제품계열 공학의 실용적 어플리케이션 공학 프로세스

장치원, 장수호, 김수동
숭실대학교 대학원 컴퓨터학과
{cwchang,shchang}@otlab.ssu.ac.kr, sdkim@ssu.ac.kr

A Practical Application Engineering Process for Product Line Engineering

Chee Won Chang, Soo Ho Chang, and Soo Dong Kim
Dept. of Computer Science, Soongsil University

요 약

제품계열공학(Product Line Engineering, PLE)은 효과적인 재사용 기법으로, 핵심자산(Core Asset) 개발 프로세스와 어플리케이션 공학 프로세스로 구성된다. 핵심자산 개발 프로세스는 제품계열의 여러 어플리케이션들의 공통 휘처(Feature)들을 모델링한 핵심자산 개발에 사용된다. 어플리케이션 공학 프로세스에서 핵심자산을 인스턴스화(instantiation)하고, 핵심자산이 제공하지 않는 어플리케이션 종속적인 기능을 모델링한 후, 이 두 모델을 통합하여 목표 어플리케이션을 생성 개발한다. 현재의 제품계열공학 연구는 핵심자산 개발과 인스턴스화 과정에 집중되어 있고, 어플리케이션 공학 프로세스의 연구는 개념적 수준에 머물고 있다. 특히, 인스턴스화된 핵심자산 모델과 어플리케이션 종속 모델의 통합의 실용적 기법이 미흡하다. 본 논문에서는 어플리케이션 공학 프로세스의 주요 활동들에 대한 작업 순서와 실용적 지침을 제공한다.

1. 서론

제품계열공학은 S/W 개발을 위한 효과적인 재사용 기법으로, 핵심자산 개발 프로세스와 어플리케이션 공학 프로세스로 구성된다.

핵심자산 개발 프로세스는 제품계열의 여러 어플리케이션들의 공통 휘처들을 모델링한 핵심자산 개발에 사용된다. 어플리케이션 공학 프로세스에서 핵심자산을 인스턴스화 하고, 핵심자산이 제공하지 않는 어플리케이션 종속적인 기능을 모델링한 후, 이 두 모델을 통합하여 목표 어플리케이션을 생성 개발한다. 현재의 제품계열공학 연구는 핵심자산 개발과 인스턴스화 과정에 집중되어 있고, 어플리케이션 공학 프로세스의 연구는 개념적 수준에 머물고 있다. 특히, 인스턴트화된 핵심자산 모델과 어플리케이션 종속 모델에 대한 개발과 통합의 실용적 기법이 미흡하다.

본 논문의 2장에서는 제품계열공학의 어플리케이션 공학에서 앞선 방법론을 소개하며, 3장과 4장에서는 핵심자산에 대한 정의와 어플리케이션 공학을 위한 중요 활동을 제시한다. 마지막으로 5장에서는 결론을 제시한다.

2. 관련 연구

KobrA에서의 어플리케이션 공학은 컨텍스트 실현(Context realization) 인스턴스화, Komponent 명세(Komponent Specification) 와 Komponent 실현(Komponent Realization) 인스턴스화의 활동으로 구성 되어 있다[1]. 컨텍스트 실현에서는 컨설턴트들이 고객과 함께 생산물의 특징들을 도출하여 프래임웍 공학에서 개발된 후보 프래임웍과 고객이 요구하는 특징이 얼마나 많이 중복되는지를 확인한다. 프래임웍안에서 Komponent의 명세와 실현의 인스턴스화를 반복적으로 수행 함으로서 명세 Komponent 포함 트리(Specific Komponent Containment Tree)를 생성한다. 모델링 생성된 Komponent 포함 트리를 이용하여 하나의 특정한 어플리케이션을 생성한다. 이 연구에서는 실용성을 위해 좀더 프로세스를 정제할 필요가 있다.

