컨트롤 흐름 경로 기반의 비즈니스 프로세스 타당성 검증 기법

김 학 수[†] · 박 찬 희[††] · 설 주 영[†††] · 손 진 현[††††]

요 약

과거에 비해 최근에는 비즈니스 프로세스가 복잡해짐에 따라 비즈니스 프로세스를 디자인할 때 발생할 수 있는 문제점이 점차 증가하고 있다. 그러나 비즈니스 프로세스 검증의 중요성이 높아지고 있지만 많은 검증 방법이 제안되고 있지 않은 실정이다. 한편, 최근 BPMI에서 주도하는 BPMN은 비즈니스 프로세스를 위한 표준화된 그래픽 표기법으로써 BPMN을 지원하는 디자인 툴을 이용하면 다양하고 복잡한 프로세스 환경을 쉽게 디자인하고 분석할 수 있다. 본 논문에서는 이러한 BPMN으로 디자인할 때 발생할 수 있는 문제점들을 효율적으로 검증할 수 있는 몇 가지 검증기법을 제시한다. 이로 인해 비즈니스 프로세스 실행 시 발생할 수 있는 에러들을 사전에 검증하여 예기치 못한 큰 비용을 줄일 수 있다.

키워드 : BPMN, 비즈니스 프로세스, 프로세스 타당성 검증

Verification Checking Mechanisms of Business Processes based on Control Flow Path

Hak Soo Kim[†] · Chan-Hee Park[††] · Joo Young Sul[†††] · Jin Hyun Son[††††]

ABSTRACT

As the current trend in e-business has led to more various and complex business processes in recent years, problems in business process models have increased gradually. Accordingly, the concern to validation of business process models has been much larger but there are few validation checking mechanisms supported so far. On the other hand, BPMN driven by BPMI is a standard graphical notation. Using the tool supporting BPMN, business process can be modeled graphically and analyzed easily. In this paper, we present technical mechanisms which can efficiently detect anomalies in a process composed of BPMN and are capable of avoiding higher unexpected costs during runtime.

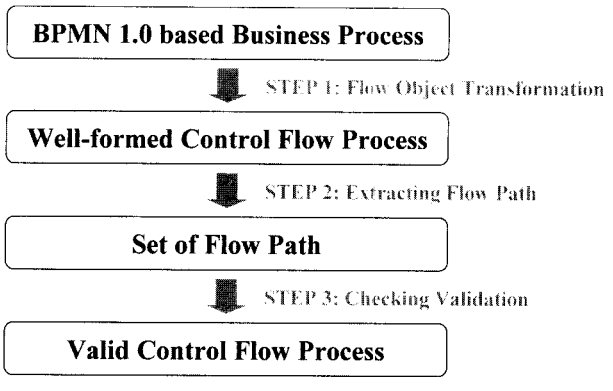
Key Words : BPMN, XPDL, Business process, Process validation check

1. 서 론

오늘날 기업 환경에서 기업 업무의 다양화 및 복잡화로 인해 프로세스 디자이너에 의해 설계되는 비즈니스 프로세스의 복잡도가 증가하고 있는 추세이다. 이로 인해 복잡한 프로세스의 타당성이 검증되지 않은 상태에서 프로세스 엔진에 의한 실행은 막대한 비용 손실을 초래하는 원인이 될 수 있다. 이에 따라 최근 연구는 복잡한 비즈니스 프로세스의 타당성을 사전에 검증할 수 있는 기법들에 대한 연구가 활발히 진행되고 있다. 비즈니스 프로세스의 타당성 검증 목적은 비즈니스 프로세스를 설계할 때 발생할 수 있는 다양한 문제점들을 비즈니스 프로세스 실행 전에 검증함으로써 예기치 못한 큰 비용을 줄이는 것이다[1, 2].

이와 더불어 복잡한 비즈니스 프로세스의 모델링을 용이하고 유지관리 하기 쉽도록 하기 위해서 그래픽 표기법의 표준화 작업이 진행되고 있다. 최근에 표준화 작업이 활발히 이루어지고 있는 그래픽 표기법으로는 BPMN(Business Process Modeling Notation)[3], UML 엑티비티 다이어그램[4, 5], UML EDOC 비즈니스 프로세스, IDEF, ebXML BPSS, ADF(Activity-Decision Flow) 다이어그램, RosettaNet, LOVeM, EPCs(Event Process Chains) 등이 있다. 이들 중에서 BPMN은 다른 어떤 비즈니스 프로세스 모델링 표기법보다 많은 기능과 확장성을 갖고 있기 때문에 본 논문에서는 BPMN을 사용해서 디자인할 경우 발생할 수 있는 에러를 검증하고자 한다.

※ 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2004-003-D00327).
본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원 사업(IITA-2006-C1090-0603-0031)의 연구결과로 수행되었음.
† 준 회 원 : 한양대학교 컴퓨터공학과 박사과정
†† 준 회 원 : 한양대학교 컴퓨터과학과 석사과정
††† 정 회 원 : LG CNS
†††† 종신회원 : 한양대학교 컴퓨터공학과 조교수 (교신저자)
논문접수 : 2006년 7월 22일, 심사완료 : 2007년 6월 11일



(그림 1) 비즈니스 프로세스 검증 단계

비즈니스 프로세스 검증 단계는 데이터 흐름에 대한 측면[6]과 컨트롤 흐름에 대한 측면[7] 두 가지로 나눌 수 있다. 데이터 흐름에 대한 측면은 비즈니스 프로세스 내의 태스크(Task) 사이에서 전달되는 데이터가 정상적인 순서로 이동되어 사용되는지 혹은 중복 및 손실 여부를 검증한다. [6]에서는 데이터가 이동될 때 발생할 수 있는 문제점을 6가지로 구분하였다. 컨트롤 흐름에 대한 측면은 수행될 태스크가 결정된 후 올바른 진행 순서대로 태스크가 디자인되어 있는지를 검증한다. 대표적인 컨트롤 플로우의 4가지 이상현상으로는 특정 단계에서 다음 단계로 진행할 수 없을 경우(데드락), 의도치 않게 특정 태스크가 여러 번 수행될 경우(동기화의 부족), 시작과 종료가 없는 경우(시작과 종료가 없는 노드) 그리고 무한하게 특정 태스크들을 수행할 경우(무한루프)가 있다. 본 논문은 컨트롤 흐름에 대한 프로세스 검증 측면에 초점을 맞추어 4가지 이상 현상을 검증할 수 있는 기법을 제안한다.

비즈니스 프로세스 검증 절차는 (그림 1)과 같은 순서로 진행된다. 먼저 BPMN1.0에서 정의한 "Flow Object Connection Rules"를 따르는 프로세스를 플로우 오브젝트 변환 과정을 통해 정형화된 컨트롤 플로우 프로세스로 변환한다. 스텝 2에서 프로세스 내에 존재하는 흐름 경로를 추출한 다음에 본 논문에서 정의한 컨트롤 흐름 경로를 통한 검증 기법을 통해서 이상현상을 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로써 기존의 검증 기법들과 비교 및 분석을 기술한다. 3장에서는 본 논문의 이해를 위해 BPMN에 대한 기본적인 내용을 요약한다. 4장에서는 정형화된 컨트롤 플로우 프로세스로 변환하는 방법을 기술하고 5장에서는 왜 프로세스 컨트롤 흐름 상에서 문제가 발생할 수 있는 지를 분석하고 그에 대한 검증 기법을 제시한다. 마지막으로 결론 및 향후 계획으로 논문을 마무리한다.

2. 관련 연구

2.1 Petri nets

Petri nets은 비즈니스 프로세스의 디자인 및 검증에 있어서 효율적인 접근 방식을 제공하기 때문에 일반적으로 많이 사용되어온 방법이다[8, 9, 10, 11]. [7]에서는 Petri nets 기반의 WF-nets(Workflow nets)으로 워크플로우를 구성하고 있고 워크플로우의 건정성(soundness)와 생존성(liveness), 제한성(boundedness), 안정성(safeness), 데드락(deadlock), 라이브락(livelock) 그리고 데드 엑티비티(dead activity)에 대한 검증에 초점을 두고 있다. Petri nets 기반의 WF-nets은 정형화된 이론에 기반을 두고 있으며 토큰 기반의 워크플로우 상태 표현을 용이하게 하고 워크플로우에 대해 이론적인 분석 및 검증을 할 수 있는 장점이 있다. 그러나 이런 장점에도 불구하고 현재 대부분의 WfMS(Work flow Management System)에서 Petri nets기반의 WF-nets을 채택해서 사용하고 있지 않다. 그 이유는 다양한 종류의 표기법을 세 개의 컴포넌트로 표현하기 때문에 설계된 프로세스가 복잡한 구조를 가지게 되어 프로세스의 유지관리에 대한 비용이 큰 단점이 있다. 또한 [7]에서는 과거에 비해 다양하고 복잡해진 비즈니스 프로세스의 기능들을 상당 부분에서 표현하지 못하기 때문에 그에 대한 타당성 검증이 제약적임을 볼 수 있다. 예를 들어, 프로세스 간의 통신, 이벤트 호출, OR 라우팅 등에 대한 표현 방식이 언급되어 있지 않다. 또한, 타당성 검증 측면에선 데드락, 동기화의 부족, 시작과 종료가 없는 노드 등은 검증이 가능하나 무한루프에 대한 검증방법은 언급되어 있지 않다.

