제품계열공학에서 어플리케이션 생성을 위한 체계적인 프로세스

(A Systematic Process for Generating Applications in Product Line Engineering)

장 치 원 † 장 수 호 † 김 수 동 †††
(Chee Won Chang) (Soo Ho Chang) (Soo Dong Kim)

요 약 제품계열공학(Product Line Engineering, PLE)은 핵심자산(Core Asset) 개발과정과 어플리케이션공학(Application Engineering, AE)과정으로 구성된다. 핵심자산 개발과정은 한 도메인에 속한 여러 어플리케이션들의 공통적인 휘쳐(Feature)를 핵심자산으로 모델링하고, 이를 재사용할 수 있도록 구체화하는 작업이다. 어플리케이션 공학 과정은 핵심자산을 각 어플리케이션의 요구사항에 맞게 인스턴스화(Instantiation)하여 어플리케이션을 효과적으로 개발하는 작업이다. 기존의 PLE 기법들은 핵심자산개발에 크게 치중되어 있고, AE 과정의 연구는 상대적으로 미흡한 실정이다. 특히, AE 과정의 실용화를 위해서는 중요한 활동인 인스턴스화하는 실용적 절차와 기법이 미비하다. PLE의 넓은 산업계의 적용을 위해서는 체계적이며 실용적인 수준의 AE 프로세스, 작업 지침, 산출물의 연구가 요구된다. 본 논문에서는 AE의 실용적 프로세스를 제안하고, 프로세스의 각 활동에 대한 상세 지침을 제시하며, 활동에 따른 산출물 제시한다. 또한 사례 연구를 통하여 제시되는 프로세스에 논리성과 실용성을 검증한다.

키워드 : 제품계열 공학, 어플리케이션 공학, 방법론, 인스턴시에이션, 가변성

Abstract Product Line Engineering (PLE) consists of two phases; Core Assets Development and Application Engineering. The core asset development is to model common features of members in a domain and to develop them. The application engineering is to effectively generate an application by instantiating the core asset. Today, PLE research mainly focuses on developing core assets, whereas activities and instructions for application engineering are weakly defined. Moreover, instructions of application engineering are not enough to be practically applied. To widely apply PLE to industry, researches on systematic and practical methods such as instantiation processes, instructions, and artifacts are needed. In this paper, we propose a practical PLE process, instructions, and artifacts about each activity. And then, we also present a case study to show applicability and practicality of the process proposed in this paper.

Key words : Product Line Engineering, Application Engineering, Process, Instantiation, Variability

1. 서 론

PLE은 어플리케이션을 개발하기 위한 시간과 경제적인 비용을 효과적으로 사용할 수 있는 재사용 기법이다. PLE의 프로세스는 핵심자산 개발과정과 AE 과정으로 구성된다. 핵심자산 개발과정은 목표 도메인의 경제적 가치와 PLE의 범위 정의 후 목표 도메인에 속한 여러 어플리케이션들의 공통성과 가변성을 분석하여 공통적인 휘쳐를 핵심자산으로 모델링하고, 이를 재사용 할 수 있도록 구체화하는 작업이다. AE 과정은 목표 어플리케이션의 요구사항을 분석하여 핵심자산을 요구사항에 맞게 인스턴스화한다. 핵심자산에서 제공하지 않는 목표 어플리케이션의 일부 기능을 모델링하며 인스턴스화된 핵심자산과 통합하여 어플리케이션을 개발한다. AE의 인스턴스화 활동은 핵심자산으로부터 어플리케이션을 생산하기 위한 성공 여부를 결정한다.

그러나 기존의 PLE 기법들은 핵심자산개발에 크게

· 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
† 학생회원 : 숭실대학교 컴퓨터공학과
cwchang@otlab.ssu.ac.kr
shchang@otlab.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터공학과 교수
sdkim@ssu.ac.kr
논문접수 : 2004년 10월 13일
심사완료 : 2005년 6월 18일

치중되어 있으며 AE의 절차 및 지침은 상대적으로 미흡한 실정이다. 특히 AE 과정에서 가장 중요한 활동인 인스턴스화하는 실용적 절차와 기법이 미비하다. PLE의 넓은 산업계의 적용을 위해서는 AE과정을 위한 체계적이며 실용적인 수준의 프로세스, 작업 지침, 산출물의 연구가 요구된다. 본 논문에서는 AE의 실용적 프로세스를 제안하고, 프로세스의 각 활동의 상세지침을 제시하며, 활동에 따른 산출물을 제시한다. 또한 사례연구를 통하여 본 논문의 프로세스의 실용성과 산출물의 적절성을 도모한다. 본 논문의 2장에서는 PLE의 AE 과정을 위한 대표적인 방법론을 소개하며, 3장에서는 핵심자산의 구성요소와 메타모델을 제시한다. 4장에서는 어플리케이션을 개발하기 위한 프로세스와 각 활동 및 지침을 제시하며 각 활동에 따른 산출물을 표현한다. 5장에서는 사례 연구를 통하여 프로세스의 실용성을 검증한다. 마지막으로 6장에서는 결론을 제시하며 본 논문을 마무리한다.

2. 관련연구

2.1 Bosch의 AE 프로세스

Bosch의 AE 프로세스는 초기화(Initial)단계와 반복(Iteration)단계로 구성되어 있다[1]. 초기화 단계는 어플리케이션 멤버의 자산으로부터 초기 소프트웨어 생산물 형상(Initial Software Product Configuration)을 생성한다. 초기화 단계의 접근방법은 조립(Assembly)과 형상선택(Configuration Selection) 두 가지이며, 조립접근은 멤버 간에 공유하는 자산으로부터 초기 소프트웨어 생산물의 형상을 생성하기 위한 방법이다. 형상 선택은 이전 프로젝트의 생산물(Old Configuration)이거나 그의 부분집합(Reference Configuration)을 선택하여 목표 생산물에 맞도록 수정, 추가, 제거, 선택을 통하여 목표 생산물 형상을 생성한다. 초기화 단계의 마지막 활동으로 초기 검증(Initial Validation)은 생산물의 초기 형상에 고객의 요구사항이 충분히 반영되었는지 판단한다. 반복단계에서는 초기화 단계의 산출물인 생산물 초기 형상을 고객의 요구사항이 충분히 구현 될 때까지 반복 수행하여 보완한다. 반복단계는 '수정'과 '검증'활동을 포함하고 있으며, 검증을 통하여 생산물의 형상과 요구사항간에 정확성 및 일관성을 판단하여 반복 여부를 결정한다. 적응(Adaptation)활동은 초기화, 반복단계에 적용되며, 적응활동은 '생산물을 명세한 적용(Product-Specific Adaptation)', '반응 진화(Reactive Evolution)', '학습에 영향 받는 진화(Proactive Evolution)'로 3가지 단계로 나눈다. 전체적인 AE프로세스를 제시하고 있으나 각 활동들과 지침들이 개념적이며 각 활동에 따른 실용적인 산출물을 제시하지 못하고 있다.

2.2 PuLSE의 AE 프로세스

PuLSE(Product Line Software Engineering)는 PLE 개발 프로세스로 독일의 IESE에서 개발 했다[2]. 이 프로세스는 배포단계(Deployment Phase), 기술적 컴포넌트(Technical Components), 지원 컴포넌트(Support Components) 3개로 구성되어 있다. 기술컴포넌트는 PuLSE-BC, PuLSE-Eco, PuLSE-CDA, PuLSE-DSSA, PuLSE-I, 마지막 PuLSE-EM으로 구성되어 있다. 특히 생산물을 생성하기 위한 기술컴포넌트의 PuLSE-I는 생산물을 명세하며, 생성하여 평가하기 위한 절차로서 5가지의 단계로 되어 있다. 첫 번째 단계는 사용자의 요구사항을 이용하여 새로운 생산품을 인스턴스화 하기 위한 계획을 수립하는 단계이다. 두 번째 단계는 생산공정 모델을 구체화하고 검증 한다. 이 단계에서는 의사결정 모델로부터 도메인 모델을 명세 하고, 사용자 요구사항으로부터 도출된 도메인 모델을 검증한다. 세 번째 단계는 아키텍처를 구체화 하고 검증하는 단계이다. 이 단계는 형상 모델과 생산물 명세서를 통하여 생산물의 아키텍처를 도출한다. 네 번째 단계는 구현이다. 이 단계는 낮은 수준의 디자인이나 코드를 구현하는 단계이다. 마지막으로 사용자에 의해 승인 테스트가 이루어 진다. PuLSE는 생산물을 생성하기 위한 전반적인 프로세스를 제공하고 있으나 실용적인 지침과 산출물을 제시하지 못하고 있다.

