소프트웨어 제품 군을 개발하기 위한 점진적 방법

주 복 규[†]·김 영 철[††]

요 약

소프트웨어 제품 군 개발 방법은 모든 멤버 시스템에 대한 공통점과 차이점들을 분석하여 표준 소프트웨어 구조와 컴포넌트들을 개발하여 생산라인을 갖추고, 제품 개발 조직에서 여러 버전을 병행하여 생산하고 관리하는 방법이다. 기존의 제품 군 방법은 성공하였을 경우에 경제적 이익은 크나, 초기 투자비가 매우 크다는 것과 첫 제품이 늦게 나온다는 것이 문제점으로 지적되고 있다. 이 논문에서 우리는 초기 비용을 적게 들이면서 궁극적으로 제품 군 개발 방법을 조직에 적용할 수 있는 점진적 방법을 제안한다. 이 방법은 초기 버전을 개발할 때부터 변화 가능성에 관한 정보를 기록하여 차기 버전의 개발에 활용한다. 이 변화 분석의 결과는 본격 제품 군 개발로의 전환을 쉽게 해 준다. 변화 가능성을 기록하는 변경 사양서의 생성과 개선 기법을 중심으로 우리의 방법을 설명하고, YBS 시스템 개발을 예로 그 적용을 보인다. 우리가 제안한 방법은 초기에 많은 투자를 할 수 없거나 초기버전을 빨리 시장에 내 놓아야 하는 경우에 적용할 수 있는 저 위험 제품 군 개발 방법이다.

Incremental Method for Developing Software Product Family

Bok-Gyu Joo[†] · R. Young-Chul Kim[††]

ABSTRACT

In a software product line approach, developers first develop common software architecture and components by analyzing the characteristics of all software members, and then produce each application by integrating components. The approach is considered very effective means for developing and maintaining in parallel a software product family. Main disadvantage of this approach is that it requires a big up-front investment in preparing product line. Therefore, it takes time to deliver the first version. In this paper, we present an incremental method to develop software families, which requires small additional cost for initial versions and allows an organization to move smoothly to full-scale product line. We present our method by explaining how to record and upgrade the results of variations analysis, and show the application of our method by developing a family of YBS. Our method is a low-risk approach that can be effectively applied to an organization that starts developing software systems but has to deliver the first versions quickly to the market.

키워드 : 소프트웨어 제품 군(Software Product Family), 제품 군 접근법(Product Line Approach), 변화 분석(Variation Analysis), 변경 사양서(Variation Specifications), 군 문서(Family Documents), 소프트웨어 재사용(Software Reuse)

1. 서 론

소프트웨어를 개발하는 회사나 조직은 하나의 시스템만을 개발하는 것이 아니라 역할이 비슷한 여러 개의 버전들을 개발하고 관리한다. 기능이 조금씩 다른 시스템을 순차적으로 개발하기도 하고, 같은 기능을 하는 소프트웨어를 서로 다른 플랫폼이나 사용자들을 위해서 여러 버전으로 개발하고 향상시킨다. 이와 같이 비슷한 기능을 하는 소프트웨어 시스템들을 한꺼번에 고려하여 이들을 '소프트웨어 군'(또는 프로그램 군)이라 부르고, 그 군에 속하는 하나의 시스템을 '소프트웨어 멤버'(또는 프로그램 멤버)라 부른다.

소프트웨어 개발 과정에서 개인이나 조직은 생산성을 높이기 위해 이전에 개발한 소프트웨어의 소스코드나 설계서 같은 산출물 뿐만 아니라 개발하면서 얻어진 여러 가지 지식과 경험을 재사용한다. 개발 결과나 경험을 조직적으로 재사용하기 위한 방법에 관해서는 지금까지 여러 측면에서 연구되고 적용되어 오고 있다. 최근에는 자동차의 여러 모델을 생산하는 '생산 라인(product line)' 개념을 소프트웨어 개발에 적용하여, 여러 버전의 시스템을 한꺼번에 고려하여 경제적이고 빠르게 개발하고 관리하는 기법을 사용하기 시작하고 있으며, 이를 '소프트웨어 제품 군(software product family)' 접근법[1]이라 부른다[4, 19].

제품 군 접근법에서는 먼저, 하나의 군에 속하는 모든 제품들에 대한 공통점과 차이점들을 분석하여 표준 소프트웨어 구조와 그에 맞는 컴포넌트들을 개발하여 생산라인을

※ 여러 버전의 YBS 시스템을 개발하는 동안, 초기 버전의 개발 이후에 고려한 많은 UI의 변화에 대하여 Java 언어로 그 변경들을 따라오며 개선해 준 Escape의 황 지익 군에게 감사의 말을 전한다.
† 종신회원 : 홍익대학교 전자전기컴퓨터공학부 교수
†† 정 회 원 : 홍익대학교 전자전기컴퓨터공학부 교수
논문접수 : 2002년 10월 19일, 심사완료 : 2003년 3월 19일

1) 소프트웨어 제품 라인(software product line) 접근 법이라고도 불린다.

갖추고, 제품 개발 조직에서 여러 버전을 병행하여 개발하고 유지 보수한다. 이 개념은 기존의 재사용 방법과는 달리, 관심의 범위를 일반적인 문제가 아니라 '한 제품 군'에만 초점을 맞춤으로써 실제 개발 조직에 적용하기 쉽다. 또한 표준 구조에 따라 미리 준비한 컴포넌트는 재사용성이 매우 높아 생산성을 획기적으로 높일 수 있다고 여겨지고 있으며, 산업계에서 더욱 널리 적용할 것으로 예상된다[12].

제품 군 접근법은 군에 속하는 멤버들이 많아서 성공하였을 경우에 경제적 이익은 크나, 문제점은 초기 투자비가 매우 크다는 것이다. 따라서 처음 제품 개발을 시작할 때(즉 초기 버전을 만들 때)에 이 방법을 적용하는 것은 위험이 따른다. 개발 초기에는 미리 멤버들의 특성을 충분히 분석할 수 없는 경우가 많고, 또한 제품이 시장에서 실패할 가능성도 있기 때문이다. 제품 군 접근법의 또 하나의 단점은 첫 버전이 늦게 나온다는 것이다.

이 논문에서 우리는, 초기 비용을 적게 들이면서 궁극적으로 제품 군 개발 방법을 조직에 적용할 수 있는 저 위험 방법인 NISE(New Incremental Software-family development Environment) 방법을 제안한다. 이 방법은 초기 버전들을 개발할 때 비교적 적은 노력으로 변화 가능성을 분석하고, 이를 다음 버전의 개발에 활용한다. 또한 이 변화 분석의 결과는 나중에 본격적으로 제품 군 개발 방법을 조직에 적용하고자 할 때 쉽게 전환할 수 있게 해 준다. NISE 방법은 제품 군 개념을 적용하고 싶으나, 초기에(시간과 비용 측면에서) 많은 투자를 할 수 없거나 초기버전을 빨리 시장에 내 놓아야 하는 경우에 효과적으로 적용할 수 있는 방법이다.

이 논문의 구성은 2장에서 소프트웨어 제품 군 접근법의 개념과 경제성, 관련 연구들을 제시하고 기존 방법들의 문제점을 분석한다. 3장에서는 우리가 제안하는 NISE 방법을 설명하고, 4장에서는 YBS(Year Book System) 개발에 우리 방법을 적용한 결과를 보이고, 이를 바탕으로 제안한 방법의 장 단점을 분석한다. 마지막으로 5장에서는 결론과 향후 연구 방향을 제시한다.

2. 관련 연구

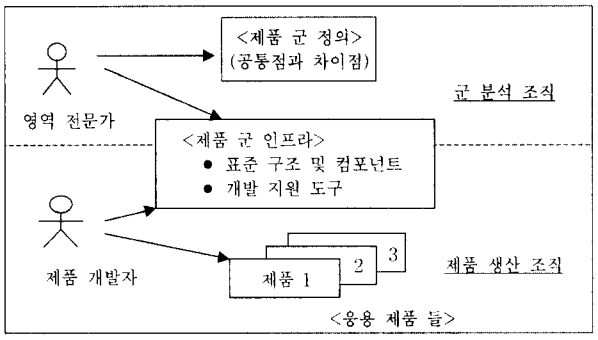
2.1 제품 군 접근법과 그 경제성

소프트웨어 시스템을 개발할 때 기존의 산출물이나 개발 경험을 조직적으로 재사용하기 위한 노력의 결과로, 지금까지 소스코드 재사용 측면에서는 공통으로 사용할 수 있는 많은 종류의 함수 또는 클래스 라이버러리가 제공되어 개발자들은 개발 도구에서 제공하는 다양한 컴포넌트를 이용해 구현하고 있다.