2.2 그래프 축소 (Graph reduction)

그래프 축소는 데드락과 동기화의 부족 이상현상에 대해서만 검증 가능하다[1]. 프로세스 내에 남아있는 노드들에게 다섯 개의 축소 규칙을 반복적으로 적용하여 줄여 나가는 방식으로써 데드락과 동기화의 이상현상을 프로세스가 포함하고 있으면 프로세스 내의 전 노드가 삭제되지 않는다. 이 다섯 가지의 축소 규칙은 터미널 축소(Terminal Reduction), 순차 축소(Sequential Reduction), 인접 축소(Adjacent Reduction), 종료 축소(Closed Reduction) 그리고 중첩 축소(Overlapping Reduction)로 구성되어 있다. 이 방식의 장점으로는 그래프 축소 알고리즘의 복잡도가 최악의 경우 $O(n^2)$로 상당히 성능이 좋다는 것이다. 하지만 루프를 포함하고 있으면 이 방식을 사용할 수 없다는 것과 이 기법 역시 단순한 기능을 표현할 수 있는 표기법에 대해서만 검증이 가능하다.

2.3 논리 기반 접근 방법 (Logic-based approach)

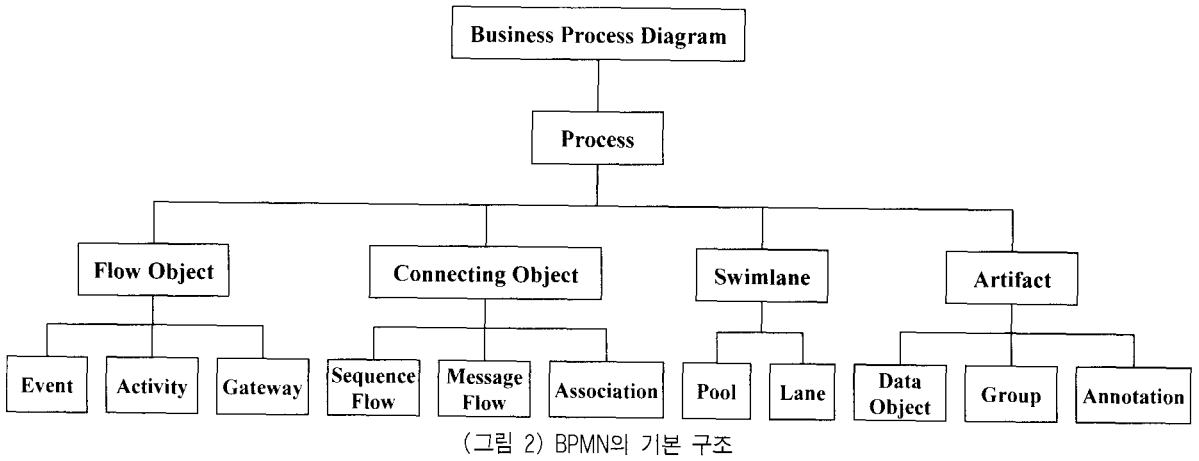
논리 기반 접근 방식은 명제 논리학(propositional logic)을 사용해서 프로세스의 이상현상을 검증하는 방법으로써 프로세스 내의 노드들을 논리적 표현으로 변환 후 [12]에서 제안하는 프로세스 추론에 의해 간단한 논리적 표현으로 줄여 나가는 방식이다. 이 방식은 알고리즘의 복잡도가 $O(n^2)$로 성능이 좋고 대표적인 4가지 이상현상 전부를 검증을 할 수 있다는 장점이 있지만 중첩된 구조를 검증할 수 없고 단순한 구조만 검증 가능하다는 단점이 있다.

2.4 비즈니스 프로세스 검증 기법 비교

현재의 비즈니스 프로세스는 과거에 비해 다양하고 복잡

〈표 1〉 기존 검증 기법과의 비교 분석

검증 기법 / 이상현상 검증	Petri nets	그래프 축소	논리 기반 접근 방법	제안된 검증 기법
데드락	O	O	O	O
동기화의 부족	O	O	O	O
시작과 종료가 없는 액티비티	O	O	O	O
무한루프	X	X	O	O
이벤트 호출	X	X	X	O
OR 라우팅	X	X	X	O
프로세스 간의 통신	X	X	X	X



(그림 2) BPMN의 기본 구조

해짐에 따라 프로세스를 디자인할 때 사용되는 모델들도 많은 부분 추가되었다(예를 들면, 프로세스 간의 통신, 이벤트 호출, OR 라우팅 등). 그러나 위의 3가지 검증 기법을 보면 상당히 단순한 형식의 프로세스를 가정에 두고 있다. 또한, 대표적인 4가지 이상현상을 모두 처리할 수 있는 기법이 없다. 가장 대표적인 검증 기법인 Petri nets 기반의 방식 역시 무한 루프가 있는 경우를 처리할 수 있다는 언급이 없다. 본 논문에서는 복잡한 프로세스를 디자인 할 수 있는 강력한 표준화 모델인 BPMN을 기반으로 4가지 이상현상을 검증하는 기법을 제공한다. 그러나 프로세스 간의 통신은 제외하였다. 왜냐하면 프로세스 간의 통신은 다른 프로세스간의 동기화 및 데이터 흐름과 관련이 있기 때문에 본 논문의 컨트롤 흐름 측면에서의 검증 기법에서 벗어나기 때문이다. 본 논문은 단일 프로세스 내에서의 검증기법에 초점을 맞춘다. 앞에서 본 기존의 검증기법과 본 논문의 검증 기법에 대한 비교 분석은 <표 1>과 같다. 본 논문의 검증 기법은 컨트롤 플로우 측면에서 프로세스 간의 통신을 제외한 모든 부분에서 검증할 수 있음을 볼 수 있다.

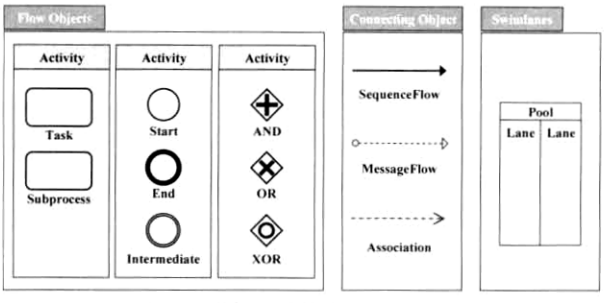
3. 배경 지식

3.1 BPMN 기본 구조

BPMN은 BPMI(Business Process Management Initiative)에서 제안한 비즈니스 프로세스에 대한 그래픽 표기법으로서 그래프 형식의 흐름 도표로 표현된다. BPMN의 목적은 비즈니스 프로세스를 정의하는 설계자와 사용자 그리고 이를 관리하는 관리자 모두가 쉽게 이해하고 사용하는데 있다. BPMN의 구조는 (그림 2)에서 보는 것처럼 최상위 단위에 비즈니스 프로세스 다이어그램(Business Process Diagram)이 있고 하나의 다이어그램은 다수의 프로세스로 구성될 수 있다. 프로세스는 하나의 업무를 수행하기 위한 규칙들을 정의하고 참여자들의 정보 전달을 정의한다. 프로세스는 크게 플로우 오브젝트(Flow Object), 연결형 오브젝트(Connecting Object), 스윔레인(Swimlane), 아티팩트(Artifact)로 나누어지며 이들은 다시 세부적인 요소로 나누어진다. 첫째, 플로우 오브젝트는 비즈니스 행동의 종류에 따라 구별된다. 프로세스 실행 중에 발생하는 사건의 행동을 나타내는 이벤트(Event), 실제적인 작업을 수행하는 액티비티(Activity), 그리고 프로세스의 흐름을 결정하는 게이트웨이(Gateway)로 나누어진다.

또한 액티비티는 다시 태스크(Task)와 서브프로세스(Sub-Process)로 나누어지는데, 서브프로세스는 다른 프로세스를 호출함으로써 주어진 작업을 다른 프로세스에 할당하며, 루프 형식에 따라 같은 행동을 여러 번 반복하거나(Standard Loop), 동시에 여러 번 행할 수 있다(MultipleInstance Loop). 둘째, 연결형 오브젝트는 순차 플로우(Sequence Flow), 메시지 플로우 (Message Flow), 연합 플로우(Association)로 나누어지는데, 순차 플로우는 프로세스 내부 개체들의 흐름을 나타내고, 메시지 플로우는 프로세스간 개체들의 흐름을 나타내며, 연합 플로우는 흐름과 관계없이 관계성을 가지는

개체들을 묶어주는 역할을 한다. 여기서 순차 플로우는 조건에 따라 게이트웨이와 함께 작동하여 엑티비티의 흐름을 제어할 수 있다. 셋째, 스윔레인은 프로세스를 대표하는 참가자의 역할을 나타내는 풀(Pool)과 프로세스를 사용하는 참가자의 역할을 나타내는 레인(Lane)으로 나누어진다. 마지막으로 아티펙트는 데이터 오브젝트(DataObject), 그룹(Group), 주석(Annotation)으로 나눠지는데 이것은 실제 프로세스의 흐름에는 영향을 주지 않는 요소로써 프로세스와 연관된 메타 정보를 기술하기 위해 이용된다. 프로세스 내의 컨트롤 플로우 흐름만을 체크하므로 흐름에 영향을 주지 않는 스윔레인, 아티팩트, 연결형 오브젝트의 메시지 플로우, 연합 플로우는 이 논문에서 언급하지 않는다. 위에서 설명한 각 오브젝트에 대한 그래픽 표기법은 (그림 3)에서 보는 것과 같다. (그림 2)에서와 같이 프로세스 내에 존재할 수 있는 플로우 오브젝트, 연결형 오브젝트, 스윔레인에 따라 (그림 3)과 같이 그래픽 표기법이 분류되어 있다.



