공통성 및 가변성 분석을 활용한 컴포넌트 설계 기법

(A Method to Design Components using Commonality and Variability Analysis)

장 수 호 † 김 수 동 ††

(Soo Ho Chang) (Soo Dong Kim)

요 약 컴포넌트 기반 소프트웨어 개발 (CBD) 기술은 재사용 가능한 컴포넌트를 조합하여 효율적으로 소프트웨어를 개발함으로써 개발 노력과 상품화 시간을 줄여주는 새로운 기술로 정착되고 있다. 이러한 CBD 컴포넌트는 한 도메인의 표준이나 공통적인 기능을 제공하여야 재사용성이 높아진다. 특히, 공통성 안의 미세한 가변적인 부분도 모델링하고, 이 가변성을 각 어플리케이션의 특성에 적합하게 특화 할 수 있도록 설계되어야 한다.

기존의 CBD 방법론에서도 이 중요성이 강조되고 있지만, 체계적이며 구체적인 개발 프로세스, 적용 지침 및 산출물 양식의 제공이 미흡하여, 도메인 컴포넌트의 개발은 비체계적인 프로세스와 개발자의 경험에 의존해 왔다. 본 논문은 컴포넌트 설계를 위한 체계적인 프로세스와 기법을 제안한다. 이 프로세스는 여러 단계와 활동으로 구성되며, 각 활동에 대한 세부 지침과 표준 양식도 포함하여 보다 효과적인 컴포넌트 개발을 도모한다. 제안된 기법의 실효성 검증을 위하여 금융 도메인에 적용한 사례연구를 제시하며, 다른 기법들과의 비교 평가도 다룬다. CBD의 공통 컴포넌트 개발에 제안된 프로세스와 지침의 사용함으로써 보다 재사용성과 적용성이 높은 컴포넌트가 비용 및 시간에 있어서 효율적으로 개발될 것으로 기대된다.

키워드 : 도메인 컴포넌트, 컴포넌트 프레임웍, 공통성, 가변성

Abstract Component-based software development (CBD) technology has been widely accepted as a new effective paradigm for building software systems with reusable components, consequently reducing efforts and shortening time-to-market. Hence, components should provide standard or common functionalities in a domain, yielding a higher level of reusability. Especially, micro-level variability within the commonality should also be modeled so that a product member-specific business logic or requirement can be supported through component tailoring or customization.

The importance of commonality and variability (C&V) analysis has been emphasized in several CBD methods, but they lack of well-defined systematic process, detailed instructions, and standard artifact templates. As the result, the development of components has been carried out in ad-hoc fashion, depending on developer's experience. In this paper, we propose a systematic process and work instructions to design components. The process consists of phases and their activities and each activity is specified with detailed instructions and artifact templates in order to facilitate effective development of components. To verify a feasibility of the propose method, a case study in a banking domain and comparison and assessment between the proposed method and other methods are additionally provided. With proposed processes and instructions, reusability and efficiency of developing components can be better supported.

Key words : Domain component, Component Framework, Commonality, Variability

· 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음

† 학생회원 : 숭실대학교 컴퓨터학과
shchang@otlab.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
sdkim@comp.ssu.ac.kr

논문접수 : 2003년 7월 22일
심사완료 : 2004년 3월 26일

1. 서 론

CBD 기술은 재사용 가능한 컴포넌트를 조합하여 효율적으로 소프트웨어를 개발함으로써 개발 노력과 상품화 시간을 줄여주는 새로운 기술로 정착되고 있다. 컴포넌트는 한 어플리케이션에서의 재사용 보다는 여러 어

플리케이션이나 여러 회사들 간의 재사용을 목표로 하는 단위이다. 따라서, 컴포넌트는 한 도메인의 표준이나 공통적인 기능 즉, 공통성을 제공하여야 재사용성과 적용성이 높아진다.

가변성(Variabilty)이란 공통성안에서 각 어플리케이션이나 회사(이하 멤버)별로 요구되는 세부적인 차이점을 의미한다[1,2]. 가변성은 가변점(Variation Point)과 가변치(Variant)로 표현될 수 있다. 가변점이란 가변성이 발생하는 위치를 의미하며 주로 기능단위로 나타난다. 가변치란 가변점에서 멤버가 실제로 갖게 되는 값 자체를 의미한다. 이러한 가변성은 잘 분석되어 컴포넌트에 반영되어 있어야 하며, 컴포넌트에 설계되어 있는 가변성을 각 어플리케이션의 특성에 적합하게 특화할 수 있도록 설계되어야 재사용성이 향상된다[3].

기존의 CBD 방법론 및 문헌에서 공통성 및 가변성의 중요성이 강조되어 왔으며, 특히 제품 계열 공학(Product-Line Engineering)에서는 이 과정을 중요한 활동으로 포함한다. 그러나, 기존의 연구에서는 도메인 컴포넌트를 설계하는 전 과정에 걸친 체계적이며 구체적인 개발 프로세스, 적용 지침 및 산출물 양식의 제공이 미흡하여, 도메인 컴포넌트의 개발은 비체계적인 프로세스와 개발자의 경험에 크게 의존해 왔다.

본 논문에서는 공통성 및 가변성이 내장된 컴포넌트 설계를 위해 요구사항 정의 단계부터 컴포넌트 생성단계까지 전반적인 단계에 걸친 체계적인 프로세스와 기법을 제안한다. 이 프로세스는 여러 단계(Phase)들로 구성되며 각 단계는 여러 활동(Activity)으로 구성되는데, 각 활동에 대한 세부 지침과 표준 양식을 제공하여 보다 효과적인 컴포넌트 개발을 도모한다.

2장에서는 공통성 및 가변성 모델링을 위한 전체적인 프로세스를 제안하며 네 단계의 프로세스와 그에 대한 지침, 산출물들에 대한 제안이 3장부터 6장에 걸쳐 나타난다. 또한 7장에서는 제안된 프로세스를 이용한 은행 도메인에서의 사례연구를 보이고, 8장에서는 본 연구에서 제안된 기법과 관련된 기법들을 여러 기준 항목으로 비교한다.

2. 관련연구

Atkinson의 KobrA 방법론은 엔터프라이즈 모델, 구조적 모델, 엑티비티 모델, 상호작용 모델, 결정 모델을 사용하여 프레임웍 공학에 있어서의 가변성을 명시한다[1]. 각 모델에서는 ≪variants≫라는 확장형을 사용하여 가변점을 표현하였으며 특히 결정모델에서는 비즈니스 프로세스와 그 외의 다양한 다이어그램에 있는 가변점들이 상세화 되어있다. 그러나 이 연구에서는 가변성의 존재여부를 결정할 수 있는 기준, 가변치와 가변성의 범위[4]를 식별하는 기준이 상세히 제시되어 있지 않다.

COMO방법론은 도메인 분석단계부터 공통 기능성을 추출하여 컴포넌트를 생성하는 클러스터링 알고리즘, 컴포넌트 설계, 명세의 전반적인 프로세스와 지침을 제안하고 있다[5]. 이 방법론은 가변성의 형태를 속성, 로직, 워크플로우로 나누어, 패밀리(family) 요구사항 명세서에서 가변점과 가변치들을 식별하는 기술도 제안한다. 그러나 공통성 및 가변성 식별 기준과 식별된 가변성을 컴포넌트에 투영하여 설계하는 프로세스는 포함되어 있지 않다.

