객체지향 소프트웨어 재사용을 위한 클래스 라이브러리 설계에 관한 연구

이 해 원† · 김 진 석†† · 김 혜 규††† · 하 수 철††††

요 약

본 논문은 객체지향 C++ 클래스 컴포넌트를 분류하여 재사용자에게 필요한 컴포넌트를 제공하기 위한 저장소의 클래스 라이브러리 설계방법을 제안한 것이다. 클래스 라이브러리를 설계하기 위해서 컴포넌트 구성 모델을 정의하였고, Enumerative 분류 방법을 이용한 멀티미디어 영역을 분류하였으며, 문서 클러스터링 방법을 확장하여 유사도에 의한 C++ 클래스를 유사한 그룹으로 분류하는 클러스터 생성 기준을 제안하고 있다. 이 유사 그룹인 클러스터는 클래스 멤버 데이터와 멤버함수 그리고 클래스 유사도를 기반으로 분류되며, 분류된 컴포넌트들은 유사도 관계의 계층구조로 구성된다. 마지막으로 객체지향 개념인 Generalization/Specialization의 C++ 상속관계를 계층구조로 표현할 수 있는 클래스 라이브러리를 설계하였다.

The Study of Class Library Design for Reusable Object-Oriented Software

Hae-Won Lee† · Jin-Seok Kim†† · Hea-Gju Kim††† · Soo-Cheol Ha††††

ABSTRACT

In this paper, we propose a method of class library repository design for provide reuser the object-oriented C++ class component. To class library design, we started by studying the characteristics of a reusable component. We formally defined the reusable component model using an entity relationship model. This formal definition has been directly used as the database schema for storing the reusable component in a repository. The reusable class library may be considered a knowledge base for software reuse. Thus, we used that Enumerative classification of breakdown of knowledge based. And another used classification is clustering of based on class similarity. The class similarity composes member function similarity and member data similarity. Finally, we have designed class library for hierarchical inheritance mechanism of object-oriented concept Generalization, Specialization and Aggregation.

1. 개 요

소프트웨어 재사용 기술은 오래 전부터 소프트웨어 개발의 생산성과 품질을 향상시키기 위한 중요한 관심사로 제시되어 왔다[1][2][3]. 소프트웨어를 마치 하드웨어 부품처럼 재사용 한다면 소프트웨어 개발은 아주 빨라질 것이며 소프트웨어 위기 현상을 해결할 것이라는 전망이 있어왔다. 그러나 현실적으로 소프트웨어 개발은 재사용이 보편화 되지 않고 있다. 이런 현상에 대하여 문제점을 제시하고 여러 방향으로 그 해결책을 찾으려는 재사용 연구가 진행되어 왔다. Kang이 주장

† 정 회 원 : 한국전자통신연구원 우정정보화팀 연구원
†† 정 회 원 : 한국전자통신연구원 우정정보화팀장 책임연구원
††† 정 회 원 : 한국전자통신연구원 우정기술연구부장 책임연구원
†††† 종신회원 : 대전대학교 컴퓨터공학과 교수
논문접수 : 1998년 11월 23일, 심사완료 : 1999년 8월 25일

전 개발 방법론은 체계적 재사용을 고려하지 않는 개발이 문제라고 지적하여 재사용 기반 개발방법론을 제시하였고[4], Diaz는 재사용 부품의 분류가 중요하다고 주장한 영역분석 분류 방법론을 제안하였다[5]. 또한 재사용을 편리하게 하는 프로그래밍 언어 즉 기능에 대한 개발 방법론이 제시되었고[6][7], 원시코드보다는 설계 및 분석 정보의 재사용 방법이 제안 되었으며[8], 재사용 도구 및 환경에 대한 연구가 Latour에 의해서 진행되었다[9]. 그리고 재사용은 기술보다는 기술 외적인 요소 즉 관리 방법이 중요 하다고 하여 이를 연구한 Tracz방법 등이 있다[10]. 최근에 와서는 객체지향 기술이 발달하면서 객체지향 프로그램이 소프트웨어 설계 방법과 프로그래밍 방법을 기반으로 소프트웨어 개발 방법으로서 재사용 효율을 극대화 시킬 수 있다고 하였다[11].

따라서 본 연구에서는 부품 정의를 통해서 소스코드 분석서, 설계서 및 예제 등을 제공 하므로서 재사용성을 높였고, 부품의 효과적인 분류를 통해서 재사용자에게 소스코드의 계보 및 사용 용도 등의 이해정보를 제공하여 부품의 전반적인 이해를 돕고있다. 이러한 분류는 지식기반 도메인을 분석하여 재사용 도메인을 Category 영역으로 분류하고 이를 계층적인 Category로 제시 하였으며, 하위 Category에 유사 부품을 대표하는 클러스터를 두어 유사 부품을 그룹화 하였다. 이 클러스터는 제안된 클러스터링 방법으로 부품을 분류하여 계층적인 클러스터가 자동 구성되게 하였다 또한 객체지향 C++코드를 기반으로 분석서, 설계서, 예제, 분류정보 등을 포함하고 있는 부품 즉 컴포넌트를 구성하였으며 이 객체지향C++ 클래스 특징인 Generalization (일반화)/Specialization(상세화)의 상속관계 및 Aggregation(집단화)의 포함관계와 클래스 라이브러리 확장성을 지원하기 위해서 클래스 유사성을 기반으로 새로운 분류 방법을 적용한 객체지향 클래스 라이브러리 설계 방법을 제안 하였다.

제2장에서는 분류 방법으로 사용하는 문서의 유사성을 기반으로 문서 분류 방법과 소프트웨어 재사용 시스템에서 사용하는 클래스 분류 방법들에 대해서 설명하고, 3장에서는 재사용 소프트웨어 적용영역 분류와 클래스 유사성을 이용한 클러스터 생성방법 등을 바탕으로 클래스 라이브러리 설계를 설명한다. 제4장에서는 기 개발된 재사용 시스템과 비교분석을 통한 성능분석을 하였고, 마지막으로 제5장은 결론이다.

2. 관련연구

2.1 문서 클러스터링 방법

문서들을 분류하고 검색하기 위한 방법으로 클러스터링 방법이 많이 사용되는데, 문서들을 클러스터링할 때에는 유사성을 바탕으로 하게 된다. 대표적인 유사성 측정 방법은 inner-product measure방법, cosine measure방법, pseudo-cosine measure방법, dice measure 방법 등이 있으며[12], 이들 방법들은 공통적으로 두 집단 (i,j)간의 가중치 weight(i,j)를 결정해야 하는데, 이 가중치들은 보통 두 집단의 같은 항목 갯수로 결정된다. 이 가중치들을 정규화 하는 식을 이용하여 두 집단의 유사성을 결정한다. 예를 들어 Dice measure 방법에서 문서들의 유사성은 다음과 같이 표현 한다.

$$S_{ij} = \frac{2^{*}C}{A+B}$$

여기서 C는 i번째 집단과 j번째 집단에 공통적으로 존재하는 의미 있는 Word의 수이고, A는 i번째 집단에 존재하는 의미 있는 Word들의 수이며, B는 j번째 집단에 존재하는 의미 있는 Word들의 수이다.

문서들을 클러스터링하는 방법들은 Single pass방법, Reallocation방법, Single linkage방법, Complete linkage 방법, Average linkage방법, Ward's Method 방법 등이 있으며[13], 이중 대표적인 몇 가지 방법을 살펴 보면 다음과 같다.

- Single linkage 방법

Sneath에 의해서 묘사된 방법으로 새로운 클러스터를 이루는 부품들은 클러스터 내의 가장 유사한 문서를 기준으로 이루어 진다. 이 방법은 한 클러스터 내의 최소한 하나의 member가 또 다른 클러스터 내의 모든 member들과 비교해서 결합이 이루어지는 클러스터에 대해 높은 유사성을 갖게 하는 방법이다. 이 방법의 특징은 내부적으로 응집력이 강하지 않은 형태의 클러스터를 형성하는 경향과 chaining으로 참조되는 구조이다.

