DOI: 10.3745/KIPSTD.2009.16D.5.791

가변성 결정기반 BPM 생성을 위한 가변성 의존관계 분석

문 미 경†

요 약

최근 서비스 지향 아키텍처 (Service Oriented Architecture SOA) 기반의 애플리케이션 개발에 맞게 비즈니스 프로세스의 유연성을 확보하고 재사용을 증진시키기 위하여 비즈니스 프로세스 패밀리 모델 (Business Process Family Model: BPFM)이 제시되었다. BPFM은 소프트웨어 프로덕트 라인 방법의 가변성 분석 기법을 사용하여 비즈니스 프로세스 군 (family)에서 나타날 수 있는 가변성을 분석하여 이를 명시적으로 표현하고 있는 모델이다. BPFM으로부터 여러 개의 비즈니스 프로세스 모델 (Business Process Model: BPM)을 개발하기 위해서는 가변성 결정 및 가지치기(Decision and Pruning) 과정을 거쳐야 한다. 이 때 가변성 사이에는 서로 협력적 또는 배타적인 관계를 가질 수 있고 이는 가변성 결정과 가지치기에 영향을 미치게 되는데, 현재 제시된 BPFM에는 이러한 바인딩 정보에 대해서 고려하지 않고 있다.

본 논문에서는 비즈니스 프로세스 군에서 식별될 수 있는 가변성들 사이의 의존관계 유형을 분석하고 이러한 가변성 정보를 독립된 의존관계 분석모델로 표현하는 방법을 제시한다. 또한 추출된 모델을 기반으로 하나의 가변성 결정으로부터 영향을 받는 다른 가변성들을 추적하여 선결정 처리 할 수 있는 방법을 제공한다. 본 방법을 이용함으로써 가변성 결정회수를 줄일 수 있고, 또한 잘못된 가변성 결정으로 인한 BPM의 기능 불일치를 해소할 수 있음을 사례연구를 통해 보인다.

키워드 : 가변성 의존관계, 가변성 결정, 가변성 분석, 비즈니스 프로세스 모델, 비즈니스 프로세스 패밀리 모델

Variability Dependency Analysis for Generating Business Process Models based on Variability Decisions

Mikyeong Moon†

ABSTRACT

Recently, the business process family model (BPFM), which is new approachfor assuring businessflexibility and enhancing reuse in application development with service oriented architecture (SOA), was proposed. The BPFM is a model which can explicitly represent the variabilities in business process family by using the variability analysis method of software product line. Many business process models (BPM) can be generated automatically through decision and pruning processes from BPFM. At this time, the variabilities tend to have inclusive or exclusive dependencies between them. This affects the decision and pruning processes. So far, little attention has been given to the binding information of variability dependency in the BPFM.

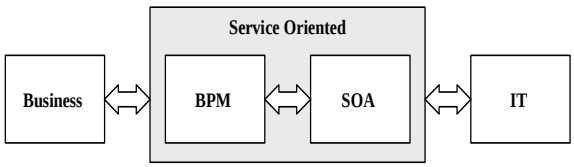
In this paper, we propose an approach for analyzing various types of dependency relationships between variabilities and representing the variability and their relationships as a dependency analysis model. Additionally, a method which can trace the variabilities affected by a decision on the dependency analysis model is presented. The case study shows that the proposed approach helps to reduce the number of variability decision and to solve a disagreement of functions in BPM produced by incorrectly deciding the variability.

Keywords : Variability Dependency, Variability Decision, Variability Analysis, Business Process Model, Business Process Family Model

1. 소 개

서비스 지향 아키텍처 (Service Oriented Architecture: SOA)는 재활용도가 높은 서비스를 기반으로 애플리케이션을 개발하고자 하는 아키텍처 모델이다. (그림 1)에서 보는 바와 같이 비즈니스 작업자는 비즈니스 프로세스 모델 (Business Process Model: BPM)을 통해서 자신이 이해할 수 있는 프로세스란 용어로 비즈니스 서비스를 사용하고, IT 개발자는SOA를 통해서 IT가 이해하기 쉬운 아키텍처란 용어로 소프트웨어 서비스를 표현한다 [1]. IT 개발자는 SOA 개념에 따라 비즈니스의 기능을 서비스로 구현함으로

※ 본 논문은 2008년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2008-531-D00030).
† 정 회 원 : 동서대학교 컴퓨터정보공학부 전임강사
논문접수 : 2009년 7월 16일
수 정 일 : 1차 2009년 8월 18일
심사완료 : 2009년 8월 22일

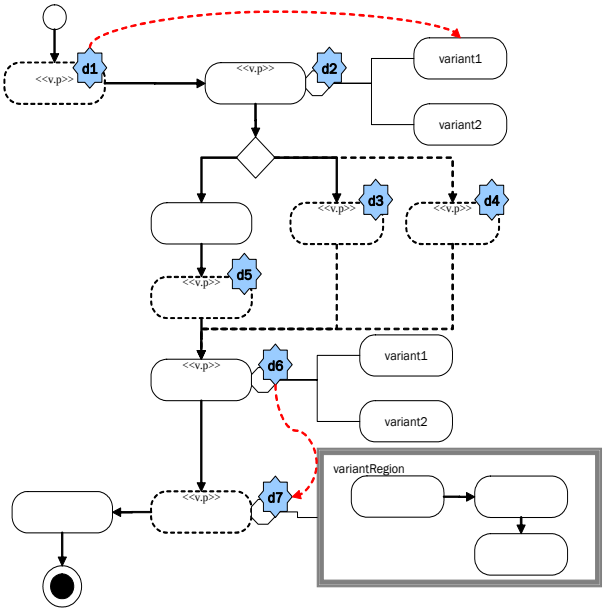


(그림 1) 비즈니스와 IT의 간격을 줄이기 위한 BPM과 SOA

써 비즈니스의 변화에 민첩하게 IT 시스템을 구축하고 변경할 수 있게 된다.

이러한 SOA 기반의 애플리케이션 개발에 맞게 비즈니스 프로세스의 유연성을 확보하고 재사용을 증진시키기 위하여 소프트웨어 프로덕트 라인 방법의 가변성 분석 기법을 사용하여 비즈니스 프로세스 군 (family)에서 나타날 수 있는 가변성을 분석하고 이를 명시적으로 표현할 수 있는 비즈니스 프로세스 패밀리 모델 (Business Process Family Model: BPFM)이 있다 [2]. 하나의 BPFM으로부터 가변성 결정 및 가지치기 (Decision and Pruning) 과정을 거쳐 하나 이상의 BPM을 개발할 수 있다. BPFM은 비즈니스 프로세스에서 발생할 수 있는 다양한 가변성을 표현할 수 있다. 그러나 현재 BPFM을 재사용 할 때 사용할 수 있는 바인딩 정보 중 가변성 의존관계 정보에 대해서는 아직 고려하지 않고 있다. 가변성 사이의 의존관계는 가변성 유형과 의존관계의 유형(필수적 또는 배타적)에 따라 재사용 시, 결정과 가지치기에 바로 영향을 준다.

