IE Interfaces
Vol. 25, No. 1, pp. 79-86, March 2012.

협업 프로세스의 독립적 변경 보장 규칙 개발

김애경 · 정재윤†

경희대학교 산업경영공학과

Integrity Checking Rules for Independent Changes of Collaboration Processes

Aekyung Kim · Jae-Yoon Jung

Department of Industrial and Management Systems Engineering, Kyung Hee University

Traditional business process management systems provide verification tools of process models to deploy and automate the models. However, there are not so many studies on how to maintain systematically collaborative process models such as supply chain processes when companies are willing to change and update the collaborative process models. In this paper, we analyze change patterns of collaborative processes and declare 19 change patterns. In addition, we apply the change patterns to the process interoperability patterns in order to identify the change problems in case of independent process changes of collaborative processes. As a result, we devise an independency checking algorithm of process changes in collaborative processes.

Keyword: workflow patterns, process change patterns, verification, collaborative process, integrity checking rule

1. 서 론

비즈니스 프로세스 자동화 및 관리는 다양한 기업정보시스템에서 중추적인 역할을 수행하고 있다(Dumas, 2005). 제조협업 분야에서도 마찬가지로, 동시공학적 설계와 같은 다양한 협업 과정에 여러 기업 또는 조직이 참여하여 협력하는 과정을 협업 프로세스로 정의하고 자동화함으로써 수행하고 있다.

본 연구에서는 제조협업을 수행하기 위하여 설계된 프로세스가 시간이 지나 업데이트되는 과정에서 발생할 수 있는 오류를 확인하고 방지하고자 하는 목적으로 수행되었다. 특히, 협업 프로세스 설계에서 하나의 기업이 독자적으로 프로세스 일부를 변경하여 반영하고자 할 때, 상대 기업 또는 전체 프로세스에 영향이나 지장을 초래하는지 확인하고자 하는 요구에서 출발하였다.

비즈니스 프로세스 관리(BPM : Business Process Management)는 정보시스템을 이용하여 자동화하는 것을 가정하기 때문에 설계의 무결성을 보장하는 것은 더욱 중요한 연구 이슈이다(van der Aalst, 1999). 여러 기업이 참여하는 협업 프로세스에서 설계의 무결성을 보장하는 것은 기존의 단일 기업의 프로세스에서 설계의 무결성을 보장하는 것과 차별화된다. 기존의 많은 BPM 시스템에서 프로세스 개선 방향을 모색하기 위한 다양한 분석도구를 제공하고 있지만, 기업 내부가 아니라 공급사슬과 같은 협업 프로세스를 공동으로 설계하고 운영할 때에 어떻게 체계적으로 관리할 것인지에 대한 연구는 상대적으로 미약한 상황이다. 협업 프로세스의 유형을 분석하고 협업 프로세스의 유효성을 자동으로 검증할 수 있는 기법이 제공된다

이 논문은 지식경제부에서 수행한 i매뉴팩처링(한국형 제조혁신)사업과, 2011년도 교육과학기술부 재원으로 수행한 한국연구재단의 기초연구사업(No. 20110003560)의 지원을 받았습니다.
†연락저자 : 정재윤 교수, 446-701 경기도 용인시 기흥구 서천동 1번지 경희대학교 산업경영공학과,
Fax : 031-203-4004, E-mail : jyjung@khu.ac.kr
투고일(2011년 12월 09일), 심사일(1차 : 2012년 01월 20일), 게재확정일(2012년 02월 06일).

면, 제조협업과 같이 점점 증가하는 협업 환경에서 비즈니스 환경의 변화에 빠르고 유연하게 대응할 수 있을 것이다.

본 연구에서는 협업 프로세스에서 발생할 수 있는 프로세스 패턴을 정리하고, 이를 바탕으로 협업 프로세스에서 초래할 수 있는 문제를 분석하였다. 또한, 이러한 문제를 협업 프로세스 변경 패턴에 기초하여 확인하는 규칙을 정리하였다.

먼저, 협업 프로세스에서 발생할 수 있는 변경 유형을 '액티비티 분할', '액티비티 결합', '액티비티 삭제', '액티비티 삽입', '구조 변경'의 다섯 가지로 구분하고, 이들을 세분화하여 총 19가지 변경패턴을 제시하였다. 또한 19가지 변경패턴들이 가지는 문제점을 세 가지 유형(미실행, 역실행, 실행누락)으로 분류하고 이를 분석하여 협업 프로세스에서 독립적 변경을 보장할 수 있는 규칙과 알고리즘을 개발하였다.

협업에 참여하는 한 기업에서 업무 처리 과정의 개선, 업무 환경의 변화 등으로 인한 프로세스 변경이 요구될 때, 본 논문에서 제시한 협업 프로세스 변경 패턴에 기반하여, 상대 기업에게 영향을 미치는지 판정할 수 있도록 함으로써 프로세스 유지보수 및 관리가 용이해진다. 또한 독립적 프로세스 변경이 협업 기업에 영향을 미칠 가능성을 사용자에게 경고함으로써 문제 발생 가능성에 신속히 대응하여 사후 업무 진행시에 문제가 발생하지 않도록 방지한다. 서비스 지향 아키텍처(SOA : Service-Oriented Architecture)와 같이 정보시스템 통합 및 코레오그래피(Choreography)의 필요성이 증대함에 따라 이와 같은 유연한 협업 프로세스 설계 분석에 대한 방법론이 효과적으로 적용될 수 있을 것으로 기대된다.

