석사학위논문

FORM-UML : UML 기반의 제품 라인 아키텍처 가변성 모델링 기법 및 지원 도구의 개발

2013년

한성대학교 대학원
정보시스템공학과
정보시스템공학전공
이 지 원

석사학위논문
지도교수 이관우

FORM-UML : UML 기반의 제품 라인 아키텍처 가변성 모델링 기법 및 지원 도구의 개발

FORM-UML : Development of Variability Modeling Technique and its Supporting Tool for UML-based Product Line Architecture

2013년 6월 일

한성대학교 대학원
정보시스템공학과
정보시스템공학전공
이 지 원

석 사 학 위 논 문
지도교수 이관우

FORM-UML : UML 기반의 제품 라인 아키텍처 가변성 모델링 기법 및 지원 도구의 개발

FORM-UML : Development of Variability Modeling Technique and its Supporting Tool for UML-based Product Line Architecture

위 논문을 공학 석사학위 논문으로 제출함

2013년 6월 일

한성대학교 대학원

정보시스템공학과

정보시스템공학전공

이 지 원

이지원의 공학 석사학위논문을 인준함

2013년 6월 일

심사위원장 _____ 인

심 사 위 원 _____ 인

심 사 위 원 _____ 인

국 문 초 록

FORM-UML : UML 기반의 제품 라인 아키텍처 가변성 모델링 기법 및 지원 도구의 개발

한성대학교 대학원
정보시스템공학과
정보시스템공학전공
이 지 원

일반적으로 프로덕트 라인 아키텍처는 프로덕트 라인 공학의 성공을 위해서 매우 중요한 요소로 인지되고 있다. 프로덕트 라인 공학의 대표적인 방법론인 FORM(Feature Oriented Reuse Method)은 개념, 프로세스, 배치, 모듈 관점에서 프로덕트 라인 아키텍처를 정의한다. 하지만 FORM의 아키텍처 모델은 자체 고유한 표기법으로 정의되어, 다른 방법론의 모델과 호환성이 떨어질 뿐만 아니라, 가변성 표현도 아키텍처 수준에서 명시적으로 표현되지 못하는 문제가 있다.

본 논문에서는 산업계의 표준으로 자리 잡고 있는 모델링 언어인 UML(Unified Modeling Language)을 이용하여 FORM의 아키텍처 모델을 정의하고 아키텍처 수준에서 가변성을 명시적으로 표현하기 위한 방법인 FORM-UML을 제안한다.

FORM-UML은 UML의 확장 메커니즘인 태그 값(Tagged Value)과 스테레오타입(Stereotype)을 이용하여 아키텍처 가변성을 표현한다. 특히 각 관점의 아키텍처 모델이 가지는 가변요소에 대한 표현뿐만 아니라, 서로 다른 관점의 아키텍처 모델 간의 대응 관계에서 존재하는 가변성 표현 기법도 제안한다. 뿐만 아니라, 제안된 가변성 표현 기법으로 모델링된 프로덕트 라인 아키텍처 모델로부터 제품 아키텍처를 추출하기 위한 자동화 도구를 개발하였다. 마지막으로 제안된 모델링 기법과 지원도구의 유효성을 평가하기 위해 바다 부표

시스템의 사례연구를 수행한다.

【주요어】 소프트웨어 프로덕트 라인 공학, Feature Oriented Reuse Method, UML, 아키텍처 모델, 가변성 모델링, 도구 개발

목 차

【 표 목 차 】

【 그 림 목 차 】

제 1 장 서 론

새로운 소프트웨어를 개발하기 위해서는 많은 시간과 인력이 소요된다. 이 때 요구되어 지는 것은 양질의 소프트웨어를 빠르게 생산해내는 것이다. 그러나 품질이 좋은 소프트웨어를 개발하기 위해서는 고려해야할 사항이 많기 때문에 시장적시성이 떨어진다. 그렇기 때문에 등장하게 된 것이 소프트웨어 프로덕트 라인 공학이다. 소프트웨어 프로덕트 라인 공학은 품질과 생산성을 모두 높이기 위해 등장한 재사용 메커니즘을 제공한다. 소프트웨어 제품을 개별적으로 개발하는 것이 아니라, 유사한 특징을 지닌 제품들을 함께 개발하도록 만든다.

유사한 기능을 갖는 시스템의 집합을 도메인이라고 한다. 도메인 내 속하는 시스템들은 서로 공통성(Commonality)과 가변성(Variability)을 가진다. 도메인 내에서 재사용이 가능한 플랫폼을 정의하고 구축하여 핵심 자산을 만든다. 핵심 자산 내 특정 마켓의 특별한 요구를 충족시키는 휘처들의 집합을 공유하는 소프트웨어 제품들의 집합을 특별히 소프트웨어 프로덕트 라인이라고 한다. 소프트웨어 프로덕트 라인의 자산을 조합하여 여러 제품 개발에 재사용이 가능하다.

그렇기 때문에 소프트웨어 프로덕트 라인 공학의 핵심은 다양한 제품 개발에 재사용이 가능한 핵심 자산의 개발에 있다. 특히 프로덕트 라인 공학을 통해 생성되는 산출물 중에서 프로덕트 라인 아키텍처는 가장 중요한 산출물 중에 하나이다. 프로덕트 라인 아키텍처는 동일한 제품군에 속하는 공통적인 컴포넌트와 다양한 제품들의 차이점이 관리되어야한다. 동일한 도메인에 속한 핵심 자산을 개발하는 소프트웨어 프로덕트 라인 공학 방법론 중 Kang(1998)의 FORM(Feature Oriented Reuse Method)은 전 세계적으로 가장 많이 사용되는 방법론으로써 휘처를 중심으로 가변성을 표현하고 관리하는 방법을 제시한다. 휘처란 다양한 이해 관계자들이 식별 가능한 두드러지고 구별되는 특징을 의미한다.

FORM 아키텍처 모델은 FORM 방법론의 가장 중요한 산출물 중의 하나이다. 그러나 FORM의 아키텍처 모델은 사실상 표준으로 이용이 되는

UML(Unified Modeling Language)을 사용하지 않기 때문에, 자체적으로 도구를 개발할 뿐만 아니라, 표준으로 널리 사용되는 UML모델과 호환이 이루어지지 않는 단점이 있다. 따라서 본 논문에서는 프로덕트 라인 아키텍처를 UML로 모델링하는 방법을 제안하고, 이를 FORM-UML이라 명명한다.

한편, FORM 아키텍처 모델에서는 아키텍처 레벨에서 가변성이 직접적으로 표현되지 않는데, FORM-UML에서는 아키텍처 레벨의 가변성이 표현되고 관리되도록 UML의 확장 메커니즘인 태그 값(Tagged Value)과 스테레오 타입(Stereotype)을 이용하는 방법을 제안한다.

마지막으로 FORM-UML의 산출물의 가변성에 대한 표현을 기반으로 프로덕트 아키텍처 모델로부터 프로덕트 모델로의 원활한 추출이 필요하게 된다. 따라서 가변성이 표현된 프로덕트 라인 아키텍처 모델로부터 프로덕트 아키텍처 모델로의 추출을 도와주는 도구를 오픈소스인 StarUML로 개발하였다.

논문의 구성은 다음과 같다. 2장에서는 소프트웨어 프로덕트 라인 공학에 대한 소개와 대표적인 프로덕트 라인 공학 방법론인 FORM에 대해서 설명한다. 그리고 기존의 UML 기반의 프로덕트 라인 모델링에 대해서 설명한다. 3장에서는 FORM-UML 아키텍처 모델링 기법을 설명하고 아키텍처 모델의 가변 요소를 정의하고 이를 표현하는 가변성 모델링에 대해서 제안한다. 그리고 프로덕트 라인 아키텍처 모델에서 프로덕트 아키텍처 모델을 생성하는 방법에 대해 설명한다. 4장에서는 3장에서 제안한 방법들을 사례에 적용시켜 이를 검증한다. 그리고 5장에서는 본 논문의 결론과 향후 과제를 설명하고 마무리 짓는다.

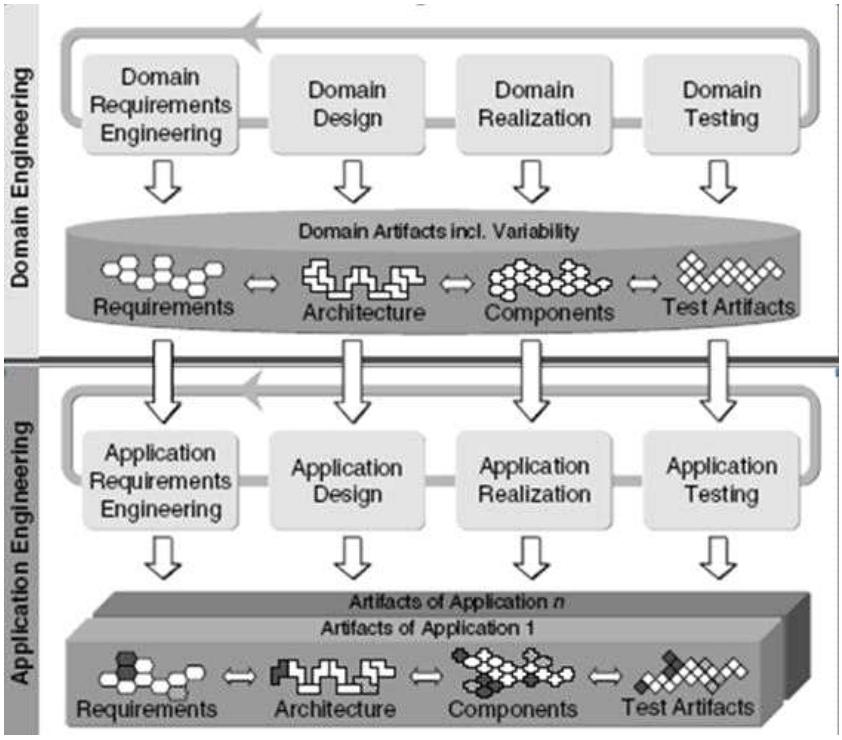
제 2 장 이론적 배경

제 1 절 소프트웨어 프로덕트 라인 공학

소프트웨어 프로덕트 라인은 유사한 특징을 지닌 여러 소프트웨어 제품들의 집합을 의미한다. 소프트웨어 프로덕트 라인 공학은 소프트웨어 제품 개발에 재사용이 될 수 있는 핵심 자산(아키텍처 혹은 구현 모듈)을 미리 개발하고 이 자산을 조합하여 원하는 시스템을 적시에 생산해내는 재사용 패러다임이다. (Atkinson, 2002)

〈그림 2-1〉에서와 같이 소프트웨어 프로덕트 라인 공학은 크게 도메인 공학(Domain Engineering)과 응용 공학(Application Engineering)으로 나누어진다. 도메인 공학은 특정 도메인에 속한 제품들의 공통점과 차이점을 분석하여 프로덕트 라인 자산(Product-Line Asset)을 만드는 것이다. 도메인 공학에는 광범위한 의미의 설계 결정(Design Decision)이 이루어지는 아키텍처 설계(Architecture Design)방법과 좁은 범위의 설계 결정이 이루어지는 컴포넌트 설계(Component Design)방법이 있다. 응용 공학은 도메인 공학에서 만들어진 프로덕트 라인 자산을 이용하여 특정한 제품을 생산하는 활동이다. (Pohl, 2005)

이러한 소프트웨어 프로덕트 라인 공학 방법론에는 FORM(Feature Oriented Reuse Method), Pohl(2005)의 OVM(Orthogonal Variability Model), Gomaa(2004)의 PLUS(Product Line UML-Based Software Engineering), Weiss(1999)의 FAST(Family Oriented Abstraction, Specification, Transformation)등이 있다.



〈그림 2-1〉 소프트웨어 프로덕트 라인 공학 프로세스 (Pohl, 2005).

특히 프로덕트 라인 아키텍처는 프로덕트 라인 공학에서 매우 중요한 산출물 중에 하나이다. 아키텍처를 기술하는 방법에는 Kruchten(1995), Hofmeister(2000), Clements(2002)의 방법 등이 있다. 이와 같이 다양한 여러 가지 아키텍처 모델링 언어로 아키텍처를 표현해왔지만, 이들 아키텍처 기술 언어는 유사한 개념을 상이한 표기법으로 정의한다. 때문에 서로 다른 아키텍처 모델링 언어로 표기된 모델 간의 호환성 문제가 발생한다.(이관우, 2011)

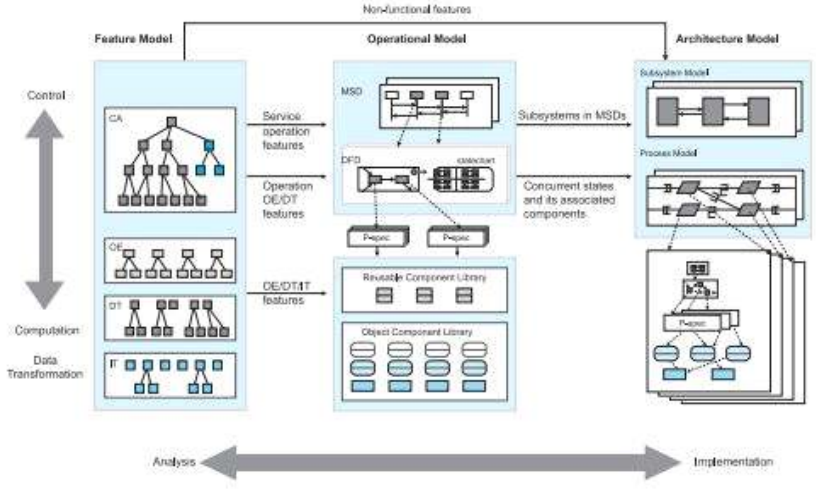
따라서 본 논문의 제 3장 제 1절에서는 사실상 표준으로 자리 잡고 있는 UML(Unified Modeling Language)을 이용하여 FORM의 프로덕트 라인 아키텍처를 모델링하는 방법에 대해 구체적으로 설명한다.

