KIPS Tr. Software and Data Eng.
Vol.4, No.11 pp.487~498 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2015.4.11.487

Modeling Traceability Between Software Product Line Requirements and Architecture

Seokhwan Eom† · Sungwon Kang†† · Jingyu Kim††† · Seonah Lee††††

ABSTRACT

Traceability enables software developers to trace up the changes occurring in software artifacts. In software product line, traceability is more complex than traceability in a single product as commonality and variability should be considered. Modeling traceability between features and requirements has been proposed in the past. However, traceability between requirements and architecture has more factors to consider, including many-to-many mappings and hierarchical structure of architectures. This paper proposes a method of systematically constructing platform traceability between platform requirements and platform architecture. This paper also shows the efficacy of the proposed mechanism through case studies.

Keywords : Software Product Line, Traceability, Platform, Requirements, Architecture, PFML, PLDT

소프트웨어 제품 라인의 요구사항과 아키텍처 간 추적성 모델링

엄 석 환† · 강 성 원†† · 김 진 규††† · 이 선 아††††

요 약

추적성은 소프트웨어 개발자에게 소프트웨어 산출물에 대한 변경을 추적하게 해 준다. 소프트웨어 제품 라인 개발에 있어서의 추적성은 공통성과 가변성을 고려해야 하기 때문에, 개별 제품에서의 추적성보다 복잡하다. 과거 연구에서 제품 라인 개발에서의 제품 피처와 요구사항 간의 추적성 구축을 제시되었다. 그러나 요구사항과 아키텍처 설계 사이의 추적성 구축은 다 대 다 관계와 아키텍처 계층으로 인해 고려해야 할 요소가 더 많다. 본 논문은 이러한 요소들을 고려하여, 제품 라인 개발에서의 요구사항과 아키텍처 사이의 추적성을 모델링하는 체계적인 방법을 제공한다. 또한 사례연구를 통하여 이 방법이 효과적임을 보인다.

키워드 : 소프트웨어 제품 라인, 추적성, 플랫폼, 요구사항, 아키텍처, PFML, PLDT

1. 서 론

추적성은 소프트웨어 개발자들에게 소프트웨어 산출물에 대한 변경을 추적할 수 있게 해준다. 소프트웨어 제품 라인 개발(Software Product Line Development: SPLD)은 도메인 프로세스와 애플리케이션 프로세스, 그리고 제품 간의 공통성과 가변성을 고려하므로, 추적성 관리가 개별 소프트웨어 프로젝트의 경우보다 복잡하다. 그러나 현재의 소프트웨어 제품 라인 개발의 추적성 연구는 도메인(혹은 플랫폼) 산출물1)들과 애플리케이션 산출물들 간의 추적성을 제한적으로만 제공하고 있다. SPLD에서의 추적성 구축은 임시방편(ad-hoc)에 의존하여 개발자마다의 추적성 구축이 달라질 수 있다.

SPLD의 추적성 연구가 단일제품에 대한 추적성 연구와 다른 점은, 가변성에 관련된 부분이다. 가변성 관련 추적성 연구는 산출물에서의 가변성에 대한 추적성을 제공한다[3, 10]. 그러나 이러한 연구는 산출물 전체에 대한 추적성을 제공하지 않는다. 가변성을 포함하여 산출물 전체에 대한 추적성을 구축하기 위해 체계적인 소프트웨어 제품 라인 개발 연구를 수행하고, 플랫폼에서의 피처와 요구사항 간의 추적성에 대한 모델링을 수행한다[5, 7]. 하지만, 상기 연구는 요구사항과 아키텍처 간의 추적성에 그대로 적용할 수 없다. 왜냐하면 요구사항과 아키텍처는 다 대 다 관계로 맵핑되며

† 비 회 원 : 현대모비스 연구원
†† 종신회원 : KAIST 전산학부 부교수
††† 비 회 원 : 국방과학연구소 선임연구원
†††† 정 회 원 : KAIST 전산학부 연구조교수
Manuscript Received : July 3, 2015
First Revision : August 18, 2015
Accepted : August 18, 2015
* Corresponding Author : Seonah Lee(saleese@kaist.ac.kr)

1) 플랫폼 산출물은 개별 제품들을 인스턴스로 도출할 수 있도록 제품군의 공통 부분과 가변 부분으로 구성된 산출물을 뜻하며, SPLD에서 이러한 산출물들은 도메인 공학 프로세스에 만들어진다.

아키텍처 계층을 추가로 고려해야 하기 때문이다.

본 논문은 소프트웨어 제품 라인 개발에서 플랫폼 요구사항과 플랫폼 아키텍처 사이에서의 플랫폼 수준의 추적성을 모델링하는 체계적인 방법을 제공한다. 본 논문에서는 플랫폼 수준의 요구사항과 아키텍처 산출물의 공통성과 가변성 분석을 통하여 둘 사이의 플랫폼 수준의 추적성 표현 방법과 추적성 모델링 절차를 제안한다. 제안 방법은 언어독립적인 플랫폼 기술 메커니즘인 PFML[6]과 제품 라인 기술 템플릿인 PLDT[6]를 이용하여 플랫폼 추적성을 표현한다. 표현에 있어서는 플랫폼 수준의 요구사항과 아키텍처 산출물을 표현하고, 이 둘 간의 플랫폼 수준의 다 대 다 관계와 아키텍처의 계층을 고려한 추적성을 구축한다. 또한 본 논문에서는 사례연구를 통하여 이들 방법이 플랫폼 수준의 표현을 위한 방법으로 실제로 어떻게 사용되고, 플랫폼 수준의 추적성의 표현에 어떻게 적절히 응용될 수 있는지 보인다.

본 논문의 구성은 다음과 같다. 2절에서는 체계적인 소프트웨어 제품 라인 개발에 대해서 소개하고, 3절에서는 플랫폼 추적성에 대한 접근 방법을 제안한다. 4절에서는 사례연구를 통하여 제안한 방법의 타당성을 확인한다. 5절에서는 추적성에 대한 관련 연구를 소개하고, 마지막으로 6절에서 논문의 결론과 향후 연구방향을 제시한다.

2. SSPLD에서의 추적성

배경 지식으로 Kang, et. al에서 제시한 체계적 소프트웨어 제품 라인 개발(SSPLD: Systematic SPLD)과 SPLD에서의 플랫폼 추적성의 표현 방법을 소개한다[7].

SPLD는 도메인 프로세스와 애플리케이션 프로세스의 단계를 구분하여 개발하는 소프트웨어 개발 방법으로, SSPLD는 이를 좀 더 체계적으로 개발하기 위해 공통성과 가변성의 명시적인 표현, 공통성과 가변성의 명시적인 바인딩과, 이에 기반한 명시적인 추적성을 표현한다. Fig. 1은 SSPLD에서 각 공정 산출물 사이의 관계를 나타낸다.

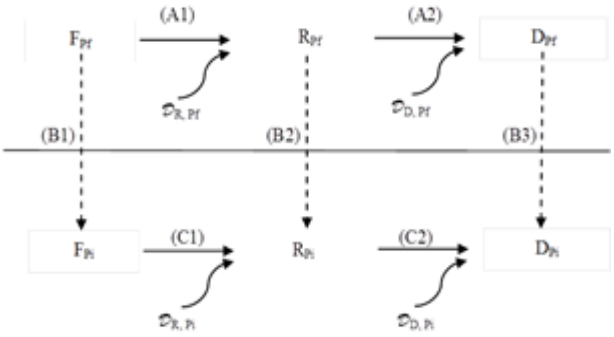

Fig. 1. The Relationships Between Software Artifacts in SSPLD

Fig. 1에서의 F, R, D는 각각 피처, 요구사항, 아키텍처 설계를 나타낸다. 도메인 프로세스에서, 플랫폼 요구사항 RPf는 요구공학의 결정 DR, Pf에 의하여 제품 라인 피처 FPf로부터 도출되고, 플랫폼 설계 DPf는 설계결정 DD, Pf에 의하여 플랫폼 요구사항 RPf로부터 도출된다. 애플리케이션 프로세스에서, 제품 Pi의 요구사항 RPi는 Pi를 위한 요구공학의 결정 DR, Pi에 의하여 Pi의 피처 FPi로부터 도출되고, 제품 Pi의 설계 DPi는 Pi를 위한 설계결정 DD, Pi에 의하여 RPi로부터 도출된다.