Griss의 연구는 공통성과 가변성을 도출하기 위해 휘처(feature)모델을 제시하며, 이를 이용하는 4-단계의 프로세스를 제안하고 있다[2]. 이에 더하여 여러 컴포넌트에 분산되어 있는 크로스커팅 기능(Crosscutting Feature)에 대한 해결 방안도 제시하고 있다 그러나, 이 연구는 각 프로세스에 대한 상세한 지침과 산출물이 추가될 수 있다.

Catalysis방법론은 도메인 분석의 비즈니스 모델로부터 컴포넌트의 명세, 내부 구현 명세, 코드의 컴포넌트 구현 과정에서 산출물간의 추적이 가능하며 특히 프로세스패턴을 도입하여 각 프로세스에 대한 패턴을 정의하고 패턴별 적용 지침들을 마련하였다[6]. 그러나, 공통 요구사항이 가용한 것을 가정으로 컴포넌트 식별을 진행하므로, 공통성 및 가변성 분석 활동의 지침과 산출물 표준이 제공되지 않는다.

3. 컴포넌트의 설계

그림 1과 같이 컴포넌트를 설계하는 전반적인 절차가 네 단계로 구성된다. 첫번째 단계인 '요구사항 정규화(Normalization) 단계'에서는 멤버들로부터 요구사항을 수집하며 공통 용어를 식별하고 식별된 공통용어를 사용하여 각 멤버들에 대한 요구사항을 재작성한다. 이 단계를 통하여 여러 멤버들의 이질성을 가진 요구사항이 정규화되므로 요구사항의 비교가 용이하게 되며 이에 따라 공통성 및 가변성이 효과적으로 추출될 수 있다. 그러므로, 요구사항 정규화 단계는 다음 단계를 진행하기 위한 필수적인 단계가 된다. 두번째 단계인 '공통성 식별 단계'는 정규화된 요구사항들을 비교하여 그 안의 공통적인 기능들을 식별하는 단계이다. 공통기능들이 식별되면 컴포넌트를 모델링 할 수 있는 패밀리 요구사항 명세서가 생성될 수 있다. 세번째 단계인 '가변성 모델링'단계는 가변점과 가변치를 설계하는데, 가변점을 설계하기 위해 가변점을 식별하는 식별 기준을 제시하며 식별된 가변점과 가변치를 표현하는 가변성 테이블을 제안한다. 네번째 단계인 '컴포넌트 모델링'단계에서는 식별된 공통기능을 클러스터링하여 초기 컴포넌트를 생

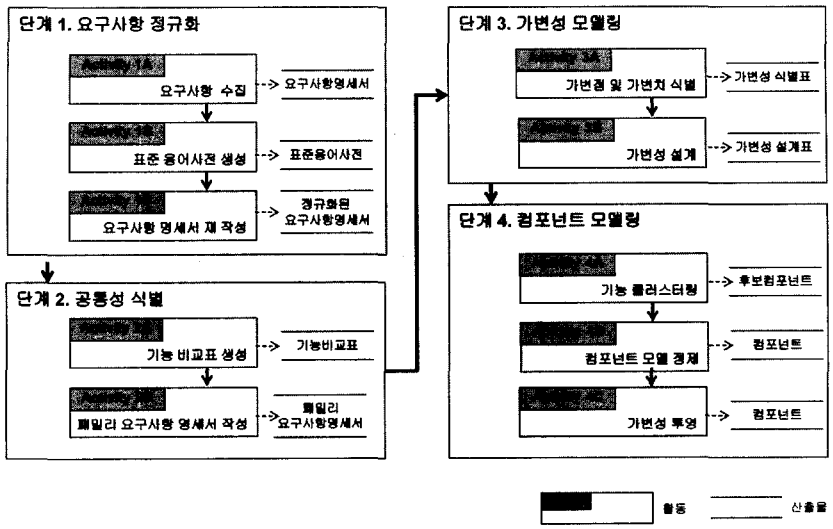

그림 1 컴포넌트 모델링 프로세스

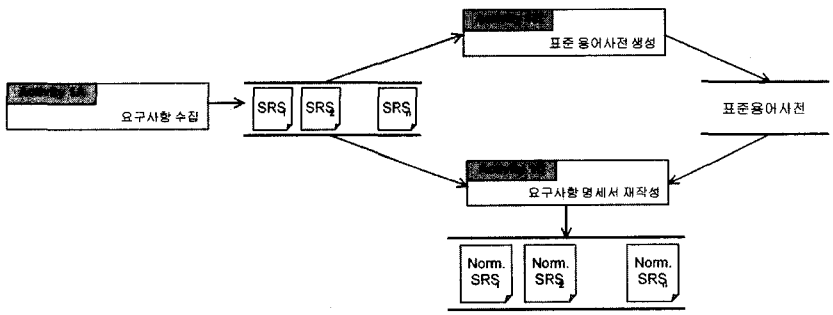

그림 2 요구사항 정규화 프로세스 및 산출물

성한다. 또한 가변점과 가변치들이 각 컴포넌트에 내장되며 이 가변성들을 다루기 위한 *Required* 인터페이스가 설계된다.

3.1 요구사항 정규화

요구사항 정규화 단계는 그림 2에서와 같이 요구사항 수집, 표준 용어사전 생성, 요구사항 명세서 재 작성의 세가지 활동으로 구성된다.

3.1.1 요구사항 수집

컴포넌트는 재사용성 향상을 위해 여러 멤버들의 요구사항이 반영되어야 한다. 이를 위해 Activity1A에서는 멤버들의 요구사항을 수집하며 이는 그림 2에서 SRS_i로 나타난다. 소프트웨어의 요구사항은 크게 기능적 요구사항과 비기능적 요구사항으로 구분된다[7]. 본 논문에서는 기능적인 요구사항을 중심으로 작성된 요구사항 명세서를 확보한다.

3.1.2 표준 용어사전 생성

공통성 식별의 어려운 점은 여러 멤버들로부터 수집된 "요구사항 명세서"들간의 이질성이 있다는 것이다. "요구사항 명세서"들간의 이질성이란 멤버가 요구사항 명세서에서 사용한 용어나 개념이 다른 멤버의 사용과 비교해 의미가 다르게 나타나는 성질을 의미한다. 이는 한 용어가 여러 멤버들 사이에서 다른 의미로 쓰이는 경우 또는 여러 멤버들이 사용하는 여러 다른 용어들이 한 의미로 쓰이는 경우이다. 그러나 멤버들간의 이질성 문제로 다양한 요구사항들의 비교가 불가능하거나 비능률적일 수 있다. 그러므로, 멤버들의 요구사항 명세서들은 표준이나 공통으로 사용될 수 있는 용어와 개념으로 표현되어야 한다.