- Complete linkage 방법

Sokal과 Michener에 의해서 묘사된 방법으로 Single linkage 방법과는 상반된 클러스터 내의 가장 유사도가 낮은 문서를 기준으로 하여 새로운 클러스터를 구성하는 방법이다. 다시 말해 한 클러스터 내의 한 mem

ber와 다른 클러스터 내의 한 member사이의 가장 낮은 유사도가 적어도 또 다른 클러스터 내의 member와 최저 유사도보다는 높은 클러스터들을 결합시키는 방법이다.

- Average linkage 방법

Sokal과 Michener에 의해서 제안된 방법으로 문서 내의 각 클러스터 member들의 평균 유사도가 가장 높은 클러스터들을 그룹화하여 짝지우는 방법이다. Single linkage나 complete linkage를 절충한 방법으로서 'non-conformist' 클러스터가 생성되는 경향이 있다.

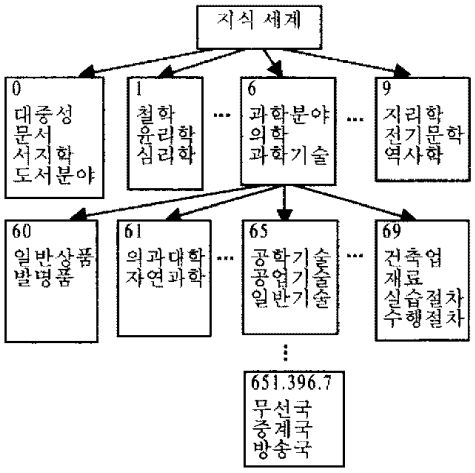
클러스터링 방법들은 문서들을 효율적으로 확장할 수 있지만 상속성을 표현하지 못하기 때문에 C++ 클래스의 상속계층 구조를 효율적으로 구성하지 못한다. 특히 문서 파일의 내용들은 단순한 word들의 집합으로 이루어져 있고, 각 문서 파일들 간에는 독립적인 관계 이지만 C++ 클래스의 내용은 클래스 내부에 일정한 의미를 지니는 구성 요소 즉 멤버 데이터, 멤버 함수, 상속관계, 글로벌 변수, 함수 로칼 변수 그리고 집단화 관계 등의 다양한 특징 있기 때문에 문서 분류에 사용되는 클러스터링 방법을 C++ 클래스를 분류하는데 직접사용하기에는 적합치 않다.

2.2 소프트웨어 분류 방법

소프트웨어 컴포넌트는 재사용의 단위가 되며, 컴포넌트를 효과적으로 분류하게 되면 컴포넌트 재사용 및 재사용 라이브러리 구성과 검색을 효율적으로 이루어지게 할 수 있다. 따라서 소프트웨어 분류가 중요하게 다루어지고 있고 소프트웨어 재사용의 개념을 기반으로 만들어진 컴포넌트를 재사용하기 위해 분류 방법에 대한 분석이 필요하다. 소프트웨어 분류 방법으로는 크게 Faceted방법, Enumerative방법, 클러스터링 방법 등이 있다.

2.2.1 Enumerative 분류 방법

Enumerative방법은 Dewey가 DDC(Dewey Decimal Classification)를 제안 함으로서 처음 시도 되었으며[20], 이후 IFD(International Federation for Documentation)에서 UDC(Universal Decimal Classification)를 제안하여 Enumerative Classification방법을 발표하였다[14]. 이들 방법들은 도서 분류를 기반으로 이루어 졌으며, DDC는 초기 도서 제목을 기반으로 핵심 키워드를 추출하고 그룹핑하여 계층적으로 도서를 분류하는 방법으로, 10진수로 이루어지는 코드이며, 621.381.530.422 등과 같이 된다. 이는 분류가 명시적이고 계층적이지만 코드를 말할 때 부 자연스러운 형태를 취하고 있다. 또한 UDC는 도서 제목 뿐만 아니라 저자의 이력 및 제목을 기반으로 지식 영역을 계층적으로 분류 하는 방법으로 DDC와 달리 주요 테이블과 보조테이블로 이루고 있다. 주요테이블은 제목을 기반으로 이루어지며, 보조테이블은 기타 유용한 정보 즉 저자, 사용 언어, 다른 도서 제목과 관계된 정보, 출판 시기 등을 기반으로 정보를 추출한다. 이 방법도 숫자로 된 코드 651.396.746.9등으로 이루어지며 코드만 인지하여 도서를 연상시키기는 무리가 있으나 코드를 기반으로 추출된 정보를 명시하면 명확히 지식구조가 (그림 1)과 같은 계층화된 도서 분류를 할 수 있다. 이 계층적인 분류는 검색 시 이해하기 쉽게 명시적인 화면을 구성할 수 있다.



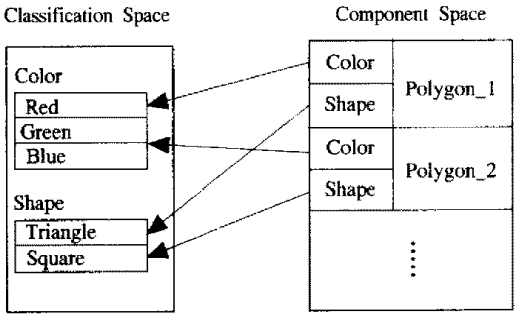
(그림 1) Enumerative 지식영역 분류

Enumerative방법은 지식을 기반으로 표현하려는 지식의 광역 범위에서 협소 범위로 Breakdown해 가며 세부 분류를 하는 방법이다. 그러나 거꾸로 좁은 범위에서 광역 범위로는 분류를 수행하지는 않는다. 이는 객체지향 개념의 상세화를 수행하는 것과 같으며, 일반화를 수행치 않는 1차원적인 방식이다. 장점으로는 도메인의 계층성을 쉽게 표현할 수 있어 재사용자로 하여금 부품을 찾아가기 쉽게 명시적으로 계층도를 제공해줄 수가 있다. 그러나 Enumerative 분류방법은 광역의 범위에서 협소 범위로 Breakdown하여 분류하는 방법으로 도메인의 계층성을 쉽게 표현할 수 있어 검

색기에게 명시적으로 도메인 계층도를 제공할 수 있지만, 본 논문의 객체지향 재사용 라이브러리 부품인 C++ 클래스들의 일반화/상세화의 다중상속관계와 집단화의 포함관계를 표현하지 못하는 단점과, 확장성에 있어 많은 부품을 분류할 경우 분류계층도가 깊고 복잡하여 사용자를 오히려 어렵게 만드는 단점이 있다. 이 방법을 이용한 재사용 시스템으로는 IFD의 Intermetrics사의 RSL 시스템[15]이 있다.

2.2.2 Faceted 분류 방법

Faceted 분류 방법은 서재 분류 방법으로 1924년 Ranghanathan에 의해서 Colon 분류 방법으로 제안 되었다[16][17]. 이후 1985년에 Prieto-Diaz가 제안한 Facet 모델이[5][18] 개발되었고 이를 기반으로 여러 시스템이 연구 개발 개발되었다. Faceted 분류 방법의 기본 개념은 아래 (그림 2)에서 보여주는 것과 같은 분류 영역으로 Color와 Shape 같은 Facet 항목(Term)으로 부품 Polygon_1, 2를 묘사하고 있다. 예를 들어 Facet으로 Square만 명시하고 검색하면 부품 Polygon_1, 2가 검색 되지만, Red와 Square를 Facet으로 명시하고 검색하면 정확히 Polygon_1이 검색되어 진다.



