韓國IT서비스學會誌
第5卷 第2號
2006年 8月

공통 아키텍처 기반 제품계열 아키텍처 설계

오영배* · 신성욱** · 김영갑** · 백두권***

Design of Product-Line Architecture based-on Common Architecture

Young-Bae Oh* · Sung-Oog Shin** · Young-Gab Kim** · Doo-Kwon Baik***

■ Abstract ■

Software product line is a software product or a set of software system, which has common functions. We can develop a specific software product, which satisfies requirements of a particular market segment and a mission in a specific domain by reusing the core asset such as the developed software architecture through the software product line.

Software development methodology based on the software product line can develop a software more easily and fast by reusing the developed software core asset. Developed countries of software technology select software product line as a core field of software production technology, and support technology development. In case of USA, CMU/SEI (Carnegie Mellon University/Software Engineering Institute) developed product-line framework 4.0 together with the industry and the Department of Defense. Europe is supporting the development of product line technology through ITEA(IT for European Advancement) program. However, industries in our country understand the necessity of software production technology based on product line concept for the purpose of increasing productivity, it is not sufficient to invest for this technology development. In this paper, we aim to construct the common architecture of software product line for production of the software product line.

Keyword : Product Line Architecture, Product Line, Core Asset, C&V Decision Model

* 수원여자대학교 컴퓨터응용학부 교수
** 고려대학교 컴퓨터학과 대학원
*** 고려대학교 컴퓨터학과 교수

1. 서 론

소프트웨어 제품계열 공통 아키텍처란 특정 소프트웨어 제품계열 내의 전체 제품들은 대표하는 소프트웨어 아키텍처를 의미하며, 일명 제품계열 아키텍처라고도 정의한다. 소프트웨어 제품계열(Software Product Line)은 공통의 유사한 기능을 지닌 소프트웨어 제품 또는 소프트웨어 시스템의 집합을 의미한다. 대부분의 소프트웨어 제품계열에 속한 제품은 비슷한 용도의 대상과 기능을 포함한다. 따라서 소프트웨어 제품 계열을 통해 특정 영역의 시장과 용도의 요구사항을 만족하여 특정 소프트웨어 제품 개발 시 미리 구축된 소프트웨어 아키텍처 등의 소프트웨어 핵심 자산을 재사용하여 개발하는 것이 가능하다. 소프트웨어 제품계열 기반의 소프트웨어 개발 방식은 미리 구축된 소프트웨어 핵심자산을 재사용함으로, 처음부터 전체 시스템을 개발하는 방식보다 쉽고, 빠르게 소프트웨어를 생산할 수 있다.

소프트웨어 기술 선진국들은 소프트웨어 제품계열을 소프트웨어 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[5] 유럽은 ITEA(IT for European Advancement) 프로그램에서 제품계열 기술 개발을 지원하고 있다[4]. 그러나 국내의 경우 소프트웨어 개발의 생산성 향상 방안으로 제품계열 기반 소프트웨어 생산기술의 필요성을 인식하고 있으나 이에 관련된 연구가 미흡한 실정이다.

지금까지 연구된 제품계열 아키텍처들은 아키텍처 구축을 위한 프로세스들 사이의 모호성과 산출물 생산을 위한 템플릿의 미흡, 그리고 결정모델과 컴포넌트 지정 등과 같은 제품계열 아키텍처 구축 전반에 있어서의 모호성 때문에 핵심자산과 제품 아키텍처간의 관계 설정을 위한 명세 및 단계가 구체적이지 못하여 실제적인 현장 적용에 어려움이 있다.

본 논문에서는 이러한 문제를 해결하기 위해 소프트웨어 제품계열 생산을 위한 소프트웨어 제품계열의 공통 아키텍처를 구축을 위한 프로세스를 정의하고 각 프로세스별 산출물을 명확히 지정한다. 이러한 소프트웨어 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 소프트웨어 핵심자산이다.

본 논문의 구성은 다음과 같다. 제 2장에서는 기존의 소프트웨어 제품계열 방법론에 대해 기술하고, 제 3장에서는 본 논문에서 제안하는 소프트웨어 제품계열 개발 방법론의 프로세스와 산출물들을 제시한다. 제 4장에서는 본 논문에서 제시한 소프트웨어 제품계열 개발 방법론을 적용한 EMR 사례에 대해 설명한다. 마지막으로 제 5장에서는 결론 및 향후 연구해야 할 사항에 대해 기술한다.

2. 관련 연구

컴포넌트 기반 개발, 아키텍처 스타일과 디자인 패턴 그리고 제품계열 방법 등의 소프트웨어 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 정확히 제시해 줄 수 있는 방법론이 필요하다. 본 장에서는 현재까지 개발된 제품계열 개발 방법론들에 대해 기술한다.

PuLSE(Product Line Software Engineering)는 매우 다양한 기업 환경에서의 소프트웨어 제품 계열에 대한 개념과 배포를 사용하기 위하여 소프트웨어 제품 계열의 방법론 중의 하나인 독일의 프라운 호퍼의 주도로 개발되었다[3]. PuLSE는 Kobra에 기반을 두고 소프트웨어 개발 시 점진적 개발을 위한 제품 계열 방법을 제시하고 있다. PuLSE 방법론은 매우 다양한 기업 환경 내에서 소프트웨어 제품 계열의 개념과 전개를 이용 가능하게 하며,

컴포넌트의 주문화, 구조적 기업 혁신에 성숙기 정도, 일부의 주요 제품 개발 상황으로의 적응에 대한 각각의 단계를 통해 강한 제품 중심적 초점으로 얻어지게 된다. PuLSE는 다양한 기업 상황을 제공하도록 맞춤화가 되어 있는 유연한 방법론이며, 도메인 공학 방법론에 비해 맞춤화가 되어 있어서, 전개하기에 보다 더 수월하다. 그러나 미리 존재하는 프로세스나 잘 정의된 제품이 없는 환경에서는 PuLSE 방법론은 문제점을 갖게 된다. 그러한 경우에 PuLSE의 맞춤화는 존재하는 프로세스로의 제품 계열 개념의 적응보다는 기본적인 소프트웨어 공학 프로세스의 도입에 더욱 관심을 가지게 된다. 특히 미성숙된 환경에서의 프로세스 정의에 대한 노력은 상당히 많아질 수 있으며, 심지어 금지될 수도 있다.

