제품 계열 방법론의 핵심 자산 구성요소 비교

박신영[1], 김수동[1], 양영종[2]
[1]숭실대학교 대학원 컴퓨터학과, [2]ETRI 기반기술연구소 SE 연구팀
sypark@otlab.ssu.ac.kr , sdkim@ssu.ac.kr, yangyj@etri.re.kr

A Comparison of Core Assets in Product Line Engineering

Shin Young Park[1], Soo Dong Kim[1], Young Jong Yang[2]
[1] Dept. of Computer Science, Soongsil University
[2]ETRI Software Engineering Research Team

요 약

제품 계열 공학 (Product Line Engineering, PLE)은 유사한 도메인에 속한 제품들로부터 공통성과 가변성을 분석하여 재사용 가능한 핵심 자산 (core asset)을 만들고, 만들어진 핵심 자산을 사용하여 어플리케이션을 개발하는 제품 개발 기술이다. 그러나 아직까지는 표준화된 방법론이 존재하지 않아 산업계와 학계는 해당 제품에 적합한 방법론 또는 프로세스의 일부를 선택하는 과정에서 어려움을 겪고 있다. 본 논문에서는 재사용성을 강조하는 PLE 방법론의 핵심 자산 구성 요소를 비교하여, 산업계나 학계가 핵심 자산을 개발하는 과정에서 효율성을 높일 수 있는 방법론을 선택하는데 도움이 되는 기반 연구를 한다. 나아가 본 논문은 비교 기준이 된 요소들과 비교 결과가 PLE 방법론을 표준화 하는 과정에서도 사용될 수 있도록 정보를 제공 한다.

1. 서론

PLE는 하나의 도메인에 속한 제품들로부터 공통성과 가변성을 분석하여 재사용 가능한 핵심 자산을 만들고 만들어진 핵심 자산을 사용하여 어플리케이션을 개발하는, 재사용 측면이 강조된 제품 개발 기술이다[1]. PLE는 제품 개발 시 만들어 놓은 핵심 자산을 사용하여 유사한 도메인에 속한 제품을 개발하므로, 핵심 자산 개발의 중요성이 강조된다. 1990년대 이후부터 최근까지 많은 제품 계열 방법론들이 발표되었으나, 지금까지도 표준화된 방법론이 존재하지 않을 뿐만 아니라 각 방법론이 제공하는 프로세스 정의나 구성요소 역시 큰 차이를 보인다. 따라서 학계나 산업계에서는 제품 계열 방법론을 사용할 때, 방법론 선택의 문제에 어려움을 겪고 있다. 본 논문에서는 PLE 방법론의 핵심 자산을 비교하여 학계나 산업계에서 핵심 자산을 가장 효과적으로 개발할 수 있도록 정보를 제공한다.

본 논문에서 의 구성은 다음과 같다. 2장의 기초 연구에서는 현재까지 발표된 PLE 방법론들을 소개하고, 3장에서는 PLE 핵심 자산의 구성요소를 비교할 기준을 설정한다. 4장의 구성요소 비교 요약 부분에서는 PLE 방법론들의 구성요소 지원 여부에 대한 비교 작업을 수행한 결과를 보여주며, 5장에서는 비교 결과에 대한 해석을 한다. 6장에서는 본 논문의 결론을 내리며 향후 연구 과제를 제시한다.

2. 기초 연구

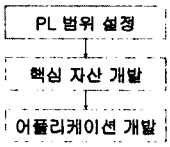
1992년 FAST 방법론을 시작으로 지금까지 여러 개의 PLE 방법론이 소개되었다. 본 장에서는 현재 발표된 PLE의 각 방법론에 대한 간단한 소개를 한다.