예를 들어 제조업체 간에 협업 프로세스를 정의하고 수행하고 있을 때, 한 기업에서 자신들의 프로세스 중 일부를 변경하고자 한다고 가정해 보자. 해당 기업은 프로세스 모델링 도구를 이용하여 기존에 설계된 프로세스 중 일부를 수정한 후에 상대 기업 또는 전체 협업 프로세스에 문제를 야기시키지는 않는지 확인해야 한다. 이 때, 본 연구에서 제시한 협업 프로세스의 독립적 변경 보장 규칙을 사용하여 프로세스 변경의 허용여부를 판단할 수 있다. 만약, 독립적 변경 불가능으로 판단되면 프로세스 설계자에게 어떠한 오류가 발생할 우려가 있는지 주의를 줄 수 있다.

본 논문은 다음과 같이 구성된다. 먼저 제 2장에서 관련 연구를 설명한다. 제 3장에서 협업 프로세스의 독립적 변경을 분석하기 위하여 사용되는 프로세스 변경 패턴을 제시한다. 제 4장에서는 제시된 프로세스 변경 패턴에 따라 협업 프로세스의 독립적 변경 가능성을 분석하고 보장 규칙을 제시하였다. 제 5장에서 본 연구의 결론을 기술하였다.

2. 관련 연구

Gamma *et al*.(1995)은 객체지향 시스템에서의 반복적으로 발생하는 가장 최소한의 상호작용을 기술하는 23가지 디자인 패턴을 최초로 제시하였다. van der Aalst는 1990년대 후반까지 페트리넷(Petri nets)에 대한 연구를 워크플로우에 접목하면서 프로세스 설계에서 나타나는 디자인 패턴에 대해 관심을 가지게 되었다. 이후 ter Hofstede와 워크플로우 패턴에 대해 공동 연구를 위해 WPI(Workflow Patterns Initiative)를 발족하였다(WPI, 1999). 기존에 발표된 워크플로우 패턴에는 제어 흐름 패턴을 기본 제어 흐름 패턴, 고급 분기 및 동기화 패턴, 구조 패턴, 복수 인스턴스 패턴, 상태 기반패턴, 취소 패턴으로 구분하여 총 20개의 패턴을 제시하였다(van der Aalst *et al*., 2003).

김동수, 김민수(2010)는 동적인 환경에서 비즈니스 프로세스 개선 및 변화 과정을 효율적으로 지원할 수 있는 전체적인 프레임워크를 제시하였으며, 워크플로우 통제 패턴에 기초하여 비즈니스 프로세스의 재설계 과정에서 자주 나타나는 변화 유형을 찾아내어 이를 패턴으로 정리하였다. 프로세스 변화 패턴을 크게 세 가지 유형으로 구분하였으며, 액티비티 분할/변경 유형에는 5가지 변화 패턴, 액티비티 확대/삭제 유형에서 4가지 패턴, 액티비티 결합/변경 유형으로 7가지를 제시하였다. 패턴에 기반을 둔 비즈니스 프로세스의 변경은 구조적인 개선 작업을 보다 체계적으로 지원할 수 있기 때문에 전체 BPM의 라이프사이클 속에서 그 동안 상대적으로 연구가 취약했던 개선 단계를 보다 체계적으로 지원할 수 있다. 프로세스 변화 패턴에 기초하여 프로세스 변화를 관리함으로써 기업이 동적인 환경 변화와 프로세스 개선으로 인해 발생하는 다양한 프로세스 변화에 보다 쉽고 효율적으로 대응할 수 있다.

Jung *et al*.(2004)은 기업간 협업을 위한 비즈니스 프로세스 코레오그래피(Choreography) 연구에서 프로세스 상호운용성 패턴을 소개하였다. 비즈니스 프로세스 간에 발생할 수 있는 여러 가지 형태를 분석하여 여섯 가지 기본적인 상호운용성 패턴을 명시하였다. 이 기본패턴은 WfMC(2001)에서 제시한 세 가지 상호운용성 모델을 프로세스 관리 측면에서 세분화한 것이다. 먼저, 연결 모델(chained model)에서는 하나의 프로세스가 다른 프로세스의 시작이나 시행을 발생시킨 후에 그 프로세스와 별개로 실행된다. 이 모델을 연결 대체(CS : Chained Substitutive)와 연결 추가(CA : Chained Additive) 패턴으로 구분하였다. 두 번째로 중첩 모델(nested model)에서는 하나의 프로세스가 다른 프로세스를 호출한 후, 특정 지점에서 실행 결과를 반환 받는다. 이 모델은 중첩 동기(NS : Nested Synchronous), 중첩 대기(ND : Nested Deferred), 중첩 병렬(NP : Nested Parallel) 패턴으로 구분하였다. 마지막으로 동기화 모델(synchronized model)은 병렬 동기(PS: Parallel Synchronized)이다.