한편, 요구분석과 설계 과정의 결과에 대한 재사용성을 높이기 위해서 제시된 방안으로 대표적인 것이 영역 공학(domain engineering) 방법이다. 이 방법은 우리의 관심을 모든 문제를 대상으로 하지 않고 특정 응용 분야, 즉 한 영역(domain)에 집중한다. 우선 영역 분석(domain analysis)을 통하여 그 영역에 맞는 '소프트웨어 인프라'[2] 즉 표준 구조(reference architecture)와 그에 맞는 컴포넌트를 개발하고, 제품 개발 단계에서는 영역 분석에서 얻어진 자산들을 사용하고, 컴포넌트들을 조합하는 방법으로 응용 시스템을 개발한다.

소프트웨어 제품 군 접근법은 영역공학 방법에서 발전한 개념으로 비슷한 기능을 갖는 소프트웨어 시스템들을 계속해서 개발 관리하는 조직에 적용할 수 있다. 이 개념은 자동차 모델 개발과 생산에서 사용하는 제품 라인 방법을 소프트웨어 개발에 적용한 것으로, 한 조직의 관심 범위를 한 영역이 아니라 재 사용을 극대화 할 수 있는 비슷한 제품 군(product family)으로 한정하였다. 따라서 한 조직의 입장에서는 가장 경제적인 방법으로 인식되고 있다[4, 19].

(그림 1)에서 보여주는 바와 같이 제품 군 접근법은, 영역 공학 방법과 같이 개발을 크게 두 개의 활동으로 구분하고, 이에 따라 조직도 '제품 군 분석 조직'과 '제품 생산 조직'으로 구분한다.



(그림 1) 소프트웨어 제품 군 접근법

제품 군 방법에서는 우선, 마케팅 계획에 따라 제품 군의 범위, 즉 제품의 라인업을 정한다. 군 분석 조직은 그 제품 군에 속하는 모든 멤버들의 요구 사항의 특성 즉 공통점과 차이점을 분석하여 군 핵심 자산인 공통 소프트웨어 구조와 그에 따른 컴포넌트, 그리고 필요한 도구들을 만든다. 제품 생산 조직은 컴포넌트들을 조합하는 방법으로 특정 응용 시스템을 개발한다. 이때 각 제품이 순차적으로 만들어지는 것이 아니고 시장 요구에 따라 병행적으로 개발, 관리된다는 점에서 '제품 라인'이라 불린다.

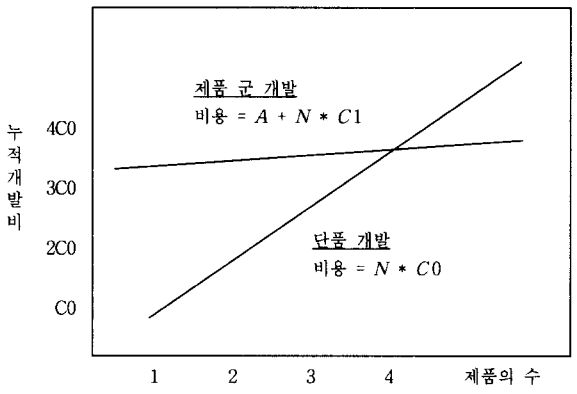
제품 군 접근 방법의 장점은 비슷한 기능을 하는 소프트웨어 제품들의 특성을 면밀히 분석하여 표준 구조와 그에 맞는 컴포넌트들을 개발하기 때문에 이들의 재사용성을 극대화시킬 수 있어 경제성이 매우 높다. 한편 조직적인 측면

2) 소프트웨어 플랫폼 또는 핵심 자산(core assets)이라고도 불린다.

에서도, 관련된 조직(마케팅 조직, 개발 조직 등) 모두의 역량을 하나의 제품 라인업을 중심으로 유기적으로 결합하여 경쟁이 심한 시장 환경에 적극적으로 대응할 수 있다.

기존의 단품 개발방법과 비교하여 제품 군 개발방법의 경제성을 분석하여 보자. 한 조직이 N개의 비슷한 제품을 개발한다고 할 때, 하나씩 개발할 때 평균 개발비를 $C0$라 하고, 제품 군 방법을 적용하여 개발할 때 평균 비용을 $C1$이라 하자. 또한 제품 군 방법을 적용하였을 때 초기 투자비(제품 군의 공통점과 차이점을 분석하고, 표준 구조와 필요한 컴포넌트를 개발하는 비용)를 A라 하자. N개의 제품을 단품 개발방법을 사용하여 개발할 때 총 비용은 $N*C0$가 되고, 제품 군 방법을 적용할 때에는 $A+N*C1$이 된다. (그림 2)는 제품 군 개발방법의 경제성을 분석한 모델을 보여준다[5, 17]. 제품 군 개발방법이 경제성을 이루기 위해서는 $C1$이 $C0$보다 아주 작아야 하고, 초기 투자비를 보상하기 위해서 N이 충분히 커야 한다. 이 그림에서 투자비 회수기간(break-even point)은 4(4개의 멤버)이고 A는 $3*C0$과 $4*C0$사이의 값을 가짐을 보여준다.

제품 군 개발 방법에서 초기 투자비는 얼마만큼 많은 준비를 하고 자동화 하였는가에 따라 다르나, A가 크면 $C1$은 작아지고, A가 작으면 $C1$은 커질 것이다. 즉, 적용 초기에 군에 속한 모든 제품들의 특성을 분석하고, 핵심 자산을 개발하기 위해 많은 투자를 필요로 하지만, 여러 제품들을 생산하고 나면 전체 비용이 단품 개발 방법보다 적어지고 제품 군에 속한 멤버의 숫자가 많으면 많을수록 경제적 효과는 더 커진다. 따라서 비슷한 소프트웨어 버전들을 많이 개발하고 관리하는 조직의 입장에서는 가장 경제적인 방법으로 여겨지고 있다.



(그림 2) 제품 군 개발 방법의 경제성

2.2 제품 군 방법론과 변화의 관리

2.2.1 제품 군 방법론

제품 개발방법으로는 Lucent Technology에서 사용하고 있는 FAST(Family-based Abstraction and Translation)가 대표적인 것이다[1, 19]. 이 방법에서는 제품 군에 대한 일반적 분석 방법으로 다음 세 가지 단계를 제시하였다. 첫째, 제품 군의 공통점과 차이점을 찾아내고, 둘째, 차이점을 파라메타화 하고, 셋째, 분석 단계의 주요 이슈들을 문서화 하는 것이다. 영역 분석의 결과는 제품 군의 사전(dictionary), 공통점과 차이점으로 정리된다. FAST 방법의 특징은, 제품 군 분석의 결과로 응용 시스템들을 모델링하기 위한 언어 AML(Application Modeling Language)를 고안해 내어서 각 버전의 사양을 정확히 기술한다. 이렇게 기술된 시스템은 프로그램으로 자동 번역되도록 하는 것이 목표이며, 여러 분야에서 이 기법을 적용한 사례를 보여준다. FAST 방법을 적용하려면 그 응용 분야에 맞는 AML을 고안해 내고, 지원 도구를 만드는데 많은 노력이 소요된다. 그렇지만, 한번 AML으로 응용 시스템들을 모델링하고 지원 도구를 만든다면, 소프트웨어 시스템의 생산은 거의 자동화 할 수 있다.

SEI(Software Engineering Institute)에서 제안되어 널리 사용되고 있는 FODA(Feature-Oriented Domain Analysis) 방법에 기반을 둔 FORM(Feature-Oriented Reuse Method) 방법론은 포항공대를 중심으로 연구되고 있고, 국내외 산업에 널리 적용되고 있다[7, 8]. FORM은 FODA 방법론과 지원 도구를 기반으로 마케팅과 제품 계획(MPP) 까지를 포함하여 제품 군 개발환경으로 발전시킨 것으로 이 방법의 특징은 제품 군 분석과 모든 개발과정에 제품 기능(product feature)을 중심으로 이루어 진다.

유럽의 IESE에서 개발 사용하고 있는 PuLSE(Product Line Software Engineering)는 다른 방법론과 마찬가지로 영역공학 방법에서부터 발전된 것이다[2]. PuLSE는 여러종류의 기업환경에서 소프트웨어 제품 군 개발방법을 계획하고 실행시킬 수 있도록 하는 통합 방법론이며, 크게 세개의 요소(적용 단계, 기술 요소, 지원 요소)로 구성되어 있다. 적용 단계는 한 조직이 제품 군을 확립하고 실행하는 활동들에 관한 방법을 기술하며 초기화, 인프라 개발, 인프라 활용, 그리고 인프라 개선과 관리 방법에 관한 것이다. 기술 요소는 제품 군 개발을 실행하는데 필요한 기술적 노-하우를 제공하는 것으로 어떻게 초기화 수행, 범위 설정, 모델링, 구조설계, 제품 군의 개선 관리를 수행하는가 하는 기술을 제공하며, 전 적용단계의 업무에 사용된다. 지원 요소는 제품 군 적용활동을 적응시키고, 점진적으로 개선하는가에 관한 가이드라인을 제공한다.