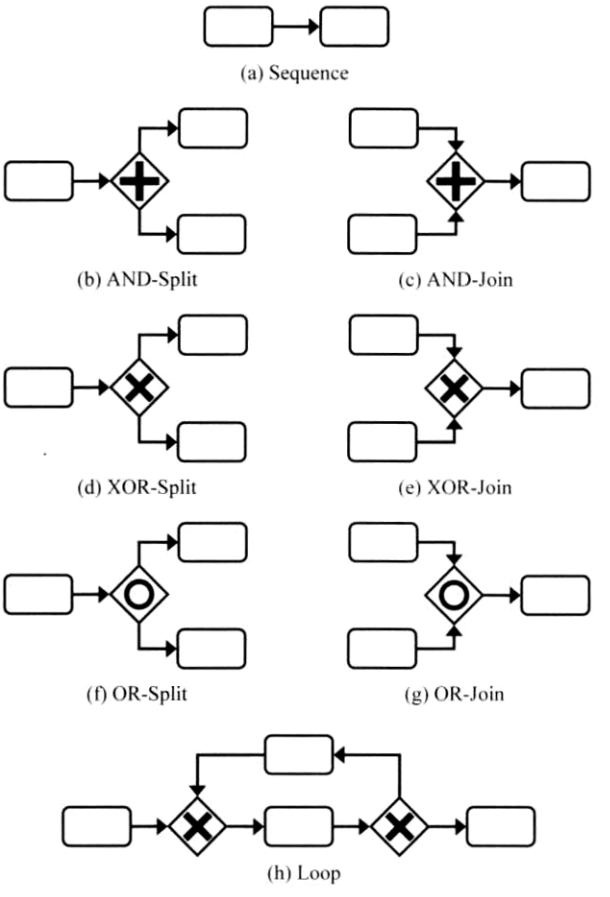
(그림 3) BPMN의 주요 엘리먼트

3.2 BPMN의 기본 구성 요소

프로세스 내의 기본 구성 요소는 (그림 4)에서와 같이 8가지로 이루어져 있다.

(a) Sequence: 특정 플로우 오브젝트에서 이와 연결된 다음 플로우 오브젝트로의 이동.
(b) AND-Split: AND 게이트웨이에서 나가는 방향에 있는 모든 플로우 오브젝트를 동시에 실행시킴.
(c) AND-Join: AND 게이트웨이로 들어오는 방향에 있는 모든 플로우 오브젝트의 객체를 동기화시킴.
(d) XOR-Split: XOR 게이트웨이에서 나가는 방향에 있는 모든 플로우 오브젝트 중 단지 하나의 플로우 오브젝트만을 실행시킴.
(e) XOR-Join: XOR 게이트웨이로 들어오는 흐름을 동기화하지 않고 바로 다음 단계로 진행시킨다. 즉, 두 개의 토큰이 특정 시간간격으로 들어왔을 때 첫 번째에 도착한 토큰을 바로 다음 플로우 오브젝트에 넘겨준다. 그런 다음에 두 번째에 도착한 토큰은 다음 플로우 오브젝트에 넘겨주지 않고 버리는 역할을 한다.
(f) OR-Split: OR 게이트웨이에서 나가는 방향에 있는 모든 플로우 오브젝트 중 특정 조건을 만족시키는 플로우 오브젝트만을 실행시킴.
(g) OR-Join: OR 게이트웨이의 이전 Split 조건에서 만족되어 들어온 모든 흐름을 동기화시킴.
(h) Loop: 특정 조건을 만족할 때까지 특정 범위를 반복 수행함.

위의 8가지 구성요소를 복합적으로 사용하여 다양한 비즈니스 프로세스를 디자인할 수 있다. 하지만 WfMC는 (a)-(e) 그리고 (h)만을 워크플로우 기본 구성 요소로 정의하고 있다[13]. 그리고 다른 여러 논문에서도 OR 게이트웨이를 언급하지 않는 경우가 많다. 그 이유는 구현하기 어렵고 AND와 XOR 게이트웨이를 혼합해서 사용하면 가능하기 때문이다.

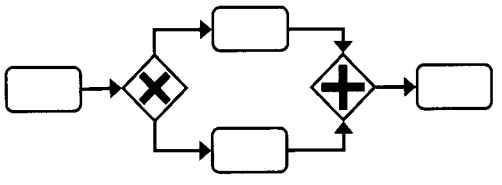


(그림 4) 프로세스 기본 구성 요소

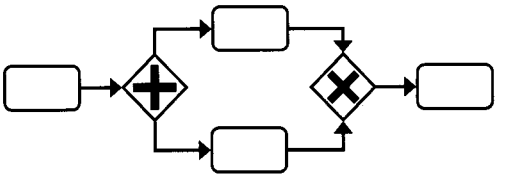
하지만 여러 상업화된 응용프로그램에서는 OR 게이트웨이를 지원하고 있으며 BPMN에서도 프로세스 기본 구성요소로 정의하고 있기 때문에 본 논문에서는 OR 게이트웨이까지도 포함하는 프로세스 검증 방법을 제시하고 있다. 본 논문의 초점은 컨트롤 플로우에 대한 타당성 검증이기 때문에 데이터의 흐름을 고려해야 하는 콤플렉스 게이트웨이는 생략한다.

3.3 대표적인 비즈니스 프로세스 이상현상

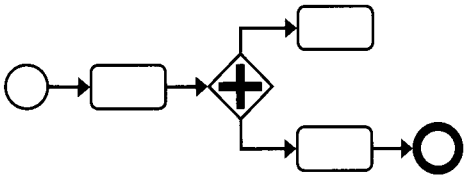
비즈니스 프로세스를 디자인할 때 발생할 수 있는 대표적인 이상현상은 (그림 5)에서와 같이 4가지로 분류된다.



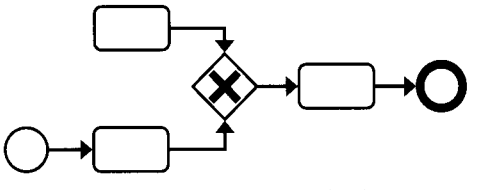
(a) Deadlock



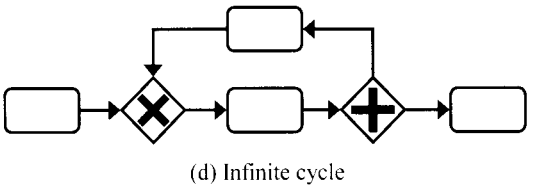
(b) Lack of synchronization



(c-1) Activities without termination



(c-2) Activities without activation



(d) Infinite cycle

(그림 5) 비즈니스 프로세스 이상현상

(a) 데드락: 특정 단계에서 다음 단계로 진행할 수 없을 경우를 말한다.

(b) 동기화의 부족: 의도치 않게 특정 태스크가 여러 번 수행될 경우를 말한다.

(c) 시작과 종료가 없는 엑티비티: 특정 엑티비티가 종료로 도달할 수 없을 경우 종료가 없는 엑티비티라고 하고 시작 이벤트에 의해서 실행될 수 없는 엑티비티를 시작이 없는 엑티비티라고 말한다.

(d) 무한루프: 특정 범위 내의 태스크들을 무한하게 수행할 경우를 말한다.

(그림 5)-(a)를 보면 XOR Split에 의해 두 개의 출력 경로 중 단지 하나의 경로만이 선택되어 진행하게 된다. 하지만 이후 AND Join의 경우 두 개의 입력 경로 모두를 기다리기 때문에 한쪽으로만 들어오게 되면 다음 단계로 진행할 수 없는 현상이 발생하게 된다. XOR Split 대신 OR Split를 사용할 경우에도 문제가 발생할 수 있다. 그 이유는 조건에 따라 OR Split도 단지 하나의 출력 경로만이 선택될 수 있기 때문이다. 이러한 경우를 '데드락'이라고 한다. (그림 5)-(b)를 보면, AND Split에 의해 다음 두 출력 경로가 동시에 선택되어 프로세스가 진행된다. 그 후에 위치한 XOR Join을 만나게 되면 동기화되지 않고 한쪽 경로라도 들어오는 즉시 다음 단계로 진행시킨다. 이러한 경우 의도치 않게 다음 태스크가 여러 번 수행될 수 있다. 같은 맥락으로 하나 이상의 경로가 선택될 수 있는 OR Split인 경우 역시 문제가 발생할 수 있다. 이러한 경우를 '동기화의 부족'이라고 한다. (그림 5)-(c-1)를 보면, 종료 이벤트로 도달하지 않는 엑티비티가 존재한다. 그리고 (그림 5)-((c-2)를 보면 시작 이벤트에 의해서 실행될 수 없는 엑티비티가 존재함을 볼 수 있다. 이러한 경우를 '시작과 종료가 없는 엑티비티'라고 말한다. 그러나 '시작과 종료가 없는 엑티비티'에 대한 검증은 이 논문에서 제외한다. 그 이유는 BPMN 1.0 Specification에서 "Flow Object Connection Rules"로 정의하였기 때문이다. 마지막으로 (그림 5)-((d)를 보면 무한루프가 발생하고 있는 구조이다. 또한, 처음에 XOR Join에 의해 다음 태스크가 실행되고 나서 AND Split을 만나게 되면 두 개의 출력 경로로 컨트롤이 진행된다. 그런 다음에 XOR Join을 만나게 되기 때문에 무한 루프가 발생된다. 결과적으로 특정 태스크가 지속적으로 수행된다. 이러한 경우를 '무한루프'와 '동기화의 부족' 이상현상이 발생했다고 한다. (그림 5)-(d)는 간단한 무한 루프의 예이지만 실 세계의 프로세스는 보다 복잡한 구조로 이루어져 있다. 이렇게 복잡한 프로세스가 구성되어 있을 때는 손쉽게 이러한 이상현상을 발견할 수 없으므로 검증 기법을 통해 체계적으로 이상현상을 검출할 필요가 있는 것이다.

