Journal of the Korea Industrial Information Systems Research Vol. 20 No. 3, Jun. 2015 :51-60
http://dx.doi.org./10.9723/jksiis.2015.20.3.051 ISSN:1229-3741

OVM 중심 가변성 추적 방법에 대한 효용성 검증

(Efficiency Validation for the OVM-based Variability Tracing Method)

이 지 현[1], 황 선 명[2)*]
(Jihyun Lee and Sunmyung Hwang)

요 약 추적성은 이해당사자들이 변경으로 인한 산출물들 간의 영향을 분석하는데 필요한 정보를 제공하는 것을 주요 목적으로 한다. 단일 제품 개발과 달리 제품군(群)을 대상으로 하는 소프트웨어 프로덕트라인은 도메인 공학과 어플리케이션 공학의 두 개발 라이프사이클이 있으면서, 가변성과 두 라이프사이클 간의 추적성을 유지하고 관리해야 하기 때문에 그 복잡도가 매우 높다. 이에 개발 산출물과 별도로 가변성을 관리하는 직교적 가변성 모델을 중심으로 하는 가변성 추적성 유지 방법이 개념적으로 제안된 바 있다. 그렇지만, 이 방법이 소프트웨어 제품라인에서 필요로 하는 추적성을 모두 설정할 수 있는지에 대해서는 확인되지 않았다. 본 논문에서는 직교적 가변성 모델을 중심으로 하는 추적성 유지, 관리 방법이 필요한 추적성을 모두 지원하는지 예제를 통하여 검증하였다. 그 결과 OVM 중심 가변성 추적 방법은 변경으로 인해 영향을 받을 수 있는 산출물들의 범위를 한정하는 데는 문제가 없었다. 그렇지만, 변경으로 인해 실제 영향을 받는 구체적이고 정확한 산출물들을 추적하지는 못했다.

핵심주제어 : 소프트웨어 프로덕트라인, 추적성, 가변성, 직교적 가변성 모델

Abstract Traceability targets provision of information to stakeholders required for analyzing impacts among artifacts due to changes. Unlike single product development, in software product line developing the family of products the complexity of maintaining and managing traceability between two life cycles, domain and application engineering is so high. Accordingly, variability traceability management approach centred on orthogonal variability model that manages variability separated from development artifacts has been conceptually proposed, but its efficiency has not verified yet. This paper verifies whether orthogonal variability model based traceability can provide required traceability through an example. As the results, the OVM-based variability tracing method supports well to narrow down artifacts affected by the changes. However, the method does not support tracing the exact artifacts or exact part of an artifact affected by the change.

Key Words : Software product line, Traceability, Variability, Orthogonal variability model

* Corresponding Author : sunhwang@dju.kr
+이 논문은 2013학년도 대전대학교 신진교수학술연구비 지원에 의해 연구되었음.
Manuscript receivrd May 11, 2015 / revised June 24 ,2015 / accepted June 30, 2015
1) 대전대학교 혜화리버럴아트칼리지, 제1저자
2) 대전대학교 컴퓨터공학과, 교신저자

1. 서 론

추적성은 요구사항, 아키텍처, 상세 설계, 구현 등 개발 라이프사이클 단계의 산출물들 간 추적 정보를 제공하는 능력을 의미한다. 추적성은 이해당사자들이 변경으로 인한 산출물들 간의 영향을 분석하는데 필요한 추적 정보를 제공하는 것을 주목적으로 한다. 보통 추적성은 순방향 즉, 이전 단계(source stage) 산출물에서 다음 단계(target stage) 산출물로의 추적 링크와 역방향 즉, 다음 단계 산출물에서 이전 단계 산출물로의 추적 링크, 그리고 이 둘을 모두 포함하는 양방향 추적 링크로 구분할 수 있다[1]. 중요도에 따라 순방향 또는 역방향을 포함한 양방향 추적성을 유지할 수 있다. 메트릭, 네비게이션 그래프 등과 같은 추적성 관련 정보[2-3]와 가변성 관리에 필요한 지식 등[4]이 관련 이해당사자에게 적절히 제공되어야 한다.

소프트웨어 프로덕트라인(SPL; software product line)의 경우[5], 추적성은 재사용 플랫폼과 그로부터 제품을 개발하고 이들을 변경, 유지, 관리를 위해 (1) 도메인/어플리케이션 공학 라이프사이클 단계 내에서 산출된 산출물들 간 추적성, (2) 서로 다른 도메인/어플리케이션 공학 개발 라이프사이클 단계들에서 산출된 산출물들 간 추적성, (3) 정의된 가변성과 관련 제품들 간의 순방향(forward) 또는 역방향(backward) 링크에 대한 유지 능력 차원으로 나눌 수 있다.