2.3 KobrA의 AE 프로세스

Kobra는 UML(unified Modeling language)을 이용한 컴포넌트 기반의 PLE 개발 프로세스이다[3]. 이 프로세스는 프래임웍 공학(Framework Engineering)과 AE로 구성 되어있다. 프래임웍 공학에서 범용 프래임웍을 생산 하며 AE에서 범용 프래임웍을 이용하여 하나의 특정 어플리케이션을 개발한다. 프래임웍 공학의 산출물인 범용 어플리케이션은 PLE에 포함된 모든 멤버의 특성을 다 가지고 있다. KobrA에서의 AE 프로세스는 배경구현(Context Realization) 인스턴스화, Komponent 명세화(Komponent Specification)와 Komponent 현실화(Komponent Realization) 인스턴스화로 되어 있다. 배경 구현에서는 컨설턴트들은 고객과 함께 생산물의 특징들을 도출하여 프래임웍 공학에서 개발된 후보 프래임웍과 고객이 요구하는 특징이 얼마나 많이 중복되는지를 확인한다. 프래임웍안에서 Komponent의 명세와 현실화의 인스턴스화를 반복적으로 수행함으로서 명세 Komponent 제약 트리(Specific Komponent Containment Tree)를 생성한다. 모델링을 통하여 생성된 Komponent 제약 트리를 이용하여 하나의 특정한 어플리케이션을 생성한다. KobrA는 구체적이며 상세한 지침을 제공 하고 있으나 실용적인 프로세스를 제시하기

위해서는 좀더 정재 될 필요가 있다.

3. 핵심 자산의 구성요소

본 논문에서는 핵심자산을 입력물로 하여 AE 프로세스의 기법을 제안한다. 그러나 핵심 자산의 정의와 구성요소가 PLE 기법간에 차이를 보이고 있어, 본 장에서는 구성요소 정의 후 이를 기반으로 4장에서의 AE 프로세스를 제안한다.

핵심자산은 PLE에서 핵심적인 역할을 하며 그림 1과 같이 핵심자산의 구성요소는 범용 아키텍처, 컴포넌트 모델, 인터페이스, 의사결정 모델이다. 아키텍처는 일반적으로 모듈 뷰타입(Module Viewtype), 컴포넌트와 커넥터 뷰타입(Component and Connector Viewtype, C&C Viewtype), 할당 뷰타입(Allocation Viewtype)의 세 가지 뷰로 정의된다[4]. 그림 1의 '범용 아키텍처'는 여러 패밀리 멤버의 제품에 공통으로 사용되는 아키텍처로서 공통성뿐만 아니라 각 패밀리 멤버마다 다른 가변성도 포함하고 있다. '모듈 뷰타입'은 설계 단계에서의 잘 정의된 인터페이스를 가지며 서로 연관된 기능으로 묶인 모듈과 모듈간의 관계를 나타내며, 이는 '기능 모델(Functional Model)'과 '유즈케이스 모델(Usecase Model)', 클래스 다이어그램을 포함하는 '객체 모델(Object Model)'로 표현될 수 있다. 'C&C 뷰타입'은 실행 시간에서 모듈간의 상호작용 관계를 나타내며, 시퀀스 다이어그램(Sequence Diagram), 협동 다이어그램(Collaboration Diagram), 활동 다이어그램(Activity Diagram)과 같은 '행동 모델(Behavioral Model)'로 표현될 수 있다. 할당 뷰타입은 어플리케이션과 해당 어플리케이션이 실행되는 환경요소간의 관계를 나타낸 것으로, 모듈 뷰타입과 C&C 뷰타입에 비해 여러 패밀리 멤버의 공통성이 상대적으로 적으므로 할당 뷰타입을 제외하고 모듈 뷰타입과 C&C 뷰타입만 고려한다. C&C 뷰타입에서 컴포넌트를 모듈로 보지 않는다. 하지만 일반적으로 제품계열 공학에서는 컴포넌트 기반으로 하므로 핵심 자산에서는 컴포넌트가 모듈에 해당될 수 있으며 소프트웨어 아키텍처의 요소로 컴포넌트가 될 수 있다.

컴포넌트 모델은 클래스들로 구성된 컴포넌트, 인터페이스, 컴포넌트간의 관계로 구성되어 있으며, 컴포넌트간의 관계는 범용 아키텍처에 표현된다. 컴포넌트 모델은 인터페이스 명세서와 컴포넌트 명세서에 표현 된다. 의사결정모델은 C&V 명세서의 가변성을 명세하고 있으며 가변점(Variation Point), 연관된 가변치(Variants), 영향(Effect), 활동(Task)으로 구성되어 있다. 의사결정 명세서의 가변성의 리스트는 모두 아키텍처 명세서, 컴포넌트 명세서, 인터페이스 명세서에 반영 및 표현된다.

4. 프로세스와 지침

본 절에서는 AE 과정을 위한 프로세스를 제안한다. 그림 2와 같이 이 프로세스는 5개의 단계로 구성되었으며 각 단계는 여러 개의 세부 활동으로 구성 되어 있다.

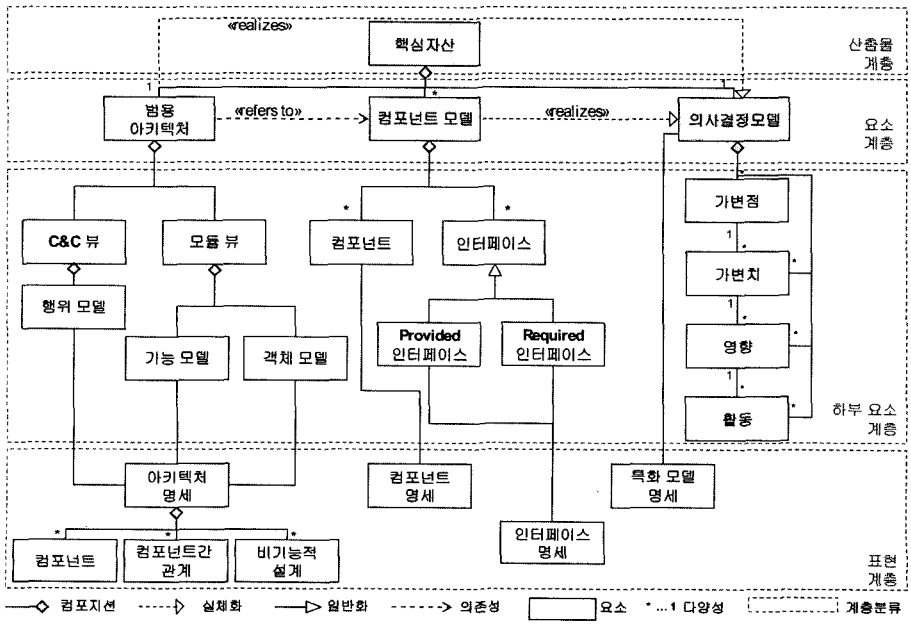

그림 1 핵심자산 메타모델

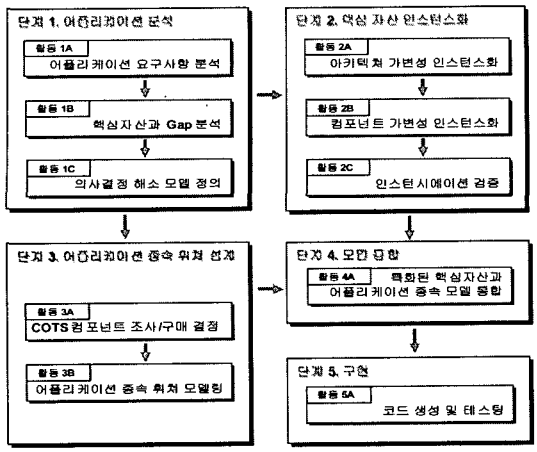

그림 2 AE 과정을 위한 프로세스

각 단계의 세부 활동은 4.1, 4.2, 4.3, 4.4, 4.5에서 상세히 다룬다. 고객의 요구사항을 입력물로 제안된 프로세스를 이용하여 최종 목표인 어플리케이션을 개발 한다.

