소프트웨어 제품 계열 방법론의 기술적 평가

박 신 영[†] · 김 수 동[††]

요 약

제품 계열 공학(Product Line Engineering, PLE)은 도메인에서 멤버들이 갖는 공통 휘처를 재사용 가능한 핵심 자산으로 만들고, 만들어진 핵심 자산을 이용해서 애플리케이션을 개발하는 방법론이다. PLE는 핵심 자산을 개발해서 재사용하므로, 개발 비용을 감소시키며 자산의 재사용성을 증가시킬 수 있다. 지금까지 여러 개의 PLE 방법론이 소개되었으나, 표준화된 PLE 방법론이 존재하지 않기 때문에, 방법론이 제시하는 프로세스나 산출물 등에서 차이가 크며, 산업계에서는 PLE 방법론을 채택하는데 어려움이 있다. 이에 프로세스의 선택과 효율적인 활용을 위해서 기존의 여러 방법론을 비교 분석 하는 작업이 요구된다.

본 논문에서는 FAST, SEI SPL, PuLSE, Bosch의 제품 계열 프로세스, FOPLE, ESAPS, KobrA/PoLITe, Alexandria, COPA, QADA 방법론 등 대표적인 PLE 방법론을 프로세스, 산출물, 적용 지침 측면 별로 비교 기준을 나누고, 비교 기준에 따라서 비교 평가를 수행한다. 또한 방법론간 공통성이 큰 항목과 적은 항목을 확인해서, PLE 방법론이 포함해야 하는 요소와 각 방법론이 개선해야 하는 사항을 확인한다. 본 논문은 적절한 프로세스를 선택 또는 재정의하는 과정에서 활용할 수 있을 뿐만 아니라, 향후 PLE 개발 방법론의 표준을 정의하는 과정에서 기반이 될 수 있을 것이다.

키워드 : 제품 계열 공학, PLE 프로세스, 방법론 평가

A Technical Assessment of Software Product Line Methodologies

Shin Young Park[†] · Soo Dong Kim[††]

ABSTRACT

Product Line Engineering(PLE) is an effective software development technique which produces applications using core assets. Because of reusing the core assets, PLE can save cost for developing products in a domain but increase reusability. There are about ten PLE methodologies available, but there are not yet common agreements on PLE process and artifacts. This makes developers harder to choose a methodology and to apply it in practice. A comprehensive technical evaluation and comparison on existing PLE methodologies would be essential for practitioners.

In this paper, we present a technical assessment of representative PLE methodologies; FAST, SEI SPL, PuLSE, Bosch's PL process, FOPLE, ESAPS, KobrA/PoLITe, Alexandria, COPA, QADA. They are compared in the criteria of process, artifacts, instructions, and special features. And we identify common or variable elements between methodologies and confirm elements to be improved in each PLE methodology. The assessment result would be well utilized in defining a practical methodology for PLE projects and in choosing an appropriate methodology among available ones.

Key Words : Product Line Engineering, PLE Process, Methodology Assessment

1. 서 론

제품 계열 공학(Product Line Engineering, PLE)은 제품 생산의 효율성 측면이 부각되면서 최근 주목 받고 있는 제품 개발 기술로, 과거 소프트웨어(Software, SW) 개발 활동이 단일 프로젝트에 의해 수행되던 것보다 재사용성을 강조한 방법론이다. PLE는 크게 핵심 자산(Core Asset) 개발과 애플리케이션 개발 프로세스로 구성된다. 핵심 자산 개발은 한 도메인에 속하는 패밀리 멤버의 공통성과 가변성을 분석하여 재사용 가능한 핵심 자산을 만들며, 애플리케이션 개발은 핵심 자산을 이용하여 제품을 개발함으로써 생산성 향상을 강조한다.

1990년대 초반 FAST 방법론이 소개 된 이후 많은 PLE 방법론이 발표되었으며 학계와 산업계의 PLE 분야에 대한 관심 역시 높아지고 있는 추세이다. 그러나 표준화된 PLE 방법론이 존재하지 않기 때문에, 방법론이 제시하는 프로세스나 산출물 등에서 큰 차이가 크며, 산업계에서는 기존에 발표된 PLE 방법론을 선택하거나 방법론을 사용해서 제품

※ 본 연구는 한국과학재단 특정기초연구(R01-2005-000-11215-0)지원으로 수행되었음
† 준 회 원 : 숭실대학교 대학원 컴퓨터학과 졸업예정(공학석사)
†† 종신회원 : 숭실대학교 컴퓨터학부 부교수
논문접수 : 2005년 9월 8일, 심사완료 : 2005년 12월 16일

을 개발하는데 어려움을 갖는다. 이에 프로세스의 선택과 효율적인 활용을 위해서 기존에 발표된 방법론을 비교 분석하는 작업이 요구된다. 각 PLE 방법론의 장단점 및 특징을 비교한 연구가 있다면 산업계와 학계에서 적절한 프로세스를 선택 또는 재정의하는 과정에서 활용할 수 있을 것이다.

본 논문에서는 FAST, SEI SPL, PuLSE, Bosch의 제품 계열 프로세스, FOPLE, ESAPS, KobrA/PoLITe, Alexandria, COPA, QADA 방법론 등 대표적인 PLE 방법론을 프로세스, 산출물, 적용 지침 측면 별로 비교 기준을 나누고, 비교 기준에 따라서 비교 평가를 수행한다. 또한 방법론간 공통성이 큰 항목과 적은 항목을 확인해서, PLE 방법론이 포함해야 하는 요소와 각 방법론이 개선해야 하는 사항을 확인한다.

2장에서는 관련 연구를 설명하며, 3장에서는 지금까지 발표된 대표적인 PLE 프로세스를 소개한다. 4장에서는 방법론을 비교하기 위한 기준을 제시하며, 5장에서는 방법론간 비교를 수행한다. 6장에서는 방법론을 비교한 결과를 요약하고 방법론간 공통성이 큰 항목과 공통성이 적은 항목을 확인하며, 각 방법론의 개선 사항을 제안한다. 그리고 7장에서는 결론을 맺는다.

2. 관련 연구

2.1 Matinlassi의 비교

Matinlassi는 이 논문에서 COPA, FAST, FOPLE, KobrA, QADA 방법론의 제품 계열 아키텍처(Product Line Architecture, PLA) 설계 방법을 설명하며, PLA 설계 방법을 평가하기 위한 비교 프레임워크를 소개한다[1]. Matinlassi가 PLA 설계 방법을 평가하는 목적은 현재 발표된 방법론들의 등급을 매기기 위해서가 아니라 PLA 공학 방법들을 전반적으로 살펴보기 위해서며, 특정 애플리케이션 개발 시 PLA가 제공하는 기능을 선택하기 위한 결정 도구(decision tool) 개발에 기반이 되는 연구를 하기 위해서이다. 그러나 Matinlassi의 논문은 현재 소개된 PLE 방법론 중 다섯 가지만을 비교했으며, 방법론 전체가 아닌 제품 계열 아키텍처 설계 방법만을 소개하고 비교한다는 한계를 가진다.