제 2 절 Feature Oriented Reuse Method

1. FORM 방법론 개요

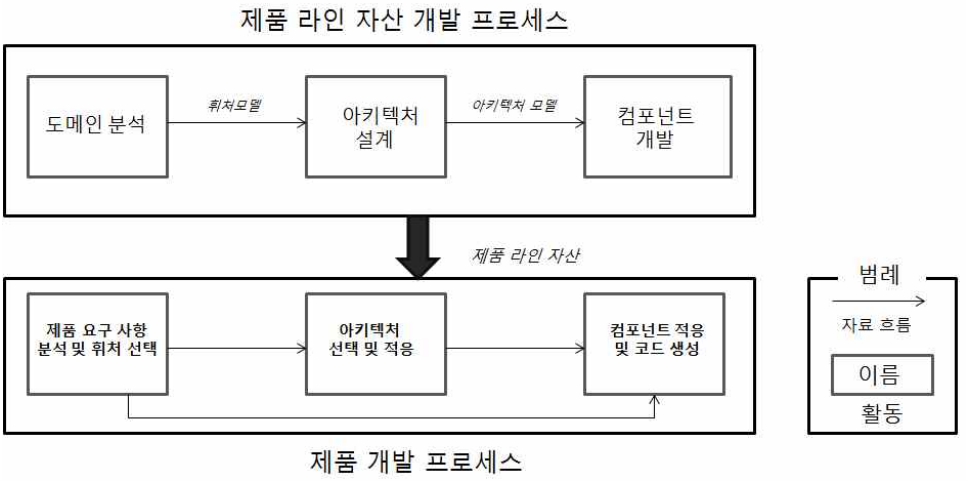
FORM(Feature-Oriented Reuse Method)은 대표적인 소프트웨어 프로덕트 라인 공학 방법론이다. 시스템 내의 특성을 지칭하는 휘처를 통해 공통성과 가변성을 분석하여 휘처 모델(Feature Model)을 작성하고, 이를 바탕으로 핵심 자산(아키텍처 및 컴포넌트)을 구축한다.

FORM 방법론의 기본 원리에는 휘처 지향(Feature Orientation), 객체 지향(Object Orientation), 관심 분리(Seperation of Concerns)가 있다. 휘처란 이해관계자가 식별할 수 있는 시스템의 구분되는 특성을 의미한다. 휘처의 성격은 도메인 내의 공통적인 특성은 필수(Mandatory) 휘처로, 시스템 내의 구별되는 차이점을 가진 특성은 선택(Optional) 휘처, 택일(Altenative) 휘처, 다중 선택(Or) 휘처로 구분한다. 또한, 휘처 모델은 이해관계자의 관심에따라 능력 계층(Capability Layer), 운영 환경(Operating Environment), 도메인 기술(Domain Technology), 구현 기술(Implementation Technology)의 4개 계층의 휘처로 분류하고 이들 간의 관계를 설정한다.



〈그림 2-2〉 FORM 모델 사이의 대응 관계.

이러한 휘처 모델은 분석 단계에서 사용되는 모델로 사용되는 것뿐만 아니라, 설계 및 구현 단계 등 소프트웨어 생명주기 전반에 영향을 미치는 모델이다. 〈그림 2-2〉[1]에서 볼 수 있듯이 휘처 모델의 운영 환경 휘처는 운영 모델을 개발하는 데 사용되고, 운영 환경, 도메인 기술, 구현 기술 휘처는 컴포넌트 모델을 개발하는 데 사용된다.



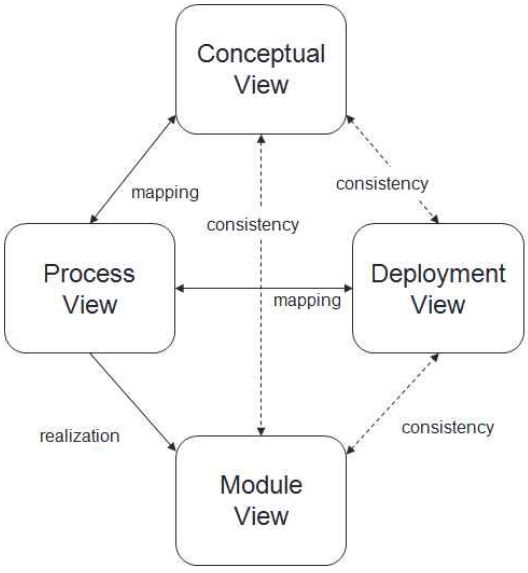
〈그림 2-3〉 소프트웨어 제품라인 방법론 적용 프로세스 전체 구성도 (강교철, 2004).

FORM 방법론을 통해 프로덕트를 개발하는 단계는 휘처를 단위로 생성한 프로덕트 라인에 속한 특정 프로덕트를 만들어내는 것이다. FORM 방법론 기반의 소프트웨어 개발에 대한 전반적인 프로세스에 대한 구성은 〈그림 2-3〉과 같다. 먼저, 도메인 분석 단계에서 도메인에 속한 소프트웨어 프로덕트들의 요구사항을 분석하고, 이들 간의 공통점과 차이점을 분석하여 휘처 모델을 생성한다. 아키텍처 설계는 도메인 분석 단계에서 산출된 휘처 모델을 토대로 시스템의 밑그림을 그리는 단계이다. 컴포넌트 개발 단계에서는 휘처 모델을 기반으로 객체 컴포넌트를 개발하고 이를 이용하여 재사용이 가능한 컴포넌트 패키지를 개발한다.

1) 정보통신산업진흥원,『소프트웨어 재사용을 기반으로 한 소프트웨어 제품라인 공학기술 적용 가이드』, 정보통신산업진흥원, 2009, p.12.

2. FORM 아키텍처 모델

FORM 개발 단계 중 도메인 분석과 아키텍처 설계 과정을 통해 나오는 산출물로는 앞서 설명한 휘처 모델과 아키텍처 모델이 있다. 도메인 분석과정의 결과로 나온 휘처 모델을 기반으로 시스템의 밑그림을 그리는 과정이 아키텍처 설계 과정이다. FORM에서는 개념, 프로세스, 배치, 모듈의 네가지 관점에서 아키텍처를 기술하도록 권고한다.



〈그림 2-4〉 FORM의 다중 관점 뷰의 관계(이관우, 2003).

개념 뷰(Conceptual View)는 시스템을 기능적인 측면에서 논리적으로 표현한다. 개념 아키텍처는 시스템 혹은 시스템들의 논리적인 기능 단위를 개념 컴포넌트(Conceptual Component)로 나타내고 이들 사이의 논리적인 상호작용을 개념 커넥터(Conceptual Connector)로 정의하여 개념 컴포넌트와 개념 커넥터의 연결관계를 통해 아키텍처를 구성한다.

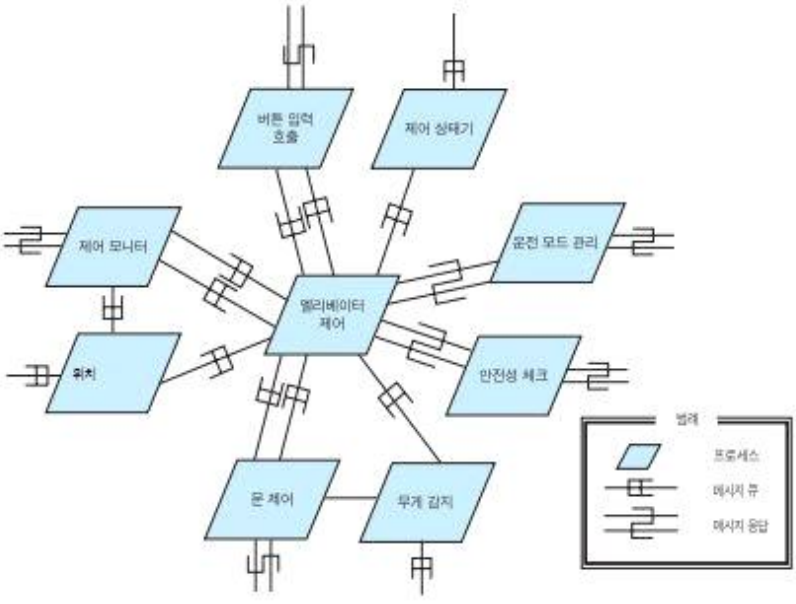
프로세스 뷰(Process View)는 각 시스템들의 동적인 행위를 표현한다. 프로세스 아키텍처(Process Architecture)는 시스템 혹은 시스템들의 동시성 구조

(Concurrency Structure)를 나타내는 것이다.

배치 뷰(Deployment View)는 소프트웨어의 구성 요소가 물리적인 하드웨어 노드에 어떻게 할당되는 지를 표현한 것이다.

모듈 뷰(Module View)는 개념, 프로세스, 배치 뷰에서 표현된 설계 결정이 어떻게 모듈로 대응되는 지에 대해 나타낸다.

그러나 FORM 방법론에서는 자체적으로 사용하는 언어로 아키텍처를 표현함으로 사실상 표준(de-facto)인 UML(Unified Modeling Language)도구를 활용할 수가 없고 자체적으로 도구를 개발해야하는 문제점이 있다.

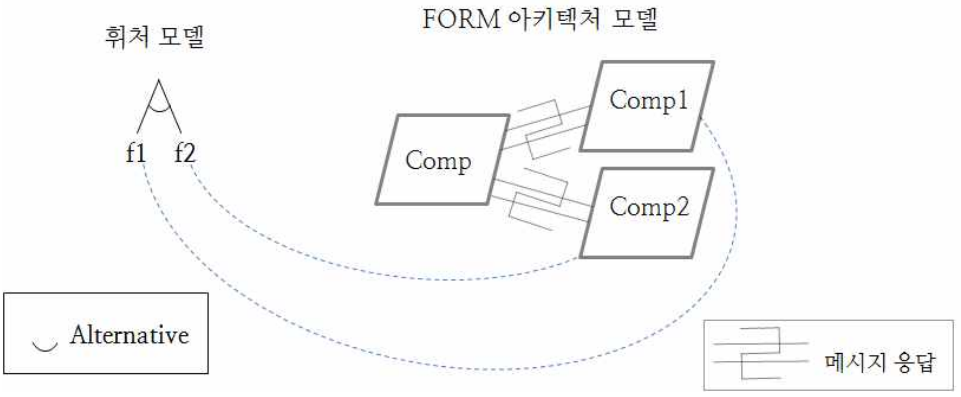


〈그림 2-5〉 FORM 프로세스 아키텍처 모델의 예시.

〈그림 2-5〉[2]는 FORM의 프로세스 아키텍처 모델에 대한 예시이다. 기존의 FORM 모델에서는 고유의 언어로 아키텍처를 표현했기 때문에 FORM 아키텍처 모델을 사용하는 이해관계자들이 FORM의 고유 언어를 사전에 숙지해야하는 번거로움이 있다. 때문에 FORM-UML은 이러한 FORM의 아키텍처

2) 정보통신산업진흥원, 전게서, p.50

모델을 확장하여 아키텍처 모델을 UML을 이용하여 모델링하고 관리할 수 있도록 함으로써 기존의 UML로 만든 자산의 활용이 가능하며, 이용자가 새로운 표기법을 배우는 부담감을 줄일 수 있는 장점을 가진다.



〈그림 2-6〉 FORM 아키텍처 모델에서의 휘처와의 대응 관계.

한편, FORM 아키텍처 모델은 휘처와의 대응으로 가변성을 표현하기 때문에, 아키텍처 레벨에서의 가변성은 표현되지 않는다. 〈그림 2-6〉의 휘처 모델은 휘처 f1과 f2가 서로 대안적인 가변성을 가진다. 휘처 f1은 컴포넌트 Comp1에 대응되고, 휘처 f2는 컴포넌트 Comp2에 대응된다. 따라서 컴포넌트 Comp1과 Comp2는 서로 대안적인 가변성을 가지고 있다는 것을 알 수 있다. 하지만 아키텍처 모델 레벨에서는 컴포넌트 Comp1과 Comp2가 대안적 가변성을 가지고 있다는 것을 알 수 없다. 따라서 FORM-UML에서는 아키텍처 수준에서의 가변성 또한 표현하고 관리한다.

FORM 아키텍처(FORM-Architecture) 모델의 요소와 UML 표현 방법은 제 3장 연구 설계의 제 1절 FORM-UML 모델링에서 설명한다.

제 3 절 UML 기반의 프로덕트 라인 모델링

한편, UML을 이용하여 프로덕트 라인 가변성을 모델링을 하는 연구는 새로운 연구가 아니다.

프로덕트 라인의 효과적인 모델링을 위해서는 프로덕트 라인의 공통성과 가변성을 효과적으로 표현할 수 있는 모델링 언어가 필요하다. 그 중 UML언어는 프로덕트 라인의 공통성을 표현하기에는 적합하지만, 가변성을 표현하는 것에는 한계가 있다. 따라서 UML의 확장 메카니즘을 이용하여 가변성을 표현하는 기존 연구들이 존재한다.

가변성을 표현하는 프로덕트 라인 공학을 위한 언어로는 Kang의 Feature Model, Gomaa의 PLUS, Atkinson(2000)의 KobrA, Pohl의 OVM, Ommering(2000)의 Koala, Hoek(2001)의 Tae등이 있다. Feature Model은 가장 널리 사용되는 대표적인 도메인 언어로써, 휘처 단위의 가변성 표현에는 탁월하지만, 아키텍처 레벨의 가변성 표현을 하지 않고, UML이 아닌 자체적인 언어를 사용한다. 그리고 Koala와 Tae방법론은 프로덕트 라인 아키텍처를 단일 아키텍처 레벨에서 가변점을 표현하여 모델링한다는 사상을 가지고 있지만, 현재 많이 사용되고 있지 않아 비교 대상으로 삼지 않는다. 또한 Pohl의 OVM 역시, 가변성 표현에는 탁월한 언어이지만, 독자적인 가변성 다이어그램을 통해 가변성을 표현하기 때문에 본 논문의 UML을 사용하여 가변성을 표현하는 연구에 대한 관점에서는 비교 대상이 되지 않는다. 이러한 이유로 본 절에서는 Gomaa와 Atkinson의 가변성 모델링 연구를 비교 연구 대상으로 삼았다. 그리고 본 논문에서 FORM-UML을 제안하는 이유를 설명한다.

1. PLUS

Gomaa는 FORM과 같이 도메인 공학, 응용 공학으로 구분되는 PLUS 방법론을 제시하였다. PLUS방법론의 도메인 공학은 3가지의 단계로 이루어진다. 첫 번째 단계는 휘처 모델링과 유스케이스 모델링으로 이루어진 요구사항 모델링의 단계이다. 두 번째 단계는 정적 모델링, 동적 모델링으로 구성된 분석 모델링, 그리고 마지막 단계로 컴포넌트 기반 프로덕트 라인 아키텍처 개발로 이루어진 설계 모델링으로 세분화 된다. (America,2000) PLUS는 각 모델의 산출물을 UML 다이어그램의 표기법을 기반으로 가변성을 표현하는 방법을

제시하였다. 특히, PLUS는 각각의 UML 다이어그램별로 다른 타입의 가변성 표현 언어를 사용하고 UML의 스테레오 타입을 이용하여 가변점과 가변치를 구분한다. 예를 들어, [표 2-1]은 PLUS의 휘처 모델링 수준에서의 휘처 모델의 가변성 표현을 스테레오 타입을 이용하여 표현한 것이다. 반면, [표 2-2]는 유스케이스 모델에서의 가변성을 표현한 것인데, 이는 PLUS에서 모델별로 다른 타입의 가변성 표현 언어를 사용하고 있음을 알 수 있게 해준다.

[표 2-1] PLUS의 휘처 모델링 수준에서의 휘처 모델의 가변성 표현.

