프로덕트 라인 공학의 핵심자산 재사용성 평가를 위한 품질시스템

(A Quality System for Evaluating Reusability of Core Assets in Product Line Engineering)

오 상 헌 † 허 진 선 † 김 지 혁 † 류 성 열 †† 김 수 동 ††
(Sang Hun Oh) (Jin Sun Her) (Ji Hyeok Kim) (Sung Yul Rhew) (Soo Dong Kim)

요 약 프로덕트 라인 공학(Product Line Engineering, PLE)은 소프트웨어 재사용을 위한 새로운 접근 방법이고, 핵심자산은 인스턴시에이션(Instantiation)을 통하여 어플리케이션을 개발하기 위한 큰 재사용 단위이다. 따라서, 핵심자산은 PLE의 중요한 요소이므로 핵심자산의 재사용성은 PLE 프로젝트의 큰 성공을 결정한다. 핵심자산은 전체가 아니라 재사용 부품(Part)에 불과하며, 고정된 기능뿐만 아니라 가변적인 기능도 포함, 지원하고 있다. 그러나, 기존의 품질모델로는 이러한 특성을 가지고 있는 핵심자산을 평가하기에는 한계가 있다. 따라서, 본 논문에서는 이러한 문제점을 해결하기 위해 ISO/IEC 9126을 기반으로 하여 핵심자산의 재사용성을 평가하기 위한 체계적인 품질시스템을 제안한다. 핵심자산의 중요한 특징들을 식별하고 식별된 재사용성의 특징을 기반으로 품질속성(Quality Attribute)을 도출한다. 이렇게 정의된 품질속성을 이용하여 메트릭을 정의한다. 또한 제안된 메트릭을 이용하기 위한 지침 및 대여 프로덕트 라인에서의 적용사례를 제시한다. 이러한 품질시스템을 이용하여 핵심자산의 재사용성을 보다 효과적이고 정확하게 평가한다.

키워드 : 프로덕트 라인, 재사용성, 품질시스템, 품질속성, 메트릭

Abstract Product line engineering (PLE) is a new effective approach to software reuse, where applications are generated by instantiating a core asset which is a large-grained reuse unit. Hence, a core asset is a key element of PLE, and therefore the reusability of the core asset largely determines the success of PLE projects. A core asset is a reusable part not a whole system, and supports not only variable functions but also common functions. However, there are limitations to evaluate reusability of core asset that has these unique characteristics. This paper proposes a comprehensive quality system for evaluating the reusability of core assets, based on ISO/IEC 9126. We first identify the key characteristics of core assets, and derive the set of quality attributes that characterizes the reusability of core assets. Finally, we define metrics for each quality attribute. In addition, we provide guidelines for applying the metrics and perform a case study based on rental product line. Using the proposed quality system, reusability of core assets can be more effectively and correctly evaluated.

Key words : Product Line, Reusability, Quality System, Quality Attribute, Metric

1. 서 론

PLE는 최근 각광받고 있는 효과적인 재사용 방법으로, 도메인 공학(Domain Engineering, DE)과 어플리케이션 공학(Application Engineering, AE)으로 구성되어 있다[1]. DE 과정은 목표 도메인의 공통적인 특징과 가변적인 특징을 정의하고, 패밀리 멤버들 사이에서 추출된 공통성과 가변성을 통해 핵심자산을 개발한다. AE 과정은 어플리케이션 요구사항 분석, 의사결정해소 모델(Decision Resolution Model, DRM)의 정의, 그리고 DRM을 준수한 핵심자산의 특화, 마지막으로 목표 제품

· 본 연구는 한국과학재단 특정기초연구(R01-2005-000-11215-0)지원으로 수행되었음
† 학생회원 : 숭실대학교 컴퓨터학과
shoh@otlab.ssu.ac.kr
jsher@otlab.ssu.ac.kr
jhkim@otlab.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
syrhew@ssu.ac.kr
sdkim@ssu.ac.kr
논문접수 : 2005년 8월 31일
심사완료 : 2006년 1월 5일

을 생산하는 활동 등으로 이루어져 있다[2].

여러 제품에 공통적으로 재사용될 수 있는 핵심자산은 PLE의 기본적인 요소이다. 핵심자산의 재사용성은 프로덕트 라인 전체의 성공에 중요한 역할을 한다. 그러나, 제품평가를 위한 표준인 ISO/IEC 9126으로 핵심자산의 재사용성을 평가하기에는 미흡하다[3,4]. 왜냐하면, ISO/IEC 9126은 완성된 단일 어플리케이션을 위한 표준이기 때문이다. 또한 현재 대부분의 연구는 객체지향 시스템이나 컴포넌트의 재사용성을 평가하기 위한 수준에만 머물러 있는 상태이다[5]. 그러나, 핵심자산은 아키텍처 수준 크기이고, 재사용될 수 있는 미완성부품으로 여러 어플리케이션 개발의 기본이 된다. 이와 같이, 본 논문에서는 표준 품질모델인 ISO/IEC 9126을 기반으로 핵심자산의 재사용성을 평가하기 위한 새로운 품질시스템을 제안한다. 본 논문의 품질시스템은 품질모델을 도출하기 위한 과정, 제안된 품질모델, 품질모델 내의 메트릭을 적용하기 위한 지침을 포함한다.

2. 관련연구

2.1 SEI의 연구

SEI(Software Engineering Institute)의 연구에서는 프로덕트 라인의 관리를 지원하기 위한 측정(Measure)을 제안하고, 소프트웨어 프로덕트 라인 내부 측정 활동 설립을 위한 가이드 라인을 제시한다[6]. 이 연구는 관리적인 측면에 대한 측정에 대해서 다양한 관리적 룰을 제시하였다. 즉, 프로덕트 라인 관리, 핵심자산 개발 관리, 제품 개발 관리이다. 이 세 가지의 관리적 룰은 소프트웨어 프로덕트 라인을 보다 신속하고 정확하게 개발하기 위함이다. 하지만, 이 연구는 핵심자산의 재사용성을 평가, 측정하기 위한 품질속성이나 실용적인 메트릭의 정의가 부족하다.

2.2 IESE의 연구

IESE(Fraunhofer Institute for Experimental Software Engineering)의 연구는 프로덕트 라인의 품질모델 개발을 위한 배경이 되는 구성 요소를 제공한다[7]. 이 연구에서는 프로덕트 라인 모델의 전반적인 개발에 대한 가이드라인을 제시한다. 이 연구에서는 프로덕트 라인 개발을 위한 프로세스 방향과 프로덕트 라인 품질모델의 일반적인 구조에 대해서 논의한다. 하지만, 이 연구는 핵심자산을 평가하기 위한 품질모델이 아닌 품질모델을 정의하기 위한 공통 프레임워크를 제공하고 있어 실용적으로 활용되기에는 한계가 있다. IESE의 공통 프레임워크에서 언급한 이해성, 변경성, 적응성에 대한 품질속성이 재사용성 평가를 위해 활용될 수 있으나, 핵심자산의 재사용성 평가를 위한 세부 품질속성에 대한 명시적인 제안은 언급되어 있지 않다.

2.3 Washizaki의 연구

Washizaki의 연구는 컴포넌트의 재사용성을 측정하기 위한 메트릭을 제안한다[8]. 또한 컴포넌트의 이해성과 적응성, 이동성의 측정을 위한 메트릭을 정의한다. 그러나, 이 연구는 컴포넌트 재사용측면 측정에 대해서는 잘 정의하였지만, 컴포넌트는 핵심자산의 일부분이기 때문에 핵심자산의 재사용측면을 측정하기에는 미흡하다.

3. 기반연구

3.1 핵심자산의 메타모델

핵심자산의 메타모델은 핵심자산의 구성요소를 보여주고 있으며, 그림 1과 같이 범용 아키텍처, 컴포넌트 모델, 의사결정 모델로 구성되어 있다[9].