Fig. 1에서의 A1과 A2는 도메인 공학의 플랫폼 산출물 간의 추적성을 표현한다. Fig. 1에서의 C1과 C2는 제품 공학의 산출물들 간의 추적성을 표현한다. Fig. 1에서의 B1, B2, B3는 도메인 공학의 플랫폼 산출물과 제품 공학의 제품 산출물 사이에는 인스턴스화 관계(instantiation relationship)를 표현한다. 도메인 공학의 결과물인 플랫폼 산출물 사이와 제품공학의 개별 제품의 산출물 사이에는 추적성 관계(traceability relationship)가 존재한다.

플랫폼 산출물은 가변점을 포함한다. 플랫폼 추적성은 플랫폼 산출물들의 가변점(VP: variation point)들 간의 맵핑을 포함한다. SSPLD에서 가변점은 가변성 유형은 세 가지로 구분된다. 예를 들어, 요구사항의 가변점은 다음 세 경우 중 하나가 된다.

- 선택적(Optional) 요구사항: 하나의 요구사항이 포함되거나 되지 않을 수 있다.
- 택일적(Alternative) 요구사항: 여러 요구사항들 중에서 한 가지만이 선택되어야 한다.
- 집합(Subset) 요구사항: 여러 요구사항이 집합적으로 선택될 수 있다.

제품 산출물은 가변점 대신에 가변값을 대입하여 얻어진다. 바인딩은 플랫폼 산출물의 가변점에 대입할 가변값들을 정의한다. 바인딩은 제품별로 그리고 산출물별로 존재한다.

3. 요구사항과 아키텍처 간의 플랫폼 추적성 모델링 방법

본 논문은 플랫폼 요구사항과 플랫폼 아키텍처 간의 추적성과 이로부터 도출되는 개별 제품의 요구사항과 아키텍처 간의 추적성을 대상으로 한다. 이를 위하여 SSPLD를 이용한 SPLD에서의 요구사항과 아키텍처 간의 플랫폼 추적성 작성 방법을 제안한다. 먼저 요구사항과 아키텍처 간의 추적성의 특징에 대해 알아보고(3.1절), 플랫폼 추적성에서 가변점을 포함하는 추상화된 맵핑의 특징을 고찰한 뒤(3.2절), 플랫폼 추적성을 구축하기 위한 모델링 절차를 제안한다(3.3절).

3.1 요구사항과 아키텍처 간의 추적성

요구사항과 아키텍처 간의 추적성은 소스(source) 요소가 요구사항이고, 대상(target) 요소가 아키텍처 설계에서의 설계 컴포넌트 사이의 맵핑의 집합이다. 여기서 맵핑은 두 개의 요소로 구성된 하나의 쌍으로, 첫 번째 요소인 소스 요소로부터 두 번째 요소인 대상 요소를 도출했음을 표현한

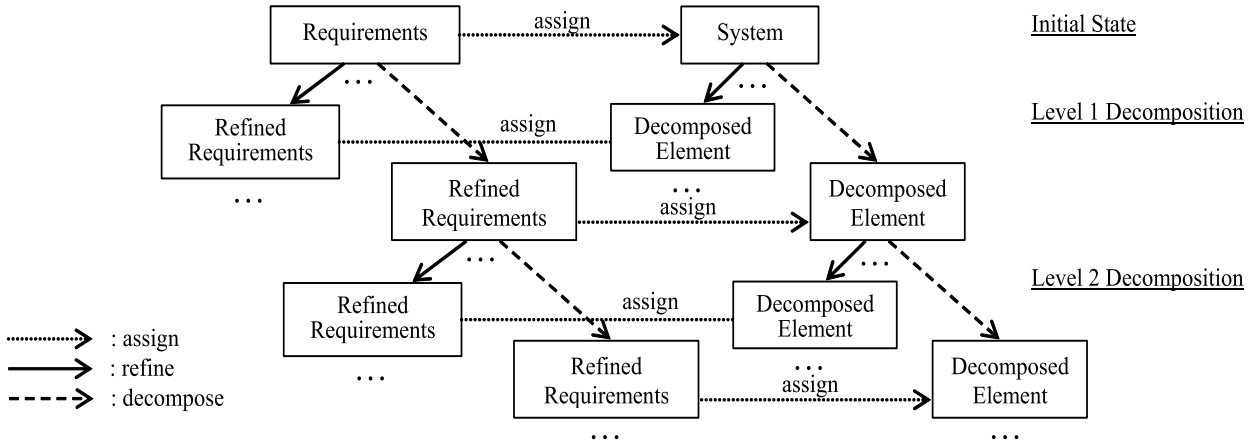

Fig. 2. The Hierarchical Traceability Between Requirements and Software Architecture Design

다. 해당 문맥에서는 소스 요소인 요구사항으로부터 대상 요소인 설계 컴포넌트를 도출했음을 의미한다. 즉, 대상 요소인 설계 컴포넌트는 소스 요소인 요구사항을 만족시킬 책임이 있음을 의미한다.

아키텍처 설계는 주로 계층적으로 구성되므로, 요구사항과 아키텍처 간의 추적성에는 아키텍처 계층을 고려해야 한다. Fig. 2는 아키텍처 설계가 높은 추상 단계에서 점점 낮은 추상 단계로 진행됨을 보이며, 이에 따라 요구사항이 상세화되어 할당되는 것을 보여준다[9]. 아키텍처 설계에서 각 추상 단계에서 상응하는 컨텍스트를 정의하고 시스템을 분할한다. 이 과정에 분할된 요소(element)와 요소 사이의 관계(relationship)가 정해진다. 각 요구사항은 요소 각각에 책임(responsibility)으로 할당된다.

3.2 플랫폼 추적성의 제약조건

제품군 추적성 구축에 있어 두 요소 간에 추적성을 명시할 경우, 가변점으로 인해 가능한 n개의 맵핑 중에서 제품군 추적성에서 존재하지 않는 맵핑이 생성될 수 있다. 따라서 이러한 존재하지 않는 맵핑을 방지하기 위해 제약 조건이 필요하다. 가변점의 유형에 의해 Table 1과 같이 12가지의 맵핑이 있다.[2)]

이러한 유형에 따라 제약사항이 있는 경우와 없는 경우를 표기할 수 있다. 예를 들어, Table 1의 Case 1은 소스요소의 가변점이 택일 가변점(alternative VP)이고 타깃요소에 가변점이 없는 경우이다. 이 경우는 가변점과 하나의 타깃요소 간의 맵핑으로, 가변점이 택일(alternative) 유형이기 때문에 n개의 가변값 중에서 하나의 값으로 대치될 수 있다. 제품군에 따라 n개의 맵핑 중에서 맵핑되지 않는 맵핑이 있을 수 있으므로 제약사항을 작성할 수 있어야 한다.

2) 가변점이 선택(optional)인 경우에 가변값으로 null이 바인딩되어 맵핑이 존재할 수 없다. 12가지의 맵핑에서는 맵핑이 존재하는 경우만을 그 대상으로 하고, 선택(optional) 가변점의 경우에 null이 아닌 가변값이 대치되는 경우만을 포함한다.

