http://dx.doi.org/10.3745/KTSDE.2013.2.11.739

A Variability Description Technique for Software Product Line: OVDL

Lee Ji Hyun† · Kang Sung Won††

ABSTRACT

Variability of the software product line that differentiates member products within a product line must be described with precise meaning and visualized so as easy to select. Moreover, it should be easy to manage. Variability description approaches can largely be divided into two approaches, integrated variability description approach and orthogonal variability description approach. Orthogonal Variability Description Language (OVDL) was developed for clear and precise description of variability without ambiguity. This paper validates the variability description capability of OVDL by translating the variability models of Inter-Working Function (IWF) product line described by using Orthogonal Variability Model (OVM) notations into variability descriptions in OVDL.

Keywords : Software Product Line, Variability, Orthogonal Variability Description

소프트웨어 프로덕트라인 가변성 기술 기법: OVDL

이 지 현† · 강 성 원††

요 약

소프트웨어 프로덕트라인에서 가변성은 프로덕트라인의 멤버제품들을 차별화하는 특성으로, 가변성의 기술은 의미가 명확하고 선택이 용이하도록 가시화되어야 하며 관리하기 용이하여야 한다. 가변성 기술은 크게 개발 모델에 통합하는 방법과 개발 모델과는 독립적으로 기술하는 방법으로 나눌 수 있다. OVDL (Orthogonal Variability Description Language)은 독립적 가변성 기술 방법이면서 타 방법들의 가지는 의미의 모호성을 개선하기 위하여 제안된 가변성 표현 언어이다. 이 논문은 독립적 가변성 기술 방법의 기반을 제공한 OVM (Orthogonal Variability Model)으로 기술된 IWF (Inter-Working Function) 프로덕트라인의 가변성이 OVDL로 번역될 수 있는지 확인하는 방식으로 OVDL의 가변성 기술 역량을 검증한다.

키워드 : 소프트웨어 프로덕트라인, 가변성, 독립적 가변성 기술

1. 서 론

소프트웨어 프로덕트라인 공학 (Software Product Line Engineering, 이하 SPL)은 소프트웨어 제품군 (product family)에서 개발할 멤버제품들 간의 공통성 및 차별화된 특성인 가변성을 개발의 전 라이프사이클에 걸쳐 체계적으로 계획하고 재사용하여 생산성 향상, 비용절감, 시장적시성 향상을 꾀하는 소프트웨어 개발 패러다임이다. 변화시키는 능력 또는 경향을 의미하는 가변성은 SPL에서 사전에 정의된 수정 가능한 산출물들을 재사용하여 맞춤형 (customized) 어플리케이션 개발을 가능하게 하는 핵심개념이다[10]. 따라서 가변성은 선택(바인딩)이 용이하도록 가시화되고 관리되어야 한다. 가변성을 구성하는 가변점, 가변값, 의존관계 (dependency), 그리고 제약조건 (constraint)을 가시화하고 검증하기 위하여 지금까지 여러 가지 가변성 모델, 가변성 기술언어가 제안되어 왔다[1-8].

SPL에서 가변성 모델링은 크게 휘처 모델, 요구사항 모델, 아키텍처 등과 같은 특정한 모델링 언어를 확장하여 가변성을 통합하여 모델링하는 방법과, 모델링 언어 독립적으로 가변성을 기술하는 독립적 가변성 모델링 방법으로 나눌 수 있다. 전자의 대표적인 연구들이 FORM (Feature- Oriented Reuse Method)[1], UML 확장을 통하여 요구분석 모델부터 아키텍처 및 상세설계 모델들까지 가변성을 포함하도록 하여 기술하는 PLUS (Product Line UML-Based Software Engineering)[2], 아키텍처 기술 언어인 ADL을 확장한 Koala[3], Koalish[4], Kumbang[5] 등이다. 이들 연구들은 가변성모델이 특정 모델에 제한되어 있어 모델링 언어마다 서로 다른 공통성 기호와 가변성 기호를 사용해야 하기 때문

※ 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(NRF-2013R1A1A3005162).
† 정 회 원 : 대전대학교 교양학부대학 조교수
†† 종신회원 : 한국과학기술원 전산학과 부교수
논문접수 : 2013년 3월 15일
수 정 일 : 1차 2013년 5월 22일, 2차 2013년 6월 6일
심사완료 : 2013년 6월 19일
* Corresponding Author : Lee Ji Hyun(jihyun30@dju.kr)

에, 사용자가 서로 다른 많은 언어를 배워야 하는 부담이 있고, 공정의 다양한 단계에 사용될 적절한 가변성 모델링 언어들을 사용자가 직접 찾아야 한다[11]. 후자의 연구로는 K. Pohl[10]등이 SPL 전 라이프사이클 동안의 통일된 가변성 표현을 위해 제안한 OVM (Orthogonal Variability Model), KobrA 방법[6]을 포함하여 많은 연구에서 논의된 의사결정 모델링 (decision modeling)[7], OMG에서 표준화가 진행 중인 CVL (Common Variability Language)[8], Kconfig와 CDL (Component Description Language)[9] 등이 있다.