(그림 2) Facet 분류

예에서 보는 바와 같이 이 방법은 각 Facet Term내에 부품의 특징을 묘사할 수 있는 Termspace두어 부품을 검색할 수 있게 하였다. 적용 분야 및 개발자에 따라 이 Facet Term 즉 Facet의 갯수가 다르게 정의되고 있고, 각 Facet은 등록되는 부품 수에 따라 자체 Termspace내에 여러 개의 부품을 연결할 수 있는 term을 갖게 한다. 또한 Facet의 구성영역 즉 Classification Space를 Facetspace라 부르며, 이 Facetspace 내에 Termspace 갯수를 몇 개로 정하는 것에 따라 부품 묘사 정도가 달라지게 되고 너무 많게 되면 등록 및 검색 시 사용자가 부품을 정확히 묘사하기가 번거롭게 되고 반면 너무 적게 정하면 부품을 정확히 묘사하기가 어렵게 된다. 따라서 등록 되는 부품마다 Facet 갯수 만큼 term의 인덱스를 갖고 저장되며 검색 시 각 Facet의 Termspace에 명시된 term의 조합으로 검색되는 개념이다. Enumerative 분류 방법을 1차원적 분류라 할 때 이 방법은 다차원적 분류 방법이라 할 수 있으며 다차원이라는 의미는 Facetspace의 Facet 구성과 Termspace의 term 구성되기 때문이다.

Prieto-diaz의 Faceted 분류 방법은 소프트웨어 소스 코드 컴포넌트를 분류하기 위하여 만들어 졌고 소프트웨어 재사용 분류 모델로 가장 많이 인용되어 소프트웨어 분류 방법의 참조 모델로 인지 되었다. 이 방법의 핵심 기능은 기능 중심 개념으로 분류하기 때문에 객체 중심의 객체지향 소프트웨어 분류에는 적용하기가 어렵다. Prieto-diaz 방법은 2개의 Facet 그룹으로 나누어져 있고 각 그룹 당 3개의 Facet Term으로 총 6객의 Facet Term으로 구성된다. 첫번째 그룹의 Facet들은 가능성 즉, 어떤 기능을 수행하는가를 묘사하는 것으로 다음과 같은 Facet들로 이루어져 있다.

- 기능 표현 : < function, object, medium>
- Function : 프로그램에 의해서 수행되는 독특한 기초 기능을 묘사
- Object : 프로그램에 의해서 만들어지는 Object(문자, 라인, 그래픽..)의 종류 묘사
- Medium : 활동이 수행되는 곳과 같은 제공하려는 부품의 속성들을 묘사

이 그룹의 기능성을 묘사하는 Facet들의 예를 들면 다음과 같다.

<search, root, b-tree> 또는 <input, character, buffer>

두 번째 그룹은 환경을 묘사하는 Facet들로 구성되며 컴포넌트가 행하는 일의 상황을 묘사하는 것으로 다음과 같은 Facet들로 이루어져 있다.

- 환경 표현 : <system type, functional area, setting>
- System type : 기능적으로 독립 기능을 묘사, 즉 어플리케이션에 독립된 모듈, 이 모듈은 하나 이상의 컴포넌트를 포함할 수 있다.
- Functional area : 컴포넌트가 적용될 영역을 묘사한다.

• Setting : 어플리케이션이 사용될 곳을 묘사한다.

두 번째 그룹은 환경성을 묘사하는 Facet들로 적용 예를 보면 다음과 같다.

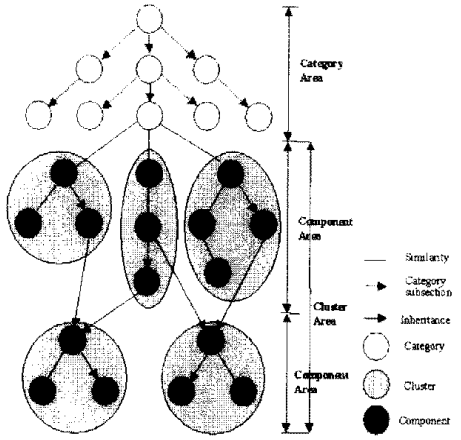
<lexical analyzer, cost accounting, cemetery>
또는
<expression evaluator, billing, barbershop>

이 분류 방법의 질의어 구성은 6개의 Term으로 놓여진 위치 즉, 나열된 순위에 따라 우선 순위가 정해져 있으며, 첫번째 팀이 제일 높고 가장 중요하다. 검색 시 하나 이상의 Facet을 나열하면 될 수 있으나 6개 Facet 전체를 나열하여 검색 했을 때 제일 정확한 검색이 이루어 진다. 그러나 Prieto-Diaz의 Faceted 분류 방법은 기능중심 개념으로 부품을 분류하기 때문에 Facet Term을 근거로 정확한 부품과 유사 부품을 검색하는 데는 유리하나, 객체지향 개념의 일반화/상세화의 상속관계와 집단화 개념의 포함 관계를 지원하지 못하는 단점과 컴포넌트 등록자마다 컴포넌트의 각 Facet Term을 정의하는 의미가 다르기 때문에 정확성에 의문이 제기되어 객체지향 재사용 라이브러리 분류 방법으로는 합당치 못하다. 이 방법을 적용한 시스템으로는 GTE사의 재사용 시스템이 있다[5][19].

3. 클래스 라이브러리 설계

객체지향 클래스 라이브러리를 설계하기 위하여 앞 장에서 언급한 방법과 새로운 클러스터링 방법의 클러스터를 바탕으로 객체지향 클래스 라이브러리를 설계하였다. 우선 재사용될 대상인 재사용 부품 클래스의 구성을 정의 하고, 이 재사용 부품이 저장될 클래스 라이브러리를 Category 계층과 클러스터 계층 그리고 부품 계층으로 나누어진 Embedded Three-tired 계층 구조를 (그림 3)과 같이 구성하였다.

(그림 3)에서 각 클러스터가 여러 개의 컴포넌트를 포함하고 있는 형태로 이루어져 있기 때문에 Embedded Three-tired 계층 구조라고 명하였고, 클래스 라이브러리를 3개 층으로 구성한 이유는 첫번째 계층에서 실세계를 반영하여, 찾고자 하는 재사용 영역을 세분화하여 개발자들의 개발영역에 부응할 수 있도록 Category 계층을 정의 하였다. 두 번째 클러스터 계층은 검색하고자 하는 세부 재사용 Category영역에서 유사한 기능을 하는 부품들을 그룹핑한 대표 기능그룹을 제공하기 위한 계층이다. 마지막으로 세 번째 계층인 Component계층은 검색하는 부품의 객체지향 C++ 클래스들의 상속관계와 부품들의 상세 유사도를 검색 시 제공하기 위한 계층으로 정의 하였다. 이렇게 3개 계층으로 분류하므로 서 검색자는 찾고자 하는 부품을 세분화된 재사용영역을 찾고, 이 영역에서 대표기능을 하는 기능그룹 찾아서, 그 밑에서 실제 찾고자 하는 부품에 합당한 기능을 수행하는 알맞은 부품을 검색하여 재사용 할 수 있게 하였다.