(그림 2)는 현재 BPFM에서 가변성 결정이 이루어지는 지점들(d_n)을 표현하고 있다. <표 1>은 BPFM을 UML2 액티비티 다이어그램을 이용하여 표현하기 위해서 추가적으로 새로 정의한 모델요소들이다. BPFM을 이용하여 가변성을 결정하는 방식은 가변성 추상화의 두 수준에서 나뉘어 결정이 이루어진다. 먼저 상위수준에서 선택적 활동에 대해 선택/제거의 결정이 내려진다. 그리고 하위수준의 가변점에서 가변치들의 선택, 선택된 가변치들 사이의 흐름 선택 등이 이루어진다. 이러한 방식은 수준의 분리로 인해 결정해야 하는 단계를 나눌 수 있다는 장점이 있고, 모델 그 자체에서 바로 결정을 할 수 있다는 장점이 있다. 그러나 아직까지 가변성 요소들 사이의 의존관계에 대한 정보가 없기 때문에 먼저 이루어진 가변성 결정이 다른 가변성 요소에 어떠한 영향을 미칠 것인지를 애플리케이션 개발자는 모르고



(그림 2) BPFM에서 일어나는 가변성 결정

결정을 내리게 된다. 그 결과 가변성 결정에 대한 일관된 판단을 행하였는지에 대한 검증이 또 다시 필요하게 된다. 예를 들어, (그림 2)에서 d1의 가변성 결정이 d2의 가변점에 바인딩 되어 있는 가변치1을 요구하는 의존관계를 가진다면, d1에서 가변성 결정만으로 d2에서 일어나는 가변성 결정은 생략될 수 있다. 이와 같이 동일한 결정을 요구하는 관련된 가변성 요소들을 추적 할 수 있다면, 개발자에게 요구하는 가변성 결정의 횟수는 급격히 줄어들 수 있을 것이다.

본 논문에서는 BPFM에 포함된 가변성 정보를 독립된 의존관계 분석모델 구성요소로 추출해 내고 각 가변성 결정유형에 따라 가변성 의존관계를 표현하는 방법을 제시한다. 추출된 모델에서 가변성 결정이 영향을 미치는 범위의 가변성들을 추적할 수 있는 방법을 제공한다.

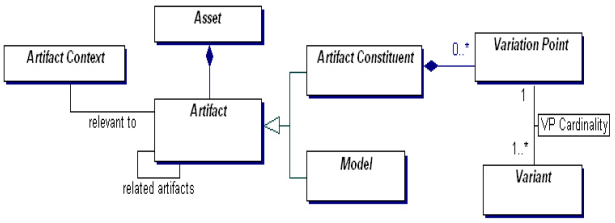
2. 기반 연구

2.1 소프트웨어 프로덕트 라인

소프트웨어 프로덕트 라인은 일련의 관련된 시스템들, 즉

〈표 1〉 가변성 표현을 위한 UML2 액티비티 다이어그램 확장 모델요소

모델 요소	의미	표시법
선택적 도메인 액티비티	도메인 액티비티 하나가 선택적인 경우를 나타낸다. 이것은 비즈니스 프로세스 흐름 중, 도메인 액티비티의 추가, 삭제를 발생시킨다.	Activity1
가변점	가변점 모델요소는 도메인 액티비티가 대체, 확장 될 수 있음을 나타내기 위해 도메인 액티비티를 나타내는 둥근 사각형에 접착하여 표현한다. 비즈니스 프로세스 흐름 중, 하나 이상의 도메인 액티비티의 대체를 발생시킨다.	vp
가변치 대응관계	가변점과 하나 이상의 가변치 (또는 가변치 영역)를 연결시키기 위한 관계 모델요소이다.	
가변치 영역 (variants region)	가변치 영역은 하나 이상의 가변치들의 묶음을 나타내기 위한 모델요소이다.	



(그림 3) 소프트웨어 프로덕트 라인 자산 메타모델 [7]

도메인 내에서 다시 재사용될 가능성이 높은 공통된 부분들을 식별하고, 시스템마다 상이하게 나타나는 가변적 요소들을 분석하는 것이다 [3, 4]. 도메인의 주요 산출물들 –요구사항 [5], 아키텍처 [6] 등– 은 분석된 공통성과 가변성을 명시적으로 나타냄으로써 프로덕트 라인의 자산(*asset*)이 된다.

(그림 3)은 소프트웨어 프로덕트 라인 자산에 대한 메타모델이다 [7]. 자산은 하나 이상의 산출물(*artifact*)들로 구성되며, 다른 산출물과 관계를 가질 수도 있다. 하나의 산출물은 요구사항 개발, 설계 또는 런타임 문맥과 같은 특정 문맥(*artifact context*)과 관련된다. 산출물은 산출물 구성요소(*artifact constituent*)와 모델(*model*)의 형태로 특수화(specialization) 될 수 있다. 산출물 구성요소는 재사용 될 때, 수정될 수 있는 지점을 나타내는 가변점(*variability point*)을 가지기도 하며, 모델은 각 형태에 따라 산출물 구성요소가 내포하고 있는 가변점을 표현할 수 있어야 한다. 하나의 가변점에는 가변성 실현 시 바인딩 될 수 있는 구체적인 값인 가변치(*variant*)가 하나 이상 연결될 수 있다. 또한 실현 시 가변점에 바인딩 될 수 있는 가변치 개수를 가변점 대응값(*vp cardinality*)으로 표현한다. 본 논문에서는 비즈니스 프로세스 분석 문맥에서 식별할 수 있는 가변성을 분석한 후, 이를 도메인 액티비티 산출물 구성요소와 비즈니스 프로세스 패밀리 모델 형태로 산출된 자산에 초점을 둔다.