	의미	스테레오 타입
Mandatory	필수 휘처	〈〈common feature〉〉
Optional	선택 휘처	〈〈optional feature〉〉
Alternative	대안 휘처	〈〈alternative feature〉〉

[표 2-2] PLUS의 아키텍처 모델링 수준의 유스케이스 모델의 가변성 표현.

	의미	스테레오 타입
Mandatory	필수적 유스케이스	〈〈kernel〉〉
Optional	선택적 유스케이스	〈〈optional〉〉
Alternative	대안적 유스케이스	〈〈alternative〉〉

PLUS는 UML을 이용하여 프로덕트 라인을 설계하는 방법을 제안하였다는 장점이 있는 반면에, 다양한 제품에 쉽게 재사용이 가능하도록 모델링하는 안내지침은 부족한 문제점이 있다.

2. KobrA

KobrA는 UML을 이용한 컴포넌트 기반의 프로덕트 라인 공학 개발 프로세스이다. KobrA에서는 프로덕트 라인 개념을 사용한 컴포넌트 공학을 프레임워크 공학(Framework Engineering)이라고 한다. KobrA 프로세스는 프레임워크 공학과 애플리케이션 공학으로 이루어져있다.

프레임워크 공학에서는 도메인 분석을 통해 프로덕트들의 공통성과 가변성을 식별한다. 프레임워크에 대한 분석이 끝나면 프로덕트들의 요구사항을 분

석하고 비즈니스 모델링, 구조 모델링, 액티비티 모델링, 인터랙션 모델링, 디시젼 모델링의 작업을 수행한다. 그리고 컴포넌트 모델링을 통해 컴포넌트를 구현한다.

애플리케이션 공학에서는 프레임워크 공학 과정을 통해 생산된 프레임워크를 이용하여 하나의 특정 애플리케이션을 개발한다. KobrA는 컴포넌트 및 프레임워크 기반 기술을 최대한 반영하여 프로덕트 생성을 위한 구체적 접근 방식을 제공하고 이를 UML 모델로 변환시켜 최종 산출물로 접근하는 일반적인 관점을 제공한다. (Atkinson, 2002) KobrA의 애플리케이션 공학 프로세스는 배경 구현(Context Realization) 인스턴스화, Komponent(KobrA Component)의 명세화, 현실화, 인스턴스화로 구성된다. 배경 구현에서는 개발할 프로덕트의 특징을 도출하여 프레임워크공학에서 개발된 프레임워크와 요구사항이 얼마나 중복되는지를 확인한다. 그리고 Komponent의 명세화, 현실화, 인스턴스화를 반복하여 수행함으로 명세 Komponent 제약 트리를 생성하고 이를 통해 하나의 애플리케이션을 생성한다. KobrA는 프레임워크와 애플리케이션을 생산해내는 상세한 지침을 제공하고 있으나 실용적으로 이를 활용하는 프로세스를 제시하기 위해서는 좀 더 정제될 필요가 있다.

반면, FORM방법론은 휘처를 기반으로 공통성과 가변성을 관리하여 핵심자산을 개발해내는 안내지침을 제시한다. 그러나 FORM방법론에서는 핵심자산으로 사용하는 다양한 모델들을 UML과 같은 표준화된 모델을 사용하지 않는다. FORM방법론에서는 Gomaa의 실시간 시스템을 위한 설계 방법론인 CODARTS(Gomaa, 1992)의 표기법을 테일러링하여 사용한다. 그렇기 때문에 UML 기반의 모델링 도구를 사용하는 사용자들에게는 FORM방법론의 활용이 제한적이다.

PLUS와 KobrA는 모두 산업계 표준인 UML을 사용한다는 장점을 가진다. 하지만 PLUS 방법론은 UML의 다양한 모델에서의 제품 라인에 대한 가변성을 표현하는 것에 대해서는 자세히 기술하고 있는 반면, FORM 방법론과 같이 재사용이 가능한 핵심자산을 개발해내는 안내지침은 제시하고 있지 않

다. 뿐만아니라 FORM에서는 PLUS에서는 고려하지 않는 다중 관점의 아키텍처를 다룬다. 또한 KobrA는 컴포넌트의 중심의 개발에 치중하고 가변성에 대한 표현에 있어서 표준 UML 표기법과는 다른 본연의 확장 UML을 사용하는 설계를 제안하였기 때문에 KobrA 프로세스를 위한 새로운 확장 UML 표현법을 배워야하는 문제점을 가진다. 때문에 본 논문에서는 FORM방법론에서 제시하는 재사용이 가능한 핵심 자산을 개발해내는 안내 지침을 기반으로 FORM의 핵심 자산의 하나인 아키텍처를 UML을 이용하여 모델링하는 FORM-UML을 제안한다.

[표 2-3] UML 기반 프로덕트 라인 방법론과 FORM 방법론의 비교.

	PLUS	KobrA	FORM
UML기반 모델링 지원	○	○	×
아키텍처 수준의 가변성 표현	○	○	×
다중 관점의 아키텍처 지원	×	×	○

제 3 장 연 구 설 계

제 1 절 FORM-UML 모델링

1. FORM-UML 아키텍처 모델링

1) 개념 아키텍처 모델링

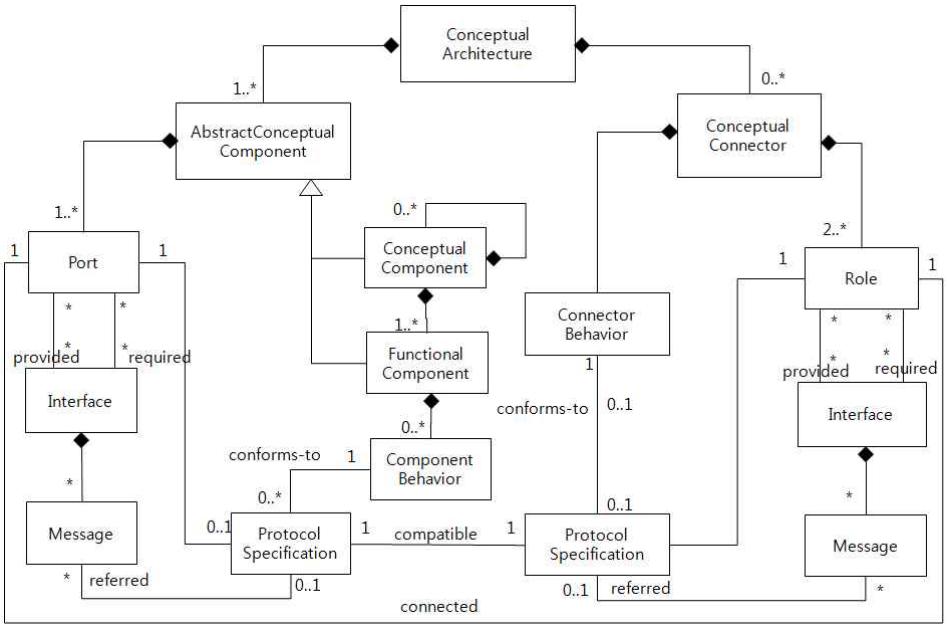
개념 아키텍처는 시스템 혹은 시스템의 논리적 기능 단위를 개념 컴포넌트 집합으로, 개념 컴포넌트들 간의 논리적인 상호 작용을 개념 커넥터의 집합으로 표현하고 개념 컴포넌트와 개념 커넥터의 연결로서 정의된다.

〈그림 3-1〉과 같이, 개념 컴포넌트는 하나이상의 포트(Port)를 통해 다른 컴포넌트와의 메시지를 주고받는다. 각 포트는 하나 이상의 인터페이스(Interface)와 관련이 있다. 요구 인터페이스(Required Interface)는 포트를 통해 나가는 메시지로 정의되고 제공 인터페이스(Provided Interface)는 포트를 통해 들어오는 메시지로 정의된다.

개념 커넥터는 커넥터 행위(Connector Behavior)와 롤(Role)으로 이루어 진다. 커넥터 행위는 컴포넌트의 컴포넌트 행위(Component Behavior)와 대응되고 롤은 포트에 대응 된다.

한편, 개념 컴포넌트는 서브 개념 컴포넌트로의 분해가 가능하다. 분해 과정에서 특별히 말단 컴포넌트를 기능 컴포넌트(Functional Component)라 명명한다. 이 기능 컴포넌트는 컴포넌트의 행위를 정의한다.

프로토콜 명세서(Protocol Specification)는 포트로 들어오고 나가는 메시지들의 순서를 정의하는 제약사항을 의미한다. 프로토콜 명세서는 포트가 속한 개념 컴포넌트의 컴포넌트 행위에 부합되어야 한다. 그러므로 각 개념 커넥터의 롤은 개념 컴포넌트의 포트와 함께 프로토콜 명세서와 관련이 되어있으며, 각각의 프로토콜 명세서가 상호 일치하면 관련된 포트와 롤은 연결



〈그림 3-1〉 개념 아키텍처 메타 모델(이관우, 2011).

(Connection)관계로 이루어진다. 이는 개념 아키텍처의 구성을 의미한다.

개념 컴포넌트와 포트는 UML 2.x의 UMLComponent와 UMLPort로 표현이 가능하다. 그러나 개념 커넥터는 UML 2.x에서 제공되는 커넥터를 이용하여 표현이 가능하지만, 복잡한 상호작용의 표현을 위하여, 커넥터 또한 UMLComponent를 이용하여 모델링한다. 그리고 〈〈c-connector〉〉스테레오타입을 사용하여 컴포넌트와 구분짓는다. [표 3-1]은 개념 아키텍처의 구성요소와 UML 표기법을 정의한 표이다.

2) 프로세스 아키텍처 모델링

프로세스 아키텍처는 시스템 혹은 시스템들의 동시성 구조(Concurrency Structure)를 나타내는 것으로, 프로세스 컴포넌트(Process Component), 프로세스 통신 커넥터(Process Communication Connector), 그리고 이들 간의 연결관계로 구성된다.

[표 3-1]개념 아키텍처의 FORM 구성요소와 UML 표기법(이관우, 2011).

FORM elements	UML elements	UML notations
Conceptual Component	Stereotyped Component	«conceptual» Name «functional» Name
Port (from Conceptual Component)	Port (from Component)	PortName
Interface (from Port)	Interface (from Port)	Provided Interface
ProtocolSpecification (from Port)	Constraint of Ports	{constraint}
Conceptual Connectors	Stereotyped Component	«c-connector» Name
Role (from Conceptual connector)	Port (from Component)	roleName «c-connector» Name roleName

프로세스 컴포넌트는 동시에 실행이 가능한 단위로 구현 시에 개발 플랫폼 혹은 언어에 따라서 쓰레드(Thread)나 태스크(Task)로 구체화된다. 프로세스 컴포넌트는 다음의 세가지 속성을 가진다.

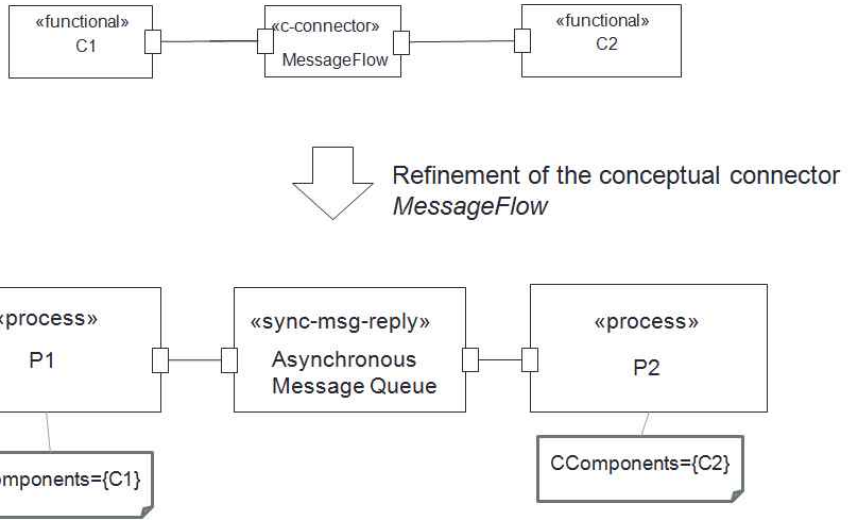
· 할당된 개념 컴포넌트들 : 하나의 프로세스 컴포넌트에는 하나 이상의 개념 컴포넌트로 대응이 가능하다. 대응이 된 개념 컴포넌트들은 하나의 실행 쓰레드를 통해 순차적으로 실행된다. 그러나 개념 컴포넌트란 기능의 논리적인 묶음이므로 반드시 동시성과 일치하지는 않는다. 따라서, 하나의 프로세스 컴포넌트에 할당된 개념 컴포넌트들은 동시성을 갖지 않고 순차적으로 수행됨을 나타낸다.

· 우선 순위 : 하나 이상의 프로세스 컴포넌트가 공유된 자원에 접근하기 위해서는 프로세스 컴포넌트들 간의 우선순위 설정이 필요하다.

· 주기 (선택 속성) : 주기적으로 수행되는 프로세스 컴포넌트는 주기를 속성으로 정의한다.

프로세스 통신 커넥터는 프로세스 컴포넌트 간의 메시지 상호작용을 나타내는 것이다. 이는 메시지 통신, 공유 데이터 접근, 이벤트 동기화의 세가지로

분류된다. 메시지 통신은 동기 메시지 통신과 비동기 메시지 통신으로 분류되고 각각은 응답, 무응답 통신으로 또다시 분류된다.



〈그림 3-2〉 프로세스 컴포넌트의 UML 표기법.

프로세스 아키텍처에서 프로세스 컴포넌트와 통신 커넥터는 UMLComponent를 이용하여 표현한다. 다만, 개념 아키텍처 요소와의 구분을 위하여 프로세스 컴포넌트는 〈〈process〉〉, 프로세스 통신 커넥터는 통신 커넥터의 방식을 나타내는 스테레오타입으로 구분한다. 〈표 3-2〉는 프로세스 통신 커넥터의 형식에 따른 스테레오타입 표기법이다.

[표 3-2] 프로세스 통신 커넥터의 스테레오타입 표기법.

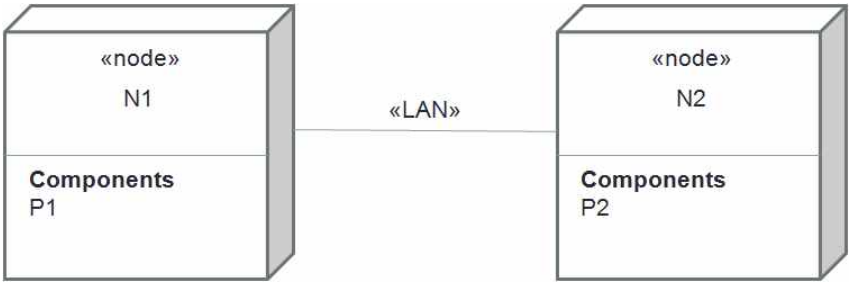
Subtype-Hierarchy			stereotypes
Message Communication	Synchronous Message Communication	Synchronous Message with Reply	«sync-msg-reply»
		Synchronous Message without Reply	«sync-msg»
	Asynchronous Message Communication	Asynchronous Message Queue	«async-msg-Q»
		Asynchronous Message with Callback	«async-msg-CB»
Shared Data Access			«shared-data»
Event Synchronization			«event-sync»
Generic process communication connector			«pc-connector»

3) 배치 아키텍처 모델링

배치 아키텍처는 소프트웨어가 물리적인 하드웨어 노드에 어떻게 매핑될 것인 지를 나타내는 것이다. 배치 아키텍처는 노드 컴포넌트, 소프트웨어 엘리컨트, 링크 커넥터와 이들 간의 연결 관계로 이루어진다.

