Journal of the Society of Korea Industrial and Systems Engineering
Vol, 29, No. 3, pp.119－127, September 2006.

비즈니스 프로세스 관리를 위한 BPML의 형식화†

이강배* · 유성열**

*동아대학교 경영정보과학부
**부산가톨릭대학교 경영학부, 교신저자

The Formalization of Business Process Modeling Language for Business Process Management

Kang-Bae Lee* · Sung-Yeol Yu**

*Division of Management Information Science, Dong-A University
**School of Business Administration, Catholic University of Pusan

In this paper, we present a systematic approach to translating BPML (Business Process Modeling Language) into the $\pi-Calculus$. BPML is an executable business process modeling language, like BPEL4WS (Business Process Execution Language for Web Services). It is difficult to find a systematic approach to formalizing these languages; but, by formalizing them, the behavior of the processes can be analyzed and compared so that optimal processes can be designed. For this formalization, we analyzed the activity types and contexts of BPML and suggested the definitions of semantics for each type and context by using the $\pi-Calculus$. In addition, we have shown the usefulness of our formalization scheme in that a typical order fulfillment process represented in BPML can be translated into the $\pi-Calculus$.

Keywords : BPM, BPML, Formalization, $\pi-Calculus$

1. 서 론

비즈니스 프로세스 관리(BPM ; Business Process Management)는 경영환경변화에 빠르게 적응하기 위한 기업들의 노력에 유용하게 사용될 수 있는 도구이자 방법론으로 최근 여러 기업들의 관심을 받고 있다. 또한 이를 구현한 시스템을 BPMS(Business Process Management Systems)라고 부르고 있다. BPMS는 워크플로우 자동화에서 출발한 시스템과 기업의 응용시스템 통합(EAI ; Enterprise Application Integration)으로부터 시작된 시스템으로 나뉘어 발전해오다 최근 들어 두 영역이 통합된 솔루션들이 나타나는 추세이다.

BPM은 프로세스를 발견, 설계, 적용, 실행, 상호작용, 운영, 최적화 및 분석하는 프로세스 라이프사이클을 완벽하게 관리하는 것이다. 이러한 비즈니스 프로세스 전체 라이프사이클 관리에 필수적인 도구가 비즈니스 프로세스 모형이라고 할 수 있다. 비즈니스 프로세스의 모형화는 기업 정보 시스템의 설계 및 기업의 업무 혁신 등을 위하여 이미 다양한 형태로 사용되어 왔다. 그러나 이러한 비즈니스 프로세스 모형들은, BPM에서 필요로 하는 "실행 가능한 코드로서의 비즈니스 프로세스"를 표현하는데 한계를 가지고 있다. 또한 공적으로 동의된 정형화된 의미론(formal semantics)을 가지지 못하는 한계도 가지고 있다. 이에 따라, 다양한 도구 또는 표현 기법을 사용하여 표현된 비즈니스 프로세스들이 특정 로직을 포함하고 있는지, 서로 다른 모형으로 표

† 이 논문은 2004학년도 동아대학교 학술연구비(신진교수)로 연구되었음.

현된 비즈니스 프로세스들이 동일한 것인지 등을 비교/분석하는 데 어려움이 있다.

따라서 본 연구에서는 비즈니스 프로세스를 표현할 수 있는 도구들 중 최근 관심을 끌고 있는 BPML(Business Process Modeling Language)을 $\pi-Calculus$라는 프로세스 대수(Process algebra)를 이용하여 정형화함으로써 이러한 한계들을 극복할 수 있는 비즈니스 프로세스 모델링 및 분석 방법을 제시하고자 한다. BPML은 BPM에 관한 표준을 제정하는 BPMI.org에서 발표한 모델링 언어로 BPEL4WS, WSCI 등과 함께 실행 비즈니스 프로세스를 모형화 하는 도구로서 주목을 받고 있다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 비즈니스 프로세스 모형과 프로세스 모형 형식화에 관한 기존 연구 동향에 관하여 살펴본다. 3장과 4장에서는 BPML에서 제시하고 있는 17가지 활동 유형과 $\pi-Calculus$의 주요 구문에 대하여 소개한다. 5장에서는 $\pi-Calculus$ 구문을 이용하여 BPML의 각 활동들을 표현하는 방법에 대해 제시한다. 이어 6장과 7장에서는 BPML 의미론의 활용 예와 활용 방안을 제시한다.

2. 관련 연구

2.1 비즈니스 프로세스 모형

비즈니스 프로세스 모형이 활용되는 분야는 다음과 같다[4].

- 프로그램 계획
- 지속적인 개선을 위한 기준선(baseline) 마련
- 지식유지 및 학습
- 프로세스의 가시화(Process Visualization)
- 훈련(training)
- 평가척도(metrics)의 틀
- 승인, 감사 및 평가
- 프로그램 실행(execution)

최근에는 위와 같은 활용 분야 이외에도 기업 간 비즈니스 프로세스의 통합을 통한 전체 공급사슬의 최적화 등을 위하여 기업 상호 간의 프로세스에 대한 이해를 증진하고 통합을 자동화하기 위한 도구로서의 역할도 필요하게 되었다. 그러나 지금까지 사용된 비즈니스 프로세스 모형들은 표현방법의 다양화로 인하여 동일한 관점 또는 기준으로 이해하기 어려운 문제점을 가지고 있다. 지금까지 개발되어 사용된 프로세스 모델링 도구들은 대략 다음과 같다[6].