4.1 단계 1. 어플리케이션 분석

단계 1에서는 목표 어플리케이션의 입력물로 요구사항을 분석 하여 이후 목표 어플리케이션을 개발 하기 위한 어플리케이션 종속 휘쳐들을 식별하고, 의사결정 해소 모델(Decision Resolution Model: DRM)을 정의 한다. 그림 3에서는 어플리케이션 분석의 세부활동들에 대한 입력물과 산출물을 보여준다.

4.1.1 활동 1A. 어플리케이션 요구사항 분석

이 활동에서는 목표 어플리케이션의 기능적 요구사항과 비기능적 요구사항을 분석한다. 사용자가 원하는 시스템의 기능들을 식별하기 위한 활동으로 기능적 요구사항을 유즈케이스 모델링(Use Case Modeling)을 통해 분석하고, 비기능적 요구사항을 품질 속성별(Quality Attribute)로 정리한다. 기능적 요구사항과 비기능적 요구사항은 유즈케이스 다이어그램의 하나의 유즈케이스 혹은 하나 이상의 유즈케이스로 상세화 된다. 휘쳐는 요구사항의 기능적 요구사항과 비기능적인 요구사항에 대한 재사용성 단위로 구분 한다. 따라서 유즈케이스는 휘쳐의 재사용성 단위로 매핑 되므로, 휘쳐와 유즈케이스는 상호 보완물 관계이다[5]. 각 유즈케이스 모델링을 통하여 목표 어플리케이션의 기능적 요구사항과 비기능적 요구사항의 어플리케이션 휘쳐 리스트를 작성한다. 이 산출물은 다음 단계인 핵심 자산과 Gap분석의 입력으로 사용한다.

4.1.2 활동 1B. 핵심 자산과 Gap 분석

이 활동에서는 핵심자산과 어플리케이션 휘쳐 리스트들 사이에 중복된 휘쳐와 어플리케이션 종속 휘쳐를 식별한다.

이상적인 PLE은 핵심자산이 각 제품 멤버에서 사용되는 공통적인 휘쳐들을 포함하고, 어플리케이션이 개발된 핵심자산의 모든 휘쳐를 사용하는 것이다. 그러나 현실적으로 핵심자산 개발단계에서 정의된 일부 휘쳐들은 어플리케이션에서 사용되지 않을 것이다. 이러한 경우는 핵심자산 개발 후 새로운 어플리케이션이 추가 되었을 때 나타날 수 있다. 이것은 그림 4의 '어플리케이션에서 사용되지 않는 휘쳐'로 표현된다. 또한, 이상적인 PLE은 핵심자산이 어플리케이션에서 필요한 모든 기능을 제공

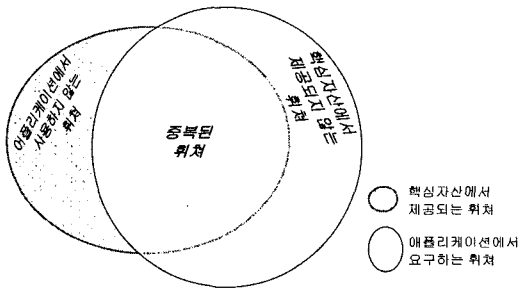

그림 4 핵심자산과 어플리케이션간의 매칭

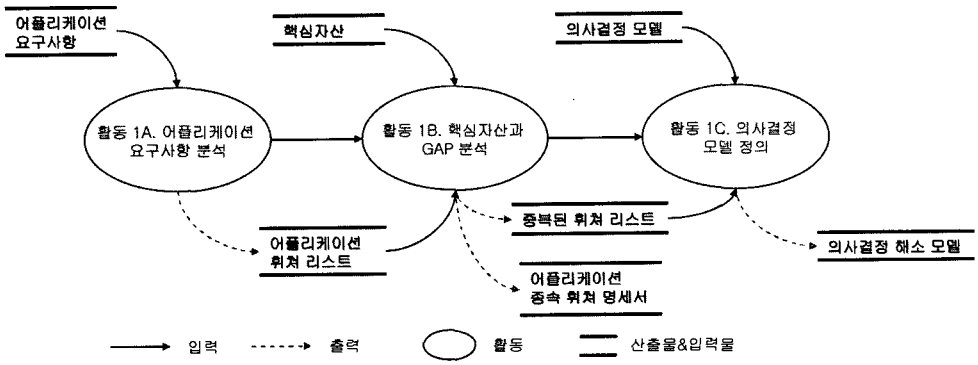

그림 3 어플리케이션 분석을 위한 입력물과 산출물

하는 것이다. 그러나 현실적으로 어플리케이션에는 핵심 자산이 제공하지 않는 훠쳐들도 존재 한다. 이것은 그림 4의 '핵심자산에서 제공되지 않는 훠쳐'로 표현된다. 그러므로 본 논문에서 제시한 핵심자산과 어플리케이션간의 Gap 분석을 통하여 핵심자산 인스턴스화와 어플리케이션 종속 훠쳐 설계 활동을 체계적이며 실용적으로 접근 할 수 있다.

이 단계의 산출물은 중복된 훠쳐 리스트와 어플리케이션 종속 훠쳐 명세서이다. 이 산출물의 각각은 의사결정 해소 모델 정의와 어플리케이션 종속 훠쳐 설계 단계의 입력으로 사용된다. 이 활동을 통해서 핵심자산과 어플리케이션간의 Gap분석뿐만 아니라 핵심자산에 추가될 수 있는 기능을 발견 할 수 있다. PLE은 오랜 기간 동안 핵심 자산을 개발 하면서 앞으로 추가될 수 있는 어플리케이션에 대하여 고려 하지 않는 경우가 있다. 따라서 핵심 자산이 제공하지 않으나, 이 활동과정에서의 목표 어플리케이션에 요구되는 기능이 핵심 자산에 포함되는 것이 경제적이며 합리적인지를 판단하는데 활용될 수 있다. 이 절차는 어플리케이션 활동 외의 핵심자산 개발의 활동인 '핵심자산 진화(Evolution)'에서 다루어져야 한다.

4.1.3 활동 1C. 의사결정 해소 모델(DRM)정의

의사결정 모델에 명세된 가변성의 가변점과 가변치는 핵심 자산에 적용되어 설계되고, 각 어플리케이션 개발 단계에서는 어플리케이션에 적합한 가변점과 연관된 가변치를 선택한다. 의사결정 해소 모델정의 단계에서는 목표 어플리케이션에 맞는 가변점과 연관된 가변치를 의사결정 모델에서 선택하거나 새로 정의한다. 먼저 활동 1B로부터 도출된 중복된 훠쳐의 리스트 중 어플리케이션에 해당되는 가변점을 식별한다. 그림 4와 같이 어플리케이션에 의해 사용되지 않는 훠쳐들이 존재하기 때문에 의사결정 모델에 정의된 가변점들은 목표 어플리케이션에 모두 사용 되지 않는다. 즉, 어플리케이션이 가지는 가변점들은 의사결정 모델에서 정의된 가변점들의 부분 집합이다. 그러므로 어플리케이션의 가변점 리스트는 의사결정 해소 모델을 작성하는 기초정보가 된다. 앞에서 정의된 어플리케이션에 해당되는 가변점과 연관된 어플리케이션의 가변치를 정의한다. 가변성의 범위는 *closed*이거나 혹은 *open*이 될 수 있다[6]. *closed* 가변성은 의사결정 모델이 가변점에 해당되는 유효한 가변치들이 이미 정의되어 있는 경우이다. 그러나 *open* 가변성은 가변점에 대한 가변치들이 의사결정 모델에 정의 되어 있지 않은 경우이다. 즉, 가변치가 아직 결정되지 않은 경우이다. 따라서 범위가 *closed*가변성일 경우, 어플리케이션을 위한 가변치를 의사 결정 모델에서 선택한다. 그러나 *open*가변성일 경우, 어플리케이션을 위해 가변치를 새롭게 정의 해야 한다.