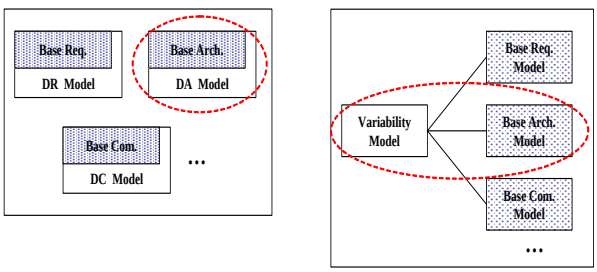
2.2 비즈니스 프로세스 모델

Process-oriented 관점으로 비즈니스 프로세스를 표현하는 모델링 방법론에는 Petri Nets, UML 액티비티 다이어그램, BPMN (business process modeling notation) 등이 있다 [8, 9]. BPMN은 모든 사용자가 사용하고 이해하기 쉬운 표기법 개발을 목표로, BPEL4WS (BPEL4 web service)와 BPML (Business Process Modeling Language)과 같이 비즈니스 프로세스의 실행을 위해 설계된 XML언어에 시각적으로 표현 가능한 공통된 표기법을 제공하려는 목적으로 개발되었다 [10]. UML 액티비티 다이어그램은 소프트웨어에서 즉시 개발되는 소프트웨어와 기술적 프로세스에 더 맞춰져 있고, Petri Nets은 좀더 넓은 범위의 목적에 사용되는 경향이 있다 [11]. 현재 이들 프로세스 모델링 언어들은 모두 비즈니스 프로세스의 설계와 프로세스의 실행 사이에 있는 차이의 표준화된 연결을 위해 노력하고 있다 [12]. 본 논문에서 제시하는 BPFM은 이 중, 개발자에게 친숙하고 크고 복잡한 SOA 애플리케이션을 좀 더 관리하기 쉬운 컴포넌트로 분해하기에 용이한 UML 액티비티 다이어그램의 스펙을 분석, 이를 확장하여 가변성을 가진 비즈니스 프로세스를 표현하고 있다.

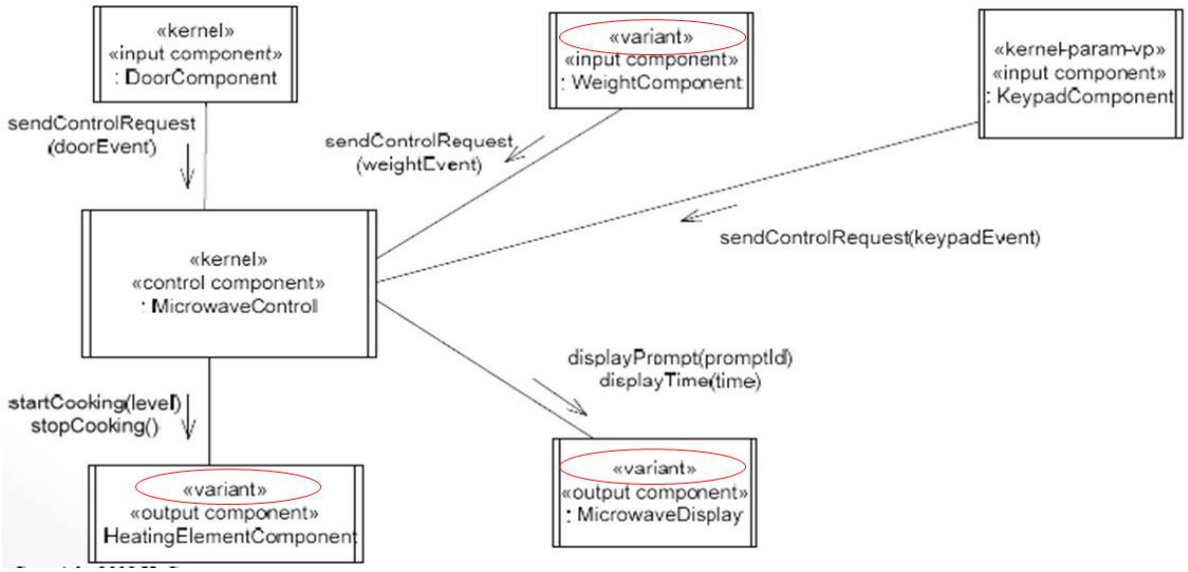
2.3 가변성 의존관계 표현방법

소프트웨어 프로덕트 라인의 기존 연구들에서 가변성을 표현하는 방법은 (그림 4)와 같이 두 가지 유형으로 분류할 수 있다. 여기서 기본형태 모델(Base~model)이란 단일 애플리케이션 개발 시 산출되는 모델과 같이 가변성이 표현되지 않는 형태를 의미한다.

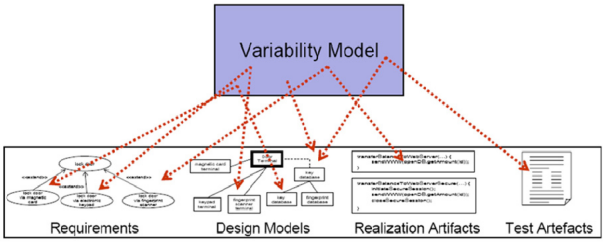
- **기본모델과 가변성을 함께 모델에 표현** – (그림 4)의 왼쪽에서 표현하고 있는 형태와 같이 기본모델에 가변성을 표현하기 위한 요소를 추가하여 특정 자산을 개발하는 것이다. 이러한 방법은 [13-16] 등의 연구에서 적용이 되었다. 이 방법은 특정 자산, 예를 들어 도메인 아키텍처 모델(DA Model)만을 고려할 때, 가변성을 나타내기 위한 확장 기법만 쉽게 이해할 수 있다면, 기존의 기본모델에서 나타내고자 하는 의미를 그대로 이용할 수 있어서 개발자에게 도메인 자산을 쉽게 이해시킬 수 있는 장점이 있다. 그러나 특정 하나의 자산뿐만 아니라 다른 자산들, 예를 들어, 도메인 요구사항 모델(DR Model), 도메인 컴포넌트 모델(DC Model))들을 함께 고려한다면, 이들 사이의 가변성의 관계를 일관성 있게 다루지 못하게 되는 단점이 있다. (그림 5)는 [16]방법에서 사용하고 있는 가변성 표현방법의 예이다. 이 접근방법에서 가변성 의존관계를 표현하는 것은 모델의 복잡성을 증가시키게 된다. 현재 BPFM은 기본모델에 가변성이 같이 표현되어 나타나기 때문에 이 접근방법을 따르고 있다. BPFM에서 사용되는 관계 모델링 요소에는 제어흐름 (control flow), 객체흐름 (object flow)과 가변점과 가변치를 연결하는 가변점 바인딩 (VPbinding)이 있다. 여기에 의존관계 모델 요소까지 추가하게 된다면, 모델의 readability는 아주 떨어지게 될 것이다.
- **기본 모델과 가변성 모델을 분리하여 표현** – 이러한 접근 방법의 대표적인 연구는 [17, 18] 등이 있다. 이 방법에서는 기본모델과 가변성 모델을 분리하여 표현한다. 그리고 하나의 가변성 모델을 기반으로 여러 자산의 기본모델에 연결 관계를 제공하고 있다. 이러한 방법은 가변성 모델을 하나의 독립된 모델로 분리함으로써 가변