범용 아키텍처(Generic Architecture)란 프로덕트 라인 내의 여러 어플리케이션에서 재사용될 수 있는 소프

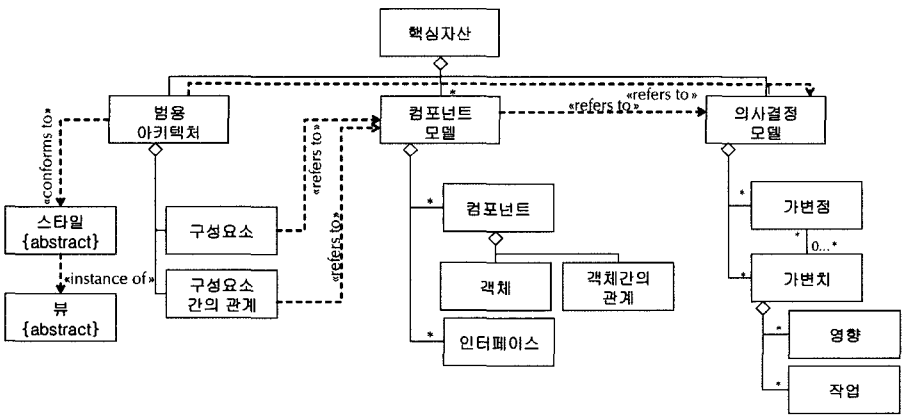

그림 1 핵심자산의 메타모델

트웨어의 아키텍처를 말한다. 범용 아키텍처에 적용되는 뷰(View)는 한 개 이상의 스타일로 표현할 수 있다. 범용 아키텍처는 구성요소와 구성요소간의 관계로 이루어져 있다. 구성요소는 컴포넌트 모델내의 컴포넌트를 참조하여 구성하고, 구성요소간의 관계는 컴포넌트 모델내의 provided 인터페이스와 required 인터페이스 사이의 관계를 참조하여 구성한다. 그리고 범용 아키텍처 내의 아키텍처 가변성(Architectural Variability)은 의사결정 모델에서 상세하게 명세된다. 아키텍처 가변성의 종류는 의사결정 모델에서 정의한다.

컴포넌트 모델(Component Model)은 컴포넌트와 인터페이스의 집합으로 구성된다. 컴포넌트 기반 개발에서의 컴포넌트는 실행 수준에서의 연관된 기능으로 이루어져 독립적인 배치가 가능한 기능 단위를 의미하지만, PLE에서의 컴포넌트는 핵심자산 컴포넌트를 지칭하고 있으며 핵심자산 컴포넌트는 연관된 기능으로 이루어져 객체보다 큰 기능 단위로써 객체와 객체간의 연관관계, 인터페이스로 구성된다. 인터페이스는 컴포넌트 외부로 보여지는 기능을 나타내며 객체와 객체간의 연관관계를 통해 기능이 구현된다.

의사결정 모델(Decision Model)은 핵심자산의 재사용을 위한 가장 중요한 요소로써 PLE에서 어플리케이션들간의 가변성을 표현하기 위한 모델이다. 의사결정 모델에서는 가변점(Variation Point), 가변치(Variant), 영향(Effect), 작업(Task) 등을 통해 가변성을 표현한다. 가변점은 가변성이 발생할 수 있는 지점을 말한다. 가변치는 가변점에 적용될 수 있는 멤버들의 값들을 표현한다. 영향은 가변치 설정 후 핵심자산이 갖추어야 할 사후조건(Post Condition)을 의미한다. 작업은 가변점에 가변치를 설정하기 위해 모델 수준에서 해야 할 행동들을 열거하며, 각 가변치마다 다르게 나타난다.

3.2 ISO/IEC 9126 품질모델

ISO/IEC 9126은 단일 소프트웨어를 평가하기 위한 표준 품질모델이다. 품질모델은 특성, 부특성 및 메트릭으로 구성된다. 또한 ISO/IEC 9126은 소프트웨어의 품질을 평가하기 위한 6개의 특성과 26개의 부특성과 내·외부 및 사용중 메트릭을 정의하고 있다[3,4].

기능성(*Functionality*)은 소프트웨어가 특정 조건에서 사용될 때, 명시된 요구와 내재된 요구를 만족하는 기능을 제공하는 소프트웨어 제품의 능력을 의미한다. 신뢰성(*Reliability*)은 명세된 조건에서 사용될 때, 성능 수준을 유지할 수 있는 소프트웨어 제품의 능력을 의미한다. 사용성(*Usability*)은 명시된 조건에서 사용될 경우, 사용자에 의해 이해되고, 학습되고, 선호될 수 있는 소프트웨어 제품의 능력을 의미한다. 효율성(*Efficiency*)은 명시된 조건에서 사용되는 자원의 양에 따라 요구된 성능을 제공하는 소프트웨어 제품의 능력을 의미한다. 유지보수성(*Maintainability*)은 변경되는 환경에서 소프트웨어의 개량과 적용 그리고 요구사항의 기능적 명세서 수정 등을 포함된다. 이식성(*Portability*)은 한 환경에서 다른 환경으로 전이될 수 있는 소프트웨어 제품의 능력을 의미한다. 여기서 환경은 조직, 하드웨어 혹은 소프트웨어 환경을 포함한다.

메트릭(Metric)에는 외부적인 측면과 내부적인 측면이 있다. 내부 메트릭은 설계나 코딩 도중에 실행할 수 없는 소프트웨어 제품(명세서나 원시 코드와 같은)에 적용할 수 있다. 이러한 내부 메트릭의 목적은 요구된 외부 품질이 성취되었는가를 확인하는 것이다. 외부 메트릭은 실행 가능한 소프트웨어나 시스템을 시험, 운영 또는 관찰해 봄으로써 그 소프트웨어가 이루고 있는 시스템 행태에 대한 제품의 측정을 위해 사용한다. 소프트웨어 제품을 구매하거나 사용하기 전에 특정한 조직이나 기술적 환경에서 제품의 사용, 개발 및 관리와 관련된 기업 목적을 토대로 외부 메트릭을 사용하여 평가해야 한다.

4. 핵심자산의 특징

본 장에서는 핵심자산의 중요한 특징을 식별하고, 이러한 특징을 바탕으로 핵심자산의 재사용성을 평가할 수 있는 품질모델을 추출하기 위한 기반을 마련한다. 단일 어플리케이션과 구별되는 핵심자산의 중요한 특징은 7가지이며 각각의 정의는 다음과 같다[1,2,10-13].

공통기능성 제공: 핵심자산은 제품간에 재사용할 수 있는 공통된 기능성을 제공한다. 핵심자산이 제공할 공통 기능성의 범위는 프로덕트 라인 스코프(Product Line Scope)안에 정의되어 있다.

제품들 사이의 가변성 지원: 핵심자산은 단일 어플리케이션이 아닌 여러 어플리케이션을 위한 재사용 단위이기 때문에 제품들간의 가변성을 지원해야 한다. 이러한 가변성을 지원하는 대표적인 명세단위로 PLE에서는 의사결정 모델을 정의하고 있다. 의사결정 모델은 가변점, 가변치, 영향, 작업 등의 구성요소로 가변성을 명세한다. 만일 의사결정 모델과 같은 명세단위가 존재하지 않으면 효과적으로 가변성을 적용하기 어렵다.

범용 아키텍처 재사용: 핵심자산은 아키텍처로부터 영향 받을 수 있는 컴포넌트와 컴포넌트들간의 관계들로 이루어져 있다. 일반적으로 소프트웨어 아키텍처는 단일 어플리케이션 개발을 위해 사용되지만, 핵심자산의 범용 아키텍처는 여러 어플리케이션들을 개발하기 위해 사용된다. 또한 범용 아키텍처는 아키텍처 가변성을 포함한다. 따라서 PLE에서는 컴포넌트뿐만 아니라 아키텍처까지 재사용된다.

아키텍처 수준의 핵심자산: 핵심자산은 객체 혹은 컴포넌트 수준보다 크며, 완성된 하나의 어플리케이션과 거의 비슷한 크기이거나 약간 작다. 또한 핵심자산은 어플리케이션 수준의 크기이기 때문에 어플리케이션 종속적인 정보를 조금만 추가하여도 완성된 어플리케이션을 개발할 수 있다. 그러므로 어플리케이션을 개발하는 과정에서 노력을 최소한으로 줄일 수 있다.

Open과 Closed 가변성 지원: PLE에서 가변성의 범위(scope)는 가변점 당 가질 수 있는 후보 가변치의 개수를 의미하며, Open 가변성과 Closed 가변성으로 분류된다[13]. Closed 가변성은 가변치가 알려져 있는 가변성을 말하며, Open 가변성은 가변치가 알려지지 않은 가변성을 말한다. Closed 가변성은 가변치가 고정되어 있어야 하는 반면에, Open 가변성은 가변치를 미리 설정하지 않기 때문에 잠재적인 어플리케이션을 생산할 수 있는 가능성이 된다.

