소프트웨어 제품계열 아키텍처 개발 프로세스

오영배

수원여대 디지털미디어학부

yboh@suwon-c.ac.kr

Development of a Software Product-Line Architecture Process

요 약

S/W 제품계열(S/W Product Line)은 공통의 유사한 기능을 지닌 S/W 제품 또는 S/W 시스템의 집합을 의미한다. S/W 제품 계열을 통해 특정 영역의 시장과 용도의 요구사항을 만족하여 특정 S/W 제품 개발시 미리 구축된 S/W 아키텍처 등의 S/W 핵심 자산을 재사용하여 개발한다. S/W 제품계열 기반의 S/W 개발 방식은 미리 구축된 S/W 핵심자산을 재사용함으로, 처음부터 전체 시스템을 개발하는 방식보다 쉽고, 빠르게 S/W를 생산할 수 있다. S/W 기술 선진국들은 S/W 제품계열을 S/W 생산기술의 핵심 분야로 선장하고 중점적으로 기술 개발을 지원하고 있다. 미국의 CMU/SEI는 산업체 및 국방성과 함께 제품계열 프레임워크 4.0 (Product-Line Framework 4.0)을 개발하였고 유럽은 ITEA(IT for European Advancement) 프로그램에서 제품계열 기술 개발을 지원하고 있다. 그러나 국내의 경우 S/W 개발의 생산성 향상 방안으로 제품계열 기반 S/W 생산기술의 필요성을 인식하고 있으나, 기술 개발 투자는 미흡한 상황이다. 본 논문에서는 이러한 S/W 제품계열 생산을 위한 S/W 제품계열의 공통 아키텍처를 정립하는 것을 목표로 하고 있다.

1. 서 론

S/W 제품계열(S/W Product Line)은 공통의 유사한 기능을 지닌 S/W 제품 또는 S/W 시스템의 집합을 의미한다. S/W 제품 계열을 통해 특정 영역의 시장과 용도의 요구사항을 만족하여 특정 S/W 제품 개발시 미리 구축된 S/W 아키텍처 등의 S/W 핵심 자산을 재사용하여 개발한다. S/W 제품계열 기반의 S/W 개발 방식은 미리 구축된 S/W 핵심자산을 재사용함으로, 처음부터 전체 시스템을 개발하는 방식보다 쉽고, 빠르게 S/W를 생산할 수 있다.

S/W 기술 선진국들은 S/W 제품계열을 S/W 생산기술의 핵심 분야로 선장하고 중점적으로 기술 개발을 지원하고 있다. 미국의 CMU/SEI는 산업체 및 국방성과 함께 제품계열 프레임워크 4.0 (Product-Line Framework 4.0)을 개발하였고[10] 유럽은 ITEA(IT for European Advancement) 프로그램에서 제품계열 기술 개발을 지원하고 있다[11]. 그러나 국내의 경우 S/W 개발의 생산성 향상 방안으로 제품계열 기반 S/W 생산기술의 필요성을 인식하고 있으나, 기술 개발 투자는 미흡한 상황이다.

본 논문에서는 이러한 S/W 제품계열 생산을 위한 S/W 제품계열의 공통 아키텍처를 정립하는 것을 목표로 하고 있다. S/W 제품계열 공통 아키텍처란 특정 S/W 제품계열 내의 전체 제품들은 대표하는 S/W 아키텍처를 의미하며, 일명 제품계열 아키텍처라고도 정의한다. 이러한 S/W 제품계열 아키텍처는 특정 제품 개발 시, 제품계열 아키텍처를 요구사항에 따라 재정의하여 제품 아키텍처를 정의하고, 이에 따라 필요 기능의 컴포넌트를 조정, 조립 또는 신규 개발하여 제품의 빠른 생산을 가능케 할 수 있는 가장 중요한 S/W 핵심자산이다.

2장에서는 현재까지 개발되어진 S/W 개발 방법론에 대해 기술하며 3장에서 아키텍처 기반의 제품계열 아키텍처를 제시하였다. 4장에서는 기존의 제품 계열 아키텍처 방법론과 본 논문에서 제시하는 방법론을 비교 평가하였다.

2. 관련 연구

컴포넌트 기반 개발 , 아키텍처 스타일과 디자인 패턴 그리고 제품계열 방법 등의 S/W 개발 방법은 최종적으로 소프트웨어 시스템 개발에 있어서 재사용성을 증가 시키는 것을 목적으로 개발되었다. 그러나 실제 문제에 적용 가능한 효율적인 개발 방법 개념을 위해서는 반드시 시기와 장소, 방법을 가이드 해주는 개발 방법론이 필요하다. 현재까지 개발된 개발 방법론들은 다음과 같은 단계로

발전하였다.