2.2 Vehkomaki & Kansala의 비교

Vehkomaki와 Kansala는 범용 제품 계열 프로세스(Generic Product Line Process, GPLP) 프레임워크를 소개하고 GPLP를 기준으로 SEI FSPLP(Software Engineering Institute : Framework for Software Product Line Practice), Synthesis DsE(Domain-specific Engineering based on Synthesis), RSEB(Reuse-driven SW Engineering Business), SPICE NRC SPF(Nokia Research Center Software Process Framework based on SPICE v2.0)를 비교한다[2]. 비교 결과에 따르면 NRC SPF만 시스템 공학 분야에 대한 고려를 하며, SEI FSPLP는 제품 포트폴리오 관리, 제품 계열로의 변화(transition to PL), 도메인 공학, 제품 공학 등에 대한 지원이 전반적으로 잘 되어있다. Synthesis는 도메인 공학 활동이 구체적으로 제시되어 있으며, RSEB는 COTS등의 활용에 대한 구체적인 언급은 없지만, 제품 계열로 변화하는 부분이 상세하게 제시되어 있다. 이 논문은 비교를 통해서 실제 SE 활동 분야와 프레임워크를 정의하고 있지만, 일부 프레임워크만을 비교하는 한계점을 갖는다.

3. 각 PLE 프로세스의 개요

3.1 FAST

AT&T는 FAST(Family-oriented Abstraction, Specification, and Translation)를 1992년에 소개하였고, Lucent Technology는 이 개념을 발전시켜서 1999년에 발표하였다[3]. FAST는 실용적인 패밀리 기반 SW 생산 개념을 포함하고 있으며, 도메인 결정(Qualifying Domain)·도메인 공학(Engineering Domain)·애플리케이션 공학(Application Domain) 프로세스로 구성된다. FAST 프로세스는 일관되고 잘 정리된 프로세스인 PASTA(Process and Artifact State Transition Abstraction) 모델로 확장 적용되는데, PASTA는 SE 프로세스 설명을 위한 체계적인 방법을 제공하는 모델이다. PASTA는 프로세스의 세부 지침을 제공하고 다른 목적으로 프로세스 모델을 이용하는 사용자들간의 의사소통을 가능하게 하며 프로세스의 반복(iteration)단계를 가진다.

3.2 SEI의 제품 계열 공학

SEI의 제품 계열 공학(Software Engineering Institute Software Product Lines, SEI SPL)은 Carnegie Mellon 대학의 소프트웨어 공학 연구소에서 2001년에 개발하였다[4]. SEI SPL은 핵심 자산 개발, 제품 개발, 관리의 세가지 필수 단계로 구성되며 핵심 자산 개발은 제품 개발 가능성을 확인하는 활동이고 제품 개발은 개별 제품을 생산하는 단계이며 관리 활동은 핵심 자산 개발과 제품 개발 활동 단계를 통해서 만들어진 제품이 요구사항을 잘 반영하고 있는지, 정의된 프로세스 단계를 잘 수행하고 있는지, 프로세스 수행 과정에서 데이터 수집이 충분한지를 확인하고 관리하는 역할을 한다. SEI SPL은 PLE에 한정된 내용만을 정의하지 않고 SE 일반적인 내용도 포함하고 있다는 특징을 갖는다.

3.3 PuLSE

PuLSE(Product Line Software Engineering)는 기업 업무 배경(enterprise context)에서 소프트웨어 제품 계열(Software Product Line, SPL)을 파악하고 개발 하기 위한 목적으로 IESE에서 개발하였다[5]. PuLSE는 배치 단계(Deployment Phases), 기술 컴포넌트(Technical Components), 지원 컴포넌트(Support Components)의 세가지 주요 요소로 구성되며 그 중 프로세스 측면에서 PuLSE는 PL 기반구조(product line infrastructure)를 생성하며 PuLSE-Eco(Economics), PuLSE-CDA(Customizable Domain Analysis), PuLSE-DSSA(Domain-Specific Software Architecture)로 구성된다.

PuLSE의 주요 특징은 기술 컴포넌트가 각각의 상황에 맞게 특화(customization) 될 수 있다는 것이다[6]. PuLSE에서 특화 개념은 제품을 요구사항에 가장 근접하게 만들 수 있도록 지원한다는 것이며, 기술 컴포넌트의 특화를 통해서 구조(construction), 사용(usage), 진화(evolution) 활동의 특화된 버전이 만들어진다. 특화의 원리는 내용 구성, 재사용 목적과 활동, 프로젝트 구조와 이용 가능한 자원을 포함하여 애플리케이션을 응용한 특징 중 하나이다.

3.4 Bosch의 제품 계열 프로세스

Bosch의 PL 프로세스는 2000년에 Jan Bosch에 의해서 소개되었다. Bosch는 아키텍처와 컴포넌트들의 개발, 제품 개발을 통한 개발, 자산의 진화를 통한 개발 등의 측면으로 PL을 강조한다[7]. Bosch의 SPL은 세가지 요소를 포함하는데, 첫 번째는 자산요소이다. 즉 PL은 SW 아키텍처, 재사용 가능한 컴포넌트들과 제품들에 기초를 두고 설명된다. 두 번째 요소는 PL에서 가질 수 있는 비즈니스 관점, 조직적 관점, 프로세스 관점, 기술적 관점을 포함한 다양한 관점과 관련되어 있다. 마지막으로 세 번째 요소는 초기 개발, 배치, 진화 같은 PL의 우선적인 활동으로 구성된다. 이와 같은 세 가지 구성요소에 의해 Bosch의 PL이 설명된다.

3.5 FOPLE

FOPLE(Feature-Oriented Product Line Engineering)는 1998년 POSTECH에서 개발한 휘처(Feature) 기반 재사용 방법론으로 1990년 SEI에서 개발한 휘처 중심의 도메인 분석 방법을 확장한 개념이다[4, 8, 9]. FOPLE 방법론은 두 가지 프로세스로 나눌 수 있는데, 마케팅과 제품 계획을 수립하고 정련하며 휘처 설계 및 제품 계열(Product Line, PL) 요구사항을 분석하는 PL 자산 개발 프로세스(Product Line Asset Development Process)와 개발된 자산을 이용하여 제품을 개발하는 제품 개발 프로세스(Product Development Process)로 구성된다[10].

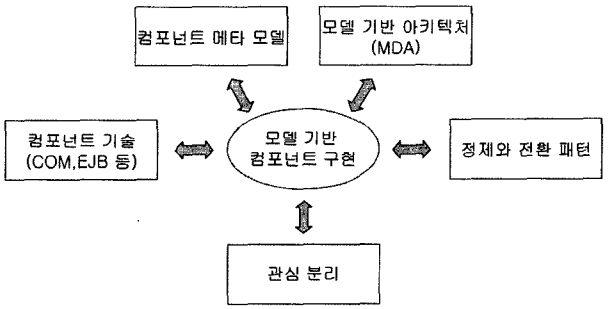
3.6 ESAPS

ESAPS(Engineering Software Architectures, Processes and Platforms for System Families)는 21개 회사와 연구소가 협력한 유럽의 연구 프로젝트로 시스템 패밀리를 위해서 아키텍처, 프로세스, 플랫폼을 사용하며, 시스템 품질을 향상시키는 방법을 연구할 목적으로 수행되었다. ESAPS는 두 단계로 구분된다. 첫 번째 단계에서는 개별적인 기술을 사용한 개발과 프레임워크 개발을 강조하며, 두 번째 단계에서는 개별적으로 검증된 기술, 접근 자동화 방법, 산업적인 규모를 검증하는 방법 모두를 통합하는데 초점을 맞춘다[11, 12].