단일 제품 개발과 달리 제품군(群)을 대상으로 하는 SPL은 도메인 공학과 어플리케이션 공학의 두 개발 라이프사이클이 있으면서 가변성과 두 라이프사이클 간의 추적성을 유지하고 관리해야 하기 때문에 그 복잡도가 매우 높다. 이에 개발 산출물과 별도로 가변성을 관리하는 직교적 가변성 모델을 중심으로 하는 가변성 추적성 유지 방법이 개념적으로 제안된 바 있다[5]. 그렇지만, 이 방법이 SPL에서 필요로 하는 추적성을 모두 관리할 수 있는지에 대해서는 확인되지 않았다. 이에 본 논문에서는 직교적 가변성 모델을 중심으로 하는 추적성 유지, 관리 방법이 필요한 추적성을 모두 지원하는지 예제를 통하여 검증한다.

논문의 구성은 다음과 같다. 먼저 2장에서는 관련연구를, 3장에서는 SPL에서 관리가 필요한 추적성 측면과 논문에서 확인하고자 하는 문제를 정의한다. 그런 다음 OVM(orthogonal variability model) 중심 가변성 추적 기법을 소개한다. 4장에서는 검증 환경과 검증 결과를 설명한다. 마지막으로 5장에서는 결론을 기술한다.

2. 관련연구

SPL에서 관리되어야 할 추적성은 SPL 추적성 디멘젼과 앤드투앤드 추적성 연구를 통하여 정의되었다[4,6,7]. 이 연구들을 통하여 직교적으로 관리되어야 할 도메인 공학, 어플리케이션 공학, 가변성 차원에서의 추적성 디멘젼과 상위수준(high-end), 하위수준(low-end) 추적성이 정의되었다. 이 연구들에서 정의한 디멘젼은 상호 독립적 특징을 갖는 것으로 동일 또는 유사한 요소에 대한 다른 표현인 어스펙트와는 다르다. Mohan 등[4]이 정의한 상위수준과 하위수준 추적성은 중요도, 프로젝트 특성, 비용 등에 따라 추적 링크가 유지하는 지식을 제공한다. 이 연구는 SPL에서 가변성 관리가 한 수준으로 관리되기 어렵기 때문에 그 수준을 어떻게 나누어야 할지에 대한 지식 프레임워크를 제시한 연구로써 상당히 가치가 있다.

Jirapanthong et al.[8]은 충족관계(satisfiability relation), 의존관계(dependency relation) 등 Mohan 등이 정의한 하위수준의 관계에 해당하는 9개의 추적성 유형을 정의하고 있다. 이들은 추적되어야 할 서로 다른 두 산출물 요소들 간에 존재할 수 있는 관계의 종류를 정의하고 이 정보를 추적 링크에 포함시키고 있다. 그렇지만, 이들 연구들은 실제 변경관리(변경 영향 분석)에 이러한 추적성 디멘젼, 추적성 관계를 어떤 방식으로 활용할지에 대한 언급이 없다.

가변성뿐만 아니라 공통성까지 포함하는 SPL 도메인 전체를 모델링 대상으로 하고 많은 연구에서 활용되고 있는 피처 모델[9]을 중심으로 가변성 추적이 가능한지에 대한 검증을 수행한 연구가 있다[10-11]. 피처모델은 가변성을 명시적으

로 기술하지 않으면서, 서로 다른 추상화 수준 간의 관계에 대한 명확한 정의가 없기 때문에 (라이프사이클 단계 별 추상화 수준마다 서로 다른 피처모델을 사용한다 하더라도 이 관계는 명확이 정의되지 않음) 단계마다 생성되는 개발 산출물과의 추적성 유지가 어렵다. 이에 Berg el al.[10]의 연구는 문제영역으로부터 솔루션영역으로의 1대1 추적성을 달성할 수 있도록 하는 개념적 가변성 모델을 제시하여 SPL에서의 복잡한 추적성 문제를 단순화시키고자 하였다. Mohalik et al.[11] 역시 피처 모델로부터 컴포넌트들로의 추적성을 유지하고 이를 지원하는 툴인 SPLANE를 구현하였다. 그렇지만 이 연구들은 요구사항이 아키텍처로, 아키텍처가 다시 상세설계, 구현, 시험으로 정제되는 과정과 이 과정에서 새로운 가변성에 계속 정의될 수 있음을 배제한 매우 한정된 가정 하에서의 추적성 문제만을 다루고 있다.

또 하나의 중요한 추적성 어스펙트인 비기능 추적성은 안전성, 성능, 사용성과 같은 비기능 요구사항과 관련된 추적성을 의미한다. Cleland-Hyang et al.[12]은 목표 중심 비기능 요구사항 추적성 방법을 제안한 바 있다. 이 연구는 비기능 요구사항과 관련된 기능 요구사항이 변경되었을 때 비기능 요구사항으로의 영향을 분석할 수 있도록 둘 간의 상호관계를 정의한다. Mirakhorli et al.[13-14]은 아키텍처적 의사결정과 품질속성 또는 피처 간의 비기능 추적성 유지방법을 제시하고 있다. 그렇지만 이들 역시 비기능 가변성과 다대다로 관계가 형성되는 개발 산출물들과의 효과적인 추적성 방법을 제시하고 있지는 못하다.