*FAST*는 1992년 AT&T의 의해 소개되었고, 이 개념은 Lucent Technology에 의해 발전되어 1999년에 발표되었다[2]. *FORM*은 1998년에 POSTECH에서 개발한 휘처(feature) 기반 재사용 방법론으로 1990년 SEI에서 개발한 휘처 중심의 도메인 분석 방법인 FODA를 확장한 방법론이다[3]. *PuLSE*는 기업 업무 배경(enterprise context)에서 소프트웨어 제품 계열을 파악하고 개발을 가능하게 하기 위한 목적으로 IESE에서 개발하였다[4]. *SEI의 제품 계열 공학(SEI SPL)*은 Carnegie Mellon 소프트웨어 공학 대학에서 2001년에 개발하였다[5]. *KobrA*는 UML을 사용한 컴포넌트 기반 PLE로서 2001년에 IESE에 의해 개발되었다[6]. *PoLITe*는 KobrA를 기반으로 IESE가 개발하였다. PoLITe의 초점은 제품 계열(Product Line, PL) 구현 관련 기술에 있으며 이 기술은 제품을 위해 요구되는 광범위한 종류에 알맞은 기술의 체계적 선택을 위한 프레임워크로 통합된다[7]. *Bosch*는 2000년에 제품 계열 방법론을 발표하였으며, 아키텍처를 강조한다[8]. *COPA*는 업무(business), 아키텍처, 프로세스, 조직 이 가장 잘 조화되는 문제에 초점을 맞춘 메서드로 통신, 화상 의료 기기, 전자 기기와 관련된 어플리케이션 도메인을 개발하는 경험에 기반을 두고 Philips 연구실에서 2000년에 발표하였다[9]. *QADA*는 요구사항 공학과 아키텍처 설계에 중심을 두고 있으며, 어플리케이션 개발 과정을 생략하고 아키텍처 분석과 설계 단계에 핵심 자산 개발의 의미를 포함하고 있는 PLE 개발 방법론으로 2002년 VTT에 의해 개발되었다[10].

3. PLE 핵심 자산 구성요소 비교 기준

각 PLE 방법론은 핵심 자산 구성요소나 프로세스의 상세 정도 등 구체적인 방법론 세부 명세에서 큰 차이를 보이고 있다. 그러나 <그림1>에서 제시하는 것처럼 도메인을 분석하여

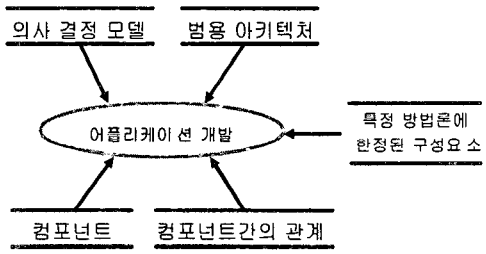
PL 범위를 설정하고, 핵심 자산을 만든 후 만들어진 핵심 자산을 사용하여 어플리케이션을 개발하는 일반적인 프로세스는 모든 PLE 방법론이 공통적으로 제시한다.



<그림 1> PLE 일반적인 프로세스

다음 장에서 비교하려는 핵심 자산에 대해 SEI SPL, Bosch 의 기법은 '재사용 가능한 자산'이라는 비교적 확실한 정의를 내리고 있으며, 그 외 방법론들의 핵심 자산 정의에 대해서는 참고 문헌이나 논문을 통해서 유사한 의미를 유추할 수 있다. PLE 방법론의 핵심자산 정의와 마찬가지로 핵심 자산 구성요소 역시 표준화된 정의가 되어 있지 않으므로 방법론간에 큰 차이를 보이고 있는데, 본 논문에서는 핵심 자산의 구성요소를 어플리케이션 개발에 사용되는 자산이라는 특화된 의미를 사용하여 다음과 같은 비교 기준으로 방법론을 비교한다.

의사 결정 모델은 어플리케이션 생산 시 핵심 자산의 가변성과 관련해서 결정해야 하는 사항들과 결정 순서에 대해 기술하는 모델[2]로써 핵심 자산의 구성요소가 된다. 범용 아키텍처[8]는 도메인 내의 모든 멤버들에게 공유되는 아키텍처로써 핵심 자산의 구성요소가 된다. 컴포넌트[5]는 아키텍처 내부에서 컴포넌트의 구조를 표현하는 역할을 담당하는 컴포넌트 간의 관계[10]와 함께 아키텍처에 포함되는 아키텍처 구성요소가 되며 또한 핵심 자산의 구성요소가 된다.



<그림 2> PLE 방법론의 핵심 자산 구성 요소

4. 구성요소 비교 요약

본 장에서는 어플리케이션 개발 과정에 직접 사용되는 일반적인 핵심 자산의 구성요소를 기준으로 각 방법론을 비교한다. 다음에 제시하는 <표1>은 비교 작업 수행 후 도출한 결과이다.

<표 1> 핵심자산 구성요소 비교

방법론	의사 결정 모델	범용 아키텍처	컴포넌트	컴포넌트 간의 관계
FAST	●			
FORM	○	●	●	
PuLSE	●	●	●	○
SEI SPL	○	●	●	
KobrA	●	○	●	○
PoLITe		○	●	
Bosch	○	●	●	●
COPA		●	●	○
QADA		●	●	○

●: 지원함 ○: 지원하는 것으로 유추됨

의사 결정 모델은 FAST, PuLSE, KobrA가 지원하고 있으며 FORM, Bosch의 기법은 휘처 모델에서 의사 결정 모델의 역할을 수행한다고 본다. SEI SPL은 첨부 프로세스가 의사 결정 모델의 역할을 수행한다.

