기존 시스템 기반의 소프트웨어 제품라인 공학 기법: 케이블 셋톱박스 소프트웨어 사례

(Legacy System-Based Software Product Line Engineering: A Case Study on Cable Set-Top Box Software)

최 현 식 † 이 혜 선 † 조 윤 호 †† 강 교 철 †††
(Hyunsik Choi) (Hyesun Lee) (YoonHo Cho) (Kyo Chul Kang)

요 약 산업 및 가전 제품에서 소프트웨어의 중요성이 커지면서 소프트웨어 재사용과 제품라인 방법론에 대한 관심이 높아지고 있다. 그러나 대부분의 제품라인 방법론은 초기 비용과 시간이 많이 들고 구체적인 절차나 성공 사례가 부족하여 산업체에서 적용하기에는 어려운 한계가 있다. 본 논문에서는 산업체에서 제품라인 기법을 쉽게 적용할 수 있도록, 기존에 개발된 소프트웨어 자산과 해당 도메인의 휘처모델을 활용하여 제품라인을 쉽게 구축할 수 있는 추출식 접근법의 구체적인 방법을 제시하고 이를 케이블 셋톱박스 소프트웨어에 적용한 사례를 소개한다. 또한, 제품라인으로의 전환 효과를 확인할 수 있는 평가 기준을 제안하고, 사례 연구를 통해 얻은 교훈을 정리하여 다른 산업 및 가전 제품의 소프트웨어에 적용할 수 있는 지침을 제공한다.

키워드 : 소프트웨어 제품라인, 휘처모델, 도메인, 추출식, 셋톱박스

Abstract Software product line (SPL) engineering is an emerging paradigm for successful software reuse and has been adopted for various industrial and consumer products to improve their productivity and quality. However, most SPL methods require high initial costs and long development time, which makes many companies hesitate to adopt the SPL paradigm. In this paper we introduce a method to construct an SPL by extracting core assets from legacy components based on the feature model, which requires less initial time and effort. We also present a case study on cable set-top box software to illustrate the applicability of this method, and lessons learned that will provide guidelines for many companies to adopt the SPL paradigm.

Key words : software product line, feature model, domain, extractive approach, set-top box

· 이 논문은 2009 한국소프트웨어공학학술대회에서 '기존 시스템 기반의 소프트웨어 제품라인 공학 기법: 케이블 셋톱박스 소프트웨어 사례'의 제목으로 발표된 논문을 확장한 것임
· 본 과제는 한국소프트웨어진흥원의 SW공학 요소기술 개발과 전문인력 양성사업의 결과물임을 밝힙니다.

† 학생회원 : 포항공과대학교 컴퓨터공학과
nllbut@postech.ac.kr
compial@postech.ac.kr
†† 학생회원 : 포항공과대학교 정보통신대학원
lucio.red@postech.ac.kr
††† 종신회원 : 포항공과대학교 컴퓨터공학과 교수
kck@postech.ac.kr

논문접수 : 2009년 3월 16일
심사완료 : 2009년 5월 26일

정보과학회논문지: 소프트웨어 및 응용 제36권 제7호(2009.7)

1. 서 론

소프트웨어가 산업 및 가전 제품에서 차지하는 중요성과 그 규모가 커지면서 체계적인 소프트웨어 개발과 유지보수에 대한 필요성도 커졌으며, 특히 비슷한 소프트웨어를 효율적으로 개발하기 위한 소프트웨어 재사용에 대한 관심이 높아지고 있다. 또한, 하드웨어의 다양성 및 잦은 환경 변화에 대처할 수 있는 기술에 대한 요구도 증가하고 있다.

소프트웨어 제품라인(product line) 공학은 최근 가장 주목 받는 재사용 기법 중 하나로, 단일 소프트웨어가 아닌, 비슷한 소프트웨어 제품들이 모인 도메인(domain)을 분석하여 도메인 내 소프트웨어 제품들 사이의 공통점과 차이점을 휘처(feature) 관점에서 찾아내고 [1,2], 이를 기반으로 핵심 자산(core asset)을 개발하여 비슷한 소프트웨어 제품을 효과적으로 생성할 수 있도

록 한다. 이러한 소프트웨어 제품라인 공학 기법을 적용하여 및 생산성 및 품질 향상, 시장 출시일 단축 등의 효과를 얻은 사례가 전 세계적으로 다양하게 발표되고 있다[3-5].

제품라인 공학의 접근법으로는 선행식(proactive)과, 반응식(reactive), 추출식(extractive)이 있다. 선행식 접근법은 제품라인의 요구사항과 예상되는 변화를 초기에 모두 분석한 후 휘처모델 및 핵심 자산을 구축하고 이를 통해 개별 제품을 생성하는 방식으로, 변화가 적고 안정된 제품라인에서는 적용 효과가 크지만 초기에 많은 시간과 인력이 필요하여 산업체에서 수용하기가 쉽지 않다. 반면 반응식 접근법은 점진적(incremental) 개발 방식으로, 단일 소프트웨어의 나선형(spiral) 개발 모델과 같이 한 순환 시 제품라인의 일부 요구사항을 수용한 제품라인 모델을 구축한 후, 새로운 제품의 요구사항이 추가되면 제품라인 모델을 확장함으로써 변화 예측이 어려운 제품라인에 적용할 수 있으며, 특히 제품간 아키텍처 변화가 적을 때 효과가 있다[6].