(그림 4) 가변성을 표현하기 위한 기존의 두 가지 접근 방법



(그림 5) [16]방법에서 사용하고 있는가변성 표현방법



(그림 6) [18]방법에서 사용하고 있는가변성 표현방법

성만을 일관성 있게 다룰 수 있으며, 이로 인해, 가변성의 의존관계도 잘 관리할 수 있는 장점이 있다. 그러나 가변성은 개발이 진행됨에 따라 가변성이 정제되어 나타나게 되고 각 자산마다 가변성이 표현되어야 하는 자산의 요소가 달라지는데, 이를 표현할 방법이 부족하다는 단점이 있다. (그림 6)은 [18]방법에서 사용하고 있는 가변성 표현방법의 예이다. 이 접근방법에서 가변성 의존관계를 표현하는 것은 일단 가변성 요소들만 분리되어 있는 가변성 모델에서 의존관계를 나타낼 수 있기 때문에 앞 접근방법보다는 표현하고 관리하기가 용이하다.

본 연구에는 기본모델에 가변성을 표현하는 방법을 유지하면서 모델의 복잡성을 증가시키지 않기 위해 의존관계 정보를 독립적으로 분리하여 표현하는 방법으로 접근한다.

3. 가변성 의존관계 분석 모델

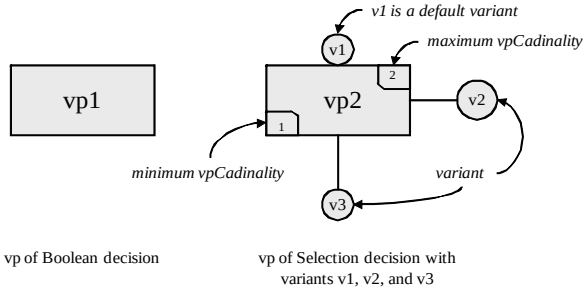
<표 2>는 BPFM에서 표현하는 가변성 유형을 정리한 것이다. 결정유형은 이러한 가변성 유형에 따라 Boolean decision과 selection decision, 그리고 flow decision으로 구분할 수 있다. Boolean decision은 단지 액티비티 또는 액티비티의 그룹이 true/false의 선택으로 비즈니스 프로세스에 삽입 또는 삭제될 수 있는 결정을 의미한다. selection decision은 후보 액티비티 중, 정의된 가변치 대응값(vp cardinality)만큼 선택되어 비즈니스 프로세스에 대체될 수 있는 결정을 의미한다. flow decision은 액티비티들 사이의 흐름을 결정하는 것으로 이는 액티비티들의 삽입, 삭제, 대체에는 영향을 주지 않는다. 본 논문에서는 가변성 결정이 도메인 액티비티의 선택 또는 제거에 영향을 주는 Boolean decision과 selection decision에 대하여 그 의존성을 분석하게 된다.

〈표 2〉 BPFM의 가변성 유형 및 결정유형

가변성유형	가변성유형 설명	결정유형	결정유형 설명
공통성/선택성 속성가변성 (CV_property variability)	도메인 액티비티 그 자체가 선택적인 경우	Boolean decision	도메인 액티비티의 선택 또는 제거
대체가능후보 선택가변성 (Alternative variants-selection variability)	후보액티비티 n개가 선택되어 대체될 수 있는 경우	selection decision	0~n개의 도메인 액티비티 선택
확장흐름 선택가변성 (Extension flow variability)	특정 조건하에서만 확장흐름이 수행되는 경우	Boolean decision	확장흐름 선택 또는 제거
흐름결정 가변성 (Flow decision variability)	두 가지 이상의 흐름유형 조합이 가변적인 경우	flow decision	도메인 액티비티들 사이의 흐름 결정

3.1 가변성 의존관계 분석모델 구성요소 정의

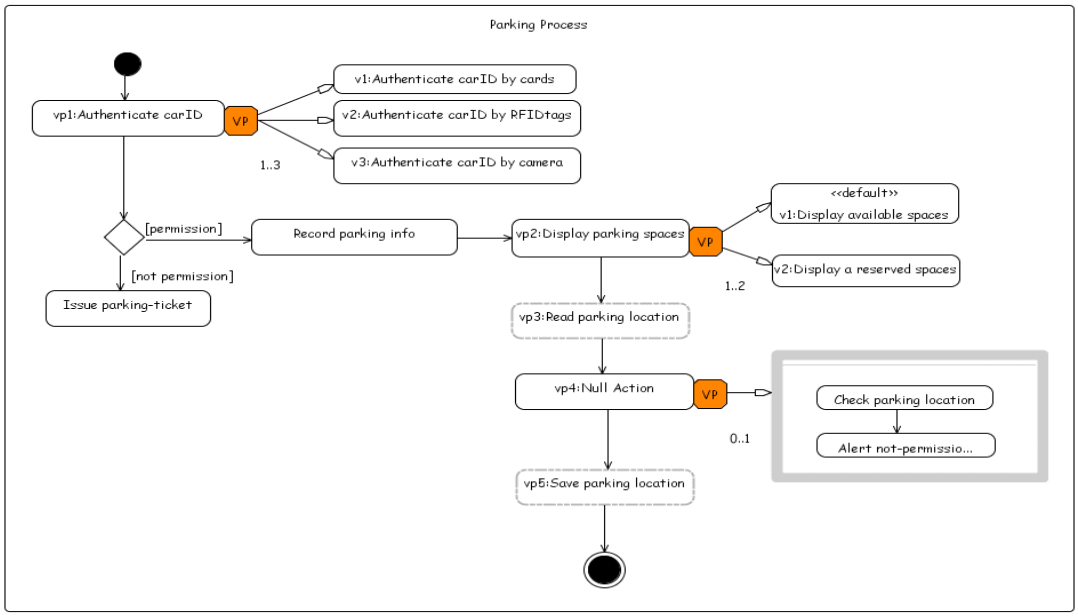
BPFM으로부터 액티비티의 선택이 이루어져야 하는 가변성들을 추출하여 독립된 가변성 의존관계 분석모델을 그린다. 이 모델이 가지는 모델요소는 가변성 결정유형에 따라 (그림 7)과 같이 두 가지 형태로 표현된다. (그림 7)의 (1)은 Boolean decision이 이루어지는 가변점을 나타내며, (그림 7)의 (2)는 selection decision이 이루어지는 가변점을 나타낸다. (2)의 경우는 가변점에 원으로 가변치들을 연결하여 표시한다. 이때 항상 선택이 이루어지는 default 가변치인 경우 선 없이 가변치가 가변점에 붙어있는 모양을 가진다. 또한 가변치 대응값을 가변점 모델링 요소 위에 그림과 같이 표현한다. 마지막으로 BPFM은 도메인 액티비티들 사이의 일련의 순서를 가진 흐름이기 때문에 가변성 결정도 이와 동일하게 흐름을 표현할 수 있어야 한다. 이를 위해 가변점 모델링 요소의 이름을 vpn (n: 순서를 나타내는 번호)으로 표시한다.