3.7 KobrA / PoLITe

KobrA(Komponentenbasierte Anwendungsentwicklung)는 UML(Unified Modeling Language)을 사용한 컴포넌트 기반 PLE로서 간단하게 정의되어 있지만 휘처에 지나친 비중을 둔 다른 프로세스와는 다르게 만들어져야 한다는 목적을 가지고 개발되었다[6]. KobrA 프로세스는 프레임워크 공학과 애플리케이션 공학으로 구성되며 산출물 외적인 품질에 대한 예측을 위해서 품질 모델(quality model)을 사용한다. KobrA는 PLE 개념 자체가 복잡하므로 방법론 자체는 간단해야 하며 확실하고 자세한 가이드 라인을 제시하는 등 체계적인 성격을 가져야 한다고 주장한다. 뿐만 아니라 하나의 기본 개념을 사용하여 크고 작은 규모의 프로젝트를 모두 수행할 수 있어야 한다고 주장하며, 모델 기반 아키텍처(Model-Driven Architecture, MDA) 모델 변환(transformation)과 일치하는 정제 개념을 갖는다.

PoLITe(Product Line Implementation Technologies) 프로젝트는 Fraunhofer IESE에 의해서 수행되었으며, KobrA 메소드를 기반으로 하는 애플리케이션 개발 중심의 기술이다[13, 14]. PoLITe는 PL 공학에서 오직 구현 측면만을 고려하며, 구현에 의해 획득된 산출물은 실행 가능한 형태로 자동적으로 변형될 수 있다[10]. 이렇게 자동 변형 될 수 있는 PoLITe 프로젝트는 모델 기반 컴포넌트 구현을 지향한다고 할 수 있다. 뿐만 아니라 컴포넌트 기술을 사용하는 PL의 구현을 돕기 위한 방법을 표현하며 MDA, 컴포넌트 기반 개발, 패턴 사용과 관심 분리(separation of concern) 같은 몇몇 영역으로부터 개념을 통합한다[15].



(그림 1) PoLITe 컨텍스트 내부의 원리와 기술 [15]

3.8 Alexandria

Alexandria는 SPL을 개발하고 진화하기 위한 프로젝트이자 방법론으로써 1999년 시작되었다[16]. Alexandria는 산업계와 협력하는 프로젝트로, 프로젝트 과정에서 획득한 결과를 재적용하고 개발된 방법과 도구에 대해서 피드백을 제공한다. 방법론은 재사용 가능한 플랫폼을 기반으로 제품을 개발하는 재사용성, PL을 이용해서 제품을 만드는 유연성(flexibility), PLA의 진화, 제품 개발 시 사용되는 인력, 시간, 비용 소비를 최소화 하는 효율성을 목표로 한다.

3.9 COPA

COPA(Component-Oriented Platform Architecting)는 업무(business), 아키텍처, 프로세스, 조직 이 가장 잘 조화되는 문제에 초점을 맞춘 메소드로 통신, 화상 의료 기기, 전

자 기기와 관련된 애플리케이션 도메인을 개발하는 경험에 기반을 두고 필립스(Philips) 연구실에서 발표하였다[13]. COPA는 최소한의 비용으로 다양한 기술을 얻기 위해 사용하는데, 프로젝트의 크기와 복잡성을 관리하고 높은 품질의 제품을 획득하며 프로젝트 변화를 관리하고 개발 시간을 절약하기 위해 사용한다. COPA는 특정 도메인에 고유한 아키텍처 스타일로 업무(business), 프로세스, 조직과 관련되어 정의된 제품 패밀리 아키텍처(Product Family Architecture, PFA)를 중심으로 아키텍처 전반을 강조한다. 그리고 PFA는 아키텍처를 만드는 단계에서는 고객을 포함한 관계자들의 표현으로 된 요구사항과 직관적으로 얻은 아키텍처를 사용하여 다소 가벼운 아키텍처를 생산한다는 특징을 갖는다.

3.10 QADA

QADA(Quality-driven Architecture Design and quality Analysis method)는 요구사항 공학과 아키텍처 설계에 중심을 두고 있으며, 애플리케이션 개발 과정을 생략하고 아키텍처 분석과 설계 단계에 핵심 자산 개발 단계 의미를 포함하고 있는 PLE 개발 방법론이다[14]. QADA의 아키텍처 설계와 품질 분석은 서로 긴밀히 연관되어 있으며 아키텍처 개발 시 표현된다. 그리고 아키텍처 설계는 설계 초기 단계에서 자세하게 기술하고 아키텍처 품질을 분석하기 위해 추상화의 개념적 단계와 구체적 단계로 나눈다는 특징을 갖는다.

4. PLE 방법론의 비교 기준

본 논문의 4,5장에서는 PLE 방법론들을 네 가지 측면으로 비교하고, 각 측면에 대한 세부 비교 항목을 정의한 후 각 방법론이 비교 항목을 얼마나 만족시키는가를 평가한다. 본 논문에서 방법론을 비교하기 위한 비교 기준은 아래에 제시하는 기존 방법론들의 구성 요소들을 고려한다.

유럽 위원회(European Commission)가 지원하는 ECOSIN 데모 개발을 위한 SW 개발 방법론은 절차(procedure), 이용 가능한 구성 요소를 언급한 프레임워크, 설계와 구현이 절차에 따라서 작성되었는지 평가하는 평가 기준(evaluation criteria), 공통성과 표준을 제공하는 표기와 용어로 구성된다[17]. Bosch의 SW 라이프 사이클 방법론은 도구, 기술 그리고 지침(guideline)을 제공하는 메소드로 구성된다[18].

IT 분야의 방법론은 메소드 또는 프로세스를 가지며 프로세스를 통해서 얻을 수 있는 결과물인 산출물로 구성된다. 뿐만 아니라 프로세스는 지침을 제시해야 하는데, 방법론 사용시 지침에 따라 프로세스를 적용해야 하므로 구체적 지침은 중요한 부분으로 판단된다. 도구나 특정 기술 등은 방법론간 차이가 크므로 특징 측면으로 구성한다. 4장의 각 절에서 제시하는 내용은 비교하려는 각 측면의 구체적인 세부 항목 산정 근거가 된다.

4.1 프로세스 측면

본 절에서는 PLE 방법론들을 프로세스 측면으로 비교하기 위해서 다음과 같은 항목으로 나누었다. 각 항목은 기존의 방법론을 연구하는 과정에서 프로세스에 중요한 측면이라고 판단되는 부분으로 결정하였다. PLE 방법론을 적용한 프로젝트가 재사용성과 비용 절감 측면에서 이익이 최대화 되도록 하려면 PL 범위 설정 단계가 필요하다. 핵심 자산 개발 단계와 애플리케이션 개발 단계는 방법론간에 이를 지칭하는 명칭의 차이가 크므로 비교가 필요하다. 패밀리 멤버를 추가하거나 특정 요구사항을 핵심 자산에 반영하여 핵심 자산을 진화시킬 필요가 있을 경우 또는 핵심 자산 진화를 효율적으로 수행하기 위해서 프로세스 변경이 필요할 경우에 PL 진화 및 유지 보수 기능이 필요하다. 프로젝트 관리 기능을 구체적으로 제시하는 정도는 방법론간 차이가 크므로 비교한다.