노드 컴포넌트는 소프트웨어 엘리먼트가 배치될 물리적인 객체이다. 이는 프로세서, 디바이스 등을 의미한다. 소프트웨어 엘리먼트는 프로세스 아키텍처 요소 및 개념 아키텍처 요소를 의미한다. 그리고 링크 커넥터는 노드 컴포넌트들 간의 물리적인 연결을 의미한다.

배치 아키텍처는 UML의 배치 다이어그램(Deploymenet Diagram)을 통해 표현이 가능하다.



〈그림 3-3〉 배치 아키텍처의 UML 표기법.

4) 모듈 아키텍처 모델링

모듈 아키텍처는 모듈과 이들간의 연결관계로 정의되는 데, 이는 개념, 프로세스, 배치 아키텍처 모델에서 내려진 설계 결정이 어떻게 구현 단위인 모듈로 대응 될 수 있는 지를 나타낸다.

모듈 아키텍처는 UML의 클래스 다이어그램을 이용하여 직접적으로 표현 가능하다.

한편, 프로덕트 라인 아키텍처는 단위 시스템을 위한 아키텍처가 아니라 모

든 시스템의 개발에 재사용이 되는 아키텍처이다. 때문에 프로덕트 라인 범위 내 시스템 간의 차이점이 아키텍처 요소 상에서 표현이 되고 관리되어야한다. 다음 절에서는 각 관점의 아키텍처가 가지는 가변요소를 주지시키고 가변성 모델링 기법을 제안한다.

제 2 절 FORM-UML 가변성 모델링

아키텍처 모델은 각 관점(개념, 프로세스, 배치, 모듈)마다 아키텍처 모델 요소는 상이하지만, 가변 요소를 표현하는 개념은 같다.

1. FORM-UML 아키텍처 가변 요소

프로덕트 라인 아키텍처의 가변성을 표현하기 위한 효과적인 방법은 가변점(Variation Point)과 가변치(Variant)를 이용하는 것이다. 가변점은 아키텍처 요소 중에서 변화 요소가 있는 지점을 의미하고, 가변치는 해당지점에 결합될 수 있는 가변 요소를 의미한다.

· Variation Point(VP) - 가변점
: 가변 요소가 있는 지점을 표현
· Variant - 가변치
: VP에 결합될 수 있는 가변 요소를 표현
· Variability Dependency
: 가변점과 가변치간의 의존 관계를 표현
- Mandatory : Variant가 필수 요소임을 표현
- Optional : 해당 Variant가 제품에 따라 선택되거나 선택되지 않을 수 있는 요소임을 표현
- Alternative : 지정된 숫자만큼의 Variant가 대안적으로 선택되어야하는 요소임을 표현

네 가지 관점의 아키텍처는 각각 가변요소를 가지고 있다.

개념 아키텍처에서는 개념 컴포넌트와 개념 커넥터 자체가 가변성과 각 개념 컴포넌트 내부의 가변성이 있을 수 있다. 개념 컴포넌트 내부의 가변성으로는 인터페이스의 가변성과 기능행위의 가변성으로 세분화 된다.

프로세스 아키텍처는 프로세스 컴포넌트나 프로세스 연결 커넥터 자체의 가변성과 프로세스 컴포넌트에 할당된 개념 컴포넌트의 가변성을 가진다.

배치 아키텍처도 프로세스 아키텍처와 마찬가지로 노드 컴포넌트와 링크 커넥터 자체의 가변성이 있고, 노드 컴포넌트에 할당되는 소프트웨어 엘리먼트의 가변성이 있다.

마지막으로 모듈 아키텍처는 모듈 컴포넌트 자체의 가변성과 모듈 인터페이스의 가변성, 모듈 구현의 가변성이 있다.

[표 3-3]은 각 관점의 아키텍처 가변 요소를 정리한 표이다.

[표 3-3] 아키텍처 모델에 따른 가변 요소.

모델	**가변 요소**
개념아키텍처 모델	서브시스템
	개념 컴포넌트
	포트
	오퍼레이션
	인터페이스
프로세스아키텍처 모델	프로세스 컴포넌트
	프로세스 커넥터
	할당된 개념 컴포넌트
배치아키텍처 모델	패키지
	노드
	배치된 프로세스 컴포넌트
모듈아키텍처 모델	클래스
	애트리뷰트
	오퍼레이션
	관계

프로덕트 라인 아키텍처 모델에서 가변성이 관리되기 위해서는 이와 같은 가변요소가 아키텍처 레벨에서 표현이 되어야 한다. 또한 아키텍처 가변 요소

는 휘처 모델에서의 가변 휘처와 대응된다. 가변 휘처와 대응되는 아키텍처 가변요소의 존재를 결정짓는 것은 가변 휘처의 선택 여부이다. 때문에 아키텍처 가변성을 모델링할 때, 대응되는 가변 휘처 정보를 기입해 줄 필요가 있다. 다음 절에서 이를 설명한다.

2. FORM-UML 아키텍처 가변성 모델링

본 논문에서는 UML의 확장 메커니즘인 스테레오타입(StereoType)과 태그 값(Tagged Value)을 이용하여 가변성을 표현한다. 이 메커니즘을 이용하면 프로덕트 라인 아키텍처의 가변 요소의 가변성과 가변 요소에 대응되는 가변 휘처의 정보를 효과적으로 정의할 수 있다.

본 절에서는 각 관점(개념, 프로세스, 배치, 모듈)의 아키텍처가 가지는 가변 요소에 대한 모델링 기법을 제안한다. 뿐만 아니라 서로 다른 관점의 아키텍처 모델에 대한 대응 관계에 대한 표현 기법에 대해서도 설명한다.

1) 각 관점 아키텍처 모델의 가변성 표현

FORM 아키텍처의 각 모델은 관점 별 가변 요소 자체는 상이하지만, 가변성을 표현하는 개념은 동일하다. 아키텍처 모델의 가변성을 표현하기 위해서는 먼저 아키텍처 모델 내에 존재하는 가변점을 찾아 각 가변점의 가변성 타입을 명시하게 된다.

아키텍처 모델의 가변점은 크게 모델 요소 자체에 가변점이 있는 블랙박스 가변점과 모델 요소 내부에 가변 요소가 있는 화이트박스 가변점으로 나누어진다. 그리고 가변점이 가지는 가변성의 종류는 선택적 혹은 대안적 가변 요소로 구분된다. 선택적 가변 요소는 시스템을 구성할 때, 시스템에 따라 선택될 수 있는 가변성을 나타내고 대안적 가변 요소는 하나의 시스템을 구성할 때, 하나의 모델 요소만이 선택될 수 있는 가변 요소를 의미한다.

본 논문에서는 아키텍처 모델 요소의 가변성과 가변점의 종류에 대한 표현을 가시화하기 위하여 UML의 스테레오타입을 이용한다. 사용자의 인식에 대

한 편의를 도모하기위하여 가변성을 상징하는 도형을 제안한다. 모델 요소의 가변성이 블랙박스임을 표현해주기 위해 가변점의 종류를 표현해주는 도형에 검정색을 채우고 화이트박스 가변성을 가지는 모델 요소의 가변점의 종류를 표현해주는 도형에는 흰 색을 채워준다. 그리고 선택적 가변성은 원형으로, 대안적 가변성은 삼각형으로 표현하였다. 그리고 컴포넌트 내부에 여러 가변 요소가 존재할 경우에는 화이트 박스 가변성으로 사각형을 표현한다. [표 3-4]는 이를 정리한 표이다.

[표 3-4] 아키텍처 모델 요소의 가변성과 가변점의 종류에 대한 표기법.

가변점의 종류 / 가변성	선택적 (Optional)	대안적 (Alternative)	필수 (Mandatory)
Black-box reuse	〈〈●〉〉	〈〈▲〉〉	×
White-box reuse	〈〈○〉〉	〈〈△〉〉	〈〈□〉〉

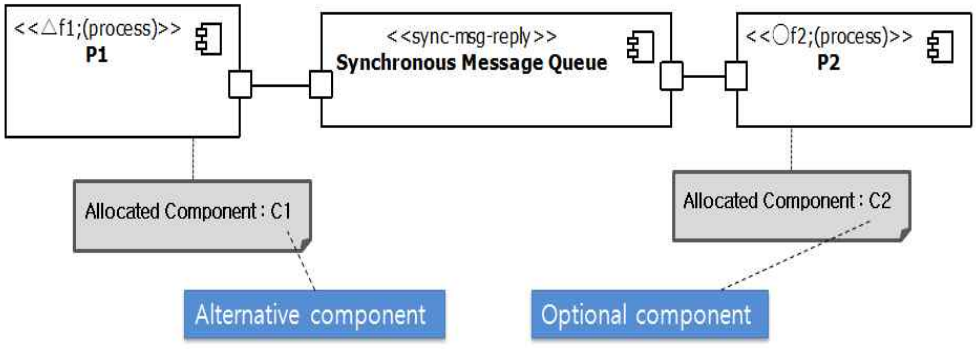
또한, 아키텍처 모델의 가변성을 관리하기 위해서 FORM에서는 휘처 모델을 기반한다. 본 논문에서는 아키텍처 요소들에 대응되는 가변 휘처 정보를 가변 요소에 명시하여 가시화한다.

2) 다중 관점의 아키텍처 모델의 가변성 모델링

〈그림 3-4〉는 앞 절에서 설명한 개념 아키텍처 가변성 모델링이다. 개념 컴포넌트에 대응되는 휘처 정보와 가변점의 종류를 스테레오타입으로 기입하여 가변요소를 표현한다. 개념 컴포넌트 C1은 블랙박스 재사용과 대안적 가변점의 성격을 가지고 휘처 f1과 대응관계에 있는 가변 요소이다. 개념 컴포넌트 C2는 블랙박스 재사용과 선택적 가변점의 성격을 가지고 휘처 f2와 대응관계 있음을 스테레오타입으로 표현한 것이다.

〈그림 3-4〉 개념 컴포넌트의 가변성 표현.

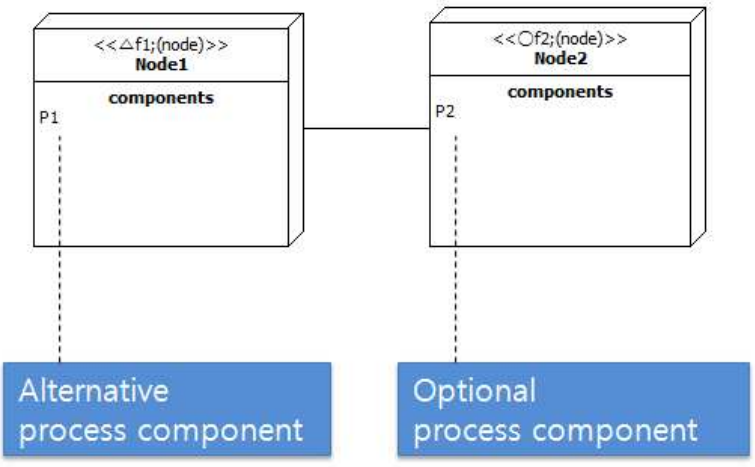
한편, 3장 1절에서 언급했듯이, 프로세스 아키텍처 모델에서 프로세스 컴포넌트에는 할당된 개념 컴포넌트들이 존재한다. 이를 표현해주기 위해 모델링 요소에 태그값(Tagged Value)을 넣어주는 기법을 활용하였다.



〈그림 3-5〉 개념 - 프로세스 컴포넌트 대응 관계 표현.

〈그림 3-5〉는 개념 - 프로세스 컴포넌트간의 대응 관계를 모델링한 것이다. 개념 컴포넌트 C1을 할당하고 있는 프로세스 컴포넌트 P1은 가변 요소 안에 대안적 가변점을 가지고 있으므로 대안적-화이트박스 가변점(〈〈△〉〉)으로 표현한다. 또 프로세스 컴포넌트 P2는 개념 컴포넌트 C2를 할당하고 있는데 이를 태그값으로 표현하고 가변 요소안에 선택적 가변점을 가지고 있으므로 선택적- 화이트 박스 가변점(〈〈○〉〉)으로 표현한다.

배치 아키텍처 모델에서도 프로세스 아키텍처 모델과 마찬가지로 컴포넌트의 대응관계를 표현해 주어야한다.



〈그림 3-6〉 노드 - 프로세스 컴포넌트의 할당 표현.

〈그림 3-6〉은 배치 아키텍처에서 프로세스 컴포넌트 할당을 표현한 것이다. 배치 아키텍처의 역할은 소프트웨어에 가변 요소가 어떻게 할당되는 지를 명확히 보여 주어야하는 것이므로 UML의 DeployedComponents 항목을 사용한다.

Node1은 대안적 프로세스 컴포넌트인 P1을 할당하고 있으므로 대안적-화이트 박스 가변점(〈〈△〉〉)으로 표현한 것이다. Node2는 선택적 프로세스 컴포넌트인 P2를 할당하고 있으므로 선택적-화이트박스 가변점(〈〈○〉〉)으로 표현한다.

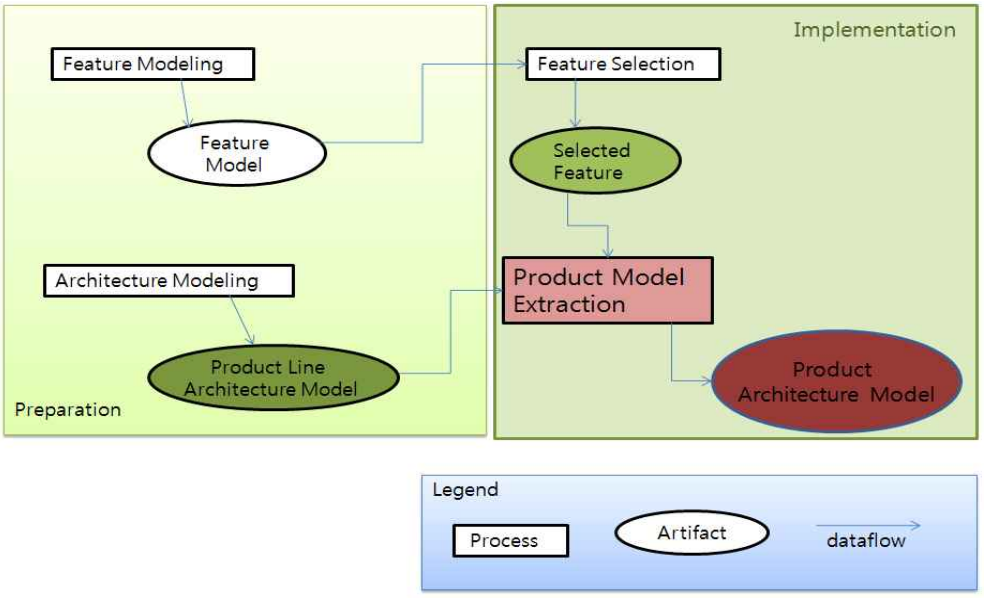
한편, 프로덕트 아키텍처 요소는 각 가변 요소에 대응되어있는 가변 휘처의 선택 여부에 따라 선택되거나 혹은 삭제된다. 다음 절에서는 가변 휘처의 선택을 기반으로 하여 가변성을 표현한 아키텍처 프로덕트 라인 모델에서 아키텍처 프로덕트 모델을 추출하는 기법을 설명한다.

