과  $wf_a \cdot t_c$ 가 실행된 값을 모두 받아야지만 실행될 수 있기 때문이다.

이러한 충돌은 특히 긴박한 상황이나, 빠른 업무 처리를 해야 될 때 문제가 발생할 수 있다. 예를 들어,  $wf_a$ 와  $wf_b$ 는 상호 협력하는 회사로  $wf_b$ 는  $wf_a$ 에서 긴급복구차량을 지원받아 공사를 수행한다.  $wf_b$ 는  $wf_a$ 에서 차량을 지원 받아야만 작업을 할 수 있는데, 만약 차량 지원 승인 결과를 보내는 작업( $wf_a \cdot t_c$ )이 <그림 8>과 같이 논리합-분기로 모델링 되어있다면  $wf_b \cdot t_y$ 는 상대편으로부터 승인 결과가 오기만을 기다리고만 있어야 된다. 비즈니스 프로세스는 효율성을 높이기 위해 활용되는데 이와 같은 모델링은 효율적인 업무처리를 지원하지 못하므로 올바른 모델링 방법이 아니다.

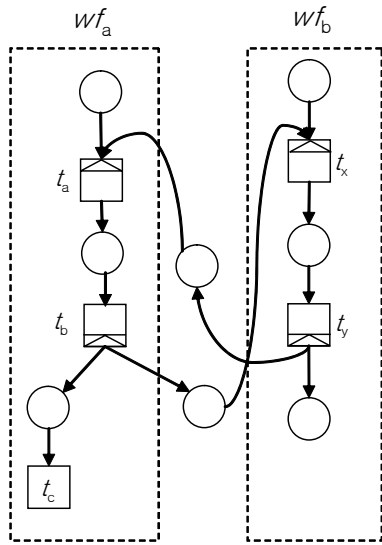
#### 4.2.4 순환충돌패턴

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 9>과 같은 모델링으로 인해 동일한 프로세스가 무한히



<그림 8> 논리합-분기와 논리곱-병합에 따른 충돌패턴

반복할 수 있다.  $wf_a \cdot t_b$ 는 논리곱-분기로 모델링되어 있어  $wf_a \cdot t_c$ 와  $wf_b \cdot t_x$ 로 실행될 것이다. 이 경우  $wf_b \cdot t_x$ 에 의해  $wf_b \cdot t_y$ 와  $wf_a \cdot t_a$ 가 실행되고 또 다시 논리곱-분기로 모델링된  $wf_a \cdot t_b$ 에 의해  $wf_b \cdot t_x$ 로 실행되어 계속 같은 프로세스가 순환적으로 빠져나가지 못하고 계속 실행하게 된다.



<그림 9> 순환충돌패턴

예를 들어 A라는 회사는 제품을 스스로 개발할 수 없어 B라는 업체에서 제품 개발과정 중 일부를 도움 받아 생산하고 있다.  $wf_a$ 는 A회사의 프로세스이며  $wf_b$ 는 B회사의 프로세스이다. A회사는 제품설계 도안을 작성하여 이 중 일부를 B업체에게 전달하고 나머지 부분은 계속 자신의 회사에서 작업을 진행한다. B회사는 도안을 받아 자신의 회사에서 실제로 제품을 개발할 수 있는지를 체크한 후, 시행할 수 없다고 판단되면 다시 A회사에게 도안을 재작성해 줄 것을 요구한다. 이 경우 <그림 9>과 같이 모델링되어 있다면 도안을 수정요청하고 다시 수정된 도안을 업체 B에게 보내는 작업을 계속 반복적으로 진행하게 될 것이다.

## 5. 결론 및 향후연구방향

많은 기업들은 기업간 상호운용의 중요성을 인식하고 있다. 본 연구는 워크플로우가 상호운용시 발생할 수 있는 여러 가지 충돌패턴들을 제시하였다. 상호운용 시 발생할 수 있는 충돌패턴으로 크게 순차충돌, 논리곱-분기(AND-Split)시 순차충돌, 논리합-분기와 논리곱-병합시 충돌, 그리고 순환충돌패턴 등이 있다.

본 연구에서는 비즈니스 프로세스의 정확한 모델링을 위해 페트리 넷을 사용하여 비즈니스 프로세스를 모델링하여 더욱 정교하고 분석적으로 프로세스의 상호운용을 분석하였다. 지금까지 진행되어온 워크플로우 상호운용성에 관한 선행연구들은 대부분 개념적 프레임워크 제안에 중점이 되어 왔으며, 단지 몇몇 연구에서만 상호운용에 따른 패턴이 다루어졌다. 특히 다중 워크플로우 상호운용에 따른 충돌패턴에 관한 연구는 찾기 힘들다.