KobrA(Komponetenbasierte Anwendungsentwicklung)는 기존의 애플리케이션은 여러 도메인끼리의 공통적인 기능이 있음에도 불구하고 도메인별로 구축되어 소프트웨어공학 관점에서 개발시간과 비용적인 측면에서 비효율적인 단점을 보완하기 위해 개발되었다[2]. 독립적이고 대치 가능한 시스템의 일부분 또는 상호 작용하는 기능적 시스템을 형성하는 독립적 단위로 정의되는 컴포넌트는 그 기능별로 요구사항에 따라 각각을 조립하여 원하는 애플리케이션을 만들 수 있기 때문에 개발 및 유지보수, 생산성 등 소프트웨어 품질을 획기적으로 향상시킬 수 있다. 또한 소프트웨어 제품 계열은 컴포넌트가 플러그인 될 수 있는 프레임워크를 제공하는 아키텍처를 기반으로 필요한 컴포넌트를 선택적으로 조립함으로써 시장의 요구사항에 맞는 시스템을 생산하는 방식을 취하는 방식으로 높은 품질의 제품을 좀 더 빠르고, 적은 비용으로 얻기 위해서 멀티 제품 계획 사이클과 교환할 수 있는 혁신적인 사용을 강조하는 접근 방식이다. KobrA 프로세스는 컴포넌트 기반과 제품 계열 공학의 개념을 모두 포괄하고 있어서 기존의 도메인 공학 방법론과 컴포넌트 기반 개발 방법론, PuLSE가 가지는 여러 문제점들을 해결 할 수 있다. 엄격한 관심의 분리와 프로세스와 산출물을 엄격히 분리하여 간결하고 체계적이며, 규모와 복잡도를 고려하여 확장이 가능하다. 가능한 일반적으로 많이 사용되는 구현 기술과 미들웨어와 호환되도록 하고, 철저히 업계 표준과 공적 표준을 따라서 보다 더 실용성을 가진다. 그러나 컴포넌트 추출에 관한 단계가 명확하게 정의되어 있지 않아서 특정 도메인에 대한 컴포넌트 기반 소프트웨어를 구축함에 있어서 컴포넌트 추출상의 모호성이 존재한다. 이에 특정 도메인에 적합하지 문제점이 있다.

PoLITe(Product Line Implementation Technologies)는 포괄적이고 재사용이 가능한 방식으로 소프트웨어를 구현하는 효율적 방법을 명확히 파악하기 위한 소프트웨어를 개발하는 구성 방법론을 제공하는 것을 목적으로 한다. 즉, 구현 레벨에서의 제품 계열 기술을 그 중심으로 한다[2]. 기존의 제품 계열 접근법은 주로 일반성의 정확한 범위를 다루는 모델을 생성하는 기능에 집중되어 있다. 즉 소프트웨어 공학 라이프사이클의 최초 기능에 집중 되어 있다. PoLITe는 제품 계열 지원을 구현 레벨에서 적용함으로서 제품 계열 방법론에 컴포넌트를 적용함에 있어 보다 더 명확히 할 수 있다. 그러나 설정 관리 솔루션을 찾는데 어려움이 있으며, 서로 다른 설정 관리 시스템 간의 상호 운영에 문제점을 갖는다. 또한 소프트웨어 제품 계열 방법론에 대한 지원과 하부 구조 모델 변화에 대한 유지 지원에 대한 방법이 제시되어 있어 않다. 구현 동안에는 여러 패턴들이 인식된 변화성에 의존하기 때문에 최신의 변화 매커니즘을 유지해야 하는 문제점이 있다.

FAST(Family-Oriented Abstraction, Specification, and Translation)은 제품군 계열 방법에서 소프트웨어를 만들어 내는 개발과정이다[13]. 제품계열 엔지니어링 프로세스를 두 가지 부분으로 나누어지는데 하나는 각각의 제품을 구현하기 위한 환경을 포함하는 핵심 자산을 제공하는데 집중하고 다른 하나는 제품군에 소속된 다른 소프트웨어 제품 생산에서의 환경을 사용한다. FAST/PASTA는 도메인 엔지니어링을 통한 패밀리의 도출 및 모델

로부터 제품개발을 위한 반복 및 피드백 개발을 강조하고 있다는 것이 큰 특징으로 볼 수 있다.

FORM(Feature-Oriented Reuse Method)는 도메인 아키텍처와 컴포넌트를 발전시키기 위한 분석 결과를 사용하고, 특징적 관점에서 도메인 안에서의 어플리케이션의 공통성과 가변성을 추출하는 것에 중점을 둔 시스템적인 방법이다[8].

3. 제품계열 아키텍처 개발 방법론

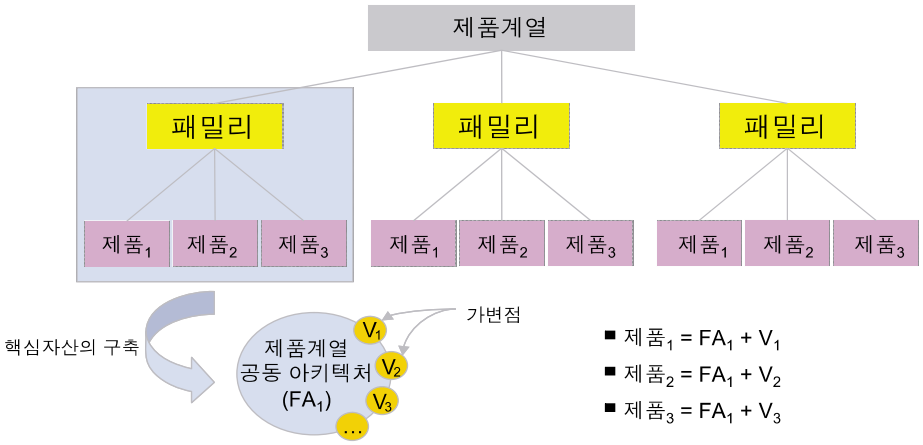
3.1 개 요

소프트웨어 아키텍처는 소프트웨어 제품 또는 응용 시스템의 전체적인 구조를 뜻하며 소프트웨어의 주요 기능 또는 서비스를 제공하는 소프트웨어 컴포넌트들과 이들 간의 관계로 표시된다. 이에 비해 소프트웨어 제품계열 공통 아키텍처는 소프트웨어 제품계열 내의 전체 제품들을 대표하는 소프트웨어 아키텍처를 뜻하며 제품계열 아키텍처라고 표현된다. 제품 패밀리(Family)내의 특정 제품을 개발할 때 제품계열 아키텍처를 요구사항에 따라 재정의 하여 제품 아키텍처를 정의하고, 이에 따라 필요 기능의 컴포넌트를 조정, 조립 또는 신규 개발하여 제품의 빠른 생산을 가능케 할 수 있는 소프트웨어 제품계열에서의 가장 중요한 소프트웨어 핵심자산이다.