산업계에서 제품 군 방법을 적용하여 개발한 사례에 대한 대표적인 결과로서는, 승무원을 위한 실시간 교육 시스템 개발에 제품 군 접근법을 적용한 사례[11], 자동차의 안전, 안락, 위험방지 등 여러 가지 서비스들을 지원하기 위한 제품의 차이점들을 모델링하여 적용한 사례[18], Nokia에서 이동 기기에 쓰이는 브라우저(mobile browser)의 개발에 제품 군 방법을 적용한 사례[6], 유럽의 주요 연구 및 적용

사례 등이 있다[10, 14].

2.2.2 변화의 관리

비슷한 기능을 하는 여러 개의 시스템을 동시에 고려하여 분석하고 개발할 때 가장 중요한 기술의 하나는, 공통점과 차이점을 찾아내어서 표준 구조를 만드는 것과 차이점 또는 변화 내용들을 효과적으로 통제하는 것이다. Parnas는 '프로그램 군'이라는 개념을 처음 정의하였고, 효과적인 방안으로 쉽게 확장 또는 축소할 수 있게 설계하는 방법을 제시하였으며, 이를 위해 요구 사양의 작성에 있어서 '최소한의 요구사항 집합'과 요구 항목의 '순차적 증감'을 사용하도록 제안하였다[15, 16].

'SCV 분석법'(Scope, Commonality, and Variability Analysis)은 소프트웨어 제품 군 개발방법에서 가장 기본적인 개념인 범위 설정과 제품들의 공통점과 차이점을 분석하는 모델을 제시하였으며 이 개념은 위에 언급한 FAST 방법 등의 이론적 근거를 제공한다[5]. 그 외에도 항공 엔진 제어 시스템 영역에서 제품 군 방법을 적용하여 요구 사항의 차이를 발견하고 이를 조직적으로 재사용하기 위한 연구, 소프트웨어 구조의 차이점을 표현하기 위해서 프레임 기술을 사용하고, 표준 구조로부터 특정 멤버의 구조로 만드는 (customizing) 방법에 관한 연구가 있다[9, 3].

이미 널리 사용되고 있는 소프트웨어 버전 관리 방법과 형상 관리 방법들도 시스템들의 차이점들을 관리할 수 있는 효과적인 기법이다. 그러나 이들은 변화의 사후 처리, 즉 주로 개발 시작 이후에 나타난 차이점들을 관리하기 위한 것이다. 제품 군 접근법에서는 제품 생산이 일어나기 전에 차이점을 분석하고 변화를 계획하고 통제하고자 한다. 따라서 제품 군 방법으로 계획된 차이점들을 관리하기 위해서 이 도구들을 사용할 수 있다.

2.3 기존 접근 방법의 문제점

지금까지 연구되어온 제품 군 개발 방법과 적용 사례들은 대부분 한 조직에서 비슷한 제품들을 개발한 결과가 있어서 그 경험과 결과로부터 군 분석을 실시하고, 표준 구조를 결정하고, 컴포넌트들을 개선하여 개발 인프라를 준비하는 방법이다. 그러나 제품 군 접근법이 약속한 효과를 내려면 다음과 같은 조건이 만족되어야 한다. 첫째로, 제품 군에 속하는 멤버들의 수가 많아야 한다. 둘째는, 개발된 공통 구조가 큰 변화없이 모두에 사용되어야 그에 따른 컴포넌트들의 재사용성이 극대화 된다. 이를 위해서는 이미 많은 시스템을 개발한 경험과 산출물들이 있고, 이로부터 그 제품 군의 특성을 잘 파악할 수 있어야 한다. 그러나 어떤 제품의 초기 단계에서는 산출물과 경험이 부족하여 그 제품 군의 특성을 충분히 이해하지 못하며, 따라서 변하지 않을 표준 구조를 만든 다거나, 재사용성이 높은 컴포넌트들을 미리 만든다는 것은 매우 어렵다.

제품 군 접근법을 적용할 때 발생하는 초기 투자비 부담은, 제품 개발이 성공하였을 경우에는 그 비용이 여러 멤버들에 나누어져 이익을 가져올 수 있지만, 제품이 시장에서 실패하여 제품 생산이 조기에 중단될 경우에는 큰 경제적 손실로 이어진다. 따라서 어떤 제품의 초기 버전 개발에 이 방법을 적용하는 것은 위험하다. 초기 버전의 개발에 이 방법을 그대로 적용하기 어려운 또 하나의 이유는 첫 버전이 매우 늦게 나온다는 것이다.

이 논문에서 제시하는 방법은 시스템의 초기 버전들을 개발할 때 기존의 단품 개발에 조그마한 추가 투자를 하고, 몇 버전을 개발한 이후에는 기존의 제품 군 접근법과 같은 효과를 낼 수 있는 점진적 접근 방법이다. 이 방법은 제품 군 접근법을 조직에 적용하고 싶으나 시간적 이유나 기술적 이유로 초기 투자를 하기 어렵거나, 초기 제품이어서 실패 가능성이 있다고 판단되는 경우에 적용하면 효과적이다.

3. NISE 방법

NISE 방법은 비슷한 소프트웨어 시스템들을 개발함에 있어, 기존 제품 군 방법들이 초기 투자비가 많이 드는 단점을 보완하여, 초기 버전 개발에 많은 추가 투자없이 진행하고, 점진적으로 제품 군 개념을 조직에 확립할 수 있는 저위험 소프트웨어 제품 군 개발 방법이다.

3.1 기본 개념 : 변화 가능성의 발견과 관리

일반적인 소프트웨어 개발과정을 자세히 살펴보면, 개발자는 요구분석 단계에서 추상적인 요구 사항들을 구체적이고, 세세한 요구 항목들로 바꾼다. 기본적이며 추상적인 요구사항을 구체화하고 애매한 사항에 의문점을 가지고 그 해답을 찾는 과정에서, 여러 가지 대안들을 고려하고 그 중 하나를 최종 선택하게 된다[13]. 또한, 설계 단계에서도 요구 사항에 맞는 최적 설계를 위한 여러 문제들(예를 들면 소프트웨어 구조, 자료 구조, 알고리즘의 선택 문제 등)에 대한 대안을 찾아내고 검토하여 현재 여건에서 가장 좋다고 판단되는 대안을 선택하게 된다. 고려한 대안들은 대부분 여건에 따라서 달리 선택될 가능성이 있는 것 들이다. 또한 우리는, 그 대안 중에서 지금 시스템의 기능으로는 결정되지 않았지만 다음에는 꼭 있으면 좋은 기능들도 발견하게 된다[3].

NISE 방법의 핵심은 초기 버전을 개발하면서, 각 요구 항목에 대하여(설계 항목에 대해서도 마찬가지 임) 변경 가능성을 인지하고, 그 대안들을 분석한 결과를 정리하여 '변경 사양서(variation specification)'를 작성하고, 이를 다음 버전의 개발에 활용하는 것이다. 또한 새로운 버전이 개발됨에 따라 이 변경 사양서를 개선시켜, 본격 제품 군 방법

3) 이러한 기능들의 목록을 보통 '희망 사항 목록'(wish list)이라 부른다.

을 적용할 때 군의 핵심 자산으로 전환하기 쉽게 하는데 있다. 다시 말하면, 개발자는 한 버전의 개발 결과로 지금 버전에 필요한 산출물들(요구 사양서, 설계서 등)만을 기록하는 것이 아니고, 발견한 변화 가능성과 고려한 대안들도 모두 정리하여 변경 사양서의 형태로 기록하여 둔다. 그리고 다음 버전을 개발할 때 기본 산출물들과 함께 이를 활용하여 개발 비용을 줄인다.

3.2 프로세스와 산출물

NISE 접근법에서 소프트웨어 제품 군의 생명 주기는 크게 세가지 단계 즉, 초기 버전 개발 단계, 제품 군으로 전환 단계, 그리고 그 이후의 본격적인 제품 군 생산 단계로 나눈다(그림 3). 초기 개발 단계는 제품 군 개념을 본격적으로 조직에 도입하기 전 초기 시스템들을 개발하는 단계로서, 처음 버전부터 변화 가능성을 분석하여 변경 사양서를 작성하고, 다음 버전의 개발에 활용하며 이를 보완해 나간다. 제품 군으로 전환 단계는, 하나 이상의 시스템을 개발한 후 이들이 시장에서 성공하여 본격적으로 제품 군 방법을 조직에 적용하고자 할 경우에, 지금까지 산출물들을 정리하여 '군 사양서들(family specifications)'과 '군 변경 사양서들(family variations specifications)'을 만들고, 이에 따라 컴포넌트들을 개선하는 단계이다. 제품 군 생산 단계는 본격적으로 제품 군 개발방법을 적용하는 단계로서, 개발자는 개발 경험의 집합체인 군 문서를 이용하여 새 멤버를 개발하고, 관리한다.

