The Journal of Korean
Institute of CALS/EC
Vol 7, No. 3, Dec. 2002

컴포넌트 유통시장 활성화를 위한 분류체계 모델링

이서정*, 조은숙*

Component classification modeling for component circulation market activation

Seojeong Lee, Eunsook Cho

Abstract

Many researchers have studied component technologies with concept, methodology and implementation for partial business domain, however there are rarely researches for component classification to manage these systematically. In this paper, we suggest a component classification model, which can make component reusability higher and can derive higher productivity of software development. We take four focuses generalization, abstraction, technology and size. The generalization means which category a component belongs to. The abstraction means how specific a component encapsulates its inside. The technology means which platform for hardware environment a component can be plugged in. The size means the physical component volume.

***Key Words** : component classification, component market, software reuse*

* 동덕여자대학교 정보학부 강의전임강사

1. 서론

지금까지의 소프트웨어 재사용의 발전속도는 하드웨어에 비해 빠르게 진행되지 못해왔으나 기존의 구조적 프로그래밍 기술에서 객체지향 기술로 진보하면서 재사용을 위한 연구가 많이 진행되었다. 클래스 위주의 재사용이 주를 이루었으나 실제 소프트웨어 개발에 있어 충분한 효과를 거두지 못했다[12]. 이러한 문제를 새롭게 극복하기 위해 컴포넌트 기술이 새로운 재사용 기술로 소개되기 시작했으며 90 년대 중반 이후 컴포넌트 개발을 포함한 관련 기술의 발전이 활발히 진행되고 있다. 그러나 아직 컴포넌트를 효율적으로 개발하기 위한 방법론이나 컴포넌트 품질 보증 기법, 컴포넌트 시험 기법, 다양한 분야의 컴포넌트들에 대한 표준 정립 등과 같은 많은 기술들이 아직 미성숙한 상태라 볼 수 있다[4]. 특히, 개발된 컴포넌트들을 효율적으로 관리하고 운영 및 유통하기 위한 컴포넌트 분류체계에 대한 연구는 거의 이루어져 있지 않다.

현재 국내 소프트웨어 유통시장은 패키지 형태의 완제품을 취급하거나 밴더사 마다 다른 기준으로 만들어진 프로그램을 고객의 요구에 맞게 커스터마이징하는, 즉, SI(System Integratio n)방식으로 이루어져 있다.

첫번째 방법은 완제품을 취급하므로, 사용자의 목적에 맞게 가공하거나 변경할 수 없어 비즈니스 용도 보다는 유틸리티 용의 제품들이 주종을 이루고 있다. 프로그램 내의 부품에 대한 재사용은 해당 밴더사의 해당 유틸리티에 국한될 수 밖에 없다. 두번째 방법은 고객의 요구에 맞출 수 있지만 작업시간과 투입인원에 따른 비용이 커져 중규모 이하의 프로젝트에서는 생산적이지 못하다.

결국, 소프트웨어의 개발비용을 줄이고, 생산성을 높이기 위해 소프트웨어의 부품 유통시장은 사용자 입장에서 뿐만 아니라 개발자 입장을 위해서도 활성화 되어야 한다.

외국의 사례를 보면 Component Source 나 Flashline 과 같이 컴포넌트 유통사이트를 통해 컴포넌트의 유통이 이루어지고 있으나 등록된 컴포넌트들이 사용자 인터페이스 용도의 컴포넌트들이 많이 있으며 또한 컴포넌트라기 보다는 어플리케이션 제품들이 등록되어 있는 경우들도 많다. 그 외의 사이트도 비슷한 경우가 대부분이다[10]. 따라서 컴포넌트를 검색해 보면 재사용하고자 하는 컴포넌트가 아니라 실제 소프트웨어가 해당 분야의 컴포넌트로 등록되어 유통되고 있는 경우도 볼 수 있다. 또한 유통사이트들의 컴포넌트에 대한 분류체계를 보면 분류체계가 체계적으로 마련되어 있지 않을 뿐만 아니라 분류 체계의 기준 또한 미비한 실정이다. 따라서 많은 회사들이 컴포넌트들을 개발하여도 이를 인터넷과 같이 네트워크 상에서 유통하기에는 아직 미약한 실정이며, 또한 컴포넌트 레포지토리에 등록할 경우에 있어서도 저장체계가 체계적으로 마련되어 있지 않다.