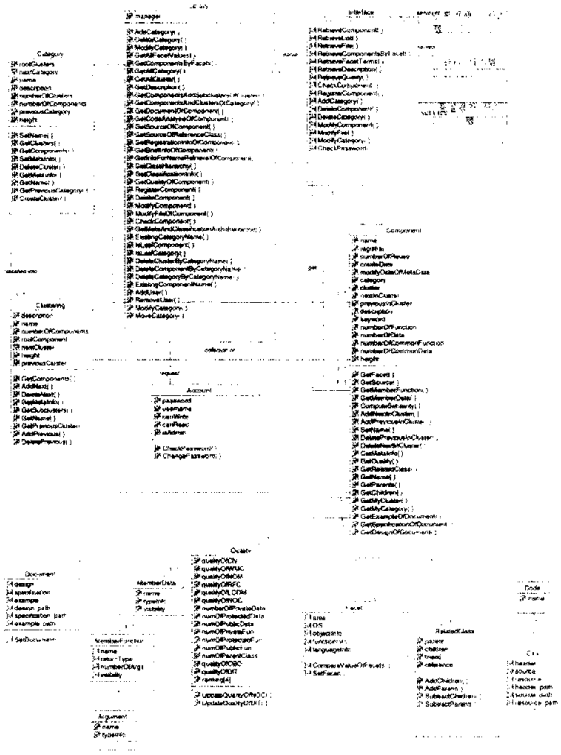
(그림 3) Embedded Three-tired Class Library Hierarchy

이 클래스 라이브러리는 효율적인 검색과 관리를 위해서 설계되었고, 제공되는 검색 방법으로는 위에서 설명한 경로 탐험 검색과 그 외에 Facet검색 방법 그리고 클래스 이름(Name)으로 검색하는 Keyword 검색 방법을 제공할 수 있도록 설계하였다.

또한 재사용 도메인 분석을 통해 영역을 정의하여 재사용자로 하여금 재사용 부품을 등록 시 기 분류된 Category가 없으면 새로운 Category부터 등록하고 부품을 등록하게 하였다. Category 및 부품 삭제 권한은 사용자로 등록되는 사용자마다 권한이 다르며, 부여되는 권한은 시스템 관리자가 사용자를 등록할 때 권한이 부여되게 하였다.

(그림 3)의 Embedded Three-tired Class Library Hierarchy를 바탕으로 객체지향 클래스 라이브러리 설계를 (그림 4)와 같이 하였다. 이 객체지향 클래스 라이브러리는 서버에서 클라이언트 요청에 대응하여 동작할 객체를 모델링한 것으로 객체지향 데이터베이스를 구축할 경우 바로 데이터베이스 스키마로 적용되어

바로 C++ 코드화 될 수 있다.

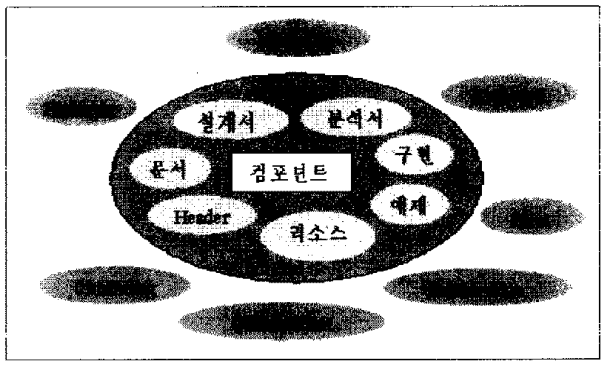


(그림 4) 클래스 라이브러리 객체 모델

(그림 4)에서 Interface객체는 클라이언트의 검색 시스템, 등록 시스템 관리 시스템으로부터 각종 요청을 받아주는 외부에 노출된 객체이고, Library객체는 클래스 라이브러리 전체를 관리하는 객체로서 실제 DB를 검색, 등록, 수정, 삭제 등의 기능을 수행하는 객체이다. Category 객체는 사용자에게 Category를 제공하여 재사용영역을 명시적으로 보여 주무로서 경로 탐색을 지원하기 위한 객체이며, Clustering 객체는 유사도를 기반으로 클러스터링을 수행하며 클러스터 정보를 관리한다. Component 객체는 재사용될 부품 즉 클래스 소스코드와 재사용에 필요한 메타 정보를 관리하는 객체이다. Facet 객체는 5개 Facet Term을 제공하여 부품 등록 시 Facet Term을 입력하여 검색 시 Facet 검색을 지원하게 하였다. 또한 Quality객체는 부품 등록 시 부품을 평가하여 부품 재사용자에게 해당 부품의 품질 정보를 제공하기 위함이며, 기타 객체는 사용자 관리와 재사용 시 재사용자에게 부품의 이해를 돕기 위해서 부품으로서 갖추어야 할 일부 메타정보를 관리하는 객체들이다.

3.1 재사용 부품 구성 방법

부품을 재사용하기 위해서는 부품의 기능이나 특성을 잘 나타낼 수 있고 효율적인 검색이 가능하도록 분류 저장되어야 하며, 검색된 부품의 기능이나 사용 범위 등을 쉽게 이해할 수 있게 구성되어야 한다. 따라서 재사용 부품의 구성을 원시 코드로서 헤더 파일(*.h), 구현파일(*.C++), 리소스 파일(*.rc)로 구성 하였으며, 메타 정보는 Enumerative, Facet, Cluster, Qualification, Hierarchy, History, Keyword, 설계서, 분석서, 예제 그리고 문서 등으로 구성하였다. 또한 클래스 라이브러리에 등록하고 검색하여 재사용할 수 있는 부품의 구성을 부품 규격으로 정의 하였으며, (그림 5) 재사용 부품 구성 모델과 같다. 부품 규격의 C++ 코드 제한은 Bjarn Stroustrup가 제안한 "The C++ Programming Language 2nd Edition에서 나온 문법을 기본으로 한다.



(그림 5) 재사용 부품 구성 모델

(그림 5)의 각 구성 모델을 살펴보면 다음과 같은 구성요소로 이루어져 있게 이해정보로 제공되어 진다.

- 구현 : C++ 소스코드를 말하며 등록될 부품은 객체지향 단일 클래스로 구현된 파일의 소스코드 Body 부분이다.
- 설계서 : 부품의 설계정보가 담긴 파일로, 이티 모델링 툴을 사용하여 생성한 설계 파일이다
- 분석서 : 부품의 상세 정보가 담긴 파일이며 부품의 전반적인 배경, 사용 범위나 방법 등을 설명하는 부분과 클래스의 각 멤버와 멤버함수를 설명한 파일이다.
- 문서 : 부품의 분석서나 설계서를 보지 않고도 간단히 부품을 이해할 수 있도록 부품의 목적, 특징

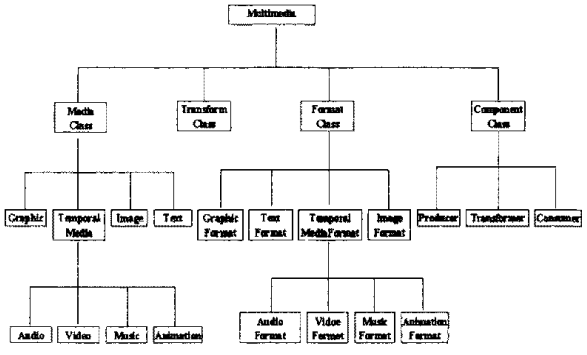
및 기능 사용법을 말한다.