제 3 절 프로덕트 모델 생성 방법

1. FORM-UML 아키텍처 프로덕트 모델 생성 방법

프로덕트 라인 모델에서 프로덕트 모델을 추출하는 과정은 원하는 제품 구성에 선택된 휘처에 따라 휘처를 구현하는 산출물(컴포넌트 혹은 아키텍처)을 추출해내는 것이다.

먼저 휘처 모델링의 결과로 나온 휘처 모델에서 사용자가 요구사항에 맞는 휘처를 선택한다. 선택된 휘처들로만 작성된 휘처 리스트를 구성한다. 그리고 가변 휘처 정보가 표현된 프로덕트 라인 모델을 준비한다. 프로덕트 추출 모듈이 휘처 리스트와 프로덕트 라인 모델의 엘리먼트에 표현된 가변 정보를 비교하여 해당 엘리먼트의 존재 여부를 결정한다. 결과적으로 휘처 리스트에 포함되어있는 휘처를 할당하고 있는 아키텍처 요소들만으로 프로덕트 아키텍처 모델이 추출되게 된다. 프로덕트 아키텍처 모델 추출 과정은 〈그림 3-7〉과 같다.



〈그림 3-7〉 프로덕트 아키텍처 모델 추출 도구 구현 과정.

한편, 프로덕트 모델 생성 방법을 이용하여 프로덕트 라인 모델로부터 휘처 선택에 따라 프로덕트 모델을 자동으로 추출해주는 도구가 필요하게 되고 본 연구에서 추출 도구를 개발하였다.

그러나 각 관점 별 프로덕트 라인 모델로부터 프로덕트 모델을 추출하는 모듈의 추가는 비슷한 접근 방법을 반복적으로 사용하기 때문에 비생산적이고 비효율적이다. 그러므로 이를 해결하기 위한 방안으로 코드를 고치지 않고 데이터 조작만으로도 재사용이 가능하도록, 네가지 뷰의 제품 라인 모델링 산출물들의 공통점을 토대로 제품 모델 추출 방식을 일반화시켰다.

그 결과 나오는 산출물은 다중 관점의 아키텍처 모델을 UML로 모델링 하였을 경우, [표 3-5]와 같다.

[표 3-5] 아키텍처 모델의 산출물.

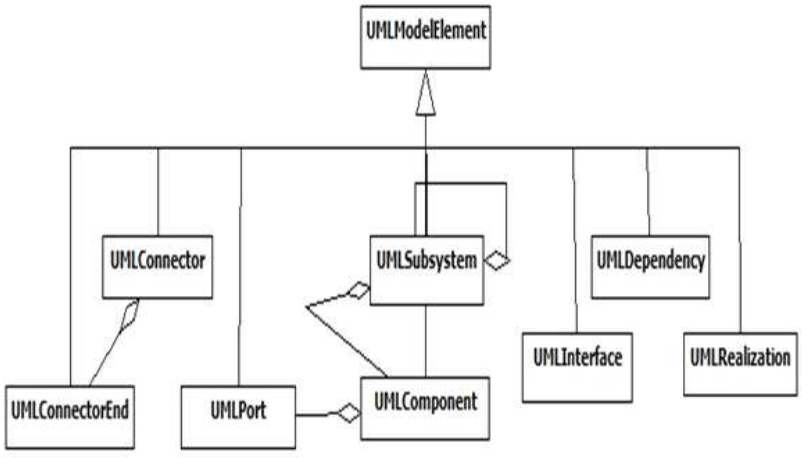
아키텍처 모델	산출물(UML Diagram)
개념 아키텍처모델	UMLComponent Diagram
프로세스 아키텍처모델	UMLComponent Diagram
배치 아키텍처모델	UMLDeployment Diagram
모듈 아키텍처모델	UMLClass Diagram

본 논문에서 UML 모델링 관련 도구로는 오픈소스인 StarUML[3]을 사용한다. StarUMLl도구를 선택한 이유는 UML 표준 명세에 기반한 모델 작성이 가능하며, UML Profile 기반의 모델링을 지원하고 모델링 플랫폼을 제공하고 있기 때문이다.

2. FORM-UML 아키텍처 프로덕트 모델 생성 방법의 일반화

먼저 개념 아키텍처 모델 단계에서의 일반화를 수행한다. 〈그림 3-8〉에서

3) StarUML, http://staruml.sourceforge.net/ko

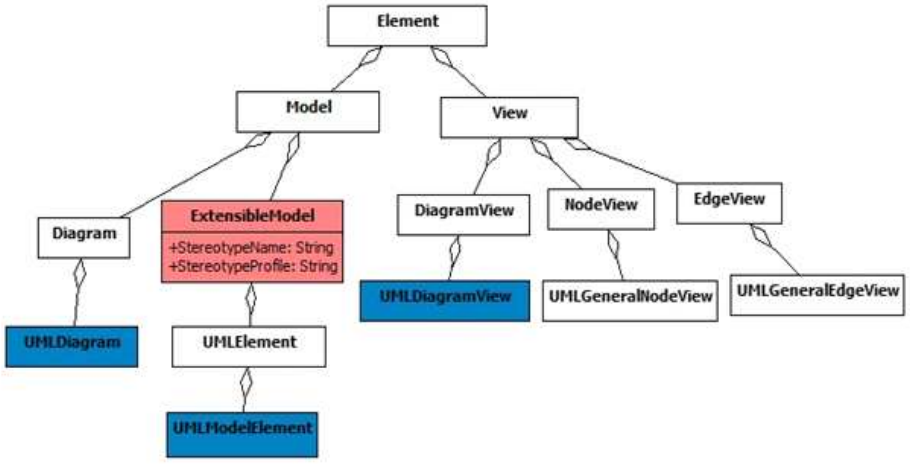


〈그림 3-8〉 개념 아키텍처 모델 구성 요소의 일반화(이지원, 2012).

표현한 엘리먼트들(서브시스템, 컴포넌트, 포트, 커넥터, 인터페이스 등)은 모두 가변점을 의미한다. 일반화를 수행하기 전에는 각각의 엘리먼트에 대응되어있는 가변 휘처와 휘처 리스트의 정보를 일일이 비교하였다. 그러나 비슷한 작업을 여러 번 수행하여야했기 때문에 비효율적이였다. 효율적인 작업 수행을 위해 이를 UMLModelElement라고 일반화시키고 프로덕트 모델 추출 과정을 한번만 수행하도록 하였다. 이는 네 가지 아키텍처의 모든 엘리먼트에 적용 가능하였다.

〈그림 3-9〉는 StarUML 도구의 메타모델[4]의 일부분이다. 프로덕트 라인 모델의 가변성을 UML의 스테레오타입을 기반으로 표현하므로 StereotypeName을 애트리뷰트로 가지는 ExtensibleModel요소의 하위에 위치한 엘리먼트들에 가변성이 있다고 판단하였다.

4) http://staruml.sourceforge.net/docs/api-doc/index.html



〈그림 3-9〉 StarUML 메타모델의 일부분(이지원, 2012).

FORM-UML 아키텍처 프로덕트 모델의 생성방법을 일반화시킨 프로덕트 모델 추출 과정은 다음과 같다.

· 프로덕트 모델 추출 과정

1) 가변성을 가지는 UMLModelElement(이하 E)에 대응되어있는 feature가 feature List에 존재하면 해당 E는 유지, 존재하지 않으면 제거
2) 제거되는 E를 참조하는 E가 존재한다면, 참조하는 E를 제거
3) 제거되는 E와 Aggregation관계에 있는 E를 제거

(예를 들어, 〈그림 3-8〉로 말미암아 UMLComponent엘리먼트가 제거된다면, 이에 포함되어있는 UMLPort엘리먼트도 함께 제거되어야한다.)

[표 3-6]은 프로덕트 모델 추출 과정을 통해 생성한 productDerivation 함수의 Pseudo Code이다.

[표 3-6] productDerivation Pseudo Code.

```
procedure productDerivation(featureList):
create a queue Q
  if UMLModelElement(이하 E) has StereotypeName:
    if StereotypeName has not feature:
        E is offered into Q.
        if E has reference attribute:
          referencing E is offered into Q.
        else if referenced E is:
          referencing Infomation is offered into Q.
        else if E has relationship:
       referencing E's parent is offered into Q.
        remove(Q.poll()).
```

먼저 입력 데이터로 휘처 리스트를 받는다. 휘처 리스트의 내용은 xml 코드로 작성되어있다. 엘리먼트들은 UMLModelElement의 스테레오타입에 휘처 이름을 입력함으로써 가변성이 표현된다. 스테레오타입 정보에서 휘처이름을 파싱하여 정보를 얻어낸다. 그리고 스테레오타입에 있는 휘처와 휘처리스트에 속한 휘처들을 대조한다.

결론적으로 표현하면, 엘리먼트가 스테레오타입을 가지고 있는지 확인하고 스테레오타입이 입력받은 휘처 리스트에 속해 있는 휘처를 가지고 있는지를 확인하는 것이다. 만약 휘처 리스트에 있는 휘처를 엘리먼트가 표현하고 있지 않다면, 해당 엘리먼트는 제거 대상이 된다. 또한 해당 엘리먼트를 제거함으로써 영향을 받는 속성이 있는 경우, 이 또한 제거 대상이 된다.

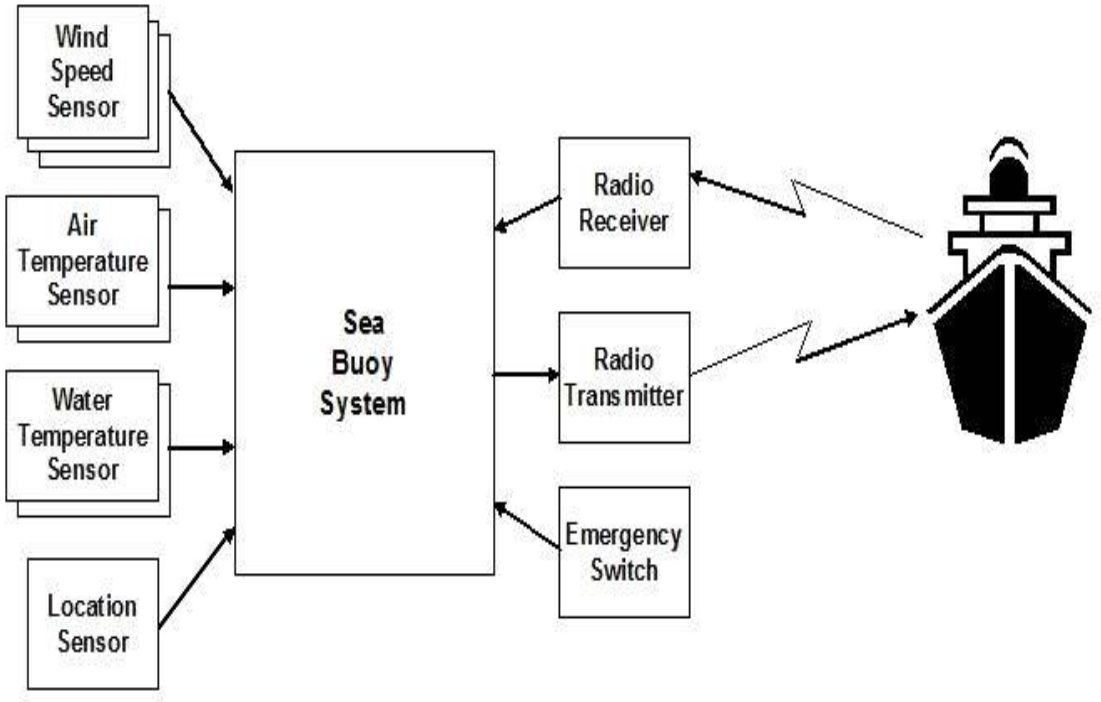
향후에는 이러한 아키텍처 수준의 산출물 모델 뿐만 아니라 전반적인 제품라인 모델링 과정에 걸친 산출물 모델에, 해당 프로시저의 효율성을 테스트하기 위하여 이를 적용할 예정이다.

제 4 장 연구 결과 및 적용 사례

본 장에서는 바다 부표 시스템 도메인을 통해 3장에서 설명한 FORM-UML 아키텍처의 가변성 모델링 예제를 작성하고, 아키텍처 프로덕트 모델 생성 도구에 이를 적용한다.

제 1 절 FORM 아키텍처 모델링 예시 및 적용

본 논문에서는 바다 부표 시스템 도메인을 제품라인으로 구축하였다. 바다 부표 시스템이란 선박의 안전한 항해를 돕기 위하여 암초나 침몰 선박 등 항해상의 위험물에 대한 존재를 경고하기 위하여 설치된다. 또한 바닷물의 온도, 날씨, 현재 위치 등을 알려주는 역할도 수행한다.



〈그림 4-1〉 바다 부표 시스템.

1. FORM-UML 기반의 바다 부표 시스템 휘처 모델

바다 부표 시스템 도메인 모델에 대한 요구사항은 다음과 같다.

· 몇몇의 부표는 공기의 온도를 10초마다 리딩한다.
· 물의 온도를 10초마다 리딩한다.
· 바람의 속도를 30초마다 리딩한다.
· 부표의 현재 위치를 10초마다 리딩한다. 현재 위치를 알아내는 방법에는 부표의 절대적인 위치와 어떤 대상과의 상대적인 위치가 있다.
· 날씨, 지역 정보를 60초마다 방송한다.
· SOS 요청을 받는다.
· SOS요청을 받으면 SOS 메시지를 방송한다.
· 선박으로부터 요청을 받는다.
· 몇몇의 부표는 Red Light 장비를 갖추고 있다.
· Red Light 장비는 sea-search 기능을 하는 동안 선박을 통해 활성화된다.
· Radio Receiver를 통해 요청을 받으면 하루 동안의 날씨, 지역 정보를 평균으로 방송한다.
· 방송의 우선 순위는 주기적인 방송 〈 요청으로 인한 방송 〈 SOS 방송 순이다.