이 두 종류의 연구는 공통적으로 다음과 같은 문제점이 있다. 첫째, 언어가 엄밀히 정의되어 있지 않다는 것이다. 어떤 경우에는 모호성이 있고 다른 경우에는 완전히 정의되어 있지 않다. 둘째로, 제품군의 모델링을 위한 언어요소들이 타입을 가지도록 요구되지 않고, 특히 원소타입과 집합타입을 엄밀히 구별하지 않아서 해석에 모호성이 발생한다. 따라서 도메인 모델의 사용자가 올바른 해석을 하는 부담을 갖게 되며, 그 결과로서 이를 사용한 모델들이 의사소통의 매체 역할을 효과적으로 하지 못하게 된다. 이러한 제약사항을 해결하기 위하여 소프트웨어 모델링 언어를 SPL 모델링 언어로 확장하는 직교적 가변성 기술 메커니즘 (OVDM: Orthogonal Variability Description Mechanism)이 제안되었고 가변성 모델링 언어인 OVDL이 정의되었다[11].

OVDL이 활용되려면 기존의 검증된 가변성 모델링 언어 수준의 표현력을 가지는 것이 우선 중요하다. 본 논문은 이를 확인하기 위하여 언어 독립적인 가변성 모델링 방법의 가변성 표현력과 OVDL을 비교한다. 많은 방법들 중에서 K. Pohl[10] 등이 언급한 독립적인 가변성 모델링 방법인 OVM은 유럽에서 진행된 ITEA 프로젝트의 주요 연구결과를 정리한 결과이고, 유럽을 중심으로 진행된 나머지 방법들의 기반을 제공한 연구라고 볼 수 있다. 따라서 본 논문에서는 독립적인 가변성 모델링 방법 중 OVM을 대상으로 OVDL의 가변성 표현 역량을 검증하고자 한다.

이하 논문은 다음과 같이 구성된다. 2장에서는 관련연구로써 비교하고자 하는 가변성 모델링 기술 언어인 OVM과 OVDL, 그리고 적용사례인 IWF (Inter-Working Function)를 설명한다. 3장에서는 논문에서 답을 얻고자 하는 질문과 선정 근거를 제시한다. 4장에서는 OVM으로 기술된 IWF의 가변성을 OVDL로 매핑하면서 질문에 대한 답을 찾아 나간다. 마지막으로 5장에서는 두 모델링 방법을 비교 분석한 결과를 요약하고 결론을 기술한다.

2. 관련 연구

2.1 OVM 개요

가변성은 개발 라이프사이클이 진행되면서 복잡도가 증가하며 가변값, 가변점, 개발 산출물들 간의 상호관계의 수도 함께 증가한다[10]. 따라서 가변성이 개발 산출물에 포함되어 있는 경우 가변성 관련 정보에 대한 일관성 유지 및 각 도메인 산출물의 추상화 수준에 맞는 개발이 어렵다. 또한 가변성은 복잡도가 높은 소프트웨어 개발 모델들을 더욱 복잡하게 만들 수 있다. 무엇보다도 가변성이 개발 산출물에 포함되어 있는 경우 가변성에 대한 전체 정보를 파악하기 어렵고 심지어 가변성 정의를 모호하게 만들기도 한다. 이에 OVM은 가변성 표현에 초점을 두고 도메인 공학 산출물에 포함되어 있는 가변성만을 별도로 추출하여 기술하는 가변성 기술 방법이다. OVM은 가변대상을 표현하기 위해 Fig. 1과 같은 그래픽 형식의 표기법 정의하고 있다.

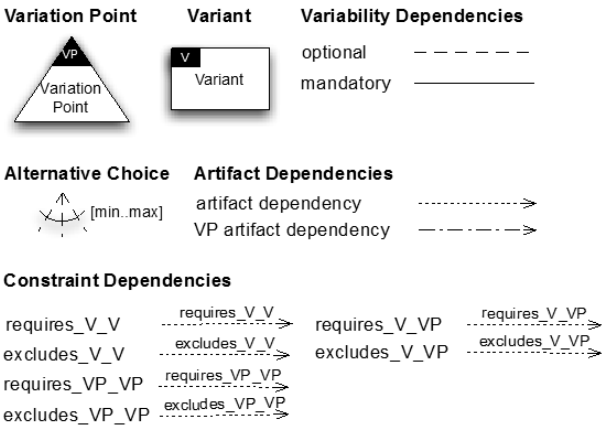

Fig. 1. OVM Notation[10]

OVM 개념은 의사결정 모델링, 수정된 휘처 모델링 방법을 포함하여 pure::variants와 같은 가변성 관리 도구에서 응용되었다[7]. Fig. 1의 표기법은 개념적 정의이며 KobrA[6], CVL[8], Kconfig와 CDL[9] 등에서는 수정된 표기법을 사용하고 있다.