만약 조직들이 사전에 올바른 모델링을 하지 않고 각 기업의 워크플로우를 통합하여 상호운용을 하고자 한다면 많은 문제가 발생할 수 있다. 그러므로 본 연구를 통해 이러한 문제 상황을 미리 방지할 수 있도록 다양한 충돌패턴들을 제시한다. 이미 구축된 시스템을 사용하다가 디자인 단계에서 잘못된 점이 발견된다면 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 기업의 워크플로우를 통합할 수 있는 중요한 정보가 될 것이다.

본 연구의 향후 과제로는 다음과 같은 것이 있다. 여기서 제시한 다섯 가지 패턴 이외의 다양한 충돌패턴들을 생각해 볼 필요가 있을 것이다. 워크



플로우 분석의 중요한 분야 중 하나인 데이터 흐름 측면까지 고려하여 비즈니스 프로세스 상호운용시 발생할 수 있는 충돌패턴을 연구할 필요가 있을 것이다. 예를 들어, 워크플로우가 실행하는 동안 데이터의 중복, 결여 등으로 인하여 충돌이 발생할 경우를 고려해 볼 수 있다. 또한 페트리 넷의 분석적인 장점을 활용하여 도달성 분석(reachability analysis)를 수행하면 좋을 것이다. 도달성 분석은 프로세스가 시작 태스크에서 마지막 태스크까지 도달 가능한지를 검증하는 분석법으로 도달될 수 없는 태스크가 있다면 워크플로우의 목적을 제대로 달성할 수 없는 경우로 수정이 필요하다는 것을 알 수 있다. 그리고 충돌패턴을 사전에 예방할 수 있도록 다양한 충돌패턴을 자동적으로 발견할 수 있는 알고리즘의 개발도 향후 중요한 연구가 될 것이다.

## 참고문헌

- Barros, O. and S. Varas, "Frameworks derived from business process patterns", Technical Report 56, Industrial Engineering Department, University of Chile, 2004, (Available at <http://www.obarros.cl>).
- Biegus, L. and C. Branki, "InDiA : a framework for workflow interoperability support by means of multi-agent systems", *Engineering Applications of Artificial Intelligence*, Vol.17, No.4(October 2004), 825~839.
- Casati, F., S. Ceri, B. Pernici, and G. Pozzi, "Semantic workflow interoperability", *In Proceeding of the 5th Conference on Extending Database Technology (EDBT)*, Lecture Notes In Computer Science, Vol.1057(1996), 443~462.
- Chen, M., D. Zhang, and L. Zhou, "Empowering collaborative commerce with Web services enabled business process management systems", *Decision Support Systems*, Vol.43 (2007), 530~546.
- 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.
- Ellis, C. A. and G. J. Nutt, "Modeling and Enactment of Workflow Systems", *Application and Theory of Petri Nets*, LNCS 691, Springer-Verlag, (1993), 1~16.
- Gronemann, B., G. Joeris, S. Scheil, M. Steinfort, and H. Wache, "Supporting cross-organizational engineering processes by distributed collaborative workflow management-The MOKASSIN approach", *Proceedings of the Sencod International Conference on concurrent multidisciplinary engineering (CME '99)*, Bremen, Germany, 1999.
- Hamadi, R. and B. Benatallah, "A Petri Net-based Model for Web Service Composition", *Proceedings of the 14th Australasian Database Conf. Database Technologies*, ACM Press, 2003, 191~200.
- Herman, J., "Making Collaborative Commerce Happen", (Available at <http://www.bcr.com/bcsmag/2002/10/p18.asp>), 2002.
- Jennings, N., T. Norman, P. Faratin, P. O'Brien, and B. Odgers, "Autonomous agents for business process management", *Applied Artificial Intelligence*, Vol.14, No.2(2000), 145~189.
- Kumar, A. and J. Wainer, "Meta workflows as a control and coordination mechanism for exception handling in workflow systems", *Decision Support Systems*, Vol.40(2005), 89