- Flowcharts
- Data Flow Diagrams
- Control Flow Diagrams
- Functional Flow Block Diagrams
- Gantt/PERT Diagrams
- IDEF(Integration Definition Methods)
- UML

따라서 최근에는 비즈니스 프로세스 표현 방법을 표준화 하려는 노력들이 있어왔고, 특히, 웹 서비스를 위한 비즈니스 프로세스 지원을 위해, BPEL4WS, WSCI, 그리고 BPML등과 같은 웹 서비스 구성과 관련된 몇 가지 명세(specification)들이, 주요 벤더들의 지원을 받는 표준화 기구들에 의하여 제안되고 개발되어 왔다. 이들 중에서 BPML과 BPEL4WS 등이 기업 내부 시스템에서 사용되는 실행 비즈니스 프로세스를 묘사하는데 사용될 수 있다[9]. Moon *et al.*[9]은 BPML과 BPEL4WS의 중요성 및 활용성을 인정하고 이들 간의 변환 알고리즘을 제안한 바 있다.

BPML과 BPEL을 비교하고 분석하기 위한 많은 연구들이 있었다. Wohed *et al.*[14]은 웹 서비스 조합을 위한 언어로서 BPEL의 활용 가능성 등을 분석하였고, Aalst *et al.*[1]은 워크플로우 관리 시스템에서 발견되는 패턴들의 집합을 이용하여 BPML을 분석한 바 있다. Peltz, C.[10]는 BPEL, WSCI 그리고 BPML 간의 관계를 명확히 규명하고 있다. Peltz, C.에 연구에 언급된 것처럼, BPEL 실행 프로세스와 BPML 프로세스는 기업 시스템 내의 실행 비즈니스 프로세스를 묘사한다.

이와 같이 다수의 연구들에서 프로세스 모형화의 도구로서 BPML의 중요성 및 활용 가능성을 언급하고 있으므로, 본 연구에서는 이들 모형화 도구 중 BPML을 선택하여 형식화의 대상으로 삼고자 한다.

2.2 프로세스 모형의 형식화

Puhlmann and Weske[11]는 워크플로우의 행태적(behavioral) 관점을 묘사하기 위한 일반 프로세스 이론(general process theory) 즉, $\pi-Calculus$의 응용에 대하여 연구하였다. 그들은 순수한 정형 정의와 실행 의미론을 가지는 워크플로우 패턴들의 정형화 모임에 대한 이슈들을 탐색한 연구에서, 정형화가 워크플로우에 대한 이론적 관점의 미래 연구에 대한 기반이 될 뿐 만 아니라 패턴 기반의 워크플로우 실행, 추론, 시뮬레이션 등의 기초로 사용될 수 있음을 주장하였다.

프로세스 모형의 형식화에 관한 다른 연구로는 Lam and Padget[7]의 연구와 Dong and ShenSheng[5]의 연구

등이 있다. 두 연구 모두 UML의 형식화를 시도한 것으로 Lam and Padget은 UML 상태 다이어그램을, Dong and ShenSheng은 UML 활동 다이어그램의 형식화를 연구하였다.

이렇듯 비즈니스 프로세스 모형의 형식화에 관한 연구는 여러 비즈니스 모형을 대상으로 진행되어 왔다. 그러나 최근 주목받고 있는 BPML에 관한 직접적인 형식화 시도는 아직 없는 실정이다. BPML을 개발하여 발표한 BPMI.org에서는 BPML이 $\pi-Calculus$에 근간하여 개발되었다고 설명하고 있다. 하지만 아직까지 BPML과 $\pi-Calculus$간의 관계에 관한 명확한 정의나 설명이 없는 실정이며, 이에 따라 BPML로 표현된 프로세스에 대한 정형화된 분석 방법 및 도구는 매우 부족하거나 거의 없는 실정이다. 따라서 본 연구에서는 BPML의 정형화를 통하여 비즈니스 프로세스의 분석 및 설계 그리고 기업간 비즈니스 프로세스의 통합 등에 도움이 되고자 한다.

이렇게 $\pi-Calculus$의 실행 구문과 증명 능력에 기초한 BPML의 정형화를 통하여 BPML로 표현된 프로세스의 실행, 타당성 부여 및 시뮬레이션 등의 기초로 사용될 수 있도록 하고, 프로세스 모델의 행태적 분석이 용이하도록 하고자 한다.

3. BPML

BPML은 XML이 비즈니스 데이터를 모형화하기 위한 메타언어(meta-language)인 것처럼, 비즈니스 프로세스를 모형화하기 위한 메타언어이다. BPML은 처리 유한 상태 기계(a transactional finite-state machine)의 개념에 기반 하여 협업적이고(collaborative) 실행적인 비즈니스 프로세스의 추상화된 실행 모형(an abstracted execution model)을 제공한다. BPML에 대해 표준화를 진행하는 단체는 BPMI.org이다. 본 연구에 사용된 BPML표준은 BPMI.org에서 2002년 11월 13일 발표한 내용을 기준으로 한 것이다[3].

BPMI.org에서 발표한 BPML 명세(specification)는 비즈니스 프로세스와 프로세스를 지원하는 지원 개체(entity)들을 표현하기 위한 추상모형을 제공한다. 즉, 기업 비즈니스 프로세스의 모든 관점(다양한 복잡성을 가지는 활동들(activities of varying complexity), 처리(transactions) 및 보상처리(compensation), 데이터 관리, 동시처리(concurrency), 예외처리(exception handling), 운영 의미론(operational semantics) 등)에 대한 추상적인 그리고 실행 가능한(executable) 프로세스들을 표현하기 위한 정형화된(formal) 모형을 정의하고 있다. 또한 BPML은 XML Schema 형태의 문법(grammar)을 제공하는데, 이는 일관된 정의를 유지하면서도 다양한 시스템과 모델링 도구들에서도 활용 가능하도록 하기 위한 것이다.