표 1 의사결정 해소 모델

아이디	가변점 (Variation Point)	가변치 (Variants)	영 향 (Effect)	활 동 (Task)
1	VP_i	$Variant_j$	$Effect_i$	$Task_j$
...	...	...	...	...

표 1은 의사결정 해소 모델을 위한 산출물의 양식으로 가변점과 가변치, 가변치 선택으로 인한 영향과 모델 과정에서 수정이 요구 되는 활동들을 포함한다.

현재 PLE은 의사결정 해소 모델의 요소로서 가변점과 가변치를 공통적인 요소로 제시한다. 그러나 가변치 선택으로 인한 영향과 모델과정에서 수정이 요구 되는 구체적인 활동들을 깊이 있게 다루지 않고 있다. 본 논문에서는 가변치 인스턴스화의 영향과 수행활동을 다음과 같이 정의한다.

'영향'이란 가변치를 선택한 후 발생하는 사후 조건이거나 영향을 받는 연관된 가변점들에 대한 설명으로 정의된다. 따라서 영향은 가변치 선택 후 핵심자산 내 연관된 모델의 변경 상태를 설명한다. 또한 영향은 가변성간의 관계에 대한 범위를 표현한다. 즉 한 가변점이 영향을 주는 다른 가변점들 간의 두 관계는 필수, 상호 배타로 구분할 수 있다. *closed*가변성일 경우, 의사결정 모델에서 포함하고 있는 가변치의 영향은 이미 정의 되어 있으나, *open*일 경우, 가변치의 영향은 모델간의 연관관계에 따라 새롭게 정의 된다. 예를 들어, 다음과 같은 내용이 영향에 들어 갈 수 있다. VP_1의 $Variant_4$를 선택할 경우 다른 VP_5의 $Variant_3$을 선택해야만 하는 경우, VP_1의 $Variant_4$를 선택할 경우 다른 VP_2의 $Variant_4$을 선택하지 말아야 할 경우 등이 존재한다.

'활동'이란 가변성 인스턴스화를 위해 요구되는 작업들, 즉 영향을 도출하기 위한 작업을 의미한다. 또한 가변점은 속성, 로직, 워크플로우, 인터페이스 등에서 다르게 나타나므로, 가변성의 각각의 형태에 맞는 다른 커스터마이제이션 기술이 필요하다. 따라서 활동은 연관된 커스터마이제이션 기술에 따라 다르게 나타난다. 예를 들어, 아키텍처 가변성의 '보고서 관리 컴포넌트'이라는 가변점에 Variant1인 '보고서 관리 컴포넌트 사용'와 $Variant_2$인 '보고서 관리 컴포넌트 사용 없음' 있다. 생산하고자 하는 어플리케이션이 $Variant_2$를 선택하였을 경우, 활동은 보고서 관리 컴포넌트 제거, 보고서 관리 컴포넌트를 호출하는 다른 컴포넌트에서 불필요한 호출 제거 등이 있다.

4.2 단계 2. 핵심자산 인스턴스화

본 단계에서는 단계 1에서 정의된 의사결정 해소 모델을 이용하여 어플리케이션에 필요한 가변성을 인스턴

스화하여 인스턴스화된 핵심자산을 생성한다. 그림 5에서는 핵심자산 인스턴스화의 세부활동들에 대한 입력물과 산출물을 보여준다.

4.2.1 활동 2A. 아키텍처 가변성 인스턴스화

소프트웨어 아키텍처는 컴포넌트와 컴포넌트간의 관계들의 구현하고 있으며 기능적, 비기능적인 사항을 포함하고 있다. 핵심자산의 범용 아키텍처는 어플리케이션들 사이에 공통된 아키텍처적인 의사결정을 포함한다[7-9]. 범용 아키텍처의 가변성은 어플리케이션의 기능적, 비기능적인 요구사항의 가변성을 반영한다. 이런 가변성은 아키텍처를 구성하는 컴포넌트와 혹은 내부 컴포넌트의 관계에 반영된다.이 활동에서는 먼저 범용 아키텍처로부터 똑같은 아키텍처 복사본을 생성한다. 이 복사본은 가변성을 인스턴스화 하기 위한 중간 아키텍처다[2]. 중간 아키텍처에 불필요한 사항은 인스턴스화된 어플리케이션에 맞게 계속적으로 삭제된다. 만약 가변성이 어플리케이션에 필요한 컴포넌트들의 집합에 있다면 어플리케이션에서 필요한 컴포넌트만을 선택한다. 예를 들어 제품계열 공학에서 보안 관리는 비기능적인 요구사항이다. 일부의 어플리케이션에서는 이 요구사항을 기능적인 컴포넌트를 구현하는 것을 요구하지 않을 것이다. 그러므로 불필요한 컴포넌트는 중간 아키텍처에서는 삭제되어야만 한다. 만약 가변성이 컴포넌트들 간의 관계에 있으면 어플리케이션에서 필요한 필수적인 내부 컴포넌트 관계를 재 정의하거나 선택 해야 한다. 이러한 사항은 목표한 아키텍처를 정의하면서 아키텍처의 스타일을 적용하거나 선택하는 것이 일반적이다.

만약 아키텍처 디자인 과정에서 아키텍처 스타일이 적용되었다면 스타일에서 명세화된 내부컴포넌트 관계는 중간 아키텍처에 적용 된다. 만약 아키텍처 스타일이 적용되지 않았다면 내부 컴포넌트의 디자인은 개발자가 어플리케이션 요구사항을 보고 이끌어내야 한다. 아키텍처의 가변성 인스턴스화는 목표한 어플리케이션에서 필요하지 않는 컴포넌트와 컴포넌트간의 관계를 중간 아키텍처에서 삭제 하는 것이다.

4.2.2 활동 2B. 컴포넌트 가변성 인스턴스화

이 활동에서는 핵심 자산 컴포넌트 모델내의 가변점에 연관된 가변치를 인스턴스화 한다. 핵심 자산 컴포넌트 모델은 closed범위의 가변점안에 모든 후보 가변치를 포함한다. 따라서 목표한 어플리케이션에 불필요한 가변치를 제거해야 한다. 불필요한 가변치 제거 후 컴포넌트 모델은 목표 어플리케이션에 최적화 된다. 컴포넌트의 가변성 인스턴스화는 컴포넌트의 구조적인 모델과 행동 모델을 수정하는 것이다. 비기능적 요구사항은 또한 컴포넌트 모델에 설계 된다. 따라서 비기능적 요구사항에 연관된 가변점 인스턴스화는 컴포넌트 모델 수정에 포함 된다. 의사결정 해소 모델에서 정의된 영향과 작업은 컴포넌트를 수정하면서 같이 병행 된다.

