비즈니스 프로세스 모델에서의 설계 이상 현상
(Design Anomalies in the Business Process Modeling)

김 건 우 † 이 정 화 † 손 진 현 ††
(Gun-Woo Kim) (Jeong-Wha Lee) (Jin Hyun Son)

요 약 비즈니스 프로세스란 기업의 목표 달성을 위하여 다양한 비즈니스 규칙에 의해 정의된 상호 연관이 있는 비즈니스 기능의 집합을 의미한다. 이러한 비즈니스 프로세스 관리를 위해 많은 기업들은 프로세스 모델링 작업을 수행하게 되는데, 이러한 모델링 작업은 사람에 의해 수행되기 때문에 예기치 못한 이상 현상이 발생할 수 있게 된다. 이러한 이상 현상이 미리 검출되지 않고 프로세스 엔진에 의해 실행된다면 막대한 비용 및 손실을 초래할 수 있기 때문에 모델링 단계에서 이상 현상이 없도록 모델링을 하거나 모델링 도구 자체에서 미리 정의된 이상 현상들을 검출하는 작업이 필요하다. 본 논문에서는 비즈니스 프로세스 이상 현상 검출작업에 활용될 수 있도록 모델링 단계에서 발생할 수 있는 이상 현상들을 타입에 따라 분류하고 정의하였다.

키워드 : 비즈니스 프로세스, 프로세스 모델링, 설계 이상 현상

Abstract Business Process is a set of interrelated business functions, which are defined by various business process rule and that will lead a company to accomplishing a specific organizational goal. Many business organizations are using process modeling methods for their business process management, but mostly these methods are accomplished by Human-Based activities. These human-based activities cause unexpected design anomalies in modeling phase. If process engine executed without design anomalies detection, that will be lead to huge loss on costs. To ensure that there is no design anomalies in modeling phase and to detect anomalies of predefined actions within modeling tools are important issues in business process management. In this paper, we provide specific types of design anomalies, which can effectively use to detect design anomalies in business process modeling phase.

Key words : Business Process, Process Modeling, Design Anomalies

1. 서 론

경영환경의 변화가 가속화 되고 있는 최근의 기업 환경 변화에 빠르게 적응하기 위해 다수의 기업들이 비즈니스 프로세스 관리에 관심을 기울이고 있다. 비즈니스 프로세스란 일반적으로 기능적 역할과 관계를 정의하는 조직 구조와 관련해서 비즈니스의 목적이나 정책 목적을 실현하기 위하여 연결된 하나 이상의 절차 또는 액티비티의 집합을 말한다.

비즈니스 프로세스 라이프 사이클은 일반적으로 프로세스 모델링 - 구현 - 실행 - 관리 이렇게 총 4가지의 단계로 구성이 된다[1]. 먼저 프로세스 모델링 단계에서는 BPMN(Business Process Modeling Notation)을 이용하여 시스템 내/외부 객체들의 상호 관계를 정의하여 비즈니스 프로세스 다이어그램을 모델링한다. 이후 구현 단계에서는 이전 단계에서 제작하였던 디자인을 실행 가능한 프로세스 모델로 변환하는 작업을 수행한다. 여기에서 BPEL을 이용하여 프로세스를 구축하게 되는데 BPEL(Business Process Execution Language)은 웹서비스 환경에서 비즈니스 프로세스를 정의하고

· 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R01-2007-000-20135-0)
· 본 연구는 지식경제부 및 정보통신 연구 진흥원의 대학 IT연구센터 육성·지원사업(IITA-2008-C1090- 0801-0031)의 연구결과로 수행되었음

† 학생회원 : 한양대학교 컴퓨터공학과
gwkim@database.hanyang.ac.kr
jwlee@database.hanyang.ac.kr
†† 종신회원 : 한양대학교 컴퓨터공학과 교수
jhson@hanyang.ac.kr

논문접수 : 2008년 8월 21일
심사완료 : 2008년 10월 21일

실행하기 위한 표준 언어이다. 다음으로 실행 단계에서 비즈니스 프로세스를 플랫폼 환경에서 구축하고 프로세스에 포함된 태스크(task)들을 실행하는 작업을 수행하게 된다. 여기에서 일련의 서비스들을 end-to-end 플로우에 통합하기 위해서 여러 가지 기술적 문제(이기종 시스템과의 바인딩, 동기식/비동기식 메시지 교환 패턴의 구현, 데이타 조작, 예외 관리)들을 고려해야 한다. 그 후 구축된 비즈니스 프로세스의 성능을 평가하기 위해서 지속적으로 프로세스의 주요 변수 및 액티비티를 모니터링 하는 관리 작업이 수행된다. 이러한 모니터링 작업을 통해 이벤트 또는 예외상황에 시기적절 하게 대응할 수 있게 된다.

앞에서 언급했던 비즈니스 프로세스 라이프 사이클 중에서 가장 먼저 수행이 되는 프로세스 모델링 작업이 중요한 부분을 차지하게 되는데 프로세스 모델링은 일관성과 완벽성을 보장하여 비즈니스 분석가들과 개발자들이 이 모델에서 획득한 비즈니스 요구 사항들을 이해할 수 있도록 해야 한다. 모델링 하는 동안 일반적인 작동은 물론이고 표준 절차에 대한 대안과 예외들도 파악되어야 하는데, 모델링 작업은 사람에 의해 수행되기 때문에 예기치 못한 이상 현상이 발생할 수 있게 된다. 이러한 이상 현상이 미리 검출되지 않고 프로세스 엔진에 의해 실행된다면 막대한 손실 및 비용을 초래할 수 있기 때문에 모델링 단계에서 미리 이상 현상이 없도록 모델링을 하거나 혹은 모델링 도구 자체에서 이상 현상을 검출하는 작업이 필요하다. 따라서 본 논문에서는 비즈니스 프로세스 이상 현상 예방 및 검출을 위해 어떠한 이상 현상이 존재하는지 미리 정의하고 올바른 비즈니스 프로세스 모델링을 위해 몇 가지 제약사항 들을 나열하였다.

본 논문의 구성은 다음과 같다. 2장에서는 비즈니스 모델링을 위한 방법론 및 모델링 기법에 대해서 살펴본다. 3장에서는 비즈니스 프로세스를 모델링하기 위한 표기법인 BPMN 및 구성요소들에 대해서 살펴보며 4장에서는 각각의 타입에 따른 비즈니스 프로세스 이상 현상들을 설명한다. 마지막으로 5장에서는 논문의 결론을 맺는다.

2. 관련연구

이 장에서는 기존에 연구되었던 비즈니스 프로세스 모델링을 위한 방법론 및 모델링 기법에 대해서 살펴본다[2,3]. 2.1절에서는 프로그램의 논리 순서, 작업 및 재조 공정 등을 도표화하는 순서도(Flow Chart), 2.2절에서는 동시성과 동기적인 사건을 표현할 수 있는 페트리넷(Petri Nets), 2.3절에서는 시스템의 동적 특성을 묘사하는 UML Activity Diagram, 2.4절에서는 객체 상태 변화와 프로세스 실행 흐름의 두 가지 측면을 제공하는 IDEF3(ICAM DEFinition 3)에 대해서 기술하며 마지막으로 2.5절에서는 사무 공정의 제어 흐름 구조를 보여주는 EPC(Event Process Chains)에 대해서 기술한다.

2.1 순서도(Flow Chart)

순서도는 단일 프로세스 내에서 어떠한 제어 구조를 가지고 자료를 처리할 것인가를 정해진 기호를 통해 묘사하기 위해 개발되었고, 그 자체로 수학적 형식화가 가능하다. 본래 목적인 알고리즘 설계를 위해 처리, 데이타 입출력 등에서 다양한 표기들을 가지고 있으나, 프로세스 설계에서는 처리, 판단, 서류, 흐름선의 4개 표기가 주축을 이룬다. 여타 프로세스 설계 방법론들과 마찬가지로 활동을 나타내는 처리를 흐름선으로 연결함으로써 프로세스를 표현하는데, AND 분기/병합을 표현하는 표기가 없다는 문제점을 지니고 있다.

2.2 페트리넷(Petri Nets)

페트리넷은 1962년 Petri라는 학자에 제안된 이후 많은 확장을 통해 통신, 생산 등 다양한 분야에서 분산 시스템의 설계, 프로세스 모델링, 성능 분석, 시뮬레이션 등의 목적에 사용되고 있다. 페트리넷은 시스템을 상태(Place)와 전이(Transition), 그리고 이들 사이를 잇는 연결(Link)로 표현한다. 페트리넷으로 모델링된 시스템은 특정 상태에서 그에 연결된 전이에 해당하는 활동 또는 사건에 의해 다른 상태로 변화한다. 페트리넷에서는 시스템의 가능한 상태 및 전이 조건뿐만 아니라 현재 시스템의 상태 또한 나타낼 수 있는데, 이는 토큰(Token)을 사용함으로써 가능하다. 토큰은 다이어그램 내의 특정 상태에 위치함으로써 시스템이 해당 상태에 있음을 나타낸다. 토큰은 여러 개의 상태들을 잇는 전이, 즉 AND 분기/병합에 의해 두 개 이상으로 늘어나거나 다시 하나로 줄어들 수 있다[4].