그러나 BPML 자체로는 어떤 응용 프로그램을 위한 의미론(application semantics)도 정의하지 않는다. 다만, 일반적인 프로세스들에 대한 추상 모형과 문법만을 제공하여 기업 비즈니스 프로세스 정의, 복잡한 웹 서비스의 정의, 그리고 다양한 집단 간의 협업(collaboration) 등을 정의 할 수 있게 하고 있다.

본 연구에서 시도하는 정형화 역시 응용프로그램을 위한 의미론의 제공이 아니라 비즈니스 프로세스 자체의 분석을 위한 의미론을 확보하고자 하는 것이다.

BPML은 17가지 활동 유형(Activity Type) (10가지 단순(simple) 활동 유형과 7가지 복잡(complex) 활동 유형)들과 활동 문맥(Activity Context)들로 프로세스를 표현한다. 본 연구에서는 이들 활동 문맥과 활동 유형 각각을 정형화된 수학적 모형인 $\pi-Calculus$로 표현할 수 있도록 함으로써 BPML로 표현된 프로세스 모형을 직접 $\pi-Calculus$로 표현할 수 있는 방법을 제공하고자 한다. 먼저 BPML에서 정의하고 있는 10가지 단순 활동 유형을 살펴보면 다음과 같다.

(1) 행동(action) : 입력/출력 메시지들을 교환하는 것을 포함한 운영을 수행하거나 의뢰(invoke)
(2) 대입(assign) : 특성(property)에 새로운 값(value)을 할당
(3) 호출(call) : 하나의 프로세스를 실행하고 그 프로세스의 종료를 기다림
(4) 보정(compensate) : 명명된 프로세스들의 보정/보상(compensation)을 의뢰
(5) 지연(delay) : 시간 경과를 표현
(6) 무위(empty) : 아무것도 하지 않음
(7) 과실(fault) : 현재 문맥상 실패를 선언
(8) 신호발생(raise) : 신호(signal)를 발생시킴
(9) 산란(産卵, spawn) : 다른 여러 프로세스를 실행하고, 그들 프로세스가 종료되기를 기다리지 않으며, 원 프로세스를 시작
(10) 동기화(synch) : 신호에 따라 동기화

7가지의 복잡 활동 유형은 다음과 같다.
(1) 전체(all) : 활동들을 병행하여 실행
(2) 선택(choice) : 여러(multiple) 집합들 중의 하나로부터, 사건(event)에 대한 응답으로 선택된, 집합에 속한 활동들을 실행
(3) 각기(foreach) : 아이템 목록(list)로부터 각 아이템들을 하나씩 실행
(4) 순서(sequence) : 순차적으로 활동들을 실행
(5) 스위치(switch) : 여러(multiple) 집합들 중의 하나로부터, 조건 값이 사실(truth)인 것을 기준으로 선택된,

집합에 속한 활동들을 실행

(6) 까지(until) : 조건의 값이 사실인 상황에서 한 활동을 한번 또는 여러 번 실행

(7) 동안(while) : 조건의 값이 사실인 상황에서 한 활동을 0번 또는 여러 번 실행. 까지와 달리 한번도 실행하지 않을 수 있음

4. $\pi-Calculus$

$\pi-Calculus$에 대하여는 Milner의 저서[8]와 Sangiorgi and Walker의 저서[12]에 자세하게 소개되어 있다. 여기서는 본 연구와 관련되어 사용될 $\pi-Calculus$의 특성 및 표현 방법에 대하여 간단히 살펴본다. 본 논문에 사용된 $\pi-Calculus$ 표현 기호는 Sangiorgi and Walker[12]의 기호에 준한 것이다.

$\pi-Calculus$는 CCS(Calculus of Communicating Systems)로부터 발전된 것으로, 프로세스간의 상호작용을 표현한다. $\pi-Calculus$에는 두 가지 기본 개념이 포함되어 있다. 하나는 명칭(name)이다. 채널(또는 포트, channel or port), 변수들(variables), 데이터 모두 $\pi-Calculus$에서는 명칭으로 표현된다. CCS와 다른 점은 $\pi-Calculus$에서는 이것들을 구분하지 않는다는 것이다. 또 다른 개념은 프로세스(또는 에이전트, processes or agents)이다. 이는 시스템 내의 엔티티(entity)들을 표현한다. 프로세스들 간의 상호작용은 상호보완적인(complementary) 포트들의 쌍을 통하여 이루어진다.

$\pi-Calculus$에서 프로세스들은 <표 1>과 같은 구문을 사용하여 표현된다. 여기서 소문자는 명칭을 나타내고, 대문자는 프로세스들을 나타낸다.

$\pi-Calculus$에서 활동의 능력(capability)은 위의 구문에 정의된 네 가지 접두사들을 사용하여 다음과 같이 표현된다.

$$\pi ::= \bar{x}y \mid x(z) \mid \tau \mid [x=y]\pi \quad \cdots\cdots (1)$$

$\pi-Calculus$에서 프로세스 P와 프로세스의 특별한 하위 클래스인 요약(summation) M은 다음과 같이 표현된다.

$$P ::= M \mid P \mid P' \mid \nu z P \mid !P \quad \cdots\cdots (2)$$

$$M ::= 0 \mid \pi.P \mid M+M' \quad \cdots\cdots (3)$$

5. BPML의 의미론

본 장에서는 앞서 소개한 $\pi-Calculus$의 구문들을 사용하여 BPML의 각 활동들을 표현하는 방법에 대하여 제시한다.