범용 아키텍처는 FORM, PuLSE, COPA에서 참조 아키텍처로 지원하고 있으며 SEI SPL, Bosch의 기법, QADA에서는 제품 계열 아키텍처로 지원한다. 그 밖에 PoLITe에서는 컴포넌트 아키텍처로 지원하며, KobrA에서는 핵심 자산을 개발하는 과정에서 범용 아키텍처의 의미를 함축하고 있으나 구체적으로 명시하지는 않는다.

컴포넌트는 FORM, PuLSE, SEI SPL, Bosch의 기법, COPA, QADA 에서 컴포넌트 또는 재사용 가능한 컴포넌트로 명시하고 있다. 그 외에 KobrA에서는 범용 컴포넌트라는 명칭을 사용하며, PoLITe의 경우에는 컴포넌트가 핵심 자산이 된다고 명시하고 있지는 않지만 논문이나 참고 자료를 통해서 컴포넌트가 핵심 자산의 구성 요소임을 확인할 수 있다.

컴포넌트간의 관계는 Bosch의 기법과 KobrA에서 명시하고 있으며 PuLSE, COPA, QADA는 컴포넌트간의 관계를 명시하고 있지는 않지만, 컴포넌트나 아키텍처와 관련되어 핵심자산의 구성 요소로 유추 가능하다.

5. 구성요소 비교 해석

지금까지 여러 PLE 방법론이 발표되었으며, 각 방법론들은 SEI SPL, COPA등과 같이 개별적으로 만들어지거나 KobrA, PoLITe와 같이 관련되어 개발되었다. 따라서 방법론간에 구성 요소를 표현하는 부분에서 큰 차이가 발생하였다. 현재 발표된 방법론들은 대부분 도메인을 분석하고 핵심 자산을 개발하며 개발한 핵심 자산을 사용하여 어플리케이션을 개발하는 전체 프로세스는 대체로 동의를 하고 있으나 각 프로세스를 표현하는 방법이나 세부 프로세스, 산출물 등에서 방법론간 큰 차이를 보이고 있다.

<표1>은 각 방법론의 핵심자산 구성요소를 비교한 결과로, 핵심자산을 사용하여 어플리케이션을 개발하는 PLE 기본 개념을 고려하여 핵심자산의 구성요소 정의를 가정했다. 그리고 현재 발표된 각 방법론의 공통적인 요소를 추출했다.

의사 결정 모델은 유사한 도메인에서 공통성과 가변성을 구별하여 특정 어플리케이션 개발 시 해당 기능을 선택하는데 사용한다. 그러므로 공통성과 가변성을 고려해야 하는 PLE 방법론에서는 꼭 필요한 구성요소가 되야 하지만, 현재 일부 방법론의 프로세스에서는 지원되지 않고 있다.

범용 아키텍처는 대부분 참조 아키텍처라는 명칭으로 지원하고 있으나 FAST는 참조 아키텍처를 언급하지 않고 있으며, PoLITe는 KobrA와 관련해서 발표된 개발 방법론으로써 어플리케이션 개발에 초점을 맞추고 있으므로 컴포넌트 중심의 개발을 하며 범용 아키텍처의 개념이 포함되었다고 유추된다.

컴포넌트는 명시되어 있거나 또는 범용 아키텍처를 설명하는 과정에서 의미 또는 개념을 확인할 수 있다. 컴포넌트는 PLE에서 재사용의 단위가 되며 CBD에서도 유사한 기능을 재사용하기 위해 고려하고 있는 단위이다. 컴포넌트는 FAST를 제외한 모든 방법론에서 명시하고 있으며 FAST는 산출물이나 핵심 자산의 구성요소가 다른 방법론과 차이가 있어서 이런 결과가 나오는 것으로 유추된다.

일반적으로 아키텍처는 컴포넌트와 컴포넌트의 관계로 구성되며 컴포넌트의 관계는 아키텍처 내에서 컴포넌트간의 구조를 보여주는 역할을 담당한다. 그러나 컴포넌트 간의 관계는 PuLSE, KobrA와 아키텍처를 강조하는 방법론인 Bosch의 기법, COPA, QADA 방법론에만 명시적 또는 간접적으로 언급하고 있으며, 그 밖의 다른 방법론들은 언급이 미비하다. 컴포넌트 간의 관계를 핵심 자산의 요소로 지원하지 않거나 이에 대한 의미를 함축하고 있지 않은 다른 방법론들의 경우에 있어서도 해당 방법론들이 아키텍처와 컴포넌트를 지원하는 것으로 유추 해 볼 때, 컴포넌트간의 관계를 구성요소가 아니라고 부정하기 보다는 언급이 단지 미비한 것으로 유추한다.