2.1 1세대 객체지향 방법론

현대의 대부분의 S/W 개발 방법론들은 객체 지향 방법론을 기반으로 한다. 객체지향 방법론이 직접적으로 컴포넌트나 디자인 패턴, 그리고 제품계열을 고려하여 작성되지 않았으며 도한 정확한 영향역이나 공헌의 정도를 수치화 하기는 어렵지만 다른 방법론의 개발과 통합에 많은 도움을 준 것은 확실하다.

● OMT (Object Modeling Technique)

Ggeneral Electric research Lab에서 개발한 OMT는 객체 지향 개발 방법론의 발전에 가장 큰 영향을 미친 방법론으로써 이후에 개발된 방법론과 표현 방법에 있어서 많은 아이디어를 제공하였다[2]. OMT는 중복 가능한 객체, 다이나믹, 기능의 세 가지 모델로 구분되며 각 모델은 시스템 분석 단계에서 생성된다.

● Fusion

Fusion은 OMT의 파생형으로써 기존 OMT의 분석 접근상의 문제점을 해결하였으며 디자인 프로세스를 개선하였으며 또한 객체 지향 방법론의 규정을 엄격히 하는데 초점을 두었다[5]. Fusion은 OMT와 같은 세 가지의 기본적인 분석 뷰 포인트를 채용하였지만 모델과 용어에 있어서 약간 다른 방식을 취한다.

● ROOM (Real-Time Object-Oriented Modeling)

Fusin과 같은 시기에 개발된 ROOM은 OMT나 Fusion과는 다르게 데이터 모델링과 구조적 개발 전략을 통합한 표현을 사용하며 뷰는 프로세스의 상호작용을 기반으로 한다[4]. 액터로 알려진 ROOM의 기본 빌딩 블록은 활동 쓰레드 또는 프로세스 뿐 아니라 상태 정보까지 은닉화 함으로써 실제 구현단계에서 자바의 쓰레드나 Ada의 태스크와 같이 상호 대응되는 구조를 가진다.

● HOOD (Hierarchical Object-Oriented Design)

HOOD는 다른 개발 방법론들과 비교하여 널리 사용되지는 않았지만 다른 방법론들에서 제공하지 않는 강력하고 독특한 개념을 포함한다[3]. HOOD는 상위, 처리 계층의 2가지 계층에 존재하는 객체들의 공동체로서 시스템을 정의한다. 상위 계층은 다른 객체들에 대한 제한사항을 트리구조로써 구성하며, 처리계층은 객체내부의 처리를 그래프 구조로 반영한다. 불행히도 HOOD는 객체 지향의 기본 기념인 객체 패러다임, 상속, 다행성 등의 부족으로 널리 사용되지는 못하였다.

● OORAM

OORAM은 HOOD와 같이 널리 사용되지는 않았지만 시스템의 역할 모델링에 관한 중요한 아이디어를 제공하였다[4]. OORAM은 시스템의 다른 관점을 모델링 프로세스 전반에 걸쳐 역할로 초점을 맞추었다. 다른 객체 지향 방법론들이 시스템과 외부 에이전트와의 관계를 역할로 통합하는데 반해 OORAM은 역할을 모든 객체 사이의 상호작용으로 시스템 내에서 표현할 수 있도록 확장한다. 하지만 OORAM에서 기본적으로 폭포수 모델을 채택하였기 때문에 분석과 디자인과 같은 고수준에서 반복을 요구하는 단점을 가진다.

2.2 2세대 컴포넌트 지향 방법론

컴포넌트 패러다임은 대부분의 현대 방법론에서 하나의 중요한 형태의 하나로 사용되었지만 일반적으로 S/W 개발 전반에 걸친 통합을 위하기 보다는 구현을 위한 편의성을 제공하기 위한 하나의 뷰로서 사용되었다. 하지만 최근 몇 년간 단순한 바이너리 코드 모듈 코드가 아니라 더 풍부한 추상화를 위한 뷰로서 개발 프로세스 전반에 걸쳐 사용하는 방법론이 탄생하였다.