선행식 및 반응식 접근법과 달리 추출식 접근법은 기존 소프트웨어 제품들을 재사용하여 초기 제품라인 모델을 구축함으로써, 기존 개발 방식에서 제품라인 방식으로 빠르게 전환할 수 있도록 제안되었다[7]. 그러나, 추출식 접근법을 실행하기 위한 구체적인 방법이나 성공 사례가 부족하여 일반 산업체에서 적용하기에는 한계가 있고, 적용 효과 또한 검증되지 못하고 있다. 이에, 본 논문에서는 산업체에서 제품라인 기법을 쉽게 적용할 수 있도록, 기존 케이블 셋톱박스 소프트웨어를 역공학(reverse engineering) 과정을 통해 제품라인 체계로 전환한 사례를 통해, 추출식 접근법의 구체적인 실행 방법과 그 적용 효과를 서술하고자 한다.

본 논문의 구성은 다음과 같다: 2장에서는 사례 연구로 수행한 케이블 셋톱박스 소프트웨어 도메인의 특성을 살펴보고, 이 특성을 바탕으로 사례 연구에 적용한 기본 공학 원칙을 3장에 정리한다. 4장에서 실제로 이 공학 원칙에 따라 기존 셋톱박스 소프트웨어를 분석하고 역공학 및 추출 과정을 거쳐 효율적으로 제품라인을 구축한 과정을 설명하고, 구축된 제품라인의 유효성을 검증하기 위한 시제품 생성 결과를 소개 한다. 5장에서는 사례 연구를 통해 얻은 교훈을 정리하며, 6장에서 향후 연구 방향과 함께 결론을 맺는다.

2. 케이블 셋톱박스 도메인(Domain) 특성

케이블 셋톱박스 도메인을 분석하기 위해, 국내 셋톱박스 미들웨어 시장에서 점유율이 가장 높은 A사 소프트웨어를 대상으로, 담당 개발자들과의 인터뷰 및 기존에 개발된 셋톱박스 제품들간의 비교를 진행하였다.

디지털 케이블 셋톱박스 소프트웨어는 그림 1과 같이 가장 하단의 하드웨어를 제어하는 시스템 소프트웨어 계층과, 중간의 미들웨어 계층, 최상위의 EPG를 포함한 응용 프로그램 계층으로 구분되는데, 이 장에서는 상위 두 소프트웨어 계층의 특성을 살펴보고, 이 계층의 도메인 및 기존 자산을 분석하는 과정 중 겪은 문제점을 서술한다.

2.1 미들웨어 도메인

미들웨어는 하위의 다양한 하드웨어 및 시스템의 변화에 적응하여, 어떤 환경에서도 상위의 응용 프로그램들이 정상적으로 동작할 수 있도록 지원하는 역할을 수행한다. 이를 위해, 미들웨어는 다양한 종류의 하드웨어(즉, 서로 다른 CPU, 메모리, 디코더, 튜너, 리모트 콘트롤러 등) 상에서 동작할 수 있도록 하드웨어 특성을 은닉화(hiding)하는 부분과, 모든 하드웨어 변화(variation)에 따라 다르게 조합되는 컴포넌트들로 구성되어 있다.

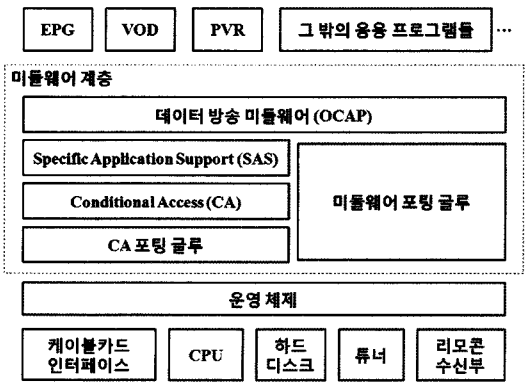


그림 1 케이블 셋톱박스 소프트웨어 개념도

A사의 미들웨어는 단일 팀이 오랜 기간 동안 다양한 하드웨어 변이 및 응용 프로그램 요구 사항에 맞게 개발하여, 구성 컴포넌트들의 품질이 우수하고 요구사항의 변화도 적어서 도메인이 안정되어 있었다. 그러나, 하드웨어 변경에 따라 구성 컴포넌트들을 조합하는 과정이 복잡하고, 조합 규칙을 일부 특정 개발자만 알고 있는 문제가 있었으며, 이를 해결하기 위해 자동화 도구를 이용해 조합 규칙을 설정하고 검증하기 위한 플랫폼을 구축하는 과정에 있었다.

미들웨어 도메인을 분석하는 과정에서 발견한 문제는 다음과 같다:

① 도메인 지식의 산재: 미들웨어 제품은 원시 코드 크기가 방대하고 지원하는 하드웨어 특성이 다양하여, 미들웨어 도메인의 세부 지식을 모두 알고 있는 전문가는 없었으며, 세부 분야별로 전문가가 나뉘어 있

었다. 따라서, 분산되어 있는 도메인 지식을 통합하기 위해서는 여러 전문가들이 동시에 모여서 함께 도메인 분석 과정을 수행해야 하는데, 실무에 바쁜 전문가들이 함께 모이기는 거의 불가능하였다.

② 공통점 위주의 도메인 특성 도출: 미들웨어 전문가들은 미들웨어 제품들 사이의 차이점 보다는 공통점 위주로 도메인 지식을 갖고 있었다. 이로 인해, 휘처 모델링 시 optional/alternative 휘처 보다는 주로 mandatory 휘처 중심으로 도메인 특성을 도출하는 한계를 보여, 제품라인 내의 변화지점(variation point) 및 변이(variant) 파악이 어려웠다.

2.2 EPG 도메인

EPG(Electric Program Guide)는 케이블 셋톱박스의 가장 중요한 응용 프로그램으로, 사용자에게 시청 가능한 채널 및 프로그램 목록을 보여 주고, 채널을 선택하여 시청하는 기능과, (미래의) 특정 프로그램을 선택하여 시청 예약 및 녹화 예약을 할 수 있는 기능 등을 제공한다.