각 방법론에서 핵심 자산의 구성요소로 언급하고는 있지만 본 논문에서 핵심 자산의 필수 구성요소로 판단하지 않아서 비교하지 않는 구성요소도 있다. FAST에는 패밀리 설계, 조립 매핑, 어플리케이션 모델링 언어(AML), 어플리케이션 공학 프로세스, 라이브러리, 생성 도구, 분석 도구, 문서가 있으며 PuLSE에선 제품 특징 정보, 영역 정의, 도메인 모델, 도메인 결정 모델, 아키텍처 결정모델, 커넥터가 있다. KobrA에는 범용 컨텍스트 실현, 범용 컨테인먼트 트리가 있으며, COPA는 패밀리 아키텍처, 컴포넌트 프레임웍, 컴포넌트 아키텍처가 있다.

본 논문에서 핵심 자산을 비교한 결과로 미루어볼 때, 각 방법론들은 핵심 자산의 구성 요소를 대체적으로 만족시키고 있다고 본다. 그러나 FAST는 일반적인 핵심 자산의 구성요소 또는 산출물이 아닌 FAST에 특징적인 요소를 많이 언급하고 있으며, PoLITe는 위에서 언급한 것처럼 핵심 자산 구현에 치중을 하므로 핵심 자산 구성 요소에 관한 정보가 많이 누락되어 있다고 본다. PLE 방법론에 관한 프로세스, 지침, 산출물 등이 잘 정의된 KobrA는 핵심 자산의 구성 요소 메타모델 역시 잘 정의되어 있으며, 현재 발표된 PLE 방법론 중에서 Bosch의 기법이 핵심 자산의 구성요소를 가장 잘 표현하고 있음을 확인할 수 있다.

6. 결론 및 향후 연구과제

SEI SPL, Bosch의 기법을 제외한 대부분의 PLE 방법론은 핵심 자산의 정의를 명확하게 내리고 있지 않지만, 관련 도서와 논문 등을 통해서 핵심 자산의 정의를 확인한 결과 대부분의 방법론이 '재사용 가능한 자산'의 의미를 포함하고 있음을 알 수 있다. 핵심 자산의 구성요소도 마찬가지로 SEI SPL을 제외한 다른 방법론들은 정확하게 명시하고 있지 않지만, 어플리케이션 개발에 재사용 될 수 있는 핵심 자산 개발 과정의 최종 산출물을 통해서 핵심 자산의 구성 요소가 유추 가능했다.

아직까지 표준화된 PLE 방법론이 존재하지 않기 때문에, 기존에 발표된 방법론들은 구성요소의 정의나 프로세스에서 차이를 보이고 있으며, 산업계나 학계에서는 개발 방법론의 선택이나 프로세스의 부분적인 선택에 어려움을 겪고 있다. 본 논문에서는 재사용 측면을 강조하는 PLE방법론의 핵심 자산을 비교함으로써 특정 방법론을 선택할 때 본 논문의 연구 결과를 활용하여 가장 합리적인 선택을 할 수 있도록 정보를 제공하였다.

본 논문에서는 PLE 방법론의 핵심 자산 구성요소를 비교 했으나, 향후에는 각 방법론에서 제시하는 도메인 개발 과정을 비교하는 등 방법론 전반에 걸친 비교 작업을 수행하여 학계나 산업계에서 해당 프로젝트에 적합한 방법론을 선택하며 나아가서는 PLE 방법론을 표준화 시키는 부분에 기여할 수 있도록 정보를 제공할 수 있는 연구로 확대시키고자 한다.

7. 참고문헌

[1] Kyo C. Kang et. al., "Concepts and Guidelines of Feature Modeling for Product Line Software Engineering," LNCS 2319, 2002, pp. 62-77.

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

[3] Kyo C. Kang et. al., "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," Annals of Software Engineering, 5, 1998, pp. 143-168.

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

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

[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] IESE, Web Site for PoLITe Project, at URL: http://www.polite-project.de.

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

[9] 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.

[10] Bosch, J., *Design and use of software architectures,* Addison-Wesley, 2000