현실적인 추적성 관리가 되려면 조직의 자원, 비용, 제품의 중요도에 따라 추적성 레벨 조정이 가능해야 한다. 산출물들 간의 단순한 추적 링크만을 수립, 유지, 관리하는 낮은 레벨부터 추적성 유형들 간의 계층적 관계 정보와 시멘틱 정보를 포함한 추적성을 수립, 유지, 관리하는 상위 레벨의 추적성이 가능하다[15-17]. 지금까지 SPL 에서의 추적성 연구는 이러한 측면에 대한 고려 없이 추적성 문제를 다루어 왔다. 따라서 추적성 레벨과 그에 따른 추적성이 유지되는 방법, 변경 영향 분석 시 활용되는 방식 등이 명확이 정의되어야 한다.

3. OVM 중심 가변성 추적 기법

본 장에서는 SPL에서 유지되어야 하는 추적성의 종류를 기술하고 이들 추적성 중 논문에서 확인하고자 하는 추적성을 선택하여 문제를 정의한다. 그리고 OVM 중심 가변성 추적 기법을 설명한다.

3.1 SPL에서의 추적성

3.1.1 도메인 공학에서의 추적성

SPL 멤버들이 재사용할 플랫폼을 개발하는 도메인 공학에서 유지되어야 할 추적성은 다음과 같다:

- **도메인 공학의 상호 추적성(Inter traceability in DE):** 도메인 공학의 서로 다른 라이프사이클 단계(즉, 도메인 요구공학, 도메인 아키텍처, 도메인 상세설계, 도메인 구현, 도메인 시험) 산출물들 간의 추적성을 의미한다. 도메인 공학 단계 간의 추적성은 방향을 갖지 않아도 된다. 그렇지만 서로 다른 단계의 도메인 산출물 간의 추적성 유형은 다양할 수 있다[6].
- **도메인 공학의 인트라 추적성(Intra traceability in DE):** 도메인공학의 동일 라이프사이클 단계 내 산출물들 간의 추적성을 의미한다. 이 추적성의 경우 동일 단계 내에서 같은 피처, 요구사항에 대한 다양한 유형의 산출물들이 존재할 수 있으므로 다양한 유형으로 기술된 동일 피처, 요구사항에 대한 추적성을 유지하자는 것이다.
- **도메인 공학의 추적성(Traceability in DE):** 도메인공학의 상호추적성과 도메인공학의 인트라 추적성에서 다루어지지 않는 기타 도메인공학 측면에서의 추적성을 의미한다. 이 추적성은 SPL 계획, 조직 차원의 SPL 목표 구현 현황 관제를 위한 메트릭, 품질

관리 등과 같은 조직관리 프로세스 그룹의 산출물과 관련 도메인공학 라이프사이클 단계 산출물들과의 추적성을 포함한다.

3.1.2 어플리케이션 공학에서의 추적성

도메인 공학 동안 개발된 플랫폼을 이용하여 개별 멤버 어플리케이션을 개발하는 어플리케이션 공학에서 유지되어야 할 추적성은 다음과 같다:

- **어플리케이션 공학의 상호 추적성(Inter traceability in AE):** 어플리케이션공학의 서로 다른 라이프사이클 단계의 산출물들 즉, 어플리케이션 요구공학, 어플리케이션 아키텍처, 어플리케이션 상세설계, 어플리케이션 구현, 어플리케이션 시험 산출물들 간의 추적성. 도메인 공학에서와 마찬가지로 서로 다른 어플리케이션 단계 간의 산출물 간의 추적성 유형이 다양하다.
- **어플리케이션 공학의 인트라 추적성(Intra traceability in AE):** 어플리케이션 공학의 동일 라이프사이클 프로세스 내의 산출물들 간의 추적성을 의미한다.
- **어플리케이션 공학의 추적성(Traceability in AE):** 어플리케이션 공학의 상호 추적성과 어플리케이션공학의 인트라 추적성에서 다루어지지 않는 기타 어플리케이션 공학 측면에서의 추적성을 의미한다. 예를 들어, 개별 어플리케이션 수준에서의 품질 관리, 개발 현황 관제 관련 산출물들 간의 추적성이다.

3.2 문제 정의

Pohl et al.[5]이 정의하고 있는 가변성 메타모델에 의하면 가변점과 개발 산출물 간에는 'represented by' 관계가, 가변값과 개발 산출물 간에는 'realized by' 관계가 존재한다. 또한 Pohl el al.은 가변성 정의 즉, 가변값이 요구사항, 설계, 구현, 시험 산출물 등 가변값 관련 개발 산출물로의 관계를 일관되게 유지하는 역할을 함을 개념적으로 기술하고 있다. 그렇지만 구체적인 구현 방법이나 실제 이러한 방식의 추적 관계 설정이 변경 영향 관리 등 추적성 정보가 중요한 경우에 실질적으로 도움을 줄 수 있는지에 대해 확인된 바 없다. 이에 논문에서는 3.1에서 정의한 SPL에서의 추적성 중 다음의 추적성이 OVM 중심 가변성 추적 방법을 통하여 다음 가변성 변경 영향을 분석할 수 있는지 확인하고자 한다:

- **도메인 공학의 상호 추적성:** A단계에서 정의된 가변값과 관련된 산출물이 B단계에서 상세화된 경우, 가변성 모델의 해당 가변값이 수정되면 A와 B단계의 관련 산출물을 추적할 수 있다.
- **도메인 공학의 인트라 추적성:** 가변값이 A단계의 여러 산출물과 관련되어 있는 경우 가변성 모델의 해당 가변값이 수정되면 관련된 모든 산출물을 추적할 수 있다. 예를 들어, A단계의 여러 산출물을 통하여 구현되었거나, 동일 가변값이 다른 종류의 산출물에서 다르게 표현된 경우의 추적성이 이에 해당한다.
- **어플리케이션 공학의 추적성:** 도메인 가변성 모델에 변경이 행해진 경우, 해당 가변값을 선택한 어플리케이션들의 관련 산출물을 추적할 수 있다.

논문에서는 복잡도를 줄이기 위하여 가변점 변경은 고려하지 않는다.

3.3 OVM 중심 가변성 추적

OVM 중심 가변성 추적 기법은 Fig. 1에서 보여주는 바와 같이 OVM으로부터 SPL 라이프사이클 단계 별 산출물과의 추적성을 유지하는 방법이다. OVM은 SPL 전체에서 정의된 가변성에 대한 독립된 뷰를 제공한다. SPL 라이프사이클 단계 별로 이들 가변성은 산출물로 구현되는데 단계 간 또는 단계 내의 가변성 관련 산출물들 간의 추적성을 OVM을 이용하여 관리하는 방법이다. OVM 중심 가변성 추적 기법은 다음 3가지 추적 링크를 관리한다:

- **도메인 가변성 추적 링크:** 도메인 가변성

모델과 도메인공학 라이프사이클 단계 별 산출물들과의 추적성을 의미한다.

- **어플리케이션 가변성 추적 링크:** 어플리케이션 가변성 모델(멤버 제품에서 선택한 가변값과 멤버 제품에 특화된 가변성을 포함하는 모델)과 어플리케이션공학 라이프사이클 단계 별 산출물들과의 추적성을 의미한다.
- **도메인 가변성 모델과 어플리케이션 가변성 모델 간의 추적 링크:** 도메인 가변성 모델과 어플리케이션 가변성 모델 간의 추적성은 도메인 자산(플랫폼)으로부터 어플리케이션으로의 인스턴스 생성(제품 도출)에 의한 플랫폼과 어플리케이션 산출물 간의 추적성을 유지하기 위해 필요하다.

플랫폼과 어플리케이션 산출물 간에는 순방향과 역방향 추적성이 모두 유지될 필요가 있다. 순방향 추적성은 플랫폼으로부터 어플리케이션 산출물로의 추적성을, 역방향 추적성은 어플리케이션 산출물로부터 플랫폼으로의 추적성을 의미한다. 플랫폼과 어플리케이션 산출물 간의 역방향 추적성은 플랫폼 산출물로부터 바인딩을 통하여 어플리케이션 산출물이 구체화(인스턴스화)되는 관계(구체화-추상화)에 대한 링크이다. 플랫폼과 어플리케이션 산출물 간의 순방향과 역방향 추적성을 모두 유지하는 것은 매우 복잡도가 높은 일이다. OVM 중심 가변성 추적 방법에서는 OVM으로부터 이들 추적성이 유지되므로 이 문제를 단순화할 수 있다.

4. 검증 및 토의

4.1 검증 환경

본 절에서는 OVM 중심 추적성 방법을 검증하기 위하여 사용한 자동차 외등 예제를 설명한다. 북미와 EU, 중동 등 국가에 따라 하이빔, 로우빔, 안개램프 등 자동차 외등에 대한 규정이 다르다. 예를 들어, 전면 안개램프는 안개, 비, 먼지, 눈 등으로 운전자의 가시성이 떨어질 때 운전자의 시야 확보를 위해 사용하는 조명으로, 투과성이 높고 빛의 조사 각도가 넓어 상대편 차량