컴포넌트의 효율적인 등록, 검색, 그리고 유통하기 위한 분류체계를 마련하는 것은 컴포넌트 유통시장의 활성화를 유도하고 이

를 이용한 소프트웨어의 개발비용을 줄이는 역할을 할 수 있다.

2. 관련 연구

2.1. Component Source

1995 년에 설립되었으며, 미국과 유럽에 지사를 갖고 있다. 상품은 컴포넌트 소프트웨어만을 대상으로 확보하여 성장한 업체로 카드결제를 통해 컴포넌트를 구매 후 다운로드 받아 즉시 사용할 수 있는 유통모델을 갖고 있다. 현재 6 천여종의 컴포넌트를 판매중이다 [6]. 특징은 컴포넌트 시험 태스트를 실시하여 자체내의 엄격한 테스트를 거쳐 상품을 등록한다. 평가판 다운로드 서비스와 글로벌 언어와 다양한 통화단위(currency)를 제공하며 업그레이드 센터를 운영하여 제작사 및 상품명 별로 컴포넌트 검색이 가능하도록 한 점이다.

비즈니스 기능에 의해 113 개의 카테고리로 분류하였고 그 내부에서 플랫폼별 및 언어별로 검색이 가능하도록 했다.

- 플랫폼별 검색: .NET/COM/EJB/CORBA/ VCL / DLL / Library
- 언어별 검색: VB/Java/C/C++/Delphi

이 사이트는 카테고리 명을 보고도 개발자가 쉽게 찾을 수 있도록 분류체계를 세분화한 장점이 있지만, 하위 분류 체계 없이 바로 소분류로 구성하여 체계적으로 컴포넌트 분류가 되어있지 않다는 단점이 있다.

2.2. Flashline

Flashline.com 은 비즈니스를 위한 소프트웨어 시스템을 보다 빠르게 개발할 수 있도록 지원하는 소프트웨어 컴포넌트 제품, 서비스, 그리고 여러 자원들을 제공하는 회사로서, 웹을 통해 온라인으로 다양한 소프트웨어 컴포넌트들을 유통시키고 있다. 특히 온라인 상의 컴포넌트 유통을 위한 Flashline Component Manager 라는 제품을 출시하여 운영하고 있는데, 현재 Java - Beans, EJB, 그리고 COM 컴포넌트를 온라인으로 판매 유통하고 있을 뿐만 아니라 품질 보증, 오픈 컴포넌트 등록 등과 같이 다양한 자원들을 함께 제공하고 있다[8].

분류 체계는 크게 대분류, 중분류, 소분류 체계로 구성되어 있다. 우선 대분류에서는 컴포넌트 아키텍처(플랫폼)를 기반으로 Java 와 COM 으로 분류하고 있다. 중분류 체계로서 기술(Technology), Internet/ WWW, 통신, 정보관리, 사용자 인터페이스, 교육, 개발도구, 유틸리티, IDE Extensions 에 따른 분류체계를 정의하고 있다. 그리고 각각의 중분류 체계별로 소분류 체계들을 정의하고 있다.

컴포넌트를 대분류, 중분류, 소분류로 나누어서 컴포넌트들을 효율적으로 관리하는 분류체계를 지니고 있다는 것이 첫번째 장점으로 들 수 있다. 두번째 장점으로는 컴포넌트 사용자들로 하여금 컴포넌트들을 한번 돌려볼 수 있도록 demo 버전을 제공한다는 것이다. 그러나, 3 가지 형태의 분류체계를 가지고 있는데, 현재의 분류체계만 가지고는 비즈니스 도메인에 대한 분류체계

가 미흡하다는 단점이 있다. 예를 들어, 은행과 관련된 컴포넌트는 현재 이 사이트에서 제공하는 분류체계에 적합한 분야가 기술과 관련된 분류체계밖에 속하지 않는다. 또한, 중분류체계에 대한 기준이 혼합되어 있다. 즉, 기술 분야, 통신 분야, 유틸리티 분야 등과 같이 분류 레벨이 서로 다른 기준이 함께 중분류속에 정의되어 있음으로 인해서 사용자들로 하여금 혼동을 가져올 수 있다.