(그림 3) NISE 방법에서 제품 군의 생명 주기

NISE 개발 방법에서는 개발시 통상적으로 작성하는 기본 산출물들(예 : 요구 사양서, 설계 사양서, 소스 코드) 외에 요구 사항에 대한 변경 가능성을 기록한 '요구 변경 사양서(RV-Spec)', 설계 사양에 대한 변화들을 기록한 '설계 변경 사양서(DV-Spec)'를 작성한다. 이들 변경 사양서들은 차기 버전의 개발 과정에서 계속 개선된다(<표 1>).

<표 1> 단계별 활동과 산출물 들

단계	초기 버전 개발	제품 군으로 전환	제품 군 생산
활동	• 소프트웨어 초기 버전 개발 • 변화 분석 및 변경 사양서 개선	• 제품 군 특성 분석 및 군 문서 작성 • 컴포넌트 개선	• 응용 시스템 개발
산출물	• 각 버전의 산출 물 (R-Spec, D-Spec, Source Code 등) • 변경 사양서 (RV-Spec, DV-Spec)	• 군 사양서 (FR-Spec, FD-Spec) • 군 변경 사양서 (FRV-Spec, FDV-Spec) • 재사용 컴포넌트	• 개선된 군 문서들

제품 군으로 전환시에는 각 버전의 요구 사양서와 설계 사양서는 각각 군 요구 사양서(FR-Spec)와 군 설계 사양서(FD-Spec)로 통합되며, 요구 변경 사양서와 설계 변경 사양서는 각각 군 요구변경 사양서(FRV-Spec)와 군 설계 변경 사양서(FDV-Spec)로 발전한다(<표 2> NISE 문서들의 약어 참조). NISE 방법에서 설계변경 사양서의 형식과 그 생성 방법은 요구변경 사양서의 그것과 같다. 따라서 이 논문에서는 요구 사양서와 요구변경 사양서를 중심으로 기술한다.

<표 2> 이 논문에서 제안한 NISE 문서들의 약어

사양서 이름	약 어	군사양서 약어
• 기본 사양서	Spec	F-Spec
◦ 요구 사양서	R-Spec	FR-Spec
◦ 설계 사양서	D-Spec	FD-Spec
• 변경 사양서	V-Spec	FV-Spec
◦ 요구변경 사양서	RV-Spec	FRV-Spec
◦ 설계변경 사양서	DV-Spec	FDV-Spec

3.3 초기 버전 개발 단계

NISE 방법에서 하나의 시스템을 개발하는 방법은 기존의 소프트웨어 개발 방법과 같다. 차이점은, 첫 버전 개발부터 기본 사양서들(즉 요구 사양서와 설계 사양서)외에 '변화 분석(variation analysis)'과정을 통하여 변화 가능성들을 정리해 변경 사양서(즉 요구변경 사양서와 설계변경 사양서)들을 만드는 것이다. 다음 버전의 개발은 전 버전의 사양서와 변경 사양서를 기반으로 산출물을 만들고, 동시에 변경 사양서들을 개선한다.

3.3.1 첫 버전의 개발

마케팅 계획서(또는 제품 개념서)를 기반으로 요구 분석과 변화 분석을 통하여 R-Spec(요구 사양서)과 RV-Spec(요구변경 사양서)을 만들어 낸다. 개발자는 요구항목 작성과 함께, 변화 가능성이 있는 항목을 표시하여 둔다(변화 인지). 변화 가능성이 있는 항목에 대해 가능한 대안들을 찾아서 기록 정리하여 RV-Spec을 작성한다. 위 기본개념 절에서 언급한 바와 같이 이 작업은 분석 과정에서 대부분 고려하는 사항이므로 변경 사양서를 작성하는 비교적 적은 노력이 소요된다.

R-Spec : R-Spec의 요구 항목들에는 크게 '불변 항목'과 '가변 항목'의 두 종류가 있다. 불변 항목은 변경 가능성이 발견되지 않은 것이고, 가변 항목은 변경 가능성이 있는 항목으로서 RV-Spec에 그 변화 값들이 정리된다. 각 요구사항 항목에는 다음과 같은 내용을 기록한다.

① 요구 사항 주제
② 요구 사항 설명
③ 타 항목과의 관계
④ 변경 항목 참조

여기서 '타 항목과의 관계 ③'는 주로 어떤 항목이 이 항목에 영향을 미치는가를 나타낸다. '변경 항목 참조 ④'는 가변 항목의 경우에 RV-Spec에 있는 해당 변경 항목을 가리키며 R-Spec으로부터 RV-Spec으로 추적을 가능하게 한다. 그 참조로는 변경항목 이름 또는 변경항목 번호를 사용한다.

RV-Spec : 요구 사항에 대한 변화 분석 결과를 기록하는 RV-Spec은 변경 항목의 집합으로, 각 항목은 R-Spec 항목 중 가변 항목에 대한 변경 정보를 가지고 있다. 각 항목에는 다음과 같은 내용을 기록한다.

① 변경 사항 이름
② 요구 항목 참조
③ 변화 내용 제목 또는 보충 설명
④ 가능한 대안들
⑤ 각 버전의 선택

여기에서 '변경 사항 이름 ①'은 R-Spec에서 가변 항목의 '변경 항목 참조'에 있는 '변화 이름'과 같은 것으로 R-Spec에서 RV-Spec으로의 추적을 가능하게 하며, 항목 ②는 이 변경항목이 기술하는 해당 요구 항목을 가리키며 RV-Spec에서 R-Spec으로의 추적을 가능하게 한다. 항목 ④는 여러 개의 가능한 대안들을 기록해 두게 되는데, 이때 각 대안을 적용하는데 참고할 정보도 함께 기록해 둔다. 또한 항목 ⑤는 위의 대안들 중에 각 시스템 버전들이 선택한 값을 나타낸다. (R-Spec과 RV-Spec의 상세한 형식은 부록 참조)

예를 들어서, 개발자가 '사용자는 키보드를 이용하여 자료를 입력한다'라는 요구사항을 만나고, 또한 추후에 다른 멤버를 개발할 때 키보드 외에 펜이나 음성으로도 가능할 것이라고 고려 되면, R-Spec과 RV-Spec에는 다음과 같이 기록한다.

```
<R-Spec의 일부>
R3 : [시스템 입력 장치] 시스템의 입력 장치에는 Mouse와 Keyboard
     가 있다.
R5 : [입력 수단] 사용자는 키보드를 사용하여 자료를 입력한다.
     /VA-V[입력수단]/DO-R[시스템 입력장치]
```

```
<RV-Spec의 입부>
V10 : [입력 수단]/FR-R5 사용자가 자료를 입력하는 수단은?
    A. 대안 들:
       1. Keyboard/ATo PC등 대부분의 컴퓨터 기기
       2. Pen/ATo PDA
       3. Voice/ATo Hand Phone
    B. 선택 : 버전 1 = 1
```

3.3.2 다음 버전의 개발

첫 소프트웨어 버전을 개발하고 나서 다음 버전을 개발할 때에도 처음과 같이 요구 분석 과정을 거치게 된다. 이때 외부의 마케팅 계획과 사용자 의견뿐만 아니라 첫 버전을 만들 때 정리한 희망사항 목록 등을 기반으로 한다. 개발자는 첫 버전의 산출물인 R-Spec과 RV-Spec을 이용해서 수정 또는 첨가를 하게 된다. 첫 개발에서 많은 문제점들을 점검하고, 변화 가능성에 대해서 고려하여 RV-Spec으로 정리해 두었으므로, 비교적 적은 노력으로 요구사항을 만들 수 있다[4].

새 버전의 R-Spec의 작성은 새롭게 필요한 항목을 추가하거나, 첫 버전의 요구항목 중 적용되지 않는 것은 삭제하게 된다. RV-Spec의 개선은 추가되는 요구항목 중에서 변화 가능성이 있는 것들에 대한 분석 결과만 추가하면 되고, 기존 가변 항목에 대해서는 이전에 고려하지 못했던 변화 가능성이 발견되면 추가한다. R-Spec의 작성시 전 버전의 요구 항목과 같은 내용에 관한 것이지만 약간의 수정이 필요한 항목이 발견되면, 이는 가변항목이 되며, 두 버전의 차이점을 새로운 변경항목으로 만들어 RV-Spec에 추가한다. 설계 과정도 D-Spec과 DV-Spec을 이용하여 같은 방법으로 진행한다.