- Header : C++ 코드의 인터페이스를 정의한 Header 파일(*.h)을 말한다.
- 리소스 : Visual C++ 등의 툴을 사용하여 GUI에 관련된 부품인 경우 리소스 파일을 말한다.
- 예제 : 부품을 재사용할 경우 부품이 사용된 실 예를 제공하여 부품의 이해 도를 높이도록 도와주는 예제 파일이다.
- Keyword : 부품을 대표할 수 있는 특징을 기술한 것으로 Keyword 검색 시 이용되는 정보이다.
- Hierarchy : 부품의 일반화/상세화의 상속관계 및 집단화의 포함관계 정보를 표현한 것으로 Runtime Library를 사용한 경우는 제외 한다.
- History : 재사용 횟수를 표현하며 검색자가 부품을 검색하여 Download할 경우 재사용 횟수는 증가한다.
- Enumerative : 지식기반으로 도메인 분류를 Enumerative 방법으로 분류 하게끔 한 것으로 이 계층을 Category 영역으로 정의하고 사용자에게 명시적인 분류 계층을 제공하여 Category를 추가 등록 및 삭제할 수 있는 도메인 분류 정보이다.
- Facet : 클래스 라이브러리에 저장되는 부품은 부품의 특징을 잘 표현하고 효율적으로 검색할 수 있도록 다섯 가지 Facet 즉 대상(Object), 기능(Function), 영역(Area), 언어(language), 운영체제(O.S)를 정의하고 부품 등록 시 Facet을 등록하게 하여 검색을 쉽게 제공하는 정보이다.
- Clustering : 유사한 기능을 수행하는 부품을 그룹핑한 정보로 부품 등록 시 클래스 유사도를 기반으로 자동 클러스터링 되는 클러스터 이다.
- Qualification : 재사용 부품의 소스 코드로부터 품질을 평가하기 위하여 8가지 메트릭 즉 복잡도, 멤버 함수의 수, 상속 트리 깊이, 상속 트리의 넓이, 자식 노드 수, 객체간의 결합도, 클래에 대한 반응도 그리고 메소드 응집도의 결핍성 등을 기반으로 이식성, 재사용성, 신뢰성, 유지보수성 등을 제공하는 정보이다.

3.2 카테고리 구성 방법

Category구성 방법은 재사용될 분야를 명시적으로 제공하기 위해서 재사용 영역을 실세계 지식을 기반으로 분류하였으며, 분류 단위는 재사용 영역 즉 도메인 영역 뿐만 아니라 분류된 한 도메인에서 세부 분류를 점차 협소 부분으로 분류할 수 있게 하였다. 이렇게 하여 경로탐색 검색 기능인 Tree검색 메뉴를 두어 사용자가 좀더 쉽게 찾고자 하는 영역(부분)을 선택하게 하고 선택된 부분에서 찾고자 하는 부품에 좀더 일치성이 높은 재사용 부품을 찾을 수 있게 하였다. 예로서 멀티미디어 분야를 분석하여 Category 영역을 분류하였으며, (그림 6)의 Root 노드멀티미디어 밑에 세부 부분을 분류하여 재사용 영역을 좀더 세분화하여 정확성을 꾀하고자 하였다. 즉 찾고자 하는 재사용 영역을 사용자가 복잡하게 느끼지 않을 정도로 세분화하여, 해당분야에서 재사용할 부품을 찾을 수 있는 분류 방법과 검색 방법을 지원하기 위한 방법으로 제시 하였다. 부품을 정형화하는 방법으로는 부품의 정보로부터 수동으로 정보를 추출하는 분류접근 방법(Classification Approach)과 자동으로 정보를 추출하여 가공하는 정보추출 방법(Information retrieval approach)[21]이 있는데 본 논문에서는 이 두 가지를 기반으로 영역 분류수준 즉 Category구성을 분류접근 방법으로 구성하게 하였고, 그 하위 수준인 클러스터구성은 정보탐색 방법을 적용하여 클러스터를 자동적으로 구성되게 하였다.

Category구성은 재사용 영역으로 멀티미디어 영역을 분석하여 (그림 6)과 같이 구성하였으며, 그림에 명시된 Category외에 운용하면서 Category를 추가 등록 및 삭제할 수 있게 설계하였다. 그러나 Category 삭제 시 삭제할 Category에 기 등록된 부품을 먼저 삭제한 후 해당 Category를 삭제되게 하였다. 설계한 멀티미디어 영역 외에 다른 영역도 (그림 6)과 같은 실세계 지식을 기반으로 넓은 영역의 컴포넌트 집합에서 점차 좁은 범위의 클래스를 분할해 가면서 그들 간의 계층적 관계를 명시하므로 서 새로운 영역의 명시적인 계층구조 Category를 새롭게 구성할 수가 있다.



(그림 6) 클래스 라이브러리 Category 구성

본 논문에서는 Enumerative방법을 적용한 Category 구성만으로는 Enumerative방법의 단점인 확장성의 한계와 객체지향 개념의 클래스 상속관계를 표현하지 못하므로 Category 하위 수준으로 클러스링 단계를 두어 클러스터를 생성하고, 클러스터에 등록되는 클래스들의 일반화와 상세화의 상속관계와 집단화의 포함관계 및 유사 부품의 분류 정보를 재사용자에게 제공하여 부품의 이해도를 높여 효율적인 재사용이 이루어지게 설계 하였다.

3.3 클래스 유사도 계산 방법

클래스 유사도는 재사용될 부품 클래스들을 클러스터링 하기 위하여 사용되는 값으로 클러스터링의 기본 데이터 이다. Dice measure 방법의 문서 유사도 계산은 단순히 같은 단어 이름을 추출하여 그것이 전체 단어에서 차지하는 비율로 유사성을 계산하는 방법이다. 그러나 C++ 클래스는 멤버 데이터, 멤버 함수, 일반화/상세화의 상속관계, 글로벌 변수, 함수 로칼 변수 그리고 집단화의 포함관계 등의 특징으로 이루어져 있어서, 문서 유사도 계산과 같이 단순히 이름이 같다고 같은 기능을 한다고는 볼 수 없다. 이 것은 일 예로 멤버 데이터와 멤버 함수 이름이 같다고 같은 기능을 한다고는 볼 수 없기 때문이다. 따라서 본 논문에서는 Dice measure 방법을 바탕으로 클래스 클러스터링에 사용될 유사도 계산 방법으로서 클래스 유사도를 정의하였으며, 클래스 유사도는 멤버 데이터 유사도와 멤버 함수 유사도 비중의 합으로 이루어지는 유사도 계산 방법을 제안하고 각각을 정의 하였다.

- 멤버 데이터 유사도

$$S_d = \frac{2^*(d(A) \bigcap d(B))}{d(A) \bigcap d(B)}$$

여기서 $d(A)$는 클래스 A에 존재하는 멤버 데이터 갯수이고, $d(B)$는 클래스 B에 존재하는 멤버 데이터 갯수이다. 또한 $d(A) \bigcap d(B)$는 클래스 A와 B의 공통된 멤버 데이터 갯수로서 공통된 멤버 데이터란 두 클래스에 모두 존재하는 동일한 멤버 데이터를 말하며 두 멤버 데이터가 동일하다는 것은 멤버 데이터의 이름(name)과 데이터 형(type)이 같다는 것으로 정의한다. 따라서 멤버 데이터 유사도 는 클래스 A와 클래스 B에 존재하는 멤버 데이터 중에 동일한 기능인 멤버 데이터 갯수에 배수하여 이를 클래스 A와 B에 존재하는 멤버 데이터 갯수의 합으로 나눈 값이 된다.

- 멤버 함수 유사도

$$S_f = \frac{2^*(f(A) \bigcap f(B))}{f(A) \bigcap f(B)}$$

여기서 $f(A)$는 클래스 A에 존재하는 멤버 함수의 갯수이고 $f(B)$는 클래스 B에 존재하는 멤버함수의 갯수이다. 또한 $f(A) \bigcap f(B)$는 클래스 A와 B의 공통된 멤버 함수 갯수로서 공통된 멤버 데이터란 두 클래스 A, B에 모두 존재하는 동일한 멤버 함수를 말하며, 두 멤버 함수가 동일하다는 것은 두 멤버함수의 이름이 같고, 반환값의 데이터 형이 같고, 인수의 이름과 데이타형 및 갯수가 같고, 지역 및 글로벌 변수의 이름과 형이 같다는 것으로 정의 한다.