4.2.3 활동 2C. 가변성 검증

이 단계에서는 어플리케이션 요구사항 명세를 이용하여 의사결정 해소 모델을 검증한다. 그림 6에서는 핵심자산, 의사결정모델, 인스턴스화된 핵심자산간의 추적맵을 제시 한다. 또한 각 산출물간의 요소를 정의하며, 각 요소들간의 추적성을 정의 한다. 어플리케이션의 요구사항을 이용하여 의사결정 해소 모델에 필수적인 가변점을 모두 가지고 있는지를 확인하다. 또한 가변점에 필요한 모든 가변치를 가지고 있는지 확인한다. *open*가변성의 경우, 추가되는 가변치가 의사결정 모델에 존재하는지를 검사한다. 만약 존재 하지 않는다면 가변치를 추가한다.

의사결정 해소 모델과 인스턴스화된 핵심자산과 일치하는지 검증을 위하여 아래의 인스턴스화된 핵심자산 검증 리스트를 이용하여 검증한다. 검증 리스트의 항목이 추가 될 수 있다.

- 의사결정 해소 모델에서 명세된 모든 가변점은 인스턴스화된 핵심자산에 반영되어야 한다.
- 의사결정 해소 모델에서 명세된 모든 가변점은 연관된 가변치와 함께 인스턴스화 되어야 한다.
- 영향에 명세된 내부 가변점 관계와 사후 조건은 인스턴스화된 핵심 자산에 나타나야 한다.

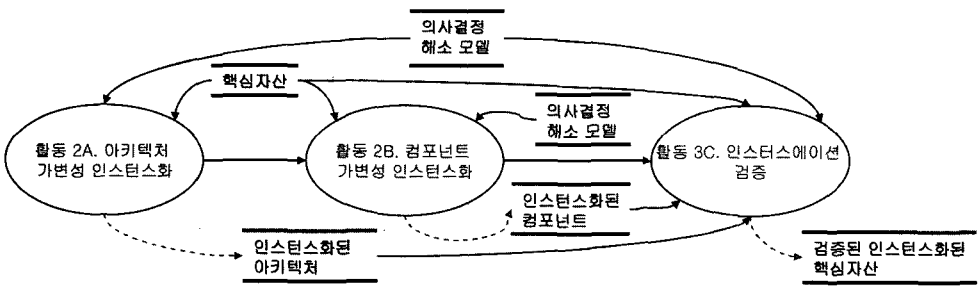

그림 5 핵심자산 인스턴스화를 위한 입력물과 산출물

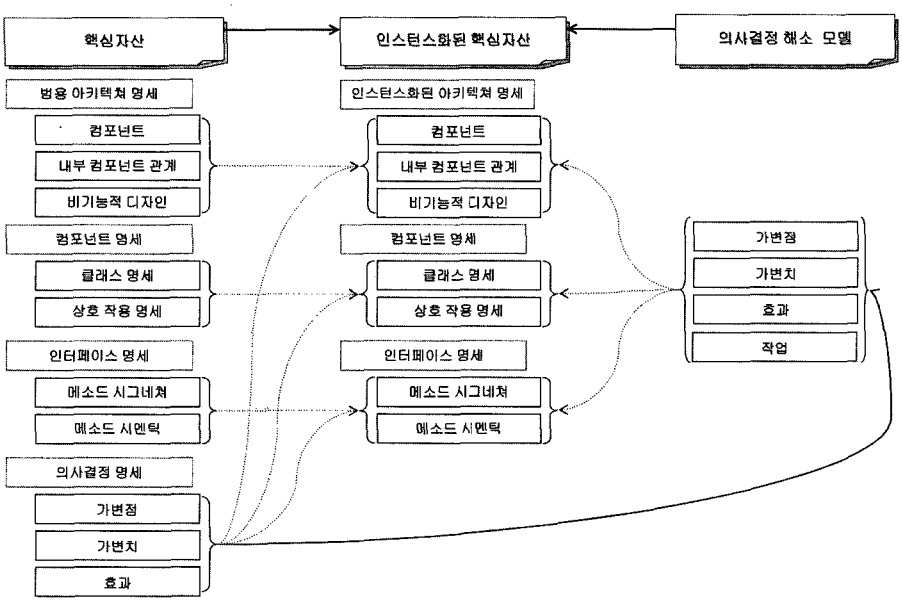

그림 6 특화된 핵심자산 검증 추적 맵

• 의사결정 해소 모델에 선택되지 않은 가변치는 인스턴스화된 핵심자산에서 제거되어야 한다.

4.3 단계 3. 어플리케이션 종속 휘쳐 설계

인스턴스화된 핵심자산은 목표 어플리케이션을 개발하기 위한 대부분의 기능을 제공할 것이다. 어플리케이션에서 개발될 기능은 하나의 컴포넌트이거나 클래스 하나를 이루는 컴포넌트와 여러 개의 작은 컴포넌트로 이루어진다. 따라서 본 단계에서는 핵심자산에서 제공되지 않는 일부의 어플리케이션 종속 휘쳐들을 설계 또는 COTS 컴포넌트를 구매한다. 그림 7에서는 어플리케이션 종속 휘쳐 설계의 세부활동들에 대한 입력물과 산출물을 보여준다. 그림 8에서는 어플리케이션 종속 휘쳐를 설계 또는 COTS컴포넌트 구매하기 위한 세부절차를 제시 한다.

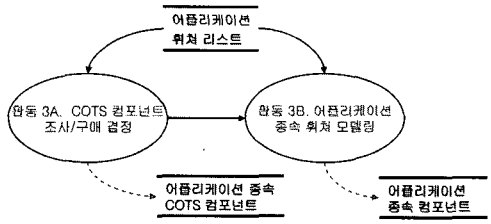

그림 7 어플리케이션 종속 휘쳐 설계를 위한 입력물과 산출물

4.3.1 활동 3A. COTS 컴포넌트 조사/구매 결정

이 활동에서는 그림 8의 세부절차를 이용하여 COTS 컴포넌트를 조사하며, 구매여부를 결정한다. COTS 컴포넌트 구매를 위한 세부절차는 5단계이며 각 절차 별로 반복적인 비교를 하며 마지막으로 경제성을 확인하여 구매 여부를 결정한다.

그림 8의 세부절차 1에서 어플리케이션 종속 휘쳐와 비슷한 COTS컴포넌트의 존재여부를 확인한다. 세부절차 2에서 COTS 컴포넌트 리스트를 생성하며, 세부절차 3의 입력물로 사용된다. 세부절차 3에서 어플리케이션 종속 휘쳐 명세서와 COTS컴포넌트를 기능별로 비교하여 일치하는지를 확인한다. 세부절차 4에서는 세부절차 3에서의 결과물인 COTS 컴포넌트와 인스턴스화된 핵심자산과 비교하여 인터페이스간의 불일치를 최소화 할 수 있으며 결합할 수 있는 COTS 컴포넌트를 확인한다. 각 세부절차 3와 4에서 일치하는 COTS 컴포넌트가 없으면 단계 2로 이동하여 다음 COTS 컴포넌트를 입력한다. 세부절차 2의 COTS컴포넌트의 리스트에 COTS 컴포넌트가 더 이상 없을 경우 어플리케이션 종속 휘쳐들 설계한다. 마지막으로 세부절차 5에서 최종으로 COTS 컴포넌트 구매비용과 컴포넌트 개발 비용을 비교하여 구매여부를 결정한다.

