제품계열 공학의 핵심자산과 어플리케이션 간의 Gap 분석 기법

오상헌, 김수동, 류성열
숭실대학교 대학원 컴퓨터학과
zippozero@selab.ssu.ac.kr, {sdkim,syrhew}@comp.ssu.ac.kr

A Technique for Analyzing the Gap between in Product Line Engineering Core Asset and Applications

Sang Hun Oh, Soo Dong Kim, SungYul Rhew
Dept. of Computer Science, Soongsil University

요 약

PLE 방법론은 단일 제품이 아니라 유사한 제품들간의 공통성(Commonality)과 가변성(Variability)을 개발하고 관리하며 소프트웨어 개발 전체 생명주기에 걸쳐 부품을 조립하는 형태로 만들어 진다. 또한 PLE 방법론은 재사용 단위가 가장 큰 방법론이기 때문에 최근에는 소프트웨어 업계에서 주목을 많이 받고 있다. 따라서 소프트웨어 재사용 분야가 점점 다양화되면서 어플리케이션의 특성에 적합한 프로세스에 대한 요구가 늘어나고 있다. 어플리케이션 과정은 요구사항 정의에 따라서 설계가 되어야 하고 이렇게 설계가 된 요구사항 정의와 핵심자산의 Gap 분석을 통해 정제된 설계를 얻을 수 있다. 하지만, 현재는 체계적인 절차와 기법에 대한 연구가 많이 미흡한 상태이다. 이렇게 체계적인 절차와 기법이 있다면 어플리케이션을 개발하는데 있어 보다 효율적이고, 보다 완성도 높은 어플리케이션이 개발 될 것이라고 기대한다. 따라서 본 논문에서는 제품계열공학의 핵심자산과 어플리케이션 간의 Gap 분석 절차를 제안 하고자 한다.

Keyword: 어플리케이션, 핵심자산, 제품계열공학

1. 서론

PLE(Product Line Engineering)방법론은 단일 제품이 아니라 유사한 제품들간의 공통성과 가변성을 개발하고 관리하며 소프트웨어 개발 전체 생명주기에 걸쳐 부품을 조립하여 어플리케이션을 만들 수 있는 메커니즘을 제공한다. PLE에서 말하는 핵심자산이란, PLE의 재사용 단위로써, 핵심자산에는 일반적으로 아키텍처, 컴포넌트, 의사결정모델 등이 있다. 또한 핵심자산은 제품라인에서 제품을 생산하기 위한 기반이 된다[1].

제품계열 공학은 Framework Engineering(FE) 부분과 Application Engineering(AE) 부분으로 나눌 있다. FE과정은 실제 도메인을 분석하는 과정이고, AE과정은 어플리케이션 요구사항을 분석하는 과정이라고 할 수 있다. FE에 대한 연구는 많이 되어 있지만, 현재 AE과정에 대한 연구가 많이 부족한 실정이므로, 본 논문에서는 어플리케이션에 대한 요구사항 분석을 통해 핵심자산과 AE간의 Gap 분석 기법을 제안 한다.

본 논문의 구성은 2장에서 어플리케이션과 관련된 관련연구를 살펴보고, 3장에서는 실제 Gap 분석 프로세스를 소개한다. 4장에서는 3장에서 소개한 프로세스의 활동별 지침을 살펴본다. 5장에서는 Gap에 대한 명세를 한다.

2. 관련 연구

KobrA[2] 방법론은 소프트웨어 재사용 문제의 해결책으로써 제시된 주요 기술인 컴포넌트 기반 개발 방법론과 아키텍처 스타일 및 디자인 패턴과 제품 계열 공학의 기술들을 통합된 방법론이다. KobrA의 제품 계열 공학 프로세스는 프레임워크 공학 프로세스와 어플리케이션 공학 프로세스로 구성 되어 있다.

FORM[3] 방법론의 핵심은 제품라인을 휘처 중심으로 분석하고 이러한 휘처를 이용하여 재사용 가능하고 변화에 적응시킬 수 있는 제품 라인 자산을 개발하는데 있다. FORM은 소프트웨어 아키텍처 설계와 객체지향 컴포넌트 개발을 지원하며, 다양한 산업계 제품 라인에 적용되어 제품 라인 소프트웨어 개발 환경과 소프트웨어 자산을 개발하는데 적용되었다. 이러한 FODA도 문제점은 있다. 쉽게 적용할 수 있는 상세한 지침이 부족하고, 따라서 상세한 설명이 없기 때문에 많은 조직들이 도메인 공학 방법론을 자주 정의하게 될 수 밖에 없다.

PuLSE[4] 방법론은 독일 IESE에서 만들어진 제품 라인 공학이다. 이 방법론은 각 단계에서 제품 라인을 위해서 필요한 기술적 노하우를 포함하고 있는 기술 컴포넌트를 제공하고, 각각의 환경에 따라서 커스터마이징이 가능하다. 변화가 자주 발생하는 전사적 업무에 맞추어

소프트웨어 제품 계열의 신속한 계획과 배포를 가능하게 하기 위한 목적으로 개발 되었다. 하지만 PuLSE 방법론은 다른 제품 라인 접근법들과 마찬가지로 패밀리를 매우 높은 추상화 수준에서 설명을 하고, 구현에 대한 정확한 지침이 부족하다. 따라서 컴포넌트의 역할을 명확히 고려하고 있지 못하다.