4. 정형화된 컨트롤 플로우 프로세스 (Well-formed Control Flow Process)

이번 장에서는 BPMN1.0 스펙에서 정의한 "Flow Object Connection Rules"를 따르는 프로세스로부터 플로우 오브젝트 변환 과정을 통해 정형화된 컨트롤 플로우 프로세스로 변환하는 방법을 제시한다. 정형화된 컨트롤 플로우 프로세스는 본 논문에서 제시하는 컨트롤 플로우 경로를 추출할

때 발생하는 모호성을 제거하기 위한 목적으로 사용된다.

4.1 정형화된 컨트롤 플로우 프로세스

"Flow Object Connection Rules"를 준수하는 프로세스는 스펙에서 정의된 연결 규칙만을 따르게 되는데 이러한 프로세스로부터 흐름 경로를 추출할 때 모호성이 발생될 수 있다. 발생되는 모호성은 이상현상과는 다른 문제점으로서 흐름 경로를 일관성있게 추출하는데 방해되는 요소들을 의미한다. 이에 본 논문은 정형화된 컨트롤 플로우 프로세스를 아래와 같이 정의한다.

정의 1: 정형화된 컨트롤 플로우 프로세스(Well - formed Control Flow Process)

1) 모든 엑티비티는 이전 플로우 오브젝트로부터 들어오는 순차 플로우와 다음 플로우 오브젝트로 향하는 순차 플로우를 각각 최대 한 개씩으로 제한한다.
2) 모든 게이트웨이는 Split과 Join 중 하나의 용도로만 사용한다.
3) 프로세스는 엑티비티, 게이트웨이 그리고 순차 플로우로만 구성된다.

BPMN에 의해 설계된 초기 프로세스는 아래의 5가지 변환과정을 통해서 정의 1을 만족하는 정형화된 컨트롤 플로우 프로세스로 변환된다.

1) 한 개 이상의 플로우 오브젝트와 연결된 태스크의 경우 태스크와 게이트웨이로 분리시킴.
2) Join과 Split 두 가지 역할을 동시에 수행하는 게이트웨이의 경우 한 가지 역할만을 수행할 수 있도록 두 개의 게이트웨이로 분리한다.
3) 링크 이벤트의 경우 순차 플로우와 직접 연결한다.
4) 중간 이벤트가 엑티비티에 부착되어 있는 경우 엑티비티와 XOR 게이트웨이로 변환한다.
5) 3)과 4)의 변환 과정 후 남아있는 모든 이벤트들을 엑티비티로 변환한다.

4.2 엑티비티의 변환

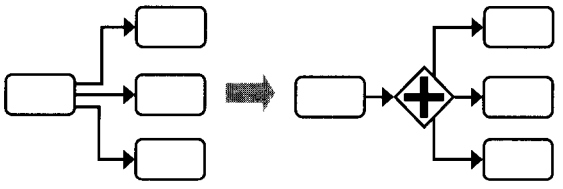
다수의 순차 플로우를 가진 엑티비티는 게이트웨이를 이용하여 하나의 순차 플로우만을 가지도록 변환한다. (그림 6)-(a)의 경우 엑티비티에 다수의 순차 플로우가 나가는 방향으로 연결되어 있다. 이때 다음에 수행되는 엑티비티들은 동시에 진행되므로 AND Join 게이트웨이와 연결된 것과 동일하다. (그림 6)-(b)의 경우는 XOR Split에 의해 연결된 것과 동일하다.

4.3 게이트웨이의 변환

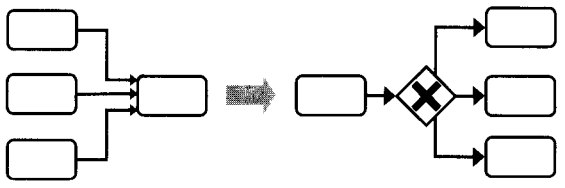
하나의 게이트웨이가 두 가지 역할을 수행하면 한 가지 역할만을 수행하도록 변환한다. (그림 7)처럼 두 가지 역할을 동시에 수행하는 게이트웨이를 같은 타입의 두 개의 게이트웨이로 분리함으로써 각각 한 가지 역할만을 수행하도록 한다.

4.4 이벤트의 변환

컨트롤 플로우 검증 측면에서 바라보면 링크이벤트와 엑

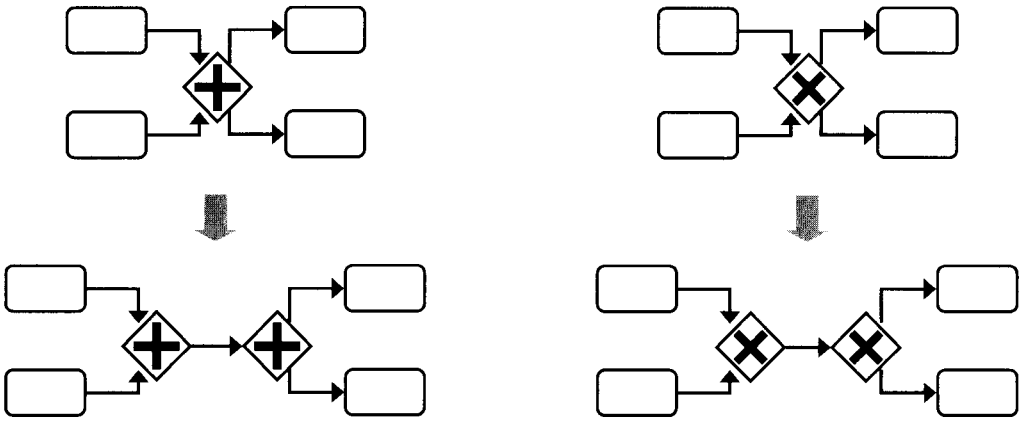


(a) Transformation for activities with out-direction sequence flows

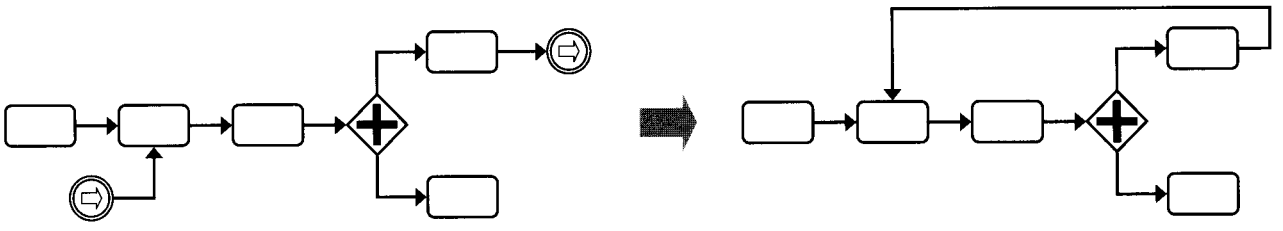


(a) Transformation for activities with in-direction sequence flows

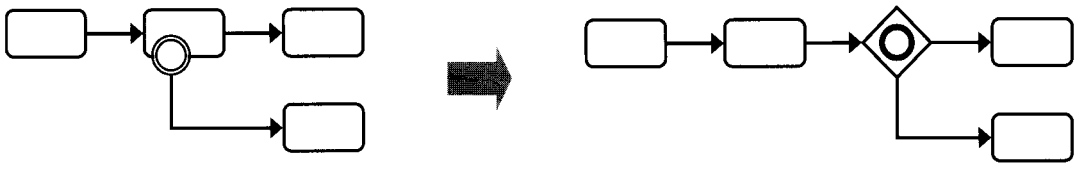
(그림 6) 다수의 순차 플로우를 가진 엑티비티를 단일 순차 플로우를 갖도록 변환



(그림 7) 두 가지 역할을 하는 게이트웨이를 하나의 역할을 하도록 변환

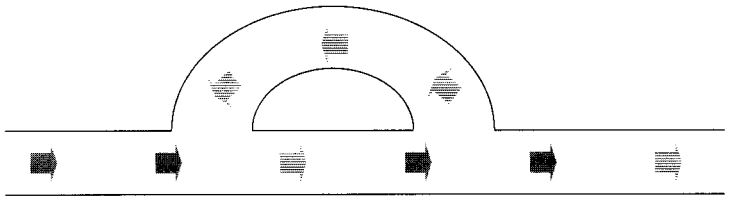


(그림 8) 링크 이벤트 삭제 후 직접 연결



(그림 9) 엑티비티에 부착된 이벤트를 삭제 후 OR 게이트웨이로 연결

(a) Single Stream



(b) Composite Stream

(그림 10) 단일 스트림과 복합 스트림

티비티에 부착된 이벤트를 제외한 모든 이벤트는 엑티비티와 동일하게 취급되어도 무관하다. 왜냐하면 본 논문은 컨트롤 플로우에 대한 타당성 검증에 초점을 두고 있기 때문이다. 먼저 링크 이벤트와 엑티비티에 부착된 이벤트를 변환하고 남아 있는 모든 이벤트들을 엑티비티로 변환함으로써 정형화된 비즈니스 프로세스는 엑티비티와 게이트웨이로만 구성된다.