다음으로는 비즈니스 프로세스 무결성 검증에 관한 연구들이 있다. Liu(2010)은 소프트웨어 시스템 사용자들의 요구사항이 다양화됨에 따라 디자이너들이 시스템을 유지 보수할 때 자동으로 충돌을 감지해주는 CDADE(Conflict Detector in Activity Diagram Evolution) 시스템을 소개하였다. CDADE는 온톨로지, 메타데이타, 충돌 감지 규칙으로 구성되며 액티비티 다이어그램을 기반으로 한다. 사용자들의 새로운 요구사항들이 수집되

어 디자이너들이 이를 반영하여 변경하고자 할 때, CDADE은 온톨로지를 표현하는 시맨틱 관계에 따라 기존의 요구사항과 새로운 요구사항 간의 충돌을 자동으로 감지한다. Weber *et al.*(2008)은 프로세스 중심의 정보시스템에서 유연성 강화를 위한 변경 패턴과 변경 지원 특성들을 소개하였다. 프로세스 변화 지원에 관한 기존의 프로세스 관리 기술을 체계적으로 비교하기 위한 18가지의 변경패턴을 제시하였다. 또한 프로세스가 변경 되었을 때, 올바른 방법으로 수행되었는지 보장하는 7가지 변경 지원 특성들을 제시함으로써 사용자들이 프로세스 변경을 쉽게 할 수 있도록 도와준다.

본 연구에서는 제조협업 환경을 참고하여 김동수, 김민수(2010)의 프로세스 변경 패턴을 재정의한 후, Jung *et al.*(2004)의 협업 프로세스 상호운용성 패턴을 적용한다. 이 조합에서 발생할 수 있는 협업 프로세스 변경 패턴을 분석하여 독립적 변경을 보장하는 규칙을 도출한다.

3. 프로세스 변경 패턴 개발

본 연구에서는 먼저 단일 기업의 프로세스에서 발생할 수 있는 비즈니스 프로세스 변경 패턴(김동수, 김민수, 2010)을 재정의하였다. 기존 연구에서는 비즈니스 프로세스의 변화 관리의 측면에서 통제 흐름의 변화, 액티비티 및 관련 문서의 변경에 대해서도 고려하였지만, 본 연구에서는 협업 프로세스의 측면에서 필요한 특징으로 통제 흐름 및 메시지 흐름에 초점을 맞추어 프로세스 변경 유형을 <표 1>과 같이 크게 다섯 가지로 구분하였다.

먼저, '액티비티 분할' 유형은 기존 하나의 액티비티를 N개의 액티비티로 상세화(specialization)하여 변경범위가 단일 액티비티이다. '액티비티 결합' 유형은 기존 N개의 액티비티를 하나의 액티비티로 추상화(generalization)하므로 변경범위가 두 액티비티 이상이다. '액티비티 삭제' 유형은 기존 하나의 액티비티를 삭제하거나 우회시켜 변경범위가 단일 액티비티이다. '액티비티 삽입' 유형은 기존의 액티비티를 그대로 유지시키고 1~N개의 액티비티를 추가하여 변경범위가 두 액티비티 이상이다. 마지막으로 '구조 변경' 유형은 기존의 액티비티들이 유지되지만 흐름 구조가 변경되어 변경범위가 두 액티비티 이상이다.

나아가, 본 연구에서는 제시한 협업 프로세스 변경 유형에 대하여 발생할 수 있는 패턴을 세분화하여 총 19개의 패턴을 정의하였다. 아래에서는 다섯 가지 프로세스 변경 유형에 속하는 각 패턴에 대하여 설명한다.

3.1 액티비티 분할 유형

액티비티 분할(Activity Divide)은 기존의 한 액티비티를 여러 개의 액티비티로 상세하게 설계 변경하는 유형이다.

- 직렬분할(C1) : 단일 액티비티를 직렬로 연결된 두 개 이상의 액티비티로 분할하는 패턴이다.
- 병렬분할(C2) : 단일 액티비티를 병렬로 연결된 두 개 이상의 액티비티로 분할하는 패턴이다.
- 선택분할(C3) : 단일 액티비티를 선택적 분기(XOR, OR, M-out-of-N)로 연결된 두 개 이상의 액티비티로 분할하는 패턴이다.

3.2 액티비티 결합 유형

액티비티 결합(Activity Merge)은 기존의 여러 액티비티를 하나의 액티비티로 추상화하여 설계 변경하는 유형이다.