마지막으로 컴포넌트의 인터페이스에 대한 정보가 미약하고, 컴포넌트 사용자가 커스터마이즈(customize) 할 수 있는 부분에 대한 정보도 제공되고 있지 않다.

2.3. SanFrancisco

샌프란시스코(SanFrancisco)는 IBM 에서 개발한 프레임워크로서 솔루션 제공자(Solution Provider) 들이 Customized 멀티 플랫폼 시스템들을 구축하는데 있어 복잡도와 비용, 그리고 time to market 을 줄일 수 있는 응용 비즈니스 컴포넌트들을 제공한다 [9].

샌프란시스코를 이용함으로써 솔루션 제공자들은 인터넷 기반의 확장성 있고 다중 클라이언트/서버 플랫폼에 전개될 수 있는 재사용 가능한 비즈니스 프레임워크 위에 견고한 솔루션들을 개발할 수 있다. 샌프란시스코는 범용성의 정도에 따라 컴포넌트들을 계층별로 구별한 프레임워크 아키텍처로서 비즈니스 도메인에서의 범용성의 정도를 체계적으로 구별하여 정의하고 있다.

따라서 새로운 분산 환경 하에서 복잡하고 대형의 어플리케이션 개발에 있어서 개발자들로 하여금 특정 어플리케이션 개발을 용이하도록 지원하는 비즈니스 컴포넌트들 뿐만 아니라 플랫폼과 관련된 시스템 서비스 기능들을 제공해 주고 있다. 또한 특정 비즈니스 도메인에 재사용할 수 있는 컴포넌트들을 효율적으로 관리할 수 있도록 계층별로 구별하여 정의하고 있다.

샌프란시스코 아키텍처는 계층화 아키텍처로서 분산 컴퓨팅과 트랜잭션 무결성과 같은 기능들을 기술적으로 구현하는 것으로부터 개발자들을 보호해주는 아키텍처이다. 샌프란시스코를 이용함으로써 어플리케이션 개발자들은 시장에서 그들의 제품을 차별화해주는 그들의 비즈니스 측면들에 대한 자원들을 재배치할 수 있다. 따라서 샌프란시스코는 최근의 어플리케이션 개발에 있어서 기술적인 측면 뿐만 아니라 특정 산업 분야를 위한 새로운 어플리케이션 개발을 위한 기초를 제공해 준다.

샌프란시스코는 다중 계층의 공유가능하고 유연한 프레임워크 기반의 시스템으로서 플랫폼에 독립적이면서 분산 객체지향 소프트웨어 솔루션 개발에 유용하다. 샌프란시스코는 또한 공인된 디자인 패턴뿐만 아니라 특수 디자인 패턴들도 함께 사용하고 있다.

샌프란시스코 시스템은 3 개의 통합 계층인 기반 계층(Foundation Layer), 공용 비즈니스 객체 계층(Common Business Objects Layer: CBOs Layer), 핵심 비즈니스 프로세스 계층(Core Business Processes: CBPs Layer) 등으로 구성되어 있

다. 샌프란시스코의 아키텍처는< 오류! 참조 원본을 찾을 수 없습니다.>과 같다.

❑ 기반 계층

기반 계층은 샌프란시스코 아키텍처의 핵심 기술 계층으로서 분산 객체 생성, 동기화, 지속성(Persistence), 그리고 일관성 있는 어플리케이션 개발 모델과 같은 기초적 서비스들을 제공한다.

❑ 공용 비즈니스 계층

공용 비즈니스 객체 계층은 여러 비즈니스 도메인들간에 공통적으로 필요한 기능들을 수행하는 객체들로 구성되어 있다.이 계층은 다시 3 개의 범주로 구별하여 객체들을 그룹화하고 있다.

○ 일반 비즈니스 객체

이 그룹은 여러 비즈니스 도메인에서 공통으로 사용하는 비즈니스 객체 또는 공통으로 사용하는 기능들로 구성되어 있다.

○ 금융 비즈니스 객체

이 그룹은 비즈니스 도메인들 간에 공통적으로 존재하는 재무와 관련된 비즈니스 객체 또는 재무와 관련된 공통 기능들로 구성되어 있다.