- ~105.
- Mayer, R., C. Menzel, M. Painter, B. Perakath, P. de Witte and T. Blinn "Information Integration For Concurrent Engineering (IICE) - IDEF3 Process Description Capture Method Report", Technical Report, (Available at [http://www.idef.com/pdf/idef3\\_fn.pdf](http://www.idef.com/pdf/idef3_fn.pdf)), September 1995.
- Murata, T., "Petri nets-properties, analysis, and applications", *Proceedings of the IEEE*, Vol.77, No.4(April 1989), 541~580.
- Petri, C. A., "Communication with automata", *In Schriften des IIM Nr. 3*, Institut für Instrumentelle Mathematik, Bonn, (1962), 16~27.
- Raghu, T. S., B. Jayaraman, and H. R. Rao "Toward an Integration of Agent- and Activity-Centric Approaches in Organizational Process Modeling : Incorporating Incentive Mechanisms", *Information Systems Research*, Vol.15, No.4(December 2004), 316~335.
- Muyllé, S., A. Basu, "Online support for business process by electronic intermediaries", *Decision Support Systems*, Article in Press, 2008.
- Riehle, D., and H. Zullighoven, "Understanding and Using. Patterns in Software Development", *Theory and Practice of Object System*, Vol.2, No.1(1996), 3~13.
- Rummler, G. A. and A. P. Brache, *Improving Performance : How to Manage the White Space on the Organization Chart*, Jossey-Bass, San Francisco, CA. Shingo, S. 1995.
- Sheer, A., "ARIS-Business Process. Modeling", Springer, Berlin-Heidelberg, 1999.
- Shen, J., G. Grossmann, Y. Yang, M. Stumptner, and M. Schrefl, "Analysis of business process integration in Web service context", *Future Generation Computer Systems*, Vol.23, No.3 (March 2007), 283~294.
- van der Aalst, W. M. P. and K. M. van Hee, *Workflow Management : Models, Methods, and Systems*, MIT Press, 2002.
- van der Aalst, W. M. P., A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns", *Distributed and Parallel Databases*, Vol.14 No.1(July 2003), 5~51.
- 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), 67~75.
- Van der Aalst, W. M. P. and A. Kumar, "XML Based Schema Definition for Support of Inter-organizational Workflow", *Information Systems Research*, Vol.14, No.1(2003), 23~46.
- 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), 21~66.
- WfMC, *Workflow Management Coalition Terminology and Glossary*, Workflow Management Coalition., WfMC-TC-1011, Feb. 1999.
- Workflow Handbook 2003*, Published in association with the Workflow Management Coalition (WfMC). Edited by Layna Fischer, 2003.
- Workflow Management Coalition, "Interface 1-Process Definition Interchange V 1.1", WfMC Specification, 1999.

Abstract

## Conflict Pattern Analysis for Heterogeneous Workflow Interoperability Among Interorganizational Business Processes

Jinsoo Park<sup>\*</sup> · Boyoun Kim<sup>\*\*</sup> · Yousub Hwang<sup>\*\*\*</sup>

The recent development of information technology has been envisioned as the next technological wave and is expected to play an important role in diminishing boundaries between business organizations. Enterprises have recently resulted in a surge in workflow management, making business processes sharable among different business organizations. To make heterogeneous workflows operational, it is crucial that workflow management systems provide efficient tools for an environment supporting interoperability of business processes among different business organizations. As the potential of workflow management is becoming widely recognized, the demand for an integrated framework that facilitates interoperability among heterogeneous workflows is concomitantly growing.

Despite the large body of work in the area of workflow management, few efforts are directed towards identifying conflict patterns for heterogeneous workflow interoperability of inter-organizational business processes. In this paper, we summarize state of the art research trends in workflow management research area and identify conflict patterns for heterogeneous workflows. We believe that this is one of the first attempts to conceptualize conflict patterns that exist on inter-organizational business processes. This paper opens up a novel avenue for workflow management research by supplementing the existing conceptual frameworks for workflow management.

**Key Words** : Workflow Management Systems, Interoperability, Business Process, Conflict Patterns

---

\* Graduate school of business, Seoul National University

\*\* College of Business Administration, Seoul National University

\*\*\* Department of Business Administration, College of Business and Economics, University of Seoul

## 저자 소개



박진수

The University of Arizona에서 경영정보시스템을 전공하여 경영학 박사를 취득했으며, University of Minnesota의 Carlson School of Management에서 조교수, 고려대학교 경영대학에서 조교수를 역임했다. 현재 서울대학교 경영전문대학원/경영대학에 부교수로 재직 중이다. 현재 국제저널인 Journal of Database Management와 International Journal of Principles and Applications in Information Science and Technology의 편집위원으로 활동하고 있으며 MIS Quarterly, IEEE Transactions on Knowledge and Data Engineering (TKDE), IEEE Computer, ACM Transactions on Information Systems (TOIS), Information Systems Frontiers, Communications of the AIS, Journal of Global Information Technology Management (JGITM), International Journal of Electronic Business, 경영정보학연구 등 국내외 우수 전문학술지에 다수의 논문을 게재하였다. 주요 관심분야는 온톨로지, 정보 시스템 통합, 지식 공유, 에이전트, 시맨틱 모델링, 웹 정보시스템 등이 있다.



김보연

덕성여자대학교 전산학 학사(2003년), 아주대학교 대학원 경영정보학 석사(2005년)를 취득한 후, 서울대학교 경영학과에서 경영정보학을 전공으로 박사과정을 하고 있다. 주요 연구분야는 비즈니스 프로세스 메타 모델, 워크플로우 시스템, 온톨로지, 시맨틱 웹 등이다.



황유섭

The University of Arizona에서 경영정보시스템을 전공하여 학사, 석사, 그리고 경영학 박사학위를 취득하였다. 현재 서울시립대학교 경상대학 경영학부에 조교수로 재직 중이다. 미국 NASA와 Raytheon의 Hydrology Resource Management Project에 참여하였으며 Photogrammetric Engineering and Remote Sensing과 ER 학회지, Information Systems Review 등에 논문을 게재하였다. 주요 관심분야는 service-oriented computing과 artificial neural network의 활용 방안 연구 등이다.