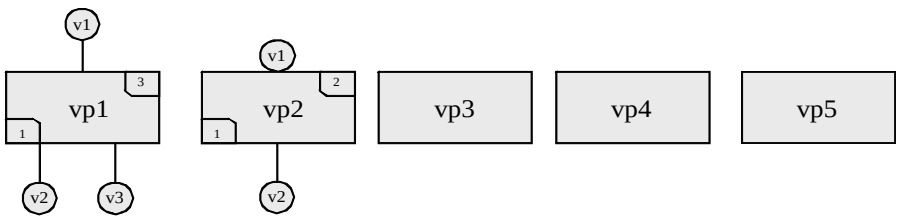
(그림 7) 가변성 의존관계 분석모델 구성요소

(그림 8)은 Free Parking System을 위한 도메인에서 입차 시 행해지는 가변성을 가진 액티비티들에 대한 BPFM이다. 이 BPFM에서는 가변성이 모두 다섯 곳에서 표현되어 있으며 그로 인해 다섯 번의 가변성 결정이 이루어지게 된다. 첫 번째 가변점을 가진 액티비티는 차ID를 인증하는 일반화된 액티비티인데, card에 의한 수동인증, RFID 태그를 이용한 자동인증, 카메라를 통한 반수동인증으로 구체적인 가변치 액티비티를 가진다. 두 번째 가변점을 가진 액티비티는 사용 가능한 주차공간을 표시하는 액티비티와 지정석을 표시하는 액티비티로 구체화 될 수 있다. 세 번째 가변점을 가진 액티비티부터 다섯 번째 액티비티는 모두 RFID 태그를 이용한 주차위치 인식관련 액티비티들이다.

(그림 8)로부터 추출된 가변성 의존관계 분석모델의 구성요소들은 (그림 9)와 같다. vp1과 vp2는 selection decision 모델요소로, vp3, vp4, vp5는 Boolean decision 모델요소로 표현된다.



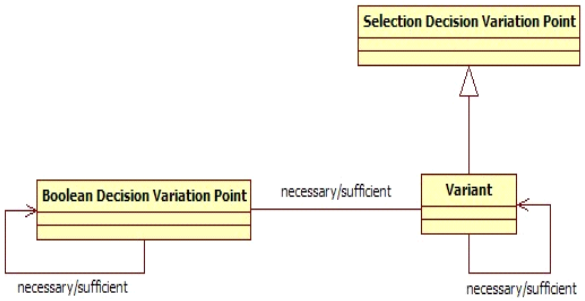
(그림 8) Free Parking System의 입차를 위한 BPFM



(그림 9) 사례연구(그림 8)에 대한 가변성 의존관계 분석모델 구성요소

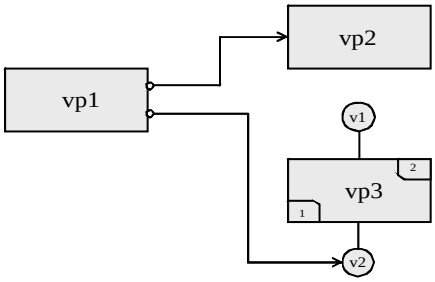
3.2 가변성 의존관계 분석

BPFM 상에 나타날 수 있는 가변성 의존관계와 그 유형을 (그림 10)과 같이 정의한다. 의존관계는 Boolean decision 가변점들 사이에, 그리고 Boolean decision 가변점과 가변치들 사이, 가변치와 또 다른 가변치들 사이에 존재한다. 의존관계 유형은 필요조건(necessary condition) 의존관계, 충분조건 (sufficient condition) 의존관계가 있다. 모든 의존관계에는 이 두 가지 유형이 나타날 수 있다.



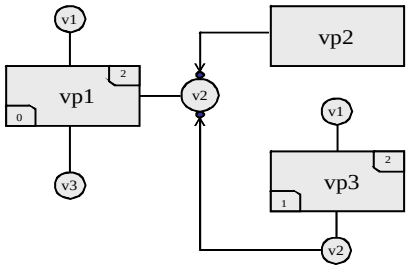
(그림 10) 가변성 의존관계 유형 및 존재관계

- **충분조건 의존관계 설정** – BPFM에서 현재 가변점 선택결정이 뒤의 가변점 선택결정에 영향을 주는 형태를 충분조건 의존관계로 설정하고 이를 앞에서 뒤 방향 화살표로 나타내며, 화살표의 시작점에 흰 동그라미를 붙인다. 이때 동그라미는 그 가변점 또는 가변치가 관계 맺고 있는 충분조건 관계수를 의미하는 토큰이 된다. 예를 들어, (그림 11)에서 vp1이 선택이 되면 vp2와 vp3의 가변치 v2가 따라서 선택되어야 함을 나타내기 위해 의존관계를 설정하게 된다. 이러한 충분조건 의존관계 설정은 가변점 결정 시, 가변점 vp1이 선택되면 v1에 '충분조건'으로 의존하고 있는 모든 가변점 또는 가변치들이 자동 미리 선택될 수 있도록 해준다.



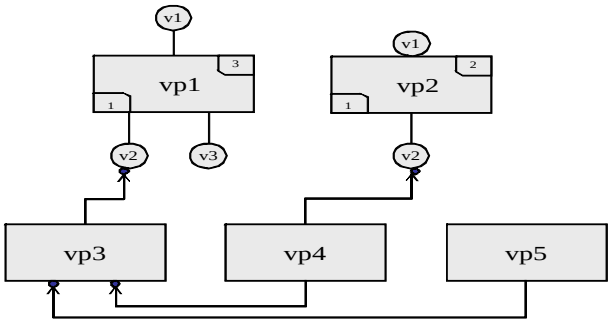
(그림 11) 충분조건 의존관계

- **필요조건 의존관계 설정** – BPFM에서 현재 가변점 선택결정이 필수적으로 앞의 가변점 선택결정을 필요로 하는 의존관계 일 때, 이를 뒤에서 앞 방향 화살표로 나타내고 화살표 끝에 검은색 동그라미를 붙인다. 이때 동그라미는 그 가변점 또는 가변치가 관계 맺고 있는 필요조건 관계수를 의미하는 토큰이 된다. 예를 들어, (그림 12)는 vp2와 vp3의 가변치 v2가 수행되기 위해서 반드시 vp1의 가변치 v2가 존재해야 함을 나타내고 있다. 이는 반대로 vp1의 v2가 선택되었다고 해서 반드시 vp2 또는 vp3의 v2가 선택될 필요는 없는 것이다. 이러한 필요조건 의존관계 설정은 가변점 결정 시, 가변점 vp1에서 만약 가변치 v2가 선택되지 못하면 v2에 '필요조건'으로 의존하고 있는 모든 가변점 또는 가변치들이 자동 가지치기 될 수 있도록 해준다.