PuLSE(Product Line Software Engineering)는 배포단계(Deployment), 기술적 컴포넌트(Technical Components), 지원 컴포넌트(Support Components) 3개로 구성되어 있다[2]. 특히 어플리케이션을 생성을 위한 단계로 기술컴포넌트(Technical Component)에 포함된 PuLSE-I(Instantiation)가 있다. PuLSE-I는 Product Line Model 인스턴스 와 Reference Architecture 인스턴스 두개 과정이 있다[3]. 이 연구에서는 상세한 수준에 지침과 각 활동에 대한 산출물의 정의가 부족하다.

3. 핵심자산

핵심자산은 범용 아키텍처, 컴포넌트 모델, 의사결정 모델로 구성된다. 범용 아키텍처는 여러 어플리케이션 멤버에게 사용되는 아키텍처로서, 컴포넌트들의 집합과 컴포넌트간의 관계를 view와 style를 이용하여 표현한다[4]. 의사결정모델(Decision Model)는 '가변점(Variation Point)', 연관된 '가변치(Variants)', '영향(Effect)', '작업(Task)'으로 구성된다.

4. 프로세스

본 절에서는 제품계열공학의 어플리케이션 공학 프로세스의 중요한 활동을 제시한다. 그림 1과 같이 이 프로세스는 총 4개의 활동으로 구성 된다.

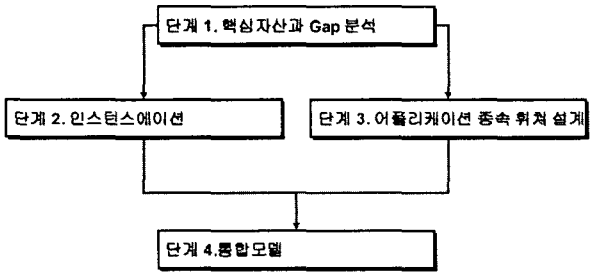

그림 1. 통합 모델을 위한 프로세스

4.1. 단계 1 핵심자산과 Gap 분석

이 단계에서는 핵심자산과 어플리케이션간에 Gap을 분석하여, 두 사이에 중복된 휘쳐와 어플리케이션 종속 휘쳐를 분류한다.

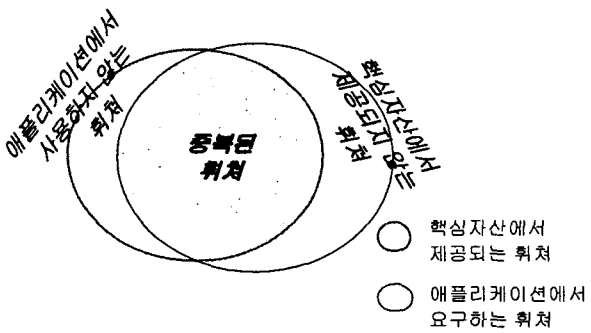

그림 2 핵심자산과 어플리케이션간의 매칭

핵심자산은 한 도메인에서 여러 어플리케이션들의 공통된 휘쳐들을 이용하여 핵심자산을 개발한다. 따라서 그림 2의 '중복된 휘쳐'와 같이 핵심자산에서 제공하는 휘쳐들은 한 도메인의 어플리케이션들의 사용하는 휘쳐들과 중복된다.

중복된 휘쳐들은 한 도메인에서 모든 어플리케이션들의 가변점과 연관된 가변치들을 가지고 있다. 따라서 이러한 휘쳐들이 한 어플리케이션에 맞도록 하기 위하여 핵심자산의 의사결정모델을 한 어플리케이션을 위한 인스턴스화 모델(Decision Resolution Model)로 생성한다. 인스턴스화 모델은 한 어플리케이션에 해당하는 가변점과 연관된 가변치를 정의 한다. 가변성의 범위는 *closed* 이거나 혹은 *open*이 될 수 있다[5]. *closed*가변성은 의사결정 모델이 가변점에 해당되는 유효한 가변치들이 이미 정의되어있는 경우이다. 그러나 *open*가변성은 가변점에 대한 가변치들이 의사결정 모델에 정의되어 있지 않은 경우 이다. 즉, 가변치가 아직 결정 되지 않은 경우이다. 범위가 *closed*가변성일 경우, 어플리케이션을 위한 가변치를 의사 결정 모델에서 선택한다. 그러나 *open*가변성일 경우, 어플리케이션을 위해 가변치를 새롭게 정의 해야 한다. 표 1은 인스턴스화 모델의 템플릿이며, '가변점'과 연관된 '가변치'에 추가적으로 가변치를 선택한 이후 나타나는 '효과'와 가변성을 인스턴스화 하기 위해 요구되는 '작업'으로 구성한다.