Table 1. The Need of Constraints for the Various Relationships Between VPs for Platform Traceability

Case	VP type for source	VP type for target	Need of Constraint
1	Alternative	No VP	YES
2	Alternative	Alternative	YES
3	Alternative	Optional	YES
4	Alternative	Subset	YES
5	Optional	No VP	NO
6	Optional	Alternative	YES
7	Optional	Optional	NO
8	Optional	Subset	YES
9	Subset	No VP	YES
10	Subset	Alternative	YES
11	Subset	Optional	YES
12	Subset	Subset	YES

3.3 요구사항과 아키텍처 간 플랫폼 추적성 모델링

Fig. 2는 플랫폼 요구사항과 플랫폼 아키텍처 간의 추적성이, 아키텍처가 갖는 계층성으로 인하여 역시 계층적인 구조를 갖게 됨을 보인다. Fig. 2의 맨 오른쪽은 아키텍처 설계가 최초에 시스템 전체를 하나의 블랙박스로 보아 출발하여, 여러 단계에 걸쳐 분해되어갈 수 있음을 나타낸다. 왼쪽의 요구사항들은 아키텍처 설계가 단계적으로 진행됨에 따라 해당 시스템에 대한 요구사항에서 해당 컴포넌트에 맞는 상세화된 요구사항으로 변형되고, 컴포넌트의 요구사항은 다시 해당 하위 컴포넌트의 요구사항으로 변형된다. 추적성은 이 과정에 어느 요구사항이 어느 아키텍처 요소들로 할당되거나 영향을 준다는 것을 소스와 타깃의 맵핑으로 나타낸다.

Fig. 1이 표현하는 체계적 소프트웨어 제품 라인 개발(SSPLD)에서의 추적성을 바탕으로, Fig. 2가 표현하는 요구사항과 아키텍처 간의 계층적 추적성을 Fig. 3과 같은 단계

로 구축한다. 먼저 사전 요건으로, 요구 분석 단계에서는 제품군의 요구사항과 각 제품의 요구사항을 입력으로 받아 가변성 분석을 수행한다. 그 결과, 가변적 요구사항에 대한 분석을 포함한, 플랫폼 요구사항을 PLDT[6]로 표현한다. 다음으로 아키텍처 설계 단계에서의 추적성 단계는 설계, 요구사항 세분화, 추적성 구축의 3개의 세부 단계로 구성된다. 먼저 1차 수준의 아키텍처 설계를 수행한 후, 해당 설계에서 도출한 설계 컴포넌트를 대상으로 가변성 분석을 수행하여 플랫폼 아키텍처를 PLDT로 표현한다(Step 1). 다음으로 각 컴포넌트에 맵핑할 수 있는 수준으로 플랫폼 요구사항을 세분화한다(Step 2). 이러한 단계를 거쳐 플랫폼 요구사항과 설계 컴포넌트가 서로 맵핑 가능한 수준으로 맞출 수 있다. 마지막 단계로 요구사항과 설계 컴포넌트 사이에 맵핑을 수행한 후에, 다시 가변성 부분에 대한 가변점 적용과 제약사항 추가를 수행하여, 플랫폼 수준의 요구사항과 아키텍처 간 추적성을 PLDT로 표현한다(Step 3).

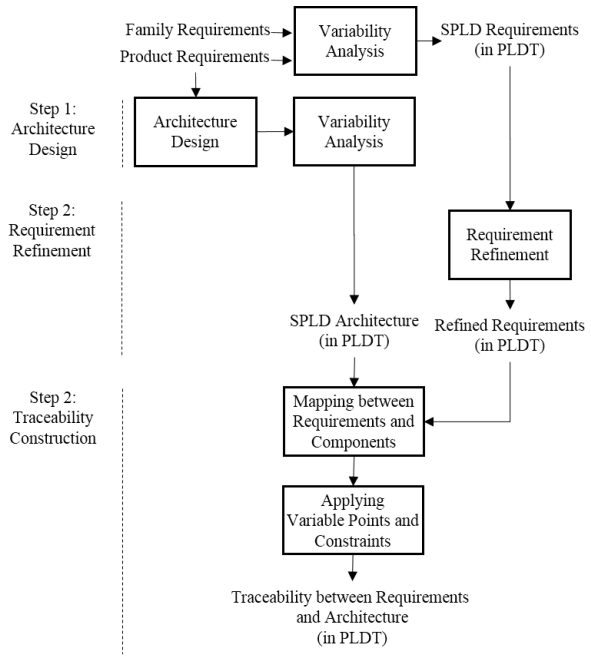

Fig. 3. The Steps for Constructing the Traceability Between Requirements and Software Architecture Design

아키텍처 설계 단계에서의 추적성은 아키텍처 각 수준의 계층적 설계의 반복에 따라 추적성을 구축한다. 따라서 Fig. 3에서 추적성 구축을 위한 단계인 단계 3(Step 3)을 재귀적으로 반복한다. 이렇게 요구사항과 아키텍처 간의 플랫폼 추적성을 재귀적으로 모델링하는 상세 절차는 Fig. 4와 같다.

Fig. 4의 플랫폼 요구사항과 아키텍처 설계 추적성 구축 절차(Construct_PFTR)는 두 개의 입력을 갖는다. 이 절차의 시작점에서의 입력은 COMP는 하나의 컴포넌트로 표현되는 시스템이고, REQCOMP는 시스템에서 만족시켜야 하는 요구사항이다.

해당 절차는 요구사항과 아키텍처 간의 플랫폼 추적성 모델링 프로세스를 재귀적으로 수행한다. 재귀적으로 수행하는 절차의 입력은 분할될 수 있는 하나의 컴포넌트 COMP와 COMP에 할당된 요구사항 REQCOMP이다. 컴포넌트가 분할될 때는 해당 컴포넌트의 책임(responsibility)이 분할되고 상세화되어 하위 컴포넌트에 할당된다.

procedure *Construct_PFTR*
Input: (1) REQ_{COMP}: SPLD Requirements
(2) COMP: Design Components
Output: Platform Traceability $\mathcal{TR}_{Req\text{-}Design,Platform}$
Begin procedure
Phase 1) Decompose COMP into sub-components, fulfilling the given requirements
Phase 2) Given a set of given sub-components DCOMP, refine REQ_{COMP} by using the following procedure:
For each dComp ∈ DCOMP
$rReq_{dComp}$ = the requirements of dComp
Describe PLDTs for dComp and $rReq_{dComp}$
Phase 3) *Construct_TR*(PLDT(PL, $rReq_{dComp}$), PLDT(PL, dComp))
For each dComp ∈ DCOMP
If dComp is decomposable **then**
Construct_PFTR($rReq_{dComp}$, dCmp)
End if
End for
End procedure

subprocedure *Construct_TR*
Input: (1) PLDT(PL, $rReq_{dComp}$): Refined platform requirements which will be traced
(2) PLDT(PL, dComp): Decomposed platform design components which will be traced
Output: $\mathcal{TR}_{rReqdComp\text{-}dComp,Platform}$: Platform Traceability between platform requirements and design components
Begin subprocedure
Create mappings between PLDT(PL, $rReq_{dComp}$) and PLDT(PL, dComp).
If VPs(PL) is a set of VPs of PL and *R*(vp) is range of vp,
For each vp ∈ VPs(PL
Introduce vp to each one's variants
Abstract the relevant mappings for PLDT($rReq_{dComp}$×dComp)
Add constraints to PLDT($rReq_{dComp}$× dComp) by checking the cases in Table 1
End for
$\mathcal{TR}_{rReqdComp\text{-}dComp,Platform}$ = PLDT($rReq_{dComp}$× dComp)
End subprocedure

Fig. 4. The Procedure for Constructing the Platform Traceability Between Requirements and Software Architecture

Fig. 4의 절차는 다음과 같다. 단계 1(Phase 1)에서 상위 컴포넌트를 구체화된 하위 컴포넌트로 분할한다. 단계 2(Phase 2)에서 상위 컴포넌트에 할당된 요구사항을 분할된 세부 컴포넌트의 분할 컨텍스트에 따르도록 세부 요구사항으로 상세화(refine)한다. 단계 3(Phase 3)에서 상세화된 요구사항과 분할된 컴포넌트 사이의 플랫폼 추적성을 구축한다. 상기 세 단계를 분할 가능한 컴포넌트에서 재귀적으로 반복한다.

하위 절차 Construct_TR에서는 입력된 플랫폼 요구사항과 플랫폼 아키텍처로부터 각각의 요구사항에 대한 아키텍처와의 맵핑을 구성한다. 이후에 플랫폼 요구사항, 플랫폼 아키텍처에 포함되어있는 모든 가변점을 각각의 대응되는 가변값에 적용시켜가면서 플랫폼 추적성을 구성하고, 필요한 제약조건을 추가하여 기술한다.