● Catalysis

Catalysis는 컴포넌트 지향 방법론에 UML을 도입한 최초의 개발 방법론 중의 하나로써 다른 재활용 기술들에서 언급한 아키텍처 스타일, 디자인 패턴, 프레임 등의 개념을 포함한다[7]. Catalysis에서는 현재 당연하게 생각되어지는 컴포넌트 지향 방법론의 요소들을 소개하였을 뿐만 아니라 여러 부분에서 UML을 엄격하게 적용하였다. Catalysis의 주목할만한 특징은 컴포넌트의 구문적인 측면과 더불어, OCL과 같은 제약어를 사용하여 컴포넌트의 의미적인 측면을 함께 기술할 수 있는 방안을 제시했다는 것이다. Catalysis는 모델링 개념, 모델링 수준, 원칙의 기본적인 3가지 범위로 구성되며, 각 범위들은 다시 세 가지의 기본 아이디어로 이루어진다.

● Select Perspective

Select Perspective는 Catalysis과 비슷한 시기에 만들어진 컴포넌트 지향 개발 방법론이다[8]. Select Perspective는 비지니스 프로세스 모델링의 중요성을 강조하고 있으며, 비지니스 프로세스를 구현 모델로 단계적으로 전환하기 위한 명확한 프로세스를 정의하고 있다. 프로세스에는 명시적인 컴포넌트 식별 뿐 만이 아니라, 레가시 시스템을 통합하기 위한 프로세스도 포함된다. Select Perspective의 가장 큰 강점은 실제 현장에서의 경험을 통해 만들어졌기 때문에 실용적인 지침과 안내서가 매우 풍부하다는 점이다. 또한, 방법론과 더불어 컴포넌트 설계/개발 및 관리도구, 프로세스 관리도구, 그리고 솔루션 등 상업적인 도구의 지원이 풍부하다는 점은 Select Perspective를 더욱 실용적으로 만든다.

● UML Components

UML Components는 소프트웨어 컴포넌트의 핵심 개념 중에 하나인 컴포넌트 명세 작성을 위한 가장 간단하고 실용적인 프로세스를 제공한다[11]. 또한 UML Component는 Catalysis와 Advisor 방법론에서 큰 영향을 받았으며, RUP를 부분적으로 수용하고 있으며 관리 프로세스와 분리되어 있어 컴포넌트 시스템의 아키텍처와 의존성을 명세화하는 방법을 집중적으로 다루고 있다. UML Components는 컴포넌트를 위한 많은 개념들과 공정 및 지원 활동들이 생략되어 있지만, 컴포넌트 기반 방법론 도입에 부담을 갖기 쉬운 초보자라도 쉽게 습득할 수 있다는 장점을 가지고 있지만 특정 회사에 종속적인 경향을 갖는 단점이 있다.

2.3 3세대 제품계열 지향 방법론

● KobrA

KobrA는 독일의 프라운호퍼 IESE가 주도적으로 개발한 컴포넌트를 기반으로 제품계열을 개발하기 위한 방법론으로 제품 계열을 위한 프레임웍을 만들어, 특정 멤버를 위한 애플리케이션을 개발할 수 있도록 하는 것을 목적으로 한다[12]. KobrA는 크게 프레임웍 엔지니어링과 애플리케이션 엔지니어링으로 구성되며, 프레임웍 엔지니어링 단계에서 시스템 계열의 공통적인 부분은 물론 가변성을 추가하여 프레임웍을 만들게 되고, 애플리케이션 엔지니어링 단계에서는 이러한 가변성을 제거하여 특정 멤버의 요구사항에 맞는 애플리케이션을 개발하게 된다. 또한, 상화, 일반화, 그리고 조립의 정도에 따라서 세 개의 독립적인 차원으로 구성하여, 모델이 어떻게 구체적인 형태로 바뀔 수 있는지 체계적으로 표현 가능하다.

● FORM

특징 관점에서 도메인 내 응용들의 공통성과 차이점을 추출하고, 특징들을 도메인 아키텍처와 컴포넌트 개발을 위한 분석 결과로 활용하기 위한 체계적인 메소드를 제시한다[1]. FORM에서는 특징들이 설계, 구현 단계로 전개되는 과정을 S/W 공학 원칙들에 따라 설명함으로써 구체적인 참조 아키텍처 및 재사용 컴포넌트 생성 방법을 제시할 수 있는 장점이 있다.