4.2 산출물 측면

본 절에서는 PLE 방법론들을 산출물 측면으로 비교하기 위해서 다음과 같은 항목으로 나누었다. 각 항목은 기존의 방법론을 연구하는 과정에서 산출물을 비교하는데 중요한 측면이라고 판단되는 부분으로 결정하였다. 범용 아키텍처는 PLE 방법론에서 핵심 자산 개발 과정의 대표적인 산출물이므로 지원 여부와 이에 대한 명칭을 비교하는 과정이 필요하다. 또한 컴포넌트를 표현하는 용어에 차이가 있어서 각 방법론간에 이를 지칭하는 용어를 비교한다. 핵심 자산 개발 과정에서 결정되는 가변점과 해당 가변점에서 선택 가능한 해결 방법(resolution)을 제시해 놓은 의사 결정 모델(decision model) 역시 중요하며, 각 방법론간 의사 결정 모델 지원 여부와 이를 지칭하는 명칭에 대한 비교 작업이 필요하다. 방법론에서 핵심 자산의 메타데이터를 제공하면, 핵심 자산의 구성 요소나 요소간의 관계가 분명해지므로 방법론을 적용하여 핵심 자산을 개발하는 과정이 수월해지며 핵심 자산을 개발하기가 편리하다. 따라서 핵심 자산의 메타데이터에 대한 지원여부가 중요하다. 각 방법론들이 제공하는 산출물에는 차이가 있다. 위에서 제시하고 있는 네 가지 산출물 외에 각 방법론이 제공하는 특별한 산출물이 무엇인지 비교해보는 과정 역시 필요하다.

4.3 적용 지침 측면

본 절에서는 PLE 방법론들을 적용 지침 측면으로 비교하기 위해서 다음과 같은 항목으로 나누었다. 각 항목은 기존의 방법론을 연구하는 과정에서 적용 지침을 비교하는데 중요한 측면이라고 판단되는 부분으로 결정하였다. 프로젝트 수행 시 적용 지침이 분명한 방법론을 선택해서 사용한다면 산출물이 명확하며 제품 개발도 용이하다. 그러나 현재 발표된 PLE 방법론들은 지침을 개념적인 부분으로만 제시하는 경우가 있어서 지침의 상세 정도를 비교 한다. 제품 개발 시간과 프로젝트의 효율성을 증가시키기 위해서는 개발 순서가 명확히 제시되어있는 방법론을 선택하는 것이 좋다. 그러므로 방법론에서 제시하는 Stepwise 진행 순서가 중요하다. 또한 방법론에서 세부 업무 지침을 제공하면 프로젝트를 수행하는데 혼란이 줄어들 것이며 프로젝트 성공 가능

성도 증가할 것이다. 소프트웨어 개발 방법론들을 연구한 결과 각 방법론이 제공하는 Stepwise 진행 순서와 세부 업무 지침의 구체적인 정도가 정비례하는 것은 아니었다. 일부 방법론은 방법론을 적용하는 진행 단계에 대한 설명은 구체적이지만, 각 단계에서 적용해야 할 지침은 명확히 기술되어 있지 않은 경우도 있으므로 두 가지 기준으로 나누어 비교한다. 도메인 공학 프로세스에서 지침 제공의 상세성은 개발자들이 방법론을 선택하는데 큰 영향을 미칠 수 있으므로 도메인 공학을 위한 지침 제공 여부에 대한 비교 작업을 한다. 그리고 아키텍처는 도메인 공학 프로세스에서 핵심 적인 산출물이며 도메인 공학 지침이 대체로 상세하게 제시되어 있다고 하더라고 아키텍처를 개발하는 설계 지침 부분이 명확하지 않으면 핵심 자산을 개발하기 어려울 것이라고 생각하여 아키텍처 설계 지침의 상세도를 도메인 공학을 위한 지침과 분리해서 비교한다. 애플리케이션 공학 프로세스를 위한 지침 제공의 상세성은 애플리케이션 개발자들이 방법론 선택을 결정하는데 영향을 주므로 비교한다.

4.4 특징 측면

특징 측면의 비교는 각 방법론의 프로세스 측면, 산출물 측면, 적용 지침 측면에서 언급하지 않은 부분과 다른 방법론과 구별되는 고유한 특징을 비교한다. 그리고 특징이 해당 프로세스에서 의미적으로 중요한 부분인지 아닌지를 확인한다.

5. PLE 방법론의 기술적 비교 결과

5.1 프로세스 측면

본 절에서는 지금까지 발표된 PLE 방법론들을 제시한 비교 기준에 따라 프로세스 측면으로 비교한다.

본 논문의 비교 결과를 프로세스 측면에서 살펴보면, 대부분의 PLE 방법론들이 프로젝트 결과물에 대한 시장성을 전망하는 비즈니스 케이스 분석이나 프로젝트 도메인 영역을 설정하는 PL 범위 설정 단계를 가지고 있음을 확인할 수 있다.

핵심 자산 개발의 경우 도메인 공학, 핵심 자산 개발, PuLSE-CDA, PuLSE-DSSA, 애플리케이션 패밀리 공학, PL 자산 개발 프로세스, 프레임워크 공학, 제품 계열 공학, 제품 패밀리 공학 등으로 이를 지칭하는 용어에는 큰 차이가 있지만, 대부분 공통성과 가변성을 분석하는 의미를 포함하므로 핵심 자산을 개발하는 단계로 본다. 핵심 자산의 구성 요소 역시 표준화가 되어 있지 않아서 방법론간 차이가 크지만, 대부분의 방법론이 범용(generic) 아키텍처, 컴포넌트, 컴포넌트 간의 관계, 의사 결정 모델 등의 요소를 포함하고 있다[7]. 그러나 FAST 방법론이 핵심 자산의 구성 요소로 각 제품을 구현하기 위한 환경을 포함해서 패밀리 멤버에게 공통적인 모든 자산을 언급하고 있는 것처럼 방법론간 특징적인 부분도 있다. 이 논문은 방법론 전반에 대한 비교 논문이므로 핵심 자산에 대한 구체적인 차이는 더 이상 언급하지 않는다.

애플리케이션 개발의 경우 핵심 자산의 개발 단계와 마찬가지로 대부분의 방법론이 지원하고 있으나 PuLSE가 PuLSE-I라는 명칭을 사용하는 것과, 방법론에서 제품 공학 또는 애플리케이션 공학이라는 용어를 일반적으로 사용하고 있는 점, QADA가 애플리케이션 개발 단계를 지원하지 않는다는 것을 확인할 수 있다.