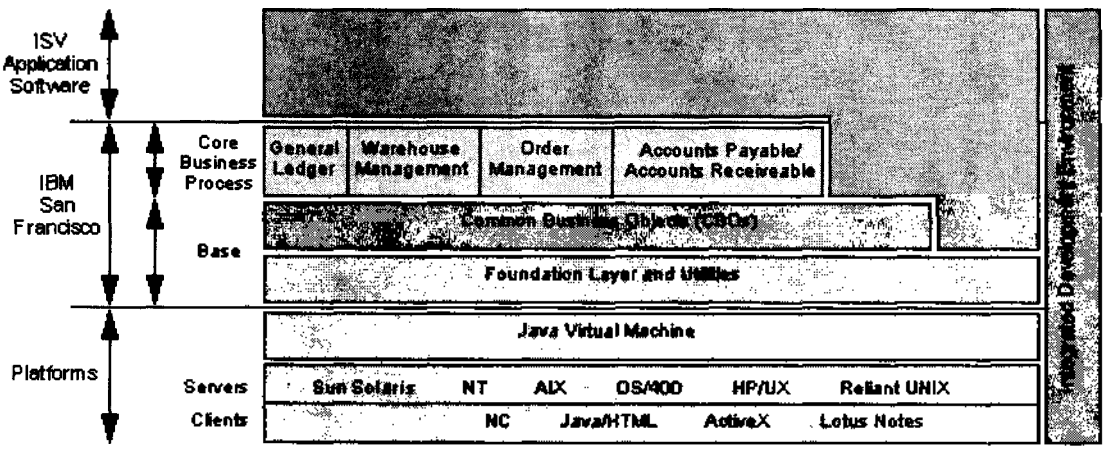
○ 일반화된 메커니즘

이 그룹은 많은 비즈니스 도메인들 간에 공통적인 문제들을 해결하는 데 사용된 기능들로 구성되어 있다. 이러한 기능은 추상화 되어서 여러 서로 다른 방법들로 사용될 수 있다.

❑ 핵심 비즈니스 프로세스 계층

핵심 비즈니스 프로세스 계층은 특정 비즈니스 도메인 상에서 하나의 어플리케이션을 위한 기본적인 빌딩 블록들의 집합을 제공하는 계층이다. 핵심 비즈니스 프로세스들은 해당 도메인에 대해 개발된 다양한 어플리케이션 구현을 자세히 관찰하고 같은 도메인 내에서 개발된 어플리케이션들 대부분에 필요한 프로세스들을 관찰해서 식별한다. 이 계층은 다시 다음 4 계층으로 나뉜다.

○ - SanFrancisco General Ledger



<그림 1> 샌프란시스코 아키텍처

- SanFrancisco Account Receivable/ Accounts Payable
- SanFrancisco Warehouse Management
- SanFrancisco Order Management

샌프란시스코는 입증되고 검증된 객체지향 기술을 바탕으로 하고 있고, 개발 주기를 줄일 수 있는 재사용 가능한 공용 어플리케이션 요소들을 제공한다. 그 결과, 다중 플랫폼에 전개될 수 있기 때문에 소프트웨어 개발자들에게 있어서 어플리케이션 개발 비용을 줄일 수 있어, 어플리케이션 개발자들은 프로그래밍과 기술 인프라스트럭처 문제들 보다는 비즈니스나 도메인 문제에 보다 초점을 둘 수 있다.

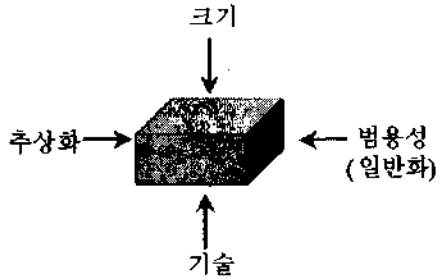
그러나, 샌프란시스코는 컴포넌트의 크기로 보면 프레임워크에 해당하는 체계를 지니고 있다. 그리고 이 프레임워크는 하부 서비스에 관련된 부분들과 비즈니스 도메인들 간에 공통된 부분과 특정 비즈니스 도메인에 구체적인 부분들을 계층으로 구별하고 있다. 따라서 이 프레임워크는 범용성의 측면에서만 컴포넌트들을 분류한 아키텍처라고 볼 수 있다.

또한 현재 공용 비즈니스 객체 계층에서 구별하고 있는 범주가 금융과 관련된 분야에 초점이 맞춰져 있으며 프레임워크로 제공하고 있기 때문에 어플리케이션 개발자들이 쉽게 접근하기가 어려운 점이 있다.