4.4.1 링크 이벤트의 변환

링크 이벤트는 두 종류의 이벤트가 한 쌍으로 사용된다. 나가는 방향의 순차 플로우가 없는 링크 이벤트와 들어오는 방향의 순차 플로우가 없는 링크 이벤트가 같은 ID를 공유하여 마치 순차 플로우로 직접 연결되어 실행되는 것과 동일하게 수행된다. 그러므로 (그림 8)에서 보는 것처럼 두 이벤트를 삭제하고 바로 순차플로우로 직접 연결한다.

4.4.2 엑티비티에 부착된 이벤트의 변환

엑티비티에 부착된 이벤트의 경우는 (그림 9)와 같이 OR Split 게이트웨이를 이용하여 엑티비티와 이벤트를 구조적으로 분리시킨다. 의미상으로는 두 개의 구조가 다르지만 컨트롤 측면에서만 바라보면 OR Split 게이트웨이를 이용해서 분리시켜도 무관하다.

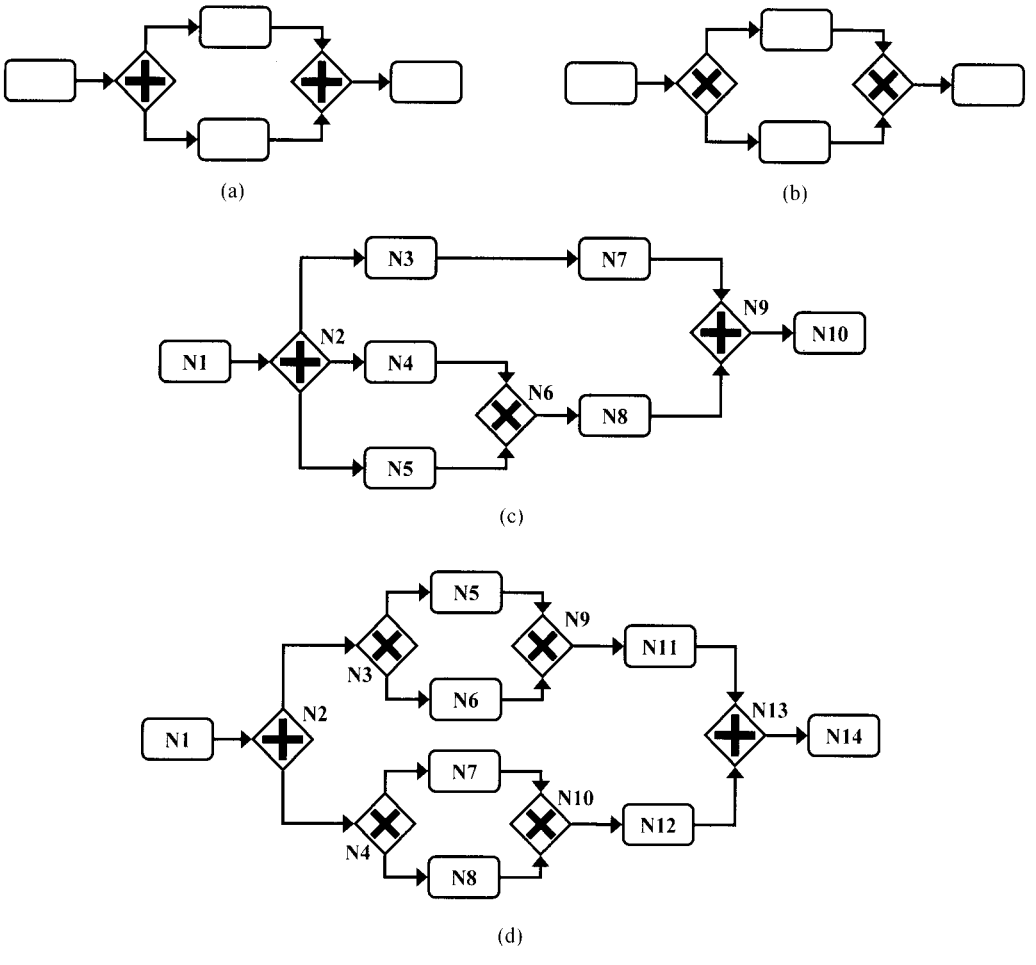
5. 비즈니스 프로세스 타당성 검증

이번 장에서는 비즈니스 프로세스를 디자인할 때 문제점이 발생할 수 있는 상황에 대해서 집중 분석하고 그 이상현상들을 검증할 수 있는 방법을 제시한다. 검증 접근 방식은 단일 스트림 상에서의 검증 방식과 복합 스트림 상에서 검증하는 방식으로 구분하여 접근하였다. 단일 스트림은 (그림 10)에서 보는 것과 같이 비즈니스 프로세스의 흐름이 한 방향으로만 진행되는 상황인 루프가 없는 프로세스를 의미한다. 복합 스트림은 프로세스의 흐름이 한 방향으로 흐르다가 역류해서 특정 구역에서는 두 흐름이 지날 수 있는 상황으로서 프로세스 내에 루프가 존재는 경우이다.

5.1 문제점 분석

5.1.1 단일 스트림 상에서의 문제점 분석

특정 Split 게이트웨이로부터 나온 두 개 이상의 경로가 서로 합쳐질 때 반드시 같은 타입의 Join 게이트웨이에서

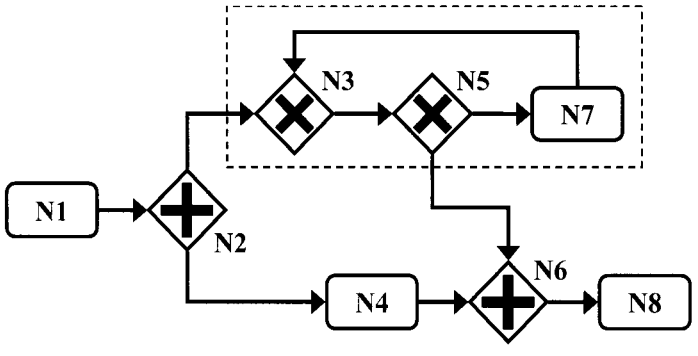


(그림 11) 단일 스트림 상의 비즈니스 프로세스

만나야 한다. 분리된 모든 경로가 최종적으로 한 곳에 모이기전에 그 중 최소 두 개의 경로라도 합쳐질 경우도 같은 타입의 Join 게이트웨이에서 만나야 한다. (그림 11)-(a)는 가장 기본적인 구조로써 AND Split 게이트웨이에서 나온 두 개의 경로가 최종적으로 AND Join 게이트웨이에서 만나는 것을 볼 수 있다. XOR의 경우인 (그림 11)-(b)도 마찬가지이다. 이 두 가지 구조의 경우는 아무런 문제가 발생하지 않는다. 그러나 서로 불일치가 생길 경우, (그림 5)비즈니스 프로세스 이상현상에서 보는 것처럼 게이트웨이의 불일치로 인한 데드락이나 동기화의 부족 현상이 발생함을 볼 수 있다. 이 보다 좀더 복잡한 구조를 가진 (그림 11)-(c)를 보면 N2에서 N3, N4, N5로 분리되고 최종적으로 같은 AND 타입의 게이트웨이인 N9에서 합쳐지는 것을 볼 수 있다. 또한, N9로 가는 도중에 세 개의 경로 중 두 개의 경로가 N6에서 합쳐진다. 이 역시 같은 타입의 AND 게이트웨이여야 한다. 그 이유는 이 두 개의 경로 역시 AND의 영향을 받고 있기 때문이다. 그러나 여기에서는 XOR 게이트웨이로 합쳐지기 때문에 아직 AND의 영향을 받고 있는 두 개의 경로는 반드시 N6에 모두 들어오게 되고 N6 이후에 놓인 액티비티 N8은 2번 실행하게 되는 경우가 발생하게 된다. 이러한 현상을 동기화의 부족이라고 한다. (그림 11)-(d)를 보면 N2는 두 개의 경로로 분리되고 최종적으로 N13에서 합쳐진다. 그리고 최종 게이트웨이인 N13으로 도달하기 전에 N9에서 합쳐짐을 볼 수 있다. 그러나 이 경우엔 N2에서 나온 경로에 대해 합치는 것이 아니라 N3에서 나온 두 개의 경로에 대해 합치는 역할을 수행한다. 따라서 N9는 N3의 경로에 의해 생긴 2개의 경로에 대해 합치는 역할을 수행하므로 N3와 같은 타입의 게이트웨이여야 한다.

5.1.2 복합 스트림 상에서의 문제점 분석

올바른 루프는 루프 내에 반드시 두 종류의 게이트웨이를 포함해야만 한다. 왜냐하면 하나의 게이트웨이는 루프를 형성하는데 사용되며 다른 하나는 루프를 벗어나는 용도로 사용되기 때문이다. (그림 12)를 보면, N3은 루프 형성에 N5는 루프를 벗어날 때 사용됨을 볼 수 있다. 만약에 N5 역할을 하는 게이트웨이가 존재하지 않으면 루프를 빠져 나갈 수 있는 길이 없으므로 무한루프가 발생하게 되고 N3이 없으면 루프를 형성할 수 없게 된다. 또한 이 두 가지의 게이트웨이는 Join과 Split의 한 쌍으로 존재하는 것이 아니라 독립적으로 사용된다. 그러므로 복합 스트림이 존재하는 프로세스의 경우 에는 게이트웨이의 존재 여부를 반드시 파악해야 한다.