Fig. 4의 하위절차(subprocedure Construct_TR)에서는 플랫폼 요구사항과 플랫폼 아키텍처의 플랫폼 추적성을 모델링한다. 하위절차의 입력값은 PLDT[6]로 각각 기술된 플랫폼 요구사항과 플랫폼 아키텍처다. 플랫폼 아키텍처는 컴포넌트 COMP가 분할되어 생성된 하위 컴포넌트 dComp들로 구성된다. 플랫폼 요구사항은 REQ_{COMP}로부터 상세화된 요구사항인 $rReq_{dComp}$들로 구성된다. dComp에 연결된 $rReq_{dComp}$는 dComp에서 만족시키는 요구사항을 나타낸다.

4. 사례 연구

이 절에서는 3절에서 제안한 플랫폼 추적성 구축 절차의 사례연구를 수행한다. 4.1절에서는 본 논문에서 사용할 사례를 소개한다. 4.2절에서는 요구사항과 아키텍처 간의 플랫폼 추적성 생성 절차의 입력인 사례연구의 플랫폼 요구사항을 제시한다. 4.3절에서는 요구사항과 아키텍처 간 플랫폼 추적성 생성 절차를 적용하여 사례연구에 대한 플랫폼 추적성을 생성한다.

Table 2. Requirements for a Family of Microwave Ovens

FR01	Microwave oven shows the cooking time
FR02	Input buttons with numeric keypad for selecting Cooking Time, Start, and Cancel
FR03	A door sensor senses when the door is open
FR04	A beeper to indicate when cooking is finished
FR05	A light is switched on when the door is open and when food is being cooked
FR06	A turntable turns during cooking
FR07	One-level heating element provides over 300 degree
FR08	Multi-level heating element provides low, medium, and high heating degrees
FR09	User can adjust power level
FR10	A weight sensor detects if there is an object in the oven
FR11	A analog weight sensor detects the weight of the food
FR12	The microwave oven displays messages to the user in English
FR13	The microwave oven displays messages to the user in French
FR14	The microwave oven displays messages to the user in Spanish
FR15	The microwave oven provides recipes for cooking the food.

4.1 사례연구의 소개

H. Gomaa는 [4]에서 UML 기반의 SPLD 아키텍처 설계의 사례연구를 제시하였다. 전자레인지 제품 라인(Microwave Oven Product Line)의 사례연구는 SPLD에서의 각 개발 단계의 산출물을 제공한다. 우리는 Gomaa가 제공한 사례연구를, 전자레인지를 개발하는 회사의 사례연구를 통해 일부 수정하였다. 해당 사례에서 제시하는 산출물을 바탕으로 본 논문에서 제시하는 요구사항과 아키텍처 사이의 추적성 모델링을 표현하는 사례연구를 진행하였다.

Table 3. Planned Products

Product	Requirement	Description
P1	FR01, FR02, FR03, FR05, FR07, FR10, FR12	The basic requirements for Microwave
P2	FR01, FR02, FR03, FR04, FR05, FR08, FR09, FR11, FR12, FR13	The high-tech requirements for Microwave which uses English and French
P3	FR01, FR02, FR03, FR04, FR05, FR06, FR08, FR09, FR11, FR14, FR15	The Automatic requirements for Microware which uses Spain

전자레인지 제품 라인 요구사항은 Table 2와 같이, FR01부터 FR15까지로 기술할 수 있다. 예를 들어 FR01은 전자레인지가 조리 시간을 보여주어야 한다는 요구사항을 기술한다.

전자레인지 제품은 Table 3과 같이 P1, P2, P3의 세 개의 제품을 만들 계획이라고 가정한다.

4.2 플랫폼 요구사항을 위한 가변성 분석

Table 2의 전자레인지 제품 라인의 요구사항은 가변성이 분석되지 않은 상태의 요구사항들이다. Table 3은 계획된 제품들의 요구사항들이다. Table 2와 Table 3의 요구사항에 대한 가변성 분석을 수행한다.

가변성 분석은 전자레인지 제품 라인의 모든 제품에서 공통적으로 만족해야 하는 공통 요구사항과 개별 제품에 따라 달라지는 요구사항을 나눈다. Table 2에 대한 가변성 분석 결과는 Fig. 5와 같이 피처모델[6]의 형식으로 나타낼 수 있다.

Fig. 5로 표현된 플랫폼 가변성 분석 결과의 요구사항은 Fig. 6과 같이 PLDT[6]를 이용하여 기술할 수 있다. Fig. 6에서 계획된 제품(Planned Product)은 Table 3에서 제시된 세 개의 제품의 요구사항과 요구사항 간의 제약조건을 나타낸다. 또한 Fig. 6에서 Platform Design에서 Platform은 전자레인지 제품 라인의 플랫폼을 정의하고 있고, VP Declarations에서 플랫폼의 VP를 정의한다. 그리고 Product Bindings에서 플랫폼에서 세 개의 제품으로 실체화할 수 있도록 하는 바인딩 정의를 제공한다. 바인딩은 Fig. 5에서 표현된 가변점에 대하여 각 제품마다의 가변값을 정의한다.