A사의 EPG는 국내 주요 방송 사업자별로 각각 다른 팀이 전담하여 개발하였는데, 이는 방송 사업자별로 특화된 화면이나 서비스가 요구되었고, 또한 시장 출시일을 맞추기 위해 여러 EPG 제품들을 병렬적으로 동시에 개발해야 했기 때문이다. 이렇게 EPG 도메인은 새로운 서비스가 계속 추가되고 요구사항의 변화도 많은 성장하는 도메인이었다.

EPG 도메인에 대한 분석을 진행하며 접한 문제는 다음과 같다:

① 도메인 지식의 체계화 부족: 방송 사업자별로 EPG 제품을 개발하는 팀들이 분리되어 독립적으로 개발하다 보니, 비슷한 EPG 제품들을 개발하면서도 도메인 지식이 서로 다른 체계로 관리되고, 공유되지 않았으며, 개발 팀별로 이해하는 정도에도 차이가 났다.

② 자산의 공유 부재: 비슷한 EPG 제품을 동시에 여러 개를 개발하고 있음에도, 개발 팀 간의 정보 공유나 코드 및 설계 지식의 공유가 없이 개별 방송 사업자의 요구 사항에만 맞추다 보니, 자산의 공유가 없이 중복된 기능을 팀별로 다른 방식으로 개발했고, 이로 인해 이들을 포괄하는 하나의 제품라인 체계를 구축하기 어려운 문제가 있었다.

이렇게 미들웨어 도메인과 EPG 도메인에서 발견된 문제들을 해결하기 위해 본 사례 연구에서 적용한 공학 원칙을 3장에서 서술하고, 이 원칙에 따라 앞서 제시한 문제들을 해결하는 과정을 4장에서 설명한다.

3. 공학 원칙(Engineering Principles)

도메인 특성에 따른 문제들을 해결하고, 효과적인 도메인 분석과 자산 개발을 수행하기 위해 다음과 같은 공학 원칙을 적용하였다.

3.1 도메인 분리(Separation of Domains)

셋톱박스 소프트웨어 도메인을 미들웨어 하위 도메인(sub-domain)과 EPG 하위 도메인으로 분리하여 서로 다른 도메인 분석 방식 및 자산 개발 방식을 적용하였다. 이것은 두 하위 도메인 사이의 경계가 분명하고, 자산도 분리되어 있으며, 도메인 사이의 의존 관계가 많지 않기 때문에 가능하였다.

미들웨어 도메인에서는 휘처들 사이의 복잡한 조합 관계를 파악하여 체계화 하는데 초점을 맞추었고, EPG 도메인에서는 방송 사업자별로 서로 다르게 개발한 자산을 비교 및 통합하여 일원화하는데 초점을 맞추었다.

3.2 하향식(Top-Down) 접근법과 상향식(Bottom-Up) 접근법 접목

도메인 분석 과정 중, 미들웨어 전문가들이 제품간 차이점을 제대로 찾지 못하고 공통점 중심으로 분석한 것은 하향식 접근법의 한계인데, 여기에 기존 개발된 제품들 사이의 비교를 통한 상향식 접근법을 접목하여 하향식 접근법의 한계를 보완하였다.

또한, 산재되어 있는 도메인 지식을 통합하기 위해 미들웨어 및 EPG 도메인에 대한 전반적인 지식을 하향식 접근법을 통해 초기 휘처 모델로 구성한 후, 하위 세부 휘처들에 대해서는 분야별 전문가들의 지식을 정리하여 휘처 모델을 정제(refine)하는 상향식 접근법을 접목하였다.

3.3 평가 지표(Metric) 적용

기존에 개발된 미들웨어 및 EPG 도메인의 자산을 최대한 활용하기 위해, 이들 자산을 평가하는 지표를 규정하여 적용하였다. 즉, 비슷한 자산들 중 품질이 우수한 것을 선택하기 위해 소스 코드의 응집도, 복잡도, 이해도를 나타내는 지표를 규정하였고, 제품들 사이의 공통점과 차이점을 파악하기 위해 역공학 기법으로 추출한 클래스 모델 간의 유사성 정도를 측정하는 지표를 적용하였다.

이러한 평가 기준은 자동화 도구를 통해 쉽게 측정하고 계산할 수 있는 항목으로 규정하였다. 이렇게 함으로써 추출식 접근법을 적용할 때 자동화 도구를 활용할 수 있는 바탕을 마련하고, 이를 통해 빠른 자산 평가를 가능하도록 하였다.

4. 셋톱박스 소프트웨어 제품라인 구축

3장의 공학 원칙을 바탕으로 A사의 케이블 셋톱박스 소프트웨어 중 미들웨어와 EPG 제품에 대해, 역공학 기법과 추출식 접근법을 통해 제품라인 체계로 전환한 과정을 아래에 상세히 설명한다.

4.1 도메인 분석 및 휘처모델 구성

미들웨어 도메인과 EPG 도메인을 분리하여(3.1절 원칙) 도메인 분석을 진행한 후 휘처 모델을 작성하였다. 이 과정 중 산재되어 있는 도메인 지식을 통합하기 위해 미들웨어 및 EPG 도메인에 대한 일반적인 지식을 가진 전문가가 하향식 접근법으로 초기 휘처모델을 구성한 후, 하위 세부 휘처들에 대해서는 분야별 전문가와 개별 인터뷰를 통해 휘처모델을 정제하는 상향식 접근법을 접목하여(3.2절 원칙), 실무에 바쁜 도메인별 전문가들이 모두 모이지 않고도 단계적인 세부 휘처모델 구성을 통해 전체 도메인에 대한 휘처모델을 완성하도록 하였다.