● PuLSE에 대한 소개 및 장단점

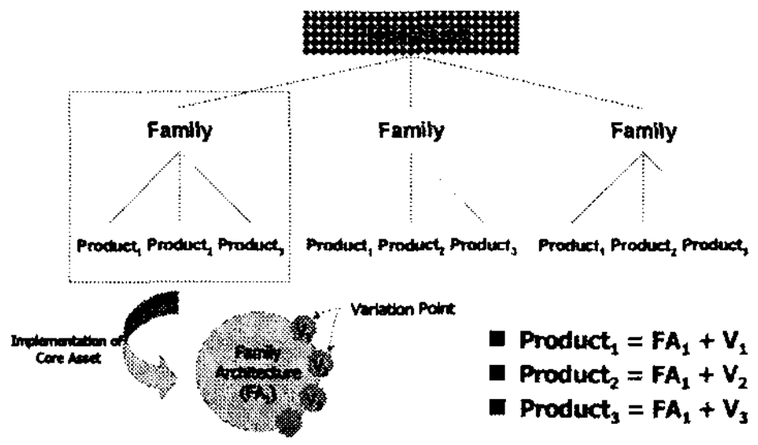
PuLSE는 제품 계열 개발을 위한 전체 생명주기를 지원하는 프레임워크로써 독일 IESE에서 개발되었다[9]. PuLSE에서는 개별 고객의 특수한 관점을 고려할 수 있는 맞춤형 프로세스를 지원하며, S/W 개발시 점진적으로 제품계열을 도입하거나, 이를 진화시킬 수 있는 실제적인 행위들을 제공한다. 또한 특정 제품의 요구 충족을 실체화 할 수 있는 6개의 기술적 컴포넌트를 제공한다.

3. 아키텍처 기반 제품계열 아키텍처 개발 방법론

3.1 고려 사항

제품계열 공학(Product Line Engineering)의 목표는 앞서 언급하였듯이 개발하려는 소프트웨어 시스템의 공통성(commonality)과 구별되는 특성, 즉 가변성(variability)을 이해하고 조절함으로써 구조적인 개발을 보장하기 위함이다. 아키텍처 수준(level)에서 공통성은 제품계열의 구조적 골격을 정의하고 가변성은 공통 골격에 요구되어지는 가변 요소를 정의한다. 제품계열 아키텍처 설계시 이러한 공통성/가변성을 판단하기 위한 결정 모델(decision model)을 필요로 하며 결정모델을 통하여 제품 계열 아키텍처의 진화를 돕고 어플리케이션 개발자가 새로운 어플리케이션을 생산하기 위한 가이드라인을 제시한다.

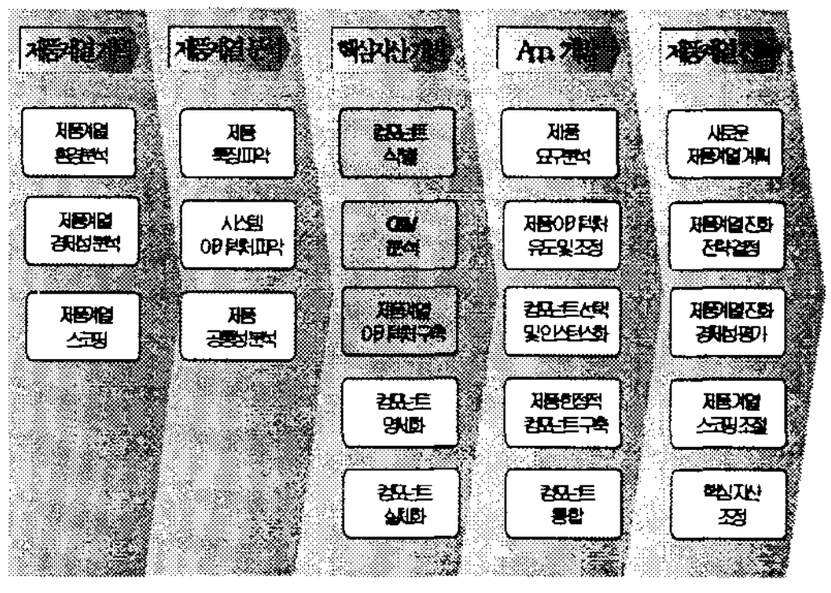
3.2 아키텍처 기반 제품계열 아키텍처 설계



<그림 1> 제품 계열 아키텍처의 개념도

<그림 1>은 제품 계열 아키텍처의 개념을 나타낸 것이다. 그림에서 표현된 것과 같이 제품 계열은 패밀리(Family)들의 집합으로 이루어져 있으며 각 패밀리는 제품군으로 구성된다[10]. 각 제품 아키텍처는 공통성과 가변성을 포함하고 있으며 하나의 새로운 제품을 생성하기 위해서 구축되어 있는 제품 아키텍처의 공통사항과 가변사항의 조합으로 생성된다.