Definition 1. BPML의 단순 활동 유형 중 행동은 다음과 같이 표현한다.

$$\pi_{action} ::= \bar{x}y \mid x(z) \mid \tau \mid [x=y]\pi_{action} \quad \cdots\cdots (4)$$

BPML에서 행동은 입력/출력 메시지를 교환하는 것을 포함한 하나의 연산을 수행하거나 이를 수행하는 행동을 의뢰하는 것을 의미한다. 따라서 $\pi-Calculus$의 행동 접두사 π에 대한 정의(식 (1))를 그대로 사용할 수 있다.

<표 1> $\pi-Calculus$ 주요 구문

구문 명칭	구문 표현	구문 설명
무행동 (Inaction)	0	아무것도 하지 않는 프로세스를 나타낸다.
입력 접두사 (Input Prefix)	$x(z).P$	명칭 x(채널)를 통하여 임의의 명칭 z를 수신하고 프로세스 P를 계속 전개(evolve)한다.
출력 접두사 (Output Prefix)	$\bar{x}y.P$	명칭 x(채널)를 통하여 명칭 y를 송신하고 프로세스 P를 계속 전개한다.
관찰 불가 접두사 (Unobservable Prefix)	$\tau.P$	활동 τ를 수행하고 프로세스 P를 계속 수행한다. 여기서 τ는 한 프로세스 내부 활동(internal action)에 대한 표현이다.
매치 접두사 (Match Prefix)	$[x=y]\pi.P$	x와 y가 동일한 명칭이면, $\pi.P$를 계속 전개한다. 그렇지 않으면 아무것도 수행하지 않는다.
합 (Sum)	$P+P'$	P와 P' 둘 중 하나의 프로세스만이 실행된다.
조합 (Composition)	$P \mid P'$	P와 P'은 서로 독립적으로 실행되고, 공유된 명칭(shared name)을 통하여 상호작용을 할 수 있다.
제한 (Restriction)	$\nu z P$	명칭 z의 범위가 프로세스 P로 제한된다. P의 컴포넌트들은 z를 사용하여 상호작용 할 수 있으나, 다른 프로세스와 z를 사용하여 직접 상호작용할 수는 없다. 단, z를 다른 명칭을 통하여 다른 프로세스에 송신함으로써 제한을 없앨 수 있다.
대체 (Replication)	$!P$	무한한 조합을 나타낸다. 즉, $!P = P \mid P \mid \cdots$을 표현하는 것이다. 따라서 $!P = P \mid !P$가 성립한다.

Definition 2. BPML의 단순 활동 유형 중 대입은 다음과 같이 표현한다.

$$\pi_{assign} ::= \bar{x}value \mid x(property)$$
$$\rightarrow \pi_{assign} ::= \{value/property\} \quad \cdots\cdots (5)$$

BPML에서 대입은 속성(property)에 새로운 값을 대입하는 것을 의미한다. $\pi-Calculus$의 변형 공리(reduction axiom)를 이용하면 이를 표현할 수 있다. 변형 공리는 다음과 같다[12, p.17].

$$P::=(\bar{x}y.P_1+M_1)\mid(x(z).P_2+M_2)$$
$$\rightarrow P::=P_1\mid P_2\{y/z\} \cdots\cdots (6)$$

여기서 프로세스 P의 두 컴포넌트들은 동일한 명칭(채널) x를 사용하여 임의의 명칭 z(일종의 매개변수 : placeholder)에 명칭 y를 대입하는 상호작용을 할 수 있고, 그 결과 변형 $\{y/z\}$가 발생하게 된다.

Definition 3. BPML의 단순 활동 유형 중 호출은 다음과 같이 표현한다.

$$\pi_{call}::=\bar{x}input.P_{called}.x(output) \cdots\cdots (7)$$

여기서 P_{called}는 다음과 같이 표현된다.

$$P_{called}::=\bar{x}(input).\tau_{called}.\bar{x}output \cdots\cdots (8)$$

호출은 하나의 프로세스를 실행하기를 의뢰하고 이의 종료를 기다리는 활동 유형을 의미한다. 따라서 해당 프로세스를 임의의 채널을 통하여 호출하고 이의 결과를 기다리는 능력들이 순차적으로 발생하는 형태로 표현할 수 있다.

Definition 4. BPML의 단순 활동 유형 중 보정은 다음과 같이 표현한다.

$$\pi_{compensate}::=\bar{x}compensate.P_{compensation} \cdots\cdots (9)$$

보정 활동은 진행된 프로세스 전체에 또는 프로세스의 일부 컴포넌트에 오류가 발생하여 해당 프로세스 전체 또는 일부를 취소하거나 다른 프로세스를 통하여 이의 오류를 정정하는 보정 프로세스를 호출 또는 산란하는 것이다. 보정 프로세스 $P_{compensation}$에는 보정 의뢰 메시지를 수신하는 부분이 포함된다.

Definition 5. BPML의 단순 활동 유형 중 지연은 다음과 같이 표현한다.

$$\pi_{delay}::=x(time).[time=time_{specified}]\tau \cdots\cdots (10)$$

지연 활동은 사전에 정의된 특정 시간까지 활동을 중단하고 기다리는 것이다. 따라서 시간 값을 수신하여 이를 사전에 정의된 시간 값과 비교하여 같을 경우에 나머지 활동 컴포넌트를 수행하는 형태로 표현될 수 있다.