(그림 12) 필요조건 의존관계

(그림 13)은 (그림 8) (Free Parking System의 입차를 위한 BPFM)에서 분석된 가변성 의존관계 분석모델이다. 가변성 결정을 해야 하는 세 번째 액티비티(vp3)부터 다섯 번째 액티비티(vp5)는 모두 RFID 태그를 이용한 주차위치 인식 관련 액티비티들이기 때문에 첫 번째 가변성(vp1)의 RFID 가변치 액티비티(v2)가 선택되었을 때만 가능하게 수행될 수 있도록 연속된 필요조건 의존관계(vp3이 vp1의 v2를, vp4는 vp3을, vp5는 vp3)를 가지게 된다. 특히, 주차 시 지정석임을 체크하여 이를 알려주는 가변점 액티비티 vp4는 RFID 관련 액티비티뿐만 아니라 지정석 안내표시 액티비티(vp2의 v2)에도 필요조건 의존관계를 보인다.



(그림 13) 사례연구(그림 8)에 대한 가변성 의존관계 분석모델

3.3 가변성 결정을 위한 의존관계 추적

BPFM에서 BPM으로 인스턴스하기 위해 가변성 결정을 내리게 된다. 이때는 분석된 가변성 의존관계를 바탕으로 서로 영향을 주는 의존관계를 추적하여 관련 가변성들에 대한 결정을 미리 수행할 수 있게 된다. BPFM은 일련의 순서를 가진 모델이기 때문에 이 순서에 따라 결정과정이 이루어진다. 만약 vp1에 해당하는 도메인 액티비티에 대한 선택

〈표 3〉 의존관계 유형에 따른가지치기 과정

의존관계 유형	가변성 결정	가지치기 과정
충분조건 의존관계	vpl 선택	vpl과 충분조건관계를 가지고 있는 모든 vp들을 추적하여 '선택' 결정한다.
	vpl 제거	vpl과 충분조건관계를 가지고 있는 어떤 vp에도 결정 영향을 주지 않는다.
필요조건 의존관계	vpl 선택	vpl과 필요조건관계를 가지고 있는 어떤 vp에도 결정 영향을 주지 않는다.
	vpl 제거	vpl과 필요조건관계를 가지고 있는 모든 vp들을 추적하여 '제거' 결정한다.

```
void traceVP(VariabilityAnalysisModel VAM)
{
   foreach vpi in VAM
      if not vpi.checked
         switch vpi.vpType
            case BooleanType:
                  decision vpi;
                  vpi.checked=true;
                  if vpi.decisionType == true then
                     selectionTraceCheck(vpi);
                  else
                     deletionTraceCheck(vpi);
               break;
               case SelectionType:
                  foreach vai in vpi.variants
                     if vpi.cardinality.upper>0 {
                        if not vai.checked {
                           decision vai;
                           vai.checked=true;
                           if vai.decisionType == true {
                              vpi.cardinality.upper--;
                              selectionTraceCheck(vai);
                           }
                           else
                              deletionTraceCheck(vai);
                        } //if
                     }//if
                  end foreach
                  vpi.checked=true;
         end switch
   end foreach
}
```

```
void selectionTraceCheck(V)
{
   foreach sti in sufficientTokens of V
      v=sti->relation;
      v.checked=true;
      v.decisionType=true;
      switch v.vpType
         case BooleanType:
            selectionTraceCheck(v);
            break;
         default: //in case variant
            v.vp.cardinality.upper--;
            selectionTraceCheck(v);
      end switch
   end foreach
}
```

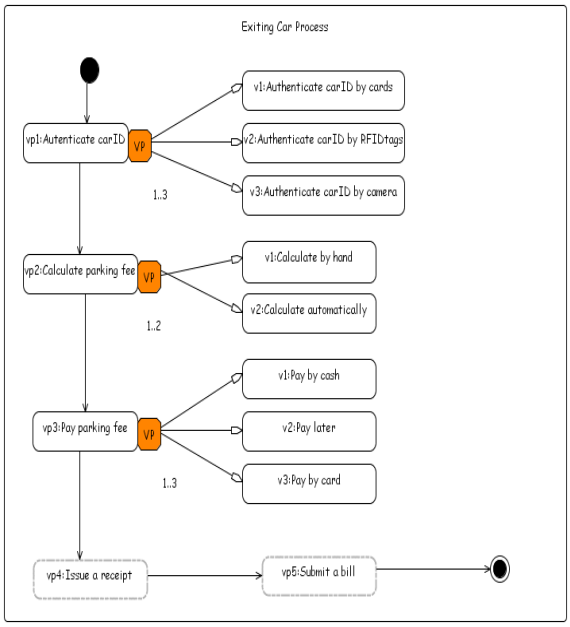
```
void deletionTraceCheck(V)
{
   foreach nti in necessaryToken of V
      v=nti->relation;
      v.checked=true;
      v.decisionType=false;
      deletionTraceCheck(v);
   end foreach
}
```

(그림 14) 가변성 의존관계 추적 알고리즘

또는 제거 결정을 내리게 되며, 이에 의존관계를 맺고 있는 다른 도메인 액티비티의 선택 또는 제거 결정이 이루어질 수 있다. 각 의존관계 유형별 결정에 따른 가지치기 과정을 <표 3>에서 보여주고 있다.

(그림 14)는 가변성 의존관계 분석모델을 기반으로 가변점 결정시 미리 선택되거나 제거될 수 있는 가변점들을 추적하기 위한 알고리즘이다.

여기서 *selectionTraceCheck(v)*는 가변점을 가진 액티비티를 '선택'결정한 경우 충분조건 의존관계를 추적하여 관계를 가진 모든 가변점을 '선택'결정한다. *deletionTraceCheck(v)*는 필요조건 의존관계를 추적하여 관계를 가진 모든 가변점을 '제거'결정한다.