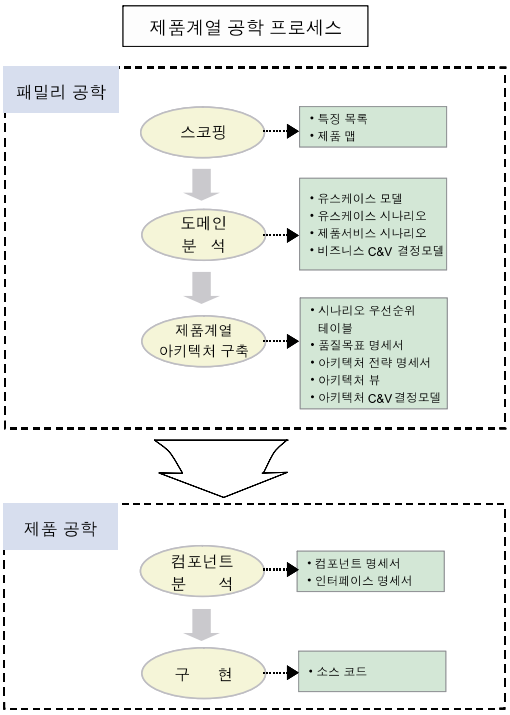
[그림 1]은 제품 계열 아키텍처의 개념을 나타낸 것이다. 그림에서 표현된 것과 같이 제품 계열은 패밀리들의 집합으로 이루어져 있으며 각 패밀리는 제품군으로 구성된다. 각 제품 아키텍처는 공통성과 가변성을 포함하고 있으며 하나의 새로운 제품을 생성하기 위해서 구축되어 있는 제품 아키텍처의 공통사항과 가변사항의 조합으로 생성된다.

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 수준에서 공통성은 제품계열의 구조적 골격을 정의하고 가변성은 공통 골격에 요구되어지는 가변요소를 정의한다. 제품계열 아키텍처 설계 시 이러한 공통성/가변성을 판단하기 위한 결정 모델(decision model)을 필요로 하며 결정모델을 통하여 제품 계열 아키텍처의 진화를 돕고 패밀리에 포함된 제품 개발자가 새로운 재품을 생산하기 위한 기준을 제시한다.

[그림 2]는 본 논문에서 제시하는 소프트웨어 제품계열 공학 프로세스의 세부 프로세스와 산출물을 표현한 전체 개요도이다.



[그림 1] 제품계열 아키텍처 개념도



[그림 2] 제품계열 공학 전체 프로세스

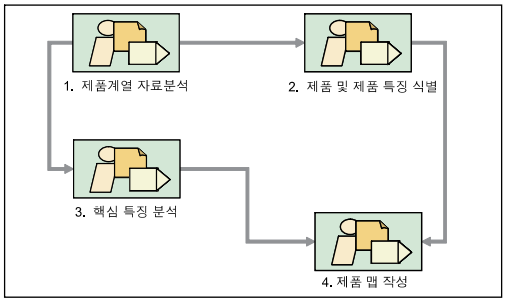
제품계열 공학 프로세스는 크게 2개의 단계와 5개의 활동으로 구분된다. 패밀리 공학(Family Engineering)은 제품공학 프로세스의 중심적인 역할을 수행하며, 제품공학 단계에서 실제 제품을 구현하기 위한 제품계열 공통 아키텍처를 구축하는 단계이다. 제품 공학(Product Engineering)은 패밀리 공학에서 작성된 핵심자산과 공통 아키텍처를 대상이 되는 제품에 따라 분석하여 실제 제품을 구현하는 단계이다.

첫 번째 활동인 스코핑(Scoping)에서는 시스템의 내부 혹은 외부에 포함되어져야만 하는 행위나 관점들의 정의를 통해 시스템의 집합을 경계 짓는 활동이다. 대부분의 경우 스코핑의 범위는 해당 제품계열이 추구하는 사업 목적에 따라 결정된다. 스코핑의 범위의 크기에 따라 제품 패밀리의 재사용성과 확장성은 상호배타적(trade-off) 관계를 가진다. 이 단계를 통해 공통아키텍처 구축을 위해 필요한 특징목록(Feature List)과 제품 맵(Product Map)이 생성된다. 도메인 분석(Domain Analysis) 활동에서는 스코핑 활동에서 추출된 특징들을 기반으로 제품계열의 비즈니스 프로세스를 구축하고 비즈니스 프로세스에서 발생할 수 있는 가변요소를 분석하기 위한 비즈니스 C&V 결정모델을 구축한다. 이 활동을 통해 비즈니스 프로세스 단위의 유스케이스 모델과 유스케이스 시나리오, 제품 서비스 시나리오를 작성하고, 비즈니스 C&V 결정모델을 작성하게 된다. 패밀리 공학에서 가장 중요한 부분을 차지하는 제품계열 아키텍처 구축 단계에서는 제품계열 공통 아키텍처와 C&V 결정모델을 구축한다. 구축된 공통 아키텍처와 C&V 결정모델을 통해 제품계열 제품을 구성하는 컴포넌트를 추출한다. 이 활동에서 컴포넌트 추출을 위한 아키텍처 뷰와 아키텍처 C&V 결정모델이 생성된다. 컴포넌트 분석 활동에서는 앞 단계에서 산출된 산출물들을 사용하여 생성하고자는 제품에 따라 제품계열 공통 아키텍처를 분석하여 실제적인 컴포넌트를 획득한다. 획득한 컴포넌트는 컴포넌트 명세서와 인터페이스 명세서로 기술되어 구현을 위한 지침서로 사용한다. 마지막 활동인 구현에서는 실제적인 제품으로써 소스코드가 생성된다. 본 논문에서는 첫 번째 단계인 패밀리 공학에 초점을 맞추어 진행하였다.