(그림 12) 복합 스트림 상의 비즈니스 프로세스

위에서처럼 두 개의 게이트웨이가 잘못 사용될 경우도 존재할 수 있다. 만약 N3가 AND Join 게이트웨이로 사용되면 데드락 현상을 발생하고 N5가 AND나 OR로 사용되면 N6이 여러 번 실행될 수 있는 동기화의 부족 현상을 초래할 수 있다.

5.2 컨트롤 플로우 경로 기반의 타당성 검증 기법

본 논문에서의 타당성 검증은 컨트롤 플로우 경로를 통해서 이루어진다. 컨트롤 플로우 경로는 정의 2에서 보는 것처럼 임의의 두 엘리먼트 사이에 존재하는 모든 경로들을 의미한다. 비즈니스 프로세스내에서 검증 대상이 되는 두 엘리먼트 사이의 컨트롤 플로우 경로를 추출한 다음에 본 논문에서 제안하는 컨트롤 플로우 경로기반의 타당성 검증 기법을 적용하게 된다.

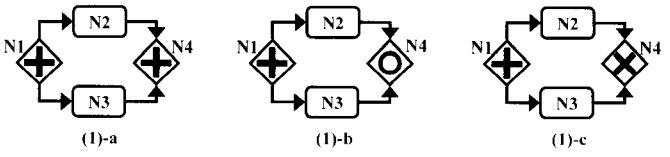
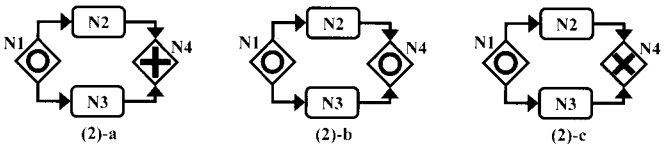
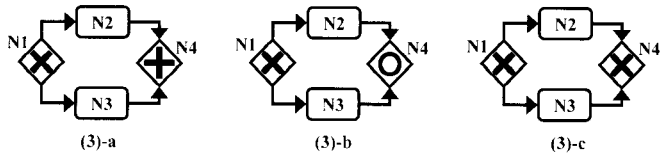
정의 2: 컨트롤 플로우 경로(Control Flow Path)
단일 프로세스내에 있는 임의의 게이트웨이 x로부터 y까지의 가능한 모든 경로를 컨트롤 플로우 경로 path(n) = path(n-1) ∧ … ∧ path(n-k)라고 정의한다. 단, y는 x로부터 시작되는 경로 상에 있는 엘리먼트이고 두 게이트웨이 사이에 가능한 컨트롤 플로우 경로가 k개 있다. 게이트웨이를 기준으로 <표 2>와 같이 컨트롤 플로우 경로가 생성될 수 있다.

<표 2>는 N1, N4의 두 게이트웨이의 종류에 따른 컨트롤 플로우 경로를 나타낸 것이다. 정의에 의해서 N1과 N4 사이의 가능한 경로를 모두 표현한다. <표 2>-(1)의 경우에는 N1이 AND Split이기 때문에 Path(1)은 두 경로 {N1, N2, N4}, {N1, N3, N4}가 ∧로 묶여지게 된다. OR Split으로 이루어진 <표 2>-(2)의 경우에는 OR 게이트웨이의 특성상 가능한 경로가 세 가지가 될 수 있다. 이때 {}는 경로가 없음을 의미한다. 마지막으로 XOR 게이트웨이로 Split이 발생될 경우에는 두 가지가 경로가 만들어질 수 있다.

5.2.1 게이트웨이 불일치 (Gateway mismatch)

게이트웨이 불일치는 정리 1에 의해서 검증될 수 있다. 두 게이트웨이 사이의 컨트롤 플로우 경로로부터 데드락과 동기화의 부족 이상현상 문제를 검증할 수 있다.

〈표 2〉 각 게이트웨이별 컨트롤 플로우 경로

(1)	 (1)-a (1)-b (1)-c	- Path(1) = {N1,N2,N4} ∧ {N1,N3,N4}
(2)	 (2)-a (2)-b (2)-c	- Path(1) = {N1,N2,N4} ∧ {N1,N3,N4} - Path(2) = {N1,N2,N4} ∧ {} - Path(3) = {} ∧ {N1,N3,N4}
(3)	 (3)-a (3)-b (3)-c	- Path(1) = {N1,N2,N4} ∧ {} - Path(2) = {} ∧ {N1,N3,N4}

정리 1: 게이트웨이 불일치 검증

특정 Split 게이트웨이 x로부터 Join 게이트웨이 y까지의 컨트롤 플로우 경로에 대해서 아래와 같은 룰에 의해서 이상현상을 검증할 수 있다. 즉, 각 Join 게이트웨이에 따라 ∧(교집합), ∨(합집합), -(차집합)와 같은 연산을 수행하였을 때 결과 집합이 x, y 원소를 포함하고 있으면 이상현상이 발생되지 않으며, 공집합이면 "데드락 이상현상"이 발생된다. 또한, 결과집합이 x, y를 제외한 다른 원소를 가지고 있으면 "동기화의 부족 이상현상"이 발생된다.

i) AND Join 게이트웨이: 각각의Path(n)에 대해서 Path(n-1) ∧ ⋯ ∧ Path(n-k) = {x, ⋯, y}이면 이상현상이 발생하지 않는다.

ii) OR Join 게이트웨이: Path(n)에 대해서 Path(n-1) ∨ ⋯ ∨ Path(n-k) = {x, ⋯, y}이면 이상현상이 발생하지 않는다.

iii) XOR Join 게이트웨이: Path(n)에 대해서 Path(n-1) - ⋯ - Path(n-k) = {x, ⋯, y}이면 이상현상이 발생하지 않는다. 단, |Path(n-1)| ≥ ⋯ ≥ |Path(n-k)|이다.

정리 1에 대한 타당성은 아래와 같이 증명될 수 있다. 모든 Split, Join 게이트웨이의 쌍에 대해서 정리 1을 적용하였을 때 이상현상을 검증할 수 있음을 보여준다.

- <표 2>의 (1)-a

Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {N2, N4, N5}

Join 게이트웨이가 AND이기 때문에 Path(1-1) ∧ Path(1-2) = {N2, N5}가 된다. 정리 1의 룰 i)에 의해서 타당한 비즈니스 프로세스가 된다.

- <표 2>의 (1)-b

Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {N2, N4, N5}

Join 게이트웨이가 OR이기 때문에 Path(1-1) ∨ Path(1-2) = {N2, N3, N4, N5}가 된다. 정리 1의 룰 ii)에 의해서 타당한 비즈니스 프로세스가 된다.

- <표 2>의 (1)-c

Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {N2, N4, N5}

Join 게이트웨이가 XOR이기 때문에 Path(1-1) - Path(1-2) = {N3}가 된다. 정리 1의 룰 iii)에 의해서 N2, N5가 결과집합의 원소가 아니기 때문에 이상현상을 가지고 있는 비즈니스 프로세스이다. 특히 결과집합이 공집합이 아니기 때문에 정리 1에 의해 "동기화 부족 이상현상"을 가지고 있다.

- <표 2>의 (2)-a

Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {}

Path(2) = Path(2-1) ∧ Path(2-2) = {} ∧ {N2, N4, N5}

Path(3) = Path(3-1) ∧ Path(3-2) = {N2, N3, N5} ∧ {N2, N4, N5}

Join 게이트웨이가 AND이기 때문에 Path(1-1) ∧ Path(1-2) = {}, Path(2-1) ∧ Path(2-2) = {}, Path(3-1) ∧ Path(3-2) = {N2, N5} 가 된다. 정리 1의 룰 i)에 의해서 Path(1)과 Path(2)의 경우에는 "데드락 이상현상"을 가지고 있지만 Path(3)의 경우에는 타당한 비즈니스 프로세스이다. 이와 같은 현상은 OR Split 게이트웨이에 연결된 순차 플로우의 컨디션에 따라 순차 플로우에 연결된 모든 플로우 오브젝트로 토큰이 전달될 수 있기 때문이다.

- 표 2의 (2)-b

Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {}

Path(2) = Path(2-1) ∧ Path(2-2) = {} ∧ {N2, N4, N5}

Path(3) = Path(3-1) ∧ Path(3-2) = {N2, N3, N5} ∧ {N2, N4, N5}

Join 게이트웨이가 OR이기 때문에 Path(1-1) ∨ Path(1-2) = {N2, N3, N5}, Path(2-1) ∨ Path(2-2) = {N2, N4, N5}, Path(3-1) ∨ Path(3-2) = {N2, N3, N4, N5} 가 된다. 정리 1의 룰 ii)에 의해서 Path(1), Path(2), Path(3)에 대한 결과 집합이 N2, N5 원소를 가지고 있기 때문에 타당한 비즈니스 프로세스이다.

- <표 2>의 (2)-c

Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {}

Path(2) = Path(2-1) ∧ Path(2-2) = {} ∧ {N2, N4, N5}

Path(3) = Path(3-1) ∧ Path(3-2) = {N2, N3, N5} ∧ {N2, N4, N5}

Join 게이트웨이가 XOR 이기 때문에 Path(1-1) - Path(1-2) = {N2, N3, N5}, Path(2-2) - Path(2-1) = {N2, N4, N5}, Path(3-1) - Path(3-2) = {N3 }가 된다. 정리 1의 룰 iii)에 의해서 Path(3)의 경우에는 "동기화 부족 이상현상"을 가지고 있지만 Path(1), Path(2)의 경우에는 타당한 비즈니스 프로세스이다. 이와 같은 현상은 <표 2>의 (2)-a와 같이 설명할 수 있다.