이렇게 정의한 이유는, 일반적으로 두 함수가 기능이 같다는 대표적인 연구는 Sampling Behavior 방법[22]에 있으나 이방법은 두 함수에서 임의의 입력 데이터에 대해서 같은 출력을 생성하면 같은 기능을 수행한다고 보고 있다. 그러나 이 방법은 모든 입력에 대해서 반복적으로 수행해야 하기 때문에 많은 시간이 걸리고, 또한 입력 자료와 출력 자료가 불투명한 것에는 이용될 수 없기 때문이다.

멤버 함수 유사도 역시 멤버 데이터 유사도와 같이 클래스 A와 클래스 B에 존재하는 멤버 함수 중에 동일한 기능인 멤버 함수 갯수에 배수 하여 이를 클래스 A와 B에 존재하는 멤버 함수 갯수의 합으로 나눈 값이 된다.

- 클래스 유사도

$$S(A,B) = P^*S_d(A,B) + (1-P)^*S_f(A,B)$$

$$P = \frac{d(A+B)}{d(A+B)+f(A+B)}$$

여기서 $d(A+B)$는 클래스 A, B의 멤버 데이터 갯수의 합이며, $f(A+B)$는 클래스 A, B의 멤버 함수 갯수의 합으로서 P는 두 클래스 A, B의 멤버 데이터와 멤버 함수의 총 갯수에서 멤버 데이터가 차지하는 비율이다. 따라서 (1-P)는 두 클래스 A, B의 멤버 데이터와 멤버 함수의 총 갯수에서 멤버 함수갯수가 차지하는 비율이 된다. 그러므로 클래스 유사도에 멤버 데이터 유사도와 멤버 함수 유사도의 비중을 같은 갯수일 때 같게 하였다.

3.4 클래스 클러스터링 방법

클래스 클러스터링은 3.3절에서 정의한 클래스 유사도를 기반으로 클러스터를 생성하고 해당 클러스터에 부품을 등록하는 과정을 말한다. 한 클러스터는 유사 기능을 수행하는 부품을 그룹핑하는 대표 부품이다. 예를 들어 등록자가 클래스 라이브러리에 부품을 등록할 때 재사용 도메인 지식을 기반으로 분류된 Category를 지정하고 등록을 요구하면, 등록될 부품이 아래 등록 제한사항을 위반하지 않고 클러스터 생성 기준에 부합되면 클러스터를 생성하고 재사용 가능한 부품으로 등록한다. 그러나 등록될 부품이 클러스터 생성기준에 부합되지 않으면 모든 클러스터에 등록된 부품들과 클래스 유사도를 계산하여 유사도 임계값 이상인 것 중에서 가장 유사도가 높은 부품 밑에 재사용 가능한 부품으로 등록한다. 이때 클러스터 생성 기준이 되는 클래스 유사도의 임계값을 0.7이하로 정하였다. 0.7이하로 임계값을 정한 이유는 Lewis-Beck[13]에 의하면 유사도가 0.7이상일 경우 유사한 것으로 볼 수 있다고 말하고 있다. 따라서 이를 기반으로 클래스 유사도의 임계값을 0.7로 정하고 맴버 데이터와 멤버 함수의 임계 유사도는 0.8로 정하였다. 0.8로 정한 이유는 객체지향 C++ 소스 코드 특성상 클래스는 멤버함수와 멤버데이타 및 클래스 접근성으로 복잡하게 이루어져 있기 때문에 좀더 유사성의 정확도를 높이기 위해서다. 따라서 갯수의 차이가 2개 이상이면 유사성이 없다고 정의할 수 있다. 예를 들면 두개의 클래스가 있을 때 멤버 데이터의 갯수 차이가 20% 이상 차이 날 경우 클래스 유사도가 0.8이 나올 수 없을 뿐만 아니라 동일한 멤버 데이터 갯수가 역시 80%이상 존재해야만 유사도가 0.8이상 나올 수가 있다. 이는 멤버 함수일 경우도 마찬 가지이다.

본 클래스 라이브러리 설계에서는 다음과 같은 제한사항을 두어 등록 되는 부품이 제한사항을 위반하지 않는 부품에 대해서는 등록을 허락 하도록 설계하였다.

- 제한사항
  - 일반화/상세화의 상속관계로 된 등록할 부품의 Parent 부품이 해당 Category에 없는 경우
  - 집단화 관계로 된 등록할 부품의 Parent 부품이 해당 Category에 없는 경우
  - 클래스 하나로 구성된 구현 파일과 Header파일이 아닌 경우
  - 동일한 클래스 이름이 이미 지정한 Category내에 등록이 되어 있는 경우

이 제한 사항을 두는 이유는 소프트웨어 개발자에 따라서 코딩 기교가 다르기 때문에 설사 C++ 문법상 오류가 없더라도 코드가 복잡하고 난해하여 이해하기 어렵고 재사용하기 복잡한 부품에 대해서는 등록을 허락하지 않게 설계하였다.

다음은 클래스가 등록될 때 클러스터 생성기준을 정의하여 유사부품을 그룹화 하였다. 한 클러스터에 속한 클래스들은 유사한 기능을 하는 유사 클래스를 그룹핑한 것으로 초기 등록된 클래스 이름이 클러스터 이름으로 등록 되며 클러스터 정보는 부품 구성의 문서 내용을 갖게 하였다.

- 클러스터 생성 기준

클래스 등록 시 클러스터링에 의한 클러스터 생성 기준은 3가지 규칙에 근거하여 이루어지게 하였다.

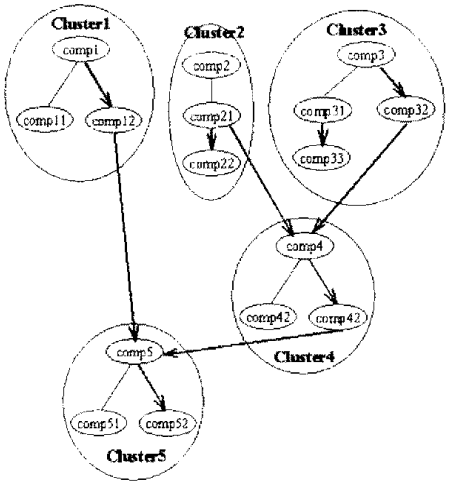
규칙 1 : 클래스가 등록될 Category에 최초로 될 때 이루어지며 이때는 유사도를 계산치 않고 클러스터 이름은 클래스 이름으로 이루어지며 분류에 관한 클러스터 정보는 등록되는 클래스의 문서 내용이 된다.

규칙 2 : 클러스터들이 2개 이상 등록되어 있고 등록되는 클래스가 상이한 2개 이상의 클러스터에 등록된 클래스로부터 다중상속 받는 클래스가 등록 될 때 이루어지며 이때도 새로운 클러스터가 생성되기 때문에 유사도를 계산치 않고 클러스터 이름은 클래스 이름으로 이루어지며 분류에 관한 클러스터 정보는 등록되는 클래스의 문서 내용이된다.

규칙 3 : 새롭게 등록되는 클래스가 지정된 해당 Category에 등록된 모든 클래스와 유사도를 계산하여 유사도가 임계값 이하일 때 새로운 클러스터를 생성하고 규칙 1, 2와 같이 클러스터 정보는 클래스의 정보를 취하게 된다.

(그림 7)은 클러스터 계층구조를 나타낸 것으로 화살표가 있는 선은 두 부품사이에 상속관계를 나타내는 것으로 화살표의 시작점이 부모 크래스이고 종점이 자식 클래스이다. 방향이없는 직선의 의미는 클러스터링에의한 유사기능의 척도를 나타내는 유사성 관계로서 유사도가 가장 큰 의미를 지니고 있는 것으로 클러스

내 내부에서만 존재하는 클러스터 논리적 계층관계를 나타낸다. 따라서 논리적으로 상위의 부품과 가장 유사기능을 수행한다고 볼 수 있다.