도메인 전문가들이 하향식으로 휘처모델을 작성하면서 겪는 문제로, 제품간 차이점을 쉽게 찾지 못하고 공통점 중심으로 분석한 것을 보완하기 위해, 기존에 개발된 제품들이 제공하는 휘처들을 비교하여(3.2절 원칙) 모든 제품에 들어가는 것은 공통(mandatory) 휘처로, 제품에 따라 서로 배타적인 것은 택일(alternative) 휘처로, 특정 제품에만 포함되는 것은 선택(optional) 휘처로 분류하여 휘처모델을 정제하였다. 특히, 기존 제품들을 비교할 때 현재 시장에 출시된 제품들 사이의 공간적 비교뿐만 아니라 과거 버전들과 현재 버전 사이의 시간적 비교도 수행 하였다.

미들웨어 도메인의 제품간 차이점은 주로 방송을 녹화하는 PDR(personal digital recorder) 기능 및 동시에 여러 채널의 화면을 시청하는 PIP(picture in picture) 기능의 지원 여부와, 방송 사업자별 CAS(conditional access system)의 차이 및 다양한 하드웨어 변경에 따른 컴포넌트 조합 규칙과 관련되어 있었다. 이러한 도메인 분석을 통해, 기존에 개발된 미들웨어 컴포넌트들 사이의 복잡한 조합 규칙을, 컴포넌트에 대응하는 휘처들 사이의 조합 규칙으로 표현할 수 있었다. 즉, 특정 휘처가 선택되었을 때 반드시 따라서 선택되어야 할 휘처 관계('require')와, 한 제품에 동시에 들어갈 수 없는 휘처 관계('exclude')를 명시함으로써 복잡한 자산 조합 규칙을 휘처모델로 추상화하여 나타내고 자동화 도구를 통해 검토할 수 있도록 하였다.

EPG 도메인의 제품간 차이점은 주로 PDR을 지원하기 위한 기능의 유무와, 방송 사업자별로 사용하는 CAS가 다르기 때문에 나타났다. 따라서 EPG 도메인에 대해서는, 방송 사업자별로 다른 체계로 관리되던 도메인 지식을 통합하고 조율하여 공통된 도메인 사전(domain dictionary)을 구축한 후 이를 바탕으로 휘처모델을 구성 하였다.

도메인 사전은 휘처모델을 구성하기 위한 정보 외에, 독립된 개발로 공통점을 찾기 힘들었던 EPG 제품들 사이의 공통점을 파악하는 기준으로도 사용되었다. 즉, 각 방송 사업자별로 이름이나 구성이 달랐던 도메인 지식 요소들을, 도메인 사전에 통합 및 조율한 후, 도메인 사전 항목과 하나씩 대응시킴으로써, EPG 제품들끼리 직접 비교하지 않고 도메인 사전에 대응된 항목들을 통해 간접적으로 비교할 수 있었다. 이를 통해, 독립적으로 개발된 EPG 제품들 사이의 비교가 쉬워졌고, 이러한 비교 방식을 자산 구축을 위한 기반 코드를 선정하는데 이용할 수 있었다. 기반 코드 선정 과정은 4.2절에서 자세히 설명한다.

4.2 추출식 제품라인 구축

기존 미들웨어 제품 코드는 다양한 하드웨어 및 응용 프로그램 사이에서 동작할 수 있도록 개별 컴포넌트가 잘 개발되어 있어서, 이 컴포넌트 코드와 휘처모델의 휘처 조합 규칙을 이용해 초기 제품라인 모델을 구축하였다.

반면 기존 EPG 제품 코드는 방송 사업자별로 독립적으로 개발되고 개발팀 사이의 코드 공유가 없었기 때문에 여러 EPG 코드를 비교하여 공통점과 차이점을 도출하기가 쉽지 않았다. 이것은 동일한 도메인 지식을 다른 이름이나 구조로 코드에 반영하였기 때문인데, 이것을 해결하기 위해 도메인 사전을 바탕으로 통합하여 구성한EPG 휘처 모델과 도메인 사전(4.1절 참조)을 이용하여, 코드 내 각 클래스의 역할을 비교함으로써 EPG 제품 코드 사이의 공통점과 차이점을 파악할 수 있었다.

세가지 EPG 제품 코드 중 제품라인 모델을 구축하는 기반 코드를 선정하기 위해서 공통점과 차이점 분석뿐만 아니라, 코드 자체에 대한 품질도 평가하였다. 코드 품질 평가 기준은, 응집도가 높고 복잡도는 낮으며 이해도가 높은 것을 좋은 것으로 규정하고, 분석 도구인 IBM Rational Software Analyzer™[8]가 제공하는 품질 지표들 중, 응집도, 복잡도, 이해도와 관련된 지표를 각각 세 개씩 선정하여 각 EPG 코드의 품질 지표를 계산하였다(3.2절 원칙). 표 1의 각 지표의 상세 의미는 부록 A.1절을 참고한다.

표 1 EPG 코드들의 품질 지표 계산 결과. 각 지표 옆에 표시된 위쪽 화살표(△)는 높을수록 좋고, 아래쪽 화살표(▼)는 낮을수록 좋은 것을 의미한다.