(그림 15) Free Parking System의 출차를 위한 BPFM

4. 사례연구 및 평가

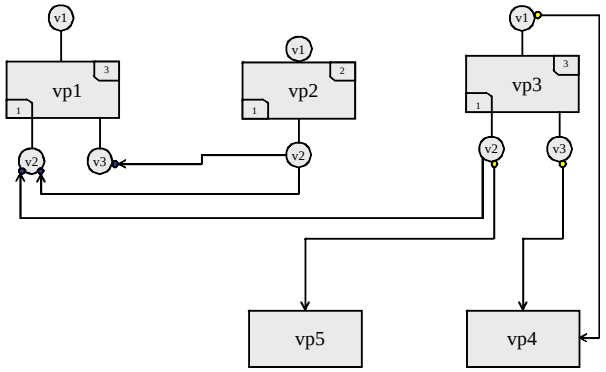
4.1 사례연구

(그림 15)는 Free Parking System의 출차 프로세스를 모델링하고 있는 BPFM이다. (그림 16)은 이로부터 추출된 가

	결정횟수	결정지점		
Case 1	**결정1✓**	vp1	가변치 v1선택결정 (v2, v3 제거)	-필요조건 의존관계추적: vp2의 v2 자동제거, vp3의 v2자동제거, vp5 자동제거
	결정2	vp2	X	-v2 제거로 인해 자동 v1선택
	결정3✓	vp3	가변치 v1선택	-충분조건 의존관계추적: v1선택되면 vp4 자동선택, v3선택되면 vp5 자동선택
			가변치 v3 선택	
	결정4	vp4	X	
	결정5	vp5	X	
	총 2번 결정			
Case 2	**결정1✓**	vp1	가변치 v1, v2, v3 선택결정	추적 없음
	결정2✓	vp2	가변치 v1, v2 선택결정	추적 없음
	결정3✓	vp3	가변치 v1선택	-충분조건 의존관계추적: v1, v3 선택되면 vp4 자동선택, v3 선택되면 vp5 자동선택
			가변치 v2선택	
			가변치 v3 선택	
	결정4	vp4	X	
	결정5	vp5	X	
	총 3번 결정			

변성 의존관계 분석모델이다. vp1, vp2, vp3은 selection decision 가변점이고, vp4, vp5는 Boolean decision 가변점이다. vp2 주차요금 계산방식에서 자동주차요금계산 방식이 결정되기 위해서는 vp1에서 RFID 태그 인증 또는 카메라에 의한 인증이 선택되어야 한다. 이것은 필요조건 의존관계에 해당된다. 또한 vp3 주차요금납부 중, 후불납부 방식이 결정되기 위해서는 vp1의 RFID태그 인증이 먼저 결정되어야 한다. 이것 또한 필요조건에 해당된다. 그러나 주차요금 후불납부 방식은 청구서 발행 활동을 수행토록 해야 하기 때문에 vp5로 충분조건 의존관계를 설정해야 한다.

이알고리즘을 통한 가변성 결정 횟수를 분석하면 다음과 같다. (그림 16)에서 vp1의 v2가 선택되지 않은 경우에, v2에 필요조건 의존관계 되어 있는 vp3의 v2가 자동 제거되고, vp3의 v2에 충분조건 의존관계 되어 있는 vp5가 제거된다. 또한 vp1의 v3도 같이 선택되지 않은 경우에, vp2의 v2가 필요조건으로 의존하고 있는 모든 관계가 없어지기 때문에 제거된다. 이런 경우 vp2에서의 결정도 더 이상 필요 없게 된다. vp3에서의 결정은vp4와 vp5의 선택 여부를 미리 결정할 수 있기 때문에 vp3 이후의 결정은 더 이상 일어나지 않는다. 이러한 경우, 전체 5번의 결정에서 2번의 결정으로 줄일 수 있다. 또한 vp1의 RFID 가변치 액티비티(v2)가 선택되지 않은 경우임에도 불구하고 vp3의 v3이 결정이 되고 vp5가 선택되는 기능의 불일치도 막을 수 있다.



(그림 16) 사례연구(그림 15)에 대한 가변성 의존관계 분석모델

4.2 평가

본논문에서 제시한 가변성 의존관계 분석 기법을 평가하기 위하여 본 논문의 접근법을2장 기반연구에서 기술하였던 관련연구 [17]의 Orthogonal Variability Model (OVM), [18]의 COVAMOF Variability View (CVV), Jan Bosch의 가변성 의존관계를 형식화 한 연구[21]와 feature 기반 의존관계 분석 연구[22]와 비교한다. 비교할 항목은 가변성 의존관계가 영향을 미치는 기본모델, 의존관계 설정을 통한 기능 일치성 고려 여부, 가변점에 바인딩 될 가변치 수 고려 여부, 그리고 제시하는 의존관계의 유형, 가변성 결정 메커니즘의 유무, 가변성 결정횟수 감소 여부이다.

<표 4>에 나타나 있는 바와 같이 OVM 연구에서는 가변성 모델을 특정 하나의 기본모델과 연결 짓지 않고 있다. 이는 하나의 가변성 모델이 여러 기본모델에 공통적으로 적용될 수 있는 장점이 있지만, 각 기본모델의 모델요소들의 특징을 반영한 가변성을 모델링하기는 어렵다는 단점이 있다. 모든 관련연구에서 의존관계 설정 시, 기능일치성을 고려하고 있었으며, 의존관계 유형을 제시하였다. 그러나 CVV 연구에서는 단지 의존상관관계(dependency interaction)로만 가변성 간의 관계를 나타내었다. 관련연구들에서는 이렇게 의존관계를 설정하기 위한 기법에만 초점을 두고 있었으며,

〈표 4〉 본 연구 접근법과 관련연구의 비교

비교연구 및 항목	기본모델	기능일치성고려	바인딩가변치수 설정	의존관계유형	가변성결정 메커니즘	결정횟수 감소
OVM [17]	Product family artifacts	O	O	-Requires -Excludes	X	-
CVV [18]	Architecture	O	O	-Dependency interaction	X	-
Bosch 연구[21]	-	O	X	-Include -Exclude	X	-
Feature기반 의존성분석[22]	Component model	O	X	-Usage -Modification	X	-
본 논문 접근법	BPFM	O	O	-Sufficient -Necessary	O	감소

가변성 의존관계 설정을 통한 가변성 결정 메커니즘에 대해서는 기술하지 않았다. 그로 인해 가변성 의존관계 추적이 어려웠고, 또한 가변성 의존관계 설정으로 인한 가변성 결정횟수가 어느 정도 감소할 수 있는지를 평가할 수 없었다.