바다 부표 시스템의 도메인 요구 사항을 참고하여 작성한 요구사항 분석 모델과 요구 사항 명세서는 [부록]을 참조하기 바란다.

바다 부표 시스템에 대해 공통성과 가변성을 구분한 내용은 다음 [표 4-1]과 같다.

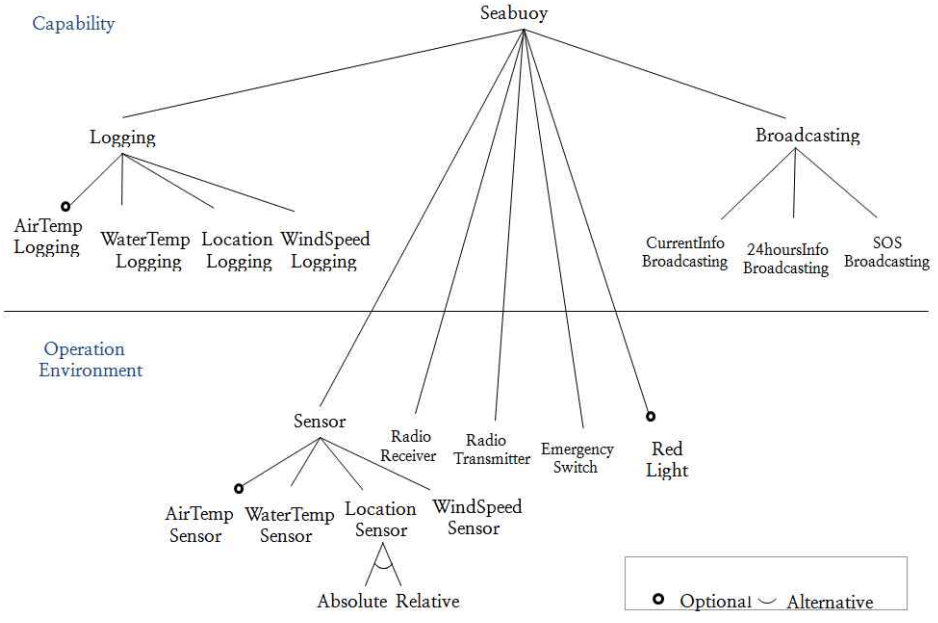
공통성과 가변성을 바탕으로 휘처 모델을 작성한다. 〈그림 4-2〉는 바다 부표 시스템의 휘처 모델이다.

바다 부표 휘처 모델은 2가지 범주로 작성되었다. 능력 계층에서는 바다 부표 시스템이 가지는 주요 특징을 기술하였고, 운영 환경 계층에서는 시스템이 운영되는 환경을 기술하였다. 몇몇의 부표에만 존재하는 공기 온도 센서와

[표 4-1] 바다 부표 시스템의 공통성과 가변성.

바다 부표 시스템	
공통성	· 바다의 정보(“바다의 온도, 바다 부표의 위치, 바람의 속도”,이하 동일) 작성 · 바다의 정보 전송 · 바다의 정보 방송 · 긴급 스위치의 작동
가변성	· 바다의 공기의 온도 작성, 전송, 방송 · 부표의 위치 센서의 종류 · Red LIght의 작동 · 바다의 정보 방송 순서

Red Light는 선택적 가변성을 가지고 있음을 표현하였다. 한편, 위치 센서의 서브 휘처인 Absolute와 Relative휘처는 바다 부표 시스템의 프로덕트 모델에서 반드시 하나는 선택되어야하는 택일적 가변성을 가지고 있다.



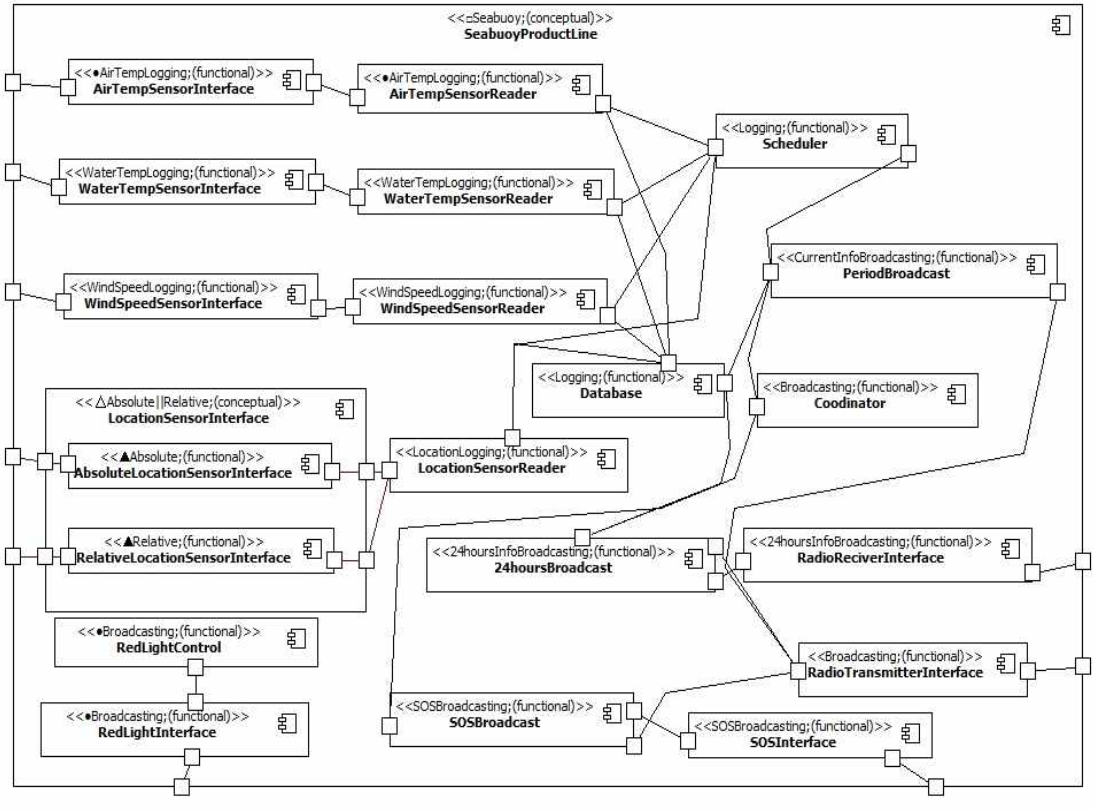
〈그림 4-2〉 바다 부표 시스템의 휘처 모델.

2. 바다 부표 시스템 아키텍처 모델 가변성 적용 사례

휘처 모델을 기반으로 바다 부표 시스템 도메인에 대해 가변성을 적용한 FORM-UML의 개념, 프로세스, 배치 뷰에 대한 모델은 다음과 같다.

1) 개념 뷰

바다 부표 시스템의 공통성과 가변성을 표현한 FORM-UML의 개념 뷰는 〈그림 4-3〉와 같다.



〈그림 4-3〉 바다 부표 시스템의 개념 뷰.

SeabuoyProductLine 개념 컴포넌트 내부에 각 센서의 Reader와 Interface, Scheduler, Coodinator, Database, Broadcast등의 하위 컴포넌트들이 위치해 있다. 특히 LocationSensorInterface는 AsoluteLocationSensorInterface와 RalativeLocationSensorInterface로 분해(Decomposition)된다.

AbsoluteLocationSensorInterface와 RalativeLocationSensorInterface는 서로 대안적(Alternative)인 성격을 가진다. 그렇기 때문에 LocationSensorInterface 개념 컴포넌트는 내부의 대안적 가변성을 가진다는 것을 스테레오타입으로 표현한다. 한편, RedLightControl과 RedLightInterface 컴포넌트는 선택적(Optional)가변성을 가진다. 각 컴포넌트의 성격과 대응되는 휘처, 그리고 가변성을 스테레오타입으로 표현한다. 각각 말단의 컴포넌트는 기능 컴포넌트로써 명명하고 스테레오타입으로 functional한 성격을 표현한다.

컴포넌트 레벨에서의 가변성 표기법을 정리하면, 우선 컴포넌트 레벨의 가변성을 표현하는 Variation_Mark를 도형으로 표현한다. Variation_Mark는 3장 2절의 〈표 3-4〉를 참고한다. 그리고 컴포넌트에 대응되는 휘처들의 연산정보를 입력하게 되는데, 해당 컴포넌트의 존재 유무에 영향을 미치는 휘처들의 연산으로 구성된다.

· 표기법

- 〈〈[Variation_Mark][Allocated Feature List];(FORM Architecture View Type)〉〉

세미콜론(;)을 중심으로 앞에는 대응되는 가변 휘처들의 정보를 기입하고 세미콜론의 뒤에는 각 요소의 뷰 타입을 기입한다. Allocated Feature List에 사용되는 연산은 예를 들어 설명하면 다음과 같다.

· [Allocated Feature List]

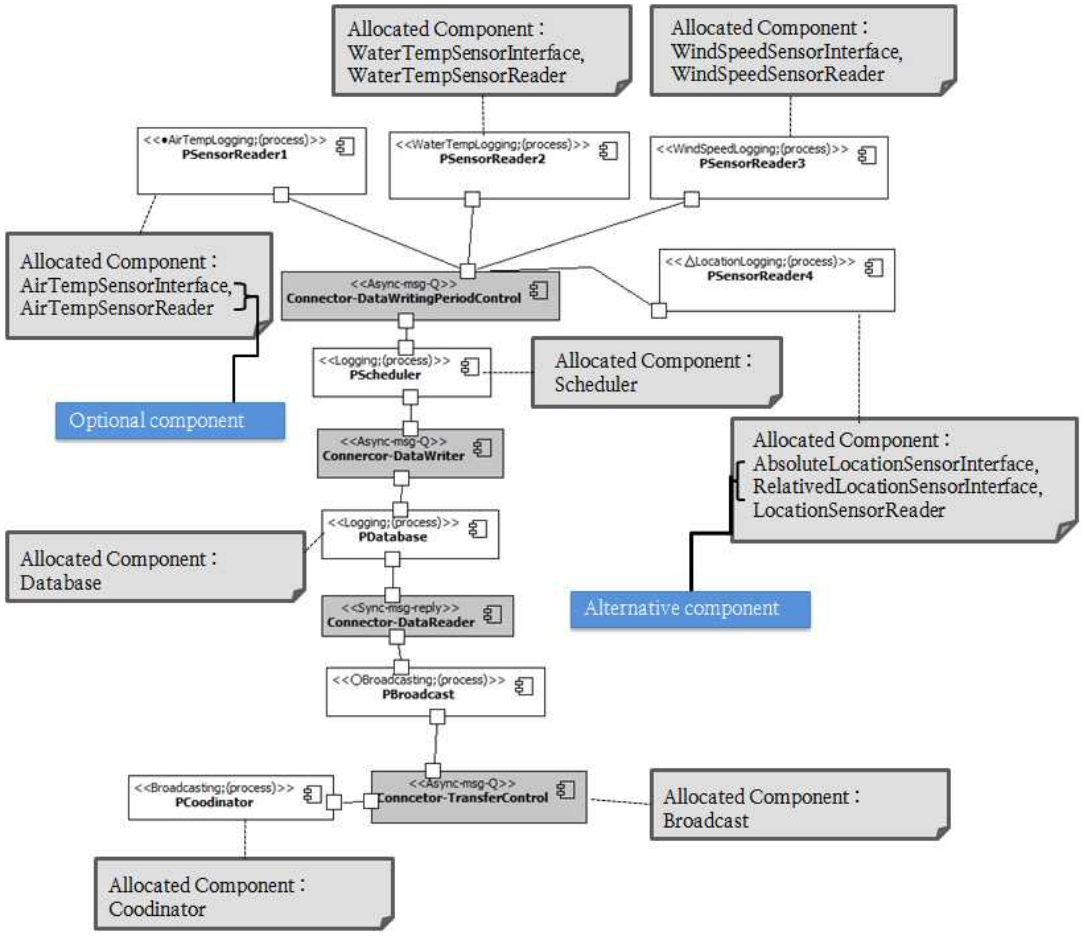
- F1&&F2; : 해당 요소가 존재하기 위해서, 휘처 F1과 F2는 반드시 선택되어야한다.
- F1||F2; : 해당 요소가 존재하기 위해서, 휘처 F1과 F2중 오직 하나만 선

택이 되어야한다.

2) 프로세스 뷰

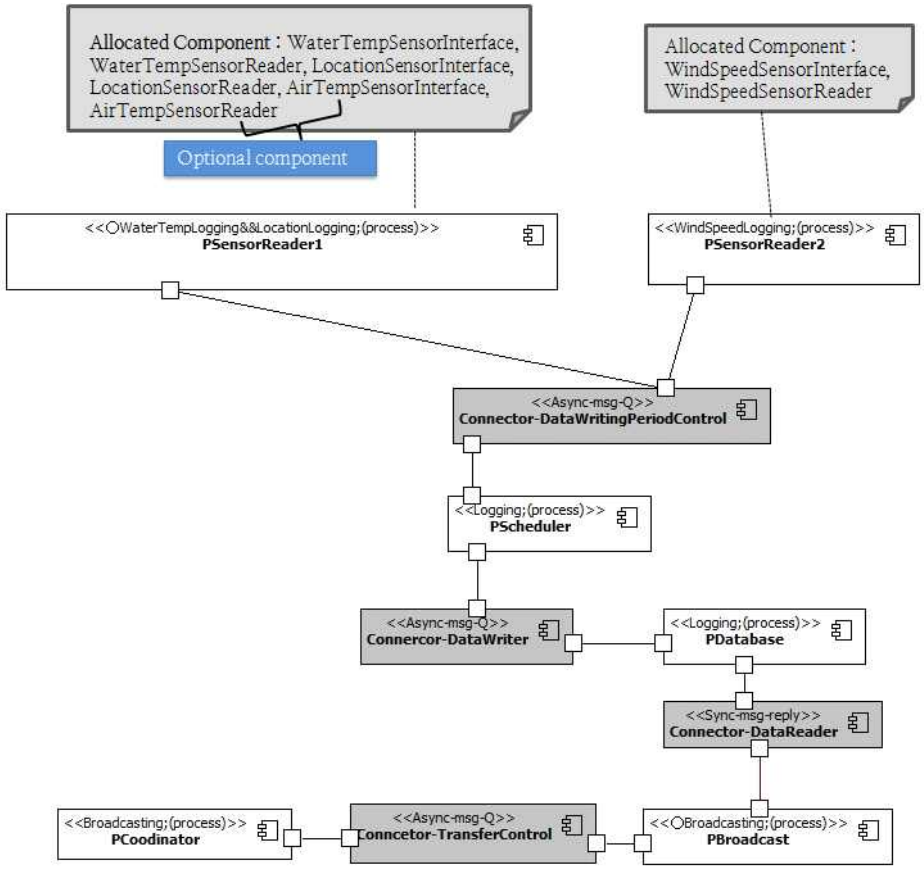
〈그림 4-4〉과 〈그림 4-5〉는 바다 부표 시스템의 프로세스 뷰에 대한 사례이다.