<그림 2>는 제품계열 생산기술 개발을 위한 전체 개발과정을 도식화한 것이다.

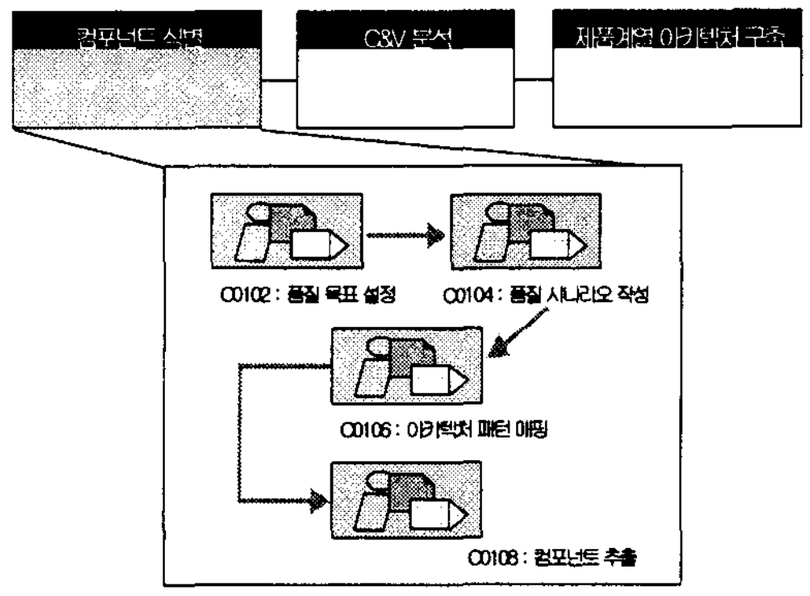


<그림 2> 제품계열 생산기술 개발 프로세스

<그림 2>에서 볼 수 있듯이 제품 계열 생산기술 개발 프로세스는 제품계열 계획, 제품계열 분석, 핵심자산 개발, 어플리케이션 개발, 제품계열 진화의 크게 5개의 영역으로 나뉜다. 본 논문에서는 핵심자산 개발 영역에서 제품 계열 아키텍처를 구축하기 위한 활동, 즉 컴포넌트 식별, C&V 분석, 제품 계열 아키텍처 구축에 중점을 두었다.

3.2.1 컴포넌트 식별

컴포넌트 식별 활동(activity)에서는 품질 목표를 달성하기 위한 기능 단위의 컴포넌트 추출을 목적으로 한다. 이를 위해 품질 시나리오를 적용하여 작성된 아키텍처 패턴을 통해 컴포넌트 리스트를 추출하기 위한 패턴 매핑 방법을 정립한다. <그림 3>은 컴포넌트 식별 활동에서의 세부 작업(task)를 나타낸다.