- 직렬결합(C4) : 두 개 이상의 직렬 액티비티들을 단일 액티비티로 결합하는 패턴이다.
- 병렬결합(C5) : 두 개 이상의 병렬 액티비티들을 단일 액티비티로 결합하는 패턴이다.
- 선택결합(C6) : 두 개 이상의 선택 액티비티들을 단일 액티

표 1. 프로세스 변경 유형별 특징과 해당 프로세스 변경 패턴

프로세스 변경 유형	설 명	activity 개수 변화		변경 범위	프로세스 변경 패턴
		변경 전	변경 후	activity	
액티비티 분할	기존의 한 액티비티를 여러 개의 액티비티로 상세하게 설계 변경	1	N	1	직렬분할(C1), 병렬분할(C2), 선택분할(C3)
액티비티 결합	기존의 여러 액티비티를 하나의 액티비티로 추상화하여 설계 변경	N	1	1+	직렬결합(C4), 병렬결합(C5), 선택결합(C6)
액티비티 삭제	기존의 액티비티가 불필요하게 되어 실행되지 않도록 설계 변경	1	0~1	1	완전삭제(C7), 부분삭제(C8), 완전대체(C9)
액티비티 삽입	기존의 액티비티를 그대로 유지한 채 새로운 액티비티를 추가하여 설계 변경	0	1~N	1+	직렬삽입(C10), 병렬삽입(C11), 선택삽입(C12)
구조 변경	기존의 액티비티들을 그대로 유지한 채 흐름 구조만을 변경하여 설계 변경	N	N	1+	순서교체(C13), 직렬의 병렬화(C14), 병렬의 직렬화(C15), 직렬의 선택화(C16), 선택의 직렬화(C17), 병렬의 선택화(C18), 선택의 병렬화(C19)

비티로 결합하는 패턴이다

3.3 액티비티 삭제 유형

액티비티 삭제(Activity Delete)는 기존의 액티비티가 불필요하게 되어 실행되지 않도록 설계 변경하는 유형이다.

- 완전삭제(C7) : 단일 액티비티를 삭제하는 패턴이다.
- 부분삭제(C8) : 단일 액티비티를 선택적으로 우회할 수 있도록 변경하는 패턴이다.
- 완전대체(C9) : 단일 액티비티를 내용이 다른 단일 액티비티로 대체하는 패턴이다.

3.4 액티비티 삽입 유형

액티비티 삽입(Activity Insert)은 기존의 액티비티를 그대로 유지한 채 새로운 액티비티를 추가하여 설계 변경하는 유형이다.

- 직렬삽입(C10) : 하나 이상의 액티비티를 직렬로 삽입하는 패턴이다.
- 병렬삽입(C11) : 하나 이상의 액티비티를 병렬로 삽입하는 패턴이다.
- 선택삽입(C12) : 하나 이상의 액티비티를 선택적으로 삽입하는 패턴이다.

3.5 구조 변경 유형

구조 변경(Structural Change)은 기존의 액티비티들을 그대로 유지한 채 흐름 구조만을 변경하여 설계 변경하는 유형이다.

- 순서교체(C13) : 기존 액티비티들의 순서 교체하는 패턴이다.
- 직렬의 병렬화(C14) : 직렬로 실행되던 여러 액티비티들을 병렬화하는 패턴이다.
- 병렬의 직렬화(C15) : 병렬로 실행되던 여러 액티비티들을 직렬화하는 패턴이다.
- 직렬의 선택화(C16) : 직렬로 실행되던 여러 액티비티들을 선택화하는 패턴이다.
- 선택의 직렬화(C17) : 선택적으로 실행되던 여러 액티비티들을 직렬화하는 시키는 패턴이다.
- 병렬의 선택화(C18) : 병렬로 실행되던 여러 액티비티들을 선택화하는 패턴이다.
- 선택의 병렬화(C19) : 선택적으로 실행되던 여러 액티비티들을 병렬화하는 패턴이다.

4. 협업 프로세스 독립적 변경 분석

4.1 독립적 변경 가능 패턴

본 연구에서 개발한 19가지 협업 프로세스 변경 패턴을 제 2 장에서 언급한 6가지 협업 프로세스 상호운용성 패턴에 적용하였을 때, 각 변경패턴들이 전체 협업 프로세스에 문제를 일으키지 않고 독립적으로 변경 가능한 지를 분석하였다. 그 결과 직렬결합, 병렬결합, 선택결합, 직렬삽입, 병렬삽입, 병렬의 직렬화, 선택의 직렬화, 선택의 병렬화 패턴을 독립적으로 변경이 가능한 패턴으로 확인되었다.

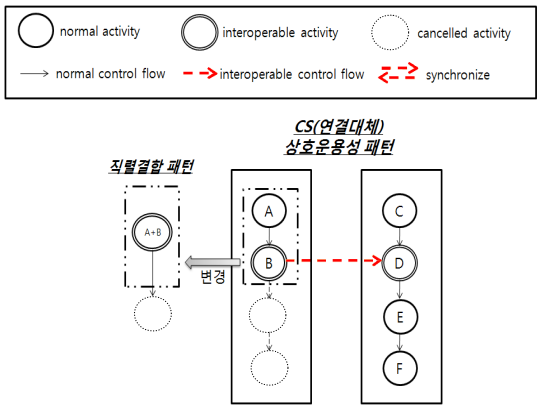

그림 1. 직렬결합 패턴(C4)

몇 가지 예를 살펴보면, <그림 1>과 같이 상호운용성 CS 패턴에서 모든 액티비티들이 직렬로 연결되어 있는데, 액티비티 B가 메시지를 송신하고, 액티비티 D는 이것을 수신한 후 다음 액티비티인 E가 진행된다. 이러한 CS 패턴에서 직렬로 연결된 액티비티 A와 B를 '직렬결합 패턴(C4)'으로 변경하여도 전체 협업 프로세스에 어떠한 문제도 일으키지 않는다. 나머지 모든 액티비티들에 적용했을 때에도 같은 결과를 볼 수 있다. 뿐만 아니라 C4는 나머지 6가지 상호운용성 패턴에 적용하여 변경했을 때에도 모두 독립적인 변경이 가능하다.