3.2 스코핑/도메인 분석

제품계열에서의 스코핑은 제품 패밀리 내 제품들의 집합에 의해 제공 되어야 하는 경계를 구분짓는 것으로 시스템의 내부 혹은 외부에 포함되어야하는 행위나 관점들의 정의를 통해 시스템의 집합을 경계 짓는 활동이다. 스코핑에서는 최종 제품에 대한 승인기준을 포함하며, 핵심적인 요구사항과 제약 사항들을 표현한다. 스코핑을 결정하기 위한 접근방법으로는 프로엑티브(Proactive) 접근과 리엑티브(Reactive) 접근 방식으로 구분 가능하다. 프로엑티브 접근 방법은 제품계열에 의해 지원되는 예상 가능한 모든 제품과 특성들을 포함하는 것

으로 미래 적용 가능성 있는 요소들을 포함하는 방식이며, 리액티브 접근 방식은 제품계열에 의해 그 시점에서 필요하다고 예상되는 요구에 따른 한정적 제품들을 포함하는 방식으로 현재의 요구사항을 충실히 반영하는 방법이다. 두 가지 접근 방법은 해당 제품의 최종목적을 기준으로 선택적으로 사용할 수 있다. 스코핑 결정 방법을 설정한 후 결정 방법에 포함되는 스코핑의 범위를 결정한다. 스코핑의 범위는 필요 이상으로 광범위한 범위를 가질 경우 제품계열의 공통성이 희박해 지며, 너무 좁은 범위를 가질 경우 제품 패밀리의 모든 사용을 수용할 수 없기 때문에 제품계열의 비즈니스 목적에 따라 결정해야 한다. [그림 3]은 스코핑 활동을 위한 작성을 나타낸다.



[그림 3] 스코핑 활동

그 룹 명	특 징 목 록
그룹 1	특징 1 특징 2 특징 3
그룹 2	특징 1
⋮	⋮

[그림 4] 특징 목록

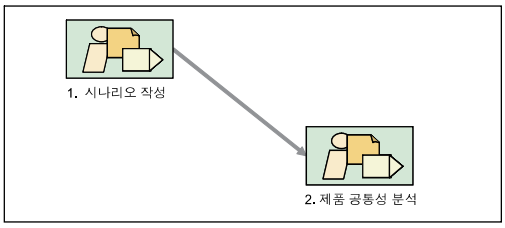
스코핑 활동은 크게 4개의 작업으로 구성된다. 제품계열 자료 분석에서는 대상 제품들의 성격에 기반하여 제품계열에 포함될 제품들을 식별한다. 제품 및 제품 특성 식별 작업에서는 활용 가능한 제품관련 문서들을 수집하고 개발자, 사용자, 유지보수자 및 제품 전문가들의 의견을 수집하여 제품 특성을 식별한다. 제품 특징을 식별한 후 특징 목록을 작성한다. [그림 4]는 특징 목록을 나타낸다. 특징 목록에서는 특징을 그룹핑한 그룹명과 그룹에 포함되는 식별된 특징을 기술한다.

세 번째로 핵심 특성 분석에서는 제품의 성능, 구조, 진화 가능성 등을 식별하고 식별된 요소들을 바탕으로 스코핑 관련 핵심 특성을 분석한다. 최종적으로 제품 맵 작성에서는 각 제품 멤버들이 포함될 수 있는 특징을 매핑하는 제품 맵을 작성한다.

특성 \ 제품		제품 A	제품 B	제품 C
그룹 1	특징 1	V	V	V
	특징 2	V	V	V
	특징 3		V	V

[그림 5] 제품 맵

[그림 5]은 제품 맵을 나타낸다. 특징은 특징목록에 표현된 정보를 사용하고, 각 특징이 제품계열의 각 제품들에 포함되어 있는 지를 체크한 후 모든 제품에 포함되지 않는 특징은 제품계열에서 구분되는 가변성 요소로서 식별된다.



[그림 6] 도메인 분석 활동

도메인 분석 활동은 시나리오 작성과 제품 공통성 분석의 2개의 작업으로 구성된다. 시나리오 작성 작업을 통해 스코핑 활동에서 추출된 특징들을 기반으로 제품계열의 비즈니스 프로세스를 구축하여 비즈니스 프로세스 모델을 작성하고 이를 유스케이스로 표현한다. 일반적인 유스케이스의 표현과 차이점은 공통성/가변성 요소를 표현하기 위

해 유스케이스를 <<optional>>, <<alternative>>, <<madatory>>의 스테레오 타입으로 표현한다. 또한 표현된 유스케이스별 시나리오를 작성하여 개발자의 이해를 돕는다. 실제적인 적용 사례는 제 4장 EMR 사례연구에서 표현한다. 추가적으로 비즈니스 프로세스들의 그룹핑이 요구될 경우 제품 서비스를 구성하고 제품 서비스별 시나리오를 작성한다. 다름 작업인 제품 공통성 분석에서는 시나리오 작성 작업에서 작성된 시나리오와 비즈니스 프로세스들을 분석하여 각 비즈니스 프로세스에서 표현되어야 하는 공통성과 가변성 요소를 식별하기 위한 비즈니스 프로세스 C&V 결정 모델을 작성한다. 작성된 비즈니스 프로세스 C&V 결정모델은 다음 [그림 7]과 같은 형태를 가진다.

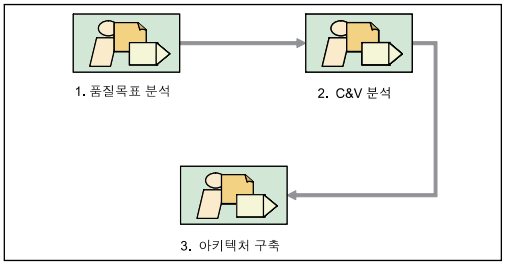
ID	가변점	재사용 분류	가변 조건	해결유무	효과
1	···	optional	···	Yes no	····· ·····
2	···	alternative	···	Yes no	····· ·····
3	···	optional	···	Yes no	····· ·····

[그림 7] 비즈니스 프로세스 C&V 결정모델

비즈니스 프로세스 C&V 결정모델에서 가변점은 가변성을 가진 특징을 나타내며, 재사용분류는 해당 가변성이 적용되는 특징의 사용여부에 따라 optional, alternative로 구분된다. 가변조건은 해당 가변점의 가변성을 판단할 수 있는 조건을 표현하여 해결유무에 따라 각각 다른 효과를 보게 된다.

3.3 제품계열 아키텍처 구축

본 논문에서는 기존의 도메인 공학 방법론의 문제점인 실제적인 지침 부족 및 제품 개발시 스코핑 오류를 해결을 목적으로 공통 아키텍처 기반 소프트웨어 제품계열 제품 생산 라인을 개발 하는 방법론을 연구하였다. 제품계열 아키텍처 구축을 위해서는 제품계열 아키텍처 구축을 위한 중요 요소인 품질목표를 효율적이고 정확히 추출하기 위한 방법을 정의하며 제품계열 아키텍처 생성기법(전술/전략 도출)을 통해 제품계열 아키텍처를 구성하는 컴포넌트를 추출하기 위한 전략을 도출한다. 마지막으로 전략의 기본이 되는 품질 목표를 만족하는 전술을 생성한다. 제품계열 아키텍처 구축 활동은 3개의 작업으로 구분되어 진행된다. 다음 [그림 8]은 제품계열 아키텍처 구축 활동의 개요도이다.