Definition 6. BPML의 단순 활동 유형 중 무위는 다음과 같이 표현한다.

$$\pi_{empty}::=0 \cdots\cdots (11)$$

무위 활동은 $\pi-Calculus$의 무행동과 같이, 아무런 활동도 수행하지 않는 것이다.

Definition 7. BPML의 단순 활동 유형 중, 과실은 다음과 같이 표현한다.

$$\pi_{fault}::=\tau.P.x(code).[code=fault]\prod_{i\in R}\bar{y}_i fault.0 \cdots (12)$$

여기서 $\prod$는 조합의 확장을 의미한다. 즉, $\prod_{i=1}^{n}P_i::=P_1\mid P_2\mid\cdots\mid P_n$을 의미한다.

과실은 앞선 프로세스 또는 프로세스의 컴포넌트 수행 결과를 수신하여 이 값이 사전에 정의된 오류와 동일할 경우, 관련된 모든 프로세스들에 오류 메시지를 전송하고 즉시 종료하는 것을 의미한다.

Definition 8. BPML의 단순 활동 유형 중, 신호발생은 다음과 같이 표현한다.

$$\pi_{raise}::=x(signal).0 \cdots\cdots (13)$$

신호발생은 특정 신호를 발생시키고 즉시 종료하는 활동을 의미한다. 특정 신호에는 과실 신호 등이 포함될 수 있다.

Definition 9. BPML의 단순 활동 유형 중, 산란은 다음과 같이 표현한다.

$$\pi_{spawn}::=\bar{x}spwan.(P_{spawned}\mid\tau_{spawn}) \cdots\cdots (14)$$

산란은 한 프로세스의 실행을 의뢰하고, 이의 종료를 기다리지 않으며 자신의 나머지 능력을 동시에 실행하는 것을 의미한다.

Definition 10. BPML의 단순 활동 유형 중, 동기화는 다음과 같이 표현한다.

$$\pi_{synch}::=\prod_{i=1}^{n}P_i((x_1(synchsignal_1).\cdots.x_n(synchsignal_n)).\tau$$
$$+([synchsignal_1=synchcondition_1,\cdots,$$
$$synchsignal_n=synchcondition_n]).\tau+0) \cdots\cdots (15)$$

여기서 P_i는 다음과 같이 표현된다.

$$P_i::=\tau_i.\bar{x}_i synchsignal_i \cdots\cdots (16)$$

동기화는 동기화 활동에 앞서, 병행하여 실행된 관련 활동 또는 프로세스들이 동기화 신호를 발생시켜 송신하기를 기다렸다가 이를 모두 수신한 후 해당 능력을 실행하는 활동을 의미한다. 경우에 따라서는 동기화 신호를 사전에 정의된 값과 비교하여 진행여부를 결정하거나 종료할 수 있다.

아래에서는 BPML의 복잡 활동 유형에 대하여 정의한다.

Definition 11. BPML의 복잡 활동 유형 중, 전체는 다음과 같이 표현한다.

$$\pi_{all} :: = \prod_{i=1}^{n} P_i \cdots\cdots (17)$$

즉, 전체는 여러 활동 또는 프로세스를 동시에 실행하는 활동 유형으로 $\pi-Calculus$의 조합을 사용하여 직접 표현할 수 있다.

Definition 12. BPML의 복잡 활동 유형 중, 선택은 다음과 같이 표현한다.

$$\pi_{choice} :: = P.\left(\left(\sum_{i=1}^{n} x_i(c_i).P_i\right)+\prod_{i=1}^{n} x_i(c_i)P_i\right) \cdots\cdots (18)$$

여기서 P는 다음과 같이 표현된다.

$$P:: = \tau.\left(\sum_{i=1}^{n} \overline{x_i} c_i\right)+\left(\prod_{i=1}^{n} \overline{x_i} c_i\right) \cdots\cdots (19)$$

선택활동은, 앞서 실행된 프로세스에서 송신된 선택 메시지의 수신 여부에 따라 해당 프로세스를 선택적으로 실행한다. 후행되는 여러 프로세스들 중에서 하나의 프로세스만이 선택될 수 있도록 선택 메시지가 송신될 수도 있고, 후행하는 여러 프로세스 중 여러 프로세스를 선택하여 실행할 수 있도록 여러 선택 메시지를 송신할 수도 있다.

Definition 13. BPML의 복잡 활동 유형 중, 각기는 다음과 같이 표현한다.

$$\pi_{foreach} :: = (((P_1.\overline{x_1} complete_1).\cdots.(P_n.\overline{x_n} complete_n)).$$
$$x_1(complete_1).\cdots.x_n(complete_n).\tau) \cdots\cdots (20)$$

각기는 실행대상이 되는 프로세스의 집합에 속한 프로세스들을 하나씩 실행하는 것이다. 프로세스의 실행 순서는 바뀔 수 있다. 단, 집합에 속한 프로세스들을 한 번에 하나씩 모두 실행하여야 한다. 모든 실행대상 프로세스의 실행이 종료된 후 후행 활동 또는 프로세스를 실행할 수 있다.

Definition 14. BPML의 복잡 활동 유형 중, 순서는 다음과 같이 표현한다.

$$\pi_{sequence} :: = P_1.P_2.\cdots.P_n \cdots\cdots (21)$$

순서는 활동들을 순차적으로 실행하는 것을 의미한다.