표 1 인스턴스화 모델

아이디	가변점	가변치	효과	작업

이상적인 핵심자산은 어플리케이션에 사용되는 모든 휘쳐들이 핵심자산에서 제공되어야 한다. 그러나 그림 2의 '핵심자산에서 제공하지 않는 휘쳐'와 같이 핵심자산은 어플리케이션에서 사용하는 모든 휘쳐들을 제공하지 않는다. 따라서 핵심자산에서 제공하지 않는 어플리케이션 종속 휘쳐 명세서를 생성한다. 어플리케이션 휘쳐 명세서에는 유즈케이스 모델을 포함한다. 이 단계의 최종 산출물은 인스턴스화 모델과 어플리케이션 종속휘쳐 명세서이다. 이 두 산출물은 단계2의 '인스턴스화'와 단계 3의 '어플리케이션 종속 휘쳐 설계의 입력물로 사용된다.

4.2. 단계 2 인스턴스화

본 단계는 단계1에서 개발된 인스턴스화 모델의 각 요소를 이용하여 핵심자산을 목표한 어플리케이션에 맞도록 가변성을 인스턴스화 한다.

범용 아키텍쳐에서는 컴포넌트와 컴포넌트간의 관계를 구성하고 있으며, 기능적 요구 사항과 비기능적 요구사항을 정의 한다. 핵심자산의 범용 아키텍쳐는 어플리케이션들 사이에 공통된 아키텍쳐적인 의사결정을 포함한다[6][7][8]. 따라서 가변성이 컴포넌트들의 집합에 존재하면 목표한 어플리케이션에 맞는 컴포넌트를 선택하며 범용 아키텍쳐에 불필요한 컴포넌트가 있다면 제거한다. 가변성이 컴포넌트간의 관계에 있다면, 목표한 어플리케이션에 맞도록 컴포넌트간의 관계를 재정의 한다.

컴포넌트모델의 가변성은 *closed*범위일 경우, 가변점에서 한 도메인의 여러 어플리케이션을 위한 가변치들이 존재 한다. 따라서 목표 어플리케이션에서 불필요한 가변치들을 제거한다. 제거된 컴포넌트는 목표한 어플리케이션에 최적화된 상태로 된다. 이 단계의 최종 산출물은 인스턴스화된 핵심자산 이다. 이 산출물은 단계 4 '모델통합'의 입력자료로 사용된다.

4.3. 단계 3 어플리케이션 종속 휘쳐 설계

이 단계에서는 핵심자산에서 제공되지 않는 어플리케이션 종속 휘쳐를 COTS 컴포넌트에서 구매하거나 설

계 한다.

어플리케이션 종속 휘쳐 명세서를 확인하여 어플리케이션 종속 휘쳐들을 구현한 COTS 컴포넌트로 존재하는지 확인한다.

COTS 컴포넌트의 구매 비용과 어플리케이션 종속 휘쳐를 설계하는 비용을 비교하여 COTS의 구매 여부를 결정한다. 어플리케이션 종속 휘쳐들이 COTS 컴포넌트에 존재하지 않으면, 어플리케이션 종속 휘쳐들은 단계 1의 산출물인 어플리케이션 종속 휘쳐 명세서를 이용하여 클래스 모델과 시퀀스 모델로 설계하며 이 들을 사용하여 어플리케이션 종속 컴포넌트를 생산한다. 단계 4에서는 어플리케이션 종속 컴포넌트는 단계 4의 입력물로 사용한다.