그 외, Object Tools, Uml, Sterling Software alc Butler Group 등의 분류체계 등을 참조할 수 있다[3,5,10].

3. 컴포넌트 분류체계

기존의 분류 체계들은 단일 관점에서 특히 일반성 각도에서만 분류하므로 사용자들의 요구를 충분히 반영하기 힘들고, 검색된 결과를 사용자가 검토하는 작업을 거쳐야 하는 불편이 많다. 본 연구에서는 이러한 불편을 줄이기 위해, 4 가지 관점, 즉, 일반화, 추상화, 크기 및 기술 관점에서 컴포넌트를 분류하여 < 그림 2> 와 같이 사용자가 원하는 컴포넌트를 효율적으로 찾을 수 있는 체계를 제안한다.



< 그림 2> 컴포넌트 분류의 4 가지 관점

3.1. 일반화 관점

범용성이라 함은 해당 컴포넌트가 사용되는 범위의 정도를 의미한다. 즉 범용성의 정도가 큰 컴포넌트는 해당 컴포넌트의 사용 범위가 넓다는 것을 의미하고 범용성의 정도가 작은 컴포넌트는 해당 컴포넌트의 사용 범위가 좁다는 것을 의미한다. 본 연구에서는 기존 연구 [1] 을 확장하고 한국산업 분류체계[2]를 참고하여 <그림 3>에 표현된 것처럼 일반성에 따른 컴포넌트를 크게 6 가지로 레벨로 규정한다. 그리고 이

러한 컴포넌트들을 개발하는 데 사용되는 개발 환경을 지원 레벨로 정의하고 있다.

<그림 3> 범용성에 따른 컴포넌트 분류 체계

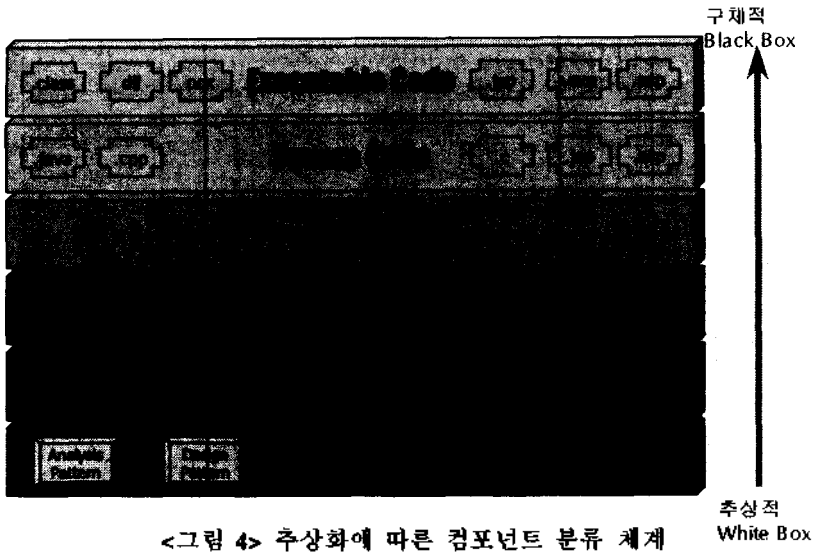
3.2. 기술에 따른 분류

여기서 기술이라 함은 해당 컴포넌트 개발에 적용된 기술을 의미한다. 현재 컴포넌트 개발을 지원하는 컴포넌트 개발 플랫폼에는 썬(Sun)사의 EJB 와 JavaBean, OMG의 CORBA Component Model(CCM), 그리고 마이크로소프트(Microsoft)사의 Component Object Model(COM) 이 있는데 본 연구에서의 기술이라 함은 이러한 기술 플랫폼 외에 컴포넌트, 프레임워크, 또는 어플리케이션 개발에 적용되는 기술들을 총망라한다[3][7]. 그 이유는 현재 운영되고 있는 컴포넌트 유통 사이트들을 보면 컴포넌트의 형태를 단지 배치(Deployment)할 수 있는 컴포넌트들 만을 취급하지 않고 자그마한 Graphic User Interface(GUI) Widget 에서부터 어플리케이션에 이르기까지 다양한 레벨의 컴포넌트들을 유통하고 있다. 따라서 이러한 유형들을 모두 처리하기 위해서는 기술들을 모두 적용하여 분류를 해야 한다.