[그림 8] 제품계열 아키텍처 구축 활동 개요도

품질목표 분석 작업에서는 품질 목표를 달성하기 위한 기능 단위의 컴포넌트 추출을 목적으로 한다. 이를 위해 기 작성된 우선순위 시나리오를 적용하여 작성된 아키텍처 전략을 통해 컴포넌트를 추출하기 위한 매핑 방법을 정립한다. 품질목표 분석을 위해서는 품질목표 설정과 아키텍처 전략구축의 절차를 수행한다. 품질목표 설정에서는 전 단계에서 작성된 비즈니스 프로세스를 바탕으로 품질 목표에 대한 이해 당사자들의 브레인 스토밍을 통해 비즈니스 프로세스를 품질 목표로 정제한다. 품질목표를 도출하기 위하여 비즈니스 시나리오의 우선순위를 파악한다. 전 활동에서 작성된 제품서비스 시나리오를 바탕으로 시나리오의 우선순위를 결정한다. 시나리오의 우선순위를 결정하기 위해서는 시나리오별 중요도, 난이도, 민감도를 구분하고, 구분된 결과를 근거로 우선순위를 작성한다. 작성된 우선순위 시나리오는 [그림 9]와 같다. 시나리오 우선순위 테이블은 제품 서비스 시나리오를 검토하여 시나리오의 중요도를 판별한다. 각 시나리오의 우선순위는 중요도, 난이도, 민감도(H, L, M)를 기준으로 설정한다.

번 호	시나리오	중요도	난이도	민감도
1	……	H	H	M
2	……	L	M	H
3	……	M	L	L

[그림 9] 시나리오 우선순위 테이블

품질목표는 제품계열 아키텍처 구축의 주요 목적으로 제품계열에 포함된 제품들이 필수적으로 만족해야하는 기능적, 비기능적 사항이다. 품질목표 추출을 위해서는 비즈니스 프로세스를 검토하여 품질목표로 설정하기 위한 제품 서비스 기능을 선택한다. 선택된 제품 서비스 기능을 바탕으로 이해당사자들간의 브레인스토밍을 통해 기능요소를 분석하고 필요한 비기능 요소를 추가한다. 최종적으로 브레인스토밍한 결과를 바탕으로 품질목표를 추출한다. 추출된 품질목표는 표준 품질목표 작성을 위한 기초 자료로 사용한다. 결정된 품질목표의 우선순위에 따라 실제 제품계열 아키텍처에서 고려되어야 할 표준 품질목표를 설정한다. 표준 품질목표란 제품 아키텍처의 목적을 달성하기 위한 가장 주요한 품질목표이거나 제품 아키텍처를 제한할 수 있는 품질목표들을 말한다. 표준 품질목표 작성 후에 산출되는 품질목표 명세서는 [그림 10]과 같다.

번 호	품질목표	설 명	가변성
1	G 1	……	
		……	
2	G 2	……	
		……	
		……	

[그림 10] 표준 품질목표 명세서

아키텍처 전략구축에서는 앞 절차에서 생성된 각 표준 품질목표에 대한 아키텍처 전략을 도출한다. 이를 위해 각 품질목표별 아키텍처 매핑을 위한 전술을 작성하고 작성된 전술의 통합을 통해 아키텍처 전략을 생성한다. 전술은 품질 목표를 만족하기 위한 기능요소를 정의한 것이다. 전략은 전술을 통합 한 것으로써 모듈단위의 컴포넌트를 추출하기 위한 아키텍처 스타일 역할을 수행하며 기능단위를 표현한다. 작성된 아키텍처 전략과 비즈니스 프로세스를 매핑하여 모듈단위의 컴포넌트를 추출한다. 추가적으로 작성된 각 모듈단위 컴포넌트에 대한 구조와 컴포넌트들간의 상호작용을 정의한다. [그림 11]은 아키텍처 전략 명세서이다.

표준 품질목표	세부 품질목표	전 략	시나리오	근 거
G1	……	S 1	……	……
			……	
	……		……	
G2	……	S 2	……	……
	……		……	
			……	

[그림 11] 아키텍처 전략 명세서

아키텍처 전략 명세서는 표준 품질목표를 구성하는 세분화된 품질 목표를 표현하는 세부 품질목표와 세부 품질목표를 만족하기 위한 스타일을 지정하는 전술, 그리고 세부 품질목표가 달성하여야 하는 상황이나 조건을 표현하기 위한 시나리오, 마지막으로 전술을 선택하게 된 근거를 표현한다. 또한 각 세부 전술과 통합된 전략에 대한 뷰를 생성하여 각 뷰별 컴포넌트와 커넥터를 식별한다. 뷰를 표현하는 방법은 모듈 뷰, C&C(Component & Connector) 뷰, 할당 뷰의 아키텍처 뷰 등으로 나타낼 수 있다.

C&V 분석 작업에서는 추출된 모듈단위 컴포넌트들에 대한 C&V 분석을 통해 작성된 C&V 결정모델을 적용하여 각 컴포넌트와 뷰(전략)에 대한 공통성과 가변성을 식별한다.

전략단위 C&V 식별을 위해서 전술단위의 가변성의 존재 유무를 판단하여 구현 시 컴포넌트의 가변성을 고려하여 작성하도록 전술단위의 가변성을 식별한다. 이를 위해서는 기 작성된 전략과 작성된 전략단위 C&V 결정모델의 관계를 분석하고 분석결과를 바탕으로 전략단위의 기능에 대한 가변성/

공통성 요소를 식별한다. 마지막으로 식별된 가변성/공통성 요소를 바탕으로 가변성 리스트를 작성한다. 또한 전략단위의 공통 요소와 가변 요소 분리를 위한 결정 모델로써 전략단위 C&V 결정모델을 작성한다. [그림 12]는 C&V 결정모델의 형식이다.

가변점	가변치	가변성 타입	가변조건	해결유무	효과
V1	···	optional	···	Yes	···
				No	···
	···	alternative	···	Yes	···
				No	···

[그림 12] C&V 결정모델

C&V 결정모델은 가변성 발생 위치인 가변점, 가변성을 포함하는 주체인 가변치, 가변성을 구분하는 가변조건, 그리고 해결유무에 대하여 “yes” 혹은 “no”의 경우에 대하여 기록하고, 각각의 효과에 대해서 표현한다.

아키텍처 구축 작업에서는 작성된 C&V 모델과 아키텍처 전략을 통해 기 작성된 모듈단위 컴포넌트에 가변성과 비기능성 요소를 적용하여 정제된 컴포넌트를 생성한다.