인스턴시에이션 메커니즘: 가변성을 설정하는 메커니즘으로 PLE에서는 인스턴시에이션을 하기 위한 메커니즘을 명시적으로 정의하고 있다. 핵심자산이 가지는 가변성에 가변치를 설정하므로, 어플리케이션 종속적인 자산이 산출된다. 이러한 과정을 인스턴시에이션(Instantiation)이라 부른다. 인스턴시에이션의 주요 활동은 의사결정 모델을 정의하고, 각 가변점에 가변치를 설정하고 바인딩하는 것이다.

컴포넌트 내장: 핵심자산의 한 부분인 컴포넌트는 컴포넌트를 대신할 수 있는 COTS 컴포넌트들을 포함할 수 있다. 컴포넌트는 표준 인터페이스를 따르기 때문에, 핵심 자산 내에 있는 컴포넌트의 기능과 COTS 컴포넌트의 기능이 일치하고 기존 컴포넌트가 참조모델을 준수했을 경우에는 핵심자산 내부에 설계된 컴포넌트는 대체할 수 있다. 범용 아키텍처를 통하여 컴포넌트의 관계와 컴포넌트 결합을 위한 정보를 획득할 수 있다.

5. 재사용성 평가를 위한 품질속성

이 장에서는 핵심자산의 재사용성을 평가하기 위한 품질속성을 정의한다. ISO/IEC 9126으로부터 도출된 2개의 품질속성과 4장에서 논의된 특성들을 반영하여 5개의 품질속성을 새로이 정의한다. 그림 2에서는 재사용성을 평가하기 위한 품질속성과 핵심자산의 특징들간의 맵핑관계를 볼 수 있다. 이해성, 컴포넌트 대체성은 ISO/IEC 9126으로부터 도출 변경되고, 기능공통성, 적용성, 비기능공통성, 가변성충족도, 특화성은 새롭게 정의한다. 또한 각 품질속성을 도출한 논리적인 근거를 제시하고, 각 품질 속성들에 대해 정의를 내린다.

5.1 ISO/IEC 9126로부터 채택된 품질속성

이해성(Understandability): 이 품질속성은 핵심자산기반의 어플리케이션을 개발할 때, 쉽게 이해할 수 있는 정도를 측정한다. 핵심자산의 명세서를 (명세서에는 기능성, 인터페이스, 의사결정 모델 등을 표현한다.) 쉽게 이해하고 적용할 수 있다면 정확하고 신속한 방법으로 개발할 수 있다. 그러나, 이해성이 낮다면 핵심자산을 재사용하는데 많은 어려움이 따를 것이다.

재사용성 평가를 위한 품질속성으로 이해성을 도출한 이유는 핵심자산의 명세서와 인스턴시에이션을 위한 메커니즘 등을 핵심자산을 통해서 이해할 수 있어야 하기 때문이다. 다른 이유로는 어플리케이션 개발을 위한 핵심자산의 재사용에서 불필요한 노력을 줄이기 위함이다.

컴포넌트대체성(Component Replaceability): 이 품질속성은 범용 아키텍처의 변경 없이 기존 컴포넌트를 다른 in-house나 COTS 컴포넌트로 대체 가능한 정도를 측정한다. 핵심자산 내부의 컴포넌트가 다른 컴포넌트로 대체될 때, 범용 아키텍처가 수정이 된다면, 핵심

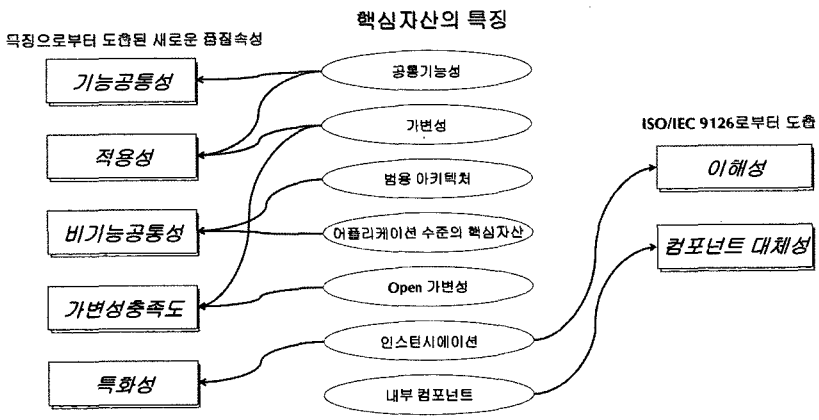

그림 2 재사용성을 위한 핵심자산과 속성의 특성들간에 맵핑

자산의 디자인이 변경될 것이고 핵심 자산을 인스턴시에이션할 때 예상치 못한 결과가 발생할 수 있다.

컴포넌트대체성은 ISO/IEC 9126의 대체성을 특화한 품질속성으로 이를 도출한 이유는 핵심자산 내부의 컴포넌트가 다른 컴포넌트로 대체가 가능하기 때문이다. 다른 컴포넌트들이 핵심자산 내부의 컴포넌트들과 같은 기능을 지원한다면, 컴포넌트를 직접 개발했을 때 보다 다른 컴포넌트를 사용했을 때에 개발 비용이나 Time to Market을 줄일 수 있다.

5.2 제안된 품질속성

기능공통성(Functional Commonality): 이 품질속성은 핵심자산이 프로덕트 라인 스코프에서 잠재적으로 개발될 수 있는 어플리케이션을 위해 공통된 기능을 지원하는 정도를 측정한다. 이러한 공통된 기능들은 어플리케이션 개발에 재사용된다.

기능공통성을 정의한 이유는 핵심자산 요구사항 명세서로부터 잠재적인 어플리케이션 개발을 위해 공통된 기능을 핵심자산이 지원해야 하기 때문이다. 비록 핵심자산이 고급 기능을 지원하더라도 핵심자산이 어플리케이션들에 공통적으로 재사용될 수 없다면 재사용하는데 한계가 있다.

적용성(Applicability): 이 품질속성은 핵심자산이 패밀리 멤버들에게 적용될 수 있는 정도를 측정한다. 핵심자산을 얼마나 많은 어플리케이션에서 재사용하는가에 따라 핵심자산의 경제적 가치를 결정하기 때문이다. 그러므로 프로덕트 라인 스코프 내에 정의되어 있는 잠재적인 어플리케이션들의 수는 적용성 측정에 영향을 미칠 수 있다.

적용성을 정의한 이유는 핵심자산의 주요한 목적이 재사용이기 때문이다. 즉, 핵심자산은 공통성과 가변성을 정의하고 어플리케이션의 적용을 고려한 어플리케이션 수준의 재사용 단위이기 때문이다.

비기능공통성(Non-functional Commonality): 이 품질속성은 핵심자산에서 제공하는 비기능적 요구사항이 프로덕트 라인내의 멤버들에게 얼마나 공통적인가의 정도를 측정한다. 비기능적 요구사항 중 상당 부분은 범용 아키텍처를 개발하기 위한 중요한 요구사항이 된다. 따라서, 비기능적 요구사항이 멤버들에게 공통적이면 범용 아키텍처는 패밀리 멤버들에 의해 공유될 수 있다.

비기능공통성을 정의한 이유는 다른 재사용 접근 방법과는 다르게 PLE가 제공하는 구별되는 특징으로 아키텍처 재사용이 있기 때문이다. 그러므로, 핵심자산의 비기능공통성을 측정함으로써, 핵심자산에서 제공하는 아키텍처의 재사용 정도를 측정할 수 있다.

가변성충족도(Variability Richness): 이 품질속성은 핵심자산이 도메인 내의 가변성을 얼마나 수용하고 있는지의 정도를 측정한다. 핵심자산에서 가변성을 충분하게 제공한다면, 더욱 많은 수의 어플리케이션에 의해 재사용될 수 있을 것이다. 또한 반대로, 제한된 가변성만 제공한다면, 제한된 수의 어플리케이션만 재사용할 수 있을 것이다.