멤버별 요구사항 명세서로부터 '표준용어사전'이 생성된다. 용이하게 표준 용어를 식별하기 위해 표 1의 용어 비교표를 사용한다. 이 표에서, 요구사항 명세서 들로부터 추출된 유사용어들 즉, T(1,1), T(2,1), … T(n,2)들

표 1 용어 비교표

컴포넌트를 사용하는 멤버 / 업무영역	M_1	M_2	···	M_n	표준 용어
	T(1,1)	T(2,1)	···	T(n,2)	CT_1
	T(1,2)	T(2,3)	···	T(n,4)	CT_2
	:	:	...	:	:
	T(1,j)	T(2,k)	···	T(n,l)	CT_n

표 2 표준 용어 사전

표준용어	정의
CT_1	
CT_2	
:	:

이 한 행에 채워지고, 공통으로 사용되거나 표준이 될 수 있는 용어들이 식별된다.

용어비교표가 채워지면, 표 2와 같이 CT_i을 추출하여 표준 용어사전을 생성한다. 이 표는 공통 용어와 함께 용어에 대한 정의를 제공하며 뒤이어 나타날 여러 활동들에서 참조된다.

3.1.3 요구사항 명세서 재작성

'요구사항 명세서 재작성'은 표준 용어사전의 용어들을 이용하여 멤버들의 요구사항 명세서를 재작성하는 활동이다. 이 과정을 통해 개정된 요구사항 명세서들은 표준 또는 공통 용어를 사용하게 된다. 멤버들의 요구사항 명세서에 사용된 용어들이 비교적 정규화되어 있거나 또는 그 이질성이 크지 않다면 이 과정은 생략될 수 있다. 그림 2의 Norm.SRSi는 재작성된 요구사항 명세서 즉, 정규화된 요구사항 명세서를 의미한다.

3.2 공통성 식별

이 단계에서는 정규화된 요구사항간의 비교를 통해 공통기능을 도출하며, 이 정보는 컴포넌트의 범위를 결정하는데 사용된다.

3.2.1 기능 비교표 생성

도메인 컴포넌트를 사용하게 될 'n'개의 멤버들에 대해, 공통성으로 추출될 수 있는 기능들을 표 3의 양식을 이용하여 비교한다.

표 3 기능 비교 표

요구사항	멤버				공통성 정도	적용된 규칙	공통기능 (Y/N)
	M_1	M_2	···	M_n			
F_1	√	√		√			
F_2		√		√			
···	√	√	√				
F_m	√		√	√			

첫번째 열은 F_1, F_2, ..., F_m의 모든 기능의 집합을 나타낸다. 이 집합은 초기에 모든 요구사항의 합집합으로 나타난다. 이들 중 어떤 기능은 멤버들간의 공통 기능일 수 있고 어떤 기능은 특정 멤버에게만 속한 기능일 수 있다. 이러한 기능을 추출할 때의 어려움은 각기 다른 요구사항 명세서에서 추출되는 기능의 크기(Granularity)를 맞춰야 한다는 것이다. 이러한 기능의 크기를 맞추는 작업은 도메인 전문가에 의해서 해결될 수 있다.

멤버들을 나타내는 열은 기능 F_i가 어떤 멤버 M_j에 적용되는지를 나타낸다. 만약 M_j가 F_i를 필요로 하면 해당 위치에 체크 표시를 한다. '공통성정도'를 나타내는 열에서는 (체크된 멤버수)/(전체 멤버수)를 간단한 계산 공식으로 제안한다. 도메인에서 특정 멤버의 요구사항은 다른 멤버들의 요구사항보다 더 중요할 수 있으므로, 이 계산 값은 주어진 기능에 대해 공통성의 대략적인 정도를 나타낸다.

'적용된 규칙'을 나타내는 열에서는 각 기능이 공통성의 집합에 포함될지 여부를 결정하는데 적용되는 몇 가지 규칙을 필요로 한다. 즉, 공통성 정도, 각 멤버의 영향력, 재정적 투자와 같은 스폰서쉽, 표준화 정도 등 몇 가지의 요소들이 고려될 수 있다. 공통성을 추출하기 위하여 이러한 요소들이 적용된 몇 가지 지침을 다음과 같이 제안한다.

지침 1. 만약 기능의 공통성 정도가 100%이면 그 기능은 공통성에 포함된다.

지침 2. 만약 기능의 공통성 정도가 100%에 가깝고, 그 기능을 필요로 하는 멤버들 중 한 멤버가 도메인의 표준화나 컴포넌트 소비자로서의 영향력을 가진 멤버라면, 그 기능은 공통성에 포함된다. 컴포넌트 소비자는 구매하거나 획득할 수 있는 컴포넌트를 이용하여, 어플리케이션을 개발하려는 고객이나 개발자를 의미한다.

지침 3. 만약 기능의 공통성 정도가 다른 기능들에 비해 낮지만, 그 기능을 필요로 하는 멤버가 도메인에서 중요 멤버(Key Player)이면서 컴포넌트 개발을 후원(Sponsor)한다면 신중한 검토를 통하여 공통성에 포함될 수 있다.

지침 4. 지침 2와 지침 3사이에 있는 그 외의 경우는 시장성과 같은 비즈니스 이슈나 멤버의 영향력, 도메인 지식을 고려하여 판단한다.

지침 5. 위의 조건에 만족하지 않지만 기능이 도메인에서 필수적이거나 표준 기능성을 가진다면 그 기능은 도메인 전문가의 신중한 평가를 통해 공통성에 포함될 수 있다.

마지막 열인 '공통기능(Y/N)' 열은 기능의 공통성 포함 여부의 결정을 표시하는 열이다. 기능의 공통성 포함 여부는 대부분 제시된 규칙을 기초로 하여 판단되며 그

외의 비즈니스 요소나 도메인 규약들도 고려될 수 있다.

3.2.2 패밀리 요구사항 명세서 작성

기능비교를 통하여 표 4와 같이 공통 기능을 나타내는 공통성 명세표를 작성한다.

표 4 공통성 명세 표

기능 ID	기능 명	설명
CF_1		
CF_2		
...		

이 표를 이용하여 다음 단계에서 컴포넌트를 개발하는데 참조될 패밀리 요구사항 명세서를 작성한다. 또한 이 정보는 컴포넌트 구매자에 제공되어 컴포넌트에 어떤 기능이 제공되는지를 알게 하여 컴포넌트의 사용여부를 판단할 수 있게 한다.

3.3 가변성 모델링

식별된 공통 기능에는 로직이나 워크플로우 또는 사용되는 속성에서 멤버들간의 약간씩 다른 가변성을 가지고 있는데, 이를 공통성 내의 가변성이라 한다. 컴포넌트 개발에서 가변성을 잘 구현하면 컴포넌트의 적용성과 재사용성을 증가 시킬 수 있다[3].

3.3.1 가변점 및 가변치 식별

가변점은 다른 멤버들간에 가변성이 발생할 장소를 식별 한다. 이를 효과적으로 모델링 하기 위해 표 5와 같은 가변성 식별표를 사용한다.