제품 계열 진화와 유지 보수 지원은 FAST PASTA가 패밀리를 변경할 수 있으며 PuLSE가 PuLSE-EM 단계를 수

〈표 1〉 PLE 방법론의 프로세스 측면 비교

비교 기준	PL 범위 설정 단계	핵심 자산 개발 단계	애플리케이션 개발 단계	PL 진화 및 유지보수성	프로젝트 관리
FAST PASTA	P.L.범위 설정	도메인 공학	애플리케이션 공학	패밀리 변경	프로젝트 관리
SEI SPL	PL 영역 설정 (패밀리 정의) 비즈니스 케이스 분석	핵심 자산 개발	제품 개발	지원 안함	관리
PuLSE	PuLSE-Eco	PuLSE-CDA, PuLSE-DSSA	PuLSE-I	PuLSE-EM	지원 컴포넌트
Bosch의 프로세스	비즈니스 케이스 분석 영역 설정	애플리케이션 패밀리 공학 (PL SE 아키텍처 개발)	애플리케이션 공학	모든 자산의 진화 (즉, 아키텍처, 컴포넌트, 제품)	지원 안함
FOPLE	도메인 영역 설정 시장 전망	PL 자산 개발 프로세스 도메인 공학	제품개발 프로세스 애플리케이션 공학	지원 안함	지원 안함
ESAPS	도메인 범위 설정 자산 범위 설정	도메인 공학	애플리케이션 공학	진화관리	프로세스 관리 자원 관리 테스트 관리
KobrA/PoLITe	지원 안함	프레임워크 공학	애플리케이션 공학	변경 관리, 유지보수 계획 진화 그래프	프로젝트 관리 형상 관리
Alexandria	도메인 분석	제품 계열 공학	제품 공학	진화	조직과 관리 SW 품질 관리
COPA	정책과 계획	제품 패밀리 공학 플랫폼 공학	제품 공학	진화 관리	기술과 인력 관리
QADA	PL 도메인 영역 설정 경제성 분석	공통성과 가변성 분석하며 아키텍처 분석과 설계에 의미 함축	지원 안함	지원 안함	지원 안함

행하는 것, Bosch가 모든 자산을 진화 할 수 있다고 말한 부분, KobrA가 변경 관리와 유지 보수 계획을 언급한 부분과 진화 그래프로 PL 진화를 지원하기 위한 산출물들을 정의하는 것, COPA의 진화 관리 과정 등을 통해서 확인 가능하다.

프로젝트 관리는 FAST PASTA의 프로젝트 관리 단계, SEI SPL의 관리, PuLSE의 지원 컴포넌트 단계, ESAPS의 프로세스 관리/자원 관리/ 테스트 관리에서 하며, KobrA는 생산 조건의 분석·조사·측정을 포함하는 프로젝트 관리와 형상 관리 부분에서 한다. 또한 Alexandria는 조직(Organization) 과 관리 단계와 SW 품질 관리 단계에 하며, COPA에서는 기술과 인력을 관리하는 부분에서 프로젝트 관리를 언급한다.

5.2 산출물 측면

본 절에서는 지금까지 발표된 PLE 방법론들을 제시한 비교 기준에 따라 산출물 측면으로 비교한다.

PLE 방법론을 산출물 측면으로 비교해보면 SEI SPL, PuLSE, Bosch의 프로세스, FOPLE, ESAPS, Alexandria, COPA, QADA가 범용 또는 참조 아키텍처라는 이름으로 범용 아키텍처를 대체적으로 잘 표현하고 있었으며, FAST PASTA는 범용 아키텍처에 관한 언급을 하지 않고, KobrA는 범용 아키텍처의 의미만을 함축하고 있음을 확인할 수 있다.

컴포넌트가 존재하는지를 비교하는 항목에서는 PLE가 컴포넌트 기반 개발의 확장 개념이라는 부분에서 예상 가능하듯, 모든 PLE 방법론이 설계와 구현 단계에 컴포넌트를 포함한다는 것을 확인할 수 있다. 그러나 각 방법론은 컴포넌트를 지칭하는 부분에서 차이가 있는데 FOPLE의 컴포넌트가 서브 시스템, 프로세스, 모듈 단위로 나뉜다는 것과 KobrA에서 말하는 KOMPONENT 단계가 명세, 실체화(realization) 단계, 구현 단계로 나뉘는 것, COPA에서 컴포넌트를 제품 패밀리 플랫폼이라고 언급하며 서브 시스템과 인터페이스를 가진다는 점이 특징적이다.

PLE 개발 방법론에서 핵심 자산 개발 단계와 관련된 산출물인 의사 결정 모델은 대부분의 프로세스가 가지고 있다. 그러나 SEI SPL이 의사 결정 모델을 가변성을 첨부한 프로세스에 정의하는 것, Bosch의 프로세스와 FOPLE가 의사 결정 모델과 관련된 부분을 휘처 모델에 포함하는 것, QADA에서는 의사 결정 모델이 정확히 명세되어 있지 않지만 시스템 환경과 관련된 가변성을 포함하며 품질 특성을 만족시키는 가변성으로 표현한다는 점이 특징적이다.

핵심 자산 메타데이터의 경우 ESAPS와 KobrA에 잘 정의되어 있다고 판단된다. ESAPS는 UML을 확장해서 메타데이터를 표현할 수 있다고 언급하며 자산, 아키텍처의 개념적 프레임워크, PL 요구사항 메타 모델 등을 제시한다. 다른 방법론들은 메타데이터를 언급하지 않거나 판단할 수 있을 만큼의 충분한 자료를 제공하지 않고 있다.

그 밖에 PLE 방법론들이 제시하는 특징적 산출물들은 FAST PASTA의 애플리케이션 모델링 언어(AML)와 개발 도구를 제시하는 것, SEI SPL이 SE뿐만 아니라 기술적으로 일반적인 분야에 대한 언급을 하고 있다는 점, PuLSE가 발전을 위한 기준(baseline) 프로파일을 정의하는 것, Bosch의 프로세스가 휘처 그래프를 사용하고 있다는 점, FOPLE이 품질 특성을 포함한 휘처 모델을 사용하는 것, ESAPS가 도메인 전문 용어와 참조 요구사항을 사용하며 개발 도구를 지원한다는 점, KobrA가 진화 그래프를 사용하는 것과 컴포넌트를 KCOMPONENT라고 지칭한다는 점, Alexandria

〈표 2〉 PLE 방법론의 산출물 측면의 비교

비교 기준	범용 아키텍처	컴포넌트	의사 결정 모델	핵심 자산의 메타데이터	기타 산출물
FAST PASTA	지원 안함	모듈 단위의 패밀리 설계	의사 결정 모델	지원 안함	AML 개발 도구
SEI SPL	범용 아키텍처	SW 컴포넌트와 COTS 컴포넌트	가변성 첨부 프로세스에 정의	지원 안함	SE와 기술적으로 일반적인 분야
PuLSE	컴포넌트와 커넥터를 포함한 참조 아키텍처	도메인 모델 워크 프로덕트에 해당	의사 결정 모델	지원 안함	발전을 위한 기준 프로파일(baseline profile) 개발 환경
Bosch의 프로세스	PL 아키텍처	컴포넌트>시스템	휘처 모델에 포함	지원 안함	휘처 그래프
FOPLE	참조 아키텍처	서브 시스템 >프로세스>모듈	휘처 모델에 포함	지원 안함	품질 특성을 포함한 휘처 모델
ESAPS	참조 아키텍처	재사용 가능한 컴포넌트	의사 결정 모델	메타데이터 잘 정의됨	도메인 전문 용어 참조 요구사항 개발 도구
KobrA/PoLITe	의미 함축	KOMPONENT와 컴포넌트KOMPONENT의 세 단계	의사 결정 모델	잘 정의됨	PL의 변경을 관리하기 위한 진화 그래프 사용 KOMPONENT
Alexandria	참조 아키텍처	컴포넌트	지원 안함	지원 안함	개발 도구
COPA	참조 아키텍처	제품 패밀리 플랫폼	지원 안함	지원 안함	아키텍처 관련 산출물 개발 환경
QADA	아키텍처가 공통성과 가변성을 포함	컴포넌트	정확히 명세하지 않음	지원 안함	아키텍처 관련 산출물

가 개발 도구를 지원 한다는 점, COPA와 QADA가 아키텍처 관련 산출물을 포함하고 있다는 것 등을 들 수 있다.