2.2 OVDL 개요

직교적 가변성 기술 방법인 OVDM은 가변성 표현언어 OVDL을 포함하고 있다[11]. OVDL은 텍스트 언어이든 다이어그램 언어이든 관계없이 같은 방법으로 기존의 단일 프로덕트의 모델링 언어를 프로덕트 라인의 모델링 언어로 확장하는데 사용하기 위해 정의된 언어이다. OVDM은 OVM과 마찬가지로 '선택 (optional)'과 '배타적 논리합 (exclusive-or, alternative)'을 기본적인 가변점과 가변값의 의존관계 유형으로 정의하고 있다.

그렇지만, OVDM은 OVM과 달리 가변점을 크게 원소가변점과 집합가변점 그리고 단순가변점과 복합가변점으로 분류하여 정의하고 있다. 원소가변점은 가변점과 가변값의 타입이 동일한 경우에 대한 정의이고, 단순가변점은 하나의 지점이 독립적으로 가변성을 갖는 경우에 대한 정의로 이들은 OVM에서 정의하고 있는 가변점과 동일하다. 반면, 집합가변점은 Fig. 2와 같이 가변값이 동일 타입의 원소들로 이루어진 집합(집합타입)인 경우에 대한 정의이다.

복합가변점은 모델의 복수개의 지점이 하나의 가변점을 형성하며, 이들이 동기화하여 각 가변값에 맞는 값을 갖게 되는 경우에 대한 정의이다. 복합가변점은 Fig. 3C와 같이

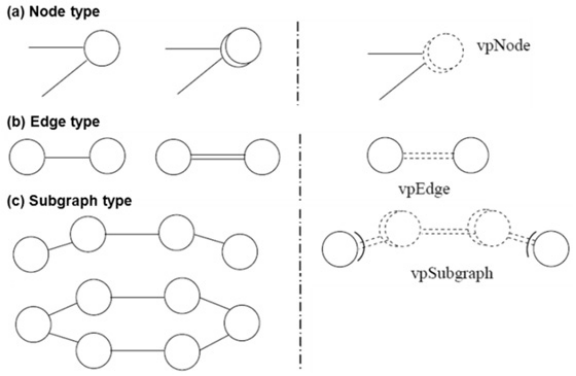

Fig. 2. Collection variability type[11]

가변성이 발생하는 지점이 두 개 혹은 그 이상인 가변점이다. Fig. 3A의 제품들에서 나타난 가변성은 Fig. 3B와 같은 가변성 모델로 기술될 수 있으며 가변점에 바인딩될 수 있는 가변값은 Fig. 3C와 같이 기술될 수 있다.

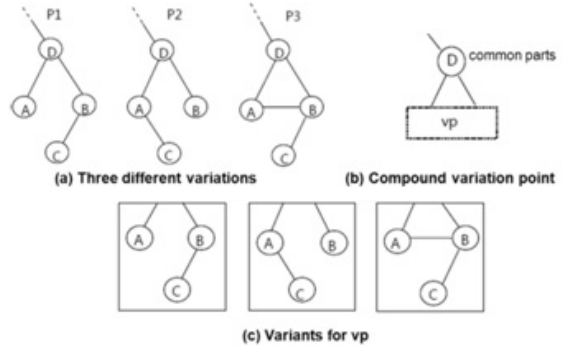

Fig. 3. Compound variability type[11]

원소가변점을 이용한 선택과 배타적 논리합에 대한 OVDL 표기법은 다음과 같다(Fig. 4):

(1) Optional
vpName: *VariantName*, null
(2) Alternative
vpName: $VariantName_1, \ldots, VariantName_k$

Fig. 4. OVDL expression for 'Optional' and 'Alternative' dependency

서로 다른 가변성 유형들의 조합을 위해 OVDL은 집합가변점 이용한다. Fig. 5는 집합가변점에 대한 OVDL 표현이다. *Set1*은 필수 (mandatory)를 표현하고 있고, *Set2*는 *Set1*에 있는 모든 항목을 포함해야 하며 'vpName의 제약조건'은 카디널리티 (cardinality)를 포함한다.

Fig. 6은 집합가변점을 이용한 포함적 논리합에 대한 OVDL 표현 예이다. vpE는 세 개의 가변값을 가지며 카디널리티에 의해 가변값 선택이 제한된다. 따라서 Fig. 6의 vpE는 세 개 중 하나 또는 두 개의 가변값을 선택할 수 있다.

$Set1 \subseteq vpName \subseteq Set2$
/* followed by */
Constraints on *vpName*
(Set1 may be omitted if an empty collection is a valid)

Fig. 5. OVDL expression for collection variability

$vpE \subseteq \{E1, E2, E3\}$
$1 \leq |vpE| \leq 2$

Fig. 6. OVDL expression for an example inclusive-or dependency

가변성 표현에서는 가변성 의존관계에 대한 표현과 더불어 가변성 제약조건 표현을 위한 표기법 또한 중요하다. 앞서 설명한 바와 같이 제약조건은 제품군의 모든 멤버제품에 해당되기 때문에 제품군의 제약조건은 OVDM의 일부이다. OVDL에서 제약조건은 OVM과 마찬가지로 '요구(requires)'와 '제외(excludes)'의 두 가지이다. Fig. 7은 OVDL이 제공하는 제약조건 표기법이다.