- <표 2>의 (3)-a

Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {}

Path(2) = Path(2-1) ∧ Path(2-2) = {} ∧ {N2, N4, N5}

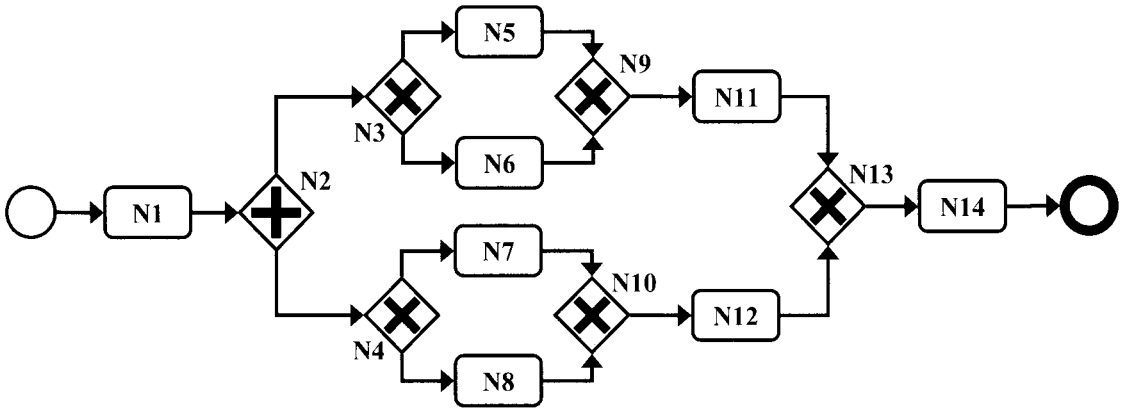
Join 게이트웨이가 AND이기 때문에 Path(1-1) ∧ Path(1-2) = {}, Path(2-1) ∧ Path(2-2) = {}가 된다. 정리 1의 룰 i)에 의해서 Path(1)과 Path(2)의 경우에는 "데드락 이상현상"을 가지고 있다.

- <표 2>의 (3)-b

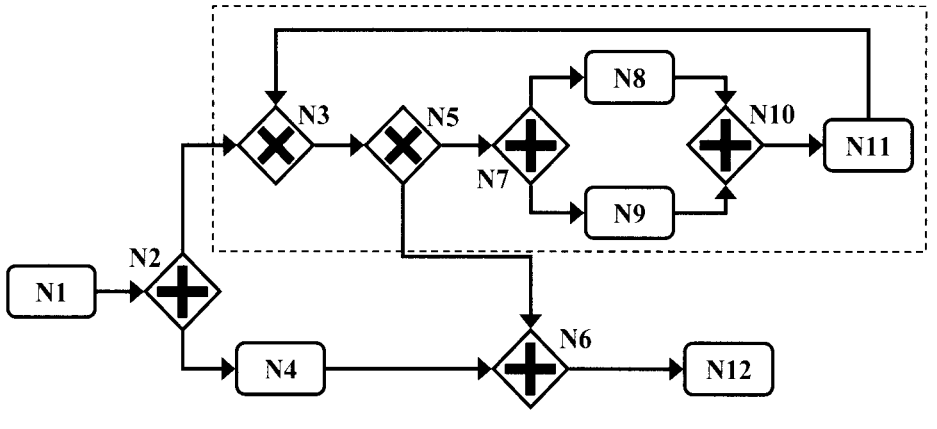
Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {}

Path(2) = Path(2-1) ∧ Path(2-2) = {} ∧ {N2, N4, N5}

Join 게이트웨이가 OR이기 때문에 Path(1-1) ∨ Path(1-2) = {N2, N3, N5}, Path(2-1) ∨ Path(2-2) = {N2, N4, N5} 가 된다. 정리 1의 룰 ii)에 의해서 Path(1), Path(2)에 대한 결과 집합이 N2, N5 원소를 가지고 있기 때문에 타당한 비즈니스 프로세스이다.



(그림 13) 단일 스트림 비즈니스 프로세스



(그림 14) 루프를 포함하고 있는 비즈니스 프로세스

- <표 2>의 (3)-c

Path(1) = Path(1-1) ∧ Path(1-2) = {N2, N3, N5} ∧ {}

Path(2) = Path(2-1) ∧ Path(2-2) = {} ∧ {N2, N4, N5}

Join 게이트웨이가 XOR이기 때문에 Path(1-1) - Path(1-2) = {N2, N3, N5}, Path(2-2) - Path(2-1) = {N2, N4, N5}가 된다. 정리 1의 룰 iii)에 의해서 Path(1), Path(2)에 대한 결과 집합이 N2, N5 원소를 가지고 있기 때문에 타당한 비즈니스 프로세스이다.

5.2.2 단일 스트림 상에서의 타당성 검증

정리 2: 쌍을 이루는 게이트웨이에 대한 경로 축소

임의의 한 쌍의 Join, Split 게이트웨이가 이상 현상이 없음이 검증되면 이 두 게이트웨이 사이의 경로는 제거될 수 있다.

정리 2는 간단하게 증명할 수 있다. (그림 13)에서 N3-N9를 보면 XOR Split과 Join으로 이루어져있기 때문에 두 게이트웨이 사이에는 이상현상이 없다. 결과적으로 N11 플로우 오브젝트까지 한 개의 토큰이 반드시 넘어 오게 된다. 즉, N3-N9를 제거하여도 N11 플로우 오브젝트는 토큰을 받을 수 있다. 이를 통해서, 정리 2는 단일 스트림 상에서의 복잡한 비즈니스 프로세스를 단순화 함으로써 경로 추출의 복잡도를 줄이는 역할을 한다.

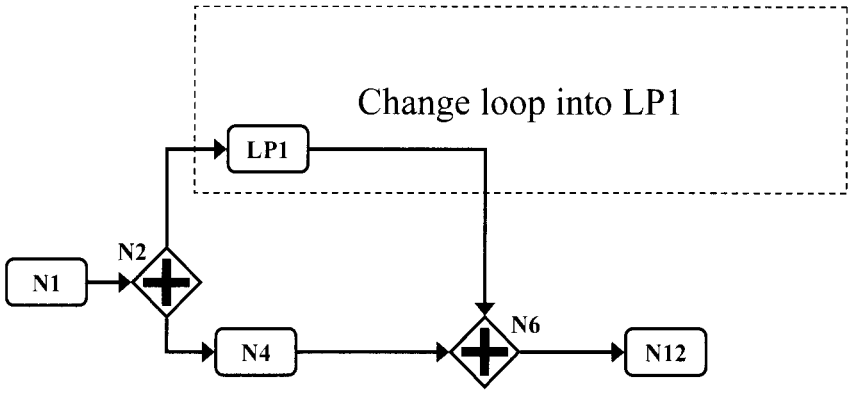
정리 2를 적용하는 예를 (그림 13)을 보고 설명한다. (그림 13)의 비즈니스 프로세스는 <표 2>의 게이트웨이 쌍들의 집합으로 구성되어 있다. 이런 구조는 기본이 되는 게이트웨이 쌍들을 먼저 검증함으로써 전체 비즈니스 프로세스 타당성을 검증할 수 있다. N3 - N9, N4 - N10은 정리 1에 의해서 타당한 게이트웨이 쌍임을 알 수 있다. 또한 정리 2에 의해서 제거한다. 그러면 N2-N13사이의 경로만을 고려하면 된다. 즉 Path(1) = {N2, N11, N13} ∧ {N2, N12, N13}이 되고 N13이 XOR Join이기 때문에 {N2, N11, N13} - {N2, N12, N13} = {N11}가 되어 "동기화 부족 이상현상"임을 알 수 있다.

5.2.3 복합 스트림 상에서의 타당성 검증

복합 스트림 상에서는 하나 이상의 루프가 존재하는 비즈니스 프로세스로서 (그림 14)와 같다. 이러한 루프도 정리 2의 경로 축소와 비슷한 방법으로 타당성을 검증할 수 있다. 정리 3은 루프 구조를 가지는 비즈니스 프로세스를 검증하기 위한 기법이다.

정리 3: 루프 축소

루프를 이루는 엑티비티가 x, y라 할 때 x, y사이에 이상현상이 없다고 검증되면 이들 사이의 모든 엑티비티를 제거하고 임의의 태스크로 연결한다. 단, x와 y사이에 루프를 벗



(그림 15) 루프를 제거한 비즈니스 프로세스

어날 수 있도록 하는 XOR 또는 OR Split이 하나 이상 존재해야만 한다.