4. EMR 사례 연구

제안한 방법론의 적용을 위해 EMR(Electronic Medical Record) 도메인의 사례를 수행하였다. EMR은 병원에서 사용되는 종이 문서를 없애고 모든 데이터를 전산매체에 저장하는 방식이다. EMR 사례 적용을 위한 제품군으로써 종합병원, 병원, 위원에서 사용되는 제품을 고려하였다.

첫 번째로 스코핑 활동에서 EMR을 적용한 도메인에서 사용하는 특징에 대해 특징 목록을 작성한다. [그림 13]은 EMR의 특징 목록이다. EMR 제품계열은 3개의 제품 멤버로 구성되며 제품 맵에서는 각 제품 멤버별 포함하는 특징에 대해 매핑을 실시한다. 또한 추출한 특징에 대해 공통성과 가변성을 구분하여 가변성이 존재할 때를 지시한다.

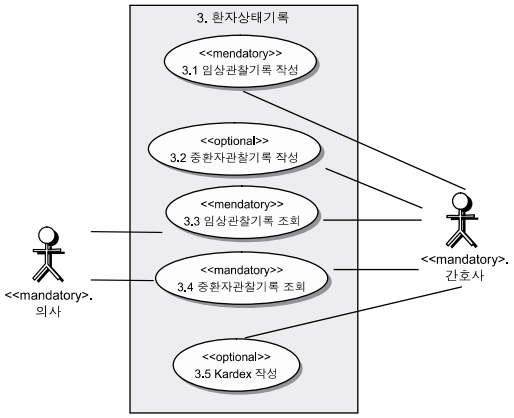
그 룹 명	Feature List
진료 기록 관리	· 진료기록 작성 · 진료기록 조회 · 경과기록 작성 · 경과기록 조회 · 병력기록 작성 · 병력기록 조회
의사진시관리	· 의사 지시 기록 · 협진 지시 기록 · 전화 구두 지시 · 투약 중단 지시 · 진료과 변경
환자상태기록	· 임상 관찰기록 작성 · 중환자 관찰기록 작성 · 임상 관찰기록 조회 · 중환자 관찰기록 조회 · Kardex 작성
· · · ·	· · · ·

[그림 13] EMR 특징 목록

특성 \ 제품		종합병원	병원	의원
진료 기록 관리	진료기록 작성	✓	✓	✓
	진료기록 조회	✓	✓	✓
	경과기록 작성	✓	✓	✓
	경과기록 조회	✓	✓	✓
	병력기록 작성	✓	✓	
	병력기록 조회	✓	✓	✓
의사 지시 관리	의사 지시 기록	✓	✓	✓
	협진 지시 기록	✓	✓	
	전화 구두 지시	✓	✓	
	투약 중단 지시	✓	✓	
	진료과 변경	✓	✓	
환자 상태 기록	임상 관찰기록 작성	✓	✓	✓
	중환자 관찰기록 작성	✓	✓	
	임상 관찰기록 조회	✓	✓	✓
	중환자 관찰기록 조회	✓	✓	✓
	Kardex 작성	✓	✓	
· · · ·	· · · ·	· · · ·	· · · ·	· · · ·

[그림 14] EMR 제품 맵

도메인 분석 활동에서는 앞 활동에서 작성된 특징 목록을 분석하여 비즈니스 프로세스와 제품 서비스를 파악한 후 비즈니스 프로세스가 갖는 기능을 유스케이스 모델로 표현한다. [그림 15]는 6개의 제품 서비스 중 환자상태 기록에 대한 유스케이스 모델이다.



[그림 15] EMR 유스케이스 모델

<<optional>> 3. 환자상태기록	간호사가 환자의 입원환경에 따라 임상관찰기록 또는 중환자관찰기록을 작성하고 Kardex를 작성하는 기능	
3.1 임상관찰기록작성 : 주치의의 입력한 처방에 따라 임상관찰 기록 내용 및 주기가 결정되어있고, 이를 간호사가 인지하고, 환자에 대하여 활력증상을 측정하고 이를 기록한다. 3.2 중환자관찰기록작성 : 중환자실 간호사 및 중환자실 환자에 대하여서는 중환자실 관찰기록을 주치의의 입력한 처방에 따라 임상관찰 기록 내용 및 주기가 결정되어있고, 이를 간호사가 인지하고, 환자에 대하여 활력증상, 투약내용, 검사내용, 신체반응내용 등을 측정하고 이를 기록한다. 3.3 임상관찰기록조회 : 특정 환자에 대해 그 동안의 임상관찰기록을 조회한다. 3.4 중환자관찰기록조회 : 특정 환자에 대해 그 동안의 중환자실기록을 조회한다. 3.4 Kardex작성 : 간호사별로 업무 기록을 작성, 조회 혹은 변경한다.		
비기능적인 요소	수행환경	데이터 요소를 그래픽화하여 표현할 수 있는 수행환경
	UI 환경	입력된 활력증상에 대하여 그래픽적으로 표현하며, 입력은 텍스트를 기반으로 한다.
	성능	개별 환자에 대한 초기로딩은 3초 이내이며, 이후 검색은 1초 이내로 한다.
	보안	환자에 대한 기록은 해당 병동에서 해당 번을 맡고있는 사용자에 의해서만 가능하다. 기록된 내용은 수정, 삭제할 수 없으며, 정정한 내용에 대한 내력을 기록해야 한다.(Kardex 제외)
	기타	-
3. 환자상태기록에서는 임상관찰기록과 중환자실 기록이 동시에 이루어질 수 없고, 병실 및 병실간호사에 의해 두 유스케이스 중 하나가 선택적으로 표현된다. 해당 두가지의 유스케이스의 순서는 없다 3.4 중환자관찰기록조회 → 3.2 중환자관찰기록작성		

[그림 16] EMR 제품 서비스 시나리오

또한 작성된 유스케이스와 제품 서비스를 바탕으로 제품서비스별 비즈니스 시나리오를 작성한다. 비즈니스 시나리오에는 비기능적인 요소를 추가적으로 반영한다. [그림 16]은 제품 서비스 시나리오를 보여준다.

다음 단계로 전 단계에서 작성된 제품 서비스 시나리오와 비즈니스 프로세스들을 분석하여 각 비즈니스 프로세스에서 표현되어야 하는 공통성과 가변성 요소를 식별하기 위한 비즈니스 프로세스 C&V 결정모델을 작성한다. 비즈니스 프로세스 C&V 결정모델에서는 가변적인 제품 서비스 및 유스케이스별 가변점을 선택하여 선택조건을 판단하기 위한 조건 및 효과를 표현한다. [그림 17]은 비즈니스 프로세스 C&V 결정모델이다.