(그림 7) 클래스 라이브러리 클러스터 구성

그림에서 클러스터가 Cluster1, 2, 3순으로 생성된다고 하면 Cluster1은 최초 등록 부품 comp1이 등록될 때 생성되며(규칙 1), Cluster2는 부품 comp2가 등록되면서Cluster1에 속한 모든 부품과 클래스 유사도를 계산하여 유사도 값이 임계값 이하 이기 때문에 새로운 Cluster2가 생성되었다(규칙 3). 또한 Cluster3도 comp3가 등록될 때 cluster1과 cluster2에 속한 모든 부품과 클래스 유사도를 계산하여 임계값 이하로 되어서 생성하게 되었다(규칙 3). 그리고 Cluster4,5는 규칙 2에 의해서 상이한 클러스터에 속한 부품으로부터 다중 상속을 받았기 때문에 생성 되었다.

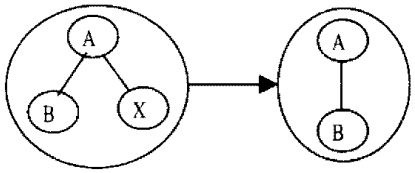
다음은 사용자가 컴포넌트 클래스 및 Category를 삭제 하고자 할 때 삭제 기준을 정의한 것으로 Category 삭제는 삭제하고자 하는 Category내에 등록된 컴포넌트가 하나도 없을 경우에 한해서 삭제를 허용한다. 여기서 삭제할 클러스터 C와 컴포넌트 X의 삭제 기준은 다음과 같다. 그리고 삭제 권한은 삭제 하고자 하는 부품을 등록한 사용자와 시스템 관리자만이 삭제 권한이 있다.

- 클러스터와 컴포넌트 삭제 기준

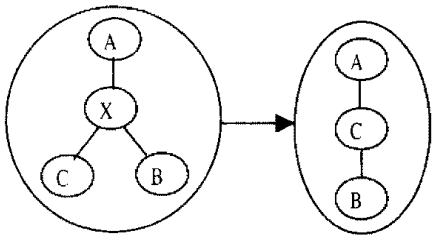
본 논문에서 설계한 클래스 라이브러리에서 부품X를 삭제할 경우 삭제할 부품이 단말노드이면 부품 이름 X와 해당 Category 이름만 지정하면 삭제된다. 왜냐하면 해당 Category에 등록되어 있는 부품 X의 이름은 유일하기 때문이다.

규칙 1 : 부품 X가 컴포넌트 계층 구조 상에서 최상위 부품이 아니경우

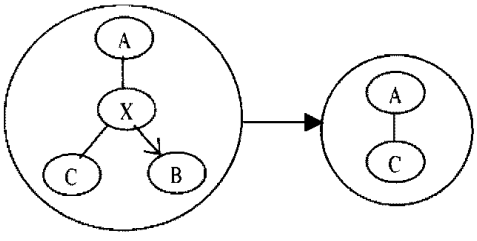
① X가 단말 노드인 경우 하위 계층의 상속 및 유사 계층상의 부품이 존재하지 않으므로 삭제하고자하는 부품은 아래 그림과 같이 부품 X만 삭제하고 부품에 관련된 부속 정보와 계층에 관련된 정보를 삭제한다.



② X의 하위 노드가 있고 하위 노드가 부품 X와 유사성 관계로 존재 하는 경우 그림과 같이 부품 Xdhk 관련 정보를 삭제하고 유사도가 가장 높은 노드 C가 부품 X를 대신하여 계층도를 재구성 한다.

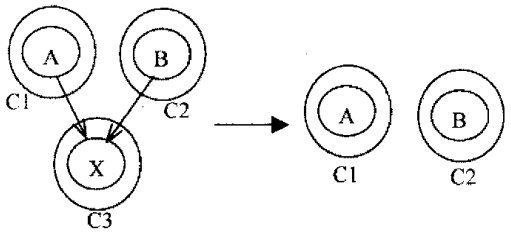


③ X의 하위 노드가 있고 하위 노드가 부품 X와 일반화/상세화의 상속관계로 존재하는 경우 컴포넌트 구조를 고려하여 아래 그림과 같이 하위노드를 먼저 삭제하고 부품 X를 삭제한 후 유사도가 가장 높은 노드 C가 X를 대신한다. 즉 부모 부품을 삭제 하고자 할 때 자식 부품에 대한 삭제 연산을 먼저 수행한 후 부모 부품을 삭제 한다. 이것은 재귀식으로 자식 부품에 대한 삭제 연산을 호출하면 된다.



규칙 2 : 부품 X가 컴포넌트 계층 구조 상에서 최상위 부품인 경우

① 부품 X가 속한 클러스터 C의 레벨이 1이거나 단말 클러스터인 경우 아래 그림과 같이 삭제하고자하는 부품 X가 하위 부품을 가지고 있으면 규칙 1의 를 수행한 후 하위 부품이 없으면 해당 클러스터를 자동 삭제 한다.



② 부품 X가 속한 클러스터 C의 레벨이 2 이상이거나 단말 클러스터가 아닌경우는 부품을 삭제하지 못하고 하위 클러스터를 삭제되게 한후 부품 X를 삭제 하여야 한다. 삭제하지 못하는 이유는 하위 클러스터는 클러스터 생성기준 2의 다중 상속에 의하여 생성되었으므로 이를 삭제할 경우 하위 전체 컴포넌트 상속 구조가 깨지게 때문이다.

3.5 Faceted 분류 방법

Faceted 분류 방법은 Prieto-Diaz가 제시한 방법론을 사용하여, 기능적인 요소를 의미하는 Term을 "Object"와 "Function"으로 정의 하였으며, 환경적 요소를 의미하는 Term을 "Functional Area", "Programming Language", "Operating System"으로 정의하였다. 이 Facet Term 분류 근거는 찾고자하는 부품이 어떤 "속성(Object)"이 어떤 "기능(Function)"과 어느 "분야(Functional area)"에 어떤 언어(Programming Language)"로 구현되어 어느 "운영체제(Operating System)"에서 재사용할 수 있는 부품을 찾고자 한다는 시나리오에 근거하여 Facet Term을 정의 하였다. 이를 정의한 Facet 구성 테이블이 <표 1>과 같다.

〈표 1〉 Facet Term 구성

Object	Function	Functional Area	Programming Language	Operating System

<표 1> Facet Term 구성을 보면 총 5개의 Facet Term으로 구성되어 있고 첫번째 "Object"는 C++ 클래스의 멤버 데이터를 정의하였다. 멤버 데이터로 정의한 이유는 일 예로 인사에 대한 직원 클래스를 정의하고자 할 때 직원 클래스 가지는 멤버 데이터는 이름, 직급, 입사일, 학력, 주소 등으로 모델링 된다면 재사용자로 하여금 직급을 조작하는 컴포넌트를 찾고자 할 때 Object Facet의 항목으로 지정하기 위해서 이다. 두 번째 "Function"은 직원의 추가 삭제 학력 등의 기능을 조작하는 것을 통상 멤버함수로 정의하기 때문에 검색시 직원의 추가 삭제를 하는 컴포넌트를 찾고자 할 때 Function Facet의 항목으로 지정하기 위함이다. 세 번째 "Functional Area"는 재사용 적용 분야인 도메인을 묘사하려고 한 것이며, 네 번째 "Programming Language"는 코딩된 Language 종류를 묘사하려고 정의한 것이다. 마지막 다섯 번째 "Operating System"은 적용하려는 컴퓨터의 운영체제를 묘사하기 위하여 정의 하였다. 이상과 같이 5개의 Facet을 부품 등록 시 묘사하게 하고 검색 시 이를 지정하게 하면 정확한 부품 컴포넌트가 찾아지게 된다. 그러나 5 Facet모두를 지정하지 않고 검색 하더라도 후보 부품을 제시하여 재사용자로 하여금 선택하여 재사용할 수 있게 하였다. 이는 검색 의도 부품의 정확성이 결여 되지만 재사용자가 Facet의 정확한 의미를 모를 경우 유용하게 사용될 수 있게 하였다.