가변성충족도를 정의한 이유는 핵심자산이 많은 수의 가변성을 포괄적으로 포함하고 있다면, 핵심자산은 더 많은 어플리케이션 개발을 위해 재사용될 수 있기 때문이다.

특화성(Tailorability): 이 품질속성은 어플리케이션 종속적인 요구사항에 맞게 효과적이고 용이하게 특화될 수 있는 정도를 측정한다. 핵심자산을 특화하기 위해서 DRM, 가변치 설정 등을 포함하는 인스턴시에이션 메커니즘을 사용한다. 그러므로 이 품질속성은 인스턴시에이션 시에 얼마나 효과적으로 DRM을 정의할 수 있으며, 선택되고 정의된 가변치에 의해 얼마나 적절하게 핵심자산이 특화되는지를 측정한다.

특화성을 정의한 이유는 각 어플리케이션에서의 재사용을 위해 핵심자산을 특화할 수 있는 성질은 핵심자산이 갖는 고유한 특징이기 때문이다. 핵심자산이 뛰어난 기능을 제공하여도, 핵심자산을 특화하기 위한 메커니즘이 효과적이지 않다면 재사용하는데 많은 어려움이 있을 것이다.

5.3 품질속성들 간의 연관관계

비록 각각의 품질속성은 평가 대상인 핵심자산의 여러 측면 중 한가지 독립적인 측면에 초점을 맞추고 있지만, 그림 3과 같이 그들 간에 포함관계가 발생한다. 적용성은 세 개의 품질속성, *기능공통성, 비기능공통성, 가변성충족도*로부터 유도된다.

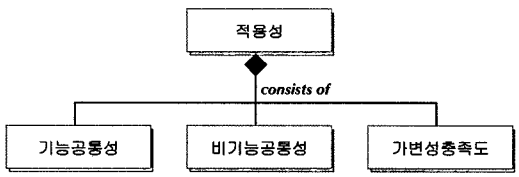

그림 3 품질속성들 간에 의존관계

6. 재사용성을 평가하기 위한 메트릭

본 장에서는 5장에서 정의한 품질속성별 평가 메트릭을 제안한다. 각 메트릭에 대해 설명, 공식, 값의 범위, 관련된 해석 등의 항목을 제시한다. 그림 4는 각 품질속성을 측정하기 위한 메트릭을 보여주고 있다.

6.1 기능공통성

핵심자산의 기능공통성은 ***Functional Coverage (FC)*** 를 통해 측정한다. 이 메트릭은 핵심자산이 멤버들 사이

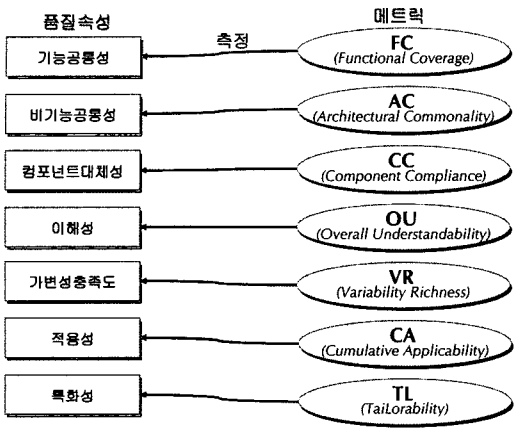

그림 4 품질속성과 메트릭들 간에 맵핑

에서 공통적으로 사용할 수 있는 기능적 휘처(Feature)들을 얼마나 많이 제공하는지 측정한다. 이 메트릭은 다음과 같이 계산된다.

FC = *(공통적인 기능적 휘처들의 수) / (전체 기능적 휘처들의 수)*

휘처의 공통성은 관련된 도메인과 프로젝트의 제약사항을 고려하여 품질 평가자로부터 판단된다.

그러므로, FC의 범위는 0..1이다. FC의 범위가 낮으면 기능공통성이 낮음을 의미하고, 멤버들 사이에서 핵심자산의 기능적 휘처들을 모두 사용할 수 있을 경우 FC는 1의 값을 가진다.

일반적으로, FC의 값이 크다면 핵심자산이 보다 폭넓게 적용될 수 있다. 즉, 많은 수의 잠재적인 어플리케이션들을 개발할 수 있다.

6.2 비기능공통성

일반적으로 비기능적 요구사항을 구현하는 방법에는 여러 방법이 있지만 핵심적인 수단으로 아키텍처를 통한 실현이 있다. 핵심자산 또한 범용 아키텍처를 통해 대부분의 공통 비기능적 요구사항을 반영할 수 있다[15,16]. 그러나, 아키텍처에 의해 모든 비기능적 요구사항을 다 해결할 수는 없다. 이러한 일부의 비기능적 요구사항에 대한 해결 방법은 여전히 풀리지 않는 문제로 남아있다[17]. 따라서 본 논문에서는 이 문제점에 대해서 다루지 않는다.

핵심자산의 비기능공통성은 ***Architectural Commonality(AC)***를 통해 측정한다. 이 메트릭은 핵심자산에서 제공되는 아키텍처가 얼마나 공통적으로 사용될 수 있는지를 측정한다. 이 메트릭은 다음과 같이 계산된다.

AC = *(범용 아키텍처를 공유하는 어플리케이션들의 수) / (프로덕트 라인내의 잠재적인 전체 어플리케이션들의 수)*

분모는 프로덕트 라인 스코프에서 고려된 어플리케이션들의 수이다.

그러므로, AC의 범위는 0..1이다. AC가 높은 값을 가지면 아키텍처의 공통성이 높고, 아키텍처가 모든 멤버들에게 공유될 수 있다면 AC는 1의 값을 가진다.

일반적으로 핵심자산 AC의 값이 1에 가까우면, 적용성이 높은 경향이 있으며 범용 아키텍처를 어플리케이션 개발에 많이 적용할 수 있다.

6.3 컴포넌트대체성

핵심자산의 컴포넌트대체성은 ***Component Compliance(CC)***를 통해 측정한다. 이 메트릭은 핵심자산의 내부 컴포넌트가 문제없이 새로이 업데이트된 컴포넌트로 대체될 수 있는 정도를 측정한다. 이 메트릭은 다음과 같이 계산된다.

CC = *(대체 가능한 컴포넌트의 수) / (핵심자산 내의 전체 컴포넌트들의 수)*

컴포넌트의 대체성은 기능적 준수성과 인터페이스 준수성을 바탕으로 판단한다.

새로운 컴포넌트가 이미 존재하는 컴포넌트의 기능과 똑같은 기능을 제공한다면 새로운 컴포넌트는 기존의 컴포넌트와 대체될 수 있다. 인터페이스 적합성을 결정할 때에는 타입구조, 강제 형변환, 객체 대체성, 다른 타입과의 적합성 등을 고려한다.

그러므로, CC의 범위는 0..1이다. CC의 값이 1이면 모든 컴포넌트는 문제없이 대체될 수 있다. 핵심자산의 CC가 높으면 효과적으로 진화(Evolution)될 수 있을 것이다.

6.4 이해성

핵심자산의 이해성은 ***Overall Understandability (OU)***를 통해 측정한다. 이 메트릭은 사용자가 핵심자산에 대해서 얼마나 용이하고, 능률적이며 정확하게 이해할 수 있는지를 측정한다. 핵심자산 설명서에는 명세서, 사용자 매뉴얼, 다른 핵심자산을 설명해 놓은 문서들이 포함되어 있다. 이 메트릭은 다음과 같이 계산된다.

OU = *(이해할 수 있는 요소들의 수) / (전체 요소들의 수)*

요소들(Element)은 핵심자산 설명서를 구성하는 대표적인 아이템들이며, 이 요소들의 이해를 위해 여러 가지 관점에 의해 표현된다. 예를 들어, 이러한 관점에는 기능 휘처, 범용아키텍처, 결정모델, 인스턴시에이션 메커니즘 등이 있다. 사용자에 의해 용이하고, 효과적이며, 정확하게 이해될 수 있도록 요소가 표현되어 있다면 그 요소는 이해할 수 있는 요소이다. 그러나, 이해할 수 있는 정도는 도메인과 관련된 프로젝트의 특성을 고려하여 품질 평가자가 결정한다.

그러므로 OU의 범위는 0..1이다. OU의 값이 낮다면 핵심자산을 이해하는데 많은 어려움이 있고, OU의 값이 1에 가깝다면 어려움 없이 핵심자산을 용이하게 이해할 수 있다.