3. 전체 프로세스

그림 1은 본 논문에서 제안하는 Gap 분석의 전체 프로세스를 보여준다. 이 프로세스는 3개의 활동으로 구성된다.

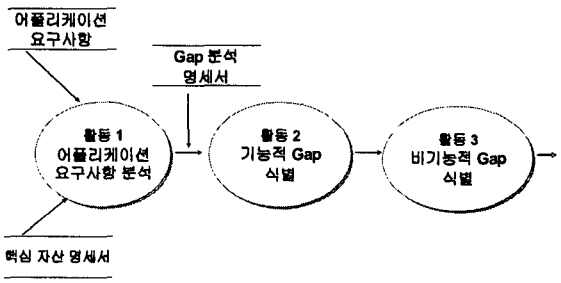

그림 1. Gap 분석 프로세스

활동 1 단계에서는 어플리케이션 요구사항과 핵심자산을 입력물로 받아서, 어플리케이션의 요구사항을 분석하게 된다. 이렇게 해서 나온 어플리케이션의 요구사항 분석과 Gap 분석 명세서를 입력물로 받게 된다.

본 활동은 전체 프로세스애 도입부분 이다. 실제 어플리케이션과 핵심자산 사이에 Gap을 분석하기 위해 어플리케이션의 요구사항을 분석하게 된다.

활동 2 단계에서는 기능적 Gap을 식별한다.

본 활동은 활동 1의 산출물인 어플리케이션 요구사항 분석과 Gap 분석 명세서를 가지고 실제 기능적 Gap을 식별하게 된다.

본 활동에 들어가기 앞서서, Gap 분석 명세서를 작성해야 하는데, 어플리케이션 안에 핵심자산의 기능이 완전히 포함된 경우와, 어플리케이션 안에 핵심자산이 일부분만 포함된 경우를 알아야 한다.

기능적 Gap 식별에서는 요구사항 분석은 휘처 단위로 분석하게 된다. 따라서 어플리케이션의 휘처와 핵심자산의 휘처와 크기를 비교해야 한다. 만약 어플리케이션의 휘처가 핵심자산의 휘처보다 클 경우 어플리케이션의 휘처를 핵심자산의 휘처 크기에 맞춘다. 그렇치 않을 경우에는 Scope외에 어플리케이션은 핵심자산의 크기와 비슷한 크기로 설계를 하고, 비교를 한다.

활동 3단계는 비기능적 Gap 식별을 하게 된다.

비기능적 Gap이란 어플리케이션과 핵심자산에서 나타난 휘처들에 대한 품질요구 사항(Extra Functional)을 말한다. 본 활동에서는 ISO9126에 있는 소프트웨어 제품 품질을 바탕으로 어플리케이션과 핵심자산에 대한 비기능적 Gap에 대한 식별을 한다. 소프트웨어 제품 품질은 품질 요소들로 나누는데, 그 밑에 세부단계로는 품질 항목들로 이루어져 있다. 따라서 비 기능적인 Gap을 식별할 수가 있다. 다음은 비 기능적인 품질 요소 및 품질 항목이다.

- 기능성 [적합성, 정확성, 보안성, 준수성]
- 이식성 [적응성, 설치성, 대체성, 공존성, 준수성]
- 사용성 [이해성, 학습성, 운용성, 친밀성]
- 유지보수성 [분석성, 변경성, 안정성, 시험성]

마지막 활동 3이 끝나면 어플리케이션 설계와 핵심 자산의 통합 등 후속 개발 작업이 진행되는데, 본 논문에서는 활동 3까지의 활동만 다루고 있다.

4. 활동별 지침

4.1. 어플리케이션 요구사항 분석

어플리케이션의 요구사항 분석은 실제 핵심자산과의 공통성 분석을 위해서 사용된다. 요구사항 분석은 휘처 단위로 분석되며, 핵심자산과 어플리케이션간에 Gap 얼마나 되는지를 알 수 있다. Gap 분석 프로세스 입력물로는 어플리케이션 요구사항과 분석된 핵심자산이 입력물이 된다.

4.2. 기능적 Gap 식별

그림 2는 핵심자산과 어플리케이션의 포함관계를 나타낸 그림이다. 그림에서 보는 바와 같이 어플리케이션 안에 핵심자산이 모두 포함되는 경우와 어플리케이션 안에 핵심자산이 일부만 포함된 경우가 있다.