두번째 열에서는 가변점의 타입이 명시된다. 로직은 프로그램의 명령문에 대응되는 스텝들로 구성된다. 따라서 로직에 나타나는 가변성은 시스템이나 비즈니스 오퍼레이션의 알고리즘이 다름을 의미한다. 또한 워크플로우는 전형적으로 메시지 호출의 시퀀스이므로, 워크플로우에서의 가변성은 시스템이나 비즈니스 오퍼레이션의 메시지 전송의 흐름이 다름을 의미한다. 예를 들어 은행 도메인에서 대출을 평가하여 승인하는 프로세스에서 다른 워크플로우가 존재할 수 있다. 기능이 사용하는 데이타로써의 속성도 사용하는 멤버에 따라 그 타입이나 범위가 다를 수 있다. 예를 들어 은행 도메인에서 고객의 ID의 경우 한 은행에서는 영문과 숫자의 조합으로 문자형 타입을 사용할 수 있으면 다른 은행의 경우는 주민등록번호와 같은 정보를 사용하여 숫자 타입을 사용할 수 있다.

멤버의 집합 M_1, M_2, ..., M_n 열에서는 공통 기능 CF_i의 가변점에서 가변치들이 식별되고 명세 된다. 가변치 $V_{i,j}$는 공통 기능 CF_i의 j번째 가변치를 의미한다. 첫 행의 CF_1에서 M_1의 가변치는 $V_{1,1}$로 명세된다. 만약 M_2의 가변치가 $V_{1,1}$과 같지 않다면 M_2는 새로운 가변치의 ID, 즉 $V_{1,2}$를 갖게 된다. 이어서 M_3의 경우는 M_1과 같은 가변치를 갖게 되므로 $V_{1,1}$으로 표시된다. 이런 내용을 반복하여 각 가변점의 가능한 가변치들이 표현되게 된다.

가변치를 모델링하는데 있어서 한 문제는 멤버 사이에 가변성이 존재하는지의 여부를 인식하는 것이다. 다시 말해, $V_{i,p}$와 $V_{i,q}$가 서로 다른가를 찾아내는 것이다. 이를 위한 비교 요소로서 사후조건(Post Condition), 입력범위(Input Domain), 출력범위(Output Range), 구현 알고리즘 등이 있다. 사후조건이란 기능이 실행된 후 만족해야 할 시스템의 제약 조건을 의미하여 데이타 상태의 변화를 예로 들 수 있다. 입력범위란 도메인에서 업무적 의미를 가지는 값의 범위를 의미한다. 예를 들어 한 은행에서 신규계좌번호를 생성하기 위한 입력 값이 A은행은 고객ID이고 B은행은 지점ID라면 두 값은 문자열로 같은 타입을 가지지만 입력되는 값은 도메인에서 업무적으로 다른 의미를 가진다. 출력범위란 기능이 실행된 후 리턴하는 값의 타입 및 범위를 의미한다. 예를 들어 계좌ID를 생성하기 위해, A은행에서는 숫자 타입을 사용하고, B은행에서는 문자열 14자리를 사용하며, C은행에서는 문자열 11자리를 사용한다면 이는 출력범위가 서로 다른 것이다. 이러한 요소를 기초로 가변성 결정 지침을 아래와 같이 제안한다.

지침 1. 만약 두 멤버 사이의 CF_i의 사후조건이 다르면, 가변성이 존재한다.

지침 2. 만약 두 멤버 사이의 CF_i의 입력범위나 출력범위가 다르면, 가변성이 존재한다.

지침 3. 만약 CF_i의 구현 알고리즘이 가변성 모델링

표 5 가변성 식별 표

공통기능	가변점타입	멤버					가변성 범위
		M_1	M_2	M_3	...	M_n	
CF_1	Logic	$V_{1,1}$	$V_{1,2}$	$V_{1,1}$		$V_{1,1}$	2
CF_2							
CF_3	Workflow	$V_{3,1}$	Open	$V_{3,2}$		$V_{3,2}$	2+Open
CF_4	Attribute	$V_{4,1}$	$V_{4,1}$	$V_{4,2}$		$V_{4,2}$	2
...							
CF_m	Logic	Open	Open			Open	Open

단계에서 결정될 수 있고 두 멤버 사이의 알고리즘이 다르다면, 가변성이 존재한다.

지침 4. 만약 두 멤버 사이의 CF_i가 사용하는 데이타의 타입이나 범위가 다르면, 가변성이 존재한다.

지침 5. 만약 위의 규칙의 어떤 항목도 CF_i에 적용되지 않는다면, 사전조건(Pre-condition), 불변성(Invariants), 의미(Semantic) 등이 비교하기 위한 요소가 될 수 있다.

마지막 열인 '가변성 범위'는 제안된 규칙에 의해 식별된 가변치의 전체 수를 나타낸다. 만약 가변치의 수가 1이면 해당 공통 기능에는 가변성이 존재하지 않는다. 그러나 만약 그 수가 2 이상이면, 해당 공통 기능 CF_i는 가변점이 되고 표시된 수 만큼의 가변치를 갖게 된다. 만약 가변점의 가변치가 알려지지 않았으면, 즉, Open이면 범위는 'Open'이라 표시한다. 만약 어떤 가변치는 알려져 있고 어떤 가변치는 알려지지 않았으면, '알려진 가변치의 수 + Open'이라 표시한다.

3.3.2 가변성 설계

가변성 식별 표에서 얻어진 정보로, 표 6과 같이 각 가변점에 가변치들이 어떻게 구현되는지를 명세한다.

표 6 가변성 설계표

기능의 가변점	가변성 타입	가변치집합	Open/ Closed	초기값	비고
CF_1	Logic	{$V_{1.1}$, $V_{1.2}$}	Closed	$V_{1.1}$	
CF_3	Workflow	{$V_{3.1}$, $V_{3.2}$}	Open	$V_{3.2}$	
CF_4	Attribute	{$V_{4.1}$, $V_{4.2}$}	Closed	$V_{4.2}$	
...					
CF_m	Logic	{ }	Open	None	

'가변치 집합'열은 식별된 모든 가변치들을 나타낸다. 가변치들은 가변성 타입에 따라 여러가지 형태로 표현될 수 있는데 그 타입이 로직일 경우는 로직에서 사용되는 서로 다른 계산 공식 등이 기술될 수 있고, 워크플로우일 경우는 워크플로우가 실행되는 서로 다른 스텝이 기술될 수 있으며 애트리뷰트일 경우는 실제로 달라지는 데이타 자체가 기술될 수 있다. 또한 가변치는 도메인 용어를 사용하여 간결하게 기술할 수도 있다. 'Open/Closed'열은 식별된 가변치들이 모든 발생 가능한 가변치를 의미하는지(Closed) 또는 추가 가능성이 있는지(Open)를 나타낸다.

3.4 컴포넌트 모델링

3.4.1 기능 클러스터링

공통성 및 가변성 모델을 기초로 개략적 컴포넌트가 설계된다. 컴포넌트를 식별하기 위해 그림 3과 같이 관련된 기능을 컴포넌트 단위로 그룹핑하는 지침을 아래와 같이 제안한다. 두 기능 CF_i와 CF_j가 다음과 같은 조건을 만족하면 두 기능은 같은 컴포넌트에 할당될 서로 관련이 있는 기능들이다.