4.3.2 활동 3B. 어플리케이션 종속 휘쳐 모델링

이 활동은 그림 8의 세부절차 수행의 결과가 컴포넌트 설계로 선택된 경우 수행한다. 단계 1의 산출물인 어플리케이션 종속 휘쳐 명세서를 이용하여 어플리케이션 종속 컴포넌트를 설계한다. 설계 작업은 어플리케이션 종속 휘쳐 명세서에 포함된 유즈케이스뿐만 아니라 컴

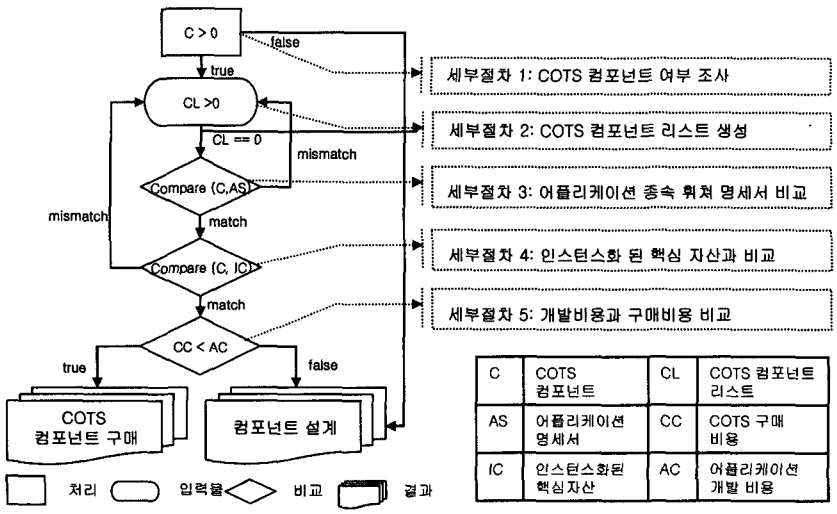

그림 8 컴포넌트 구매 혹은 설계를 위한 세부절차

포넌트 설계를 위한 클래스 모델, 시퀀스 모델, 컴포넌트 인터페이스 명세로 이루어 질 수 있다.

어플리케이션 종속 훼쳐 설계 시 인스턴스화된 핵심자산과 어플리케이션 훼쳐간의 인터페이스간의 불일치를 최소화 할 수 있도록 인스턴스화 된 핵심자산과 비교해 가며 모델링 한다. 또한 어플리케이션 종속 훼쳐 명세서에 인터페이스간에 매핑점을 명시하여 두 컴포넌트간의 통합과정을 쉽고 불일치 없이 수행하며 모델통합 과정에서 검증활동으로 활용한다. 이 활동의 최종 산출물은 어플리케이션 종속 컴포넌트 모델이다.

4.4 단계 4. 모델통합

그림 9에서는 모델통합의 세부활동에 대한 입력물과 산출물을 보여준다.

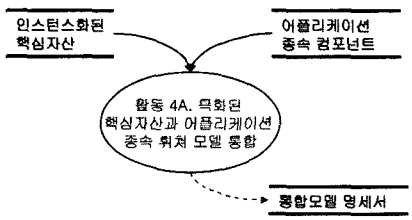

그림 9 모델통합을 위한 입력물과 산출물

4.4.1 활동 4A. 특화된 핵심자산과 어플리케이션 종속 훼쳐 모델 통합

본 단계는 단계 2와 단계 3의 산출물인 '인스턴스화된 핵심자산'과 '어플리케이션 종속 컴포넌트'를 통합하는 단계이다. 인스턴스화된 핵심자산은 그림 10과 같이 인스턴스화된 아키텍쳐, 인스턴스화된 컴포넌트 모델, 의사결정 해소 모델로 구성하며, 의사결정 모델은 목표 어플리케이션으로 인스턴스화 하기 때문에 더 이상 표현하지 않는다. 인스턴스화된 핵심자산의 각 구성요소는 의사결정 해소 모델을 이용하여 목표 어플리케이션에 맞도록 인스턴스화한다.

어플리케이션 종속 훼쳐들은 단계 3의 활동 3A의 산출물인 COTS 컴포넌트이거나 혹은 활동 3B의 산출물인 어플리케이션 종속 컴포넌트이다. 어플리케이션 종속 훼쳐들은 핵심자산에서 일부 제공되지 않는 작은 기능이다. 따라서 인스턴스화된 핵심자산과 어플리케이션 종속 컴포넌트간에 기능호출 시 아무런 불일치 없이 제공할 수 있는 인터페이스를 제공 해야 한다. 단계 3의 활동 3B의 산출물인 어플리케이션 종속 컴포넌트와 인스턴스화된 핵심자산과의 통합일 경우, 두 컴포넌트간의 통합은 어플리케이션 종속 훼쳐 설계와 병행하여 수행한다. 따라서 모델 통합 단계에서는 어플리케이션 종속 훼쳐 명세서에 명시된 매핑점을 이용하여 두 컴포넌트간의 통합이 정상적으로 수행되었는지 검증을 한다. 이상적인 모델통합은 인스턴스화된 핵심자산에서 원하는 기능을 어플리케이션 종속 컴포넌트의 인터페이스에서 어떠한 수정과 작업 없이 제공하는 것이다. 현실적으로 COTS 컴포넌트를 구매한 경우, 핵심자산과 통합을 하면서 불일치가 발생 할 수 있다. 따라서 두 컴포넌트의 통합을 위하여 COTS 컴포넌트를 수정 작업 또는 두 컴포넌트간의 매핑 작업이 필요하다. 이 단계의 산출물인 통합 모델 명세서는 단계 5의 구현을 위한 입력물로 사용한다.

4.5 단계 5. 구현

그림 11에서는 구현 세부활동의 입력물과 산출물을 보여준다.

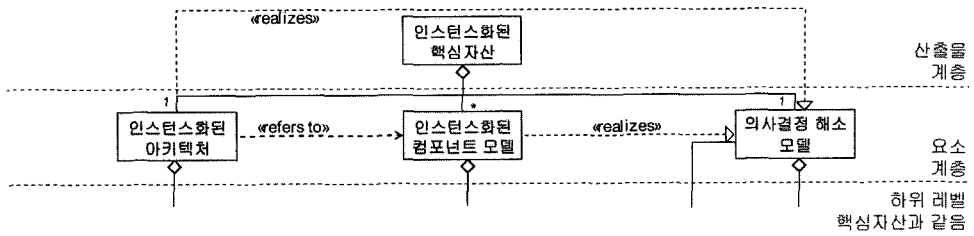

그림 10 스턴스화된 핵심자산

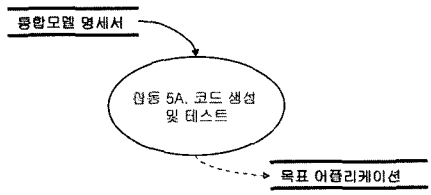

그림 11 구현을 위한 입력물과 산출물

4.5.1 활동 5A. 코드 생성 및 테스팅

단계 4의 산출물인 모델 통합 명세서를 이용하여 코드를 생성하며, 코드 튜닝 및 정제, 테스팅을 통하여 어플리케이션을 생성한다. 모델 통합 명세서의 구조적인 모델과 행위 모델 등을 이용하여 소스 코드화 한다. 생성된 코드의 알고리즘 확인 후 알고리즘상 문제가 발생하거나 성능상에 문제가 있을 경우, 튜닝활동을 통하여 알고리즘을 최적화 한다. 튜닝활동 이후 코드에서 목표 어플리케이션을 생성하는데 불필요한 코드와 어플리케이션에 필요한 코드를 확인하여 정제 활동을 수행한다. 소스코드 개발 이후 최종 산출물인 어플리케이션에 대한 에러 검출, 유지 보수 비용 향상, 어플리케이션의 성능 향상을 위하여 테스팅 활동을 수행한다[10]. 테스팅은 단위(Unit) 테스팅, 통합(Integration) 테스팅, 시스템(System) 테스팅을 순으로 수행한다. 어플리케이션의 최소 구성요소인 컴포넌트가 정상적으로 수행 하는지 확인하기 위하여 단위 테스팅을 수행하고 컴포넌트간의 기능적 호출 등이 정상적으로 수행하고 있는지 확인하는 통합테스팅을 수행한다. 어플리케이션의 모든 요소들이 통합되고 올바른 기능을 수행하는지를 확인하기 위한 시스템 테스팅을 수행한다. 최종적으로 고객에게 개발된 어플리케이션을 고객에 인도한다.

5. 사례 연구

이 장에서는 본 논문의 프로세스와 산출물을 대여 도메인에 적용한 사례를 보여주며, 본 논문의 프로세스의 실용성 및 산출물의 적절성을 검증한다. 대여 도메인에는 자동차 대여, 어린이 장난감 대여, 웨딩드레스 대여, 도서 대여, 비디오 대여 등 다양 어플리케이션을 포함하고 있다.