평가 지표		제품X	제품Y	제품Z
응집도 △	LCOM1 ▼	2.00	3.00	2.00
	LCOM2 ▼	0.77	0.89	0.80
	LCOM3 ▼	0.86	0.97	0.98
복잡도 ▼	CC ▼	3.08	3.39	2.53
	MI △	231.20	204.68	217.27
	WMPC ▼	4532	1526	1204
이해도 △	ANOCOM △	118.00	73.43	81.00
	CR △	9.58%	7.72%	12.06%
	ABD ▼	1.80	1.88	1.69

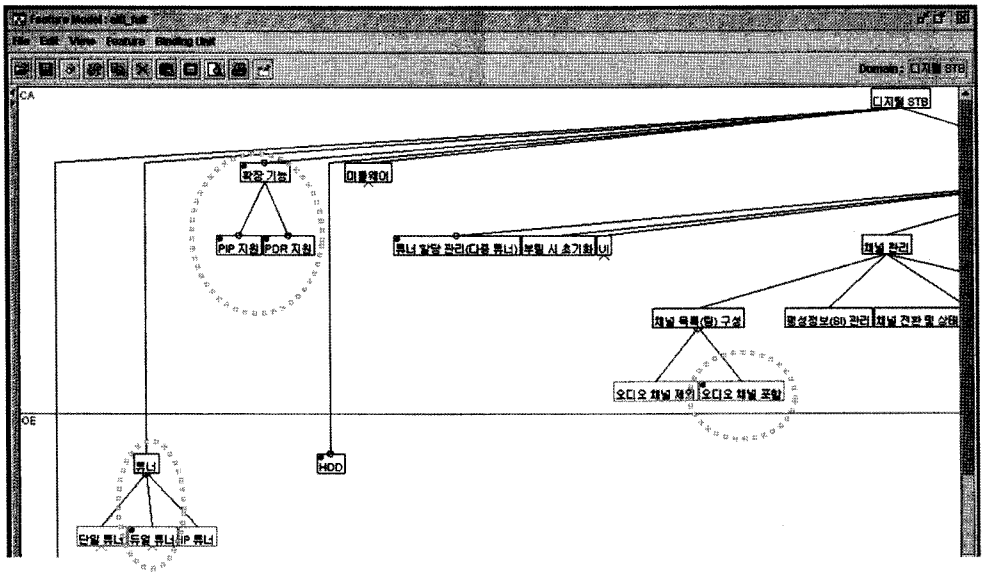


그림 2 자동화 도구에서 고급형 제품의 휘처를 선택한 결과

위와 같은 기존 자산 분석을 통해, 제품X의 코드가 품질도 우수하며 휘처 모델을 가장 잘 만족하고 있는 코드로 선정되어, 이 제품X의 코드를 바탕으로 나머지 제품Y와 제품Z의 코드와의 차이점을 매크로를 이용해 코드에 포함하도록 하여 초기 제품라인 모델을 구축하였다.

이 초기 제품라인 모델은 현재 A사가 제공하고 있는 세 방송 사업자의 EPG 제품을 모두 생성할 수 있다. 더 나아가, A사가 제공하고 있는 또 다른 두 방송 사업자의 EPG 제품을 생성하도록 하기 위해서 반응적(reactive) 접근법을 적용하여 제품라인을 확장할 예정이다.

4.3 개발된 자산의 유효성 검증

앞에서 설명한 과정을 통해 추출식 방식에 의해 구축된 제품라인 모델 및 자산에 대한 유효성을 검증하기 위해, 응용 공학(application engineering) 단계로, 구축된 자산을 이용해 두 가지(기본형/고급형) 다른 셋톱박스 하드웨어 위에서 동작하는 미들웨어 및 EPG 소프트웨어를 개발하였다.

기본형 셋톱박스는 튜너가 1개뿐이며 내장된 하드 디스크도 없어서, 동시에 다른 채널을 작은 화면으로 보여주는 PIP 휘처 및 특정 채널 프로그램을 녹화하는 PDR 휘처가 제공되지 않는다. 또한, 채널링(channel ring)[1] 에 오디오-온리[2] 채널도 포함되지 않는다. 반면, 고급형 셋톱박스는 튜너가 2개이고 내장 하드 디스크가 있어서 PIP 및 PDR 휘처를 제공하고 채널링에 오디오-온리 채널도 포함한다.

휘처 모델링 도구인 ASADAL[9]을 이용하여 미들웨어 및 EPG 휘처 모델을 포함한 디지털 셋톱박스 도메인의 전체 휘처 모델을 입력한 후, 앞서 설명한 두 가지 제품 중 고급형 셋톱박스를 위한 휘처를 선택한 것을 그림 2에 나타내었다.

선택된 휘처 정보는 XML 파일로 추출되고, 전처리기가 XML 파일의 내용을 분석하여 자산 소스 코드의 매크로 부분(4.2절 참조)을 고급형 셋톱박스에 맞게 치환하며, 이 코드를 컴파일 하면 고급형 셋톱박스를 위한 소프트웨어 제품이 생성된다.

이러한 과정을 통해 기본형 및 고급형 셋톱박스 제품을 체계적이고 효율적으로 생성할 수 있었고, 생성된 두 제품 모두 각 하드웨어 환경에서 정상적으로 동작함을 시연하는데 성공하였다. 그림 3에는 시연 과정 중 PDR/PIP 기능이 있는 고급형 셋톱박스 제품의 메뉴와(a), PDR/PIP 기능이 없는 기본형 제품의 메뉴(b)의 TV 화면 캡처 영상을 제시하였다.

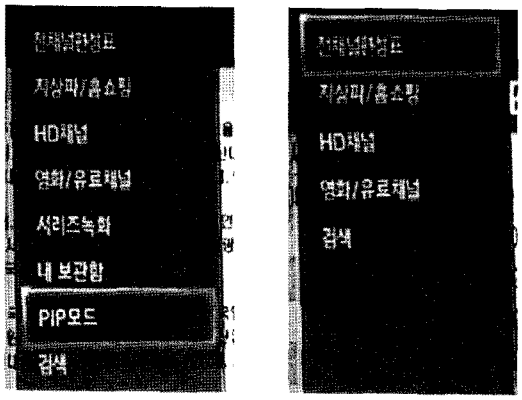
5. 교 훈

본 장에서는 사례 연구를 진행하면서 얻은 주요 교훈에 대해서, 휘처모델의 효용성과 추출식 접근법의 효과 측면에서 설명한다.