4.4. 단계 4 모델 통합

이 단계는 단계2와 단계3의 산출물인 '특화된 핵심자산'과 '어플리케이션 종속 컴포넌트'를 통합하는 단계이다. 핵심자산의 단위는 컴포넌트이며 어플리케이션 종속 휘쳐들의 산출물 또한 컴포넌트이다.

새로운 어플리케이션 종속 휘쳐 설계에 의해 개발된 모델의 통합 일 경우 두 컴포넌트 간의 결합은 어플리케이션 종속 휘쳐 명세서를 이용하여 인터페이스간에 결합이 이루어진다. 구매된 컴포넌트와의 결합 일 경우, 두 컴포넌트간의 인터페이스 충돌이 발생할 수 있으며, 이로 인하여 부적합한 영향이 발생 할 수 있다. 그러므로 인터페이스간의 매핑 작업이 필요하다.

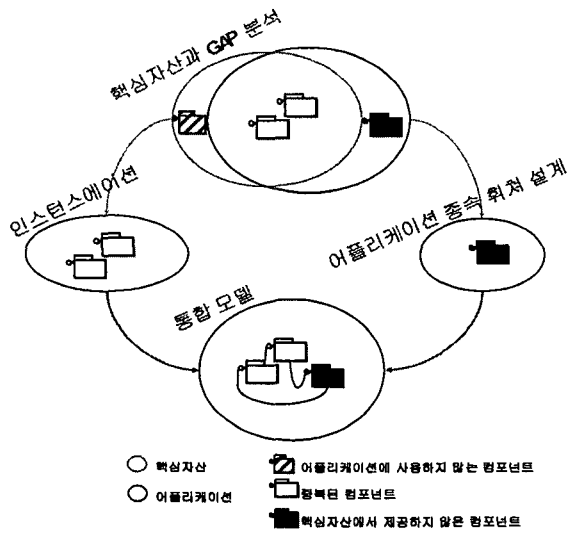

그림 3 두 컴포넌트간의 통합 표현

어플리케이션 종속 휘쳐 명세서는 단계 1의 어플리케이션 종속 유즈케이스를 포함하고 있으며, 또한 이 명세서는 단계 3을 수행하는 동안 모델 통합을 위한 통합점을 명세한다. 그림 3은 핵심자산에서 인스턴스화된 컴포넌트와 어플리케인션 종속 컴포넌트들이 인터페이스를 이용하여 통합하는 과정을 표현한다.

5. 결론

제품계열공학은 핵심자산 개발 프로세스와 어플리케션 개발 프로세스 구성된다. 핵심자산 개발 프로세스는 제품계열의 여러 어플리케이션들의 공통 휘쳐들을 모델링 한 핵심자산 개발에 사용된다. 어플리케이션 개발 프로세스는 핵심자산의 요소를 한가지 특정 어플리케이션 맞도록 인스턴스화된 핵심자산을 개발하며, 핵심자산에서 제공하지 않는 어플리케이션에 종속된 휘쳐를 설계 혹은 구매하고 통합하여 어플리케이션을 개발한다.

본 논문에서는 어플리케이션 개발을 위한 중요한 활동에 대하여 실용적인 활동과 지침을 제공 하였으며 각 활동에 실용적인 산출물을 제시 하였다. 특히, 인스턴스화된 핵심자산과 어플리케이션 종속 컴포넌트간의 통합을 위한 구체적인 산출물과 산출물의 요소를 제시 하였다.

본 논문에서 제시한 활동과 지침은 제품계열공학을 이용한 어플리케이션 개발을 좀더 실용적으로 접근할 수 있으며, 비용과 시간을 효율적을 사용할 수 있다.

6. 참고문헌

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

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

[3] Bayer, J., Gacek, C., Muthig, D., and Widen, T., "PuLSE-I: Deriving Instances from a Product Line Infrastructure," *Proceeding of 7th International Conference and Workshop on the Engineering of Computer Based Systems, IEEE*, 2000.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.