Requires:
- (vp1) req (vp2, v)
- (vp1) req (vp2)
- (vp1, v1) req (vp2, v2)
- (vp1, v) req (vp2)

Excludes:
- (vp1) excl (vp2, v)
- (vp1) excl (vp2)
- (vp1, v1) excl (vp2, v2)
- (vp1, v) excl (vp2)

Fig. 7. OVDL expression for variability constraints[11]

2.3 IWF (Inter-Working Function) 개요

IWF[12]가 동작하는 도메인은 특정 구역 내에 설치된 장치에 결함이 발생한 경우, 치명적 결과를 초래할 수 있고 문제 발생 시 즉각적으로 관련 기술자 파견이 용이하기 않은 환경이다. 따라서 설치된 장치, 서비스의 상태 변화 혹은 결함을 지속적으로 원격 모니터링하고, 필요 시 복구 등의 원격 제어 기능이 요구된다. 원격유지보수를 수행하기 위해서는 대상 장치의 상태정보를 수집하고 모니터링하며 필요한 제어를 수행할 수 있어야 한다. 그러나 장치 미들웨어들은 각기 다른 요청방식, 메시지 포맷, 전송방식 등을 정의하고 있어 상호운용이 용이하지 않다. 이에 원격 유지보수 시스템과 다양한 대상 시스템 사이에서 완충 역할을 할 수 있는 IWF를 정의하였다. IWF는 장치 미들웨어와 자원, 장치, 네트워크 등의 지능화된 구성관리를 수행하는 미들웨어 간의 상호운용을 가능하게 하는 모듈이다(Fig. 8).

IWF는 상호운용할 시스템 (예, EMS, VRS 등)들이 공통으로 요구하는 기능과 시스템에 따라 차별화된 기능들이 존재한다. 또한 기능 별로 입력 정보의 형태 및 정보 요청 방

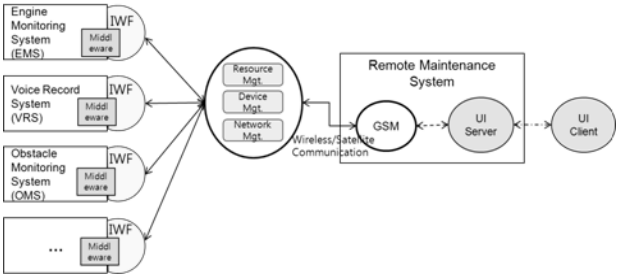

Fig. 8. Conceptual structure of remote maintenance service[12]

법 등에서도 차이가 존재한다. 동일 시스템이라 할지라도 브랜드에 따라 상호운용에 필요한 기능 및 사용 정보에 차이가 존재한다. IWF 설계에 프로덕트라인을 적용하여 가변적인 요구들을 유연하게 수용할 수 있는 구조를 정의하고 현재의 멤버시스템뿐만 아니라 향후, 추가될 시스템들에도 적응이 용이하도록 설계되었다.

3. 질문 정의

본 논문은 OVM으로 표현이 모호했거나 복잡했던 가변성 기술이 OVDL로 표현했을 경우, 모호성 및 복잡도로 인한 확장성 (scalability) 문제가 해결될 수 있었는지를 IWF 프로덕트라인의 가변성 모델을 통하여 확인하고자 한다. 다음은 논문에서 답을 얻고자 하는 질문과 선정한 근거이다:

- *문1 (가변성 표현 역량).* OVDL은 OVM이 정의한 모든 가변성을 표현할 수 있는가? 먼저 OVDL이 OVM에서 정의한 가변성 모델링 표기법을 정의하고 있는지 확인이 필요하다.
- *문2 (의미 전달 역량).* OVDL은 가변성 의존관계, 가변성 제약조건의 의미를 모호성 없이 정확하게 표현할 수 있는가? OVDL이 실제 SPL에서 활용되기 위해서는 우선 OVDL이 OVM과 동등하거나 더 우수한 가변성 표현력, 특히 의미의 정확한 전달 역량을 가져야 한다. 이는 간접적으로 OVDL을 검증할 수 있는 방법이다.
- *문3 (확장 역량).* OVDL이 수백 개에 이르는 경우의 가변성을 표현할 수 있는가? OVDL이 OVM의 중요한 단점으로 제기되는 모델의 복잡도로 인한 확장성 문제를 해결할 수 있는지 확인하는 것이 중요하다.