5.1 휘처모델의 효용성

휘처모델은 기존에 도메인 지식을 제품간 공통점과 차이점 측면에서 체계적으로 표현하여, 도메인 전문가와 개발자 사이의 효과적인 소통을 돕는 도구로 인정받아

1) 채널 이동 버튼을 통해 전환되는 채널들 리스트
2) 음악 채널이나 라디오 방송처럼 주로 소리만 방송되는 채널



(a) 고급형 메뉴 (b) 기본형 메뉴

그림 3 휘처 선택에 의해 생성된 셋톱박스 메뉴

왔지만, 본 사례 연구를 통해 그 뿐만 아니라 자산의 조합 규칙을 추상화하여 간결하고 이해하기 쉽게 표현하는 데에도 유용함을 확인하였다. 또한, 자산을 조합할 때의 제약 조건이 만족하는지 여부를 휘처모델 지원도구를 통해 쉽게 검증할 수 있음을 확인하였다.

도메인 전문가들은 초기에는 휘처모델 구축을 부담스러워하고 그 가치를 이해하지 못했지만, 휘처모델을 구성하는 과정을 통해 본인 스스로 도메인에 대한 이해를 높일 수 있음을 경험하면서, 휘처모델이 기존 전문가들의 지식을 체계화 할 수 있을 뿐만 아니라 신입 개발자들을 교육하는 데에도 효과적으로 이용될 수 있음을 인정하였다.

휘처모델과 유사하게 OVM(Orthogonal Variability Model)[5]도 가변성을 잘 나타내지만, 독립적으로 도메인 지식을 표현하기 보다는, 제품 모델이나 행위 모델, 또는 설계 모델과의 연결을 통해 이들 모델들에 가변성을 제공하기 위해 사용되며, 그 모델간 연결 관계가 자칫 복잡해지기 쉽기 때문에 본 연구에서는 사용하지 않았다.

5.2 추출식 접근법의 효과

A사 소프트웨어를 추출식 접근법을 통해 제품라인 체계로 전환하는 데에는 총 7명의 개발자 및 연구원들이 참여하여 약 4개월의 기간이 소요되었다. 이것은 기존 자산 및 도메인지식을 최대한 활용하는 추출식 접근법이었기에 가능하다. 이렇게 제품라인으로의 전환에 소요된 기간이 짧고, 그 결과 구축된 제품라인이 효과적인 것을 보여줌으로써, 산업체에서 제품라인 기법을 더욱 적극 수용할 수 있게 되리라 기대한다.

6. 관련 연구

우리가 제시한 추출식 제품라인 기법과 관련된 연구로, 특히 리팩토링 및 역공학 분야에서 많은 연구가 있었다[10]. 많은 사람들이 소프트웨어 개발 비용과 노력을 줄이기 위해 기존 시스템을 재사용하는데 주력했지만, 아직까지 소프트웨어 제품라인 관점의 역공학 성공 사례 는 많지 않다.

J. Bayer 팀은 기존의 소프트웨어 자산을 제품라인 아키텍처로 전환하는 'RE-PLACE' 프레임워크를 개발하였다[11]. 그들은 제품들의 휘처를 추출하여 자산 리엔지니어링과, 제품라인 디자인, 아키텍처 모델링을 수행하였다. R. Kolb 팀은 PuLSE™-DSSA 프로세스를 적용하여 이미지 메모리 핸들러 컴포넌트를 역공학 과정을 통해 소프트웨어 제품라인으로 전환하였다[12]. 이 두 연구팀들은 자산을 개발하기 위한 역공학 과정을 초기에 소개했으나, 도메인 내 제품들 사이의 공통점과 차이점을 체계적인 방법으로 분석하고 관리하지 못했다는데 한계가 있다.

FODA[1]가 소개된 이후 몇몇 연구팀은 휘처 중심(feature-oriented) 관점에서 역공학을 연구하였다. 특히, 텍사스 대학에서는 시스템을 휘처로 분해한 후, 휘처를 기반으로 역공학을 적용하는 FOR(Feature Oriented Refactoring) 프로세스를 소개하고 이를 바탕으로 JAVA 기반의 오픈 소스 데이터 베이스 시스템과 'AHEAD Tool Suite'을 개선하였다[13,14]. 그러나, 우리가 소개한 방법과 달리 이들의 방법은 제품라인 자산 구축을 위해 기존 제품 컴포넌트로부터 적당한 코드를 추출하는 데에는 초점을 맞추지 않았다.

V. Alves 팀은 관점 중심(aspect-oriented) 기법을 통해 모바일 게임 제품에서 제품라인을 추출하고 발전시키는 방법을 소개하였다[15]. 그들의 방법은 간단한 리팩토링 규칙을 이용하여 제품 라인을 추출하는 장점이 있지만, 기존 제품들 사이에 높은 디자인 유사성이 있다고 가정하였기 때문에, 본 논문의 EPG 제품 사례와 같이 제품들 사이에 구조적인 유사성을 찾기 힘든 경우에는 그들의 방법을 적용할 수 없다. 본 연구에서는 통합된 도메인 사전을 구축하고, 각 제품의 추출된 클래스들을 도메인 사전의 항목에 대응시킴으로써 디자인 유사성이 없는 제품 간 비교 및 기반 코드 선정이 가능하다.

7. 결 론

본 연구를 통해 산업체에서 제품라인 기법을 쉽고 빠르게 적용할 수 있는 추출식 접근법의 효과와 가능성을 확인할 수 있었다.