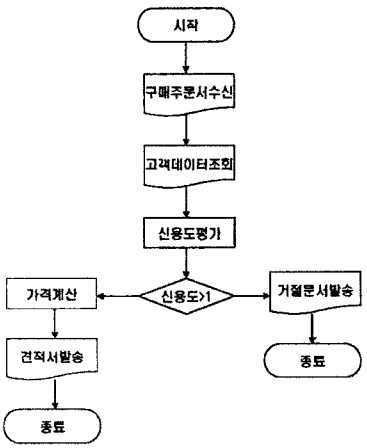


그림 1 순서도 프로세스 예시

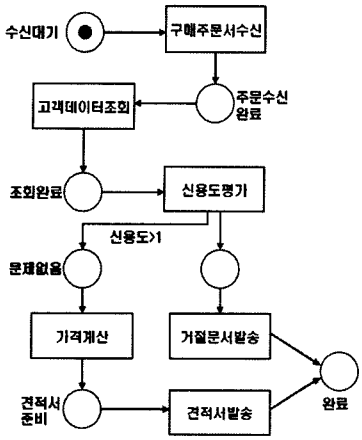


그림 2 페트리넷 프로세스 예시

2.3 UML Activity Diagram

UML은 객체지향 소프트웨어 설계를 위해 OMG에 의해 제안된 모델링 방법론이다. UML은 시스템의 정적/동적 구조를 나타내는 여러 개의 다이어그램들로 구성되는데, Activity Diagram은 시스템의 동적 설계를 프로세스의 형태로 나타낸 다이어그램이다.

UML Activity Diagram은 프로세스의 핵심을 구성하는 활동(Activity)과 그들을 잇는 순차(Sequence)로 표현된다. 또한 외부와의 상호작용을 위한 사건(Event)를 표현하기 위해 신호(Signal) 노테이션을 제공한다. 신호는 외부에서 내부로, 내부에서 외부로의 방향성을 가지고 표기된다. Activity Diagram에서는 활동들 간의 데이타 입출력을 나타내기 위해 객체(Object)를 제공하는데, 객체는 활동에 핀 형태로 첨부되어 입력과 출력을 나타내거나 순차에 첨부되어 선행 활동에서 후행 활동으로 전달되기도 하고, 또는 활동과 같이 순차에 의해 연결되어 객체 흐름을 나타낼 수도 있다. 그 외에도 프로세스의 예외상황(Exception), 루프 등의 제어문, 수행자 등을 나타내기 위한 표현 방법을 제공한다.

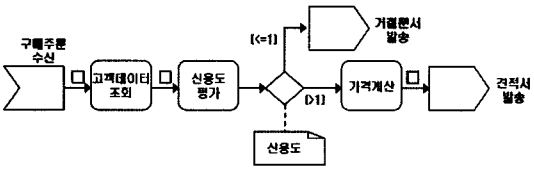


그림 3 Activity Diagram 프로세스 예시

2.4 IDEF3(ICAM DEFinition 3)

IDEF 모델에는 몇가지 타입이 존재하는데, IDEF3는 IDEF 모델링 방법론들 중 프로세스 흐름과 자원 입출력관계를 표현하기 위해 제안된 방법론이다[5]. IDEF3에서는 프로세스 설계도와 객체 상태 전이도의 두 가지 다이어그램을 제공한다. 프로세스 설계도는 여타 모델링 방법론들과 유사한 형태로 프로세스를 표현한다.

IDEF3 프로세스 설계도에서 중심이 되는 표기는 행동단위(UOB; Unit Of Behavior)로, 프로세스의 활동을 나타낸다. 프로세스는 이 행동단위들이 순차(Precedence Link)로 연결됨으로써 표현되는데, 순차는 그들이 가지는 제약조건에 의해 여러 가지 형태로 분화될 수도 있다. 프로세스 설계도에서는 행동 단위들 간의 관계를 나타내기 위해 관계(Relational Link)를 제공하는데, 이는 하위 프로세스 등 사용자가 나타내고자 하는 관계들을 자유로이 정의하여 나타낼 수 있도록 한다. 하지만 IDEF3에서는 활동, 사건, 시작/종료 등을 구분하지 않고 이들의 표현을 위해 행동단위 표기만을 제공한다.

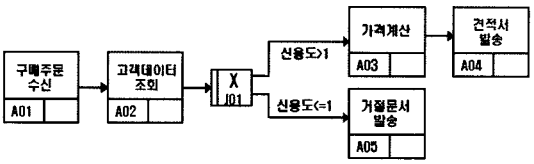


그림 4 IDEF3 프로세스 예시

2.5 EPC(Event Process Chains)

EPC는 SAP와 Saarland 대학의 연구원들 사이에 협동 노력으로 90년대 초반에 개발되었다. EPC는 최초의

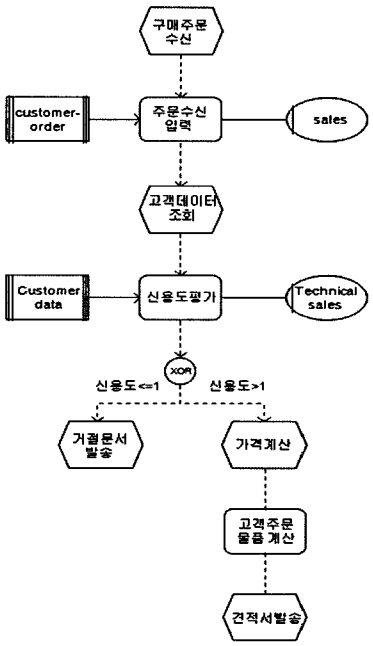


그림 5 EPC 다이어그램 예시

사건에 의해 유발되고 그 결과로서 새로운 사건을 발생시키는 기능 사슬로 표현되는 동적 프로세스의 자유로운 형식의 그래프이다. Event, Functions, Connectors를 이용하여 입/출력 데이타, 조직단위, function view에 있는 Object 사이의 연결 관계를 표현하며 프로세스의 흐름을 분석함으로써 비용 및 시간 속성을 결정할 수 있다[6].

3. 배경지식

이 장에서는 비즈니스 프로세스 모델링 표기법인 BPMN 및 구성 요소들에 대해서 살펴본다. 3.1절에서는 BPMN의 개요 및 표준 BPM 스택에 대해서 기술하고, 3.2절에서는 BPMN의 전체적인 구성 요소에 대해서 기술한다.

3.1 BPMN 개요

BPMN은 비즈니스 프로세스를 가시화 하기위한 모델링 표기법으로써 미국 비영리단체 BPMI에서 비즈니스 프로세스 모델에 관해 업계 표준을 만들고자 생겨났다. BPMN은 모든 사용자가 사용하고 이해하기 쉬운 Notation 개발을 목표로, BPEL4WS와 BPML과 같이 비즈니스 프로세스의 실행을 위해 설계된 XML언어에 시각적으로 표현 가능한 공통된 Notation을 제공하려는 목적으로 개발되었다. BPMN은 가독성(readability)를 위해 흐름도 표기법의 전통을 따르면서, 한편으로는 실행 가능한 언어 구성체(constructs)로의 변환(mapping)을 제공한다[13].

아래 그림 6은 현재 BPMI에서 완성을 목표로 하고 있는 BPM 표준 구조이다.

BPSM은 프로세스 의미 설계의 표준이다. 프로세스 메타 모델을 설계, 통합할 수 있게 하는 것을 목표로 하고 있다. BPMI에서는 BPML의 제안 이후 BPML을 BPMS의 프로세스 실행 언어로 사용하였으나, 2004년 6월 프로세스 실행언어의 표준으로 OASIS의 BPEL을 받아들였다. BPMI에서는 BPEL을 BPM 표준 구조에 포함시키는 대신 BPEL의 확장 메커니즘을 이용, BPEL의 기능을 확대하기로 하였는데 이 확장 내용이 BPXL이다. BPQL은 프로세스 쿼리 언어의 표준안으로 제안 작성이 진행 중이다. BPQL은 프로세스 인스턴스에 대해 질의를 실행할 수 있도록 함으로써 프로세스 모니터링 및 분석에 이용될 수 있으니 비즈니스 활동 모니터링의 기반 기술이 된다. WS-CDL은 기업간 웹서비스 협업을 설계하기 위한 언어로 W3C에서 제안 하였다. Web Services Stack은 BPM 표준 구조를 통한 프로세스 구축과 실행의 기반기술이 되는 계층으로 WSDL, UDDI등 웹서비스의 기반 표준들을 의미한다[3].