지침 1. 만약 기능 CF_i와 CF_j가 고객이 정의한 같은 기능군에 속하게 되면 CF_i와 CF_j는 서로 관련된 기능이다. 기능군이란 시스템, 모델, 기능적 분류, 배치 분류 등에 기초하여 그룹 지어진 단위를 의미한다. 일반적으로 고객은 도메인에 많은 지식을 가지며 그 도메인 지식을 기초한 기능적 분류 체계를 가진다. 따라서 만약 고객이 두 기능을 같은 기능군으로 묶는다면 이 기능은 서로 관련된 기능이다.

지침 2. 만약 기능 CF_i와 CF_j가 같은 데이타나 정보를 사용한다면 CF_i와 CF_j는 서로 관련된 기능이다. 한 기능은 고객이 요구하는 작은 단위의 기능을 가지며 이 기능을 수행하기 위해 데이타를 사용한다. 만약 두 기능이 거의 또는 완전히 같은 데이타를 사용한다면 이 두 기능은 한 컴포넌트로 그룹 지어질 수 있다.

지침 3. 만약 두 기능 CF_i와 CF_j 사이에 높은 의존도가 있다면 이 두 기능은 서로 관련된 기능이다. 객체지향과 CBD에서 의존도란 메시지 호출의 관계가 있음, 즉 ≪usage≫ 관계의 의존도를 의미한다[8,9]. 컴포넌트 간의 결합도를 줄이기 위해 의존도가 있는 두 기능은 한 컴포넌트에 들어가야 한다.

지침 4. 만약 두 기능 CF_i와 CF_j가 시스템 레이어에 속하고 그들이 관련된 시스템 오퍼레이션이나 트랜잭션을 수행한다면 이 두 기능은 서로 관련이 있다. 일반적으로 시스템 오퍼레이션을 수행하는 기능은 시스템 레이어에 속한다. 그리고 그러한 기능들은 한 컴포넌트로 그룹지어져야 한다. 따라서 이 두 기능은 영구적인 데이타를 다루는 기능들과는 구분되어야 한다.

지침 5. 만약 두 기능 CF_i와 CF_j가 비즈니스 레이어에 속하고 그들이 영구적인 데이타나 객체를 다루면 이 두 기능은 서로 관련이 있다. 지침 4와 반대로 이 두 기능은 비즈니스 레이어에 속하므로 시스템 오퍼레이션이나 트랜잭션을 수행하는 기능들과는 구별되어야 한다.

의존도가 높은 기능의 클러스터링을 메트릭을 이용하여 수행하는 기법으로는 위의 적용 기준과 병행하여 COMO[5] 기법을 사용할 수 있다.

3.4.2 컴포넌트 모델 정제

이전 활동에서의 클러스터링 규칙은 그룹핑하는 것이 도메인 지식에 크게 의존하고, 예외사항도 많이 존재하므로 정제작업이 필요하다. 제안된 규칙을 적용하여 기능을 컴포넌트로 그룹핑 할 때 그림 3의 CF_3와 같이 어느 컴포넌트에도 할당되지 않은 기능이 생기거나 CF_2와 같이 두 개 이상의 컴포넌트에 할당된 기능이 생길 수 있다.

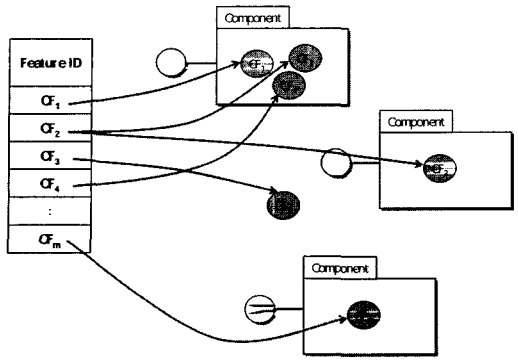

그림 3 기능 클러스터링

어떤 컴포넌트에도 할당되지 않은 기능의 경우는 그 기능과 가장 가까운 기능을 가진 컴포넌트에 할당되거나 또는 할당되지 않은 기능의 수에 따라 유틸리티 컴포넌트로 그룹핑 할 수 있다. 만약 여러 개의 컴포넌트에 나타나는 기능이라면 아래와 같은 지침을 적용할 수 있다.

지침 1. 기능이 데이타를 수정하기 보다는 조회하는 기능이라면 이 기능은 편리성이나 효율성을 위해 여러 개 컴포넌트에 기능을 할당한다.

지침 2. 기능이 조회보다는 데이타를 수정하는 기능이라면 그 기능을 집중적으로 이용하는 한 컴포넌트에 할당할 수 있다. 그리도 다른 컴포넌트에서는 기능이 할당된 컴포넌트의 인터페이스를 통해서 기능을 수행한다.

지침 3. 만약 기능이 지침 2의 특징을 가졌으나 그 기능이 할당될 컴포넌트를 찾는 것이 적합하지 않으면 그러한 기능들은 공통 컴포넌트로 그룹핑 할 수 있다. 또한 다른 컴포넌트는 이 공통 컴포넌트의 인터페이스를 통해 기능을 수행 한다.

3.4.3 가변성 투영

공통 기능들이 컴포넌트 단위로 그룹되면 표 6의 가변성 정보를 컴포넌트에 투영(Projection)시켜야 한다. 그림 4에서 하나의 컴포넌트는 내부에 공통 기능 CF_1, CF_3를 가지고 있으며 컴포넌트의 기능을 제공하기 위해 **P**로 표현된 *Provide* 인터페이스와 커스터마이징을 위해 **R**로 표현된 *Requied* 인터페이스를 가진다. 그리고 가변점에 대해 가변치를 적용하는 것은 표현된 인터페이스 중 *Required* 인터페이스를 통하여 수행된다.

컴포넌트를 한 멤버의 애플리케이션에 맞게 커스터마이징 하는 것은 컴포넌트의 인터페이스를 통해 이루어지지만 컴포넌트로부터 서비스를 얻기 위해 메시지를 호출하는 것과 다른 의미를 갖는다. 컴포넌트 커스터마이징은 일반적으로 컴포넌트를 설치 시점에 한번 발생한다. 그러나 메시지 호출은 애플리케이션 실행시 컴포넌트의 기능을 사용할 때마다 빈번하게 발생한다. 따라서 커스터마이징을 통한 가변치 설정의 결과는 컴포넌트 내에 영구적으로 남아 있어야 하나, 메시지 호출을 통한 데이타는 임시적이다.

가변점의 범위에 따라 사용되는 *Required* 인터페이스의 오퍼레이션은 *Select()*와 *PlugIn()*을 들 수 있다. 가변점이 'Closed'범위를 갖는다면 가변치 세팅을 위해 *Select()*메소드가 사용며, 컴포넌트 내에 이미 내장된 가변치 중 애플리케이션이 요구하는 가변치가 어떤 것인지를 입력 받아 영구적으로 설정한다. 만약 가변점의 범위가 'Open'이면 *PlugIn()*메소드를 사용하여 커스터마이징 한다. 이 메소드는 외부 함수나 객체, 컴포넌트를 참조하여 직접 가변치를 입력한다. 또한 가변점의 범위에 구체적인 가변치가 존재하고 확장가능성이 있다면, 즉 부분적으로 Open이라면 *Select()*메소드와 *PlugIn()* 메소드를 동시에 사용할 수 있다.