부록 A.2절에서 제시한 평가 지표에 따라, 추출식 제품라인 체계를 도입하기 전과 후의 A사의 조직과 셋톱박스 소프트웨어의 아키텍처 평가 결과는 다음 표 2와 같이, 전반적으로 아키텍처 수준과 조직 역량 모두에서

표 2 제품라인 기법 적용 전후 역량 평가

평가 지표		적용 전	적용 후
아키텍처 수준	자산 재사용 수준	2	3
	참조 아키텍처	3	4
	변화 가능성 관리	2	4
조직 역량	역할과 책임	2	3
	조직 구조	2	3
	협력 계획	2	3

개선이 있었는데, 특히 아키텍처가 변화 가능성 관리 측면에서 큰 개선을 보였다.

위에서 확인한 정성적 평가 지표 외에, 제품 당 개발 시간, 제품 당 개발 인원 수, 제품 당 버그 수, 버그 당 해결 시간 등의 정량적인 평가 결과도 향후 다양한 제품라인 체계로의 전환을 통한 장기간의 통계 수치를 통해 확인할 수 있을 것으로 기대한다.

추출식 접근법은 기존에 소프트웨어 재사용 및 제품라인 체계로의 전환을 희망했지만 초기 비용 및 시간의 투자를 꺼려하던 산업체에서 쉽게 수용할 수 있는 강점을 보였다. 다만, 이를 위해서는 해당 도메인에서의 개발 경험이 우수한 자산으로 구축되어 있고 이를 휘처모델을 통해 체계화 할 때에 가능하다.

본 연구를 통해 확인한 제품라인 기법의 적용 방법 및 사례, 평가 지표 등은 소프트웨어진흥원과 함께 지침서로 출간하여, 향후 산업체에서 제품라인 체계를 도입하고자 할 때 참조할 수 있도록 하였다.

참 고 문 헌

[1] K. Kang, S. Cohen, J. Hess, W. Nowak and S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Tech. Report CMU/SEI-90-TR-21, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, Nov. 1990.

[2] K. Kang, J. Lee and P. Donohoe, "Feature-Oriented Product Line Engineering," In *IEEE Software*, vol.19, no.4, pp.58-65, Jul./Aug. 2002.

[3] P. Clements and L. Northrop, Software Product Lines: Practices and Pattern, Addison Wesley, Upper Saddle River, NJ, 2002.

[4] D. M. Weiss and C. T. R Lai, Software Product-Line Engineering: A Family-Based Software Development Process, Addison Wesley Longman, Inc., Boston, MA, 1999.

[5] K. Pohl, G. Bockle and F. V. D Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer, 2005.

[6] W. B. Frakes and K. Kang, "Software Reuse Research: Status and Future," In *IEEE Trans. Software Engineering*, vol.31, no.7, pp.529-536, Jul. 2005.

[7] C. Krueger, "Eliminating the Adoption Barrier," In *IEEE Software*, vol.19, no.4, pp.29-31, Jul./Aug. 2002.

[8] http://www-01.ibm.com/software/awdtools/swanalyzer/?S_TACT=105AGX15&S_CMP=LP.

[9] K. Kim, H. Kim, M. Ahn, M. Seo, Y. Chang and K. Kang, "ASADAL: A Tool System for Co-development of Software and Test Environment Based on Product Line Engineering," In *Proc. of the 28th Int'l Conf. on Software Engineering*, pp. 783-786, May 2006.

[10] T. Mens and T. Touwe, "A Survey of Software Refactoring," In *IEEE Trans. Software Engineering*, vol.30, no.2, pp.126-139, Feb. 2004.

[11] F. J. Bayer, J. Girard, M. Wijrthner, J. Debaud and M. Apel, "Transitioning Legacy Assets to a Product Line Architecture," *Proc. the European Software Engineering Conf. and ACM SIGSOFT Int'l Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pp. 446-463, Sep. 1999.

[12] R. Kolb, D. Buthing, T. Patzke and K. Yamauchi, "A Case Study in Refactoring a Legacy Component for Reuse in a Product Line," In *Proc. of the 21st IEEE Int'l Conf. on Software Maintenance*, pp.369-378, Sep. 2005.

[13] J. Liu, D. Batory and C. Lengauer, "Feature Oriented Refactoring of Legacy Applications," In *Proc. of the 28th Int'l Conf. on Software Engineering*, pp.112-121, May 2006.

[14] S. Trujillo, D. Batory, and O. diaz, "Feature Refactoring a Multi-Representation Program into a Product Line," In *Proc. of the 5th Int'l Conf. on Generative Programming and Component Engineering*, pp.191-200, Oct. 2006.

[15] V. Alves, P. Matros Jr., L. Cole, P. Borba and G. Ramalho, "Extracting and Evolving Mobile Games Product Lines," In *Proc. of the 9th International Software Product Line conference*, pp.70-81, Sep. 2005.

[16] F. V. D. Linden, J. Bosch, E. Kamsties, K. Känsälä and H. Obbink, "Software product family evaluation," In *Proc. of the 3rd Int'l Conf. on Software Product Lines*, 2004, pp.110-129.

A. 부록: 평가 지표

본 절에서는 본문에서 소개한 코드 품질 평가 지표와 제품라인 체계 평가 지표에 대해 설명한다.

A.1 코드 품질 평가 지표

자동화 도구를 통해 계산된 소스 코드의 품질 지표들 중 본 연구에서 적용한 기준은 다음과 같다.

(1) 응집도(Cohesion)

- 응집도 결여도 1(LCOM1): 메소드가 사용한 변수들로 평가.

- 응집도 결여도 2(LCOM2): 메소드와 변수간의 관계로 평가.
- 응집도 결여도 3(LCOM3): 메소드와 변수간의 관계를 이용하되, 메소드에 가중치를 두어 평가.

위 세 가지 응집도 결여도는, 0 이상의 값이면 하나의 모듈이 두 개 이상으로 나누어 져야 함을 뜻한다.