6.5 가변성충족도

가변성충족도는 두 개의 메트릭 ***Coverage of Variability(CV)***와 ***Realization of Variability(RV)***를 통해 측정된다. CV 메트릭은 SRSs(Software Requirement Specifications)에서 식별된 가변성들이 얼마나 많이 프로덕트 라인 스코프 안에 포함되었는지를 측정한다. 이 메트릭은 다음과 같이 계산된다.

CV = *(프로덕트 라인 스코프내에 포함된 가변점의 수) / (명시적이고 잠재적인 가변점의 수)*

분자는 SRSs로부터 식별된 가변점들 중에서 프로덕트 라인 스코프에서 포함하고 있는 가변점들의 수이다. 경제적인 가치를 고려하여 일부 가변점은 프로덕트 라인 스코프에 포함되지 않을 수도 있음에 유의해야 한다.

그러므로 CV의 범위는 0..1이다. CV의 값이 크면 핵심자산은 포괄적이고 많은 가변점들을 포함하게 된다.

가변성충족도를 위한 두 번째 메트릭 RV는 프로덕트 라인 스코프 내에 정의된 가변점들 중 얼마나 많은 가변점들이 핵심자산에 의해 실제적으로 구현되는지의 정도를 측정한다.

RV = *(핵심자산에서 구현한 가변점들의 수) / (프로덕트 라인 스코프에서 포함하고 있는 가변점의 수)*

그러므로 RV의 범위는 0..1이다. RV의 값이 크면 핵심자산은 프로덕트 라인 스코프 내의 가변점들 중 많은 수의 가변점을 실제 구현한 핵심자산임을 의미한다.

마지막으로, *Variability Richness(VR)*의 값은 두 개의 메트릭(CV & RV)으로부터 유도된다.

가변성충족도(Variability Richness), VR = *CV * RV*

그러므로 VR의 값의 범위는 0..1이다. VR의 값이 1을 가지면, SRSs가 가진 명시적이고 묵시적인 모든 가변점들을 핵심자산에서 구현하고 있음을 지시한다.

일반적으로 핵심자산의 VR값이 높게 측정되면 다양한 어플리케이션들이 폭넓게 개발될 수 있음을 의미한다.

6.6 적용성

핵심자산의 적용성은 ***Cumulative Applicability(CA)***를 통해 측정된다. 이 메트릭은 핵심자산에 의해 얼마나 많은 수의 어플리케이션을 개발할 수 있는지에 대한 정도를 측정한다. 5.3절에서 기술되어 있는 것처럼 CA는 다른 세 개의 메트릭(FC, AC, VR)으로부터 유도된다.

메트릭과 관련된 가중치는 적용되는 도메인 및 프로젝트에 따라 달라질 수 있다. 그러므로, 세 개의 메트릭을 위한 별도의 가중치를 정의하고 각각의 값은 프로젝트의 특징 등을 고려하여 품질 평가자에 의해 결정된다. 이 CA 메트릭은 다음과 같이 각 메트릭(FA, AC, VR)을 이용하여 CA값을 계산한다.

$$CA = W_{FC} * FC + W_{AC} * AC + W_{VR} * VR$$

W_{FC}, W_{AC}, W_{VR}는 각 메트릭을 위한 가중치이며 이 세 값의 합은 1이다.

그러므로, CA의 범위는 0..1이고 CA의 값이 높게 측정된다면 관련된 도메인 내에 많은 잠재적인 어플리케이션들이 핵심자산을 적용할 수 있음을 의미한다.

6.7 특화성

특화성은 얼마나 많은 가변점들이 효과적으로 특화 가능한지, 그리고 얼마나 많은 가변점들이 특화된 후에 유효한지를 측정한다. 여기서 유효하다는 것의 의미는 부작용(Side Effect)이나 결점(Fault) 없이 가변점이 가변치에 의해 설정되는 것을 의미한다.

첫 번째, 효과적으로 특화될 수 있는 가변점의 측정은 ***Effectiveness of Tailoring(ET)*** 메트릭으로 측정할 수 있다. 이 메트릭은 얼마나 많은 가변점들을 정확하고 효과적으로 해결(Resolve)할 수 있는지에 대해 측정한다.

ET = *(효과적으로 해결할 수 있는 가변점의 수) / (전체 가변점의 수)*

효과적으로 해결할 수 있는 가변점이란 가변점의 제약사항을 고려하여 인스턴시에이션하기 위한 메커니즘을 효과적이고 적절하게 정의한 가변점들을 의미한다.

그러나 효과적으로 해결할 수 있는 가변점을 정량적으로 측정하기는 어렵다. 따라서, 각 가변점이 효과적으로 해결 가능한지의 여부를 결정하기 위해 체크리스트(Check List) 사용을 권장한다. 체크리스트는 Binary, Option, Alternative, Open 등과 같은 가변성의 타입이나 스코프 등을 고려하여 정의한다. 예를 들어, 품질 평가자는 Open 가변성을 위한 Plug-in 메커니즘이 객체 전달을 하기에 적절하게 정의되어 있는지 조사해야 한다.

두 번째로, 얼마나 많은 가변점이 정확하고 유효하게 특화되었는지를 평가하기 위해 ***Tailorability of Closed variability(TC)***와 ***Tailorability of Open variability(TO)*** 메트릭을 사용한다.

TC 메트릭은 DRM에 정의된 가변점 중 Closed 가변점에 대해 부작용이나 결점의 발생 없이 해결할 수 있는 정도를 측정한다. Closed 가변점의 경우, 모든 가변치는 알 수 있고 각 가변치에 따라 다른 결과를 발생시킬 수 있다. 그러므로, 각 가변점에 대해 발생할 수 있는 모든 바인딩의 수를 고려하여 TC를 다음과 같이 계산한다.

$$TC = \frac{\sum_{i=1}^{n} (VP_i \text{에 대한 유효한 가변치의 수})}{\sum_{i=1}^{n} (VP_i \text{에 대한 전체 가변치의 수})}$$

VP_i는 i 번째 Closed 가변점이고, 유효한 가변치는 부작용이나 결점 없이 해결(즉, 바인딩)할 수 있는 가변치를 말한다.

TC의 범위는 0..1이다. TC의 값이 크면 많은 수의

Closed 가변점이 문제없이 해결될 수 있다.

TO 메트릭 측정은 DRM에 정의된 가변점 중 Open 가변점에 대해 부작용이나 결점의 발생 없이 해결할 수 있는 정도를 측정한다. Open 가변점의 경우, 가변치를 알 수 없다. 따라서 다음과 같이 계산된다.

$$TO = (\textit{유효한 Open 가변점의 수}) / (\textit{전체 Open 가변점의 수})$$

TO의 범위는 0..1이다. TO의 값이 크면 많은 수의 Open 가변점을 유효하게 해결할 수 있음을 의미한다.

본 논문에서는 디자인 모델 수준의 핵심자산을 평가하고 있기 때문에, 유효한 가변치와 가변점들은 리뷰(Review)나 인스펙션(Inspection)을 통해 정성적으로 평가한다. 그러나, 구현수준의 핵심자산을 평가할 경우에는 정량적 측정이 가능하다.

앞에서 나온 메트릭들을 사용하여, 마지막으로 ***Tailorability(TL)***를 측정한다.

$$TL = W_{ET} * ET + W_{TV} * (TC와\ TO의\ 평균)$$

TC와 TO의 평균은 유효한 가변점의 전체적인 비율이다. W_{ET}와 W_{TV}은 *Effectiveness of tailoring*과 *Tailorability of open and closed variability*의 가중치이다. 각 가중치의 값은 프로젝트의 특징을 고려하여 품질 평가자에 의해 결정되며 이 가중치들의 합은 1이다.

그러므로, TL의 범위는 0..1 이다. TL의 값이 크면 각 가변점을 특화하기 위한 메커니즘은 효과적으로 정의되어 있고 부작용이나 결점 없이 해결할 수 있음을 의미한다.

7. 사례연구

일반적으로, 품질모델은 개념적이기 때문에 실제 프로젝트에 적용하기가 어렵다. 이러한 품질모델의 한계를 극복하기 위해 본 장에서는 PLE 프로젝트에 품질모델을 적용하기 위한 실용적인 가이드라인을 제안한다. 또한 본 논문에서 제안한 품질모델의 실용성을 보여주기 위하여 제안된 품질모델을 *대여(Rental)* 프로덕트 라인의 핵심자산을 평가하는데 적용한다.