3.2 BPMN 구성 요소

BPMN의 기본 요소 및 구성은 아래 그림 6에 나타나 있는 바와 같이 흐름객체(Flow Object), 연결객체(Connecting Object), 스윔레인(Swimlane), 부가객체(Artifact)로 구분할 수 있다.

3.2.1 흐름객체(Flow Object)

흐름객체(Flow Object)는 프로세스의 실행 중에 발생하여 해당 프로세스의 흐름을 변경시키거나 다른 이벤트의 발생을 촉발시키는 이벤트(Event)와 실제 작업을 의미하는 액티비티(Activity), 그리고 프로세스의 흐름을 결정하는 게이트웨이(Gateway)로 구성된다.

3.2.2 연결객체(Connecting Object)

연결객체는 같은 풀(Pool)내의 액티비티들 사이의 연

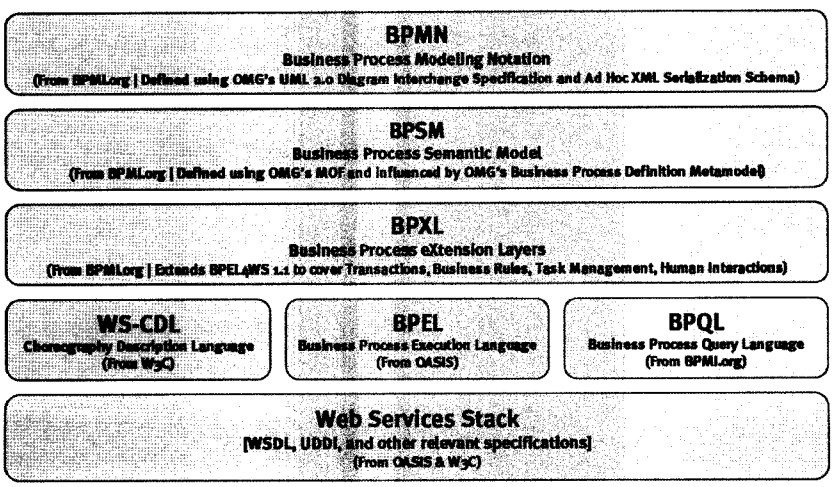


그림 6 표준 BPM 스택

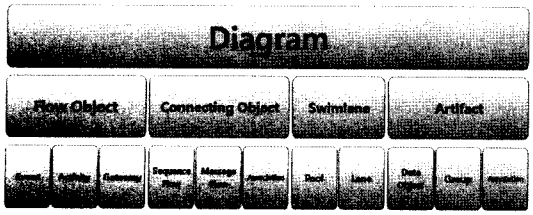


그림 7 BPMN 구조

그림 8 흐름객체(Flow Object)

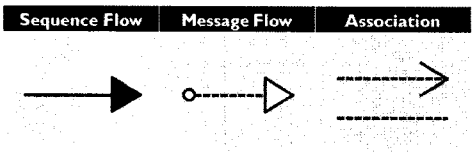


그림 9 연결객체(Connecting Object)

결을 정의하는 순서흐름객체(Sequence flow), 다른 풀들에 각각 존재하는 액티비티들 사이의 상호 통신을 정의하는 메시지흐름객체(Message flow), 프로세스내의 액티비티들과 관련 데이타나 설명문, 액티비티 들의 그룹화 등으로 대표되는 부가객체(Artifact)들 사이의 연관 상태를 표현하는데 이용되는 연관성객체(Association)로 구성되어 있다.

3.2.3 스윔레인(Swimlane)

스윔레인(Swimlane)은 풀(Pool)과 레인(Lane)으로 구성되어 있는데 BPM에서는 풀과 풀 안의 레인 이라는 그래프 표기를 이용하여 프로세스와 그 프로세스에 연관된 액티비티들의 역할이나 조직의 구성단위를 가시적으로 정의한다.

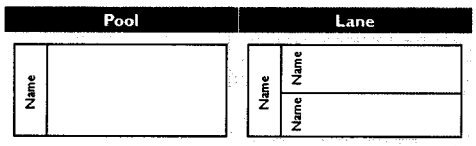


그림 10 스윔레인(Swimlane)

3.2.4 부가객체(Artifact)

마지막으로 부가객체(Artifact)는 비즈니스 프로세스의 순서흐름객체나 메시지흐름객체와 직접적으로 관련

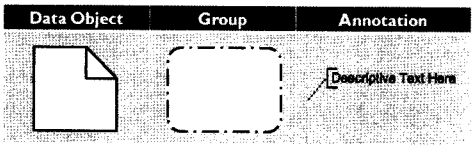


그림 11 부가객체(Artifact)

되지 않은 프로세스에 대한 추가적인 정보를 표시하기 위해 사용 된다. 여기에는 문서, 데이타, 다른 오브젝트들이 프로세스 내에서 어떻게 사용되고 업데이트 되는지를 나타내는 데이타객체(Data Object), 프로세스 내의 요소들을 묶어주는 그룹(Group), 다이어그램의 해석을 위해 추가적인 정보를 제공하는 주석(Annotation)으로 구성되어 있다.

아래 그림은 위에서 언급한 BPMN의 구성요소들로 이루어진 비즈니스 프로세스 다이어그램의 예를 나타낸다. BPMN에서는 메시지를 수신하거나 발신하는 행위가 각각 다른 행위로 표기된다. 즉, 사건과 활동이 구분된다.

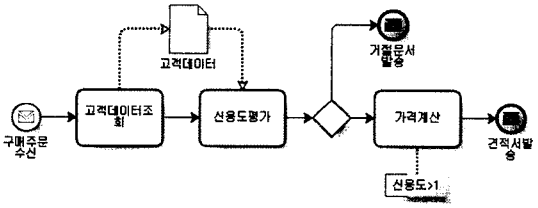


그림 12 BPMN 다이어그램 예시

4. 비즈니스 프로세스 모델에서의 이상 현상

일반적인 경영환경에서 비즈니스 프로세스를 개발하고 효과적으로 관리하기 위해서는 전체적인 프로세스에 대한 이해와 프로세스의 통합 및 각 부처에 대한 경계선을 이해하기 쉽게 표현하는 것이 중요하다. 이는 비즈니스 프로세스 관리에서 중요한 핵심요소로써 이러한 요소들을 표현하기위해 비즈니스 프로세스 모델링 작업을 수행하게 된다. 하지만 현업에서 일반적으로 사용되는 비즈니스 프로세스는 복잡한 구조로 구성되어 있어 모델링 단계에서 예상치 못한 이상 현상들이 발생할 수 있다. 기존에 이러한 이상 현상 검출에 관련된 논문에서 이상 현상에 관한 몇 가지 분류들을 살펴볼 수 있었는데 대부분이 체계적인 분류가 없이 이상 현상을 정의하였다. 예를 들면 그래프 감소 기법에 관한 논문[10] 에서는 데드락과 동기화의 부족에 관한 이상 현상을 '구조적 충돌(Structural Conflict)'이라 정의하였고, 데드락 패턴에 대한 쿼리로 이상 현상을 검출하는 논문[11]에서

는 같은 이상 현상에 대해 구조적 에러(structural error)로 정의하였으며, semantic table을 이용하여 이상 현상을 검출하는 논문[12]에서는 위와 같은 이상 현상을 의미론적 에러(semantic error)로 정의하였다. 위와 같이 이상 현상에 대해 일관된 명칭이 존재하지 않기 때문에 본 논문에서는 이상 현상의 타입 및 속성에 따라 크게 3가지로 분류하였다.

비즈니스 프로세스를 모델링할 때 발생할 수 있는 이상 현상들은 구문론적(Syntactic) 이상 현상, 구조적(Structural) 이상 현상, 그리고 의미론적(Semantic) 이상 현상으로 나눌 수 있다. 4.1절에서는 비즈니스 프로세스 모델에서의 구문론적 이상 현상에 대해서 기술하고, 4.2절에서는 비즈니스 프로세스 모델에서의 구조적 이상 현상에 대해서 기술하며 마지막으로 4.3절에서는 비즈니스 프로세스 모델에서의 의미론적 이상 현상에 대해서 기술한다.

4.1 비즈니스 프로세스 구문론적 이상 현상

비즈니스 프로세스 구문론적 이상 현상은 실제 작업을 수행하는 액티비티 내부의 데이타 정보 및 토큰들을 고려하지 않으며 단지 모델링 요소들의 잘못된 사용만을 고려한다. 즉 모델링 요소 중 게이트웨이가 순서흐름객체(Sequence Flow)가 아닌 메시지흐름객체(Message Flow)와 연결되어 도입흐름(Incoming Flow) 또는 방출흐름(Outgoing Flow)을 가지고 있을 때 비즈니스 프로세스 구문론적 이상 현상이라고 정의한다.

비즈니스 프로세스 구문론적 이상 현상은 프로세스 모델링 단계에서 사용되는 모델링 툴에 적용할 수 있지만 대부분의 모델링 툴에서는 일반적으로 본 논문에서 분류한 구문론적 이상 현상 중 연결객체에서의 잘못된 사용만 적용이 되어있다. 이처럼 대부분의 모델링 툴에서 모든 구문론적 이상 현상이 적용되어있지 않기 때문에 본 논문에서는 비즈니스 프로세스 구문론적 이상 현상을 크게 3가지로 분류하여 제시하였다.