(2) 복잡도(Complexity)

- 사이클로매틱 복잡도(CC): 메소드 간의 연관 관계로 평가. 값이 높을수록 복잡도 높음.
- 유지보수성 지표(MI): 복잡도 관련 지표들의 연산으로 유지보수성 평가. 높을수록 유지보수 용이.
- 클래스당 메소드 가중치(WMPC): 사이클로매틱 복잡도로 계산된 메소드의 가중치를 이용해 클래스의 복잡도 평가. 값이 높을수록 복잡도 높음.

(3) 이해도(Understandability)

- 평균 주석 개수(ANOCOM): 클래스당 평균 주석의 개수.
- 주석/코드 비율(CR): 전체 코드에 대한 주석 코드의 비율.
- 평균 블록 깊이(ABD): 코드 블록의 중첩된 횟수의 평균.

A.2 제품라인 체계 평가 지표

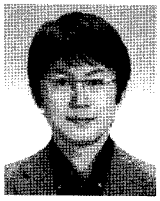
제품라인 체계 평가 지표는 BAPO(Business, Architecture, Process, Organization) 모델[16]에서 아키텍처 수준과 조직 역량 수준을 평가하는 기준을 차용하여 적용하였다. 상세 내용은 다음과 같다.

아키텍처 수준 평가 지표

	자산 재사용 수준	참조 아키텍처	변화가능성 관리
단계 1: 개별적인 개발	재사용이 없거나 잘 조직되어 있지 않다.	소프트웨어 제품라인 참조 아키텍처가 없다.	변화가능성이 전혀 관리되지 않는다.
단계 2: 표준화된 기반구조	재사용이 기반구조의 제품들을 이용하여 체계적이지 않고 임시방편으로 이루어진다.	참조 아키텍처가 기반구조로부터 만들어진다.	기반구조로부터 변화가능성이 제한적으로 제공된다.
단계 3: 소프트웨어 플랫폼	도메인의 자산이 정의된 공동의 플랫폼이 있고 재사용이 이 안에서 제한적으로 이루어진다.	제품이 참조 아키텍처를 이용한다. 참조 아키텍처가 플랫폼의 사용을 제한한다.	참조 아키텍처가 변화지점을 결정하고 이 점에서 제품이 변화할 수 있는 범위를 둔다.
단계 4: 다양하게 변하는 제품들	변화가능성을 지원하는 자산 저장소 안에서 재사용이 체계적이고 조직적으로 이루어진다.	참조 아키텍처가 다양한 제품 아키텍처를 만들 수 있다.	참조 아키텍처가 변화 가능점과 이 점에서 변화할 수 있는 부분을 제한한다. 제품은 이 제한을 따른다.
단계 5: 설정	4단계의 조건에서 변화가능성을 설정할 수 있다.	참조 아키텍처가 제품 아키텍처를 결정한다. 자동화된 설정으로 특정한 제품을 만들 수 있다.	변화가능성의 관리가 아키텍처와 통합된다. 설정에 의해 변화하는 부분이 자동적으로 만들어진다.

조직 역량 평가 지표

	역할과 책임	조직구조	협력계획
단계 1: 프로젝트	전문적이지 않은 응용 공학 역할이 있다.	조직이 프로젝트마다 단일하게 구성된다.	인적자원은 공유되더라도 소프트웨어 자산은 공유되지 않는다.
단계 2: 재사용	응용 공학 전문가가 프로젝트에 공동으로 참여한다.	이전에 만들어진 자원이 프로젝트를 위해 사용된다.	프로젝트 사이에 교섭과 정보공유가 이루어진다.
단계 3: 약한 협력	도메인 공학과 응용 공학의 역할과 책임이 독립적으로 정의되어 있다.	이전 단계와 마찬가지로 프로젝트를 중심으로 조직이 구성되지만, 도메인 공학과 응용 공학의 역할이 분리되어 있다.	도메인 공학과 응용 공학의 프로젝트 사이에 문서를 기반으로 한 협력이 이루어진다.
단계 4: 동시 진행	도메인 공학과 응용 공학 사이에 대등한 역할이 있다.	조직구조가 주구조와 부구조로 나뉘어져 있다. 주구조는 도메인 공학과 응용 공학의 주요한 부분을 담당하고, 부구조는 기능적인 부분을 담당한다.	다양한 조직 사이에 강한 협력이 이루어지고, 역할수행을 위한 정기적인 모임이 있다.
단계 5: 도메인 중심	도메인 공학이 소프트웨어 개발에 중요한 역할을 하고 응용 공학은 작은 부분을 차지한다.	조직구조가 도메인 공학을 중심으로 구성된다. 주구조는 도메인 공학을 담당하고 부구조는 응용 공학과 기능적인 부분을 담당한다.	사람들이 필요에 따라 도메인 공학이나 응용 공학의 역할을 맡는다.



최 현 식

포항공과대학교 컴퓨터공학과 졸업(1996년 학사, 1998년 석사). 2001년~2008년 LG전자 디지털TV연구소 주임/선임/책임 연구원. 1998년~2000년 및 2008년~현재 포항공과대학교 컴퓨터공학과 박사과정, 관심분야는 소프트웨어 재사용, 제품라인 공학, 실시간 내장형 시스템, 휘처 모델링

이 혜 선

2009년 포항공과대학교 컴퓨터공학과 졸업(학사), 2009년~현재 포항공과대학교 컴퓨터공학과 대학원 통합과정. 관심분야는 소프트웨어공학, 소프트웨어 재사용, 제품라인 공학, 추출식 제품라인 접근

조 윤 호

정보과학회논문지 : 소프트웨어 및 응용 제 36 권 제 6 호 참조

강 교 철

정보과학회논문지 : 소프트웨어 및 응용 제 36 권 제 6 호 참조