7.1 도메인 명세

대여도메인은 도서대여, 자동차대여, 비디오대여 등 여러 대여 서비스를 포함한다. 본 사례연구의 평가대상은 대여 도메인내의 대여 핵심자산으로써, 7개의 멤버들 사이의 공통성과 가변성을 분석하였으며 DREAM[14]을 기반으로 개발되었다. 대여 핵심자산으로부터 지원되는 주요 기능으로는 품목관리, 회원관리, 예약관리, 대여관리, 회계관리 등이 있다.

대여 프로덕트 라인 요구사항에 정의된 전체 기능들의 수는 36개이다. 대여 핵심자산은 33개의 기능을 제안하고 그 중 공통된 기능은 30개이며, 가변적인 기능은 11개이다. 또한 비기능적 휘처는 13개이고 7개의 컴포넌트들로 구성되어있다. 대여 핵심자산의 재사용성 평가를 위해 핵심자산 산출물을 이용하여 평가 메트릭의 값을 획득하였다.

7.2 기능공통성의 계산(FC)

6.1에 정의된 바와 같이, *FC*는 핵심자산에서 기능적 휘처가 공유되는 정도를 측정한다. 이 메트릭을 위한 측정치는 프로덕트 라인 스코프나 공통성 가변성(C&V)분석 모델들로부터 획득할 수 있다[18].

대여 핵심자산에서 제공하는 기능은 33개이며, 이 중 30개는 여러 멤버들에 의해 사용되는 공통적인 기능이다. 따라서 FC의 값은 30 / 33 = 0.91이 나오며, 이는 대여 핵심자산이 제공하는 기능성 중 91%가 7개의 멤버들에서 공통적으로 사용되는 기능임을 의미한다.

7.3 비기능공통성의 계산(AC)

6.2에 정의된 바와 같이, AC는 핵심자산에서 비기능적 휘처가 공유되는 정도를 측정한다. 분모의 값(프로덕트 라인내의 잠재적인 전체 어플리케이션들의 수)은 프로덕트 라인 스코프로부터 획득할 수 있다. 그리고 분자의 값(범용 아키텍처를 공유하는 어플리케이션들의 수)은 아키텍처 의사결정 모델과 같은 아키텍처 모델링 결과로부터 획득할 수 있다[19]. 즉, 아키텍처 의사결정 모델에서 가변치 집합으로부터 어플리케이션의 수를 추출할 수 있다.

대여 프로덕트 라인은 13개의 비기능적 휘처를 가지고 있다. 8개의 휘처는 아키텍처와 관련이 있고, 5개의 휘처는 아키텍처와 관련이 없어, 8개의 휘처를 근거로 아키텍처를 설계하였다. 따라서 아키텍처 의사결정 모델에서 8개의 휘처를 지원하는 가변성 모델링을 하였다. 그러나, 7개의 멤버 중 이들 8개의 휘처를 사용하는 멤버는 5개였다. 그러므로 AC의 값은 5 / 7 = 0.714이다. 따라서, 대여 핵심자산에서 비기능적 휘처는 7명의 멤버들 사이에서 71.4%의 공통성을 가진다.

7.4 컴포넌트대체성의 계산(CC)

6.3에서 정의된 바와 같이, CC는 핵심자산 내부에 정의된 컴포넌트의 대체 가능한 정도를 측정한다. 분모 값(핵심자산 내의 전체 컴포넌트들의 수)은 컴포넌트 클러스터링 결과인 컴포넌트 리스트를 통해 획득된다[20].

분자의 값(대체 가능한 컴포넌트의 수)은 핵심자산 내에 설계된 컴포넌트와 COTS 컴포넌트들을 비교해봄으로써 획득된다. 그러나, 모든 COTS 컴포넌트와 비교하는 것은 불가능하기 때문에 특정 기준을 통하여 컴포넌트들을 대체할 지 결정한다.

- 만약 목표 도메인의 컴포넌트 인터페이스를 위한 표준이 있다면 컴포넌트는 표준을 준수해야 한다.
- 만약 컴포넌트에 대한 표준이 없다면, 목표 컴포넌트와 다른 컴포넌트 사이의 의존성이 낮아야 한다.

• 인터페이스와 컴포넌트는 명백하게 구분되어야 한다.
• 컴포넌트 대체 후에 결함이 없거나 최소화되어야 한다.
• 컴포넌트의 명세서는 충분히 제공되어야 한다.

대여 프로덕트 라인에서 컴포넌트 클러스터링을 통하여 CInventory, CMembership, CRental 등 7개의 컴포넌트를 식별하였다. 위에서 제안된 체크리스트를 통하여 5개의 컴포넌트는 교체 가능함을 알 수 있었다. 그러므로 CC의 값은 5 / 7 = 0.714이다. 즉, 이는 대여 핵심자산의 컴포넌트 대체율은 71.4%임을 의미한다.

7.5 이해성의 계산(OU)

6.4절에서 정의된 바와 같이 OU는 핵심자산을 얼마나 쉽고, 효과적이며 정확하게 이해할 수 있는지에 대한 정도를 측정한다.

OU 메트릭을 위한 측정치는 핵심자산의 산출물로부터 획득된다. 핵심자산 산출물은 핵심자산 명세서, 범용 아키텍처 명세서, 컴포넌트 명세서, 인터페이스 명세서, 의사결정 모델과 다른 문서들이다. 분모의 값은 핵심자산 산출물의 요소로부터 획득된다. 분자의 값은 사용자가 이해할 수 있는 요소들의 수이다. 그리고, 이해할 수 있는 요소들을 판단하기 위한 기준은 다음과 같다;

• 각각의 요소들은 쉽고 간결하게 서술되어 있는가?
• 각각의 요소들의 구조는 쉽게 표현되어 있는가?
• 각각의 요소들 사이의 관계가 복잡한가?

대여 핵심자산에 대해, 이해할 수 있는 산출물의 수를 체크할 수 있는 테이블을 사용하였다.

테이블로부터 OU의 값은 0.862을 얻었다. 즉, 대여 핵심자산의 요소들 중 용이하게, 효과적이고 명확하게 이해할 수 있는 요소들이 86.2%이다.

7.6 가변성충족도의 계산(VR)

6.5에서 정의한 바와 같이, VR은 도메인 내에 존재하는 가변성을 핵심자산이 포함하는 정도를 측정하고, CV와 RV 메트릭을 사용하여 계산한다. CV 메트릭의 분모값(명시적이고 잠재적인 가변점들의 수)은 그림 5의 초기 C&V 명세서와 같은 초기 C&V 분석 결과로부터 획득할 수 있다. 초기 C&V 분석은 프로덕트 라인 스코핑 이전에 수행되고, 명시적이고 잠재적인 가변점은 그림 5

표 1 OU의 계산 테이블

산출물 이름	서브-섹션	이해할 수 있는 요소들(A)	요소들(B)	비율(A/B)
범용 아키텍처 명세서	Module Viewtype	9	12	0.75
	Component-and-Connector Viewtype	7	8	0.875
	Allocation Viewtype	8	8	1
컴포넌트 명세서	CInventory	37	40	0.925
	CRental	14	17	0.824
	...	...	...	...
...	...	...	...	...
결정 모델	VP_1	5	6	0.833
	...	...	...	...
Total		213	247	0.862