비즈니스 프로세스 구문론적 이상 현상은 흐름객체(이벤트, 액티비티, 게이트웨이)에서의 잘못된 사용, 연결객체(순서흐름객체, 메시지흐름객체, 연관성객체)에서의 잘못된 사용 그리고 스윔레인(풀, 레인)에서의 잘못된 사용 이와 같이 크게 3가지로 존재한다.

4.1.1 흐름객체에서의 잘못된 사용(Incorrect Usage in Flow Objects)

흐름객체에서의 잘못된 사용은 이벤트, 액티비티 또는 게이트웨이의 잘못된 위치 선언으로 인해 생기는 이상 현상이다. 흐름객체에서의 잘못된 사용은 이벤트, 액티비티, 게이트웨이에 따라 각각의 세부 이상 현상이 존재한다.

4.1.1.1 이벤트의 잘못된 사용(Incorrect Usage of Event)

이벤트의 잘못된 사용은 주로 중간 이벤트의 유형 설정 시에 나타난다. 중간 이벤트의 특성상 액티비티 경계에 부착되어 사용 되거나 정상적인 흐름에서 독립적으로 사용되게 되는데 중간 이벤트의 잘못된 유형설정으로 인하여 이상 현상이 발생할 수 있다.

이벤트의 잘못된 사용은 액티비티 경계에 부착된 중간 이벤트의 잘못된 트리거 유형 사용 그리고 정상적인 순서흐름에서 중간이벤트의 잘못된 트리거 유형 사용이와 같이 크게 2가지의 이상 현상이 존재한다.

가) 액티비티 경계에 부착된 중간 이벤트의 잘못된 트리거 유형 사용(Incorrect Usage of Intermediate Event Trigger Type in Attached Boundary of Activity): 하나 또는 이상의 중간 이벤트(Intermediate Event)는 비즈니스 프로세스 모델링 단계에서 정의된 액티비티 경계에 직접적으로 부착될 수 있다. 하지만 액티비티의 경계에 부착되기 위해 중간 이벤트는 반드시 Message, Timer, Error, Cancel, Compensation, Conditional, Signal과 Multiple 트리거 유형 중 하나이어야 한다.

액티비티 경계에 부착된 중간 이벤트의 잘못된 트리거 유형 사용은 위에 제시된 트리거 유형이 아닌 다른 형태의 트리거 유형(None, Link)이 중간 이벤트로 설정되어 액티비티 경계에 부착된 후 사용될 경우를 의미한다.

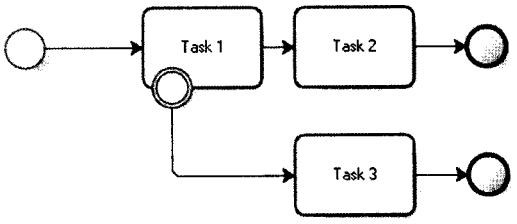


그림 13 액티비티 경계에 부착된 중간 이벤트의 잘못된 트리거 유형 사용 - None Trigger Type

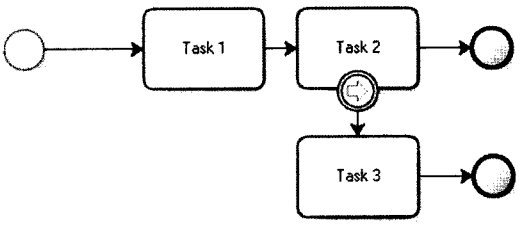


그림 14 액티비티 경계에 부착된 중간 이벤트의 잘못된 트리거 유형 사용 - Link Trigger Type

나) 정상적인 순서 흐름에서 중간 이벤트의 잘못된 트리거 유형 사용(Incorrect Usage of Intermediate

Event Trigger Type in Normal Flow): 하나 또는 이상의 중간 이벤트(Intermediate Event)는 비즈니스 프로세스 모델링 시 정상적인 흐름에서 액티비티와 같이 도입순서흐름(Incoming Sequence Flow) 및 방출순서흐름(Outgoing Sequence Flow)을 가지며 독립적으로 사용되어 질 수 있다. 하지만 정상적인 흐름에서 중간 이벤트가 사용되기 위해서는 트리거 유형이 None, Message, Timer, Exception, Compensation, Conditional, Link 및 Signal 형태여야 한다.

정상적인 순서흐름에서 중간 이벤트의 잘못된 트리거 유형 사용은 위에 제시된 트리거 유형이 아닌 다른 형태의 트리거 유형(Cancel, Error, Multiple)이 중간 이벤트에 설정되어 정상적인 순서흐름에서 사용될 경우를 의미한다.

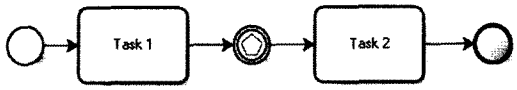


그림 15 정상적인 순서흐름에서 중간 이벤트의 잘못된 트리거 유형 사용 - Multiple Trigger Type

4.1.1.2 액티비티의 잘못된 사용(Incorrect Usage of Activity)

액티비티의 잘못된 사용은 시작이 없는 액티비티, 종료가 없는 액티비티, 그리고 수신 태스크의 잘못된 사용이와 같이 크게 3가지로 존재한다.

가) 시작이 없는 액티비티(Activities without Activation): 모델링 요소 중 시작 이벤트는 선택사항(Optional)이다. 즉 비즈니스 프로세스 모델링 시 시작 이벤트는 반드시 요구되지는 않는다. 하지만 하나의 프로세스 수준(Level), 즉 한 개의 풀(Pool) 내부에 정의된 비즈니스 프로세스모델에서 하나의 종료 이벤트가 사용 되었다면, 적어도 하나의 시작 이벤트가 사용되어야 한다.

시작이 없는 액티비티는 시작 이벤트와 종료 이벤트가 모두 사용되었지만 도입순서흐름(Incoming Sequence Flow)을 가지지 않는 흐름객체(Flow Object)들이 존재하는 경우를 의미한다.

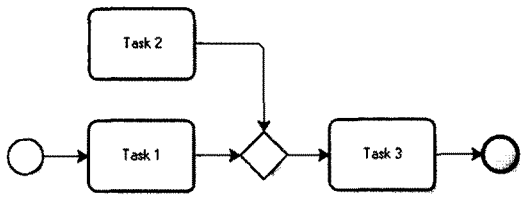


그림 16 시작이 없는 액티비티

나) 종료가 없는 액티비티(Activities without Termination): 모델링 요소 중 종료 이벤트는 시작 이벤트와 마찬가지로 선택사항 (Optional)이다. 즉 종료 이벤트 또한 비즈니스 프로세스 모델링 시 항상 요구되는 것은 아니다. 하지만 시작 이벤트와는 다르게 종료 이벤트는 하나의 프로세스 수준(Level)에서 한 개의 시작 이벤트가 사용되었을 때 여러 개의 종료 이벤트가 사용될 수 있다.

종료가 없는 액티비티는 시작 이벤트와 종료 이벤트가 모두 사용되었지만 방출순서흐름을 가지지 않는 흐름객체들이 존재하여 프로세스 종료가 수행되지 않는 경우를 의미한다.

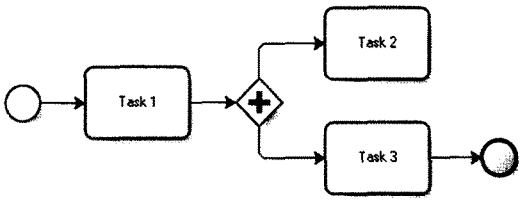


그림 17 종료가 없는 액티비티

다) 수신 태스크의 잘못된 사용(Incorrect Usage of Receive Task): 비즈니스 프로세스에서 수신 태스크(Receive Task)는 비즈니스와 관계된 외부의 참여자로부터 메시지가 도착 하는 것을 기다리도록 지정하는 당일의 태스크이다. 일반적으로 비즈니스 프로세스 모델링 단계에서 수신 태스크는 그림 18과 같이 표현된다.

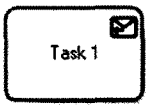


그림 18 비즈니스 프로세스 모델링 작업에서의 수신 태스크 표현

수신 태스크는 메시지를 받는 즉시 태스크가 종료된다. 즉 메시지를 받음으로써 프로세스가 부트트랩 되는 태스크로 프로세스를 시작하는데 주로 쓰인다. 그러므로 수신태스크를 사용하는 비즈니스 프로세스에서는 다음의 2가지 조건중 하나가 충족되어야 한다. 첫째 수신 태스크가 사용된 프로세스는 시작 이벤트를 가지지 않으며, 수신 태스크는 도입순서흐름(Incoming Sequence Flow)을 가지지 않는다. 둘째 수신 태스크를 위한 도입순서흐름을 가질 경우 도입순서흐름의 근원(Source)은 항상 시작 이벤트만을 사용하여야한다. 즉 다른 도입순서흐름은 수신태스크에서 사용될 수 없다.