Table 1. OVDL expressions for OVM variability relationships

OVM variability relationship	OVDL representation
Exclusive-or /Alternative	*vp*: $e_1,\cdots, e_k$
Inclusive-or	*vp*: $v = v' \subseteq \{e1,.., ek\}$ $min \le \|v'\| \le max,\ min \ge 1$
Optional	(1) Used without *inclusive-or* *vp*: *e*, *null* (2) *Inclusive-or optional* Used with *inclusive-or* and its range is [min.. max], *vp*: $v \subseteq v' \cup \{e\}$ $v' \subseteq \{e1,\cdots, ek\}$, $min \le \|v'\| \le max$ (3) *Exclusive-or optional* *vp*: *e1*,⋯,*ek* (same meaning with *exclusive-or*)
Mandatory	(1) Used without *inclusive-or* *vp*: *v* /* special case of *exclusive-or* */ (2) Used with *exclusive-or* and range is [min..max], *vp*: $\{e\} \subseteq v \subseteq \{e, e1,\cdots, ek\}$ $min+1 \le \|v\| \le max+1$ (3) In the case of being used with *exclusive-or* its meaning is same with case(1).
Requires	(*vp1*) req (*vp2*, *v*), (*vp1*) req (*vp2*), (*vp1*, *v1*) req (*vp2*, *v2*), (*vp1*, *v*) req (*vp2*)
Excludes	(*vp1*) excl (*vp2*, *v*), (*vp1*) excl (*vp2*), (*vp1*, *v1*) excl (*vp2*, *v2*), (*vp1*, *v*) excl (*vp2*)

4. OVM에서 OVDL로의 매핑

이 장에서는 먼저 OVM 표기법을 OVDL로 일대일 대응시켜보면서 OVM의 OVM을 이용해 기술한 IWF 프로덕트라인의 가변성 모델이 OVDL로 어떻게 표현되는지를 확인하면서 제기한 세 질문에 대한 답을 찾아나가고자 한다.

4.1 문1 (가변성 표현 역량)

본 절에서는 *문1 (가변성 표현 역량)*을 확인하고자 OVM이 정의하고 있는 표기법을 토대로 가능한 가변성 관계 유

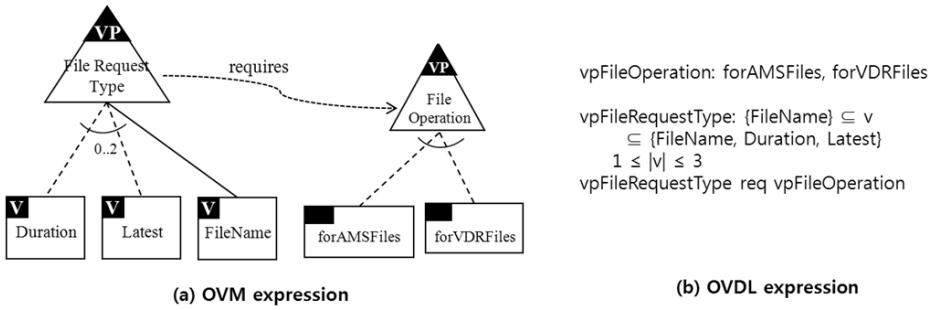

Fig. 9. OVM and OVDL expression for 'File Request' requirement

형을 OVDL로 표현해 본다. Table 1은 Fig. 1의 OVM 기본 표기법과 이들 기본 표기법의 조합으로 가능한 모든 경우 [11, 13]에 대한 표현 결과이다. Table 1에서 보여주듯이 OVDL은 기본적으로 OVM에서 정의된 모든 경우를 표현할 수 있다.

4.2 문2 (의미 전달 역량)

OVM이 제공하는 그룹핑 메커니즘은 휘처 트리로 기술된 가변성 모델에 존재할 수 있는 잘못된 해석의 문제를 해결할 수 있다고 확인된 바 있다[10]. 이 절에서는 OVM으로 기술된 IWF의 가변성을 OVDL로 기술함으로써 잘못된 해석의 여지없이 정확한 의미를 전달할 수 있는지에 대한 답을 얻고자 한다. 먼저, 몇 가지 대표적인 경우의 가변성에 대한 OVM 표현을 OVDL이 기술할 수 있는지 확인하는 것을 시작으로 질문에 대한 해답을 점검한다.

Fig. 9A의 OVM 모델은 IWF가 시스템 점검을 위해 기록파일을 요청하는 방법과 관련된 가변성이다. 일부 시스템은 특정 기간 동안에 생성된 파일을 요청(Duration)하고 일부 시스템은 가장 최근에 생성된 파일을 요청(Latest)하기도 한다. 그리고 모든 시스템은 특정 파일이름에 해당하는 파일을 요청(FileName)할 수 있다. 시스템의 파일 정보에 접근하려면 시스템에 맞는 파일 오퍼레이션이 필요하다. 따라서 vpFileRequestType은 vpFileOperation을 'requires'한다.

OVDL이 과거 연구에서 제기되었던 복잡한 가변성 의존관계와 가변성 제약조건을 정확히 표현할 수 있으며, OVDL의 표현력이 기존 가변성 표현 방법들과 같은 수준의 표현력을 가지고 있음은 이미 확인된 바 있다[11]. 이제 복잡한 IWF 요구사항 중 하나인 가변성 사례-1을 예로 들어보기로 한다.

> **가변성 사례-1:**
> IWF는 오류 복구(fault recovery)가 필요한 경우, IWF가 설치되는 시스템의 종류에 따라 유지보수 시스템은 파일을 전송하거나 재부팅한다. 파일을 전송하는 경우라도 상호운용할 시스템에 따라 파일 적용 후 재부팅을 하거나 리프레쉬(refresh)를 필요로 하기도 한다.