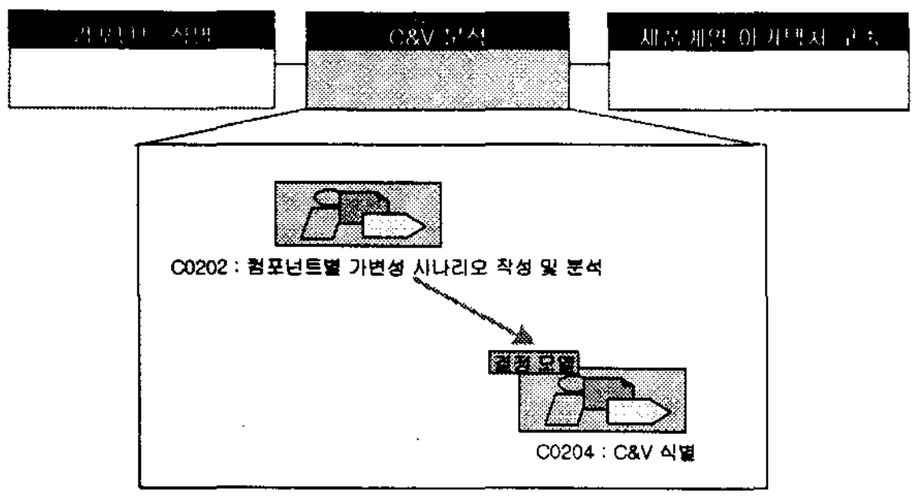


<그림 3> 컴포넌트 식별 활동

컴포넌트 식별 활동은 크게 4개의 작업으로 이루어진다. 우선 이해 당사자들이 가지고 있는 도메인의 품질속성을 추출함으로써 구축하려는 제품계열 아키텍처의 가장 기본적인 품질목표들을 추상적으로 정의하는 작업(C0102)이 수행된다. 품질 목표를 이루기 위한 가능성 있는 상황들을 도출하여 도출된 가능 상황들을 바탕으로 시나리오를 작성(C0104)하고 정의된 품질 시나리오를 구현하기 위한 전략을 구조적 방법으로 표현한다. 구조적으로 표현된 아키텍처 패턴을 생성하기 위해 시나리오에 대한 세분화된 매핑방법을 도출하고 그룹핑한다(C0106). 마지막으로 아키텍처 패턴의 매핑을 통해 기능단위의 컴포넌트를 추출한다(C0108).
이러한 컴포넌트 식별 활동을 통해 품질 목표, 품질 시나리오, 패턴 및 컴포넌트 정의서와 같은 산출물들이 생성된다.

3.2.2 C&V(Commonality & Variability, 공통성/가변성) 분석

C&V 분석 활동에서는 추출된 컴포넌트들에 대한 C&V 분석을 통해 C&V 결정모델을 작성하고 각 컴포넌트와 인터페이스에 대한 공통성과 가변성을 식별한다. <그림 4>는 C&V 분석 활동에서의 세부 작업을 나타낸다.



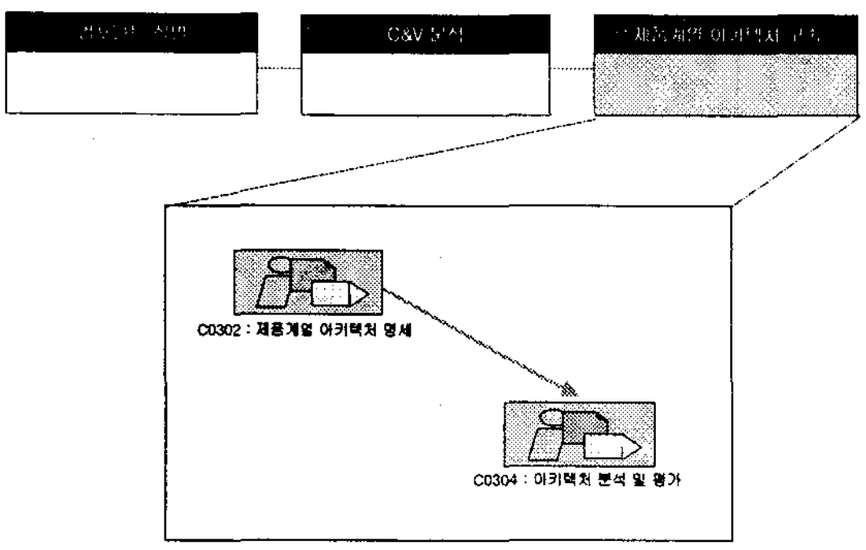
<그림 4> C&V 분석 활동

C&V 분석 활동은 크게 2개의 작업으로 이루어진다. 우선 이해 당사자들의 참여를 통해 기능적 요소와 예측되는 변경 가능요소에 대한 정밀한 시나리오 작성을 유도한다. 컴포넌트별 시나리오 분석을 위하여 쓰임새도(usecase diagram)와 활동도(activity diagram) 작성 및 기능 요구분석을 수행한다. 또한 컴포넌트 시나리오를 바탕으로 각 컴포넌트에 대한 기능 요구사항을 분석한다(C0202). 분석된 컴포넌트 시나리오를 바탕으로 컴포넌트와 인터페이스에 대한 공통성/가변성을 식별한다(C0204).

C&V 분석 활동을 통해 쓰임새 및 활동도, 각 컴포넌트 단위별 시나리오, 요구사항 명세서, 컴포넌트와 인터페이스의 공통 요소와 가변 요소를 분리하기 위한 결정 모델(decision model), 가변성을 가지는 컴포넌트와 인터페이스 목록을 표현한 가변점 목록(variation point list)과 같은 산출물이 생성된다.

3.2.3 제품계열 아키텍처 구축

제품계열 아키텍처 구축 활동에서는 제품 패밀리 전체 멤버들에 대해 적용할 수 있는 공통의 참조 아키텍처를 구축하고 아키텍처에 대한 공통/가변 요소를 명세한다. <그림 5>는 제품계열 아키텍처 구축 활동에서의 세부 작업을 보여준다.