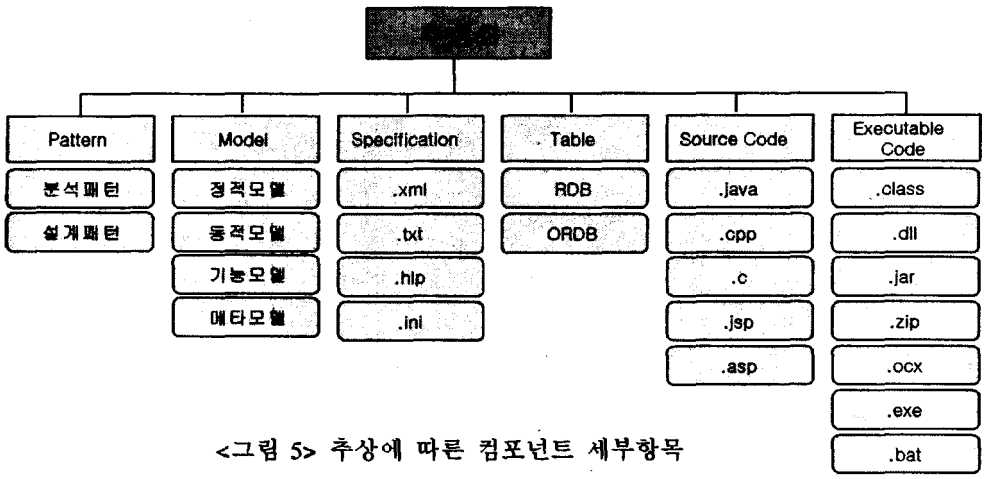
현재의 기술을 분류한다면, Java, CORBA, COM, XML, PHP 등과 각 기술의 세부분야로 나눌 수 있다.

3.3. 추상화에 따른 분류

추상화(Abstraction)란 불필요한 것을 제거하고 사물의 본질을 분리해 내고 요약하여 핵심 정보만을 추출해 내는 것이다. 컴포넌트를 분류하는 데 있어서 이러한 추상화를 적용하는 이유는 컴포넌트에 대한 정의가 너무 다양하고 컴포넌트를 정의하는 수준이 넓은 의미를 가지고 정의하는 경우도 있고 좁은 의미를 가지고 정의하는 경우도 있다. 예를 들어 어떤 사람은 컴포넌트를 실제 시스템 구동시 배치되어 돌아가는 독립적인 모듈로 정의하는 가 하면, 반면에 어떤 사람은 재사용할 수 있는 것은 모두 컴포넌트가 된다고 간주하는 경우도 있다. 후자의 경우를 보면 설계 패턴(Design Pattern)도 재사용이 되면 컴포넌트가 되고, 소스코드(Source Code) 도 재사용이 되면 컴포넌트가 되는 것이다. 따라서 본 연구에서는 이러한 다양한 정의들을 반영하기 위해서 컴포넌트의 추상화 정도를 가지고 분류체계를 마련한 것이다. 추상화의 정도에 따른 분류 체계는 <그림 4> <그림 5>와 같다.



<그림 4> 추상화에 따른 컴포넌트 분류 체계



<그림 5> 추상에 따른 컴포넌트 세부항목

3.4. 크기에 따른 분류

컴포넌트의 크기라 함은 어플리케이션 개발에 재사용되는 단위의 정도를 말한다. 현재 컴포넌트 크기에 대한 표준 규격이 마련되어 있지 않은 상태이다. 따라서 컴포넌트의 단위를 규정하는 시각들이 다양하다. 재사용되는 하나의 클래스를 컴포넌트라고 하거나, 여러 컴포넌트들로 구성된 프레임워크도 컴포넌트로 보거나, 또는 다른 컴포넌

트를 포함하는 복합 컴포넌트를 컴포넌트로 보는 경우가 있다. 이처럼 컴포넌트의 크기들이 다를 수 있기 때문에 동일한 이름의 컴포넌트라고 하여도 크기가 다른 컴포넌트들이 된다. 이러한 문제점을 고려하여 컴포넌트를 크기에 따라 다음과 같이 분류할 수 있다.