Definition 15. BPML의 복잡 활동 유형 중, 스위치는 다음과 같이 표현한다.

$$\pi_{choice} :: = P.\left(\prod_{i=1}^{n} [condition = c_i] P_i\right).\pi_{dafault} \cdots\cdots (22)$$

여기서 P는 다음과 같이 표현된다.

$$P:: = \tau.\overline{x} condition \cdots\cdots (23)$$

스위치는 조건(condition)의 사실 여부에 따라 여러 활동 집합들 중 하나를 실행한다.

Definition 16. BPML의 복잡 활동 유형 중, 까지는 다음과 같이 표현한다.

$$\pi_{until} :: = !(\tau.(\overline{x} count | x(count_{increment})).$$
$$[count = count_{condition}]0) \cdots\cdots (24)$$

까지는 조건이 충족될 때까지 활동을 반복하여 실행하는 것이다. 활동을 한번 실행한 후 조건과의 비교 값을 변경$((\overline{x} count | x(count_{increment})) \rightarrow \{count/count_{increment}\}$하고 변경된 값을 조건 값과 비교하여 계속 실행여부를 결정하게 된다.

Definition 17. BPML의 복잡 활동 유형 중, 동안은 다음과 같이 표현한다.

$$\pi_{until} :: = !((\overline{x} count | x(count_{increment}).$$
$$[count = count_{condition}]\tau) \cdots\cdots (25)$$

동안은 까지와 유사한 활동으로 조건이 충족될 때는 동안 활동을 반복하여 실행하는 것이다. 단, 조건이 충족되지 않으면 실행을 한 번도 하지 않을 수 있다.

지금까지 BPML에 포함된 17가지 활동 유형에 대한 의미론을 정의하였다. 마지막으로 BPML에서 사용되는 문맥에 대하여 정의한다. 문맥은 표현된 BPML 문에서 사용되는 프로세스들 또는 신호 등에 대한 상세설명으로 일종의 선언문 성격을 지닌다. 보통 BPML 문의 앞부분에 정의된다. 이는 $\pi-Calculus$의 문맥을 그대로 사용한 것으로, 다음과 같이 정의할 수 있다.

Definition 18. BPML의 문맥은 다음과 같이 표현한다.

$$C::=[\cdot]\|\pi.C+M|\nu aC|C|P|P|C|!C \cdots\cdots\cdots (26)$$

즉, 홀(hole, $[\cdot]$)을 사용하여 문맥 내에 채워질 프로세스들을 사전에 정의하는 것이다.

이렇게 BPML을 구성하는 주요 활동 유형과 프로세스 및 구문에 대한 의미론을 정의함으로써 BPML문을 $\pi-Calculus$로 번역하여 전체 의미론을 파악할 수 있는 기반을 마련하였다. 다음에는 이러한 의미론 정의를 실제 예에 적용하여 그 유효성을 보이도록 한다.

6. BPML 의미론의 적용 예

본 장에서 사용하는 예는 BPML 사양[3, pp.51-56]에 포함된 것으로 전형적인 주문처리 프로세스를 표현하고 있다. 또한 주문처리 프로세스에서 발생할 수 있는 예외처리, 오류 취급 및 보정에 관한 사항을 모두 포함하고 있는 프로세스로 BPML 활동 유형의 대부분을 포함하고 있다.

프로세스는 주문 요청(order request)을 접수하면서 시작된다. 주문 요청에는 주문에 관한 상세 사항과 기대 종료 시간 등이 포함되어 있다. 프로세스는 고객에 대한 요금부과(chargeCustomer)와 배송(shipProduct) 프로세스들에 실행을 의뢰하는 단계로 나뉘어 진행된다. 프로세스는, 주문 취소 메시지를 전송하는 경우 언제든지 중단될 수 있고, 중단의 경우 예외 처리 프로세스인 주문 취소(cancelRequest) 프로세스를 실행하게 된다. 주문처리 프로세스가 기대시간 이내에 종료되지 않을 경우, 강제로 중단된다. 오류가 발생하는 경우, 프로세스는 정상적으로 종료될 수 없으며, 중단되기 전에 적절한 알림 메시지를 전송한다. 취소 요청이나 시간 초과에 따른 중단의 경우, 사용자가 이러한 조건들을 알고 있으므로 알리지 않아도 무방하다.. 프로세스는 과실이나 예외가 발생했을 때, 'chargeCustomer'까지 완료되었거나 또는 'chargeCustomer'와 'shipProduct' 모두가 완료되었을 수 있다. 앞의 경우 환불(refund)을 실행하는 보정 프로세스를 실행토록 의뢰하고 후자의 경우 환불(refund) 및 배송취소(cancelShipment) 모두를 실행하는 보정 프로세스를 실행토록 의뢰한다. 프로세스가 모두 완결되었을 때에도, 취소 요청을 송신하여 프로세스를 보정할 수 있다. 그러나 이에 대한 보정은 다른 문맥에서 실행된다. 이때에는 배송이 완료된 상태이므로 환불하기 전에 배송된 물품이 환수되기를 기다린다.

설명된 프로세스의 BPML 표현을 위의 정의들을 이용하여 $\pi-Calculus$로 번역하면 다음과 같다.

BPML에서 프로세스에 관한 정의에는 프로세스 명칭과 프로세스에 포함된 활동 집합 이외에 프로세스 구분자(identity) 등이 포함되어 있고, 프로세스에서 사용하는 속성에 대한 정의가 포함될 수 있다. 속성은 속성 명칭, 유형, 초기 값, 값의 고정 유무(fixed) 등이 포함된다. 이를 $\pi-Calculus$에서는 모두 명칭으로 표현할 수 있다. 본 논문에서는, 프로세스 구분자 및 속성은 모두 명칭으로 직접 표현하고, 속성의 유형 및 값의 고정 유무는 속성 명칭에 하이픈(-)을 이용하여 연결하여 표기한다. 속성의 인자(element) 또한 하나의 명칭으로 표현한다.