<그림 5> 제품계열 아키텍처 구축 활동

제품계열 아키텍처 구축 활동은 제품계열 아키텍처 명세(C0302)와 아키텍처 분석 및 평가(C0304)의 작업으로 구성된다. 제품계열 아키텍처 명세에서는 모듈 아키텍처 뷰, 프로세스 아키텍처 뷰, 배치 아키텍처 뷰를 나타내기 위한 가이드라인을 사용하여 아키텍처 명세 및 가변적 명세 방법을 기술한다. 아키텍처 분석 및 평가 작업에서는 선행 작업에서 작성된 아키텍처를 분석 및 평가 지침서를 통해 제품계열 아키텍처를 분석 평가한다.
제품계열 아키텍처 구축 활동을 통해 아키텍처의 구조와 프로세스를 명세한 제품계열 아키텍처 명세서, 아키텍처 평가 결과서를 얻을 수 있다.

4.비교평가

본 절에서는 기존의 PLE 방법론과 본 논문에서 제시하는 제품 계열 아키텍처 구축 방법론과의 비교 평가를 하였다.
FAST/PASTA는 도메인 엔지니어링을 통한 패밀리의 도출 및 모델로부터 제품 개발을 위한 반복 및 피드백 개발을 강조하고 있고, FORM은 도메인 내 응용들의 공통성과 변경성 추출을 위한 특징(feature) 모델을 이용하는 것이 중심이 된다. KobrA는 프레임웍 엔지니어링과 애플리케이션 엔지니어링으로 구분되는 각 단계에서 제품계열 방법을 적용하는 프로세스를 제안하고 있고, PulSE는 KobrA에 기반을 두고 소프트웨어 개발시 점진적 개발을 위한 제품계열 방법을 제시하고 있다. PoLITE는 컴포넌트 기반의 제품계열 적용을 위한 방법을 제시하고 있는데 반해 본 논문이 제시하는 방법은 품질 목표 분석 및 C&V 분석을 통한 제품계열 아키텍처를 도출하고 아키텍처를 기반으로 하는 제품계열 방법을 제시하였다는 것이 기존 방법과 가장 큰 차이며 장점이다.
각 방법론이 가지고 있는 프로세스(Process), 산출물(artifact), 그리고 특이점(special feature)과 같이 크게 3개의 평가 기준을 이용하여 비교 평가 하였으며 그 결과는 다음 표에 제시한다.

4.1 프로세스비교(Comparison by Process)

제품 계열 아키텍처 구축을 위한 프로세스를 도메인 획득, 핵심자산 개발, 어플리케이션 개발, 진화 및 유지 보수, 프로젝트 관리의 크게 5가지 평가기준을 이용하여 비교 하였다. 본 논문에서 제안하는 방법론은 도메인의 품질속성을 추출함으로써 제품계열 아키텍처가 가져하여하는 품질목표를 설정하

고 아키텍처를 기반으로 핵심자산 개발하는 방법을 제안하였다. 표 1 은 기존의 방법론들과 제안 방법론을 여러 프로세스에 의해 비교 평가한 것이다.

[표 1] 프로세스 비교

평가 기준	FAST PASTA	FORM	PuLSE	KobrA	PoLITe	제안 방법론
도메인 획득	Qualifying Domain inc. Biz. Case Analysis	Scoping only (Context Analysis)	PuLSE-Eco	None on Scoping, Biz. C.Anal.	None	Quality Goal & Scenario
핵심자산 개발	Engineer Domain	Domain Engineerin (including Feature Modeling)	PuLSE-DCA, -DSSA	Framework Engineering	No Modeling	Architecture Based
어플리케이션 개발	Engineer Application	Application Engineering	PuLSE-I	Application Engineering	Instantiation Only (2types)	Instantiation
진화 및 유지보수	Change Family	None	PuLSE-EM	EM (Evolution Graph)	Evolution of Implementation	Evolution of Architecture
프로젝트 관리	Manage Project	None	Support Components	Project Management	Configuration Management	Manage Project

4.2 산출물비교(Comparison by Artifacts)

본 논문에서 제안하는 아키텍처 개발 방법론은 특정 제품 계열 개발을 위한 공통 아키텍처를 제시해 줄 뿐만 아니라 컴포넌트 정의서, 컴포넌트의 공통성/가변성을 식별을 위한 결정모델을 제시하여 준다. 또한 다른 개발 방법론에서는 제시되지 않는 핵심 자산이 메타모델을 제시하며 그 밖에 컴포넌트 식별을 위한 시나리오 및 패턴을 추출한다. 표 2 은 기존의 방법론들과 제안 방법론을 산출물 의해 비교 평가한 것이다.