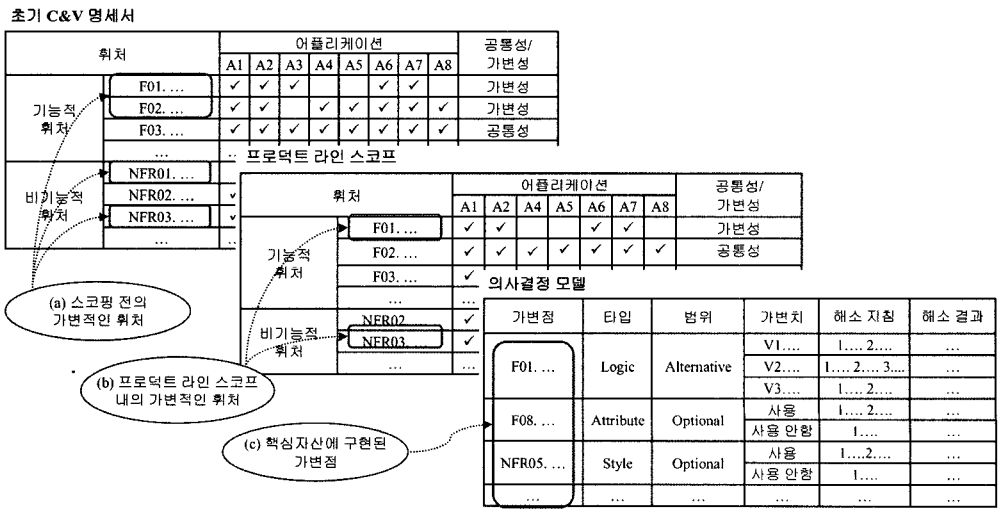

그림 5 VR 메트릭 계산을 위한 산출물

의 (a)와 같은 초기 C&V 명세서에 기술된 가변적인 휘처로부터 도출된 가변점들의 수이다. 분자값(프로덕트 라인 스코프에서 포함하고 있는 가변점들의 수)은 프로덕트 라인 스코프로부터 획득할 수 있다. 스코핑 과정 중에 스코프 외의 휘처는 제거된다. 따라서 제거된 휘처를 제외한 그림 5의 (b)와 같은 스코프 내의 휘처들로부터 도출된 가변점들의 수가 분자값이 된다.

RV 메트릭의 분모값(프로덕트 라인 스코프에서 포함하고 있는 가변점들의 수)은 CV 메트릭의 분자값과 동일하다. 그리고 분자값(핵심자산에서 구현한 가변점들의 수)은 그림 5의 (c)와 같은 의사결정 모델에 정의된 가변점의 수로부터 획득할 수 있다.

대여 프로덕트 라인에서 스코핑 전에 8개의 멤버들 사이에서 추출된 가변점의 수는 21개이다. 스코핑 활동 동안에 한 멤버가 삭제되었고, 이와 관련된 가변점 3개가 삭제되었다. 그리고 18개의 가변점 중에 16개의 가변점이 의사결정 모델에서 상세화되어 정의되었다. 2개의 가변점은 설계 시 충돌되어 삭제되었다. 그러므로 CV의 값은 18 / 21 = 0.857이고, RV의 값은 16 / 18 = 0.889이다. 마지막으로 VR은 0.857 * 0.889 = 0.762의 값을 가진다. 즉, 이는 대여 도메인 내에 존재하는 가변성의 76.2%를 대여 핵심자산에서 포함하고 있다는 것을 의미한다.

7.7 적용성의 계산(CA)

6.6에서 정의된 바와 같이 CA는 핵심자산이 개발될 어플리케이션에 얼마나 적용되는지를 측정하고, 이 메트릭은 FC, AC, VR의 메트릭을 사용하여 계산된다.

대여 프로덕트 라인에서 WFC, WAC, 그리고 WVR에 각각 0.4, 0.3 그리고 0.3의 가중치를 주었다. 그리고 위에서 계산된 FC, AC, VR 값으로부터, CA의 값은 0.91*0.4 + 0.71*0.3 + 0.76*0.3 = 0.81을 얻었다. 따라서, 대여 핵심자산이 패밀리 멤버들 사이에서 81% 적용 가능하다.

7.8 특화성의 계산(TL)

6.7에서 정의한 바와 같이, TL은 어플리케이션 요구사항에 맞게 핵심자산이 쉽고 정확하게 특화될 수 있는 능력을 측정하며, ET, TC, TO 메트릭을 이용하여 계산한다.

ET 메트릭의 분모값(전체 가변점의 수)은 그림 6의 (a)와 같은 의사결정 모델 내에 정의된 전체 가변점의 수로부터 획득한다. 그리고 분자값(효과적으로 해결할 수 있는 가변점의 수)은 각 가변점 당 가변성 메커니즘이 올바르게 설계되었는지를 확인하기 위한 체크리스트를 적용하여 평가함으로써 획득한다. 체크리스트에는 다음과 같은 사항이 있다[21].

- 가변치를 선택하기 위한 메커니즘으로 Optional, Binary, Alternative가 효과적인가?
- Open 가변성을 위한 plug-in 메커니즘이 객체를 전달하는데 효과적인가?
- 효과적으로 사용자의 결정을 전달하기 위한 외부 프로파일(External Profile)이 설계되었는가?
- 가변치를 구현하는 소프트웨어 개체들과 바인딩 타임을 적절히 고려하여 해소 지침이 정의되었는가?

TC 메트릭의 분모값은 각 가변점 당 가변치의 수의 총계로 획득할 수 있다. 예를 들어, 그림 6에서 (b1), (b2), (b3)를 더함으로써 구할 수 있다. 반면 분자값은 정량적으로 평가할 수 없어 각 가변치의 설계가 유효한지를 평가하기 위한 체크리스트를 적용한다. 체크리스트에는 다음과 같은 질문들이 포함된다.

- 각 가변치를 위한 특화 메커니즘이 효과적으로 정의되었는가?
- 해소 결과가 명확하게 정의되어 있는가?
- 해소 지침이 효과적으로 설계되어 있는가?
- 가변치나 가변점 사이에서 찾지 못한 의존관계가 있는가?

TO 메트릭의 분모값(전체 Open 가변점의 수)은 의사결정 모델로부터 획득할 수 있다. 즉, 그림 6에서 (c)와 같은 Open 가변점의 수를 더함으로써 구한다. 분자

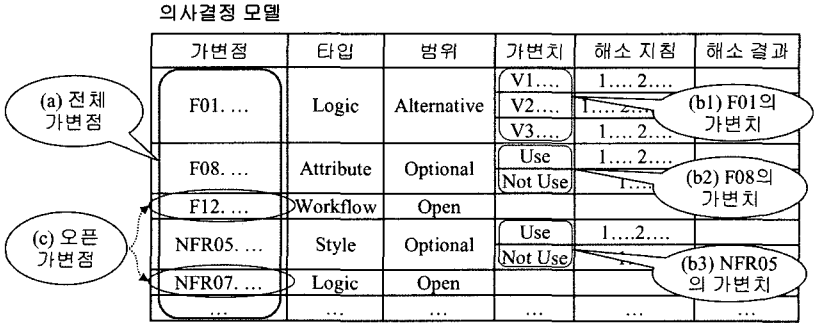

그림 6 의사결정 모델

값(유효한 Open 가변점의 수)은 다음과 같은 체크리스트를 적용함으로써 평가한다.

- Plug-in 명세서가 명확하게 정의되어 있는가?
- Plug-in 명세서에 명세된 사항을 구현하기 위한 기술이 존재하는가?
- 프로토콜이 유효한가?

대여 프로덕트 라인에서 의사결정 모델에 설계된 가변점의 전체 수는 16개이다. 16개의 가변점 중 11개는 Closed 가변점이고 5개는 Open 가변점이다. 11개의 가변점 당 정의된 가변치의 수를 모두 합치면 35개이다. 3명의 평가자들의 평가로부터, 2개의 가변점은 효과적으로 해소될 수 없고 6개의 가변치는 유효하지 않으며 한 개의 오픈 가변점이 유효하지 않음을 알 수 있었다. 따라서, ET 값은 14 / 16 = 0.875이고, TC 값은 27 / 33 = 0.818이고, TO 값은 4 / 5 = 0.8 이다. 그리고 TL의 값을 구하기 위해 WET와 WTV에 각각 0.5의 가중치를 준다. 즉, TL의 값은 0.5 * 0.875 + 0.5 * (0.818+0.8)/2 = 0.842이고, 이는 대여 핵심자산이 용이하고 정확하게 특화될 수 있는 정도가 84.2%임을 의미한다.

7.9 재사용성 계산

QA_i를 본 논문의 품질모델에 정의된 i번째 품질속성이라고 가정하자. 따라서 i의 범위는 1부터 7이 될 것이다. 품질속성과 관련된 중요도는 적용되는 프로젝트에 따라 달라질 수 있다. 따라서, 각 품질속성을 위해 가중치를 준다. QA_i와 관련된 가중치를 W_i라고 하고 7개 품질속성의 가중치를 모두 더한 값은 1이 된다. 핵심자산의 재사용성, ***REusability(RE)***은 다음과 같이 계산될 수 있고, RE 값의 범위는 0부터 1사이의 값이 되며, 높은 값일수록 재사용이 좋다는 것을 의미한다.

$$RE = \sum_{i=1}^{7} W_i \cdot QA_i$$

표 2는 본 논문에서 권장하는 각 품질속성 별 가중치로서 높음/낮음의 방법으로 표현하였다.