3.4 소프트웨어 군으로의 전환

초기 버전의 개발이후, 제품 군 방법을 적용하여 계속 개발하고 관리할 필요가 있다고 판단되면, 본격 군 개발을 위한 준비를 한다. 이때의 주요 활동은 다음과 같다.

① 제품 군 분석 : 전체 시스템 입장에서 다시 한번 제품 군에 속하는 멤버들의 공통점과 차이점을 검토한다.
② 군 문서 작성 : 기존 산출물로 부터 통합된 군 사양서를 작성하고, 그때까지 개선된 V-Spec을 보완하여 군 변경 사양서를 완성한다. 표준구조는 군 설계 사양서와 개선된 설계변경 사양서에 반영된다.
③ 컴포넌트 개선 : 군 설계 사양서에 따라 지금까지 개발된 컴포넌트를 이 구조에 맞게 재사용이 용이하도록 개선한다.

이 단계에서 일어나는 활동과 비용을 기존의 방법과 비교하면, ①과 ②는 통일된 형식과 항목으로 만들어진 사양서들이 있고 변경 가능성에 대한 분석을 정리해둔 변경 사양서가 있으므로, 기존의 제품 군 방법보다 매우 적은 노력으로 진행된다. 다만, 컴포넌트 개선 작업 ③은 기존 제품 군 방법과 동일하게 일어나야 하므로 노력도 비슷하게 소요된다.

3.4.1 군 문서들

한 제품 군에 속하는 여러 버전들을 유지 관리하기 위해, 각 시스템에 대한 사양서들을 각각 보유할 필요없이 제품 군 전체에 대해 하나의 군 사양서로 통합 관리한다. 즉 군 전체의 요구 사항과 설계 사항을 통합하여 군 요구 사양서

4) 이때 얼마만큼의 노력이 드는가 하는 것인 첫 버전과 둘째 버전의 차이 정도에 따라 다르다.

(FR-Spec)와 군 설계 사양서(FD-Spec)로 만들고, 그리고 이들의 가변 항목에 대한 변경 내용을 군 요구변경 사양서(FRV-Spec)와 군 설계변경 사양서(FDV-Spec)로 통합 관리한다. NISE 방법에서는, 설계 사양서와 설계 변경 사양서의 형식과 생성 및 개선 방법은 요구 사양서와 요구 변경 사양서의 그것과 동일하다. 따라서 이 논문에서는 기존 버전의 요구 사양서와 요구 변경 사양서로부터 군 요구 사양서와 군 요구변경 사양서를 생성하는 방법을 중심으로 군 전환 방법을 설명한다.

군 요구 사양서(FR-Spec)는 초기 버전들의 요구 사양서들($R\text{-}Spec_i$)을 통합해서 쉽게 만들 수 있다. 왜냐하면 후기 버전의 R-Spec은 이전 버전의 R-Spec을 기반으로 만들어 졌으며, 비슷한 항목은 수정 보완하고, 새로운 항목은 추가하였기 때문이다. 각 버전의 R-Spec들을 통합하게 되면 어떤 항목은 모든 멤버에 적용되고(이른 '공통적용 항목'이라 부른다) 어떤 것은 일부 멤버에만 적용된다(이를 '부분적용 항목'이라 부른다). 따라서, FR-Spec의 각 항목에 대하여 이 항목이 어떤 멤버에 적용되는 가를 명시해 두어 군 사양서로부터 각 멤버의 사양서를 쉽게 만들어 낼 수 있다.

군 요구 사양서(FR-Spec) 항목의 주요 내용은 아래에서 보는 바와 같이 각 버전의 R-Spec 항목과 같으며 다만 이 항목이 어떤 멤버에 적용되는 가를 나타내는 항목 ⑤이 추가되었다.

① 요구 사항 주제
② 요구 사항 설명
③ 타 항목과의 관계
④ 변경 항목 참조
⑤ 적용 멤버들 : 이 항목이 적용되는 멤버 이름 리스트

군 요구 변경 사양서(FRV-Spec)는 계속 개선되어 온 RV-Spec에다 군 분석 과정에서 발견된 변경 내용을 추가하고 FR-Spec과 일관성을 유지하도록 하면 된다. 각 변경 항목에서 한 시스템 버전이 어떤 대안 값을 선택 했는가는 RV-Spec이 개선되는 과정에서 이미 기록되어 있다.

두 개의 군 사양서로부터 우리는 각 멤버 사양서를 만들어 낼 수 있다. 예를 들어, FR-Spec과 FRV-Spec에서 첫 멤버의 R-Spec을 만들어 내기 위해서는 FR-Spec의 항목 중, '적용 멤버들' 항목 ⑤에 첫 멤버 이름이 들어있는 것만 골라서 모으면 된다. 이들 중에서 가변 항목에 대해서는, 그 항목에 표시된 FRV-Spec의 변경항목을 찾아가서 대안 값들 중 첫 멤버가 선택한 값을 찾아서 요구항목 표현을 수정하면 된다.

3.4.2 새로운 멤버 시스템의 개발

제품 군으로 전환이 이루어 진 후, 군에 속하는 새로운 멤버를 만들어 내는 방법은 기존의 제품 군 방법에서와 같은 방법으로 이루어진다. 즉 설계 사양서와 설계변경 사양서에 따라 컴포넌트들을 조립하여 개발한다. 다만 군 사양서를 중심으로 요구 분석 결과에 따라 새 멤버에 필요한 요구 사항을 추출해 내거나 추가하고, 군 문서들이 새 멤버에서 추가된 기능을 포함하도록 개선하는 작업이 필요하다.

4. 적용 예 : YBS(Year-Book System) 군 개발

이 절에서는 YBS(Year Book System) 제품 군의 개발을 예로 들어서 NISE 방법의 적용을 설명한다. 제안된 방법에 따라 군 개발을 수행하기 위하여서는 모든 문서들의 서술은 항목화(itemize) 되어 있어야 하고, 또한 각 문서들은 문법에 맞게 작성되어야 한다(문서들의 규격은 부록 참조). 이러한 제한은 각 문서 항목들간의 추적(trace)을 용이하게 하기 위함이다.

4.1 YBS 개념

YBS는 개인의 역사를 기록하여 두었다가 필요할 때 찾아 볼 수 있는 PC에서 동작하는 소프트웨어이다. 기본적인 요구 사항은 다음과 같다.

1. [목적] 나와 관련된 사건들을 기록하여 두었다가 필요할 때 찾아본다.
2. [사용자 수준] 사용자는 컴퓨터 전문가가 아닌 일반 사용자이다
3. [사용자 이력] 지금까지 주로 수첩이나 종이에 개인적인 일이나 사건들을 기록하던 사람들이 대부분이다. 따라서 사용하기 쉬워야 한다.
4. [운용 환경] Windows OS를 사용하는 PC에서 운용된다.

4.2 초기 버전의 개발

4.2.1 첫 버전 개발

주어진 YBS의 기본 개념으로부터 요구 분석을 실시하였다. 이때 R-Spec의 작성과 함께 변화 가능성이 있는 항목을 모두 기록하여 두었다. 다음에는 시스템 운영 시나리오(operational scenario)를 작성하면서 UI 설계를 수행하고, 소프트웨어 구조 설계와 상세 설계를 진행하여 운용 시나리오를 포함한 소프트웨어 설계서(D-Spec)를 작성하였다. 이때 제기된 문제들로부터 R-Spec을 보완하였다. 첫 멤버의 개발이므로 꼭 필요한 기능만을 우선 포함하도록 하였다.

일단 설계가 끝난 상태에서 지금까지 발견된 변화 가능성이 있는 항목을 분석하고, 변경 항목으로 정리하여 RV-Spec을 작성하였다. 이에 따르면, 대부분의 요구 사항에 대해서 변화 가능성이 있음을 발견하였고, 일부 항목에 대해서는 나중에 변경사항을 고려하기로 하고 분석과 설계를 완료하고, 구현에 들어갔다. 구현시에 발견한 문제점을 바탕으로 D-Spec과 DV-Spec을 보완하였다.