예를 들어, Fig. 6에서 제품 P1은 FR11을 포함하지 않는다. 즉, 음식을 넣었을 때, 음식이 있음을 탐지하지만(FR10),

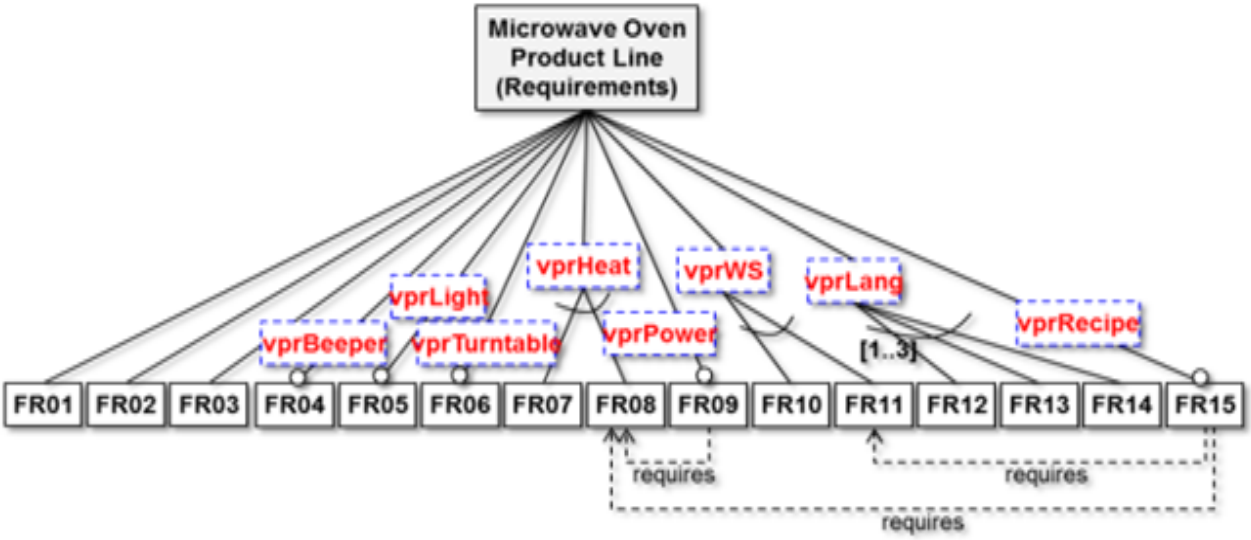

Fig. 5. Requirements Variability for the Family of Microwave Ovens

PL Artifact	Microwave Oven Product Line (Requirements)		
Planned Product	P1		FR01, FR02, FR03, FR05, FR07, FR10, FR12
	P2		FR01, FR02, FR03, FR04, FR05, FR06, FR08, FR09, FR10, FR12, FR13
	P3		FR01, FR02, FR03, FR04, FR05, FR06, FR08, FR09, FR11, FR14, FR15
	Constraints		FR09 requires FR08; FR15 requires FR11; FR15 requires FR08
Platform Design	Platform		FR01, FR02, FR03, vprBeeper, vprLight, vprTurntable, vprHeat, vprPower, vprWS, vprLang, vprRecipe
	VP Declarations		vprBeeper: FR04, null
			vprLight: FR05, null
			vprTurntable: FR06, null
			vprHeat: FR07, FR08
			vprPower: FR09, null
			vprWS: FR10, FR11
			vprLang: v ⊆ {FR12, FR13, FR14}, 1≤|v|≤3
			vprRecipe: FR15, null
	Constraints		(FR09 at vprPower) req (FR09 at vprHeat), (FR15 at vprRecipe) req (FR11 at vprWS), (FR15 at vprRecipe) req (FR08 at vprWS)
	Product Bindings	P1	vprBeeper\null, vprLight\FR05, vprTurntable\null, vprHeat\FR07, vprPower\null, vprWS\FR10, vprLang\FR12, vprRecipe\null
		P2	vprBeeper\FR04, vprLight\FR05, vprTurntable\FR06, vprHeat\FR08, vprPower\FR09, vprWS\FR10, vprLang\{FR12, FR13}, vprRecipe\null
		P3	vprBeeper\FR04, vprLight\FR05, vprTurntable\FR06, vprHeat\FR08, vprPower\FR09, vprWS\FR11, vprLang\FR14, vprRecipe\FR15

Fig. 6. The Platform Requirements for the Family of Microwave Ovens, in PLDT

무게를 재지 않는다(FR11). 이에 따라 플랫폼에서는 무게를 재는 부분을 가변점 vprWS로 정의하고, 이에 들어갈 수 있는 값을 FR10, FR11과 null로 선언한다. P1의 제품으로 실체화할 수 있는 바인딩 정의에서 vprWS에 들어가는 값을 FR10으로 지정함으로써 P1에서 FR10을 포함하고, FR11을 포함하지 않음을 표현한다.

4.3 요구사항과 아키텍처 간 플랫폼 추적성 모델링

4.2절에서 도출한 플랫폼 요구사항에 대하여 플랫폼 아키텍처 간의 추적성의 구축 절차를 아키텍처 설계의 단계별로 적용한다. 하위 절에서 전자레인지 제품 라인의 요구사항과 아키텍처 간의 플랫폼 추적성을 구축하는 단계를 설명한다.

1) 단계 1: 시스템 분할 및 컴포넌트 설계

단계 1은 시스템을 분할한다. Fig. 7은 시스템의 첫 번째 분할 과정을 거쳐 도출된 전자레인지 제품 라인 아키텍처를 보여준다. Fig. 7에서 가변성 유형을 나타내는 스테레오 타입이 표현된 컴포넌트는 가변점 컴포넌트이다. 가변점 컴포넌트는 가변점의 실체화 과정을 통해 가변값의 컴포넌트로 대체된다.

예를 들어, Fig. 7에서 무게를 측정하기 위하여 구현하는 컴포넌트는 WeightCMP이다. 해당 컴포넌트는 서로 다른 컴포넌트로 대체할 수 있기 때문에 가변점 컴포넌트이며, variant로 표기된다.

Table 4는 Fig. 7에서 시스템이 분할되어 도출된 컴포넌트들을 목록으로 보여준다. Table 4에서는 Fig. 7에서 WeightCMP로 표기한 컴포넌트를 BooleanWeightCMP (DS03)과 Analog-WeightCMP(DS04)로 구분하여 나열하였다.

Table 4의 컴포넌트들을 대상으로 가변성 분석을 수행한다. 그 결과인 Fig. 8은 분할된 컴포넌트들의 플랫폼 아키텍처에 대한 PLDT 표현이다.

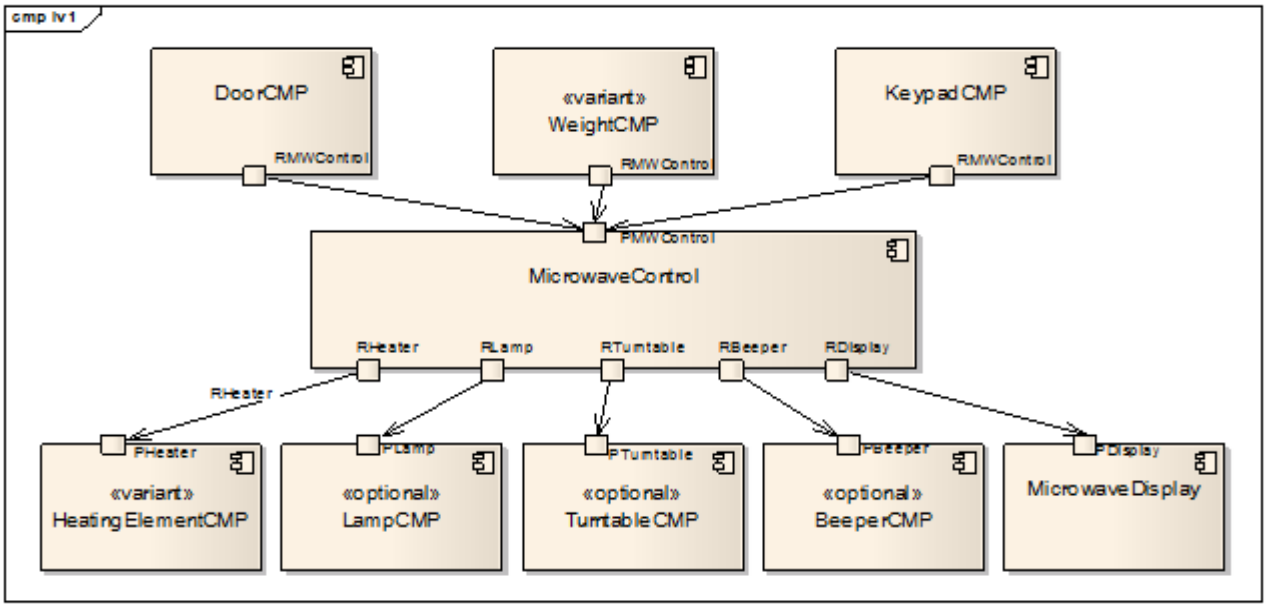

Fig. 7. The Top-Level Product Line Architecture for the Family of Microwave Ovens

PL Artifact	Microwave Oven Product Line (architecture)		
Planned Products	P1		DS01, DS02, DS03, DS05, DS06, DS08, DS11
	P2		DS01, DS02, DS03, DS05, DS07, DS08, DS09, DS10, DS11
	P3		DS01, DS02, DS04, DS05, DS07, DS08, DS09, DS10, DS11
	Constraints		
Platform Design	Platform		DS01, DS02, vpWeight, DS05, vpHeater, vpLamp, vpTurntable, vpBeeper, DS11
	VP declarations		vpWeight: DS03, DS04
			vpHeater: DS06, DS07
			vpLamp: DS08, null
			vpTurntable: DS09, null
			vpBeeper: DS10, null
	Variability Constraints		
	Product Bindings	P1	vpWeight\DS03, vpHeater\DS06, vpLamp\DS08, vpTurntable\null, vpBeeper\null
		P2	vpWeight\DS03, vpHeater\DS07, vpLamp\DS08, vpTurntable\DS09, vpBeeper\DS10
		P3	vpWeight\DS04, vpHeater\DS07, vpLamp\DS08, vpTurntable\DS09, vpBeeper\DS10

Fig. 8. The Platform Architecture for the Family of Microwave Ovens

Table 4. The Design Components of the Top-Level Software Architecture for the Family of Microwave Ovens

COMP	Decomposed Components (dComp)	
Microwave Oven Product Line	DS01	DoorCMP
	DS02	KeypadCMP
	DS03	BooleanWeightCMP
	DS04	AnalogWeightCMP
	DS05	MicrowaveControlCMP
	DS06	One-levelHeatingElementCMP
	DS07	Multi-levelHeatingElementCMP
	DS08	LampCMP
	DS09	TurntableCMP
	DS10	BeeperCMP
	DS11	MicrowaveDisplayCMP

Fig. 8에서 제품 P1은 DS03을 포함하고, DS04를 포함하지 않는다. 즉, 음식을 넣었을 때 감지는 하지만, 무게를 측정하지 않는다. 플랫폼에서는 무게를 감지 및 측정하는 컴포넌트를 가변점 vpWeight로 정의하고, 이에 들어갈 수 있는 값을 DS03, DS04와 null로 선언한다. P1의 제품으로 실체화할 수 있는 바인딩 정의에서 vpWeight에 들어가는 값을 DS03으로 지정함으로써 P1에서 DS03을 포함하고, DS04를 포함하지 않음을 표현한다.