OVM을 이용한 가변성 기술에서는 이를 Fig. 10의 두 가지 경우와 같이 기술할 수 있다.

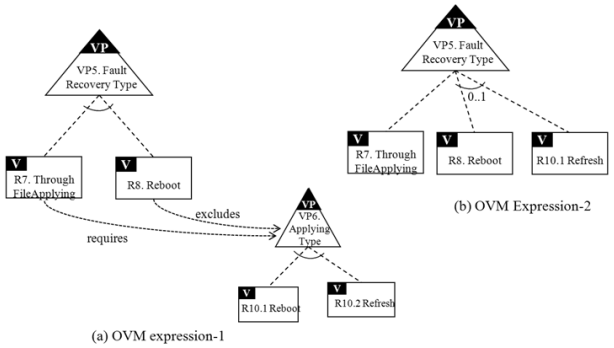

Fig. 10. OVM expressions for the Variability Case-1

Fig. 10A의 경우 가변성 사례-1을 정확히 표현할 수 있다. 그렇지만 'Reboot' 가변값이 서로 다른 요구사항과 관련되어 두 번 나오게 된다 (사실, OVM의 그룹핑 메커니즘은 이러한 문제를 회피할 수 있다고 생각되어 왔다). 이를 피하기 위하여 Fig. 10B와 같이 표현할 경우 {}, {ThroughFileApplying}, {Reboot}, {Refresh}, {ThroughFileApplying, Reboot}, {ThroughFileApplying, Refresh}의 6가지 경우의 인스턴스가 가능하다. 그렇지만 Fig. 10B 표현은 'ThroughFileApplying' 가변값이 'Reboot' 또는 'Refresh'를 'requires' 함에도 {ThroughFileApplying}이 가능하게 되어 잘못된 해석이 된다. 무엇보다도 Fig. 10B의 표현은 가변성 제약조건을 정확히 기술하지 않고 있으며 이는 그로 인해 발생한 문제라고도 볼 수 있다. 이와 같이 OVM으로 엄밀히 표현하는 것이 용이하지 않은 반면, OVDL의 집합가변점 Fig. 11을 이용하여 표현하면 OVM의 Fig. 10A와 Fig. 10B와 같은 표현에서 발생하는 문제를 해결하면서 표현할 수 있다.

```
vpFaultRecoveryType: v ⊆ {ThroughFileApplying} ∪ v'
                     v' ⊆ {Reboot, Refresh}
                     |v'| = 1
```

(a) OVDL expression-1

```
vpFaultRecoveryType: {ThroughFileApplying, Reboot},
                     {ThroughFileApplying, Refresh},
                     {Reboot}
```

(b) OVDL expresson-2

Fig. 11. OVDL expressions for the Variability Case-1

4.3 문3 (확장 역량)

다음으로 한 가변성의 가변점, 가변값들과 다른 가변성의 가변점, 가변값들의 관계가 복잡한 경우의 예를 통하여 *문3 (확장 역량)* 문제에 대한 답을 얻고자 한다.

> **가변성 사례-2:**
> 오류 복구를 위해 유지보수 시스템은 파일을 전송하거나 재부팅할 수 있다. 파일 전송을 하는 경우에는 파일이 적용되기 이전의 상태로 되돌리는 롤백 기능을 필요로 한다. 반면 재부팅을 통하여 오류 복구를 하는 경우 원격 재부팅 컨트롤 기능이 필요하다. IWF는 원격 유지보수 시스템이 이 기능을 수행하도록 하기 위한 상호연동 기능을 가져야 한다.

OVM은 가변성 모델의 복잡도와 함께 확장성 제어하기 위해 가변점들을 통합하고 해당 가변점들의 가변값들의 바인딩을 사전에 정의한 추상화된 가변점들인 패키지 정의가 가능하다. 패키지들 중 하나를 선택하면 여러 개의 가변점이 선택된다[10]. Fig. 12는 OVM의 패키지 가변점을 이용한 기술이다.

Fig. 13은 OVDL을 이용한 기술이다. OVDL은 복잡도가 높은 가변성을 표현할 수 있을 뿐만 아니라 OVM의 패키지 가변점이 보여주는 것과 같은 의미적 모호성 없이 표현이 가능하다는 장점이 있다.