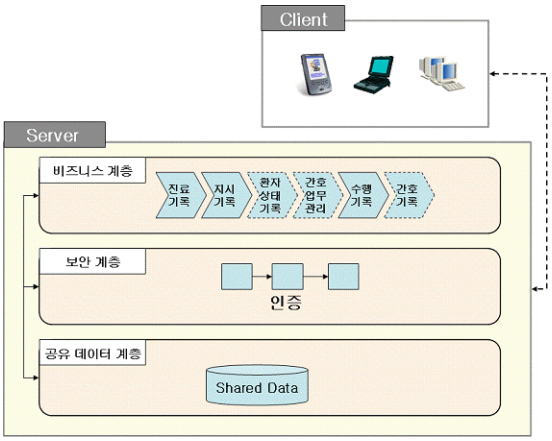
ID	가변점	재사용 분류	가변 조건	해결유무	효 과
2	환자상태관리	optional	환자상태 체크가 필요한가?	Yes No	스테레오타입제거 환자상태관리를 삭제
3	간호업무관리	Optional	간호업무 관리가 필요한가?	Yes No	스테레오타입제거 일반간호기록을 삭제
4	간호기록	Optional	간호기록이 필요한가?	Yes No	스테레오타입제거 검사조회를 삭제

[그림 17] EMR 비즈니스 프로세스 C&V 결정모델

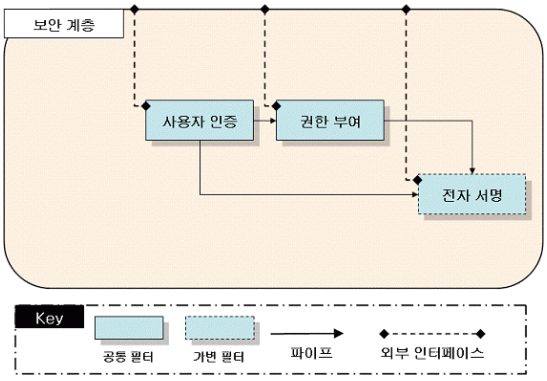
EMR 품질목표를 도출하기 위해 비즈니스 시나리오의 우선순위를 파악하고 시나리오 우선순위 테이블로부터 표준 품질목표를 도출한다. 도출한 품질목표를 만족하기 위한 전략을 아키텍처 전략 명세서를 통해 기술한다. [그림 18]은 EMR 아키텍처 전략 명세서이다.

표준 품질목표	세부 품질목표	전 략	시나리오	근 거
데이터 무결성	일관성	공유 데이터	1. 진료 기록은 변경 불가	의무기록을 무결성을 보장하기 위한 단일 데이터베이스 유지
	추적 가능성		2. 환자 상태 기록은 생성, 수정 및 삭제한 경우 변경 내용에 대한 나력을 기록.	
			3. 수행 내역의 작성시 수정 및 삭제할 수 없으며, 수정시 수정 내역에 대한 이력을 보관	
성능	응답시간	클라이언트-서버	4. 그래픽 데이터는 최대 7초 이내에 제공	-
			5. 지시항목 검색 응답 시간이 3초 이내로 제한	
⋮	⋮	⋮	⋮	⋮

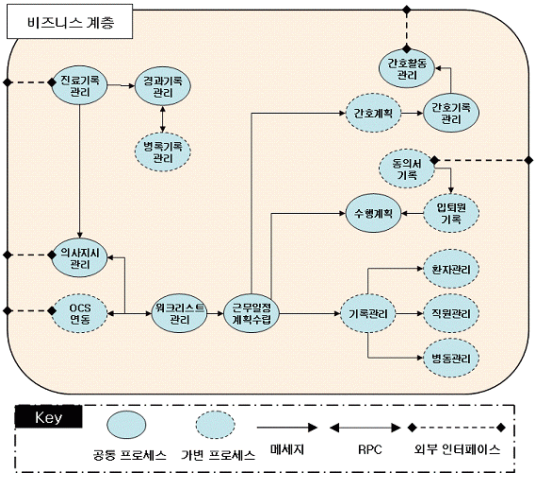
[그림 18] EMR 아키텍처 전략 명세서



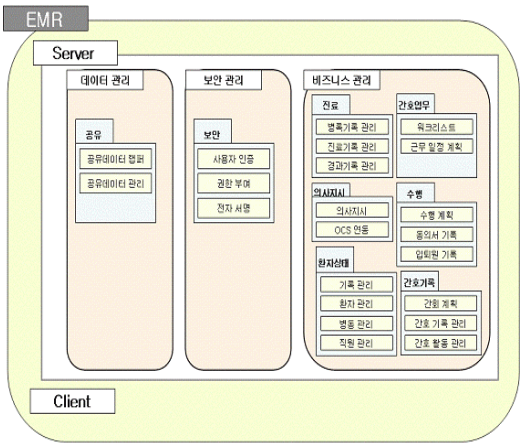
[그림 19] EMR 통합 뷰



[그림 20] EMR 파이프-필터 뷰



[그림 21] EMR 프로세스 뷰



[그림 22] EMR 모듈 뷰

아키텍처 구축을 위해서는 전략 명세서와 함께 아키텍처 뷰를 사용하여 표현한다. 아키텍처 뷰는 시스템의 관점을 나타내는 C&C 뷰, 시스템의 구현 관점을 나타내는 모듈 뷰, 시스템의 흐름을 표현하는 파이프-필터 뷰로 표현된다.

다음으로 각 뷰별 가변성을 파악하고 가변치에 따른 선택조건에 따라 효과를 기술하기 위해서 아키텍처 C&V 결정 모델을 작성한다. [그림 23]은 컴포넌트 단위의 C&V 결정모델이다.

가변점	가변치	가변성 타입	가변 조건	해결유무	효과
프로세스 뷰	환자상태관리 컴포넌트	선택	환자상태 체크가 필요한가?	Yes No	컴포넌트 선택 컴포넌트 삭제
	간호업무관리 컴포넌트	선택	간호업무 관리가 필요한가?	Yes No	컴포넌트 선택 컴포넌트 삭제
	간호기록 컴포넌트	선택	간호기록이 필요한가?	Yes No	컴포넌트 선택 컴포넌트 삭제

[그림 23] EMR 컴포넌트 C&V 결정모델

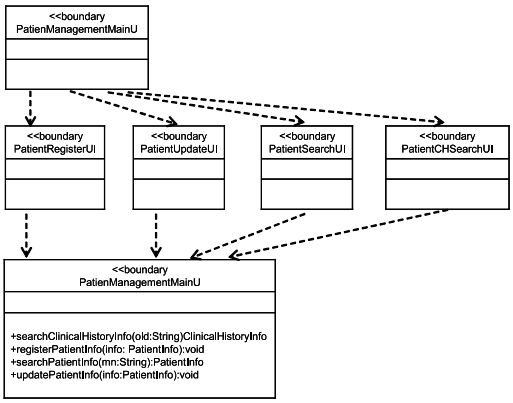
추출된 모듈단위 컴포넌트와 아키텍처 C&V 결정모델을 통해 정제된 컴포넌트를 도출하고 이를 표현하기 위해 [그림 24]와 같이 컴포넌트를 명세하고 컴포넌트 상호간의 관계를 표현하기 위기 위해서 정적 및 동적 모델을 이용한다.