5.3 적용 지침 측면

본 절에서는 지금까지 발표된 PLE 방법론들을 제시한 비교 기준에 따라 적용 지침 측면으로 비교한다.

적용 지침 측면의 비교는 각 방법론이 비교 기준을 얼마나 상세하게 표현하고 있는가를 강하게 지원, 지원, 약하게 지원, 의미 함축, 지원 안함으로 등급을 나누어 표현하였다. 강하게 지원은 Stepwise 진행 순서나 지침을 구체적이며 상세하게 제공하고 있는 경우이며 지원은 Stepwise 진행 순서나 지침을 전반적으로 지원은 하고 있지만 구체적이며 직접적으로 제공하지 않는 경우이다. 약하게 지원은 지침의 일부만을 간략하게 제공하고 있는 경우이며 의미 함축은 방법론에서 지침을 구체적으로 제공하지는 않지만 여러 자료를 통해서 유추할 수 있는 경우이다. 그 외에 지침을 제공하지 않거나 지침에 관한 자료가 없다고 판단되는 경우에는 지원 안함이라는 용어를 사용하였다.

<표 3>에서 확인할 수 있듯이 PuLSE, ESAPS 방법론은 자세한 프로세스 Stepwise 진행 순서를 제시하며 FAST PASTA, FOPLE, KobrA의 경우 일부 단계만 제시하므로 지원으로 결정한다.

세부 업무 지침을 제시하는 정도는 ESAPS가 체계적이지는 않지만 지침을 강하게 지원한다고 판단되며 FOPLE, KobrA는 ESAPS 보다는 부족하지만 대부분 제시한다. 그 외의 방법론들은 지침을 일부 제시하거나 또는 거의 지원하지 않는다고 본다. 그 중 SEI SPL의 경우에는 업무 지침 중에서 애플리케이션 공학 부분은 거의 지원하지 않음을 확인할 수 있으며, Alexandria는 지침에 대한 지원 여부를 확인할 수 있는 자료가 제한적이다.

아키텍처 설계 지침 항목에 있어서 Bosch의 프로세스, QADA가 자세한 지침을 제시하며 PuLSE, FOPLE, ESAPS, COPA가 지침의 일부를 제시한다. 그리고 KobrA는 지침과 관련된 의미만을 일부 내포하고 있다.

도메인 공학을 위한 지침의 경우 ESAPS가 자세하게 제시하고 있다. QADA는 ESAPS 보다는 구체적이지 않지만 많은 부분을 지원하고 있으며, KobrA는 공통성과 가변성 지침을 일부 포함하여 지원하고 있다. 그 밖에 SEI SPL은 메커니즘 목록을 통해서 도메인 공학을 위한 지침을 약하게 지원하고 있고, PuLSE와 Bosch의 프로세스 역시 도메인 공학을 위한 지침을 일부 제공하고 있다. FAST PASTA는 도메인 공학을 위한 지침을 도메인 전문가에게 의지하고 있으며, FOPLE은 훠처 모델링에 지침의 의미를 함축하고 있음을 확인할 수 있다.

애플리케이션 공학을 위한 지침은 대부분의 방법론이 자세한 지침을 제공하지 않는다. 그 중 ESAPS와 KobrA가 일부 지침을 제시하고 있으나 각 방법론의 다른 개발 단계 지침의 상세화 정도에는 미치지 못하는 수준이며, Bosch의 프로세스가 가이드 라인의 일부만을 제시하고 있다.

5.4 특징 측면

본 절에서는 앞에서 제시하고 있는 방법론간 비교 결과에는 포함되지 않는 특징을 중요도에 따라 비교한다.

PLE 방법론의 고유한 특징 측면 비교는 각 방법론의 특징을 중요함, 보통, 중요하지 않음의 세 단계로 구별하여 비교한다.

FAST가 도메인에 특정한 애플리케이션 공학 프로세스를 정의하는 것, PuLSE가 프로세스를 각 PL에 맞도록 특화하며 이전 방법론을 사용한 경험으로부터 획득한 정보인 베이스 라인 프로파일을 향상시킬 수 있는 것, FOPLE가 훠처 모델을 사용하는 것, ESAPS의 핵심 자산이 구현 수준이며 프로세스를 테일러링하며 메타 모델을 사용해서 산출물을 정의하는 것은 중요한 특징이다. KobrA도 ESAPS와 마찬가지로 메타 모델을 통해서 산출물을 정확하게 정의하며 Alexandria가 구현된 핵심 자산을 제시하며 도구를 지원한다는 특징 역시 중요하다.

FAST PASTA가 AML을 사용해서 애플리케이션을 생성하고 분석하며 코드와 문서를 생성할 때 사용되는 애플리케이션 공학 수행 환경 과정을 포함한다는 것, SEI SPL이 SE 일반적인 분야에 대해 언급하는 것, Bosch의 방법론이 아키텍처 기반의 접근을 한다는 점, ESAPS가 산출물간 추적성과 자동화 도구의 필요성을 강조한다는 것, KobrA가

〈표 3〉 PLE 방법론의 적용 지침 측면 비교

비교 기준	Stepwise 진행 순서	세부 업무 지침	아키텍처 설계 지침	도메인 공학을 위한 지침	애플리케이션 공학을 위한 지침
FAST PASTA	지원	약하게 지원	지원 안함	의미 함축	지원 안함
SEI SPL	약하게 지원	약하게 지원	지원 안함	약하게 지원	지원 안함
PuLSE	강하게 지원	약하게 지원	지원	약하게 지원	지원 안함
Bosch의 프로세스	약하게 지원	약하게 지원	강하게 지원	약하게 지원	약하게 지원
FOPLE	지원	지원	지원	의미 함축	지원 안함
ESAPS	강하게 지원	강하게 지원	지원	강하게 지원	지원
KobrA/PoLITe	지원	지원	의미 함축	지원	지원
Alexandria	약하게 지원	지원 안함	지원 안함	약하게 지원	지원 안함
COPA	약하게 지원	약하게 지원	지원	지원 안함	지원 안함
QADA	약하게 지원	약하게 지원	강하게 지원	지원	지원 안함

〈표 4〉 PLE 방법론의 특징 측면 비교

중요도	의미적으로 중요함	보통	의미가 덜 강조됨
FAST PASTA	도메인에 특정한 애플리케이션 공학 프로세스가 정의됨	AML 애플리케이션 공학 수행 환경	
SEI SPL		SE 일반적인 분야 언급	
PuLSE	프로세스를 각 P.L에 맞도록 특화 가능 Baseline profile이 향상됨		
Bosch의 프로세스		아키텍처 기반의 접근	
FOPLE	휘처 모델		
ESAPS	구현된 핵심 자산 프로세스 테일러링 메타 모델을 통한 산출물 정의	산출물간 추적성 강조 자동화와 도구 필요성 강조	UML 확장해서 메타 모델 표현
KobrA/PoLITe	메타 모델을 통한 산출물의 정확한 정의	KOMPONENT의 세 단계 MDA의 모델 변환과 일치하는 정제 개념	UML을 따름 스테레오 타입으로 가변성 표현
Alexandria	도구 (tool) 지원 구현된 핵심 자산	Eclipse 기반 연구	UML 확장 진화를 위해서 MDD 개념 사용
COPA		컴포넌트 프레임워크 컴포넌트 Aspect Design	도메인 분석 시UML사용
QADA		아키텍처 기반의 접근 설계에서 품질 특성을 고려	

KOMPONENT의 속성을 설명하는 명세 단계와 KOMPONENT에 한정된 디자인을 보여주며 어떻게 요구사항을 만족시키는지를 설명하는 구체화 단계, 구체화 단계를 표현하는 구현 단계의 총 세 단계를 갖고 MDA 모델 변환과 일치하는 정제 개념을 포함 한다는 것, Alexandria가 eclipse 기반의 연구를 한다는 것, COPA가 컴포넌트 기반의 연구를 하며, QADA가 아키텍처 기반의 접근을 한다는 것 등은 각 방법론의 특징이라고 추출한 만한 사항이다.