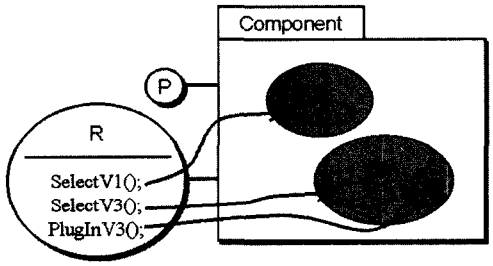

그림 4 가변성 투영

그림 4에서와 같이, CF_i은 Closed 범위의 로직 가변점을 가진다. 따라서 *Required 인터페이스*에서 CF_i의 가변치 설정을 위해 *SelectV1()* 오퍼레이션을 사용하며 그 메소드는 컴포넌트 내에 구현 된다. FC_3의 경우에는 워크플로우 가변점을 Open범위로 가지며 두개의 가변치 $V_{3,1}$, $V_{3,2}$도 함께 가진다. 따라서 가변성 V3를 커스터마이징 하기 위해서 *SelectV3()*와 *PlugInV3()*를 함께 사용한다.

4. 사례 연구

이 장에서는 제안된 프로세스와 산출물을 은행 도메인에 적용한 사례를 보여주며, 이는 CIP 프로젝트[10]의 일환으로 세 개의 기업이 공통으로 수행한 과제의 결과이다.

4.1 단계 1. 요구사항 정규화

• Activity 1A. 요구사항 명세서 수집

프로젝트에 참여하게 된 은행은 광주(KJ)은행, 국민(KM)은행, 조흥(CH)은행이다. 사례연구에서의 기능영역은 '고객관리', '계좌관리'로 국한한다. 요구사항 명세

표 7 은행 도메인에서의 용어 비교 표

멤버 / 업무영역	KJ	KM	CH	공통용어
고객관리	Add New Customer	Register New Customer	Register Customer	Register Customer
	Update Customer	Update Customer	Modify Customer	Update Customer
	Inquire Customer	Inquire Customer	Inquire Customer Info.	Inquire Customer
		Close Customer	Inactivate Customer	Inactivate Customer
...	...	...	...	...

표 8 기능 비교표

항목 / 업무영역	기능	멤버			공통성 정도	공통성 추출 적용 규칙	공통기능 (Y/N)
		KJ	KM	CH			
Customer Management (CM)	Register Customer	√	√	√	1	지침1	Y
	Update Customer	√	√	√	1	지침1	Y
	Inquire Customer	√	√	√	1	지침1	Y
	Inactivate Customer		√	√	2/3	지침4	Y
	Register Accident	√			1/3		N
	Modify Accident	√			1/3		N
	Evaluate Customer		√		1/3	지침3	Y
...	...	...	...	...	...	...	...

서는 지면의 한계로 나타내지 않는다.

• Activity 1B. 표준 용어 사전 생성

사용하는 용어에 있어서 세 은행의 요구사항 명세서에서는 표현이나 기능 분류에 있어서 서로간의 이질성이 나타났다. 따라서 표 7과 같이 공통 용어를 도출하기 위해 용어 비교표를 생성한다.

표 7로부터 표준용어사전을 만들 수 있으며 이 표는 뒤이어 나오는 여러 프로세스에 사용될 용어의 공통된 정의를 제공한다.

• Activity 1C. 요구사항 명세서 재작성

표준용어사전을 사용하여 멤버들의 요구사항 명세서를 재 작성한다. 표준 용어를 사용하여 정의된 요구사항 명세서이므로 공통성 및 가변성을 찾기가 용이하다. 그러나 멤버들의 요구사항 명세서의 개념과 용어가 비교적 일치하면 이 작업은 생략될 수 있다.

4.2 단계 2. 공통성 식별

• Activity 2A. 기능 비교표 생성

컴포넌트 내에 구현되어야 할 공통 기능을 식별하기 위해 멤버들의 기능을 비교해야 한다. 표 8에서, 두번째 열은 멤버들에게서 발견될 수 있는 모든 기능의 집합을 나타낸다.

마지막 열인 '공통기능(Y/N)'는 각 기능의 공통성 포함여부를 나타내며 결정은 Activity2A에 제안된 규칙에 근거한다. 그리고 근거가 되는 규칙은 공통기능 이전 열인 '적용규칙'에 나타낸다. 예를 들어 Inactivate Customer은 공통성 정도를 2/3를 가진다. 그러나 은행 도메인에서 필수적이고 표준 도메인 업무를 따르는 기능이므로 Section 3.2.의 공통성 추출을 위한 지침4에 근거하여 공통성에 포함된다.

• Activity 2B. 패밀리 요구사항 명세서 작성

표준용어사전을 이용하여 멤버들간의 공통된 기능을 포함하는 패밀리 요구사항 명세서를 작성한다.

4.3 단계 3. 가변성 모델링

• Activity 3A. 가변점 및 가변치 식별

단계 2의 '공통성 식별'을 통해 식별된 공통 기능들로부터 표 9를 만들어 가변점과 가변치를 식별한다. 첫번째 열인 '기능'에는 멤버들의 모든 기능 집합을 나타내며 이 기능 중에는 가변성을 가진 기능도 있고 가변성을 가지지 않은 기능도 있다. 가변성을 가진 기능에 대해서만 두번째 열인 '가변성 타입'에 Logic, Workflow 또는 Attribute를 표시한다. 따라서 Update Customer의 경우에는 가변성을 가지지 않으므로 가변성 타입 및 기타 정보를 표시하지 않는다.

EvaluateLoanApplication의 경우 KM은 신청된 대출을 평가하는 워크 프로우가 알려져 있으나 KJ나 CH의 경우에는 워크플로우가 알려지지 않아 가변치를 설정하는 시점이 컴포넌트의 배치 시점으로 미루어지게 된다. 마지막 열인 '가변성 범위'의 경우 Register Customer의 로직 가변성이나 애트리뷰트 가변성의 가변치들은 2개로 이미 알려져 있으므로 그 범위가 '2'가 된다. Computer Interest의 경우에는 KJ와 KM의 경우에는 이자계산방식이 복리계산과 단리계산으로 알려져 있으나 CH의 경우는 아직 명확하게 나타나지 않았으므로 Open이 되고 따라서 가변성의 범위는 2+Open이 된다.

표 9 가변성 식별표

기능 \ 항목	가변성 타입	멤버			가변성범위
		KJ	KM	CH	
Register Customer	Logic	RRN	RRN	UnitId+Serial	2
	Attribute	Number	Number	String	2
Update Customer					
...	...	...	...	...	...
Evaluate Loan Application	Workflow	Open	Evaluate Customer and Account Balance	Open	Open
...	...	...	...	...	...
Compute Interest	Logic	Compound	Simple Interest	Open	2+Open

표 10 가변성 설계 표

가변점	가변성타입	가변치 집합	Open orClosed	초기값	비고
Register Customer	Logic	{RRN, UnitId+Serial}	Closed	RRN	Customer ID
	Attribute	{Number, String}	Close	String	Customer ID
Evaluate Customer	Workflow	{Evaluate Customer and Account Balance}	Open	None	
...	...	...	...	...	...
Compute Interest	Logic	{Compound,Simple, Open}	2+Open	Compound Interest	

• Activity 3B. 가변치 설계

가변점 및 가변치 식별이 끝나면 표 10과 같이 가변성을 가진 기능들만 수집하여 가변성 설계 표를 생성한다. 이 표에서, 새로운 고객을 등록하기 위한 CustomerID 생성 로직에서는 주민등록번호를 초기값으로 한다. 가변성 설계 표는 컴포넌트내부에 가변성을 투영하는 단계에서 Required 인터페이스 설계 시 다시 이용된다.