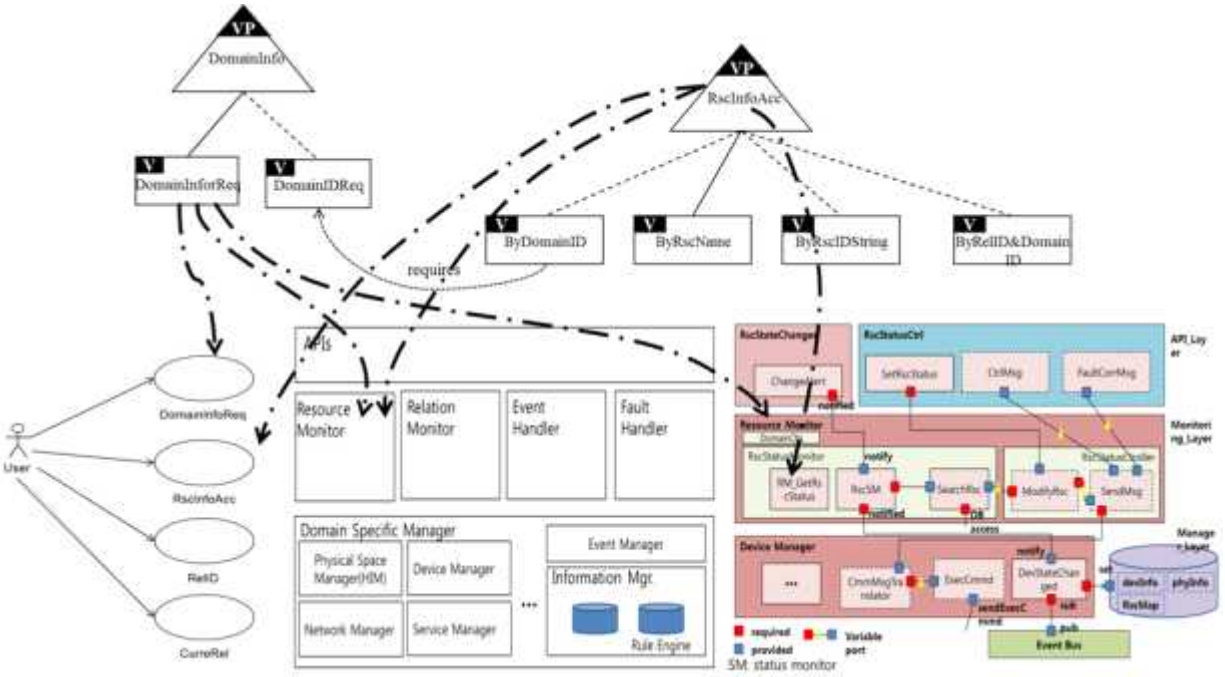

Fig. 1 Concept of OVM-based variability tracing method

에 자신의 위치를 알려주거나 보행자의 존재를 확인하는데 중요한 역할을 한다. 우리나라의 경우, 전조등(헤드라이트)은 백색광을 쓰도록 규정하고 있는 반면, 안개램프는 노란빛과 하얀빛 중 하나를 선택해서 쓸 수 있다. 노르웨이, 스웨덴은 안개램프가 필수이지만 그 외 EU 국가는 선택적으로 장착할 수 있다. 다른 예로 하이빔의 경우 EU는 상대 차량 반사에 대한 규제가 북미보다 상대적으로 약하다. 북미의 경우 반대 쪽 차선의 차량과 마주치는 경우 하이빔 사용을 금지하고 있다. 이와 같은 차이로 인해 자동차 외등의 작동을 제어하는 소프트웨어의 경우 2^n개가 존재할 수 있다. 여기에서 n은 외등의 종류를 의미한다.

이 자동차 외등 제어 소프트웨어 개발에 SPL을 적용, 도메인 스코핑, 도메인 요구공학, 도메인 설계 프로세스를 수행하면서 도메인 아키텍처를 도출하였다. 도메인 설계는 SPL에서 계획한 공통성과 가변성 구현을 가능하게 할 도메인 아키텍처를 개발한다. 도메인 설계는 아키텍처 구조와 텍스처를 포함하는 도메인 아키텍처 생성을 목적으로 한다. 아키텍처 텍스처는 도메인 아키텍처를 사용하는 멤버제품들이 준수해야 할 규칙, 제약조건 등을 명시한 것으로 아키텍처 스타일, 패턴 등을 포함한다.

Fig. 2의 컴포넌트 다이어그램은 ECU(전자 제어 유닛) 하드웨어 레이어와 기본 소프트웨어 레이어, 어플리케이션 소프트웨어 레이어로 설계한 결과이며 AUTOSAR 소프트웨어 아키텍처와 개념적으로 동일하다. 도메인 설계 수준에서 엔지니어들의 의사결정에 따라 제논, 할로겐 이외 LED 또한 광원으로의 니즈가 많을 것으로 예측하고 광원(LightSource)으로 추가하였다. 그리고 여러 ECU 소프트웨어 컴포넌트들을 모드로 관리하기 위한 'ModeManage' 컴포넌트를 추가하였다.

Fig. 3은 하이빔과, 안개램프 동작을 기술한 상태 다이어그램이다. 가변성과 제약조건은 상태 전이선(transition)에 노트를 이용하여 표시하였다.

이러한 라이프 사이클 단계를 거치면서 3.2절에서 정의한 추적성 유형을 OVM을 이용하여 추적 링크를 설정하였다. Fig. 4는 OVM의 가변점(VP) 및 가변값(Variants)와 "represented by", "realized by" 관계에 있는 산출물과의 추적성을 기술한 결과이다. 예를 들어 'BeamConfiguration'