수신 태스크의 잘못된 사용은 위에 제시한 2가지 조건을 충족시키지 않고 비즈니스 프로세스를 표현할 경우를 의미한다.

아래의 그림 19는 시작 이벤트와 종료 이벤트가 쓰이지 않은 비즈니스 프로세스에서 수신태스크가 도입순서흐름을 가지는 잘못된 사용으로 인한 이상 현상을 나타낸 것이다.

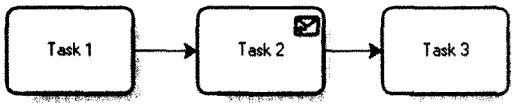


그림 19 수신 태스크의 잘못된 사용 - 시작과 종료 이벤트가 쓰이지 않을 때의 이상 현상

아래의 그림 20은 수신 태스크의 도입순서흐름으로 시작 이벤트가 사용 되었지만 루프 연결을 통하여 다른 도입순서흐름이 수신 태스크와 연결된 이상 현상을 나타낸 것이다.

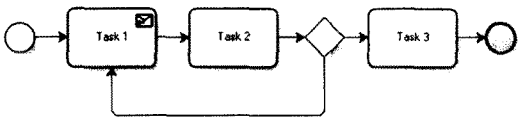


그림 20 수신 태스크의 잘못된 사용 - 루프연결을 통한 이상 현상

4.1.1.3 게이트웨이의 잘못된 사용(Incorrect Usage of Gateway)

게이트웨이의 잘못된 사용은 주로 데이타 기반 및 이벤트 기반의 XOR 게이트웨이에서 나타난다. 게이트웨이의 잘못된 사용은 데이타 기반의 XOR 게이트웨이의 잘못된 사용 그리고 이벤트 기반의 XOR 게이트웨이의 잘못된 사용 이와 같이 크게 2가지로 존재한다.

가) 데이타 기반의 XOR 게이트웨이의 잘못된 사용 (Incorrect Usage of Data-Based XOR Gateway): 데이타 기반의 XOR 게이트웨이는 비즈니스 프로세스에서 2개 이상의 방출순서흐름 중 하나만을 택해야 할 경우 사용된다. 데이타 기반의 XOR 게이트웨이는 입력된 토큰에 기반 하여 프로세스 데이타의 값을 통해 방출순서흐름 경로를 결정하게 된다.

데이타 기반의 XOR 게이트웨이의 잘못된 사용은 데이타 기반의 일반 태스크를 목표로 한 방출순서흐름이 아닌 중간 이벤트를 목표로 한 방출순서흐름에서 사용될 경우를 의미한다.

아래의 그림 21에서는 중간 이벤트를 목표로 한 방출흐름을 가지는 이상 현상을 나타낸 것이다.

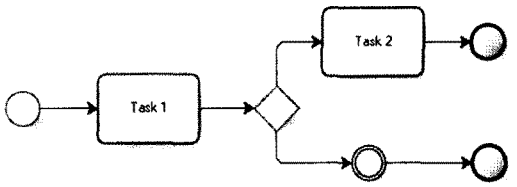


그림 21 데이타 기반의 XOR 게이트웨이의 잘못된 사용

나) 이벤트 기반의 XOR 게이트웨이의 잘못된 사용 (Incorrect Usage of Event-Based XOR gateway): 이벤트 기반의 XOR 게이트웨이는 데이타 기반의 XOR 게이트웨이와 마찬가지로 2개 이상의 방출흐름 중 하나만을 택할 경우 사용되지만 출력 단계에서 프로세스 데이타 값이 아닌 이벤트의 출현에 기반을 둔 게이트웨이이다. 이벤트 기반의 XOR 게이트웨이는 태스크 유형이 수신 태스크 또는 중간 이벤트의 Message, Timer, Rule 또는 Link 트리거 유형을 목표로 한 방출순서흐름에 의해서 배열된다.

아래의 그림 22와 23은 수신 태스크를 및 Message 이벤트를 사용하는 이벤트 기반 XOR 게이트웨이의 예제이다.

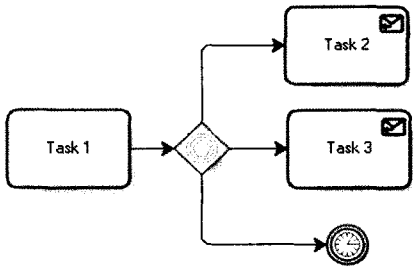


그림 22 수신 태스크를 사용하는 이벤트 기반의 XOR 게이트웨이의 예

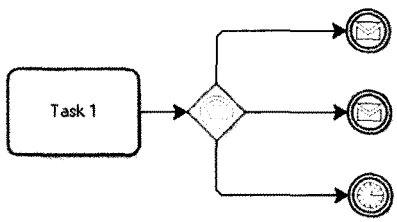


그림 23 Message 이벤트를 사용하는 이벤트 기반의 XOR 게이트웨이의 예

이벤트 기반의 XOR 게이트웨이의 잘못된 사용은 태스크 유형이 수신 태스크가 아닌 다른 유형의 태스크가 사용될 때 또는 중간 이벤트와 혼합해서 사용될 경우와

중간 이벤트의 트리거 유형이 Message, Timer, Rule 또는 Link 가 아닌 다른 유형으로 사용될 경우를 의미한다.

아래의 그림 24는 이벤트 기반의 XOR 게이트웨이에서 수신 태스크가 아닌 다른 유형의 태스크(사용자 태스크)와 잘못된 중간 이벤트 트리거 유형(Conditional Trigger Type)이 혼합해서 사용된 이상 현상을 나타낸 것이다.

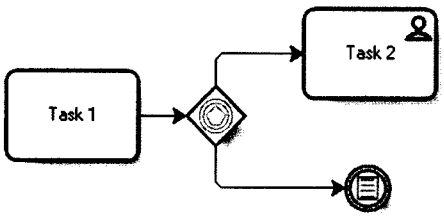


그림 24 이벤트 기반의 XOR 게이트웨이의 잘못된 사용

이벤트 기반의 XOR 게이트웨이가 프로세스 시작으로 사용될 경우 다음의 3가지 조건중 하나를 반드시 충족하여야한다. 첫째 프로세스는 시작 이벤트를 가지지 않아야 하며 이벤트 기반의 XOR 게이트웨이는 도입순서흐름을 가지지 않아야 한다. 둘째 이벤트 기반의 XOR 게이트웨이를 위한 도입순서흐름을 가질 경우 도입순서흐름의 근원(Source)은 항상 시작 이벤트만을 사용하여야한다. 마지막으로 셋째 이벤트 기반의 XOR 게이트웨이의 방출순서흐름의 목표는 Timer 트리거 유형의 중간 이벤트는 사용될 수 없다.

이벤트 기반의 XOR 게이트웨이가 프로세스 시작으로 사용될 경우의 잘못된 사용은 위에 제시한 3가지 조건을 충족하지 않고 비즈니스 프로세스를 표현할 경우를 의미한다.

아래의 그림 25는 이벤트 기반의 XOR 게이트웨이가 프로세스 시작으로 사용될 경우 Timer 유형의 중간 이벤트 트리거 유형이 사용된 이상 현상을 나타낸 것이다.

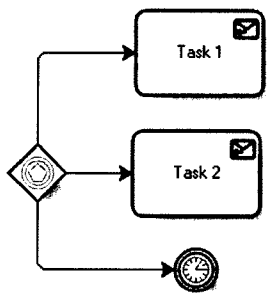


그림 25 이벤트 기반의 XOR 게이트웨이가 프로세스 시작으로 사용될 경우의 잘못된 사용

4.1.2 연결객체에서의 잘못된 사용(Incorrect Usage in Connecting Objects)

연결객체에서의 잘못된 사용은 순서흐름객체, 메시지 흐름객체 또는 연관성객체의 잘못된 위치 선언으로 인해 생기는 이상 현상이다.

가) 시작 이벤트에서 들어오는 순서흐름객체를 가질 경우(Incorrect Usage of Sequence Flow with Start Event): 비즈니스 프로세스 모델링에서 사용되는 시작 이벤트는 제어흐름객체 중 항상 나가는 방출흐름객체(Outgoing Sequence Flow)만을 가져야하며 흐름객체 중 시작 이벤트를 제외한 이벤트, 액티비티, 게이트웨이와 연결하기위해 사용된다.