기능적공통성, 비기능적공통성, 가변성충족도, 적용성은 다른 품질속성들 보다 상대적으로 높은 가중치를 가진다. 왜냐하면, 핵심자산의 재사용성을 평가하기 위한 일차적인 요소가 되기 때문이다. 그리고 5.3절에서 논의한 바와 같이 적용성은 기능공통성, 비기능공통성, 가변성충족도로 구성된다. 반면에, 나머지 품질속성들은 이차적인 품질속성으로써 일차적인 요소들이 우선 만족되었을 경우에 재사용성에 영향을 미친다. 그러나 이러한 가중치는 고정된 것이 아니라, 본 논문에서 제안하는 품질모델을 적용하여 핵심자산의 재사용성을 평가할 경우, 각 프로젝트의 특성을 고려하여 품질 평가자가 다르게 가중치 값을 적용할 수 있다.

표 2 품질속성을 위한 가중치

품질속성	가중치(높음/낮음)
기능적공통성	높음
비기능적공통성	높음
컴포넌트대체성	낮음
이해성	낮음
가변성충족도	높음
적용성	높음
특화성	낮음

8. 결 론

핵심자산은 여러 제품들에 공통적으로 재사용될 수 있는 자산이기 때문에 PLE의 중요한 요소가 된다. 따라서 핵심자산의 재사용성은 PLE 프로젝트의 성공을 결정한다. 핵심자산은 전체가 아니라 재사용 부품에 불과하며, 고정된 기능뿐만 아니라 가변적인 기능도 포함, 지원하고 있다. 그러나, 기존의 품질모델로는 이러한 특성을 가지고 있는 핵심자산을 평가하기에는 한계가 있다. 따라서, ISO/IEC 9126과 현재의 연구들은 핵심자산의 재사용성을 정확하게 평가하기에는 부족하다.

그러므로, 본 논문에서는 PLE에서의 핵심자산의 특징을 고려하여 핵심자산의 재사용성을 평가하기 위하여 핵심자산의 중요한 특징 7가지를 식별하고, 식별된 특징을 기반으로 7 개의 품질속성을 도출하였다. 7 가지 품질속성 중 이해성, 컴포넌트대체성은 ISO/IEC 9126으로부터 도출되었으며 기능공통성, 비기능공통성, 가변성충족도, 적용성, 특화성은 새로이 제안되었다. 또한 이렇게 정의된 각 품질속성을 측정하기 위한 메트릭을 정의하였다. 총 7 개의 메트릭을 제안하였으며 각 메트릭에 대한 설명, 공식, 값의 범위, 관련된 해석 등을 제시하였다. 또한, 사례 연구를 통해 제안된 품질모델을 PLE 프로젝트에 적용하기 위한 실용적인 가이드라인과 대여 핵심자산을 평가한 결과를 제시하였다. 제안된 품질시스템을 사용하여 핵심자산 재사용성을 더 효과적이고 정확하게 평가할 수 있다.

참 고 문 헌

[1] Clements, P., et al., *Software Product Lines*, Addison-Wesley, 2002.

[2] Bayer, J. et al., "PuLSE: A Methodology to Develop Software Product Lines," *Proceedings of Symposium on Software Reusability '99*, May 1999.

[3] Software Engineering-Product Quality-Part 1: Quality Model. ISO/IEC 9126-1, June, 2001.

[4] Software Engineering-Product Quality-Part 3:

Internal Metrics. ISO/IEC TR 9126-3, July, 2003.
[5] Cho, E., Kim, M., Kim, S., "Component Metrics to Measure Component Quality," *Proceedings of APSEC 2001*, pp.419-426, 2001.
[6] Zubrow, D. and Chastek, G., *Measures for Software Product Lines*, Technical Notes CMU/SEI-2003-TN-031, 2003.
[7] Schmid, K., *A Framework for Product Line Quality Model Development: The PuLSE-Eco Meta Quality Model*, IESE-Report No. 047.00/E, June, 2001.
[8] Washizaki H., et al., "A Metrics Suite for Measuring Reusability of Software Components," *Proceedings of the Ninth International Software Metrics Symposium (METRICS'03)*, pp.211-223, September 2003.
[9] Kim, S., Chang, S., and La, H., "Traceability Map: Foundations to Automate for Product Line Engineering," *3rd ACIS International Conference on Software Engineering Research, Management & Applications (SERA2005)*, To be Appeared in August 2005.
[10] Bosch, J., *Design and Use of Software Architectures*, Addison-Wesley, 2000.
[11] Kim, S., Chang, S., and Chang, C., "A Systematic Method to Instantiate Core Assets in Product Line Engineering," *Proceedings of APSEC 2004*, pp.92-98, 2004.
[12] Kim, S. and Park, J., "C-QM: A Practical Quality Model for Evaluating COTS Components," *Proceedings of the 21st IASTED International Conference*, 2003.
[13] Kim, S., Her, J., and Chang, S., "A Theoretical Foundation of Variability in Component-Based Development," *Information and Software Technology(IST)*, Vol. 47, pp.663-673, July, 2005.
[14] Kim, S., Min, H., Her, J., and Chang, S., "DREAM: A Practical Product Line Engineering using Model Driven Architecture," *Proceedings of the Third International Conference on Information Technology and Applications(ICITA 2005)*, Volume 1, pp.70-75, July 2005.
[15] Bass, L., et al., *Software Architecture in Practice*, Addison-Wesley, 2003.
[16] Matinlassi, M., Niemela, E., and Dobrica, L., *Quality-driven architecture design and quality analysis method: A revolutionary initiation approach to a product line architecture*, VTT Technical Research Center of Finland, ESPOO2002, 2002.
[17] Wijnstra, J., "From problem to solution with quality attributes and design aspects," *The Journal of Systems and Software*, Vol.66, pp.199-211, 2003.
[18] Choi, S., et al., "A Systematic Methodology for Developing Component Frameworks," *Lecture Notes in Computer Science 2984, Proceedings of FASE'04*, pp.359-373, 2004.
[19] Kim, S., et al., "A Systematic Process to Design Product Line Architecture," *Lecture Notes in Computer Science 3480, Proceedings of ICCSA 2005*, p.46-56, 2005.
[20] Kim, S. and Chang, S., "A Systematic Method to Identify Software Components," *Proceedings of the APSEC 2004*, pp.538-545, 2004.
[21] Kim, S., Min, H., and Rhew, S., "Variability Design and Customization Mechanisms for COTS Components," *LNCS 3480, Proceedings of ICCSA 2005*, pp.57-66, 2005.

오 상 헌
2004년 건양대학교 정보전산공학과 공학사. 2004년~현재 숭실대학교 컴퓨터학과 석사과정. 관심분야는 제품계열 공학(PLE), 소프트웨어 품질, 테스팅, 유비쿼터스(Ubiquitous)

허 진 선
2001년 숭실대학교 컴퓨터학부 공학사 2003년 숭실대학교 컴퓨터학과 공학석사 2003년~현재 숭실대학교 컴퓨터학과 박사과정. 관심분야는 제품계열 공학(PLE), 소프트웨어 품질, 컴포넌트 기반 개발(CBD)

김 지 혁
2003년 숭실대학교 컴퓨터학부 공학사 2005년 숭실대학교 컴퓨터학과 공학석사 2005년~현재 숭실대학교 컴퓨터학과 박사과정. 관심분야는 제품계열 공학(PLE), 품질 보증(Quality Assurance), 테스팅

류 성 열
1997년 아주대학교 컴퓨터학부 공학박사 1997년 3월~1998년 3월 George Mason University 교환교수. 1981년 3월~현재 숭실대학교 컴퓨터학부 교수. 관심분야는 리엔지니어링, 소프트웨어 유지보수, 소프트웨어 재사용, 소프트웨어 재공학

김 수 동
정보과학회논문지 : 소프트웨어 및 응용 제 33 권 제 1 호 참조