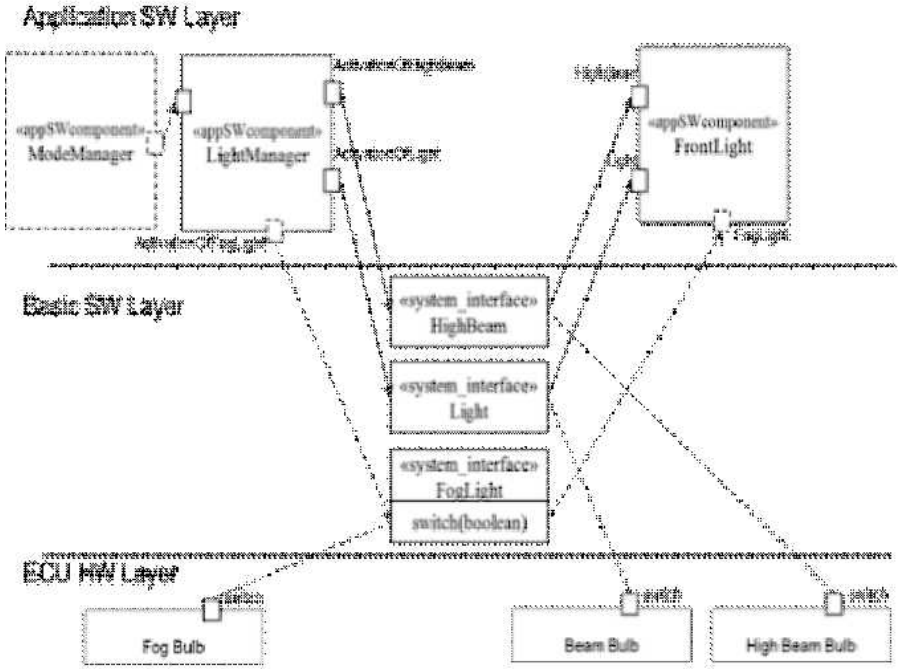

Fig. 2 Example for d omain architecture

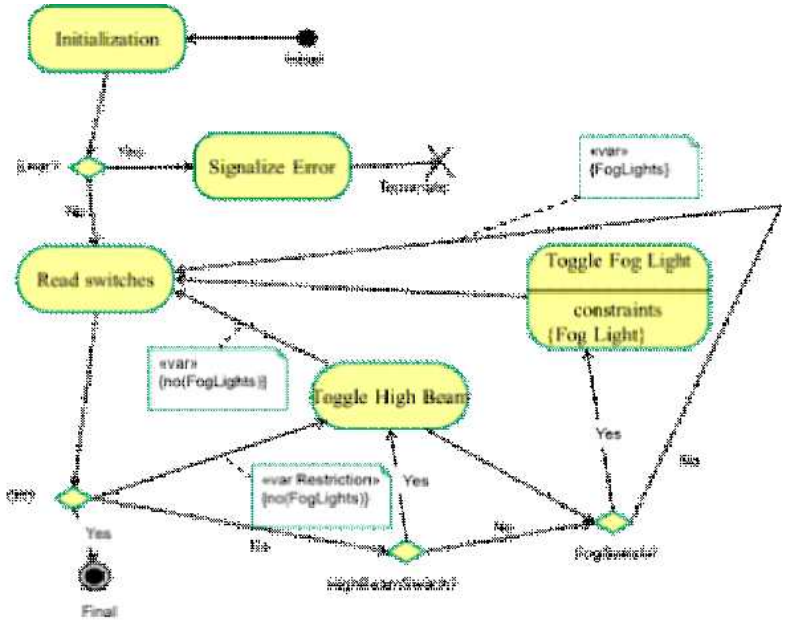

Fig. 3 State diagram in architecture stage

가변점은 C_FrontLight과 CLightManager 컴포넌트들과 'LowBeam' 가변값은 C_FrontLight 컴포넌트의 P_Light 포트와 추적 링크를 가진다. 또한 이 가변값은 상세설계 산출물인 FrontLight 클래스의 변수 V_Light와 추적 링크를 가진다.

먼저, 도메인 공학의 상호 추적성 문제를 확인한다. 'FogLights'를 모드 방식 또는 파라미터 방식 중 하나로 제어해 왔지만 일부 제품에서 두 방식을 동시에 사용하는 방식으로의 전환이 제기되었다. 따라서, 'FogLights' 관련 도메인 산출물이

VP	Variants	Requirements		Architectural Elements			Detailed Design Elements	
BeamConfiguration				C_FrontLight	C_LightManager			
	LowBeam	DR3		C_FrontLight/P_Light	C_LightManger/P_ActivationOfLight	SI_LowBeam	FrontLight/V_Light	LightManager/V_switchLights
	HighBeam	DR2		C_FrontLight/P_HighBeam	C_LightManger/P_ActivationOfHighBe	SI_HighBeam	FrontLight/V_HighBea	LightManager/V_switchFogLigh
	FogLights	DR4		C_FrontLight/P_FogLight	C_LightManger/P_ActivationOfFogLigh	SI_FogLight	FrontLight/V_FogLight	LightManager/V_switchFogLigh
	DaytimeRunningLight							
LightControl								
	Modes			ModeManager				
	ParameterData			C_LightManager/P_SetParas				
AccessableParameter								
	ConstData							
	FixedData							
	CalibratedData							

Fig. 4 Snapshot for OVM-based traceability links

4.2 검증 및 토의

본 장에서는 3.2절에서 정의한 문제를 OVM 중심 가변성 추적 방법이 해결할 수 있는지를 자동차 외등(external car light) 제품라인 예제를 통하여 검증한다.