5.1 단계 1. 어플리케이션 분석

목표 어플리케이션은 도서 대여 관리 시스템이며, 고객으로부터 도서 관리 시스템의 요구사항을 받아 분석한다. 도서 관리시스템 요구사항에서 요구하는 서브 시스템 구성은 회원관리, 대여관리, 예약관리, 대여 품목 관리 등으로 되어 있다.

5.1.1 활동 1A. 어플리케이션 요구사항 분석

도서 대여 관리 시스템에 필요한 휘처를 분석 하기 위하여 유즈케이스를 작성한다. 각 유즈케이스에는 도서 관리 시스템에서 요구하는 기능적 요구사항과 비기능적 요구사항이 존재한다.

5.1.2 활동 1B. 핵심자산과 Gap분석

핵심자산의 휘처 리스트와 어플리케이션 종속 휘처 리스트간의 중복성 여부를 검사하여 아래와 같은 중복된 휘처 리스트와 어플리케이션 종속 휘처 명세서를 작성한다.

중복된 휘처 리스트	어플리케이션 종속 휘처 리스트
✓ Register Product	✓ Generate Rental Report
✓ Update Product	✓ Generate Account Report
✓ Search Product	✓ Generate Reservation Report
✓ Remove Product	✓ Generate Inventory Report
✓ ...	✓ ...

5.1.3 활동 1C. 의사결정 해소 모델(DRM)정의

핵심자산에서 제공하는 의사결정 모델을 목표 어플리케이션 위한 의사결정 해소 모델로 정의한다. 의사결정 해소 모델을 정의하기 위하여 도서 관리 시스템에 맞는 가변점과 연관된 가변치를 선택한다. 도서 대여 관리 시스템 적합한 가변점은 표 2의 'Attribute for Registering Product'와 'Registering Product Algorithm'이며 연관 가변치는 각각 V2={Basic Attribute + Author, Genre, Loan Period}, V2 = {VIPmember component 사용 안함}이다. 또한 아키텍처 가변성으로 가변점은 'Component For using VIPmember' 이며 연관된 가변치는 V2 = {VIPmember component 사용 안함}이다.

표 2의 핵심자산의 의사결정 모델에 선택된 가변점과 연관된 가변치를 이용하여 도서대여 관리 시스템을 위

표 2 의사결정 모델

아이디	가변점 (Variation Point)	가변치 (Variants)		영 향 (Effect)	활 동 (Task)	확인 (Check)
1	(Architecture Variability) Component For using VIPmember	V1 = { VIPmember component 사용}				
		V2 = { VIPmember component 사용 안함}			1. C_VIPmember component 제거 2. C_Member의 등급 조정 기능 제거 3. Rental 할인율 계산 기능 수정	√
2	Attribute for Registering Product	Basic Attribute ={ProductID, Product Name, Manufacturer, Release Date, Register Date, Rental Fee, Copies}	V1={ Basic Attribute + Genre, Memo, Loan Period }	물품등록 알고리즘 변경(V1)선택	Author, Type Attribute 제거	
			V2={ Basic Attribute + Author, Genre, Loan Period }	물품등록 알고리즘 변경(V2)선택	Type, Memo Attribute 제거	√
			...	...	...	...
3	Registering Product Algorithm		V1={Register (Basic Attribute + Genre, Memo, Loan Period)}		Author, Type Parameter 제거 입력 Parameter로직 구성	
			V2={Register (Basic Attribute + Author, Genre, Loan Period)}		Type, Memo Parameter 제거 입력 Parameter로직 구성	√
			...	...	...	...
...	...	...		...	...	...

표 3 의사결정 해소 모델

아이디	가변점 (Variation Point)	가변치 (Variants)		영 향 (Effect)	활 동 (Task)
1	Architecture Variability	V2 = { VIPmember component 사용 안함}			1. C_VIPmember component 제거 2. C_Member의 등급 조정 기능 제거 3. Rental 할인율 계산 기능수정
2	Attribute for Registering Product	Basic Attribute ={ProductID, Product Name, Manufacturer, Release Date, Register Date, Rental Fee, Copies}	V2={ Basic Attribute + Author, Genre, Loan Period }	물품등록 알고리즘 변경(V2)선택	Type, Memo Attribute 제거
3	Registering Product Algorithm		V2={Register (Basic Attribute + Author, Genre, Loan Period)}		Type, Memo Parameter 제거 입력 Parameter로직 구성
...	...	...		...	...

한 의사결정 해소 모델을 정의한다. 도서 대여 관리 시스템을 위해 정의된 의사결정 해소 모델은 표 3과 같다.

5.2 단계 2. 핵심 자산 인스턴스화

5.2.1 활동 2A. 아키텍처 가변성 인스턴스화

단계 1의 산출물인 의사결정 해소 모델을 이용하여 아키텍처 가변성을 인스턴스화 한다. 아키텍처의 복사본 생성 후 아키텍처를 구성하고 있는 컴포넌트와 컴포넌트간의 관계를 재 정의하며, 불필요한 컴포넌트를 제거한다. 표 3의 의사결정 해소 모델에 정의된 아키텍처 가변성의 가변치는 V2 = { VIPmember component 사용 안함}이므로 C_VIPmember 컴포넌트를 사용하지 않는다. 따라서 그림 12와 같이 C_VIPmember 컴포넌트를 제거한다.

5.2.2 활동 2B. 컴포넌트 가변성 인스턴스화

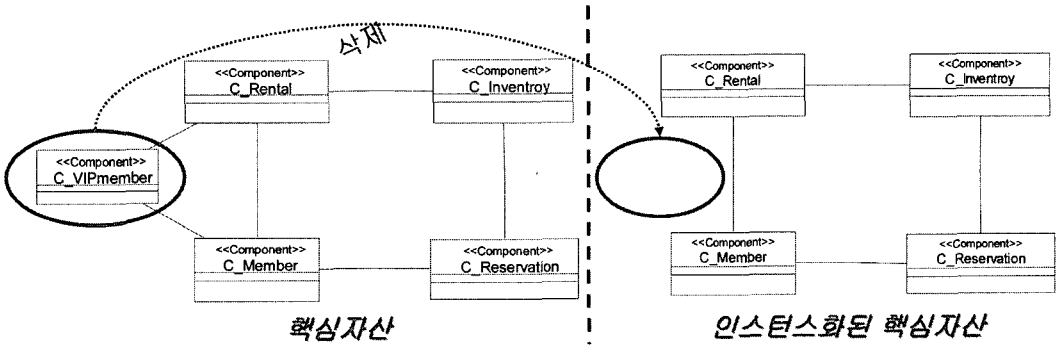

그림 12 아키텍쳐 가변성 인스턴스화

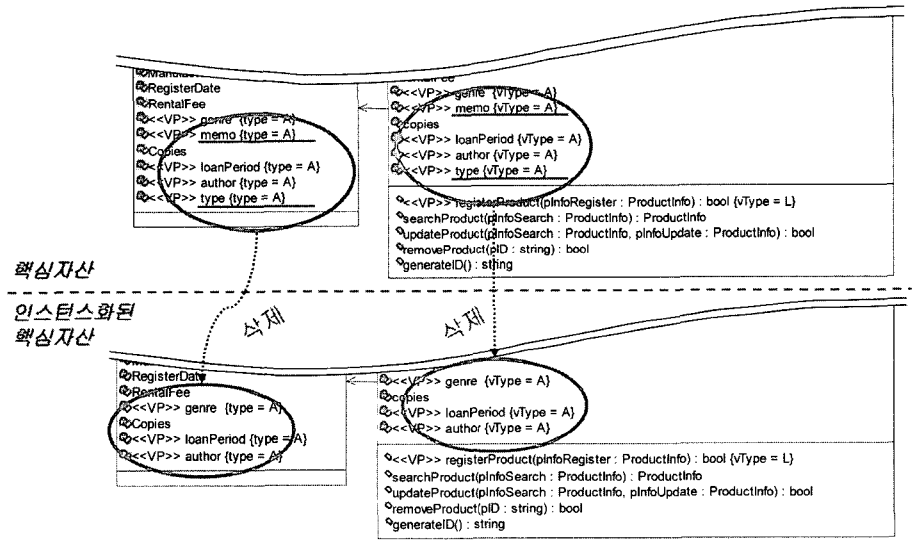

그림 13 컴포넌트 가변성 인스턴스화

단계 1에서 정의한 의사결정 해소 모델을 이용하여 컴포넌트의 가변성을 인스턴스화 한다. 그림 13과 같이 도서 관리 시스템에 불필요한 가변치를 제거한다. 표 3의 의사결정 해소 모델에 정의된 컴포넌트 가변성의 가변치는 V2={Basic Attribute + Author, Genre, Loan Period}, V2 = {VIPmember component 사용 안함}이므로 불필요한 memo{type=A},type{type=A}를 제거한다.