2) 단계 2: 요구사항 상세화

단계 1 아키텍처 설계 과정에서 컴포넌트는 여러 개의 하위 컴포넌트로 분할된다. 상위 수준의 컴포넌트가 하위 수준의 컴포넌트로 분할되면서 상위 컴포넌트에 할당된 요구사항은 상세화되어 하위 컴포넌트에 할당된다. Fig. 9는 이와 같은 상세화 과정의 결과로서 Fig. 5에서 도출된 요구사

PL Artifact	Microwave Oven Product Line (Requirements)		
Planned Products	P1		FR01.01, FR01.02, FR10.01, FR10.02, FR12.00
	P2		FR01.01, FR01.02, FR05.01, FR05.02, FR10.01, FR10.02, FR12.00, FR13.00
	P3		FR01.01, FR01.02, FR05.01, FR05.02, FR11.01, FR11.02, FR14.00, FR15.01, FR15.02
	Constraints		FR05.01 req FR05.02; FR10.01 req FR10.02; FR11.01 req FR11.02; FR15.01 req 15.02; FR15.01 req FR11.01
Platform Design	Platform		FR01.01, FR01.02, vprLight1, vprLight2, vprWS1, vprWS2, vprLang, vprRecipe1, vprRecipe2
	VP declarations		vprLight1: FR05.01, null
			vprLight2: FR05.02, null
			vprWS1: FR10.01, FR11.01
			vprWS2: FR10.02, FR11.02
			vprLang: v ⊆ {FR12.00, FR13.00, FR14.00}, 1≤|v|≤3
			vprRecipe1: FR15.01, null
			vprRecipe2: FR15.02, null
	Variability Constraints		(FR05.01 at vprLight1) req (FR05.02 at vprLight2); (FR10.01 at vprWS1) req (FR10.02 at vprWS2); (FR11.01 at vprWS1) req (FR11.02 vprWS3); (FR15.01 at vprRecipe1) req (15.02 at vprRecipe2); (FR15.01 at vprRecipe1) req (FR11.01 at vprWS1)
	Product Bindings	P1	vprLight1\null, vprLight2\null, vprWS1\FR10.01, vprWS2\FR10.02, vprLang\{FR12.00}, vprRecipe1\null, vprRecipe2\null
		P2	vprLight1\FR05.01, vprLight2\FR05.02, vprWS1\FR10.01, vprWS2\FR10.02, vprLang\{FR12.00, 13.00}, vprRecipe1\null, vprRecipe2\null
		P3	vprLight1\FR05.01, vprLight2\FR05.02, vprWS1\FR11.01, vprWS2\FR11.02, vprLang\{FR14.00}, vprRecipe1\FR15.01, vprRecipe2\FR15.02

Fig. 9. Refined Platform Requirements for the Family of Microwave Ovens

항을 나타낸다. 즉, Fig. 9는 상세화된 플랫폼 요구사항의 PLDT 표현이다.

예를 들어, Fig. 6에서 음식을 탐지하는 요구사항인 FR10은 Fig. 9에서 음식의 존재를 알리는 Boolean weight sensor의 결과를 탐지하는 요구사항(FR10.01)과 음식이 존재함의 결과를 기록하는 요구사항(FR10.02)으로 세분화할 수 있다.

3) 단계 3: 플랫폼 추적성 구축

이 단계에서는 단계 2에서 도출한 플랫폼 요구사항과 플랫폼 아키텍처로부터 플랫폼 아키텍처 사이의 플랫폼 추적성을 도출한다.

a) 요구사항과 아키텍처 간의 맵핑

플랫폼 추적성을 생성하기 위해 우선적으로 모든 가변값의 요구사항과 아키텍처 컴포넌트 사이의 맵핑이 필요하다. 둘 사이의 관계는 아키텍처 컴포넌트가 분할되는 과정에서 컴포넌트의 책임이 주어지고, 그에 따라 요구사항이 분할되는 과정에서 도출된다.

Fig. 10은 전자레인지 제품 라인 아키텍처의 최상위 수준(Level 1)의 컴포넌트들과 그들의 상세화된 요구사항 사이의 관계를 표현한다.

예를 들어, Boolean weight sensor의 값을 탐지하고 저장하는 BooleanWeightCMP(DS03)은 요구사항 FR10.01과 맵핑 관계를 가진다.

b) 가변점 적용 및 제약조건 추가

모든 요구사항과 아키텍처 컴포넌트 사이의 맵핑이 이루어진 후에는 가변값들을 가변점으로 추상화한다. 가변값을

P1	Req_Binding	vprLight1\null, vprLight2\null, vprWS1\FR10.01, vprWS2\FR10.02, vprLang\FR12.00, vprRecipe1\null, vprRecipe2\null
	Design_Binding	vpWeight\DS03, vpLamp\DS08
	Constraints	exclude: FR11.01→DS03, FR10.01→DS04
P2	Req_Binding	vprLight1\FR05.01, vprLight2\FR05.02, vprWS1\FR10.01, vprWS2\FR10.02, vprLang\{FR12.00, FR13.00}, vprRecipe1\null, vprRecipe2\null
	Design_Binding	vpWeight\DS03, vpLamp\DS08
	Constraints	exclude: FR11.01→DS03, FR10.01→DS04
P3	Req_Binding	vprLight1\FR05.01, vprLight2\FR05.02, vprWS1\11.01, vprWS2\FR11.02, vprLang\FR14.00, vprRecipe1\FR15.01, vprRecipe2\FR15.02
	Design_Binding	vpWeight\DS04, vpLamp\DS08
	Constraints	exclude: FR10.01→DS04, FR11.01→DS03

Fig. 11. The Mappings Between Platform Requirements and Software Architecture for the Family of Microwave Ovens

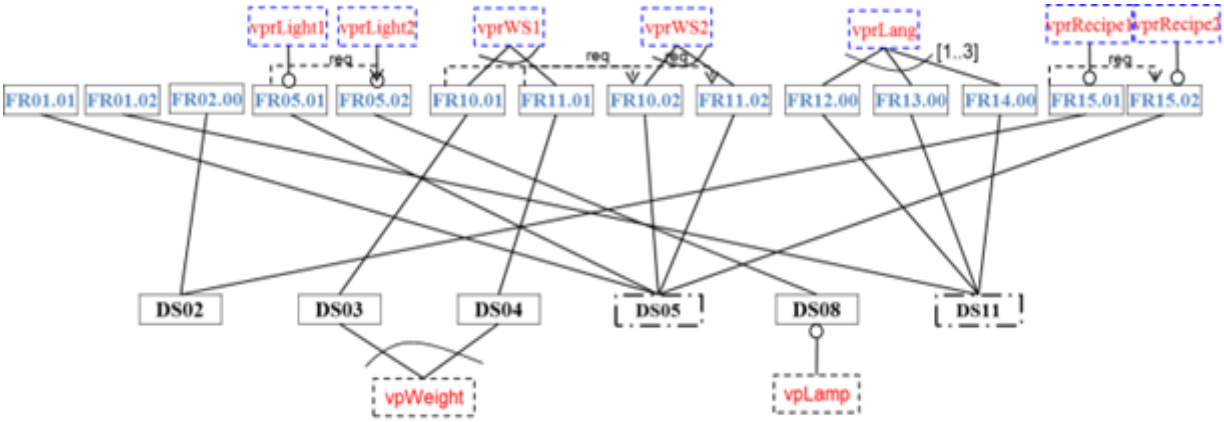

Fig. 10. The Mappings Between Platform Requirements and Software Architecture for the Family of Microwave Ovens