Table 2. Summarized answers for each question

Questions	OVM	OVDL
Q1. Variability representation capability	OVM represents variation point, variants, variability dependencies, and variability constraints together with graphic notations.	OVDL represents variation point, variants, variability dependencies, and variability constraints in an enumerated manner. And OVDL can describe all OVM expressions.
Q2. Semantic precision	OVM avoids misinterpretation by describing variability separately with development models[11], but IWF case study revealed that OVM might lead users to misinterpretation.	OVDL expresses variability of IWF case without leading users to misinterpretation.
Q3. Scalability	OVM expresses variability constraints with a network with no associated levels. This makes variability modeling be complicated even a slight increase of variability. In the case that the number of variability increases and variability relationships becomes complicated, the complexity of variability model grows very high, so even an intuitive grasp of variability become low. OVM defines 'packaged variants' concept, but OVM does not define its notation.	PFML that includes OVDL has a hierarchical structure, so OVDL expressions included in different product line model[11] may have hierarchical relationships. OVDL thus can represent the large numbers of variability easier than OVM, and it is easier to handle the complexity of variability constraints than OVM. Because OVDL represents variation point and their variants in an enumerated manner, it is easy to express the large numbers of variability. OVDL can control the abstraction levels of complicated variability relationships by using collection variability. OVDL defines compound variability, but the utility of compound variability could not be confirmed because variability case related to it did not exist in IWF case.

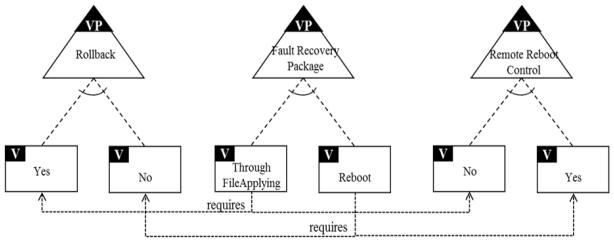

Fig. 12. OVM expression for the Variability Case-2

```
vpRollback: Yes, No
vpRemoteRebootControl: Yes, No
vpFaultRecoveryPackage: TroughFileApplying, Reboot

(vpFaultRecoveryPackage, TroughFileApplying) req (vpRollback, Yes)
(vpFaultRecoveryPackage, TroughFileApplying) req (vpRemoteRebootControl, No)
(vpFaultRecoveryPackage, Reboot) req (vpRollback, No)
(vpFaultRecoveryPackage, Reboot) req (vpRemoteRebootControl, Yes)
```

Fig. 13. OVDL expression for the Variability Case-2

OVM이 가변성 기술의 복잡도를 제어하기 위해 '패키지'라는 개념을 언급하고 있긴 하지만 이 개념이 별도의 정의를 가지고 있진 않다. 'requires' 관계를 이용한 정의이기 때문에 가변성 관계가 복잡한 경우 또는 가변값의 수가 많아지는 경우 모델의 복잡도가 높아지면서 확장성 문제가 발생할 수밖에 없다. 반면 OVDL은 직관적인 정보전달은 안되지만 가변성 정의와 'requires' 제약조건를 사용하여 OVM이 기술하고자 하는 가변성을 확장성 문제없이 표현할 수 있다.

5. 토의 및 결론

OVDL은 타입과 집합을 이용한 가변성 기술 방법으로 모델링 언어에 관계없이 그 언어로부터 PFML (Product Family Modeling Language)을 정의하는데 사용되며 가변성 기술의 복잡도를 줄이면서 정확성을 확보하기 위한 목적으로 정의되었다[11]. OVM을 이용하여 가변성을 기술한 IWF 사례에 OVDL을 적용한 결과, OVDL은 OVM보다 직관적인 이해는 어려웠지만 OVM이 가진 만큼의 가변성 표현력을 가지고 있었다. 복잡도가 높은 가변성 모델의 경우, OVM보다 정확한 표현이 가능하며 다수 개의 복잡한 가변성 관계의 표현에서는 OVM보다 용이하다는 장점이 있었다. 3장에서 제시한 3가지 질문에 대한 답은 Table 2와 같이 정리될 수 있다.

지금까지 가변성 기술언어인 OVDL이 유사한 독립적 가변성 모델링 방법인 OVM의 가변성 표현력을 가지고 있는지 확인하였다. OVM은 개발 산출물들에 포함되어 있는 가변성들을 따로 분리하여 가시화하고 관리하고자 하는 방법이다. OVM은 휘처 모델링 등과 같이 공통성과 가변성을 함께 표현하는 방법보다 단순하면서도 용이하게 가변성의 종류와 범위를 SPL 엔지니어들에게 전달할 수 있다는 장점이 있다. OVM이 지향하는 개념은 다양한 의사결정 모델링[7] 방법 및 OMG의 CVL[8]로 이어지고 있다. 그렇지만 OVM은 실제 SPL 적용에 사용될 경우 복잡도와 확장성 문제를

가지고 있다. OVDL은 원소타입과 집합타입을 명확히 구분하고 가변성 기술방법을 단순화하여 이러한 문제를 해결하고자 하였다. 또한 OVDL이 지금까지 활용되고 있는 OVM 만큼의 표현력을 가지는지에 대하여도 OVM을 이용하여 기술한 IWF를 OVDL로 표현한 결과 OVDL은 OVM 가변성 표현을 모두 표현할 수 있었을 뿐만 아니라 복잡했던 OVM 표현과 OVM에서 모호했던 표현을 보다 정확히 표현할 수 있었다.