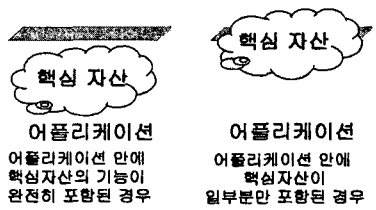

그림 2 핵심자산과 어플리케이션간의 포함관계

그림 3 기능적 Gap 식별 에서 기능적 Gap 식별은 위에서 분석된 어플리케이션의 요구사항 분석에서 요구사항 분석은 휘처단위로 분석이 되기 때문에, 분석에 작은 단위는 휘처단위가 된다. 어플리케이션의 휘처는 AF로 핵심자산의 휘처는 CF로 표기한다.

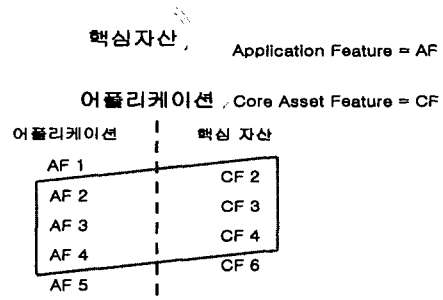

그림 3 기능적 Gap 식별

어플리케이션이 기존 Scope내에 해당 되는 경우에는 이미 기존에 분석했던 결과를 바탕으로 AF와 CF의 휘처 크기를 비교한다. 하지만, Scope외에 어플리케이션은 핵심자산의 크기와 비슷한 크기로 설계를 하고, 비교를 한다. 따라서, 각 휘처당 사용하는 데이터와 로직을 이용해서 Gap를 알 수 있다. 만약 어플리케이션의 휘처가 핵심자산의 휘처보다 클 경우 어플리케이션의 휘처를 핵심자산의 휘처 크기에 맞춘다.

4.3. 비기능적 Gap 식별

비기능적 Gap이란 어플리케이션과 핵심자산에서 나타난 휘처들에 대한 품질요구 사항(Extra Functional)을 말한다. 즉, 기능성, 이식성, 사용성, 유지보수성 등이 품질요구사항이 된다. 그림 4는 ISO 9126에 있는 소프트웨어 제품 품질을 바탕으로 어플리케이션과 핵심자산에 대한 비기능적 Gap에 대한 식별을 나타내었다. 소프트웨어 제품 품질은 품질요소들로 나누는데, 그 밑에 세부단계로는 품질 항목들로 이루어져 있다[5].

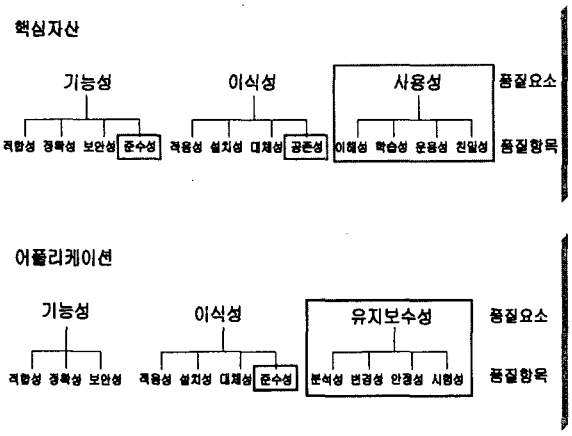

그림 4 비기능적 Gap 식별

그림 4 에 나타나 있는 어플리케이션과 핵심자산에 내용을 살펴보면 크게 품질요소로는 사용성과 유지보수성이 일치하지가 않는다. 세부항목인 품질항목을 보면 기능성에서는 준수성, 이식성에서는 준수성이 새로이 추가 되었다. 그림 4 를 바탕으로 하여 비기능적인 Gap 을 식별할 수가 있었다..

5. Gap 분석 명세

본 절에서는 Gap 분석 프로세스의 작업을 수행한 후, 이를 명세하는 방법을 제안한다.

전체 프로세스는 어플리케이션의 요구사항과 핵심자산의 명세서를 입력물로 받아 이 입력물로 통해 어플리케이션요구사항 분석을 한다. 이렇게 분석된 요구사항을 Gap분석 명세서와 함께 다음 활동인 기능적 Gap을 식별하고, 비기능적 Gap 식별을 한다.

표 1 Gap 분석 명세서

	기능적	비기능적
핵심자산에 추가적으로 있는 요구사항		
핵심자산에서 부족한 요구사항		
핵심자산이 제공하는 기능이 충분한 요구사항		
핵심자산이 제공하지만 커스터마이징 해야하는 요구사항		

6. 결론

PLE(Product Line Engineering)은 단일 제품이 아닌 유사한 제품들간의 공통성과 가변성을 개발하고 관리하며 소프트웨어 개발 전체 생명주기에 걸쳐 부품을 조립하여 어플리케이션을 만들 수 있는 메커니즘을 제공한다. 본 논문에서는 어플리케이션 과정에 대한 요구사항 정의와, 핵심자산의 Gap분석을 통해서 정제된 설계를 얻고자, 제품계열 공학의 핵심자산과 어플리케이션간의 Gap분석 기법을 제안하였다.

참고문헌

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

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

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

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

[5] ISO/IEC 9126, International Standard Information Technology - Software Product Evaluation – Quality characteristic and guidelines for their use, 1991.