4.4 단계 4. 컴포넌트 모델링

• Activity 4A. 기능 클러스터링

Section 3.4.1에서 제안된 기능 클러스터링을 위한 규칙중 지침 1이 기능을 그룹핑하는 규칙으로 가장 많이 사용되었으며 그 결과는 그림 5와 같이 고객관리, 수신관리, 여신관리 컴포넌트가 있다. 사용자 요구사항으로부터 추출된 기능들은 시스템외부에서 시스템을 통해 얻고자 하는 기능이다. 따라서 그러한 기능들을 통해 얻어진 컴포넌트는 비즈니스 컴포넌트보다는 시스템 컴포넌트가 대부분이다.

• Activity 4B. 컴포넌트 모델 정제

기능적으로 독립적인 세 부분의 기능영역이 나누어지므로 언급된 기능의 할당 문제는 들어나지 않는다. 정제하는 단계에서의 두번째 역할은 비즈니스 레이어의 컴포넌트를 정의하는 것이다. 비즈니스 컴포넌트는 사용자 요구사항으로부터 도출된 객체 모델(Object Model)의 클래스를 그룹핑하여 만들어지는 영구적인 데이타를 관리하는 컴포넌트이다. 본 사례연구에서 그 결과는 그림 5의 CustomerComponent, AccountComonent로 도출될 수 있으며 그 상세한 프로세스는 본 논문에서는 다루지 않는다.

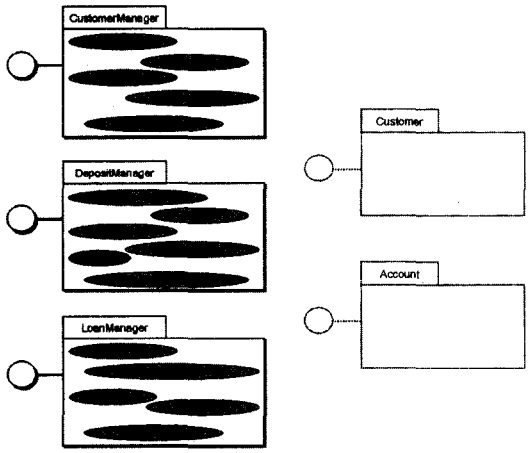

그림 5 컴포넌트 모델 정제

• Activity 4C. 가변성 투영

가변성 설계표의 각 가변점들은 가변성의 범위에 따라 *Select()* 또는 *PlugIn()*의 *Required 인터페이스*를 갖게 된다. 본 사례연구에서는 그림 6과 같이 6개의 가변점이 시스템 컴포넌트에 투영되었다. Customer-Manage 컴포넌트에는 RegisterCustomer, Evaluate-Custom등 두개의 가변성을 가진 기능이 있으며 각각 커스터마이징을 위한 Required인터페이스의 오퍼레이션들이 있다.

가변점의 구현기술은 사용되어질 프로그래밍 언어나 미들웨어에 따라 여러 형태로 나타날 수 있다. 개발 플랫폼에 따른 가변성의 상세 설계는 별도의 연구로 수행될 수 있다.

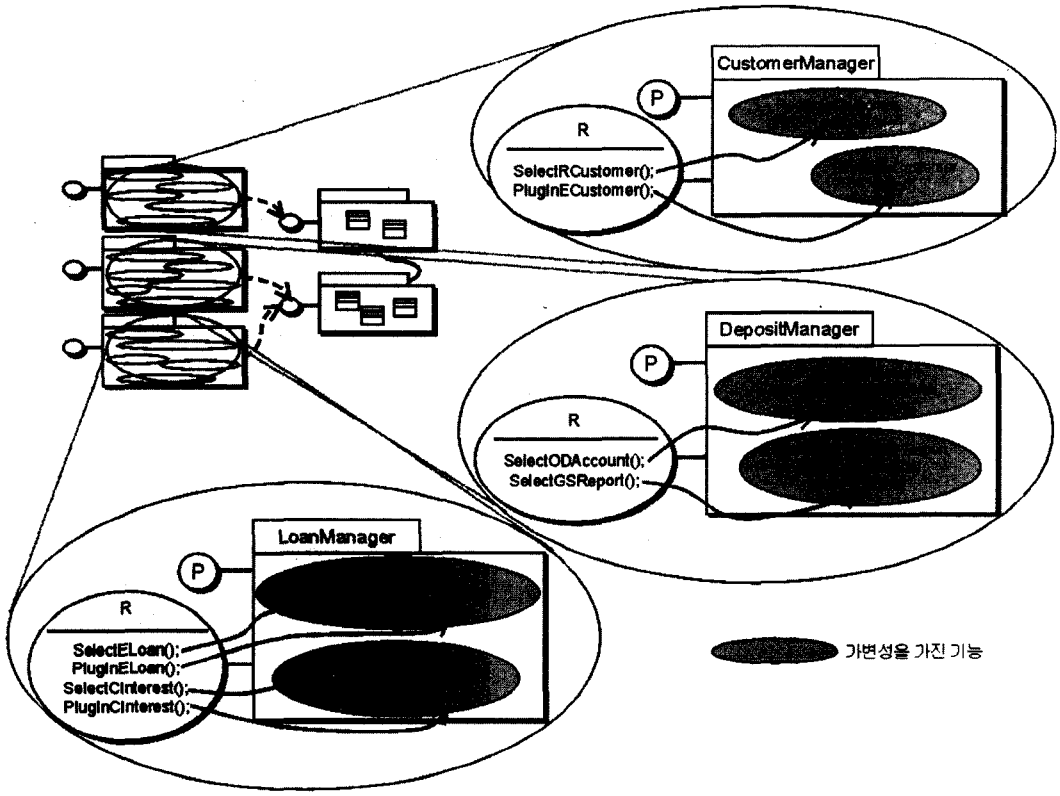

그림 6 컴포넌트내의 가변성

5. 평가

제안된 프로세스를 평가하기 위한 항목은 다음의 기준으로 선정되었다.