- Component based Application Level
- Framework Level
- Component Level
- Class Level

4. 컴포넌트 분류 코드 표기법

컴포넌트 분류 코드는 컴포넌트의 체계적인 관리를 위해 필요하고, 해당 컴포넌트 분류 체계를 식별하기 위한 식별자 역할을 한다[11]. 본 논문에서는 앞에서 분류한 컴포넌트 분류체계를 관리할 수 있는 코드를 제안한다. 분류 코드 형식은 각각의 관점에 따라 표현된다.

Co.{Ge|Gr|Te|Ab}.XXX.

분류코드의 CO 는 전체 컴포넌트 분류체계를 의미하는 문자로서 모든 컴포넌트의 상위 클래스인 Component 의 약어인 " Co" 이다.

두번째 분류 {Ge|Gr|Te|Ab} 는 컴포넌트의 분류 관점을 표현하는 문자로서, 각각의 관점을 영어로 표기한 단어를 축약해서 표현한 것으로 다음과 같은 의미를 갖는다.

✓ Ge: Generality
✓ Gr: Granularity
✓ Te: Technology
✓ Ab: Abstraction

세번째 분류 코드는 각각의 관점별로 표현한 세부 분류 체계들에 표현 형식이다. 이 형식은 관점에 따라 각각 달리 표현하며, 관점마다 세부 분류 체계의 수준이 다를 수 있다.

각각의 관점별로 표현한 세부 분류 체계들에 대한 표현 형식은 다음과 같다.

4.1. 범용성 관점의 분류 코드 형식

Co.Ge.{UIL|PLL|SSL|CBL|CRL}.XXX.XXX. 이 분류 코드 형식은 범용성 관점에 대한 코드 형식으로 세번째 분류코드는 범용성 관점에서의 각각의 계층들을 표현하는 문자열이다. 네번째 분류 코드는 각각의 계층별로 분류한 세부 분류 체계들을 표현한 것이다.

✓ UIL: User Interface Layer
✓ PLL: PLatform Layer
✓ SSL: System Service Layer
✓ CBL: Common Business Layer
✓ CRL: CoRe business Layer
✓ APL: APlication Layer

다섯번째 분류 코드 이후부터는 네번째 분류 코드의 세부 분류 항목들을 표현한 것으로서, 이것은 계층에 따라 존재할 수 도 있고, 존재하지 않을 수도 있다. 예를 들어, CRL 과 같은 핵심 비즈니스 도메인에 대한 분류 코드는 다섯번째 분류코드 이후도 존재가능하지만 PLL 과 같은 계층에 대한 분류 코드는 다섯번째 분류코드 이후가 존재하지 않는다.

4.2. 크기 관점의 분류 코드 형식

Co.Gr.{CLL|COL|FRL|CAL}

크기 관점의 분류코드 형식에서 세번째 코드는 크기 단위 수준에 따른 분류 코드이다. 코드 표현은 영문자를 함축하여 표현한 것이다.

✓ CLL: CLass Level
✓ COL: COmponent Level
✓ FRL: FRamework Level
✓ CAL:Component-based Application Level

4.3. 기술 관점의 분류 코드 형식

Co.Te.{Java|COM|CORBA|기타}.XXX

기술 관점의 분류 코드 형식에서 세번째 코드는 기술별로 분류한 코드로서, 코드 표현은 대표기술 별로 표현한 것이다. 그 다음으로 나오는 네번째 코드는 각각의 기술별로 분류한 세부 기술들에 대한 표현 형식이다. 예를 들면, Java 인 경우에 있어서 EJB, JavaBean, JSP 등과 같은 항목들이 여기에 표현되는 것이다. 세부 항목들은 기술 관점 분류 체계를 참조하여 표현한다.

4.4. 추상화 관점의 분류 코드 형식

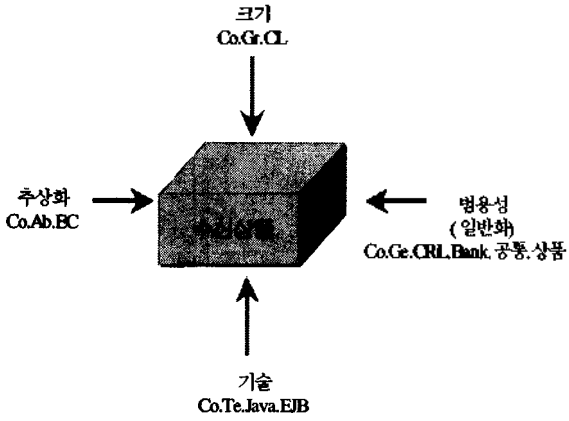
Co.Ab.{PT| MD | SP | TB | SC | EC }.XXX

추상화 관점의 분류 코드 형식에서 세번째 코드는 각각의 추상화 수준 별로 분류한 코드로서, 코드 표현 방식은 영문 표기법을 함축하여 표현한 것이다.