컴포넌트 명세서				
컴포넌트 이름	PatientMgt 컴포넌트			
컴포넌트 종류	비즈니스 컴포넌트			
설명	병원에 등록된 환자의 일반적인 정보와 치료 중인 환자의 병력 정보를 관리하는 기능을 제공한다.			
제공 인터페이스	IPatientMgt	오퍼레이션 명	입력 매개변수	출력 매개변수
		searchClinicalHistoryInfo	String piid	ClinicalHistoryInfo
		registerPaientInfo	PatientInfo info	void
		searchPainetInfo	String rrn	PaientInfo
		updatePaientInfo	PaientInfo info	void
품질속성	–			
가변점	–			
제약사항	–			

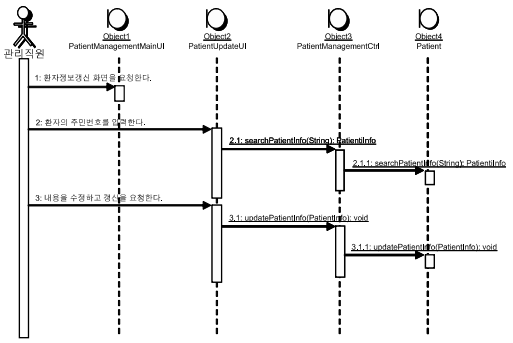
[그림 24] EMR 컴포넌트 명세서

컴포넌트 인터페이스 명세서				
인터페이스 명	IDossierMgt			
설명	간호사가 환자의 입원 환경에 따라 임상 관찰기록 또는 중환자 관찰기록을 작성하고 Kardex를 작성하는 기능들을 제공한다.			
컴포넌트 명	DossierMgt 컴포넌트			
오퍼레이션	오퍼레이션 명	입력 매개변수	출력 매개변수	
	showObRecordingList	String sid	Vector	
	registerObRecording	ObservationRecordingInfo info	void	
	showAllObRecordingList	–	Vector	
	searchOR	String pid	ObservationRecordingInfo	
	searchOBRecordingForDate	String fDate, String sDate	Vector	
	registerKardex	KardexInfo info	void	
가변점	public static void registerObRecording(ObservationRecordingInfo info) { } <<vp>>			
	ObservationRecording(ObservationRecordingInfo info){ }<<vp>>			
	public static void registerKardex(KardexInfo info) { }<<vp>>			
	Kardex(InkKardexInfo info){ }<<vp>>			
요청 인터페이스	인터페이스 명	오퍼레이션 명	입력 매개변수	출력 매개변수
	IWardMgt	searchWardInfo	String wid	wardInfo
	IPatientMgt	searchClinicalHistoryInfo	String piid	ClinicalHistoryInfo
제약사항	–			

[그림 25] EMR 인터페이스 명세서



[그림 26] EMR 정적 모델



[그림 27] EMR 동적 모델

5. 결론 및 향후 연구

본 논문에서는 제품의 다품종 생산 환경에 적응하기 위한 소프트웨어 개발 방법론으로써 공통 아키텍처 기반의 제품계열 아키텍처 설계를 제안하였다. 기존 방법론들의 모호한 프로세스를 보완하기 위하여 본 논문에서 제안한 제품계열 아키텍처 개발 프로세스는 스코핑, 도메인 분석, 제품계열 아키텍처 구축, 컴포넌트 명세, 구현의 크게 5개의 단계로 구분하였으며 패밀리 공학 영역에서 제품계열 아키텍처를 구축하기 위한 활동, 즉 스코핑, 도메인 분석, 제품계열 아키텍처 구축에 중점을 두었다.

본 연구는 임베디드 시스템과 같은 다양한 모델이 생산되며 모델 변경이 급속히 이루어져야 하는

소프트웨어 개발 방식에 적용할 수 있는 접근방식이다.

향후 연구로 아키텍처 재사용성 향상을 위한 가장 큰 주류인 MDA(Model Driven Architecture)와 제품계열 공학의 통합을 통해 생산성을 향상하기 위한 핵심자산과 아키텍처 설계에 적용사항에 대해 연구가 필요하다.

참 고 문 헌

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

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

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

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

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

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

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

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

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

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

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

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

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

◈ 저 자 소 개 ◈

오 영 배 (yboh@swc.ac.kr)

고려대학교 공학사, 고려대학교 컴퓨터학과에서 박사수료 하였다. 그리고 정보처리기술사 자격증을 보유하고 있다. 캘리포니아대학(UCI)에서 객원연구원으로 근무한 바 있고, 한국전자통신연구원(ETRI)에서 책임연구원(실장)으로 근무하였고, 현재 수원여대 컴퓨터응용학부 교수로 재직 중이다. 현재 관심분야는 소프트웨어 아키텍처, 소프트웨어 품질평가 및 전자상거래이다.

신 성 욱 (stnoble@software.korea.ac.kr)

고려대학교 컴퓨터학과에서 전산학 석사와 박사 수료를 취득하였다. 현재 박사과정 중이며, 주요 관심분야는 모바일 데이터베이스, 소프트웨어 아키텍처, 모바일 분산쿼리 처리 등이다.

김 영 갑 (ygkim@swsys2.korea.ac.kr)

고려대학교 컴퓨터학과에서 전산학 석사와 박사 수료를 취득하였다. 현재 박사과정 중이며, 주요 관심분야는 컴퓨터 보안 및 소트프웨어 아키텍처 등이다.

백 두 권 (baikdookwon@hotmail.com)

Wayne State Univ에서 전산학 석사와 박사를 취득하였다. 고려대학교 정보통신대학 학장, 정통부 정책자문 위원 및 한국시뮬레이션학회 회장을 역임하였으며 현재 고려대학교 컴퓨터학과 교수로 재직 중이다. 현재 관심분야는 소프트웨어 엔지니어링, 시뮬레이션이다.