'병렬결합 패턴(C5)'과 '선택결합 패턴(C6)'도 어떤 상호운용성 패턴의 임의의 액티비티에 적용하여도 전체 협업 프로세스에 문제를 일으키지 않는다. <그림 2>와 <그림 3>을 보면 직렬 또는 병렬 또는 선택적으로 진행되던 액티비티들이 송/수신 메시지를 포함하더라도 '액티비티 결합 유형'의 변경 패턴들로 변경되면 기존의 기능들은 원래대로 실행되기 때문이다.

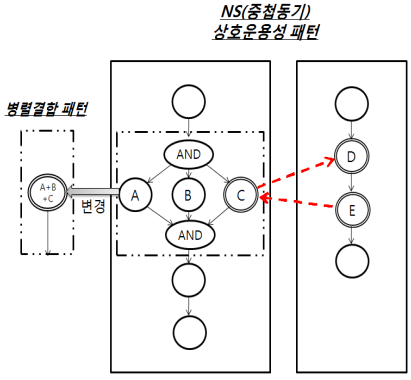

그림 2. 병렬결합 패턴(C5)

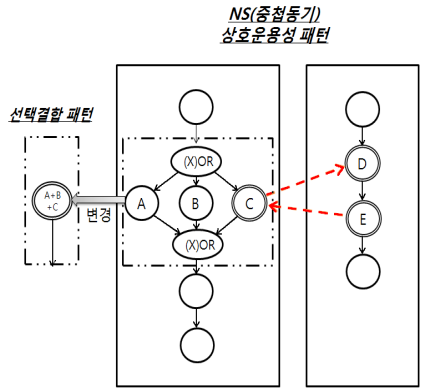

그림 3. 선택결합 패턴(C6)

'액티비티 삽입 유형'을 모든 상호운용성 패턴에 적용하여 변경해 보면 '직렬삽입 패턴(C10)'과 '병렬삽입 패턴(C11)'은 모든 상호운용성 패턴의 모든 액티비티들에 적용하여 변경하여도 전체 프로세스에 문제를 일으키지 않고 독립적으로 변경 가능함을 알 수 있다. 그러나 '선택삽입 패턴(C12)'은 일부 문제를 일으킨다. C12가 독립적 변경이 불가한 이유는 이후에 살펴보겠다.

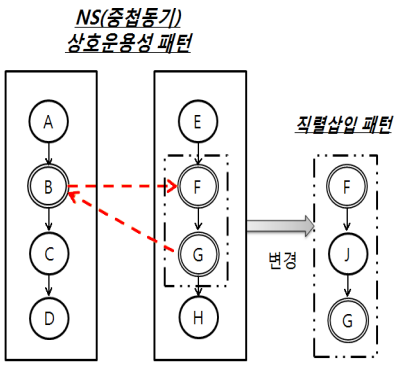

그림 4. 직렬삽입 패턴(C10)

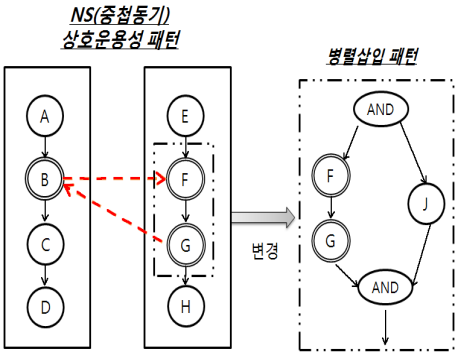

그림 5. 병렬삽입 패턴(C11)

'구조 변경 유형'의 '병렬의 직렬화 패턴(C15)', '선택의 직렬화 패턴(C17)', '선택의 병렬화 패턴(C19)'을 살펴보면, <그림 6>~<그림 8>과 같이 병렬 또는 선택적으로 실행되던 액티비티들을 직렬화 패턴으로 변경하여도 기존의 액티비티들이 모두 실행되기 때문에 어떠한 문제도 일으키지 않는 것을 알 수 있다. C19 또한 선택적으로 실행되던 액티비티들을 병렬화로 변경하여도 기존 액티비티들이 모두 실행되어 전체 협업 프로세스에 문제를 일으키지 않는 것을 확인할 수 있다. 나머지 변경 패턴들이 독립적으로 실행 불가한 이유는 이후에 살펴보겠다.