- 프로세스 내 활동들의 순서 논리성: 개발 프로세스의 여러 활동간의 순서가 논리적이며, 활동간 큰 격(Gap)이 없는지, 같은 수준의 활동으로 제안되었는지, 활동의 크기에 따라 구조적으로 제안되어 상위 활동에 대해 세부 활동이 명시 되었는지에 대한 평가 기준
- 각 활동에 대한 상세지침 제공: 각 활동을 수행하기 위한 세부적인 기준과 지침이 제공되는지의 평가 기준
- 각 활동별 적용 예제 제공: 각 활동의 지침을 적용한 사례를 제공함으로써 제안된 개발 프로세스의 이해를 높여 적용성을 향상시키는 지의 평가 기준
- 각 활동별 산출물 내용과 양식: 각 활동에 대한 최적화된 산출물 양식을 제공함으로써 산출물 생성에 경제성을 제공하는지의 평가 기준

이상의 비교 항목으로 표 11에 Griss [2]등 네 개의 대표적인 기법을 비교한다. 비교한 기법들의 대부분은 개발 프로세스를 제시하고 있으나, 각 활동에 대한 세부 지침은 미비하고, 상용 개발방법론이 갖추어야 할 산출물에 대한 정의가 미흡한 것으로 나타났다.

표 11 개발 프로세스 비교 평가

기법 / 비교 항목	Griss	COMO	Catalysis	KobrA	본 논문의 기법
프로세스 내 활동들의 순서 논리성	△		○	△	○
각 활동에 대한 상세지침 제공	X	△	△	X	○
각 활동별 적용 예제 제공	△	△	△	△	△
각 활동별 산출물 내용과 양식	X	△	X	○	○

○ : 제공, △: 부분 제공, X: 미흡

컴포넌트를 개발하는 프로세스의 활동들에 대해 다음의 항목으로 비교 한다.

- 공통성 식별 지침: 재사용성을 위해서 컴포넌트는 기능의 공통성을 가져야 한다. 따라서 컴포넌트를 설계하는 전체 프로세스에서 공통성을 식별할 수 있는 프로세스와 지침은 포함되어야 한다.
- 가변성 식별 지침: 여러 도메인에서 재사용되기 위해

컴포넌트는 가변성을 지원하는 기능이 포함되어있어야 한다. 이를 위한 단계로서 가변성을 식별하는 프로세스 및 지침이 제공되어야 한다.

- 낮은 상호의존성을 갖는 컴포넌트 식별 지침 제공: 기능군으로써 생성된 컴포넌트는 다른 컴포넌트의 기능에 상호 의존성이 적어야 한다. 따라서 컴포넌트 단위 추출시 기능군을 만드는 기준이 제시되어야 한다.
- 가변성의 컴포넌트 투영 지침: 식별된 가변성을 종류에 따라 컴포넌트에 투영하는 기법이 제공됨으로써 가변성이 설계된 컴포넌트가 생성될 수 있다.

표 12 컴포넌트 설계 지침의 비교 평가

기법 / 비교 항목	Griss	COMO	Catalysis	KobrA	본 논문의 기법
공통성 식별 지침	△	○	△	△	○
가변성 식별 지침	X	△	X	△	○
낮은 상호의존성을 갖는 컴포넌트 식별 지침 제공	△	○	○	△	○
가변성의 컴포넌트투영 지침	X	X	X	△	○

○ : 제공, △: 부분 제공, X: 미흡

CBD 컴포넌트의 용도를 고려하면, 공통성과 가변성 모델링이 핵심 활동이지만 비교된 기법들에서 가변성의 분석과 이를 컴포넌트 설계에 투영하는 과정이 크게 미흡한 것으로 조사되었다.

6. 결 론

컴포넌트 기반 소프트웨어 개발 기술은 재사용 가능한 컴포넌트를 조합하여 효율적으로 소프트웨어를 개발함으로써 개발 노력과 상품화 시간을 줄여주는 새로운 기술로 정착되고 있다. CBD 컴포넌트는 한 어플리케이션에서의 재사용 보다는 여러 어플리케이션이나 여러 회사들 간의 재사용을 목표로 하는 단위이다. 따라서, CBD 컴포넌트는 한 도메인의 표준이나 공통적인 기능을 제공하여야 재사용성과 적용성이 높아진다. 공통성안의 세부적인 차이점인 가변성 또한, 잘 분석되고 컴포넌트에 반영되어 있어야 하며, 컴포넌트에 설계되어 있는 가변성을 각 어플리케이션의 특성에 적합하게 특화할 수 있도록 설계되어야 재사용성이 향상된다.

본 논문에서는 도메인 컴포넌트 설계를 위한 체계적인 프로세스를 제안하였다. 이 프로세스는 네 개의 단계로 구성되고, 전체적으로 10개의 활동으로 구성되는데, 각 활동에 대한 세부 지침을 예제와 함께 정의하였고, 사용되는 산출물의 양식을 구체적으로 제시하여 개발자에게 필요한 프로세스, 지침 및 표준 양식으로 사용되도록 하였다.

제안된 기법의 실효성 검증을 위하여 금융 도메인에 적용한 사례연구를 7장에서 상세히 제시하였으며, 8장에서 다른 기법들과의 비교를 다루었다. CBD의 공통 컴포넌트 개발에 제안된 프로세스와 지침의 사용함으로써 보다 재사용성과 적용성이 높은 컴포넌트 개발을 비용 및 시간적으로 효율적으로 개발될 것으로 기대된다.

참 고 문 헌

[1] Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J, Zettel, J., "Product Line Concepts," chapter 14 of Component-based Product Line Engineering with UML, Addison Wesley, 2001.

[2] Griss, M., "Product-Line Architectures," Chapter 22 of Component-Based Software Engineering, Addison Wesley, 2001.

[3] Kim, S., AND, Park, J., "C-QM: A Practical Quality Model for Evaluating COTS Components," Proceedings of International Association of Science and Technology for Developmen(IASTED) International Conference on Software Engineering (SE'2003), Innsbruck, Austria, PP.991-996, Feb. 10-13, 2003.

[4] Geyer, L., Becker, M., "On the Influence of Varibilities on the Application Engineering Process of a Product Family," SPLC2002, LNCS 2379, pp.1-14, 2002, Springer-Verlag Berlin Heidelberg 2002.

[5] Lee, Sang Duck, Yang, Young Jong, Cho, Eun Sook, AND, Kim, Soo Dong, "COMO : A UML-Based Component Development Methodology," Proceedings of Asia-Pacific Software Engineering Conference (APSEC99), Takamachu, Japan, PP. 54 - 61, Dec. 7-10, 1999.

[6] D'Souza, D., Wills, A., Objects, Components, and Frameworks with UML, Addison Wesely, 1999.

[7] Graham, I., Object Oriented Methods, Addison Wesely, 2001.

[8] Rumbaugh, J., Jacobson, I,, Booch, G., The Unified Modeling Language Reference Manual, Addison Wesely, 1999.

[9] Braude, E., Software Design From programming to Architecture , Wiley, 1999.

[10] Kim, S., "Lessons Learned from a Nationwide CBD Promotion Project," Communications of the ACM, October 2002.

[11] ISO/IEC, "Information Technology," Software Life Cycle Processes, International Standard 12207, 1995.

[12] The Rational Unified Process Product. The browser-based *online documentation for the RUP*, sold by Rational Corp.

장 수 호

2003년 숭실대학교 컴퓨터학부 공학사
2003년~현재 숭실대학교 컴퓨터학과 석사과정. 관심분야는 컴포넌트 기반 개발(CBD), 제품 라인 공학(PLE), 모델 기반 아키텍처(MDA)

김 수 동

정보과학회논문지 : 소프트웨어 및 응용
제 31 권 제 1 호 참조