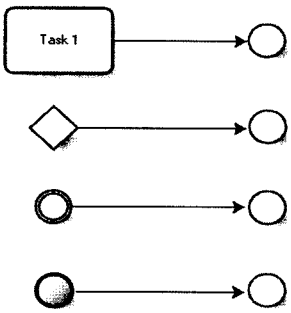


그림 26 시작 이벤트에서 들어오는 제어흐름 객체를 가진 이상 현상

나) 종료 이벤트에서 나가는 순서흐름객체를 가질 경우(Incorrect Usage of Sequence Flow with End Event): 비즈니스 프로세스 모델링에서 사용되는 이벤트 중 하나인 종료 이벤트는 앞서 제시한 시작 이벤트와는 반대로 순서흐름객체 중 항상 들어오는 도입흐름객체(Incoming Sequence Flow)만을 가져야하며 흐름객체 중 종료 이벤트를 제외한 이벤트, 액티비티, 게이트웨이와 연결하기위해 사용된다.

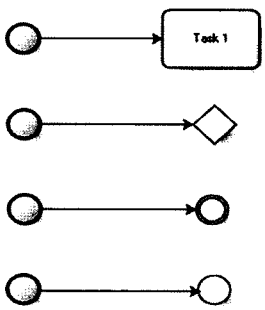


그림 27 종료 이벤트에서 들어오는 제어흐름 객체를 가진 이상 현상

다) 액티비티 경계에 부착된 중간 이벤트가 들어오는 순서흐름객체를 가질 경우(Incorrect Usage of Incoming Sequence Flow with Intermediate Event in Activity Boundary): 비즈니스 프로세스 모델링 단계에서 중간 이벤트는 액티비티 경계에 부착되어 사용 될 수 있다. 아래의 그림 28은 중간 이벤트가 액티비티의 종류 중 하나인 태스크와 붙어있는 것을 표현한 그림이다.

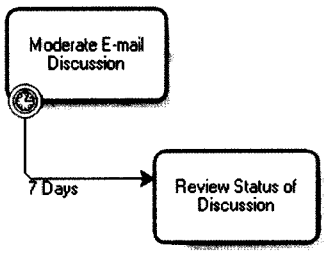


그림 28 태스크와 태스크 경계에 부착된 중간 이벤트

하지만 액티비티 경계에 부착된 중간 이벤트는 절대로 순서의 목표가 될 수 없다. 즉 액티비티 경계에 부착된 중간 이벤트는 도입흐름객체를 가질 수 없다.

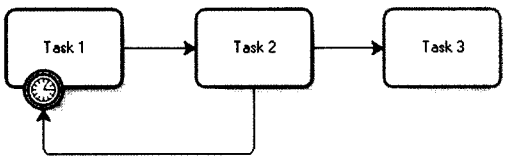


그림 29 액티비티 경계에 부착된 중간 이벤트가 도입흐름객체를 가지는 이상 현상

라) 액티비티 경계에 부착된 중간 이벤트에서 하나 이상의 나가는 순서흐름객체를 가질 경우(Multiple Outgoing Sequence Flow with Intermediate Event in Activity Boundary): 비즈니스 프로세스 모델링 단계에서 하나 이상의 중간 이벤트는 어떠한 액티비티 경계에서도 직접적으로 부착되어 사용될 수 있다. 중간 이벤트가 액티비티의 경계에 부착되어 사용될 경우 중간 이벤트는 순서 흐름의 근원(Source)으로 사용 되며 오직 하나의 방출 흐름만을 가질 수 있다.

마) 조건부 순서흐름객체의 잘못된 사용(Incorrect Usage of Conditional Sequence Flow): 비즈니스 프로세스 모델링 단계에서 조건부 흐름 객체는 프로세스 내부에서 제어되는 흐름을 가진 게이트웨이의 성질을 나타내기 위하여 사용된다. 그림 31은 조건부 순서흐름객체를 사용한 태스크 결정을 표현한 그림이다.

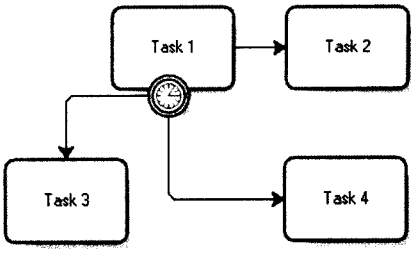


그림 30 액티비티 경계에 부착된 중간이벤트에서 하나 이상의 방출흐름객체를 가지는 이상 현상

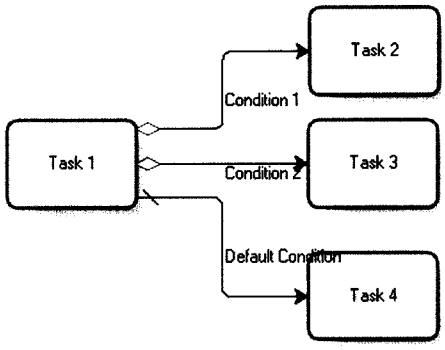


그림 31(a) 조건부 순서흐름객체를 사용한 태스크 결정

조건부 순서흐름객체의 사용에는 4 가지의 제한조건이 있다. 첫째 근원(Source) 객체는 이벤트가 절대 될 수 없다. 둘째 근원 객체가 게이트웨이일 경우 조건부 순서흐름객체의 작은 다이아몬드는 표시되지 않는다. 셋째 AND 게이트웨이는 근원 객체로 사용될 수 없다. 마지막으로 넷째 만약 근원 객체가 액티비티일 경우 적어도 근원 액티비티로 부터 하나이상의 방출순서흐름이 존재하여야한다.

조건부 순서흐름 객체의 잘못된 사용은 위에 제시한 4가지의 제한 조건을 충족시키지 않고 비즈니스 프로세스를 표현할 경우를 의미한다.

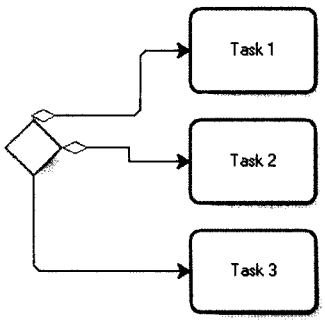


그림 31(b) 조건부 순서흐름객체를 사용한 태스크 결정

바) 메시지흐름객체의 잘못된 사용(Incorrect Usage of Message Flow): 비즈니스 프로세스 모델링 단계에서 사용되는 메시지흐름객체(Message Flow)는 2개 이상의 분리된 풀(Pool)의 경계선 또는 풀(Pool) 내부의 흐름객체를 연결하기위해 사용되며 시작 이벤트 또는 게이트웨이와 연결하여 사용될 수 없다.

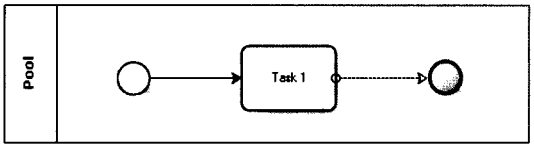


그림 32 메시지흐름객체가 한 개의 풀(Pool) 내부에 정의된 이상 현상

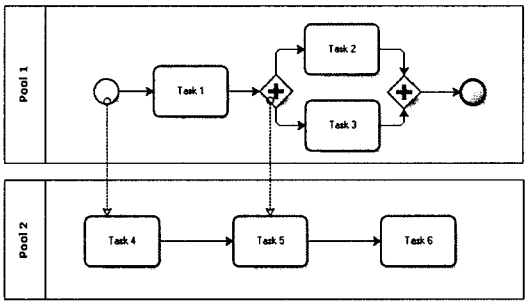


그림 33 메시지흐름객체가 시작 이벤트와 게이트웨이와 연결되어 사용되는 이상 현상

4.1.3. 스윔레인에서의 잘못된 사용(Incorrect Usage in Swimlanes)

스윔레인에서의 잘못된 사용은 풀 또는 레인의 잘못된 위치 선언으로 인해 생기는 이상 현상이다.

가) 풀의 잘못된 사용(Incorrect Usage of Pool): 비즈니스 프로세스 모델링 단계에서 사용되는 풀(Pool)은 프로세스의 참여자를 나타낸다. 비즈니스 프로세스에서 사용되는 참여자는 하나의 비즈니스 객체를 구체화(예를 들어, 회사내부의 부서)하여 표현 할 수도 있고 비즈니스 역할을 좀 더 일반화(예를 들어 B2B 관계인 판매자, 소비자 또는 생산자)해서 표현 할 수 있다.

풀(Pool)은 일반적으로 비즈니스 프로세스 모델링 단계에서 정의된 흐름객체(이벤트, 액티비티, 게이트웨이)들 사이의 순서흐름을 모두 포함하며 프로세스 다이어그램을 좀 더 명확하기 위해 하나의 풀에서 다이어그램의 전체 길이를 확장 또는 축소 할 수 있는 특징을 가지고 있다. 하지만 풀 내부에 정의된 흐름객체들 사이의 순서흐름은 풀의 경계를 교차해서 사용될 수 없다. (단 메시지흐름객체는 풀의 경계를 교차하여 사용될 수 있다)

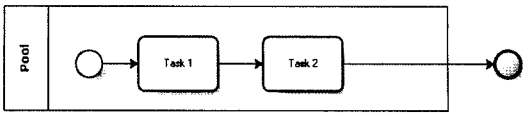


그림 34 풀 경계를 교차하여 표현된 이상 현상

나) 레인의 잘못된 사용(Incorrect Usage of Lane): 비즈니스 프로세스 모델링 단계에서 레인은 풀 내부의 하부 파티션으로 사용된다. 즉 프로세스 참여자 내부의 역할들, 시스템, 내부영역 등으로 다양하게 사용될 수 있다. 이처럼 레인은 풀 내부의 하부 파티 파티션으로 사용되기 때문에 풀 보다 먼저 사용될 수 없다.