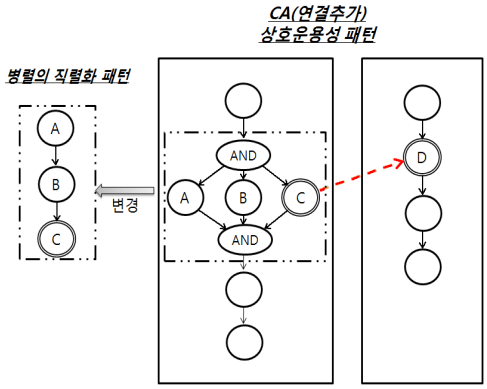

그림 6. 병렬의 직렬화 패턴(C15)

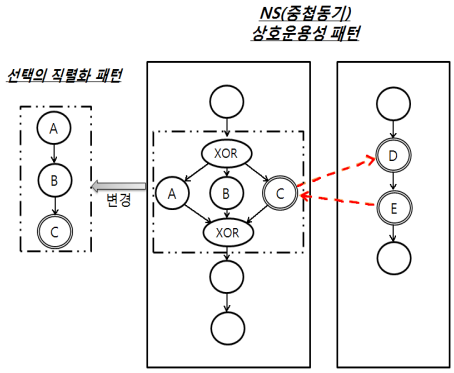

그림 7. 선택의 직렬화 패턴(C17)

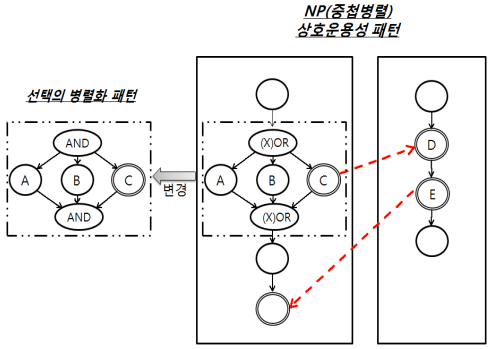

그림 8. 선택의 병렬화 패턴(C19)

4.2 독립적 변경 불가 패턴

제 4.1절에서 살펴본 패턴들 외의 나머지 변경 패턴들은 독

립적으로 변경이 불가한 패턴들이다. 나머지 변경 패턴들을 모든 상호운용성 패턴에 적용하여 변경했을 때 어떤 문제점들이 발생하는지 살펴보겠다.

상호운용성 패턴에 협업 프로세스 변경 패턴들을 적용하여 변경하였을 때 발생하는 문제점들은 크게 세 가지로 분류할 수 있다. 첫 번째는 기존에 실행되던 액티비티가 변경 후에는 실행되지 않을 가능성이 있는 '미실행(non-execution)' 문제, 두 번째는 기존 액티비티의 순서가 역으로 바뀌어 발생하는 '역실행(reverse execution)' 문제, 세 번째는 ND 상호운용성 패턴에서만 나타나는 문제로 대기 중이던 일부 액티비티가 실행되지 못하는 '실행누락(partial execution)' 문제이다.

먼저 '미실행' 문제가 존재하는 변경패턴에 대해 살펴보자. <그림 9>에서 보는 것과 같이 송/수신 메시지 기능을 가지고 있는 액티비티 B를 '선택분할 패턴'으로 변경했을 때 변경된 프로세스에서 메시지 수신 기능이 없는 B_1, B_2가 선택되면 메시지를 수신할 수 없어 실행되지 않을 가능성이 있다.

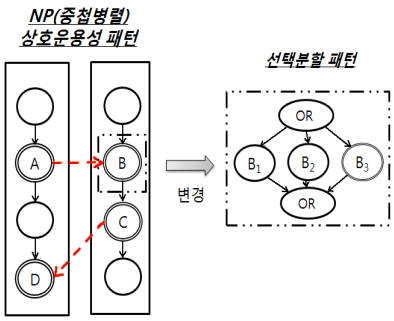

그림 9. 선택분할 패턴(C3)

또한 <그림 10>을 살펴보면 액티비티 삭제 유형의 세 가지 변경 패턴은 모두 '미실행' 문제를 가지고 있다. '완전삭제 패턴(C7)'의 경우 송/수신 메시지 기능을 가지고 있는 액티비티에 적용하여 변경하면 실행되지 않을 문제점이 존재한다. 아래 <그림 10>에서 보면 송/수신 메시지를 가지고 있는 모든 액티비티들은 삭제하면 '미실행' 문제점이 발생한다.

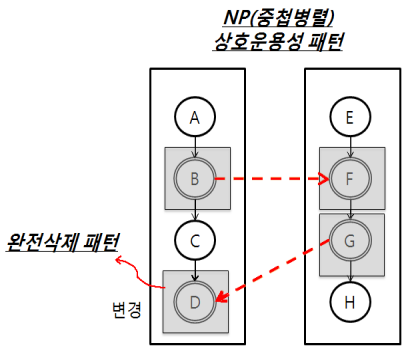

그림 10. 완전삭제 패턴(C7)

두 번째로 '역실행' 문제가 존재하는 변경패턴에 대해 살펴보자. <그림 11>에서 보는 것과 같이 송/수신 메시지 기능을 가지고 있으며 직렬인 액티비티 B와 액티비티 C를 '순서교체 패턴'으로 변경했을 때 변경된 프로세스에서 액티비티 C가 액티비티 B에게 메시지를 수신하고 진행될 수 없기 때문에 실행되지 않을 가능성이 있다.