$\pi-Calculus$에서는 프로세스 내에서 자유롭게(free) 사용되는 이름의 집합을 **fn**(P)로 표현한다. 따라서 예제 프로세스에 사용되는 명칭의 집합은 다음과 같이 표현할 수 있다.

$$\begin{aligned}\mathbf{fn}(P_{twoStepOrder}) = \{&y1,y2,y3,y4,y5,y6,y7,y8,y9,\\ &orderDetails, details, timeLimit-typeDuration,\\ &timeToComplete, orderID-typeOrderID,\\ &orderID-typeOrderID-fixed, status, started,\\ &timeout, complete, returned, canceled,\\ &fault, aborted\}\end{aligned} \cdots\cdots (27)$$

프로세스는 다음과 같이 표현된다.

$$\begin{aligned}P_{twoStepOrder} ::= \pi_{action-receiveOrder}.(C|(&\pi_{assign-statusStarted}.\\ &\pi_{call-chargeCustomer}.\pi_{call-shipProduct}.\pi_{assign-statusComplete}.\\ &\pi_{action-notifyComplete}.P_{compensation-cancelRequest}))\end{aligned} \cdots\cdots (28)$$

프로세스는 주문을 접수하는 행동으로부터 시작된다.

$$\begin{aligned}\pi_{action-receiveOrder} ::= &((y1(orderDetails)|\overline{y1}details)|\\ &(y2(timeLimit-typeDuration)|\\ &\overline{y2}timeToComplete)|\tau_{order-portTypeOrderService}|\\ &(\nu x\bar{x}orderID-typeOrderID|\\ &x(newIdentifierOrderID))|\\ &\overline{y3}orderID-typeOrderID)\end{aligned} \cdots\cdots (29)$$

여기서, 문맥 C는 다음과 같다.

$$\begin{aligned}C::= &\pi_{schedule-timeToComplete}|\\ &(P_{exception-cancelRequest}|\pi_{fault}.P_{faults})\end{aligned} \cdots\cdots (30)$$

여기서, 일정(schedule)활동은 시간을 변화시키면서 조건에 합당한 시간에 도착하였을 때 이에 해당하는 메시지를 송신하는 활동으로 다음과 같이 표현 가능하다.

$$\begin{aligned}\pi_{schedule-timeToComplete} = &(\nu time,\times econd(\bar{x}time|\\ &x(time+\times econd)).([time=timeLimit]\\ &\overline{code}timeout+\pi_{schedule-timeToComplete}))\end{aligned} \cdots\cdots (31)$$

예외처리 프로세스로서의 주문취소 프로세스는 다음과 같이 표현된다.

$$P_{exception-cancelRequest} :: = (\pi_{action-receiveCancelRequest} \cdot \pi_{assign-statusCanceled} \cdot \pi_{compensate-AB}) \cdots\cdots (32)$$

주문취소 예외처리 프로세스를 구성하는 활동들은 다음과 같다.

$$\pi_{action-receiveCancelRequest} :: = (y7(orderID-typeOrderID) | [orderID-typeOrderID = orderID-typeOrderID-fixed] \tau_{cancelRequest-portTypeOrderService}) \cdots\cdots (33)$$

$$\pi_{assign-statusCanceled} :: = \nu x(x(status) | \overline{x} canceled) \cdots (34)$$

$$\pi_{compensate-AB} :: = \nu x \overline{x} compensate P_{compensation-refund-cancelShipment} \cdots (35)$$

오류처리 프로세스는 다음과 같이 표현된다.

$$\pi_{fault} :: = \nu x x(code).[code = fault]\overline{code} fault \cdots\cdots (36)$$

$$P_{fault} :: = (code(timeout).(\nu x x(status) | \overline{x} timeout). \pi_{compensate-AB} + code(fault).(\nu x x(status) | \overline{x} aborted). \pi_{action-notifyError} \cdot \pi_{compensate-AB}) \cdots\cdots (37)$$

다음은 프로세스 $P_{twoStepOrder}$의 나머지 구성활동 및 프로세스에 대한 표현이다.

$$\pi_{assign-statusStarted} :: = \nu x x(status) | \overline{x} started \cdots\cdots (38)$$

$$\pi_{call-chargeCustomer} :: = \overline{y1} details. P_{chargeCustomer} \cdot y4(details) \cdots\cdots (39)$$

$$\pi_{call-shipProduct} ::= \overline{y4} details. P_{shipProduct} \cdot y5(details) \cdots (40)$$

$$\pi_{assign-statusComplete} ::= \nu x x(status) | \overline{x} complete \cdots\cdots (41)$$

$$\pi_{action-notifyComplete} ::= (\tau_{MyCompletion-portTypeOrderService} \cdot \overline{y6} orderID-typeOrderID) \cdots\cdots (42)$$

$$P_{compensation-cancelRequest} :: = (\pi_{action-receiveCancelRequest} \cdot \pi_{call-receiveReturn} \cdot \pi_{compensate-chargeCustomer} \cdot \pi_{assign-statusReturned}) \cdots\cdots (43)$$

$$\pi_{call-receiveReturn} ::= \overline{y7} details. P_{receiveReturn} \cdot y8(details) \cdots (44)$$

$$\pi_{compensate-chargeCustomer} ::= \nu x \overline{x} compensate P_{chargeCustomer} \cdots\cdots (45)$$

$$\pi_{assign-statusReturned} ::= \nu x x(status) | \overline{x} retirned \cdots\cdots (46)$$

7. BPML 의미론의 활용

비즈니스 프로세스를 BPML을 사용하여 모델링하고, 이에 대한 형식화를 시도하는 것은 해당 비즈니스 프로세스의 특성을 확인하기 위한 것이다. 특히, $\pi-Calculus$를 사용하여 형식화할 경우 다음과 같은 장점을 가진다[5].