그 외에 ESAPS가 UML을 확장해서 메타 모델을 표현하는 것, KobrA가 UML을 따르고 스테레오 타입을 사용해서 가변성을 표시하며, Alexandria가 UML을 확장하고 진화를 위해서 MDD(Model Driven Development)를 사용하는 것, COPA가 도메인 분석 시 UML을 사용한다는 특징도 의미적으로 중요성이 강조되지는 않지만 각 방법론의 특징이다.

6. 방법론 비교 요약 및 해석

6.1 비교 결과 요약

본 논문에서는 지금까지 발표된 PLE 방법론을 정해진 기준에 따라 비교함으로써 방법론 간 공통점 및 차이점과 특징을 확인했다. <표 5>는 5장의 방법론 비교 결과를 요약한 표이다. 잘 지원됨의 의미는 방법론이 비교 요소와 관련된 내용을 충분이 제공하고 있다는 의미이며 지원은 방법론이 산출물이나 프로세스 등을 제공하고는 있지만 구체적이거나 세부적인 내용까지는 언급하지 않는다는 의미이다. 부분 지원은 방법론이 해당 비교 요소에 대한 언급 정도만 하고 있는 경우이다. 특징 측면은 다른 방법론과 구별되며 방법론의 표준화 작업이 수행될 때 포함되면 좋을 만한 특징을 가지고 있는 방법론의 경우 중요한 특징이라고 표현하며 기타 다른 방법론과 구별되는 특징이 있지만 PLE 개념에서 중요하게 판단되지 않는 내용의 경우는 특징 부족으로 표현한다.

프로세스 측면에서 살펴보면 방법론들 중에서 FAST PASTA, SEI SPL, PuLSE, ESAPS, KobrA가 프로세스와 프로세스의 유지 보수 부분까지 전반적으로 명확하고 분명하게 제시하고, 그 외의 방법론들은 프로세스와 유지 보수 부분을 언급하고는 있지만 일부 단계만 한정해서 제시하거나 또는 프로세스의 특정 단계만을 언급하는 방법론도 있다.

〈표 5〉 비교 결과 최종 요약

비교 기준	프로세스 측면	산출물 측면	적용 지침 측면	특징 측면
FAST PASTA	잘 지원됨	지원	지원	중요한 특징
SEI SPL	잘 지원됨	지원	부분 지원	특징 부족
PuLSE	잘 지원됨	지원	부분 지원	중요한 특징
Bosch의 프로세스	지원	지원	부분 지원	특징 부족
FOPLE	지원	지원	부분 지원	중요한 특징
ESAPS	잘 지원됨	잘 지원됨	잘 지원됨	중요한 특징
KobrA/ PoLITe	잘 지원됨	잘 지원됨	잘 지원됨	중요한 특징
Alexandria	부분지원	부분 지원	부분 지원	중요한 특징
COPA	지원	부분 지원	부분 지원	특징 부족
QADA	부분 지원	부분 지원	부분 지원	특징 부족

산출물 측면으로 살펴보면 ESAPS와 KobrA 방법론이 핵심 자산의 메타데이터 등을 제시함으로써 산출물을 충분히 제시하고 있다. 적용 지침 측면으로 살펴보면 ESPAS와 KobrA가 가이드 라인을 포함한 지침을 전반적으로 잘 표현하고 있었으며 대부분의 방법론들은 애플리케이션 공학을 위한 지침을 제시하는 부분에서는 많이 부족했는데, 애플리케이션 개발보다는 핵심 자산의 개발에 중점을 두고 있음을 알 수 있다. 각 방법론들은 다른 방법론들과 구별되는 특징을 가지고 있다. 그 중에서 FAST PASTA가 제시하는 도메인에 특정한 애플리케이션 공학 프로세스, PuLSE가 제시하는 프로세스 특화 방법, FOPLE이 제시하는 휘처 모델,

ESAPS와 Alexandria가 핵심 자산을 구현 수준으로 정의하는 것, KobrA가 제시하는 메타 모델 활용 방법 등은 중요한 특징으로 판단된다.

6.2 방법론간 공통성이 높은 항목

앞 절에서 제시한 비교 결과를 통해서 확인할 수 있듯이 대부분의 PLE 방법론들은 핵심 자산을 개발하는 단계나 애플리케이션을 개발하는 단계에 대한 내용을 포함하고 있다. 그리고 도메인의 제품들간 생산 효율성을 중시하고 개발 과정에서 많은 비용을 감수해야 하므로 비즈니스 케이스 분석과 도메인의 영역을 설정하는 단계를 중요시 한다. 뿐만 아니라 PLE 방법론의 대표적 산출물인 범용 아키텍처와 공통성과 가변성을 포함한 의사 결정 모델도 대부분의 방법론이 제시하고 있다. 핵심 자산 개발을 위한 지침을 제공하는 상세 정도를 비교 했을 때 애플리케이션 공학을 위한 지침의 지원이 상대적으로 부족한 수준으로 제공하고 있음을 알 수 있는데, PLE 방법론들이 핵심 자산 개발에 비중을 두고 있기 때문이다. 현재 PLE 기반의 개발을 지원하는 여러 방법론이 존재하고 있지만 아직까지 방법론 표준화는 이루어지지 않고 있다. 본 논문에서 제시하고 있는 방법론의 공통 항목들은 PLE 방법론의 필수 요소로, 표준 방법론을 개발하는데 있어서 기본 프로세스와 산출물을 정하는데 도움이 될 수 있을 것이다.

6.3 방법론간 공통성이 적은 항목

지금까지 발표된 PLE 방법론들 중에는 프로세스를 발전시키고 유지 보수하는 단계와 프로젝트 관리 단계를 지원하지 않는 방법론이 많이 있다. 그리고 ESAPS와 KobrA가 메타 모델을 제시하고 있는 점과 ESAPS와 Alexandria가 핵심 자산을 구현 수준으로 언급하는 것도 특징적이다. 그 밖에 PLE 개발 단계에 기본적으로 포함되지는 않지만 각 방법론이 고유하게 제시하고 있는 특징들이 있는데, 그 중에서 FAST PASTA가 제시하는 도메인 특정 애플리케이션 공학 프로세스, PuLSE가 제시하는 프로세스 특화 방법, FOPLE이 제시하는 휘처 모델, KobrA가 제시하는 메타 모델 활용 방법 등의 특징들이 PLE 방법론을 사용하는 제품 개발에 전반적으로 활용되면 유용할 것이다. 본 논문에서 제시한 비교 결과를 통해서 PLE 방법론에 필요한 요소임에도 불구하고 표현되어 있지 않은 단계와 활동을 비교할 수 있으며, 각 방법론을 정제하거나 표준 방법론을 표현할 때, 이 부분을 참고해서 활용 할 수 있을 것으로 예상한다.