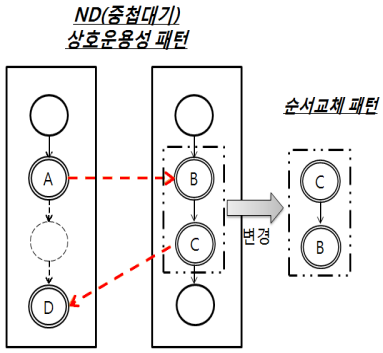

그림 11. 순서교체 패턴(C13)

세 번째로 '실행누락' 문제가 존재하는 변경패턴에 대해 살펴보자. '실행누락'은 ND(중첩대기) 상호운용성 패턴에서만 발생할 수 있는 오류이다. <그림 12>에서 액티비티 D는 액티비티 C로부터 메시지를 수신하고 재시작되어야 한다. 그러나 그림과 같이 액티비티 D가 액티비티 D_1, D_2, D_3로 직렬분할되고 액티비티 D_1이 아닌 D_2, D_3가 메시지를 수신한다면, D_1이 재시작되지 못하므로 '실행누락'이 발생한다.

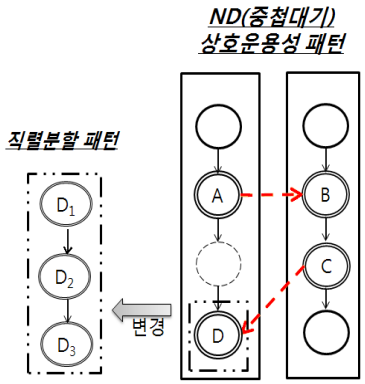

그림 12. 직렬분할 패턴(C1)

4.3 독립적 변경 보장 규칙

본 연구에서는 협업 프로세스의 독립적 변경 보장 규칙을 개발하기 위하여, 제 4.2절에서 설명한 세 가지 오류 발생 가능성 여부를 분석하고, <표 2>와 같이 19가지 프로세스 변경 패턴에서 야기할 수 있는 문제 유형을 정리하였다.

기존에 두 조직에서 정의되어 수행되던 협업 프로세스 중의 일부를 변경하고자 할 때, 전체 프로세스 또는 상대 프로세스에 영향을 미치는지 확인할 수 있는 규칙을 개발하였다.

표 2. 독립적 변경 시 발생 가능한 오류

Process change patterns	Error types of independent changes		
	non-execution (E_{NE})	reverse execution (E_{RE})	partial execution (E_{PE})
Serial split(C1)			O
Parallel split(C2)			O
Selective split(C3)	O		O
Merge of serial(C4)			
Merge of parallel(C5)			
Merge of selection(C6)			
Complete deletion(C7)	O		O
Partial deletion(C8)	O		O
Replacement(C9)	O		O
Serial insertion(C10)			
Parallel addition(C11)			O
Selective addition(C12)	O		O
Reverse order(C13)		O	
Parallelization of serial(C14)		O	
Serialization of parallel(C15)			
Selective split of serial(C16)	O	O	
Serialization of selection(C17)			
Selective split of parallel(C18)	O		
Parallelization of selection(C19)			

협업 프로세스의 독립적 변경을 확인하는 과정은 <그림 13>과 같다. 먼저, 프로세스 변경 사항을 확인하여, <표 2>에서 제시된 것처럼 변경 문제가 발생할 수 있는 패턴인지 확인한다. 만약 미실행 가능성이 존재하면 변경이 불가하고, 역순서 발생 가능성이 존재하면, 상호운용성 패턴 및 송신 액티비티(a^S), 수신 액티비티(a^R)의 위치 등을 확인하여 상세한 검사를 수행한다. 이를 통과하면, 재시작 불가능성이 존재하는지 확인하고 마찬가지로 상호운용성 패턴 및 송수신 액티비티의 위치를 확인하는 단계를 거친다. 이 세 가지 변경 문제 발생 가능성 검사를 통과하면 최종적으로 프로세스 변경이 허용된다.

5. 결 론

본 연구는 협업 프로세스에서 나타날 수 있는 19가지 협업 프로세스 변경 패턴을 프로세스 상호운용성 패턴에 적용함으로써, 협업 프로세스의 독립적 변경을 보장할 수 있는 규칙을 개발하였다. 현재까지 협업 프로세스의 설계 및 상호운용성의 중요성을 인식하고 있었지만 기술적인 문제해결에 중점이 되어 왔을 뿐, 협업 프로세스의 설계 변경에 대하여 많은 연구가 진행되지 못하였다. 특히 변경 패턴을 정리하고 독립적으로