정리 3에서 보는 것처럼 비즈니스 프로세스 상에서 루프를 이루는 x, y 엑티비티를 찾고 이들 엑티비티 사이의 이상현상을 먼저 검증하게 된다. 이상현상이 없음이 검증되고 루프 안에 루프를 벗어날 수 있도록 하는 XOR 또는 OR Split이 있으면 이 루프를 제거하고 임의의 이름을 가지는 태스크로 교체하는 것을 나타낸다. 이에 대한 증명은 그림을 통해서 설명한다. (그림 14)에서 보면 N3 - N11이 루프임을 나타낸다. 이 루프의 타당성 검증은 N3부터 시작해서 N11까지 이루는 경로를 정리 2에 의해서 검증하면 N7 - N10은 이상현상이 없기 때문에 제거 되어서 Path(1) = {N3, N5, N11}만 남게 된다. 즉, 하나의 경로를 가지는 일반적인 프로세스가 되기 때문에 이상현상이 없는 루프이다. 만약 N5가 없는 루프이면 N3부터 N11까지 무한 루프 이상현상을 가지게 된다. 그렇기 때문에 N5를 통해서 루프를 벗어날 수 있도록 해야 된다. 결과적으로 루프를 (그림 15)와 같이 제거하고 LP1로 교체하게 되면 루프를 벗어났을 때의 모습이 되기 때문에 N1-N12사이의 컨트롤 플로우 경로는 (그림 14)와 같은 모습을 가지게 된다.

5.2.4 비즈니스 프로세스 타당성 검증 알고리즘의 절차

정리 1, 2, 3에 의한 비즈니스 프로세스 타당성 검증 알고리즘의 절차는 아래와 같다.

(1) Start Event로부터 순차플로우를 따라 비즈니스 프로세스를 탐색한다.
(2) 탐색 도중에 Split 게이트웨이를 만나면 컨트롤 플로우 경로 검증 기법을 시작하기 위해 경로를 기록한다.
(3) 경로를 기록 도중에 Join 게이트웨이를 만나게 되면 가장 가까운 Split 게이트웨이에 대해서 정리 2를 적용하여 이 쌍을 경로에서 제거한다. 그렇지 않으면 이상현상을 보고하고 종료한다.
(4) 탐색 도중에 루프를 이루는 엑티비티 x, y를 만나면 정리 3을 적용하여 루프를 제거하고 임의의 태스크로 교체한다. 그렇지 않으면 이상현상을 보고하고 종료한다.
(5) 비즈니스 프로세스의 End Event에 도달하면 종료한다.

알고리즘의 적용 예를 보면 (그림 13)의 비즈니스 프로세스에 대해서 다음과 같이 진행된다. 비즈니스 프로세스 탐색 순서는 N1, N2, N3, N5, N9, N6, N11, N4, N7, N10, N8, N10, N12, N13, N14이라고 가정한다.

i) N2: AND Split 게이트웨이 → Path(1) = {N2}
ii) N3: XOR Split 게이트웨이 → Path(2) = {N3}
iii) N5, N9: XOR Join 게이트웨이 → Path(2) = {N3, N5, N9} ∧ {}
iv) N6, N9: XOR Join 게이트웨이 → Path(3) = {} ∧ {N3, N6, N9}
v) N9: N9의 모든 가능한 경로가 추출되었기 때문에 컨트롤 플로우 경로 검증기법을 적용 ⇒ 타당한 게이트웨이 쌍이기 때문에 이 경로를 제거함
vi) N11: Path(1) = {N2, N11} → Path(2)와 Path(3)은 제거됨
vii) N13: XOR Join 게이트웨이 → Path(1) = {N2, N11, N13}
viii) N4: XOR Split 게이트웨이 → Path(2) = {N4}
ix) N7, N10: XOR Join 게이트웨이 → Path(2) = {N4, N7, N10} ∧ {}
x) N8, N10: XOR Join 게이트웨이 → Path(3) = {} ∧ {N4, N8, N10}
xi) N10: N10의 모든 가능한 경로가 추출되었기 때문에 컨트롤 플로우 경로 검증기법을 적용 -> 타당한 게이트웨이 쌍이기 때문에 이 경로를 제거
xii) N12: Path(1) = {N2, N11, N13} ∧ {N2, N12} → Path(2)와 Path(3)은 제거됨
xiii) N13: XOR Join 게이트웨이 → Path(1) = {N2, N11, N13} ∧ {N2, N12, N13}
xiv) N13: 더 이상의 경로가 없기 때문에 Path(1)에 대해서 컨트롤 플로우 경로 검증기법을 적용 → {N2, N11, N13} - {N2, N12, N13} = {N11}이기 때문에 "동기화 부족 이상현상"을 보고하고 종료.
xv) N14: N13이 AND Join 게이트웨이 이면 이상현상이 없기 때문에 N2 - N13까지의 경로를 제거 → Path(1) = {}
xvi) End Event: 알고리즘 종료

본 논문에서 제안하는 컨트롤 플로우 경로 기반의 알고리즘은 이상 현상이 없는 게이트웨이 쌍에 대해서 경로를 제거하면서 비즈니스 프로세스를 탐색하기 때문에 경로 추출에 대한 오버헤드를 줄일 수 있다. 결과적으로 전체 비즈니스 프로세스 내에 있는 모든 액티비티들을 한번만 탐색하면서 이상현상을 검증할 수 있기 때문에 O(n)의 복잡도를 가진다.

6. 결론 및 향후 과제

본 논문에서는 BPMN으로 비즈니스 프로세스를 디자인할 때 발생할 수 있는 대표적인 네 가지 이상현상들을 검증할 수 있는 여러 기법들을 제시했다. 검증방법은 프로세스를 플로우 오브젝트 변환 과정을 통해 정형화된 컨트롤 플로우 프로세스로 변환하고 컨트롤 플로우 경로를 이용하여 4 가지 검증 기법을 통해 다양한 이상현상을 검증하였다. 특히, OR 게이트웨이와 이벤트를 포함하고 있는 경우도 처리할 수 있으며 루프가 존재하는 구조에 대해서도 쉽게 검증할 수 있다. 또한, 비즈니스 프로세내에 있는 모든 엑티비티를 한번 탐색함으로써 이상현상을 검증할 수 있기 때문에 매우 효율적이다.

향후 과제로는 비즈니스 프로세스 간의 통신을 할 수 있는 협력 프로세스를 검증할 수 있도록 검증기법을 확장할 계획이다. 또한 이 논문에서 제시한 검증 기법을 적용한 툴을 구현할 예정이다.

참 고 문 헌

[1] W.Sadiq and M.E. Orlowska, Analyzing process models using graph reduction techniques, Information System 25(2), 2000.

[2] W.M.P. van der Aalst and Arthur H.M. ter Hofstede, Verification of workflow task structures: A Petrinet-based approach, Information Systems 25(1) (2000) 43-69.

[3] Businss Process Modeling Notation (BPMN) (version 1.0 - May 3, 2004).

[4] R. Eshuis and R. Wieringa, Verification support for workflow design with UML activity graphs, In Proceedings of the 24rd International Conference on Software Engineering, pp.166-176. 2002.

[5] OMG Unified Modeling Language Specification (version 1.4, 2001,9, http://www.omg.org.

[6] Shazia Sadiq, Data Flow and Validation in Workflow Modeling, Proccedings of the fifteenth conference on Australasian database - Volume 27 (ACM), 2004.

[7] W.M.P van der Aalst. Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques, Business Process Management, Models, Techniques, and Empirical Studies, 2000.

[8] W.M.P. van der Aalst, The application of Petri nets to workflow management, Journal of Circuits, Systems and Computers 8(1) (1998) 21-66.

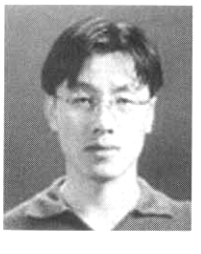
[9] W.M.P van der Aalst. Modeling and analysis of production systems using a Petri net based approach. Proceedings of the conference on Computer Integrated Manufacturing in the Process Industries, East Brunswick, USA, 1994.

[10] R. Eshuis and R. Wieringa, Comparing petri nets and activity diagram variants for workflow modeling - a quest for reactive petri-nets. In H. Ehrig, W.Reisig, and G.Rozenberg, (eds.) Petri Net Technologies for Communication Based Systems, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2002.

[11] W. Reisig and G. Rozenberg, Lectures on Petri Nets I: Basic Models, volume 1491 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1998.

[12] HENRY H. BI, Applying Propositional Logic to Workflow Verification, Information Technology and Management, Volume 5, 2004.

[13] WfMC, Workflow management coalition terminology & glossary (WFMC-TC-1011, Issue 3.0), Workflow Management Coalition (1999).



김 학 수

e-mail : hagsoo@cse.hanyang.ac.kr

2004년 한양대학교 전자 컴퓨터 공학과 (학사)

2006년 한양대학교 컴퓨터공학과 (석사)

2006년~현 재 한양대학교 컴퓨터공학과 박사과정

관심분야 : 데이터베이스, 시멘틱 마이닝, e-비즈니스

박 찬 희

e-mail : parkch@cse.hanyang.ac.kr

2006년 천안대학교 컴퓨터학과 (학사)

2006년~현 재 한양대학교 컴퓨터과학과 석사과정

관심분야 : e-비즈니스, 시멘틱 웹, RFID 시스템

설 주 영

e mail : sul0429@nate.com

2004년 세명대학교 소프트웨어학과(학사)

2006년 한양대학교 컴퓨터공학과 (공학석사)

2006년~현 재 LG CNS 근무

관심분야 : e-비즈니스, 시멘틱 웹, 센서 네트워크

손 진 현

e-mail : jhson@cse.hanyang.ac.kr

1996년 서강대학교 전산학과 (학사)

1998년 한국과학기술원 전산학과 (석사)

2001년 한국과학기술원 전자전산학과 (박사)

2001년 9월~2002년 8월 한국과학기술원 전자전산학과 박사 후 연구원

2002년 9월~현 재 한양대학교 컴퓨터공학과 조교수

관심분야 : 데이터베이스, e-비즈니스, 유비쿼터스 컴퓨팅, 임베디드 시스템