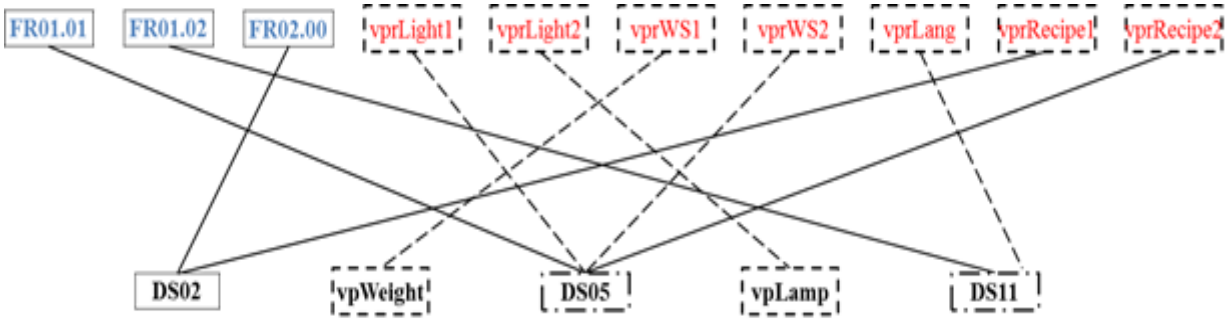

Fig. 12. Feature Model Representation for the Result of Applying VPs to the Mappings in Fig 10

Product line name		PL(rReq × dComp)	
Imported Source and Target		PL(requirements), PL(dComp)	
Platform Design	Platform	{FR01.01, vprLight1, vprWS2, vprRecipe2} → DS05, {FR01.02, vprLang} → DS11, vprLight2 → vprLamp, vprWS1 → vpWeight, {FR02.00, vprRecipe1} → DS02	
	VP declarations	/* vprLight, vprWS, vprLang, vprRecipe are all inherited from PL(requirements) and PL(architecture). No VPs are newly added. */	
	Variability Constraints	P1	exclude: FR11.01→DS03, FR10.01→DS04
		P2	exclude: FR11.01→DS03, FR10.01→DS04
		P3	exclude: FR10.01→DS04, FR11.01→DS03

Fig. 13. The Traceability Between Platform Requirements and Architecture for the Family of Microwave Ovens

가변점으로 추상화하게 되면 맵핑 또한 추상화되게 되고, 그에 따라 Fig. 4에서 존재하지 않았던 맵핑이 포함될 수 있다. 따라서 관련되지 않은 맵핑을 제외시키기 위한 제약조건을 추가할 필요가 있다.

Fig. 9에서 각 제품별로 가변값들을 가변점으로 추상화하고, 그에 따른 제약조건을 추가한 결과는 Fig. 11과 같다. Fig. 10의 가변점을 적용한 결과는 Fig. 12이다.

예를 들어, Fig. 7과 8의 가변점 정의에 따르면, Fig. 10의 피쳐 모델의 표현에서 FR 10.01과 FR 11.01은 vprWS1로 추상화되고 DS03과 DS04는 vpWeight로 추상화되었다. Fig. 11에 있어, 제품 P1에서는 요구사항의 바인딩 정보와 아키텍처에서의 바인딩 정보를 기술한 후에, 관련되지 않는 맵핑을 제외시키기 위한 제약 사항을 명시한다. 예를 들어 vprWS1→vpWeight의 맵핑이 플랫폼 추적성에 포함될 경우, "exclude: FR11.01→DS03, FR10.01→DS04"은 vprWS1 vpWeight의 인스턴스로 얻어지는 네 개의 맵핑들 중에 FR11.01→DS03과 FR10.01→DS04이 vprWS1→vpWeight이 나타내는 맵핑들로부터 제외됨을 표현한다.

c) 플랫폼 추적성의 PLDT 기술

Fig. 11과 12의 플랫폼 추적성을 PFML을 이용한 PLDT로 기술한 결과는 Fig. 13과 같다. 이러한 분석을 통해서 우

리는 요구사항 FR10.01을 포함시킬 경우 컴포넌트 DS04를 제외해야 하며, 요구사항 FR11.01을 포함시킬 경우, 컴포넌트 DS03을 제외해야 한다는 제약 조건을 도출할 수 있었다.

지금까지는 시스템을 분할한 최상위 수준의 아키텍처 컴포넌트와 이 컴포넌트에 할당된 요구사항 사이에 플랫폼 추적성을 도출하였다. 상세한 아키텍처 설계를 위하여 추적성 구축 절차는 재귀적으로 두 번 더 반복되어야 한다.

5. 관련 연구

관련연구는 (1) 추적성 프레임워크에 대한 연구, (2) SPLD에서의 가변성 추적에 관한 연구, (3) 피처와 요구사항간의 플랫폼 추적성에 대한 연구로 나눌 수 있다.

첫째, 추적성 프레임워크에 대한 연구는 소프트웨어 개발에 있어서 생성된 다양한 산출물 사이의 추적성을 관리하기 위해, 사전에 추적성 대상을 결정하는 프레임워크를 제시하는 연구이다. 이 중 SPLD에서의 추적성 프레임워크 연구는 다음과 같다. Bayer 등은 [2]에서 SPLD에서의 개발 공정에서의 산출물을 뷰(view) 기반으로 정의하고, 뷰 사이의 관계를 정의한 메타모델을 제시하였다. 그러나 Bayer의 연구는 메타모델을 통한 추적성의 이용방법의 가능성에 대해서 언급하고 있을 뿐, 구체적인 모델링 방법을 제안하지 못하였다. Kim 등은 [8]에서 SPLD 산출물의 메타모델을 정의하고, 각 산출물 사이의 추적 링크와 맵핑 관계를 정하여 기술하는 추적성 지도를 제안했다. 추적성 지도에서는 11개의 추적 링크를 정의하고 각 추적 링크에서는 세부적인 맵핑 유형과 룰을 정의하여 각 산출물 사이의 관계를 정의한다. 이와 같이 정의된 결과로써 SPLD 프로세스를 자동화할 수 있는 기반을 제공한다. 하지만 제품군 수준에서의 추적성 관리보다는 제품군의 각 제품의 개발과정상의 추적성 관리에 집중한다. 즉, 플랫폼 추적성을 모델링하는 방법은 제공하지 않으며, 도메인 산출물과 제품 산출물 사이의 추적성의 관계를 구체적으로 표현하지 못한다. Anquetil 등은 [1, 12]에서 소프트웨어 제품 라인을 위한 모델 기반의 추적성 체계(ATF: AMPLE Traceability Framework)를 제안하였다. ATF는 산출물과 산출물 사이의 추적성을 표현하는 하이퍼링크를 정의하는 추적성 메타모델을 기반으로 하여 이 메타모델을 인스턴스화(instantiation)함으로써 다양한 개발 프로세스에 적합하도록 만들 수 있다. 또한 SPLD을 위해 필요한 추적성을 네 가지로 분류한다. 그 네 가지 분류는 상세화(refinement), 유사성(similarity), 시간(time), 그리고 가변성(variability)이다. 그러나 AFT에는 도메인 산출물과 응용 산출물 사이의 관계에 대한 정의는 없다. 즉, 응용 산출물이 어떤 도메인 산출물로부터 파생되었는지에 대한 추적성 관계를 설명하지 않는다. Ramesh 등은 [11]에서 두 개의 추적성 이용 그룹을 식별하여 각 그룹에서 이용되는 추적성의 참조모델을 제시하였으며, 그 모델에서 나타나는 추적성 링크를 유형별로 구분하여 정의하였다. 이는 개별 제품 개발 과정에서의 추적성만을 대상으로 하고 있다.