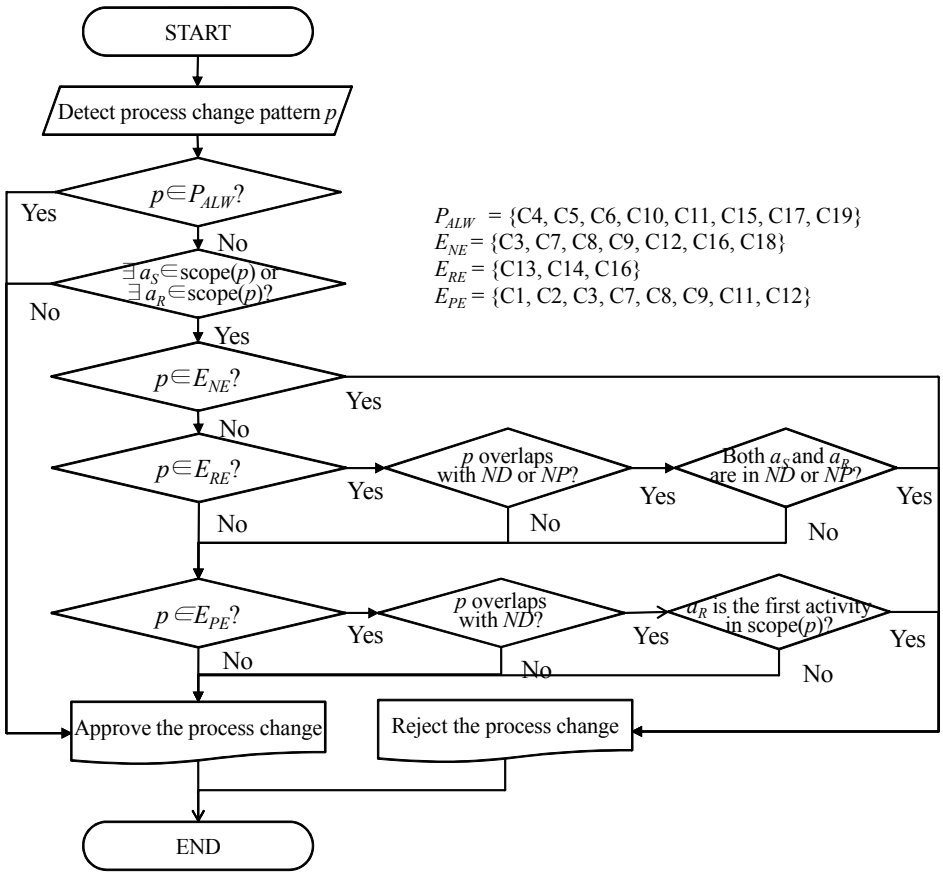

그림 13. 협업 프로세스의 독립적 변경 보장 알고리즘

변경 보장할 수 있는 방법론을 제시한 연구가 보고된 바 없다.

실제 협업 상에서는 기업들이 독립적으로 프로세스 변경 및 수정을 필요로 하는 경우가 많지만, 이러한 임의적인 변경요구는 전체 협업 프로세스에 많은 문제를 야기할 수 있다. 본 연구에서는 이러한 요구를 반영하여 독립적 변경 가능 패턴들을 쉽고 빠르게 판독할 수 있는 알고리즘을 개발하였다. 본 연구에서 제시한 협업 프로세스 변경 패턴들과 독립적 변경 보장 알고리즘은 비즈니스 프로세스에서 효율적으로 여러 기업이 상호 협력할 수 있도록 지원하는데 중요한 기반을 제공할 것이다.

참고문헌

Dumas, M., van der Aalst, W. M. P., ter Hofstede, A. H. (2005), *Process-Aware Information Systems : Bridging People and Software Through Process Technology*, Wiley-Interscience, Hoboken, NJ.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA.

Jung, J.-Y., Hur, W., Kang, S.-H., and Kim, H. (2004), Business Process Choreography for B2B Collaboration, *IEEE Internet Computing*, **8**(1), 37-45.

Kim, D., Lee, N., Kang, S., Cho, M., and Kim, M. (2010), Business Process Version Management Based on Process Change Patterns, *International Journal of Innovative Computing, Information and Control*, **6**(2), 567-575.

Liu, C.-L. (2010), CDADE : Conflict detector in activity diagram evolution based on speech act and ontology, *Knowledge-Based Systems*, **23**(6), 536-546.

van der Aalst, W. M. P. (1999), Formalization and Verification of Event-driven Process Chains, *Information and Software Technology*, **41**(10), 639-650.

van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., and Barros, A. P. (2003), Workflow Patterns, *Distributed and Parallel Databases*, **14**(3), 5-51.

Weber, B., Reichert, M., and Rinderle-Ma, S. (2008), Change patterns and change support features-Enhancing flexibility in process-aware information systems, *Data and Knowledge Engineering*, **66**(3), 438-466.

WfMC (2001), *Workflow Standard-Interoperability Wf-XML Binding*, Workflow Management Coalition, WFMC-TC-1023, Lighthouse Point, FL.

WPI (1999), Workflow Patterns Home Page. http://www.workflowpatterns.com/.

김 애 경

경희대학교 산업경영공학과 학사, 석사
현재 : 경희대학교 산업경영공학과 박사과정
관심분야 : 비즈니스 성과 관리, 비즈니스 인텔리전스, 프로세스 마이닝

정 재 윤

서울대학교 산업공학과 학사, 석사, 박사
현재 : 경희대학교 산업경영공학과 조교수
관심분야 : 비즈니스 프로세스 관리, 유비쿼터스 서비스 컴퓨팅, 인터넷 비즈니스