〈그림 4-4〉의 프로세스 뷰에서는 네 개의 PSensorReader 프로세스 컴포넌트들이 각각 네 개의 SensorReaderInterface와 SensorReader를 할당하고 있다.



〈그림 4-4〉 바다 부표 시스템의 프로세스 뷰.

특히 PSensorReader1은 선택적(Optional)가변성을 가지는 AirTempSensorInterface 개념 컴포넌트와 AirTempSensorReader 개념 컴포넌트를 할당하고 있다. 그리고 PSensorReader1 프로세스 컴포넌트 자체가 선택적(Optional) 성격을 가지고 있기 때문에 이를 〈〈●〉〉노테이션으로 표현하였다. PSensorReader4 프로세스 컴포넌트는 프로세스 컴포넌트 자체는 필수(Mandatory) 성격을 지니지만, 대응되어있는 AbsoluteLocationSensorInterface와 RelativeLocationSensorInterface가 서로 대안적(Alternative) 가변성을 가지고 있기 때문에 이를 내부에 대안적 가변성이 있다는 표현으로 〈〈△〉〉노테이션을 기입한다.

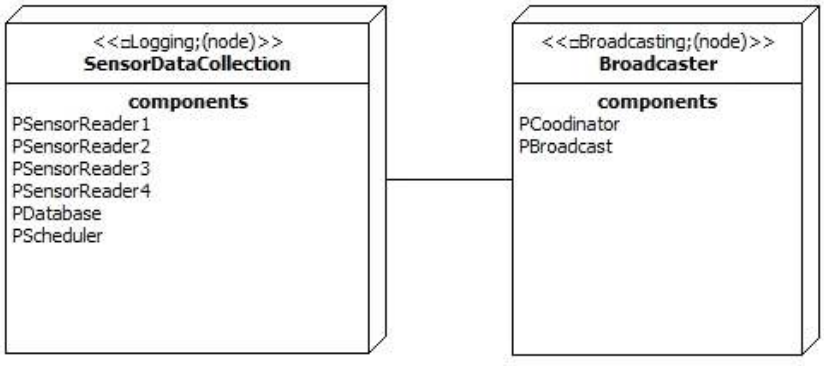


〈그림 4-5〉 바다 부표 시스템의 프로세스 뷰.

한편, 도메인 모델의 요구사항에서 AirTempSensor, WaterTempSensor, LocationSensor는 10초마다 바다의 정보를 읽어 들이고 WindSpeedSensor는 30초마다 바다의 정보를 읽어 들인다고 하였다. 스케쥴링 오버헤드를 줄이기 위하여 같은 시간 단위로 바다의 정보를 읽어 들이는 개념 컴포넌트에 대응하고 있는 프로세스 컴포넌트들을 하나의 프로세스 컴포넌트로 통합하였다. 〈그림 4-5〉는 이를 표현한 것이다. 〈그림 4-5〉의 프로세스 아키텍처 모델은 〈그림 4-4〉의 프로세스 아키텍처 모델에 비해서 시스템 전체 퍼포먼스를 개선 시키는 모델이라고 볼 수 있다. 한편, PSensorReader1 프로세스 컴포넌트에 대응되어있는 AirTempSensorInterface와 AirTempSensorReader 컴포넌트가 선택적 가변성을 가지고 있으므로 PSensorReader1 프로세스 컴포넌트는 내부에 선택적 가변성을 가지고 있다고 표현하였다. 그리고 AirTempLogging 휘처는 PSensorReader1 프로세스 컴포넌트의 존재 유무에 영향을 미치지 않으므로 요소의 스테레오타입에 기입하지 않았다.

3) 배치 뷰

〈그림 4-6〉는 바다 부표 시스템의 배치 아키텍처를 모델링한 것이다. SensorDataCollection 노드는 센서를 읽어 들이는 프로세스 컴포넌트들과 PDatabase, PScheduler 프로세스 컴포넌트를 할당하고 있고, Broadcaster 노드는 PCoodinator, PBroadcast 프로세스 컴포넌트를 할당하고 있다. 또한



〈그림 4-6〉 바다 부표 시스템의 배치 뷰.

SensorDataCollection노드는 Logging휘처를 할당하고 있고, Broadcaster 노드는 Broadcasting 휘처를 할당한다. 두 노드는 내부에 여러 가지 가변성을 가지고 있기 때문에 〈〈□〉〉노테이션으로 표현하였다.

다음 절에서는 가변성을 표현한 바다 부표 시스템의 프로덕트라인 아키텍처 모델에서 프로덕트 모델을 생성하는 도구를 사용한 사례를 설명한다.

3. 아키텍처 프로덕트 모델 생성 도구 적용 사례

프로덕트 라인 아키텍처 모델에서 프로덕트 아키텍처 모델을 생성하는 도구에 먼저 선택된 휘처 정보를 담은 휘처 리스트(xml 파일)를 입력시킨다. 선택되지 않은 휘처에 대응되는 각 뷰의 요소들은 삭제되고 프로덕트 아키텍처 모델이 생성되어진다.

〈그림 4-7〉은 가변 정보를 가지고 있는 휘처 리스트와 선택된 휘처 리스트이다.

Optional한 성격을 가진 AirTempLogging 휘처와 AirTempSensor 휘처가 선택되지 않았고, Alternative한 성격을 가진 Absolute 휘처와 Relative 휘처 중 Absolute 휘처만이 선택된 것을 알 수 있다.

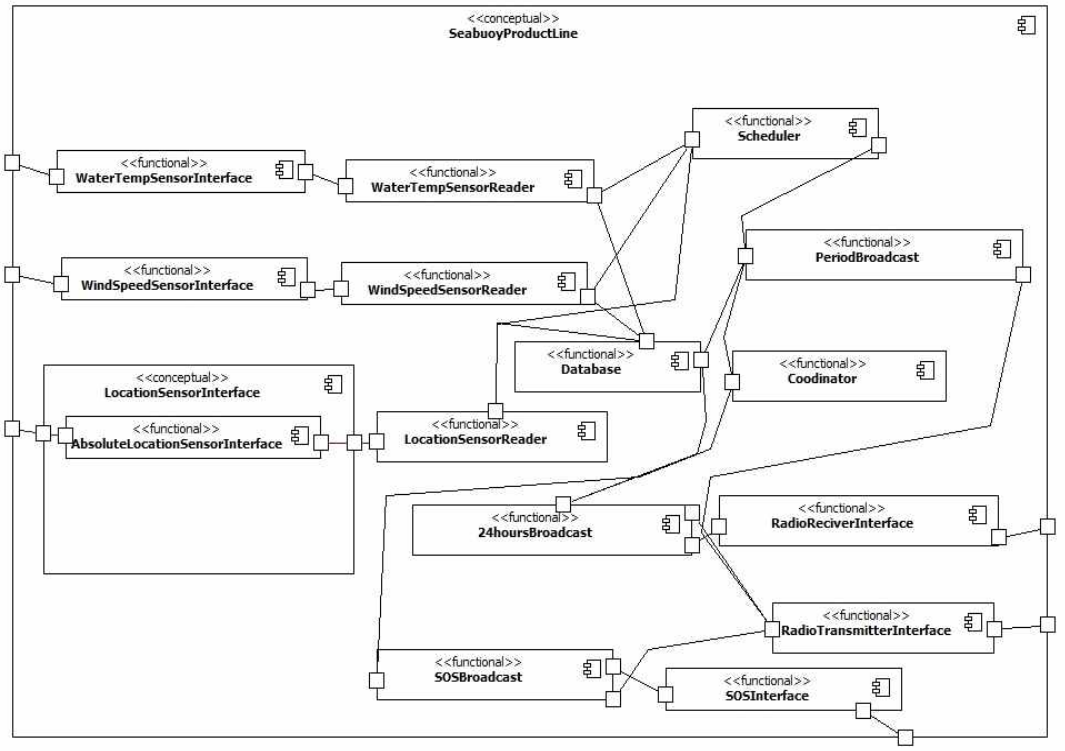
FORM 아키텍처 프로덕트 모델을 추출하는 기능을 수행하면, 휘처 리스트에 선택되어있는 휘처 정보에 따라 아키텍처 가변 요소의 존재를 결정한다. 그 후, 존재할 수 있는 요소들로만 FORM 아키텍처 프로덕트 모델로 구성되어진다.

〈그림 4-8〉은 FORM 아키텍처 프로덕트 모델을 추출하는 기능을 수행한 결과이다. 휘처 리스트에 있는 휘처의 정보에 따라 프로덕트 모델이 생성된 바다 부표 시스템의 개념 프로덕트 모델이다. 선택적 가변성을 가지고 있던 AirTempSensorReader컴포넌트와 AirTempSensorInterface컴포넌트는 대응되어있던 AirTempLogging 휘처가 존재하지 않았으므로 프로덕트 모델에서 제거되었다. 그리고 LocationSensorInterface는 내부에 대안적 성격을 가지는 서브 컴포넌트들을 가지고 있었는데, 이 들 중 하나만이 선택되어 개념 프로덕

〈그림 4-7〉 가변 정보를 가진 휘처 리스트와 선택된 휘처 리스트.

트 모델이 구성되었다.

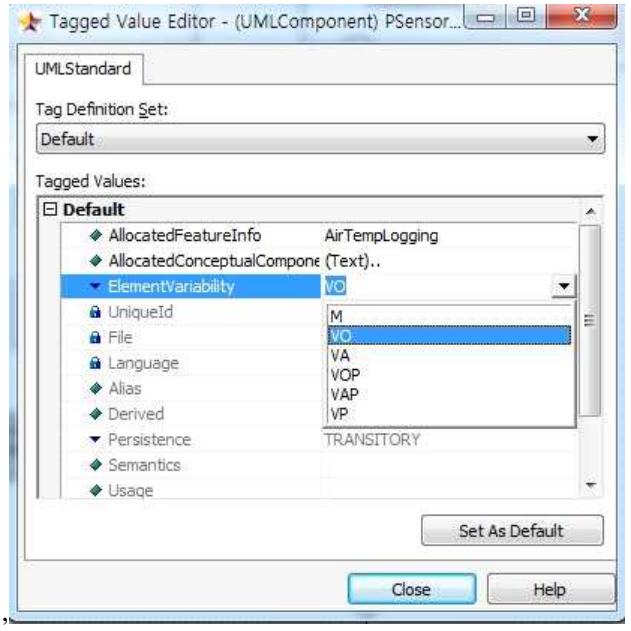
또한, 프로덕트 모델은 가변 정보가 필요없으므로 생성될 때, 가변 정보가 모두 사라지게 되며, 휘처 정보 또한 스테레오타입에서 제거된다. 프로덕트 모델의 뷰 타입만이 남게 된다.



〈그림 4-8〉 바다 부표 시스템의 개념 프로덕트 모델.

한편, 〈그림 4-9〉는 프로세스 아키텍처 모델에서 사용할 수 있는 태그 값 편집기(Tagged Value Editor)이다. 대응되는 휘처 정보와 개념 컴포넌트 정보, 그리고 각 요소의 가변성 정보를 기입할 수 있다. 태그 값은 아키텍처 요소의 다양한 정보를 입력하기에는 좋으나, 가시적인 표현이 어려운 문제점을 가진다.

따라서 가변성 표현이 필요한 프로덕트 라인 모델을 표현할 때, 부가적으로 구현한 Product Line Model Variability Expression 기능을 사용한다. Product Line Model Variability Expression 기능은 프로덕트 라인 아키텍처 모델에서 기존의 태그 값(Tagged Value)으로 표현한 가변 정보가 가시적으로 스테레오타입으로 표현하는 역할을 한다. 4장 1절은 가변성 표현의 가시화를 가정하고 가변성 모델링의 예시를 설명한 것이다.



〈그림 4-9〉 Product Line Model Customized Tagged Value Editor.

〈그림 4-9〉의 ElementVariability 항목의 M은 Mandatory의 약자로써 필수 요소를 나타내는 것이고 이는 UML모델 상에서 표현되지 않는다. 한편 VO, VA는 각각 Variability Optional, Variability Alternative를 나타내며, 이는 스테레오타입으로 가시화할 때, 〈〈●〉〉, 〈〈▲〉〉으로 표현한다. 그리고 VOP (Variability Optional Point), VAP (Variability Alternative Point) 는 내부의 가변성을 나타내며, 〈〈○〉〉, 〈〈△〉〉으로 표현된다.

제 5 장 결 론

소프트웨어 프로덕트 라인의 목적은 동일한 도메인 내에 속한 제품을 개발할 때, 미리 구축된 핵심자산들을 조합하여 생산해 냄으로써 소프트웨어 프로덕트를 보다 빠르게 생산해내는 데 있다. 그렇기 때문에 핵심자산의 가변성을 표현하고 관리하는 것이 무엇보다 중요하다.

본 논문에서는 소프트웨어 프로덕트 라인을 개발할 때, 관리해야하는 가변성을 소프트웨어 아키텍처 레벨과 컴포넌트 레벨에서 정의하고, FORM-UML 모델링 기법을 제안하였다. 그리고 가변성이 표현된 프로덕트 라인 모델로부터 프로덕트 모델로의 추출을 도와주는 도구를 개발하였다.

소프트웨어 프로덕트 라인 개발 단계에서 가변성을 명확히 표현하고 관리하기 때문에 컴포넌트 기반의 프로덕트 라인을 구축할 경우, 생산성을 향상시킬 것이다.

향후 연구 과제로는 프로덕트 모델 추출 도구를 확장하여 실제 기업체 예제에 적용해 방법론과 도구의 효율성을 평가할 예정이다.

【참고문헌】

1. 국내문헌

강교철 외 3인, 「임베디드 시스템 개발 생산성 향상을 위한 소프트웨어 제품 라인 공학」, 포항공대, 2004.

이관우, 「프로덕트 라인 아키텍처의 재사용성, 적응성, 구성성을 위한 휘처 중심의 설계」, 2003.

이관우 외 1인, 「UML 기반의 프로덕트 라인 아키텍처 모델링」, 한국 정보과학회, 한국 컴퓨터 종합 학술대회 논문집 (A), pp.210-213, 2011.

이지원 외 1인, 「제품라인모델로부터 제품모델을 추출하는 기법 및 도구의 일반화」, 한국 정보처리학회 추계학술발표대회 논문집 제 19권 2호, pp.1555-1558, 2012.

정보통신산업진흥원, 소프트웨어 재사용을 기반으로 한 소프트웨어 제품라인 공학 기술 적용 가이드, 정보통신산업진흥원, 2011.

2. 국외문헌

America, P., Obbink, H., Muller, J., & van Ommering, R. 2000, "COPA : A Component-Oriented Platform Architecting Method for Families of Software Intensive Electronic Products,". Dever, Colorado; The First Conference on Software Product Line Engineering.