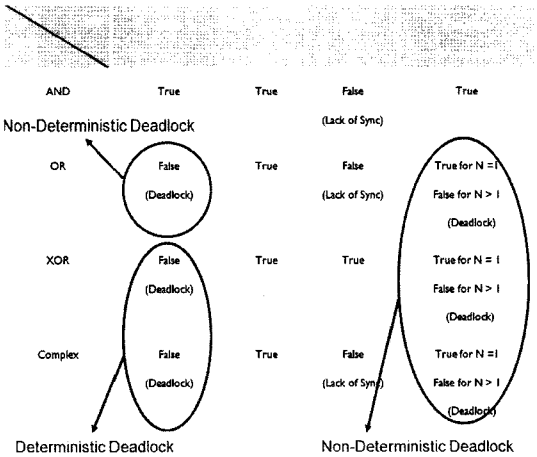
4.2 비즈니스 프로세스 구조적 이상 현상

비즈니스 프로세스 구조적 이상 현상은 잘못된 게이트웨이 사용으로 인하여 토큰의 병합 또는 분산되는 상호작용을 제어하지 못할 때 주로 발생한다. 기존에 존재하는 비즈니스 프로세스 이상 현상 검출 기법에서는 대부분이 OR 게이트웨이와 Complex 게이트웨이를 제외한 AND와 XOR 게이트웨이에 대해서만 다루고 있었는데 본 논문에서는 제어흐름 테이블(Control Flow Table)을 통해 BPMN에서 정의된 모든 타입의 게이트웨이를 지원하게끔 하였다. 대표적인 비즈니스 프로세스 구조적 이상 현상으로 데드락, 무한루프, 동기화의 부족, 수행되지 않는 액티비티가 존재한다[7-9].

4.2.1 데드락(Deadlock)

프로세스의 특정 단계에서 더 이상의 액티비티가 수행되지 않는 경우를 의미한다. 일반적으로 데드락은 조인(Join) 게이트웨이에서 발생하고, 게이트웨이 특성에 따라 결정가능 데드락과 결정불가능 데드락으로 나뉘게 된다. 아래 그림의 데드락 컨트롤 플로우 테이블에 각 게이트웨이의 경우에 따른 데드락 종류가 표시되어 있다.

표 1 Deadlock control flow table



가) 결정가능 데드락(deterministic deadlock) : AND-Join 게이트웨이와 연결된 액티비티 중 다음 단계로 진행되지 않는 액티비티가 항상 존재하는 경우를 의미한다.

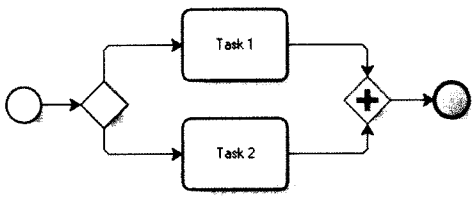


그림 35 결정가능 데드락

나) 결정불가능 데드락(non-deterministic deadlock): OR-Join 게이트웨이 또는 "N out of M 패턴"에서 N이 1보다 클 때, Complex-Join 게이트웨이와 연결된 액티비티 중 다음 단계로 진행되지 않는 액티비티가 존재 할 가능성이 있는 경우(런타임 조건에 의해 결정)를 의미한다.

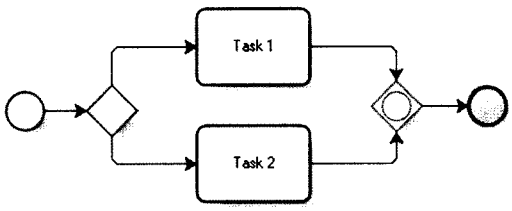


그림 36 결정불가능 데드락

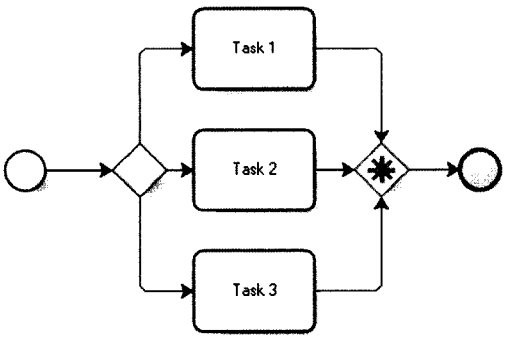


그림 37 결정불가능 데드락

4.2.2 무한루프(Livelock or Infinite loop)

프로세스의 특정단계에서 어떠한 액티비티가 영원히 반복적으로 수행되는 경우를 의미한다. 일반적으로 무한루프는 스플릿(Split) 게이트웨이에서 이전 액티비티 혹은 게이트웨이로의 흐름객체가 존재할 때 발생하고, 게이트웨이 특성에 따라 결정가능 무한루프와 결정불가능 무한루프로 나뉘게 된다.

가) 결정가능 무한루프(Livelock or Infinite loop) : AND-Split 게이트웨이와 연결된 액티비티 중 영원히 반복적으로 수행되는 액티비티가 항상 존재하는 경우를 의미한다.

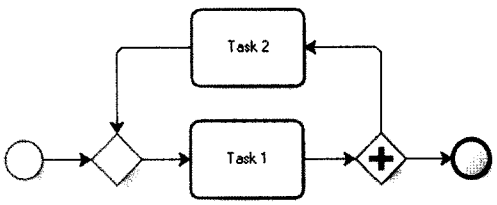


그림 38 결정가능 무한루프

나) 결정불가능 무한루프(Livelock or Infinite loop) : AND-Split 게이트웨이와 연결된 액티비티 중 영원히 반복적으로 수행되는 액티비티가 존재 할 가능성이 있는 경우(런타임 조건에 의해 결정)를 의미한다.

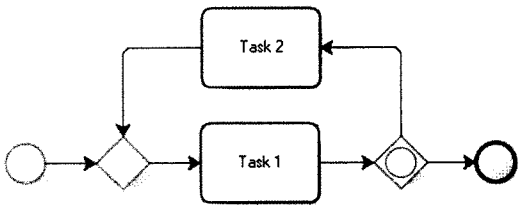


그림 39 결정불가능 무한루프

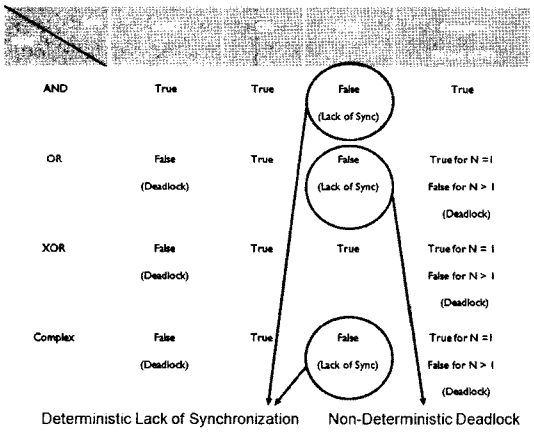
4.2.3 동기화의 부족(Lack of Synchronization)

프로세스의 특정 단계에서 의도하지 않게 해당 액티비티가 여러 번 수행되는 경우를 의미한다. 일반적으로 동기화의 부족은 XOR-Join 게이트웨이에서 발생하고, 게이트웨이 특성에 따라 결정가능 동기화의 부족과 결정불가능 동기화의 부족으로 나뉘게 된다. 아래 그림의 동기화의 부족 컨트롤 플로우 테이블에 각 게이트웨이의 경우에 따른 동기화의 부족 종류가 표시되어 있다.

가) 결정가능 동기화의 부족(deterministic lack of synchronization): XOR-Join 게이트웨이와 연결된 액티비티 중 의도하지 않게 여러 번 수행되는 액티비티가 항상 존재하는 경우를 의미한다.

나) 결정불가능 동기화의 부족(non-deterministic lack of synchronization) : OR-Join 게이트웨이와 연결된 액티비티 중 의도하지 않게 여러 번 수행되는 액티비티가 존재 할 가능성이 있는 경우(런타임 조건에 의해 결정)를 의미한다.