6.4 방법론의 개선점

아직까지 프로세스나 지침이 완벽하게 정의되어 있는 PLE 방법론이 존재하지 않으며, 각 방법론이 표현하는 정보의 상세 정도는 차이가 많다. 그러므로 각 방법론들은 부족한 측면을 보완해야 할 것이다. 이에 각 방법론이 개선되어야 할 부분을 <표 6>에서 제시한다.

대부분의 PLE 방법론이 개념적인 측면으로만 제공되고 있기 때문에, 제품 개발을 위한 지침이나 산출물에 대한 정의와 작성 형식에 대한 정보는 부족하다. 따라서 각 방법론들은 지침을 구체적으로 제시해야 하며, 그 중에서 ESAPS는 정의 되어 있는 지침을 체계화 시킬 필요가 있다고 생각한다. 또한 대부분의 방법론이 애플리케이션 개발 과정과 산출물에 대해서는 제시하는 정보가 부족하다. 애플리케이션 개발 과정과 생산성을 향상시키는 방법에 대한 연구가 지속되어야 할 것이다.

〈표 6〉 각 방법론의 개선점

방법론	각 방법론의 개선점
FAST PASTA	범용 아키텍처와 애플리케이션 공학을 위한 지침이 부족
SEI SPL	지침의 상세도가 낮음
PuLSE	애플리케이션 공학을 위한 지침이 지원되지 않음
Bosch의 프로세스	지침의 상세도가 낮음
FOPLE	도메인 공학에 비해 애플리케이션 공학 지침이 미흡
ESAPS	프로젝트를 기반으로 한 방법론이므로 지침이나 산출물에 대한 정보는 자세하나 체계적으로 정리되어 있지 않음
KobrA/PoLITe	전반적인 지침은 자세하지만 일부 핵심적인 지침 부족
Alexandria	지침이 많이 부족
COPA	아키텍처 설계 지침을 제외하고는 전반적인 지침 부족
QADA	아키텍처 설계 지침을 제외하고는 전반적인 지침 부족

7. 결 론

PLE는 SW 재사용을 강조한 개발 기술로 최근 학계와 산업계의 주목을 받고 있다. PLE는 1990년경에 소개되었으며 유럽에서는 PLE 방법을 적용해서 ESAPS, KobrA, Alexandria, Café, ARES, PRAISE 등 다수의 프로젝트가 진행되었다. 또한 이 프로젝트 중에서 Alexandria, ESAPS, KobrA는 방법론으로 정립되는 등 많은 연구가 진행되고 있다. 본 논문에서는 현재 발표된 PLE 방법론들을 비교하여 방법론을 선택하는 사용자들이 좀 더 합리적인 결정을 할 수 있도록 방법론 평가를 수행했다. 방법론들을 비교하기 위해서 네 가지 비교 기준을 선정하고 각 비교 기준에 해당하는 구체적 항목을 고려해서 FAST, SEI SPL, PuLSE, Bosch의 프로세스, FOPLE, ESAPS, KobrA, Alexandria, COPA, QADA 방법론을 비교했다. 본 논문에서 제시한 비교 결과를 사용하면 PLE 개념과 필수 요소를 확인할 수 있으며, 각 방법론에서 부족한 개발 단계나 산출물을 확인한 비교 결과는 PLE 방법론을 선택할 때 기준이 되는 정보가 되기도 한다. 또한 기존의 PLE 방법론을 정제하는 과정에서 활용할 수 있으며 PLE 방법론을 표준화 시키는 부분에서도 기반이 되는 연구가 되고자 했다.

참 고 문 헌

[1] Matinlassi, M., "Comparison of Software Product Line Design Methods: COPA, FAST, FORM, KobrA and QADA," icse, 5, pp.127-136, 2004.

[2] Vehkomaki, T. Kansala, K., "A Comparison of Software Product Family Process Frameworks," LNCS 1951, pp. 135-145, 2000.

[3] Weiss, D. et al., Software Product-Line Engineering, Addison-Wesley, 1999.

[4] Clements, P. and Northrop, L., Software Product Lines: Practices and Patterns, Addison Wesley, Aug., 2001.

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

[6] Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wust, J. and Zettel, J., Component-Based Product Line Engineering with UML, Addison Wesley, 2002.

[7] Jan, B., Design and use of software architectures, Addson-Wesley, 2000.

[8] 이재준, 강교철, "프로덕트 라인 소프트웨어 개발 프로세스," 정보과학회지, Vol.20, No.3, pp.23-30, 3, 2002.

[9] Streitferdt, D. et al., "Details of Formalized Relations in Feature Models Using OCL," Engineering of Computer-Based Systems, 2003. Proceedings. 10th IEEE International Conference and Workshop on the, 7-10 April, 2003

[10] Jaejoon L., Kyo K., and Sagoong K., "A Feature-Based Approach to Product Line Production Planning," SPLC2004, LNCS3154, pp.183-196, 2004

[11] Knauber, P., and Succi, G., "Perspective on Software Product Lines, " Workshop #15 at 22nd ICSE

[12] ITEA, Web Site for ESAPS, at URL : http://www.esi.es/en/Projects/esaps/esaps.html

[13] Philips Research, "COPA-A Component-Oriented Platform Architecting Method for Families of Software-Intensive Electronic Products," Presentation Material (in PowerPoint/PDF), SPLC, 2000.

[14] 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.

[15] Kettemann, S., Muthig, D., and Anastasopoulos, M., Product Line Implementation Technologies : Component Technology View, Technical Report, No. 015.03/E, IESE, March, 2003.

[16] ILMENAU, Web Site for Alexandria, at URL: http://www.theoinf.tu-ilmenau.de/~riebisch/pld/index.html.

[17] ESS, Web Site, at URL : http://www.ess.co.at/ECOSIM/development.html.

[18] Bosch, F., "Evaluation of Sofware Development Life Cycle Methodology Implementation," ACM'SIGSOFT, SOFTWARE ENGINEERING NOTES, Vol.7 No.1, pp.45, Jan, 1982.

박 신 영

e-mail : sypark@otlab.ssu.ac.kr

1999년~2003년 동덕여자대학교 국어국문학과, 데이터정보학과 복수전공(학사)

2004년~2006년 숭실대학교 대학원 컴퓨터학과 졸업예정(공학석사)

관심분야 : 컴포넌트 기반 개발 (CBD), 제품 계열 공학 (PLE), 모델 기반 아키텍처 (MDA)

김 수 동

e-mail : sdkim@comp.ssu.ac.kr

1984년 Northeast Missouri State University 전산학(학사)

1988년 The University of Iowa 전산학(석사)

1991년 The University of Iowa 전산학(박사)

1991년~1993년 한국통신 연구개발단 선임연구원

1994년~1995년 현대전자 소프트웨어연구소 책임연구원

1995년~현재 숭실대학교 컴퓨터학부 부교수

관심분야 : 객체지향 S/W공학, 컴포넌트 기반 개발(CBD), 제품계열 공학(PLE), 모델 기반 아키텍처(MDA)