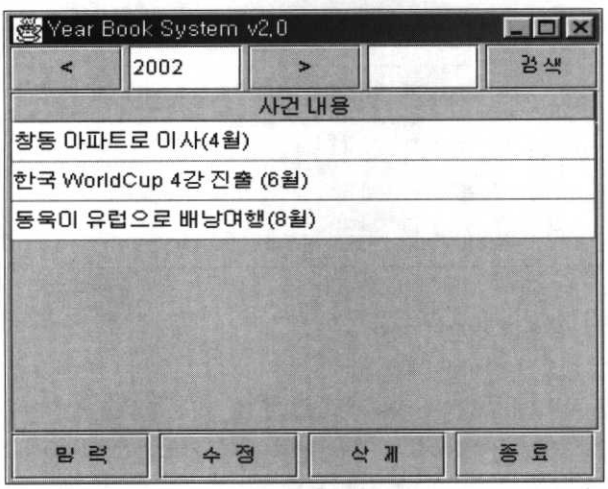
4.2.2 두 번째 버전의 개발

첫 버전의 개발을 끝내고, 다시 차기 멤버의 개발 계획을 세웠다. 사용 하면서 불편했던 점과 첫 버전의 사양서

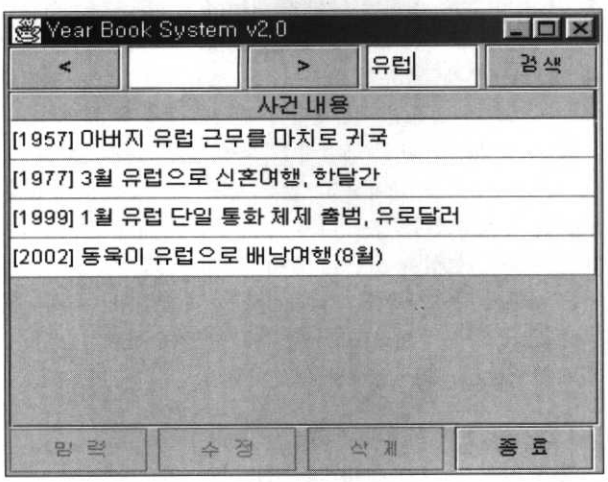
를 작성하면서 기록해 둔 '희망 사항 목록'을 중심으로 새로운 요구사항을 작성하였다. 두 번째 버전에 추가된 주요 기능은 다음과 같다.

- 입력시, 사건 기록이 들어갈 위치(그 해의 사건들 중 위치)를 지정하는 기능
- 한해에 같은 사건 기록이 들어가지 못하게 검사하는 기능
- 내용 검색 기능 : 기록된 사건을 연도 뿐만 아니라 그 내용으로도 찾아 볼 수 있도록 하는 기능

R-Spec 1을 기반으로 R-Spec 2를 작성하고, D-Spec 1에 근거하여 D-Spec 2를 작성하고 RV-Spec과 DV-Spec을 보완하였다. 두 번째 사양서들은 몇 가지 항목에서 추가 변경이 있었다. 특히, 내용 검색 기능이 추가 됨에 따라 D-Spec의 UI 부분에 주요 변화가 있었다. 아래 (그림 5)는 두 번째 버전의 주 화면을 보인 것이며, (그림 6)은 내용 검색을 시행한 화면을 보인 것이다.



(그림 5) YBS의 주 화면



(그림 6) YBS 시스템에서 내용 검색의 결과

4.3 군 문서의 작성

두 버전의 YBS를 개발한 후, 우리는 이 YBS를 체계적으로 발전시키기 위하여 소프트웨어 군 접근법을 본격적으로 도입하기로 결정하였다. 두 버전의 요구 사양서(R-Spec 1과 R-Spec 2)와 개선된 RV-Spec을 기반으로 군 문서 FR-Spec, FRV-Spec을 작성하였다. 그리고 제품 군의 특징을 다시 한번 모든 측면에서 검토하여 변화 가능성을 분석하여 위 문서를 보완하였다. 설계 문서에 대해서도 같은 작업을 수행하여 군 설계 문서 FD-Spec과 FDV-Spec을 작성하였다. 이때 설계 사양서와 설계변경 사양서에 기술된 내용에 맞게 컴포넌트들을 정리하였다.

4.3.1 YBS의 FR-Spec

(그림 7)은 YBS 시스템의 FR-Spec의 일부이다.

2. [사용자 수준] 사용자는 컴퓨터 전문가가 아닌 일반 사용자이다 /VA-V[사용자수준]/SEL-M1, M2
5. [운용 시스템] Windows OS를 사용하는 PC에서 운용된다 /VA-V[운용 환경]/SEL-M1, M2
6. [입력 장치] Mouse, keyboard/VA-V[입력 장치]/SEL-M1, M2
10. [기록 수단] 사용자는 Keyboard를 이용하여 사건을 기록한다 /DO-R[입력장치]/VA-V[입력수단]/SEL-M1, M2
11. [기록 위치] 새 사건을 입력할 때 사용자가 그 위치를 지정할 수 있다 /VA-V[사건 기록 위치]/SEL-M1, M2
13. [내용 검색] 기록 내용에 의해서도 사건 기록을 검색할 수 있다 /SEL-M2
14. [내용검색 방법] 사용자가 기록에 있으리라고 예상되는 텍스트 (글자 열)를 주면, 그 텍스트가 들어있는 사건 기록들을 모두 찾아서 보여 준다/DO-R[내용 검색]/SEL-M2

(그림 7) YBS F-Spec의 일부

변경항목 참조는 '/VA-변경항목 이름'으로 표시하고, 선택 멤버는 '/SEL-멤버이름 리스트'로 표시하였다. 각 요구항목의 설명에서 RV-Spec의 변경 값에 따라 변화되는 부분은 밑줄로 표시하였다. 이 '변경부분'이 다른 멤버들에게는 다른 선택 값으로 구체화(instantiation)되는 부분이다.

4.3.2 YBS의 FRV-Spec

(그림 8)은 (그림 7)에 있는 FR-Spec의 가변 항목들에 해당하는 변경항목을 보여 준다.

1. [사용자의 수준]/FR-R2 /Affects User Interface
4. [운용 환경]/FR-R5
   A. Choice :
      1. PC with Windows OS
      2. Server with Unix OS
      3. PDA with Palm OS
   B. Selection : M1 = 1, M2 = 1
5. [입력 장치] /FR-R6
   A. Choice : any reasonable combination of input devices :
      1. (1) Keyboard
      2. (2) Mouse
      3. (3) Pen
      4. (4) Mike
   B. Selection : M1 = 1 and 2, M2 = 1 and 2

8. [입력수단]/FR-R10 사용자가 입력하는 수단은?
   A. Choice :
      1. Keyboard
      2. Pen
   B. Selection : M1 = 1, M2 = 1
9. [사건 기록 위치] /FR-R11 새로운 사건을 넣는 위치는?
   A. Choice :
      1. 항상 그 해의 마지막에 기록됨
      2. 사용자가 기록시에 새로운 사건이 들어갈 위치를 지정할 수 있게 함
   B. Selection : M1 = 1, M2 = 2

Notation : M1/M2 = first/second member of YBS family

(그림 8) YBS의 FRV-Spec의 일부

4.4 개발 경험

NISE 방법을 적용하여 YBS 제품 군을 개발한 경험을 정리하면 다음과 같다.

① 요구 분석과 설계 과정은 계속 상세화(refinement)하면서 개발자는 계속 물음을 갖게 되고, 그 해답으로서 여러 가지 가능성을 검토하게 된다. 이것들을 기록하여 각각 RV-Spec, DV-Spec을 만든다.

② 첫 멤버의 개발시 많은 변화 가능성들을 인지하게 되고, 그 분석기록도 상당부분 진행된다. 얼마만큼 깊게 분석하는가 하는 것은 첫 버전의 분석과 설계 단계에서 얼마큼 시간을 사용할 수 있는가에 달렸다.

③ 요구 사항은 요구 분석 후 일단 정리 되지만, 운영 시나리오 작성 및 UI 설계, 시스템 설계시에도 미처 발견하지 못 했던 문제점들이 나타나고 이에 따라 이미 만들어진 R-Spec과 RV-Spec이 수정 보완된다.

④ R-Spec 1으로부터 R-Spec 2를 만드는 일, RV-Spec을 개선하는 일, 두 개의 R-Spec으로부터 FR-Spec을 만드는 것, 그리고 FR-Spec과 최종 RV-Spec으로부터 FRV-Spec을 만드는 일들은 거의 기계적인 작업이었다. 설계 과정에 대해서도 마찬가지이다. 어려운 점은 요구 분석 및 설계와 같은 개발 과정에 있었다.

⑤ 각 문서들의 추적성을 표현하기 위해서 사용하는 참조에는 항목 번호 보다는 이름을 사용하는 것이 더 편리했다.

⑥ 실제로 요구 사양서의 항목 중에는 불변 항목 보다는 가변 항목이 더 많다. 그러나 설계 사양서의 경우에는 가변항목이 더 적었다. 제품의 기본 개념이 같으면 소프트웨어 구조에는 큰 차이가 없다.