✓ PT: Pattern
✓ MD: Model
✓ SP: Specification
✓ TB: Table
✓ SC: Source Code
✓ EC: Executable Code

네번째 코드는 각각의 추상화 수준별로 분류한 세부 분류 항목들을 의미한다. 예를 들어, 세번째 코드가 Pattern 인 경우에 있어서는 Analysis Pattern 또는 Design Pattern 과 같은 항목들이 여기에 표현되는 것이다. 세부 항목들은 추상화 관점 분류 체계를 참조하여 표현한다.

지금까지 정의한 분류코드 표기법을 기반으로 예를 들어 특정 컴포넌트가 소속하는 분류 코드를 4 가지 관점에서 <그림 6>으로 표현할 수 있다.



<그림 6> 분류코드 표기법 적용 예

5. 참고문헌

[1] ETRI, "*영역별 컴포넌트 분류 방법에 관한 연구*", 1999, 12.

[2]통계청,"*한국표준산업분류*" (http://www.nso. o.kr/stat/indclass/k-ind1.htm), 2001

[3] Active-X.COM (http://www.active-x.com)

[4] Butler Group, "*Component-Based Develop ment: Application Delivery and Integration Using Componentised Software*", September, 1998.

[5] Component Bank (http://www.Component Bank.com)

[6] ComponentSource (httP://www.componentsource.com)

[7] Desmond F. D'souza and Alan C. Wills, *Objects, Components, and Frameworks with UML*, p91-234, Addison-Wesley, 1998.

[8] Flashline (http://www.flashline.com)

[9] IBM, "*Architecture of the San Francisco rameworks*", (http://www.research.ibm.com/journal/sj/372/bohrer.html), Dec., 1997.

[10] ImagiCom (http://www.imagicom.com)

[11] Robert C.Seacord, "*Software Engineering Component Repository*", ICSE 1999.

[12] Szyperski,C., *Component Software: Beyound Object-Oriented Programming*, Addison Wesley Longman, Reading, Mass., 1998.

저자 소개

이서정(*sjlee@dongduk.ac.kr*)
1998년~현재 동덕여자대학교 정보학부강의교수
1998년 숙명여자대학교 대학원 전산학과 박사과정 졸업
1991년 숙명여자대학교 대학원 전산학과 석사과정 졸업
1989년 숙명여자대학교 전산학과졸업
관심분야: 시스템 분석 및 설계, 컴포넌트 공학, 데이터리파지토리
프로젝트:
1997년 숙명여자대학교 행정전산화 시스템 분석 및 설계(LG-EDS)
2000년 e-biz 마이닝 설계(SDS),
2001년 KCSC 컴포넌트 표준안,정보통신부

조은숙(*escho@dongduk.ac.kr*)
2000년~현재 동덕여자대학교 정보학부강의교수
2000년 숭실대학교 대학원 컴퓨터학과 박사과정 졸업
1996년 숭실대학교 대학원 컴퓨터학과 석사과정 졸업
1993년 동의대학교 자연과학대학 전산통계학과 졸업
관심분야: 소프트웨어 공학, 컴포넌트 방법론, 메타모델
프로젝트:
1997년 인터넷 기반 클라이언트/서버 인터페이스 개발,한국전자통신연구원
1998년 인트라넷 프레임워크 모델링 기법 및 프레임워크 구현에 관한 연구,한국전자통신연구원
1999년 인트라넷 워크플로우 프레임워크 모델링 및 개발,한국전자통신연구원
1999년 KT ICIS 개발 방법론 보강,한국통신
2001년 KCSC 컴포넌트 분류 표준체계 개발,정보통신부
2001년 레거시 시스템을 CBD 시스템으로 변환하기 위한 개발 방법론 개발, 한국전자통신연구원