- $\pi-Calculus$의 "Weak Bisimulation"을 이용하여 두 가지 프로세스가 서로 동일한 것인지 확인할 수 있다. 이를 통하여 비즈니스 프로세스의 최적화가 가능하다.
- $\pi-Calculus$의 분석도구(예 : MWB(Mobility Workbench), [13])를 활용하여 프로세스 모델이 특정 조건(예 : 안전성, 생존성 등)을 만족하는지 확인할 수 있다.
- 프로세스의 정확한 종료 여부 등의 조건을 검증할 수 있다. 즉, 비즈니스 프로세스가 정확한지 여부는 다음의 두 가지 조건을 만족하는지 여부를 확인하는 것으로 가능하다[2].
- 시작 상태에서 시작하면, 항상 최종 상태에 도달 가능하다.
- "Deadlock"이 존재해서는 안 된다.
- "Bottom-up" 접근방법을 사용하여 하위 프로세스들로부터 상위(전체) 프로세스를 구축하는 것이 가능하다.

8. 결 론

본 논문에서는 비즈니스 프로세스의 정확한 모델링과 분석을 위하여 비즈니스 프로세스 실행 모형을 위한 언어 중 하나인 BPML의 형식화를 시도하고 이의 활용방안에 대하여 정리하였다. BPML의 형식화를 위하여, BPML의 활동 유형들에 대한 의미론을 정의하였으며, 예외처리가 포함된 주문처리 프로세스를 대상으로 정의된 의미론을 적용하였다. 그 결과 우리는 BPML로 표현된 프로세스 모형의 형식화가 가능함을 확인할 수 있었다.

그러나 우리는 이러한 결과가 BPML에 대한 의미론을 완성한 것이라고는 생각하지 않는다. 다만, 비즈니스 프로세스 관리를 위하여 필요한 비즈니스 프로세스의 모형화 및 분석에 유용한 하나의 도구로서 $\pi-Calculus$를 활용한 의미론의 부여가 가능함을 보인 것이다. BPML 또한 다른 프로세스 실행 모형 언어들과 비교하여 아직 완성된 것으로 보기 어렵고, BPEL 등 다른 언어들의 활용도가 높아질 가능성도 있다. 따라서 향후 BPEL 등 다른 언어들의 형식화에 대한 시도와 다양한 실제 프로세스들의 비교 및 분석에 발전된 의미론을 적용하는 방안에 대한 연구가 필요할 것이다.

참고문헌

[1] van der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M., and Wohed, P.; "Pattern based analysis of BPML (and WSCI)," Technical report No. FIT-TR-2002-050, Brisbane, Australia: Queensland University of Technology, 2002.

[2] van der Aalst, W. M. P.; "The application of Petri-nets for Workflow management," *The journal of Circuits, Systems, and Computers*, 8(1) : 21-66, 1998.

[3] Arkin, A.; *Business Process Modeling Language*, BPMI.org, 2002.

[4] Browning, T. R.; "Process Integration Using the Design Structure Matrix," *Systems Engineering*, 5(3) : 180-193, 2002.

[5] Dong, Y. and ShenSheng, Z.; "Using $\pi-Calculus$ to Formalize UML Activity Diagrams," Proceedings of 10th International Conference and Workshop on Engineering of Computer-Based Systems, IEEE Press, NY, pp. 47-54, 2003.

[6] Dufresne, T. and James M.; "Process Modeling for E-Business," Term Paper for INFS 770 - Methods for Information Systems Engineering (Larry Kirschberg), George Mason University, pp. 1-28, 2003.

[7] Lam, V. S. W. and Padget, J. A.; "Formalization of UML Statechart Diagrams in the p-Calculus," Proceedings of Australian Software Engineering Conference, pp. 213-223, 2001.

[8] Milner, R.; Communicating and Mobile Systems : the $\pi-Calculus$, Cambridge University Press, Cambridge, UK, 1999.

[9] Moon, J., Lee, D., Park, C., and Cho, H.; "Transformation Algorithms between BPEL4WS and BPML for the Executable Business Process, Proceedings of the 13th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE'04), pp. 135-140, 2004.

[10] Peltz, C.; Web Service Orchestration. A Review of emerging technologies, tools, and standards, Hewlett Packard, CO., 2003.

[11] Puhlmann, F. and Weske M.; "Using the π-Calculus for Formalizing Workflow Patterns," Proceedings of Business Process Management : 3rd International Conference, BPM 2005, pp. 153-168, 2005.

[12] Sangiorgi, D., and Walker, D.; *The $\pi-Calculus$: A Theory of Mobile Processes*, Cambridge University Press, Cambridge, UK.

[13] Victor, B., and Moller, F.; "The Mobility Workbench - a tool for the $\pi-Calculus$," Technical Report DoCS 94/45, Department of Computer Systems, Uppsala University, 1994.

[14] Wohed, P., van der Aalst, W. M. P., Dumas, M., and ter Hofstede, A.; "Analysis of Web service composition languages : The case of BPEL4WS," Proceedings of 22nd International Conference on Conceptual Modelling (ER), Chicago, pp. 13-16, 2003.