4.6 분 석

NISE 방법의 핵심은 초기 버전을 개발할 때부터 변화를 분석하고, 이를 V-Spec을 이용하여 효과적으로 관리하는데 있다. 즉, 각 개발 단계에서 변화 가능성을 인지하고, 그 대안들을 분석하여 V-Spec에 기록하고 보완해 나간다. 이때 V-Spec과 표준 문서(예 : R-Spec)와의 관계를 명확히 표시하였고, 통합된 군 요구 사양서와 변경 사양서로부터 각 멤버의 요구 사양서를 만들어 낼 수 있도록 하였다.

4.6.1 경제성 분석

우리가 제안한 NISE 방법의 경제성을 기 제안된 제품 군 방법과 단품 개발방법과 비교 분석한다. 모두 N개의 제품을 개발할 때, 첫 k개의 초기 버전을 개발한 후 본격 군 전환 단계를 거친 후 나머지를 개발한다고 가정하자. 또한, $C0$ = 단품 개발 방법의 평균 개발비, $C1$ = 기존 제품 군 방법의 평균 개발비, 그리고 A_F = 기존 제품 군 개발방법에서의 초기 투자비, A_N = NISE 방법에서의 군 전환비라 하자. 여기서 $C1$은 $C0$보다 아주 작다. 단품 개발 방법의 총 비용 T_S, 기존 제품 군 개발방법에서의 총 비용 T_F, NISE 방법에서의 총 비용 T_N은 다음과 같이 표시된다.

$$T_S = N * C0$$

$$T_F = A_F + N * C1$$

$$T_N = \sum_{i=1}^{k}(c_i + d_i) + A_F + \sum_{i=k+1}^{N} C1$$

$$= \sum_{i=1}^{k} c_i + \sum_{i=1}^{k} d_i + A_N + (N-k) * C1$$

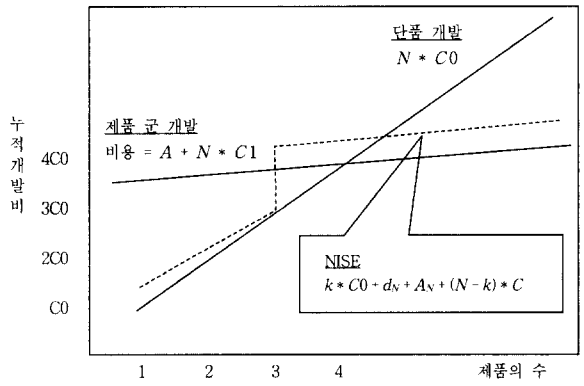
이 식에서 T_N은 $k=0$이면 $A_N = A_F$가 되어 T_F와 같아지고, $k=N$이면 $A_N=0$이 되므로 T_S와 같아진다. NISE 방법에서 초기 버전의 개발은 단품 개발비 (c_i)와 변경 사양서 작성비용 (d_i)으로 구성된다. 이때 개발은 단품 개발처럼 진행되므로 c_i는 $C0$와 비슷하다. 변경사양서 작성비용 (d_i)만이 추가로 소요된다. 이는 일반 개발비 (c_i)에 비해서 매우 작다. 따라서 두 번째 항은 비교적 작은 값이다 (이하 이를 d_N으로 표현한다). 따라서 T_N은 다음과 같이 표현할 수 있다.

$$T_N = k * C0 + d_N + A_N + (N-k) * C1$$

여기서, 제품 군 전환 비용 (A_N)은 기존 방법의 초기 투자비 (A_F) 보다 적다. 왜냐하면, 군 요구사양서와 군 설계 사양서 그리고 그들의 변경사양서가 간단히 만들어지고, 이들 기록에는 많은 변경 사항들이 기록되어 있으며, 소위 표준 구조의 내용도 설계사양서와 그 변경 사양서에 들어 있기 때문이다. 추가 비용은 미래 제품들에 대한 변화 분석과 기존 컴포넌트들의 개선에 필요한 비용 뿐이다. A_N은 A_F의 반 또는 1/3정도로 예상된다.

초기 버전 k개의 개발비의 경우에는 $T_S = k * C0$, $T_F = A_F + C1$, $T_N = k * C0 + d_N$이 된다. (그림 2)에서 보는 바와 같이 기존 제품 군 방법에서 투자비 회수기간은 4(멤버)이고, A_F는 $3 \sim 4 * C0$ 정도이다. 이렇게 가정했을 때 $C1/$

$C0 \cong 0.25$가 된다. 따라서 T_N은 T_S보다 조금 크고, T_F보다는 작게 된다. (그림 9)는 NISE 방법의 경제성을 다른 방법과 비교한 것을 보여준다.



(그림 9) NISE 방법의 경제성 분석

초기에 k개의 제품을 개발하고 제품 군으로 전환 후 모두 $N-k$개의 제품을 개발했을 때 총 개발비는 다음과 같이 표현된다.

$$T_S = N * C0$$
$$T_F = k * C1 + A_F + (N-k) * C1$$
$$T_N = k * C0 + d_N + A_N + (N-k) * C1$$

T_N을 T_S와 비교하면 제품 군 전환 직후에는 약 A_N 만큼 크지만 얼마 후에는 작아지고 그 이후에는 그 차이가 급속히 벌어진다($C1$이 $C0$보다 아주 작으므로). T_N과 T_F를 비교하면 제품 군 전환 후 약 A_N 만큼 크지만 그 차이는 계속 유지된다(같은 기울기 $C1$으로 증가하므로). (그림 9)는 $k=3$일 때 T_N을 표시하였다.

결론적으로 NISE 방법은 초기 버전 개발 단계에서는 단품 개발 보다는 조금 비용이 더 들고, 기존 제품 군 방법보다는 매우 적게 든다. 그리고 제품 전환 단계 이후에는 단품 개발 방법보다 비용이 아주 적고 기존 제품 군 개발방법 보다는 A_N 만큼 크고, 그 차이는 계속 유지된다. 다시 말하면, 군 전환 단계에서 얼마간의 투자를 한 후에는 전체 비용은 단품 개발 방법보다는 아주 적고, 제품 군 방법보다는 조금 많음을 보여준다. 여러 멤버의 개발 후 누적 개발비는 NISE 방법이 기존 제품 군 방법들보다 비용이 조금 많이 들지만, 초기 개발 도중에 일어날 수 있는 제품 군의 취소로 인한 투자비 손실 위험은 매우 낮다.

4.6.2 적용성 분석

NISE 방법은 모든 문서(요구 사양서, 설계사양서)들이 항목화(itemize) 되어있는 것을 가정하였고, 변경 부분에 관하여서는 텍스트를 파라메타화하고, 각 버전에 따라 구체화(instantiate)하는 방법을 택했다. 따라서 이는 소프트웨어 제품의 영역에 관계없이 적용할 수 있다. 현재까지 연구된 결과로는 요구 사양서와 설계 사양서들이 텍스트(서술) 형식으로 작성하였을 때 가장 효과적으로 적용할 수 있다. 만약, 문서가 텍스트뿐만 아니라 그래픽 등을 포함하는 항목이 있는 경우에는 변경 부분만이 아니고 전체 항목을 파라메타화해야 하므로 변경내용의 서술 측면에서 정밀성이 떨어지는 점이 있다.

5. 결 론

우리는 이 논문에서, 비슷한 소프트웨어 시스템들의 군을 개발함에 있어, 초기 비용을 적게 들이면서 궁극적으로 제품 군 개발 방법을 조직에 적용할 수 있는 점진적인 NISE 방법을 제안하였다. 이 방법은 초기 버전을 개발할 때부터 변화 가능성에 관한 정보를 기록하여 차기 버전의 개발에 활용하였고, 이는 본격 제품 군 개발로의 전환을 쉽게 해준다. 우리는 핵심 자산인 요구 사양서와 설계 사양서 그리고 이들의 변경 사양서의 생성과 개선 기법을 중심으로 프로세스와 산출물을 제시하였으며, 간단한 응용 시스템 YBS 시스템 군의 개발을 예로 그 적용을 보였다. 또한 우리가 제안한 방법을 개발비용 측면에서 기존의 방법들과 비교하여 그 효율성을 보였고, 적용성 측면에서의 문제점을 분석하였다.

NISE 방법을 적용하여 개발할 때에는 초기 버전의 개발은 단일 제품의 개발처럼 진행한다. 다만 개발과정에서 당연히 고려되는 여러 가지 문제점과 대안(변화 가능 값)들을 기록하고 정리하여 차기 개발에 활용한다. 몇 개의 버전을 개발한 후에 군 개발 접근법을 채택하기로 결정할 경우, 기존 버전의 산출물과 변경 사양서들로부터 군 문서들을 만든다. 이때 추가적인 변화 분석을 수행하고, 설계 사양서와 설계변경 사양서에 맞게 컴포넌트들을 정리하여, 본격적인 제품 군 개발을 준비한다.