[표 2] 산출물 비교

평가 기준	FAST PASTA	FORM	PuLSE	KobrA	PoLITe	제안 방법론
공통 아키텍처	None	Reference Architecture	Reference Architecture	Implied	Component Architecture	Reference Architecture
컴포넌트 (설계와 구현 수준)	Family Design (Modules)	Subsystem, Process, Module	Domain Model Work Products	Komponents and Components	Component at Impl. Level	Definition Document of Component
결정 모델	Decision Model	Included in Feature Model	Decision Model	Decision Model	None	Decision Model
핵심 자산의 메타 모델	None	None	None	Well Defined	None	Defined
기타 산출물	Application Modeling Language Tools	Feature Model including Quality attributes	Baseline Profile for Evolution Environment	Evolution Graph Komponent	Refine pattern Translation pattern	Quality Goal, Pattern, Scenario

4.3 특이점 비교 (Comparison by Special Features)

본 논문에서 제안하는 아키텍처 개발 방법론은 제품군으로 이루어진 패밀리에 대한 공통 아키텍처를 구축하고 재사용함으로써 특정 제품 계열을 생산하는데 있어 전체 시스템을 쉽고 빠르게 생산할 수 있도록 한다. 표 3 은 기존의 방법론들과 제안 방법론을 특이점 의해 비교 평가한 것이다.

[표 3] 특이점 비교

평가 기준	FAST PASTA	FORM	PuLSE	KobrA	PoLITe	제안 방법론
상 (Significant)	Domain specific A.E. process defined	Featue Model	Process tailored Baseline profile	Precise definition of artifacts	None	Architecture by Family
중 (Moderate)	AML Environment	None	None	3 Levels of Komponents	Implementation Level Framework	Implementation by Architecture
하 (Minor)	None	None	None	UML compliant Variability using sterotypes	Utilizing MDA KobrA-based UML profile needed	None

5.결론및향후연구

본 논문에서는 제품의 다품종 생산 환경에 적응하기 위한 S/W 개발 방법론으로써 제품계열 아키텍처 설계를 제안하였다. 또한 제안한 제품계열 아키텍처를 기존의 제품계열 아키텍처 방법론과 비교하여 산출물과 프로세스의 향상점을 확인하였다.

향후 연구로는 본 논문에서 제시한 각 단계의 세부적인 프로세스를 정의하고 산출물에 대한 실용가능한 템플릿을 작성할 예정이다.

6.참고문헌

[1] Kang, K.C., Cohen, S.G.., Novak, W.E. and Peterson, A.S., "Feature-oriented Domain Anlaysis(FODA) Feasibility Study, "Technical Report CMU/SEI-90-TR-21, SEI, November 1990.

[2] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W., Object-Oriented Modelling and Design, Prentice Hall, 1991.

[3] Robinson, P. J., Hierarchical Object-Oriented Design, Prentice Hall, 1992.

[4] Reenskaug, T., Wold, P. and Lehne, O., Working with Objects: The OOram Software Development Method, Manning/Prentice Hall, 1996.

[5] Selic, B., Gullekson, G. and Ward, P.T., Real-Time Object-Oriented Modeling, John Wiley and Sons, 1994.

[6] Coleman, D., Arnold, P., Bodoff. S., Dollin, C., Gilchrist, H., Hayes, F. and Jeremaes, P., Object-oriented Development. The Fusion Method, Prentice Hall, 1994.

[7] D'Souza, D.F. and Wills A.C., Objects, Components and Frameworks with UML: The Catalysis Approach, Addison-Wesley, 1998.

[8] Allen, P. and Frost, S., Component-Based Development for Enterprise Systems, Applying the SELECT Perspective, Cambridge University Press/SIGS, Cambridge, 1998.

[9] Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid. K., Widen, T. and Debaud, J., "PuLSE. A Methodology to Develop Software Product Lines, " in Proceedings of the Symposium on Software Reuse (SSR '99), May 1999.

[10] Weiss, D. and Lai, C., Software Product-line Engineering: A Family-Based Software Development Process, 1999

[11] Cheesman, J. and Daniels, J., UML Components: A Simple Process for Specifying Component-Based Software, Addison-Wesley, 2000.

[12] Atkinson, C., Bayer, J., et al, Component-based Product Line Engineering with UML, 2002

[13] Clements, P., Northrop, L, Software Product Lines:Practices and Patterns, 2002