이를 수용하도록 수정될 필요가 있다. 이와 관련된 가변성은 'BeamConfiguration'과 'LightControl'이다. 이에 수정이 필요한 도메인 산출물들을 추적한 결과 수정 대상 산출물들은 다음과 같다:

아키텍처 단계

- C_FrontLight/P_FogLight
- C_LightManger/P_ActivationOfFogLight
- SI_FogLight
- C_ModeManager
- C_LightManager/P_SetParas

상세설계 단계

- FrontLight/V_FogLights
- LightManager/V_switchFogLights
- ModeManager/S_ModeTypes
- LightManager/S_SetParas

실제 수정이 필요한 도메인 산출물은 추적성 경로에 모두 포함되어 있었다. 그렇지만 추적성 경로 내에 도메인 인트라 산출물까지 모두 포함되어 있어 검토해야할 산출물이 실제 수정이 필요한 산출물(C_LightManger/P_ActivationOf- FogLight, LightManager/V_switchFogLights) 보다 많은 산출물들이 포함되어 있다는 단점이 있었다.

두 번째로 도메인 공학의 인트라 추적성 문제를 확인한다. 아키텍처 단계의 산출물들 간의 추적성이 OVM을 통해서 이루어지는지 확인한다. 'FogLights'는 아키텍처 단계에서 상세화된 도메인 상태 다이어그램의 요소와 다음과 같은 추적성을 가진다:

- state: ToggleFogLight
- transition: FogSwitch? -> state: ReadSwitches

만일 'FogLight' 관련 가변성에 대한 변경으로 인해 영향 받을 가능성이 있는 산출물들을 모두 추출할 수 있다. 그렇지만 실제 변경과 관련된 구체적인 산출물을 정확히 찾아내기는 쉽지 않다. 예를 들어, 'C_FrontLight/P_ActivationOf -FogLight'에 변경이 가해진 경우 해당 산출물과 관련된 산물물을 정확히 알기 어렵다. 왜냐하면 가변점/가변값은 하나의 산출물이나 산출물의 부분과 추적 링크를 가지지는 않기 때문이다. 이는 가변성으로부터의 추적성이 실제 산출물들 간의 상세 수준의 추적성들을 모두 반영하지 못함을 의미한다.

세 번째로 가변성이 'alternative' 관계에서 'optional' 관계로 수정되었기 때문에 가변성 모델을 통해 관련된 어플리케이션 산출물들을 추적할 수 있는지 확인하였다. 가변점 이름은 'LightControl'이었으며 가변값은 'Modes'와 'ParameterData'이다. 문제는 가변점인 'LightControl'을 표현하는 개발 산출물이 정의되어 있지 않다("represented by relation"에 있는 개발 산출물)는 점에 있었다. 도메인 가변성 모델과 어플리케이션 가변성 모델 간의 추적성을 통해 'Modes'와 'ParameterData' 관련 어플리케이션 산출물은 추적이 가능하다. 그렇지만, 실제 도메인에서 이루어진 변경과 영향을 받는 산출물 간의 추적이 정확히 이루어지지 않는다. 가변성 모델에서의 수정과 관계없이 도메인 공학에서의 변경이 어플리케이션 공학으로의 전파됨에 따른 수정은 이루어질 수 있다. 만일 가변점 'LightControl'이 'LightManager'와 "represented by" 관계를 가진다면 도메인 가변성 모델의 변경에 따른 도메인 산출물 변경의 영향이 어플리케이션 모델, 어플리케이션 산출물로 전파되는 추적 경로를 정확히 분석할 수 있다.

5. 결 론

지금까지 OVM 중심 가변성 추적 방법이 정의된 문제, 즉 도메인 공학의 상호 추적성, 도메인 공학의 인트라 추적성, 어플리케이션 공학의 추적성을 어느 정도까지 지원하는지 자동차 외등 예제를 통하여 확인하였다. 그 결과 OVM 중심 가변성 추적 방법은 변경으로 인해 영향을 받을 수 있는 산출물들의 범위를 한정하는 데는 문제가 없었다. 그렇지만, 변경으로 인해 영향을 받는 구체적이고 정확한 산출물들을 추적하지는 못했다. 가변성은 동일 개발 단계에서도 여러 산출물들과 관계가 있고, 같은 산출물이라고 해도 여러 부분과 관계를 가질 수 있다. 가변성에 대한 변경은 여러 산출물과 연관되기도 하지만 특정 산출물에만 연관이 될 수 있다. OVM 중심 가변성 추적 방법은 도메인 공학의 상호 추적성과 인트라 추적성 부분에서 가변성과 관련된 모든 산출물과의 다대다 추적성을 유지해 주기 때문에 관련될 가능성이 있는 모든 산출물들을 추적해 준

다. 이는 장점일 수도 있지만 구체성이 떨어져 변경의 영향을 받는 산출물을 자동으로 추적하였더라도 수동으로 실제 변경의 영향을 받는 산출물을 다시 추적해야 함을 의미한다. 특히, OVM 중심 가변성 추적 방법은 가변성 의존관계가 변경되는 경우에 취약하였다. OVM 중심 가변성 추적 방법은 산출물에서 가변점이 명시적으로 정의되어 있고, OVM과 해당 산출물의 가변점 간의 추적 링크가 정의되어 있다는 전제 하에서만 효용성이 있는 것으로 확인되었다.