이 논문의 공헌은, 개발 과정에서 고려되는 여러 가지 질문 그리고 그 대안들을 정리하여 변화 가능성에 대한 사양서를 만들고 이것을 다음 멤버 개발에 이용하는 일련의 과정에서, 버전의 차이점들을 효과적으로 기록 발전시키고 여러 버전들을 개발한 후에 본격적인 제품 군 접근법으로 쉽게 전이할 수 있도록 하는 방법을 보인 것이다.

개발 비용측면에서는 초기 버전의 개발단계에서는 단품 개발방법보다는 다소 추가 비용이 들고 기존 제품 군 개발 방법보다는 비용이 적게 든다. 또한, 제품 군으로 전환 이후에는 기존 제품 군 개발방법 보다는 조금 많이 소요되지만 단품 개발방법 보다는 적게 들고, 많은 멤버를 개발한 이후에는 기존 제품 군 방법과 거의 비슷한 효과를 낸다. 따라서 이 방법은 제품의 초기 버전을 개발할 때 관련 소프트웨어 개발 경험이 없거나 초기에 많은 투자를 할 수 없을 경우 또는 첫 버전이 빨리 출시 되어야 할 경우 효과

적으로 적용할 수 있는 개발 방법이다.

한편, NISE 방법은 기존 제품 군 개발 방법들과는 달리 개발 프로세스 전 과정을 지원하는 통합 환경과 지원 도구가 개발되지 않았으므로 각 단계를 수동으로 처리해야 하는 단점이 있다. 우리는 향후, 이 방법을 여러 개발 과제에 적용하여 보완하고 지원 도구를 개발할 예정이다.

참 고 문 헌

[1] Mark A. Ardis, David A. Cuka, "Defining Families-Commonality Analysis," Proceedings of 21st International Conference on Software Engineering, pp.671- 672, May, 1999.

[2] Joachim Bayer, et. al, "PuLSE : A Methodology to Develop Software Product Lines," Proceedings of the 5th symposium on software reusability (SSR 99), pp.122-131, May, 1999.

[3] Yu Chye Cheong and Stanislaw Jarzabek, "Frame-based method for customizing generic software architectures," Proceedings of the 5th symposium on software reusability (SSR 99), pp.103-112, May, 1999.

[4] Paul Clements and Linda Northrop, "Software Product Lines : Practices and Patterns," Addison-Wesley, 2002.

[5] James Coplien, Daniel Hoffman, and David Weiss, "Commonality and Variability in Software Engineering," IEEE Software, Vol.15, No.6, pp.37-45, November/December, 1998.

[6] Ari Jaaksi, "Developing Mobile Browser in a Product Line," IEEE Software, Vol.19, No.4, pp.73-80, July/August, 2002.

[7] 이재준, 강교철, "프로덕트 라인 소프트웨어 개발 프로세스," 정보과학회지, 제20권 제3호, pp.23-30, 2002.

[8] K. C. Kang, J. Lee and P. Donohoe, "Feature-Oriented Product Line Engineering," IEEE Software, Vol.19, No.4, pp. 58-65, July/August, 2002.

[9] W. Lam, J. A. McDermid and A. J. Vickers, "Ten Steps Towards Systematic Requirements Reuse," Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, pp.6-15, 1977.

[10] Frank Linden, "Software Product Families in Europe : The Escaps & Caf Projects," IEEE Software, Vol.19, No.4, pp. 41-49, July/August, 2002.

[11] Randall R. Macala, et. al, "Managing Domain-Specific, Product-Line Development," IEEE Software, Vol.13, No.3, pp. 57-67, May, 1996.

[12] John McGregor, et. al, "Initiating Software Product Lines," IEEE Software, Vol.19, No.4, pp.24-27, July/August, 2002.

[13] John Mylopoulos, et. al, "Exploring Alternatives during Rquirements Analysis," IEEE Software, Vol.18, No.1, pp. 92-96, January/February, 2001.

[14] Linda M. Northrop, "SEI's Software Product Line Tenets," IEEE Software, Vol.19, No.4, pp.32-40, July/August, 2002.

[15] David L. Parnas, "On the design and development of program families," IEEE Trans. on Software Engineering, SE-2, No.1, pp.1-9, March, 1976.

[16] David L. Parnas, "Designing software for ease of extension and contraction," Proceedings on the 3rd international conference on Software Engineering, pp.264-277, May, 1978.

[17] Klaus Schmid and Martin Verlage, "The Economic Impact of Product Line Adoption and Evolution," IEEE Software, Vol.19, No.4, pp.50-57, July/August, 2002.

[18] Steffen Thiel and Andreas Hein, "Modeling and Using Product Line Variability in Automotive Systems," IEEE Software, Vol.19, No.4, pp.66-72, July/August, 2002.

[19] Weiss, D. M., "Software Product-Line Engineering," Adisson-Wesley, 1999.

부록 A : 군 요구 사양서(FR-Spec)

A.1 요구 사양서 항목의 주요 내용

- 항목 번호
- 요구사항 주제
- 요구사항 설명
- 타 요구 항목과의 관계 : 주로 어떤 항목이 이 항목에 영향을 미치는가를 표시
- 변경 항목 참조 : 가변 항목의 경우 V-Spec의 해당 변경 항목을 가리킴
- 적용 멤버들 : 이 항목이 적용되는 군 멤버 이름들

A.2 요구 항목의 Syntax(Extended BNF Notation)

```
<r-specification item> ::= <requirement item> | <comment item>
<requirement item> ::= <r-number> <r-subject> <r-description>
        <dependence relation> <variation relation> <applied to>
        <comment item>
<r-subject> ::= [ <subject name> ]
<dependence relation> ::= /DO-<ri-reference>
<ri-reference> ::= R<r-number> | R<r-subject>
<variation relation> ::= /VA-<vi-reference>
<vi-reference> ::= V<number> | V [ <variation name> ]
<applied to> ::= /SEL-<member list>
<member list> ::= <member name> {, <member name> }*
<member name> ::= text
<comment item> ::= <observation item> | <note item>
<observation item> ::= /OBS text
<note item> ::= /NOTE text | /N text

참고: 정의되지 않은 non-terminal 모두 text 또는 number 임.
```

부록 B : 군 요구 변경 사양서(FRV-Spec)

B.1 변경 사양서 항목의 주요 내용

- 항목 번호
- 변경 사항 이름

- 요구 항목 참조 : 이 변화가 발생한 요구 사양서 항목
- 변화 내용 제목 또는 보충 설명
- 가능한 대안들
- 멤버 선택 값 : 각 멤버가 어떤 값을 선택하였는가를 표시함

B.2 변경 항목의 Syntax(Extended BNF Notation)

```
<rv-specification item> ::= <variation item>
<variation item> ::= <v-number> <v-name> <r-spec-reference>
        <variation spec> { <relation> }*
<v-number> ::= number
<v-name> ::= [ <variation name> ]
<variation name> ::= text
<r-spec reference> ::= /FR-<ri-reference>
<ri-reference> ::= R<r-number> | R [ <r-subject> ]
<variation spec> ::= { <v-title> } <v-values> <selections>
<v-title> ::= text
<v-values> ::= Choice: { <value type> | <a-item desc list> }
<v-type> ::= Type = <primitive type name>
<a-item desc> ::= <a-value> {<a-info>}*
<a-value> ::= text | <primitive type value> | <yes-no>
<yes-no> ::= YES | NO
<a-info> ::= <application info> | <note item>
<application info> ::= /AppliedTo text
<relation> ::= /DO-<si reference> | /SEE <si-reference> |
 /Affects text
<si-reference> ::= <ri-reference> | <vi-reference>
<selections> ::= Selection : <a-selection> {, <a-selection>}*
<a-selection> ::= <member-name> = <number>
```


주 복 규

e-mail : bkjoo@wow.hongik.ac.kr
1977년 서울대학교 계산통계학과 졸업
1980년 한국과학원 전산학과(이학석사)
1990년 University of Maryland(공학박사)
1990년~1998년 삼성종합기술원, 삼성전자 중앙연구소 수석연구원
1998년~2000년 동양시스템즈 연구소장
2000년~2001년 한국과학기술원 전산학과 초빙교수
2001년~현재 홍익대학교 전자전기컴퓨터공학부 교수
관심분야 : Software Reuse, Software Testing, Mobile IP, Intelligent Systems

김 영 철

e-mail : bob@wow.hongik.ac.kr
1985년 홍익대학교 전산학과 졸업
1987년 광운대학교 전산학과(이학석사)
2000년 Illinois Institute of Technology (이학박사)
2000년~2001년 LG 산전 중앙연구소 책임연구원
2001년~현재 홍익대학교 전자전기컴퓨터공학부 교수
관심분야 : Use Case 방법론 및 툴 개발, Design driven Testing, Maturity models, 데이터 모델링, Information Architecture