Atkinson, C., Bayer, J., & Muthig, D. 2000, "Component-based product line developmenet: the KobrA approach," in 1st Int'l Software Product Line ConfDenver, Colorado, United States: Kluwer Academic Publishers, pp. 289-309.

Atkinson, C., Bayer, J., & Muthig, D. 2002, *Component-based product line engineering with UML*. Addison-Wesley. London, New York.

Clauss, M. 2001, Modeling variability with UML. GCSE 2001 - Young Researchers Workshop.

Clements, P., Garlan, D., Little, R., & Nord, R. 2002, *Documenting Software Architectures: Views and Beyond*, Addison Wesley.

Gomaa, H., 1992, *Software Design Methods for Concurrent and Real-Time Systems*, Addison-Wesley.

Gomaa, H., 2004, *Desinging Software Product Line with UML : From Use Cases To Pattern- based Software Architectures*, Addison-Wesley.

Hoek, A., Mikic-Rakic, M., Roshandel, R., & Medvidovic, N. 2001, "Taming Architectural Evolution," Proceedings of the Ninth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE).

Hofmeister, C., Nord, R., & Soni, D., 2000, *Applied Software Architecture*, Addison-Wesley, Reading, MA.

Kang, K., Kim, S., Lee, J., Kim, K., Shin, E., & Huh, M. 1998,

"FORM: A Feature-Oriented Reuse Method with Domain Specific Reference Architectures", Ann. Soft. Eng. 5 : pp. 143-168.

Kang, K., Cohen, S., Hess, J., Nowak,W., & Peterson, S. " Feature-Oriented Domain Analysis(FODA) Feasibility Study", Technical Report CMU/SEI-90-TR-21, Pittsburgh, PA, Software Engineering.

Kruchten, P., 1995, *Architectural Blueprints - The "4+1" View Model of Software Architecture,* IEEE Software 12 (6): pp. 42-50,

Ommering R., van der Linden, F., Kramer, J., & Magee, J. 2000, "The Koala Component Model for Cunsumer Electronics Software," IEEE Computer Vol 33, Issue 3 : pp.78-85.

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

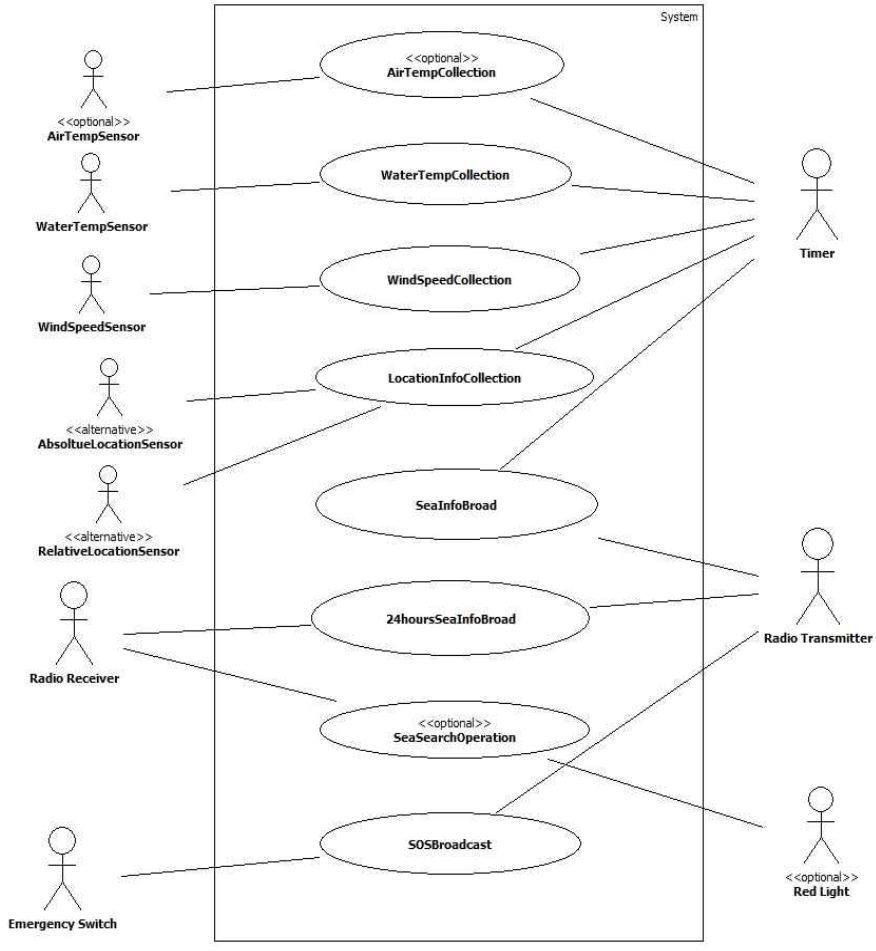
Spinczyk, Olaf., Beuche, D. 2004, Modeling and Building Software Product Lines with Eclipse., Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages and applications : pp.18-19.

Weiss, D. M., & Lai, C. T. R. 1999, *Software Product-Line Engineering : A Family-Based Software Development Process, Reading, MA*: Addison Wesley Longman, Inc.

【부 록】

바다 부표 시스템 요구 사항 명세

그림은 요구 사항을 토대로 구성한 바다 부표 시스템의 유스케이스 모델이다.



〈그림〉 바다 부표 시스템의 유스케이스.

다음 표는 바다 부표 시스템의 Actor 명세서이다.

〈표〉 바다 부표 시스템의 Actor 명세서.

Actor	Description
AirTemp Sensor	free-floating buoy의 external device로 공기의 온도를 측정하는 센서이다. 모든 부표에 다 있는 device는 아니며, optional한 device이다.
Location Sensor	free-floating buoy의 external device로 위치를 읽는 센서이다. Location Sensor에는 절대적인 위치로부터 위치를 읽는 센서와 상대적인 위치로부터 위치를 읽는 센서가 있다. 그리고 이들은 서로 alternative한 성격을 가진다.
WaterTemp Sensor	free-floating buoy의 external device로 물의 온도를 측정하는 센서이다.
WindSpeed Sensor	free-floating buoy의 external device로 바람의 속도를 측정하는 센서이다.
Emergency Switch	free-floating buoy의 external device로 위급 상황시 누르는 위급상황 스위치이다.
Radio Receiver	free-floating buoy의 external device로 지나가는 선박을 통해 요청을 받는 수신기이다.
Radio Transmitter	free-floating buoy의 external device로 정보를 외부로 방송하는 송신기이다.
Red Light	free-floating buoy의 external device로 지나가는 선박에 의해 활성화 혹은 비활성화 되어지는 red light이다. 모든 부표에 다 있는 device는 아니며, optional한 device이다.
Timer	free-floating buoy의 external device로 부표시스템에 시간을 알려주는 타이머이다.

바다 부표 시스템의 유스케이스 명세서는 다음과 같다.

1) 24hoursSeaInfoBroad

가) Description

바다의 24시간 정보를 방송한다.

나) Flow of Events

(1)Basic Flow

1. RadioReceiver는 시스템에게 24시간동안의 데이터를 방송해줄 것을 요청한다.
2. 시스템은 방송의 우선순위설정에 따라 Transmitter에게 24시간데이터를 방송해줄 것을 요청한다.
3. Transmitter는 시스템에게 성공적으로 방송이 되었음을 알린다.

(2) Alternative Flow

2.a SOS요청을 받았을 때, 방송을 멈추고, SOSBroadcast 유스케이스를 시작한다.

다) Pre-Conditions

1. SOS요청이 비활성화상태이다

라) Post-Conditions

1. 시스템에 24시간의 바다 정보가 방송된 이력이 남는다.

2) AirTempCollection

가) Description

바다의 공기 온도를 설정시간단위(10초)로 수집한다.

나) Flow of Events

(1) Basic Flow

1. Timer가 설정된 시간(s)이 되었을 때, 시스템에 설정된 시간이 되었다고 알린다.
2. 시스템은 외부에 존재하는 모든 AirTempSensor에게 바다의 공기 온도 정보 데이터값을 달라고 요청한다.

3. AirTempSensor는 시스템에 감지한 바다의 공기 온도 정보를 전달한다.

4. 시스템은 바다의 공기 온도 정보를 데이터베이스에 저장한다.

다) Pre-Conditions

1. AirTempSensor가 존재한다.

2. Timer에 count할 시간(10초)이 설정된 상태이다.

라) Post-Conditions

1. Air Temp Sensor를 통해 바다의 현재공기 온도 데이터가 update 된다.

3) LocationInfoCollection

가) Description

바다의 위치 정보를 설정된 시간 단위(10초)로 수집한다.

나) Flow of Events

(1) Basic Flow

1. Timer가 설정된 시간(s)이 되었을 때, 시스템에 설정된 시간이 되었다고 알린다.

2. 시스템은 Sensor에게 바다의 위치 정보 데이터 값을 달라고 요청한다.

3. Sensor는 시스템에 감지한 바다의 위치 정보를 전달한다.

4. 시스템은 정보를 데이터베이스에 저장한다.

다) Pre-Conditions

1. AbsoulteLocationSensor와 RelativeLocationSensor 중 하나의 Sensor 가 선택된 상태이다.

2. Timer에 count할 시간(10초)이 설정된 상태이다.

라)Post-Conditions

1. Location Sensor를 통해 바다의 현재위치 데이터가 update된다.

4) SOSBroadcast

가) Description

Emergency Switch를 통해 SOS요청을 받고 SOS신호를 주기적으로 방송한다.

나) Flow of Events

(1)Basic Flow

1. Emergency Switch가 시스템에게 위급상황을 알린다.
2. 시스템이 Transmitter에게 위급상황을 지속적으로 방송하라고 요청한다.
3. Transmitter가 시스템에게 성공적으로 위급상황이 방송되었음을 알린다.
4. 2~3이 연속적으로 반복된다.
5. Emergency Switch가 off된다.(선원에 의해서)
6. 시스템은 Transmitter의 SOS방송을 멈출 것을 요청한다.

다) Post-Conditions

1. 시스템에 SOS방송 기록 데이터가 남는다.

5) SeaInfoBroad

가) Description

바다의 정보를 설정시간단위(60초)로 방송한다.

나) Flow of Events

(1) Basic Flow

1. Timer가 해당시간이 되었을 때, 이를 시스템에 알려준다.
2. 시스템은 바다의 정보를 RadioTransmitter에게 방송할 것을 요청한다.
3. Radio Transmitter는 시스템에게 방송이 성공적으로 되었다고 알려준다.

(2) Alternative Flow

2.a. SOS 요청을 받았을 때, 방송을 멈추고 SOSBroadcast 유스케이스를 시작한다.

2.b. 24시간데이터 방송 요청을 받았을 때, 주기적인 방송을 멈추고 24hoursSeaInfoBroad 유스케이스를 시작한다.

다) Pre-Conditions

1. SOS요청이 비활성화상태이다.
2. Timer에 방송해줄 시간(60초)이 설정된 상태이다.

라) Post-Conditions

1. 시스템에 주기적인 방송기록이 남는다.

6) SeaSearchOperation

가) Description

바다를 검색하고 Red Light을 활성화한다.

나) Flow of Events

(1) Basic Flow

1. RadioReceiver는 시스템에게 지나가는 선박에 의한 활성화 요청을 전달한다.
2. 시스템이 Red Light을 킨다.

(2) Alternative Flow

Radio Receiver가 시스템에게 선박의 비활성화 요청을 전달하는 경우에, 2.a. 시스템이 Red Light을 끈다.

다) Pre-Conditions

1. 시스템에 Red Light이 포함되어있다.
2. 부표시스템이 sea-search operation을 수행중에 있다.
3. Red Light을 활성화 시키는 요청을 받을 때, Red Light은 비활성화 상태에 있다.
4. Red Light을 비활성화 시키는 요청을 받을 때, Red Light은 활성화 상태에 있다.

라) Post-Conditions

부표가 SeaSearchOperation을 하는 동안, Red Light이 활성화/비활성화 된다.

7) WaterTempCollection

가) Description

바다의 물의 온도를 설정시간단위(10초)로 수집한다.

나) Flow of Events

(1) Basic Flow

1. Timer가 설정된 시간(s)이 되었을 때, 시스템에 설정된 시간이 되었다고 알린다.
2. 시스템은 WaterTempSensor에게 바다의 물 온도 정보 데이터 값을 달라고 요청한다.
3. WaterTempSensor는 시스템에 감지한 바다의 물 온도 정보를 전달한다.
4. 시스템은 바다의 물 온도 정보를 데이터베이스에 저장한다.

다) Pre-Conditions

1. Timer에 count할 시간(10초)이 설정된 상태이다.

라) Post-Conditions

1. Water Temp Sensor를 통해 바다의 현재 물 온도 데이터가 update된다.

8) WindSpeedCollection

가) Description

바다의 바람의 속도를 설정시간단위(30초)로 수집한다.

나) Flow of Events

(1) Basic Flow

1. Timer가 설정된 시간(s)이 되었을 때, 시스템에 설정된 시간이 되었다고 알린다.
2. 시스템은 외부에 존재하는 WindSpeedSensor에게 바다의 바람 속도 정보 데이터 값을 달라고 요청한다.
3. WindSpeedSensor는 시스템에 감지한 바다의 바람 속도 정보를 전달한다.

4. 시스템은 바람의 속도 정보를 데이터베이스에 저장한다.

다) Pre-Conditions

1. Timer에 count할 시간(30초)이 설정된 상태이다.

라) Post-Conditions

1. Wind Speed Sensor를 통해 바다의 현재 바람 속도 데이터가 update된다.

ABSTRACT

FORM-UML : Development of Variability Modeling Technique and its supporting Tool for UML-based Product Line Architecture

Lee, Ji-won
Major in Information System Engineering
Dept. of Information System Engineering
The Graduate School
Hansung University

In general, a product line architecture is considered the key to the success of product line engineering. FORM(Feature Oriented Reuse Method), which is a representative methodology of product line engineering, defines product line architectures in terms of concept, process, deployment, and module. FORM architecture models, however, have a compatibility problem with models from different methodologies, as they are represented in their own way. Moreover, they fail to represent variability explicitly at the architectural level.

This thesis proposes FORM-UML, which defines FORM architecture models using UML(Unified Modeling Language)-industrial de facto standard for modeling language, and presents the way of modeling variability explicitly at the architecture level.

FORM-UML uses the extension mechanisms (i.e., tagged values and stereotype) to express architectural variability. Especially, it presents how

to model variability in each type of the architectural models as well as variability in mapping between different types of the architectural models. Moreover, we have developed an automation tool for extracting product architectures from product line architectures modelled with the proposed technique. Finally, the proposed technique and the automation tool are validated through the case study of the sea buoy system.

[Key words] Software Product Line Engineering, Feature Oriented Reuse Method, UML, architectural models, variability modeling, tool development.