OVDL은 기존 모델링 언어를 SPL 지원 모델링 언어로 확장하는 방법에서 제안한 제품군의 가변성 모델링 언어를 도출하는 방법인 OVDM[11]을 적용하여 도출한 가변성 기술 언어이다. OVDL이 OVM과 동등한 표현력을 가지면서 OVM의 단점을 해결할 수 있음은, 기존의 단일제품을 위한 요구사항 명세 언어, 아키텍처 기술 언어 등을 SPL 지원 언어로 확장할 때도 OVDM이 잘 적용될 수 있음을 의미한다. 또한, OVDL은 휘처 모델이나 OVM의 한계를 보완하기 위한 가변성 기술 언어로써 상호보완적으로 활용될 수 있으며, 향후 OVDM을 이용한 기존 제품 모델링 언어를 제품군 모델링 언어로의 확장에서 가변성 표현을 지원할 수 있을 것으로 기대된다.

그렇지만, 가변성은 제품라인 개발 라이프사이클 전체에 걸쳐 정제되고 상세화되므로, 가변성 표현 역량에 대한 비교만으로 OVDM이 가지는 개발 라이프사이클 단계별로 다른 도메인 산출물들과의 추적성 유지, 바인딩 정보, 다양한 가변성 메커니즘 지원 등의 역량에 대하여 결론을 내릴 수는 없다. 향후, 개발 산출물과의 추적 링크, 바인딩, 제약사항을 포함하여 제품라인 라이프사이클 전체에서의 OVDM 역량을 휘처 모델링을 포함한 의사결정 모델링, CVL 등과 비교, 검증할 예정이다. 또한, 스마트 홈 네트워크 시스템인 HERA 시스템에 SPL 적용 사례를 활용하여 OVDL에 대한 추가적인 검증도 병행할 예정이다.

참 고 문 헌

[1] 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," in *Annals of Software Engineering*, Vol.5, 1998, pp.143-168.

[2] H. Gomaa, *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architecture,* Addison-Wesley Professional, 2004.

[3] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee, "The Koala Component Model for Consumer Electronics Software," *IEEE Computer*, Vol.33(3), 2000, pp.78-85.

[4] T. Asikainen, T. Soininen, and T. Mannisto, "A Koala-Based Approach for Modelling and Deploying Configurable Software Product Families," in *Software Product-Family Engineering*, Vol.3014, 2004, pp.225-249.

[5] T. Asikainen, T. Soininen, and T. Mannisto, "Kumbang: A domain ontology for modelling variability in software product families," in *Advanced Engineering Informatics*, Vol.21, 2007, pp.23-40.

[6] C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Peach, J. Wust, and J. Zettel, *Component-based Product Line Engineering with UML*, Addison-Wesley, 2002.

[7] K. Czarnecki, P. Grunbacher, R. Rabiser, K. Schmid, and A. Wasowski, "Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches", in *Proceedings of VaMoS'12*, 2012.

[8] Common variability language (CVL), OMG Initial Submission. Available on request., 2010.

[9] T. Berger, S. She, R. Lotufo, A. Wasowski, and K. Czarnecki. Variability Modeling in the Real: A Perspective from the Operating Systems Domain. in *Proceedings of the 25th IEEE/ACM Conference on Automated Software Engineering*, ACM, 2010, pp.73-82.

[10] K. Pohl, G. Böckle, F. van der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer, 2005.

[11] S. Kang, "A Method for Extending Software Modeling Languages to Languages for Modeling Families of Software," *The Korean Institute of Information Scientists and Engineers (KIISE) Journal of Software Engineering Society*, Vol.23(2), Jun., 2010.

[12] J. H. Lee and D. Lee, "IWF Design for Remote Maintenance Service using Software Product Line", in *Proceedings of Korea Conference on Software Engineering (KCSE2011)*, 2011.

[13] K. Czarnecki, "Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models", Ph D. dissertation of Computer Science and Automation, Technical University of Ilmenau, Oct., 1998.

이 지 현

e-mail : jihyun30@dju.kr

1993년 전북대학교 공과대학 정보통신공학과(학사)

2000년 전북대학교 교육대학원 전자계산교육학과(석사)

2005년 전북대학교 자연과학대학 컴퓨터과학과(박사)

2005년~2008년 한국정보통신대학교 연구조교수

2009년~2011년 한국과학기술원 전산학과 연구조교수

2012년~현 재 대전대학교 교양학부대학 조교수

관심분야 : Software Product Line, Software Testing, Software Architecture Evaluation

강 성 원

e-mail : sungwon.kang@kaist.ac.kr

1982년 서울대학교 사회과학대학(정치학사)

1989년 Univ. of Iowa 전산학(전산학석사)

1992년 Univ. of Iowa 전산학(전산학박사)

1993년~2001년 한국통신 연구개발본부 선임연구원

1995년~1996년 미국 국립표준기술연구소(NIST) 객원연구원

2001년~2005년 한국정보통신대학교 조교수

2003년~현 재 미국 Carnegie-Mellon University 소프트웨어 공학석사과정 겸임교수

2005년~2009년 한국정보통신대학교 부교수

2009년~현 재 KAIST 부교수

관심분야 : Software Architecture, Enterprise Architecture, Software Testing, Formal Method