5.3 단계 3. 어플리케이션 종속 휘쳐 설계

5.3.1 활동 3A. COTS 컴포넌트 조사/구매 결정

목표 어플리케이션에 필요한 SummaryReport기능에 대하여 COTS컴포넌트가 존재하는지를 조사한다. 본 사례연구에서는 목표어플리케이션에서 원하는 COTS 컴포넌트가 없으므로 활동 3B을 수행한다.

5.3.2 어플리케이션 종속 휘쳐 모델링

그림 14와 같이 어플리케이션 종속 휘쳐 명세서를 이용하여 SummaryReport기능에 대한 어플리케이션 종속 휘쳐를 설계한다. 어플리케이션 종속 휘쳐 설계는 인스턴스화된 핵심자산을 이용하여 수행하며 인스턴스화된 핵심자산과 어플리케이션 종속 휘쳐와 통합 하면서 불일치가 최소화 될 수 있도록 설계를 한다. 어플리케이션 종속 휘쳐 설계를 하면서 컴포넌트간의 통합이 발생하는 매핑점을 어플리케이션 종속 휘쳐 명세서에 명시한다.

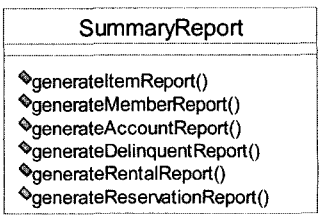

그림 14 어플리케이션 종속 휘쳐

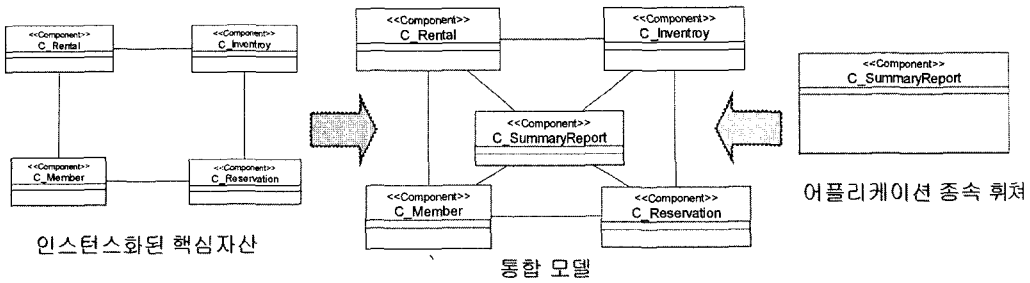

그림 15 통합 모델의 아키텍처

5.4 단계 4. 모델 통합

5.4.1 활동 4A. 특화된 핵심자산과 어플리케이션 종속 휘쳐 모델 통합

어플리케이션 종속 휘쳐 모델 통합일 경우 두 컴포넌트간의 통합은 그림 15와 같이 이미 휘쳐 모델 설계 단계에서 이루어 졌으므로 이 단계에서는 C_Summary Report 컴포넌트 설계 시 작성된 매핑점을 이용하여 인스턴스화된 핵심자산과 C_SummaryReport 컴포넌트 간의 통합이 정상적으로 이루어졌는지 검증활동을 수행한다.

5.5 단계 5. 구현

5.5.1 활동 5A. 코드 생성 및 테스팅

단계 4에서의 산출물인 도서 대여 시스템을 위한 모델 통합 명세서를 이용하여 코드를 생성한다. 생성된 코드를 단위 테스팅, 통합 테스팅, 시스템 수행 후 어플리케이션을 완성한다.

6. 결 론

PLE은 핵심자산을 이용하여 어플리케이션을 경제적이며 효과적으로 개발하기 위한 재사용 기법이다. 기존의 연구에서는 핵심자산을 개발하기 위한 절차나 활동, 산출물에 대한 연구가 많이 이루어진 반면에 어플리케이션을 개발하기 위한 연구는 많이 부족하다. 또한 AE 과정의 중요한 활동인 인스턴스화 활동뿐만 아니라 어플리케이션 종속 휘쳐와 인스턴스화된 핵심자산간의 통합을 위한 연구가 많이 부족하다.

본 논문에서는 어플리케이션과 핵심자산간의 Gap분석 활동을 위한 상세지침을 제공하였으며 핵심자산을 인스턴스화 하기 위한 실용적인 활동 및 산출물을 제시하였다. 또한 핵심자산에서 제공하지 않는 어플리케이션 종속 휘쳐 설계를 위한 체계적이며 실용적인 알고리즘과 활동을 제시하였으며 인스턴스화된 핵심자산과 어플리케이션 종속 휘쳐간의 통합을 위한 구체적 지침을 제시하였다. 마지막으로 본 논문의 프로세스를 적용한 사례연구를 통하여 프로세스와 산출물에 대한 실용성을 제공한다. 본 논문에서 제시한 활동과 지침은 PLE을 이용한 어플리케이션 개발을 좀더 실용적으로 접근할 수 있으며, 비용과 시간을 효율적을 사용할 수 있다.

참 고 문 헌

[1] Deelstra, S., Sinnema, M., and Bosch, J., "A Product Derivation Framework for Software Product Families," *Proceedings of PFE2003, LNCS 3014*, Springer, pp. 473-484, 2004.

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

[3] Atkinson, C., et al., *Component-based Product Line Engineering with UML*, Addison Wesley, 2001.

[4] Clements, P., et al., *Documenting Software Architectures Views and Beyond*, 2003.

[5] Gomma, H., *Designing Software Product Lines with UML from Use Cass to Pattern-Based Software Architectures*, Addsion-Wesley, 2004.

[6] Choi, S., et al., "A Systematic Methodology for Developing Component Frameworks," *Lecture Notes in Computer Science 2984, Proceedings of the 7th Fundamental Approaches to Software Engineering Conference*, 2004.

[7] 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 publication 456, VTT Technical Research Center of Finland, ESPOO2002*, 2002.

[8] Kang, K., Kim, S., Lee, J., Kim, K., Shin, E. and Huh, M., "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," *Annals of Software Engineering, vol.5*, p.143- p.168, 1998.

[9] Anastasopoulos, M., Bayer, J., Flege, O., and Gacek, C., *A Process for Product Line Architecture Creation and Evaluation PuLSE-DSSA-version 2.0*, Technical Report, No. 038.00/E, IESE, June 2000.

[10] Pressman, R., *Software engineering: A Practitioner's Approach*, Mc Graw Hill, 2001.

장 치 원

2004년 대전대학교 컴퓨터학과 공학사
2004년~현재 숭실대학교 컴퓨터학과 석사과정. 관심분야는 제품계열 공학(PLE), 컴포넌트 기반 개발(CBD), 모델 기반 아키텍처(MDA)

장 수 호

정보과학회논문지 : 소프트웨어 및 응용
제 32 권 제 4 호 참조

김 수 동

정보과학회논문지 : 소프트웨어 및 응용
제 32 권 제 4 호 참조