둘째, SPLD에서의 가변성 추적에 관한 연구들이 있다. 소프트웨어 공학은 전통적으로 단일 시스템 개발에서 수평적, 수직적 추적성을 고려하였다[4, 13]. 수평적 추적성은 같은 수준의 추상화에 있는 산출물 간의 링크이고, 수직적 추적성은 서로 다른 수준의 추상화를 가지는 산출물 사이의 링크이다. SPLD에서는 그 외에도 가변성 관련 추적성이 추가적으로 필요하다. Pohl 등은 [10]에서 두 가지 유형의 가변성 추적성을 구별하였다. 하나는 가변성 모델과 서로 다른 개발 공정에서의 산출 간의 관계이다. 이는 가변점을 포함하는 도메인 공학의 개발 공정 사이의 추적성이다. Pohl 등은 [10]에서 Orthogonal Variability Model(OVM)을 이용한 도메인 공학 산출물 사이의 가변성 추적을 보여준다. OVM과 각 개발 공정의 산출물 사이에 추적성 링크를 연결하여 모든 산출물의 가변성을 직교적으로 OVM을 통해 표현하고, 각 개발 단계에서의 가변성의 추적을 가능하게 한다. K. Berg 등은 [3]에서 피처를 중심으로 도메인 공학 산출물 사이의 가변성 추적에 대한 연구를 하였다. 피처와 요구사항, 피처와 아키텍처 요소, 피처와 구현 아이템과 같이 피처를 중심으로 각 개발 공정의 산물들을 추적성 링크로 연결하여 가변성을 관리하였다. 그러나 [3]의 피처 모델과 [10]의 OVM 모두 산출물에서의 가변성에 대한 추적성만을 제공하고, 각 산출물 간의 관계에 대한 추적성은 지원하지 못한다.

셋째, 피처와 요구사항 간의 플랫폼 추적성에 대한 모델링 연구가 있다. 논문 [5], [7]은 SPLD에서 피처와 요구사항 간의 플랫폼 추적성에 대한 정형화된 모델링 방법을 제안하였다. 제품군 내의 각각의 제품들의 피처와 요구사항간의 맵핑을 기반으로 제품군의 공통성, 가변성을 도출하고, 이로부터 제품군의 피처와 요구사항간의 플랫폼 추적성을 모델링 하며, 정형화된 모델링을 기반으로 자동화된 추적성 구축 절차와 도구를 제시한다.

6. 결 론

본 논문은 플랫폼 수준의 요구사항과 아키텍처 간의 추적성에 대해 아키텍처의 계층적 모델에 따라 재귀적으로 추적, 명시하는 모델링 절차를 제안하였다. 또한 본 논문에서는 플랫폼 추적성의 모델링 절차를 전자레인지 제품 라인 사례연구에 적용하여 제안 절차와 표현 방법으로 추적성을 표현할 수 있음을 보여주었다.

SPLD의 추적성 모델링에 관한 연구는 가변점을 명시적으로 고려하는 체계적인 방법을 논문 [5][7]에서 제시하였으나, 이 논문들은 피처와 요구사항 간의 플랫폼 추적성만을

다루고 있다. 요구사항과 아키텍처 간의 플랫폼 추적성은 아키텍처가 계층적인 구조를 가질 수 있는 특징으로 인하여, 과거에 제시된 방법이 그대로 적용될 수 없다. 본 논문은, 이를 해결하는 재귀적인 방법을 제시함으로써 체계적인 플랫폼 추적성 구축의 범위를 피처와 요구사항을 넘어 아키텍처까지 확장하였다.

향후 연구로, 우리는 추적성 구축의 연구 범위를 구현과 시험의 단계를 모두 포함할 수 있도록 확장할 예정이며, 본 논문에서 제안하는 플랫폼 추적성의 형식적 관계가 모든 공정에 대하여 적용될 수 있음을 보이는 연구 결과를 제시할 것이다. 또한 이러한 추적성 구축을 자동화하기 위한 도구를 제안하고 개발할 것이다. 이러한 연구는 단기적으로는 임베디드 시스템과 같이 소프트웨어 중심적인 시스템(software-intensive systems)의 제품 라인 개발에 있어서 점증적 개발(incremental development)이나 유지보수/진화에 들어가는 개발비용을 크게 감소시킬 것이고, 장기적으로는 본 연구가 제시하는 추적성 구축의 확장을 통하여 다양한 사용자 서비스(즉, 서비스 제품군)를 제공하기 위하여 공통플랫폼을 사용하는 시스템의 유지보수/진화의 비용 감소에도 기여하게 될 것이다.

References

[1] N. Anquetil, U. Kulesza, R. Mitschke, A. Moreira, J. C. Royer, A. Rummler, and A. Sousa. 2010. "A model-driven traceability framework for software product lines," *Software & Systems Modeling*, 9(4) (27, June): 427-451. doi:10.1007/s10270-009-0120-9.

[2] J. Bayer and T. Widen, "Introducing traceability to product lines," *Product Family Engineering (PFE 2002)*, Bilbao, Spain, pp.127-147, 2002.

[3] K. Berg and D. Muthig, "Critical analysis of using feature models for variability management," Technical Report. University of Pretoria, May, 2005. [Internet], http://polelo.cs.up.ac.za/personal-pages/kathrin/index.

[4] H. Gomaa, "Designing software product lines with UML: from use cases to pattern-based software architectures," Addison Wesley, 2004.

[5] S. Kang, "A Formal Approach to Modeling Traceability between Platform-level Features and Requirements in Software Product Line Engineering," KAIST MS Thesis, Computer Science, Feb., 2011.

[6] 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, pp.143-168, 1998.

[7] S. Kang, J. Kim, S. Kang, and S. Eom, "A Formal Representation of Platform Feature-to-Requirement Traceability for Software Product Line Development," *The 38th International Computer Software and Applications Conference*, (COMPSAC 2014), Sweden 21-25 July, 2014.

[8] S. D. Kim, S. H. Chang and H. J. La, "Traceability Map: Foundations to Automate for Product Line Engineering," *SERA*, pp.340-347, 2005.

[9] A. J. Lattanze, "Architecting Software Intensive Systems: A Practitioner's Guide," Auerbach Publications, 2009.

[10] K. Pohl, G. Bockle, and F. V. D. Linden, "Software product line engineering: Foundations, Principles, and Techniques," Springer, 2005.

[11] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Transactions on Software Engineering*, Vol.27, pp.58-93, 2001.

[12] A. Sousa, Kulesza, U., Rummler, A., Anquetil, N., Moreira, R. M. A., Amaral, V., and Araújo, J. A., "A model-driven traceability framework to software product line development," *ECMDA Traceability Workshop (ECMDA-TW) 2008 Proceedings,* Sintef, Trondheim, pp.97-109. 2008.

엄 석 환

e-mail : esh21c@mobis.co.kr
2012년 KAIST 전산학(석사)
2012년~현 재 현대모비스 연구원
관심분야 : 소프트웨어 아키텍처, 소프트웨어 제품 라인

강 성 원

e-mail : sungwon.kang@kaist.ac.kr
1982년 서울대학교 사회과학대학(학사)
1989년 Univ. of Iowa 전산학(석사)
1992년 Univ. of Iowa 전산학(박사)
1993년~2001년 한국통신 연구개발본부 선임연구원
1995년~1996년 미국 국립표준기술연구소(NIST) 객원연구원
2001년~2005년 한국정보통신대학교 조교수
2005년~2009년 한국정보통신대학교 부교수
2009년~현 재 KAIST 전산학부 부교수
관심분야 : 소프트웨어 아키텍처, 소프트웨어 제품 라인, 소프트웨어 시험

김 진 규

email : jinlooks@kaist.ac.kr
2013년 KAIST 전산학(박사)
2013년~현 재 국방과학연구소 선임연구원
관심분야 : 소프트웨어 아키텍처, 시스템 운용성 분석, 소프트웨어 제품라인, 시스템 상호운용성

이 선 아

e-mail : saleese@kaist.ac.kr
1997년 이화여자대학교 전산학(학사)
1999년 이화여자대학교 전산학(석사)
2005년 카네기멜론대학교 소프트웨어 공학(석사)
2013년 KAIST 전산학(박사)
1999년~2006년 삼성전자 선임/책임연구원
2013년~2015년 KAIST 박사후연구원
2015년~현 재 KAIST 전산학부 연구조교수
관심분야 : 소프트웨어 아키텍처, 소프트웨어 리파지터리 마이닝, 추천 시스템, 소프트웨어 진화