표 2 Lack of Synchronization control flow table



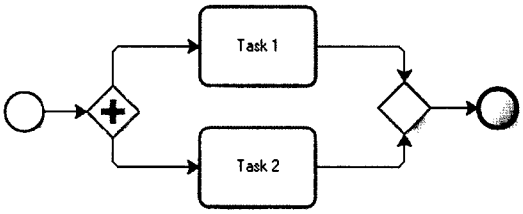


그림 40 결정가능 동기화의 부족

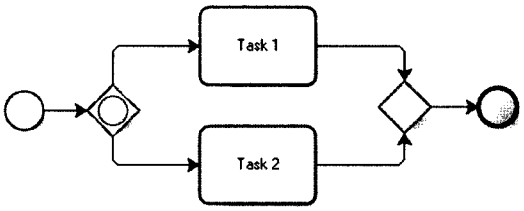


그림 41 결정불가능 동기화의 부족

4.2.4 수행되지 않는 액티비티(Dead Activities) : 수행되지 않는 액티비티가 존재하는 경우를 의미한다.

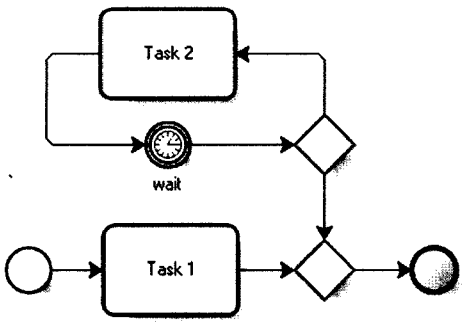


그림 42 수행되지 않는 액티비티

4.3 비즈니스 프로세스 의미론적 이상 현상

비즈니스 프로세스에 있어서 의미론적 이상 현상은 비즈니스 프로세스의 최종 목표와 일치하지 않는 프로세스를 의미론적 이상 현상이 존재하는 프로세스라고 한다. 비즈니스 모델링 단계가 수행된 후 구현 단계에서 모델링한 비즈니스 프로세스를 BPEL으로 변환하는 작업을 수행하게 되는데 이때 의미론적 이상 현상을 포함하는 프로세스는 변환 과정 시에 오류를 포함하게 되거나 혹은 전체적인 비즈니스 프로세스가 완벽하게 매핑이 되지 않는 경우가 생길 수 있다.

아래 그림과 같이 예외 이벤트로 인해 발생한 흐름이 이전 액티비티가 수행을 마치기 전에 루프를 따라 이전 흐름으로 되돌아갔을 경우 비즈니스 프로세스가 완벽하게 매핑이 되지 않을 수 있다.

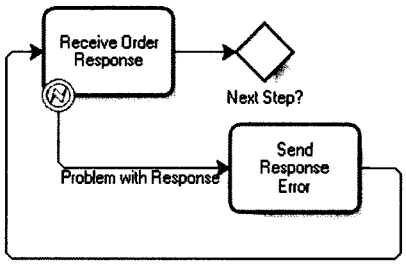


그림 43 Exception Flow Looping Back into the Normal Flow Upstream

위와 같은 경우에 태스크에서 파생된 또 다른 프로세스를 아래와 같이 연결한 후 기존의 액티비티가 정상적으로 수행을 마칠 수 있을 때까지 파생된 액티비티가 생성되어 있어야 한다. 아래의 그림을 예로 들어 설명하면 예외 상황이 발생하였을 경우 원래의 액티비티는 재수행 될 필요가 있다. 그러므로 원래의 액티비티는 다른 프로세스 내에서 중복될 것이고, 에러 핸들러는 다른 프로세스를 생성하는 one-way 액티비티를 포함할 것이다. 추가로, 원래의 프로세스는 파생된 프로세스로부터 원래의 액티비티가 정상적으로 종료되었다는 메시지를 위한 receive 액티비티를 기다릴 것이다.

5. 결 론

본 논문에서는 비즈니스 프로세스를 모델링 하는데 있어서 어떠한 이상 현상들이 존재하는지 세 가지 타입으로 구분하고 정의하여서 모델링 시에 이상 현상이 없도록 모델링을 하거나 모델링 도구에서 검출기능을 지원하고자 할 때 이러한 이상 현상들을 표본으로 삼아 잘 정의된 비즈니스 프로세스 모델을 생성하는데 도움을 주고자 하였다. 예를 들면 위에서 분류한 이상 현상

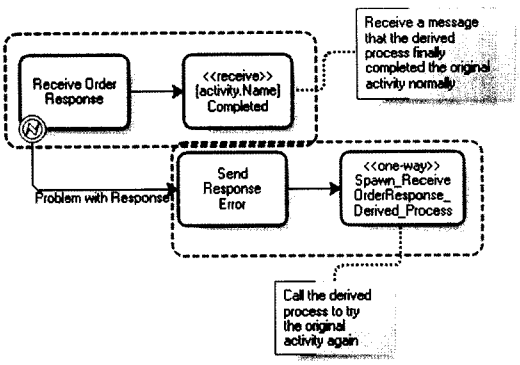


그림 44 Modified Exception Flow Looping Back into the Normal Flow Upstream

들 중 구문론적 이상 현상의 경우는 이미 대다수의 모델링 툴에서 잘못 설계된 비즈니스 프로세스를 저장 하려고 할 때에 에러메시지를 출력하는 형식으로 지원하고 있다. 하지만 구조적 이상 현상이나 의미론적 이상 현상과 같은 경우는 검출 알고리즘을 적용하거나 모델링 요소들이 내부적으로 동작하는 방식을 정확하게 알고 있어야 하기 때문에 대부분의 모델링 툴에서는 이러한 기능을 지원하지 않고 있다. 따라서 차후에 개발할 모델링 툴에서 이러한 이상 현상들을 바탕으로 검출기법을 구현해 나갈 계획이다.

참 고 문 헌

[1] Bhagat Nainani, "Closed Loop BPM using Standards based tools," Oracle Corporation, November, 2004.

[2] Arnd Schenieders, Frank Puhlmann, Mathias Weske, "Process Modeling Techniques," PESOA-Report No. 01/2004, February 6, 2004.

[3] 한국전산원, "비즈니스 프로세스 관리를 위한 질의 언어 및 모델링 표기법 표준화 연구", September 2005.

[4] Rik Eshuis, Roel Wieringa, "Comparing Petri Net and Activity Diagram Variants for Workflow Modelling-A Quest for Reactive Petri nets," Petri Net Technology for Communication-Based Systems, pages 321-351. Volume 2472 of LNCS, Springer-Verlag, November 2003.

[5] 문신명, 임춘성, 김훈태, 박승규, "참조모델을 활용한 비즈니스 프로세스 표준화를 위한 모델링 방법론 개발", 한국경영과학회/대한산업공학회 춘계공동학술대회, May 16, 2003.

[6] Hefedh Mili, Guitta Bou Jaoude, Eric Lefebvre, Guy Tremblay, Alex Petrenko, "Business Process Modeling Languages: Sorting Through the Alphabet Soup", Laboratorie de Recherche sur les Technologies du Commerce Electronique, 55 pages, November 2003.

[7] 김학수, 박찬희, 설주영, 손진현 "컨트롤 흐름 경로 기반의 비즈니스 프로세스 타당성 검증 기법", 정보처리학회논문지D, August 3, 2007.

[8] Wasim Sadiq, Maria E. Orlowska, "On Correctness Issues in Conceptual Modeling of Workflows," In Proceedings of the 5 th European Conference on Information Systems (ECIS '97), Cork, Ireland, June 19-21, 1997.

[9] Qianhui Althea Liang, J. Leon Zhao, "Verification of Unstructured Workflows via Propositional Logic," Seventh IEEE/ACIS International Conference on Computer and Information Science, 2008.

[10] H. Lin, Z. Zhao, H. Li, Z. Chen, "A Novel Graph Reduction Algorithm to Identify Structural Conflicts," hicss,pp.289, 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9, 2002.

[11] Ahmed Awad, Frank Puhlmann, "Structural Detection of Deadlocks in Business Process Models," 11th International Conference, BIS 2008, Innsbruck, Austria, May 5-7, 2008.

[12] LING Hong, ZHOU JiangBo, "Research on workflow process structure verification," icebe, pp. 158-166, IEEE International Conference on e-Business Engineering (ICEBE'05), 2005.

[13] "BPMN Specification Releases : BPMN1.1," January 17, 2008.

김 건 우

2006년 호주 뉴캐슬 대학교 컴퓨터공학과 학사. 2007년 호주 뉴캐슬 대학교 정보공학과 석사. 2008년~현재 한양대학교 컴퓨터공학과 박사과정. 관심분야는 데이타베이스, E-Business, RFID, BPM, Database

이 정 화

2007년 한양대학교 전자컴퓨터공학부 학사. 2007년~현재 한양대학교 컴퓨터공학과 석사과정. 관심분야는 Mobile, 데이타베이스, 데이타 마이닝

손 진 현

1996년 서강대학교 전산학과 학사. 1998년 한국과학기술원 전산학과 석사. 2001년 한국과학기술원 전자전산학과 박사. 2001년 9월~2002년 8월 한국과학기술원 전자전산학과 박사후 연구원. 2002년 9월~현재 한양대학교 컴퓨터공학과 부교수. 관심분야는 데이타베이스, e-비즈니스, 유비쿼터스 컴퓨팅, 임베디드 시스템