5. 결 론

본 논문에서는 SOA 애플리케이션 개발에 맞게 비즈니스 프로세스의 유연성을 확보하고 재사용을 증진시키기 위해 선행 연구된 비즈니스 프로세스 패밀리 모델을 기반으로 가변성 사이의 의존관계를 분석하기 위한 방법을 제시하였다. 먼저 BPFM에 포함된 각 가변성 결정유형에 따라 표현 방법을 제시하고 이를 독립된 의존관계 분석모델을 위한 구성요소로 추출해 내었다. 추출된 가변성 정보는 충분조건 의존관계와 필요조건 의존관계를 분석하여 의존관계 분석모델로 나타내었다. 이를 바탕으로 가변성 결정이 영향을 미치는 범위의 가변성들을 추적할 수 있는 결정관계 추적 알고리즘을 제공하였다. 본 방법을 이용함으로써 의존성 결정회수를 줄일 수 있었고 잘못된 가변성 결정으로 인한 BPM의 기능 불일치를 해소할 수 있음을 보였다.

현재 이클립스 (eclipse) 플러그인인 GMF (Graphical Modeling Framework) [19]를 통해 개발된 BPFM 도구에 가변성 의존관계 구성요소들을 자동 추출해 내고, 가변성 의존관계 정보를 표현할 수 있도록 도구를 확장 개발 중이다 [20]. 이 도구는 결정-가지치기-적응(Decision-Pruning-Adaptation: DePA) 과정을 끊어짐 없이 지원해 줄 수 있게 될 것이다.

참 고 문 헌

[1] 전병선, *SOA What and How: A Road to SOA*, 와우북스, 2008.

[2] M.K. Moon, M.W. Hong, and K.H. Yeom, "Two-level Variability Analysis for Business Process with Reusability and Extensibility", Proceedings of 32nd Annual IEEE International Computer Software and Applications Conference, July. 2008.

[3] P. Clements, and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison Wesley, 2001.

[4] H. Gomma, *Designing Software Product Lines with UML, From Use Cases to Pattern-Based Software Architectures*, Addison-Wesley, 2004.

[5] M.K. Moon, K.H. Yeom, and H.S. Chae, "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability in a Product Line," IEEE Transactions on Software Engineering, Vol.31, No.7, pp.551-569, Jul., 2005.

[6] M.K. Moon, H.S. Chae, and K.H. Yeom, "A Metamodel Approach to Architecture Variability in a Product Line", Proceedings of 9th Int. Conf. ICSR 2006, Lecture Notes in Computer Science Vol.4039, pp.115-126, Jun., 2006.

[7] M.K. Moon, H.S. Chae, K.H. Yeom, and J. Park, "Two-dimensional Framework for Analyzing Variabilities in Software Product Lines", Proceedings of the 7th Int. Conf. on Computer and Information Science, 2008.

[8] P. Wohed, W.M.P van der Aaslt, M. Dumas, A.H.M.ter Hofstede, and N. Russel, "On the Suitability of BPMN for Business Process Modelling", Proceedings of the 4th Int. Conf. on Business Process Management, Sep., 2006.

[9] R. Eshuis, R. Wieringa, "Comparing Petri Net and Activity Diagram Variants for Workflow Modelling-A Quest for Reactive Petri nets", Petri Net Technology for Communication -Based Systems, LNCS 2472, pp.321-351, 2003.

[10] Business Process Modeling Notation Specification 1.0, OMG Final Adopted Specification, 2006.

[11] N. Russell, W. van der Aalst, A. ter Hofstede and P. Wohed, "On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling", Proceedings of the 3rd Asia-Pacific Conf. on Conceptual Modelling (APCCM 2006), Springer Verlag, 2006.

[12] C. Ouyang, M. Dumas, S. Breutel, and A.H.M. ter Hofstede, "Translating Standard Process Models to BPEL", Proceedings of the 18th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2006), June, 2006.

[13] K. Kang,S. Kim,J. Lee,K. Kim,G.J. Kim, and E. Shin,"FORM: A Feature-Oriented Reuse Method with Domain Specific Reference Architectures", Annals of Software Engineering, Vol.5, No.1, pp.143-168, 1998.
[14] C. Atkinson, et al., *Component-based product line engineering with UML*, Addison-Wesley, London, New York, 2002.
[15] H. Gomaa, M.E. Shin, "A Multi-View Meta-modeling Approach for Variability Management in Software Product Lines", Proceeding's of 8th Int. Conf. (ICSR8), Madrid, Spain, LNCS Vol.3107, pp.274-285, 2004.
[16] H. Gomaa, *Designing Software Product Lines with UML*, From UseCases to Pattern-Based Software Architectures, Addison-Wesley, 2004.
[17] M. Sinnema, S. Deelstra, and J. Nijhuis, J. Bosch, "COVAMOF: A Framework for Modeling Variability in Software Product Families", Proceedings of the Second Software Product Line Conference (SPLC4), LNCS Vol.3154, pp.197-213, 2004.
[18] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering: Foundations*, Principles, and Techniques, Springer, 2005.
[19] Graphical Modeling Framework Home, http://www.eclipse.org/gmf/index.php
[20] M.W. Hong, M.K. Moon, and K.H. Yeom, "An Automatic Business Process Model Generation Tool Using Business Process Family Models", *Journal of KIISE : Software and Applications*, Vol.35, No.8, pp.479-492, 2008. (in Korean)
[21] M. Jaring and J. Bosch, "Architecting Product Diversification - Formalizing Variability Dependencies in Software Product Family Engineering", Proceedings of the 4th Int. Conf. on Quality Software, 2004.
[22] K. Lee and K. C. Kang, "Feature Dependency Analysis for Product Line Component Design", Proceedings of 8th Int. Conf. ICSR 2004, Lecture Notes in Computer Science Vol.3107, pp.69-85, Jun., 2004.

문 미 경

e-mail : mkmoon@dongseo.ac.kr
1990년 이화여자대학교 전자계산학과(학사)
1992년 이화여자대학교 전자계산학과 (이학석사)
2005년 부산대학교 컴퓨터공학과 (공학박사)
2005년 3월~2005년 8월 부산대학교 차세대물류IT기술사업단 박사 후 연구원
2005년 9월~2006년 8월 부산대학교 컴퓨터 및 정보통신 연구소 기금교수
2006년 9월~2008년 2월 부산대학교 정보컴퓨터공학부 연구교수
2008년 3월~현 재 동서대학교 컴퓨터정보공학부 전임강사
관심분야 : 소프트웨어 프로덕트 라인 공학, SOA기반 소프트웨어 개발, RFID 미들웨어 개발 등