향후, SPL에서 유지, 관리되어야 하는 나머지 추적성에서의 OVM 중심 가변성 추적 방법의 효용성을 검증할 계획이다. 또한, 상위수준, 하위수준의 추적성 유지를 지원하도록 제안한 방법을 확장, 정교화해 나갈 예정이다.

References

[1] H. U. Asuncion, F. Francois, and R. N. Taylor, “An end-to-end industrial software traceability tool,” 6th Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on The Foundations of Software Engineering, New York, USA, pp. 115-124, 2007. *(conference)*

[2] A. Sousa, “Traceability support in software product lines,“ Master Thesis, Department of Information system and Computer Science, New University of Lisbon, 2008. *(journal)*

[3] UNL/FCT, Traceability Requirements, “Aspect-oriented, model-driven, product line engineering (AMPLE),“ AMPLE Internal Documentation, 2007. *(journal)*

[4] K. Mohan and B. Ramesh, “Tracing variations in software product families. Communications of the ACM, Vol. 50, No. 12, pp. 68-73, 2007. *(journal)*

[5] K. Pohl, G. Bockle, and F. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques,: 1st ed., Springer-Verlag, Berlin, Heidelberg, 2005. *(book)*

[6] N. Anquetil, U. Kulesza, R. Mitschke, A. Moreira, J. C. Royer, A. Rummler and A. Sousa, “A model-driven traceability framework for software product lines,” Software and Systems Modeling, Vol. 9, No. 4 pp. 427-451, 2007. *(journal)*

[7] J. G. Kim, S. W. Kang, and J. H. Lee, “A comparison of software product line traceability approaches from end-to-end traceability perspectives,” International Journal of Software Engineering and Knowledge Engineering, Vol. 24, No.4, pp. 677-714, 2014. *(journal)*

[8] W. Jirapanthong and A. Zisman, “XTraQue: Traceability for product line systems”, Software and Systems Modeling, pp. 117 - 144, 2009. *(journal)*

[9] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., Peterson, A. S., “Feature-Oriented Domain Analysis (FODA) Feasibility Study,” SEI Technical Report, 1990. *(Technical Report)*

[10] K. Berg, J. Bishop, and D. Muthig, "Tracing Software Product Line Variability -- From Problem to Solution Space," Annual Conference of the South African Institute of Computer Scientists and Information Technologists(SAICSIT05), pp. 182-191, 2005. *(conference)*

[11] S. Mohalik, S. Ramesh, J-V. Millo, S.N. Krishna, and S.K. Narwane, “Tracing SPLs Precisely and Effciently”, 16th international software product line conference, 2012. *(conference)*

[12] J. Cleland-Huang, R. Settimi and O. BenKhadra, “Goal-centric traceability for managing non-functional requirements,” 27th International Conference on Software Engineering, pp. 362 - 371, 2005.

(conference)

[13] M. Mirakhorli, Y. Shin, J. Cleland-Huang and M. Cinar, "A tactic-centric approach for automating traceability of quality concerns," 34th International Conference Software Engineering, pp. 639 - 649, 2012. *(conference)*

[14] M. Mirakhorli and J. Cleland-Huang, "Tracing non-functional requirements," Software and Systems Traceability, Springer-Verlag, pp. 299 - 320, 2012. *(book)*

[15] J. Lee and S. Hwang, "Value-Based Enterprise Architecture Framework: VBEAF," Journal of The Korea Industrial Information System Society, Vol. 19, No. 6, pp. 77-85, 2014. *(journal)*

[16] J. Lee and J. Chung, "Comparative Analysis of the Performance of Robot Sensors in the MSRDS Platform," Journal of The Korea Industrial Information System Society, Vol. 19, No. 5, pp. 57-68, 2014. *(journal)*

[17] S. Hong, E. Han, H. Lee, and J. Kim, "The Conceptual Model for a Co-creation Platform," Journal of The Korea Industrial Information System Society, Vol. 19, No. 3, pp. 127-136, 2014. *(journal)*

이 지 현 (Jihyun Lee)

- 정회원
- 전북대학교 정보통신공학과 공학사
- 전북대학교 전자계산교육학과 교육학석사
- 전북대학교 전산통계학과 이학박사
- 대전대학교 리버럴아츠칼리지 조교수
- 관심분야 : SPL, 소프트웨어 테스팅, SPL 테스팅, 프로세스 개선

황 선 명 (Sunmyung Hwang)

- 정회원
- 중앙대학교 컴퓨터공학과 공학사
- 중앙대학교 컴퓨터공학과 공학석사
- 중앙대학교 컴퓨터공학과 공학박사
- 대전대학교 컴퓨터공학과 교수
- 관심분야 : 소프트웨어품질보증, 프로세스 심사, 테스팅 도구, V&V