4. 비교분석

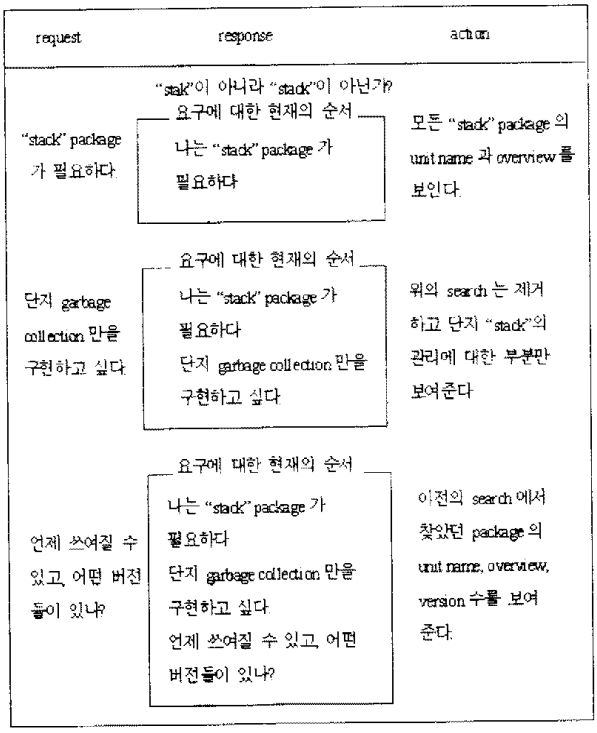
4.1 Intermetrics사의 RSL시스템

RSL 시스템[15]은 소프트웨어를 재사용하기 위해서 사용자가 재사용 가능한 컴포넌트를 쉽게 검색하고, 이를 재사용하기 위한 분류 기법과 데이터베이스에 기초한 라이브러리 구조로 설계되었다.

RSL 시스템의 S/W 분류 기법은 두 가지 분류 기법을 사용한다. 첫번째 방법은 라이브러리에 등록되는 각 컴포넌트에 계층적인 Category Code를 부여하는 방식으로 이 Category Code는 다른 컴포넌트와의 관련성 및 컴포넌트 타입을 표현하고 있다. 두 번째 방법은 컴포넌트에 Category Code와 관련이 없는 5개의 서술적인 Keyword을 부여하여 분류의 확장성을 지원하고 있다.

RSL 시스템의 모든 서브 시스템들은 데이터베이스를 액세스하여 필요한 정보를 얻게 된다. 메뉴 방식으

로 저장되는 사용자 질의는 레이블과 같은 색인용에 필요한 정보를 바탕으로 작성된다. 하지만 저장소로 네이터베이스를 사용하기 때문에 사용자 질의를 자연어 검색을 지원하고 있다. 그러나 객체지향 C++의 일반화/상세화의 상속관계와 집단화의 포함관계 등을 계층적으로 표현하지 못하기 때문에 객체지향 클래스 라이브러리로 적용하기 힘들다. 검색 방법으로는 메뉴 방식을 기반으로 Keyword 검색의 author, unit name, category code 검색과 자연어 검색을 지원하고 있다. 여기서 unit name은 Procedure, package, subroutine 등의 이름을 말한다. (그림 8)에서 자연어 검색 예를 보이고 있다.



(그림 8) 자연어 질의어를 통한 부품 검색

4.2. GTE 재사용 시스템

GTE 회사의 재사용 시스템은 Facet 분류 방법, 개념 그래프, 기능적으로 동등한 부품들을 평가하고 등급을 매기는 방법들을 통합한 프로토타입 라이브러리 시스템으로서 높은 정확도와 분류에서의 확장성을 지닌다. 모듈들이 갖는 여러 속성들을 각 Facet별로 나누어 이해하므로 모든 모듈들은 각 Facet에 해당하는 term(항목)들을 조합시켜서 원하는 기능을 정의할 수 있고, 분류 대상에 따라 적절히 적응하기 쉬울 뿐 아니라 지속적으로 모듈의 집합이 확장되는 경우에도 계층적 방법보다 쉽게 대처할 수 있다.

Facet분류 방식은 재사용 가능한 컴포넌트를 선택하기 위해 비슷한 항목들을 모아서 그룹화 시키는데, GTE 재사용 시스템에서는 소프트웨어 부품을 Facet에 의하여 function, object, medium, system-type, functional-area, setting 등으로 분류된다. 소프트웨어 부품들은 소프트웨어 재사용 시스템에 등록하기 위해 Facet값을 결정해야 하는데 이를 위하여 GTE 재사용 시스템에서는 동의어 사전 관리를 두고 있다.

프로그램은 무엇을 할 것인가에 대한 함수(functionality)를 설명하는 부분과 그 함수가 수행될 환경(environment) 부분으로 구성된다고 하여, Functionality 부분은 <function, object, medium> 등의 Facet으로 나누어지고, environment 부분은 <system-type, functional-area, setting>등의 Facet으로 나누고 있다.

<function, object, medium>은 프로그램의 수행과 관련된 Facet들로 일반적인 <action, object, agent>와 대등한 관계이다. "function" 항목은 프로그램에서 수행되어지는 함수와 관련된 부분이며 move, start, compare 등의 함수가 여기에 해당되고, "object" 항목은 객체들의 조작과 관련된 작업을 설명하는 부분으로 character, lines, variables 등이 해당된다. "medium" 항목은 "agent"로 대신할 수 있으며, action이 수행되는 곳을 취급하는 객체들로 간주되어지며, 함수를 위한 structure를 지원한다.

<system-type, functional area, setting>은 환경과 관련된 Facet들을 설명하고 있다. 일반적으로 환경은 프로그램을 실행하는 내부 환경과 프로그램을 적용시키는 외부 환경으로 나누어진다. "system-type" 항목은 하나 이상의 컴포넌트들이 기능적으로 동일한지에 대하여 알아보기 위한 것으로 보통 소스 코드 리스트나 설계 문서가 결과로 나타나며, report formatter, lexical analyzer, scheduler, retriever, expression evaluator, interpreter 등이 해당된다. "functional area" 항목은 절차(procedure)로써 정의되어지는 활동으로 애플리케이션에 종속적인 활동이며, general ledger, cost control, purchasing 등이 이에 해당된다. "setting" 항목은 애플리케이션이 어디에서 실행되며, 어떤 연산들이 어떻게 수행되는지에 대한 설명을 한다. Print shop, chemical plant, department store가 여기에 해당된다. 이러한 Facet 정의를 통해서 정의한 것이 <표 2>와 같다.

〈표 2〉 Facet에 의해 분류된 리스트

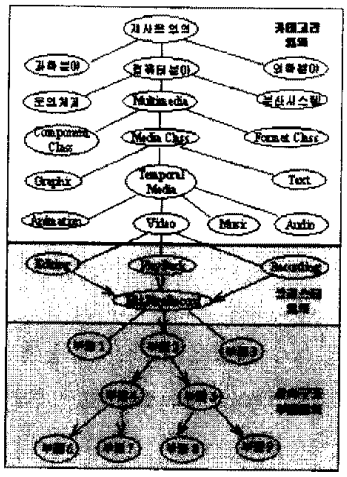
function	Objects	Medium	System Type	Functional Area	setting
add	Arguments	Array	Assembler	accounts pay-	advertising
compare	Characters	Disk	Compiler	able	appliance
create	Descriptors	Line	evaluator	auditing	auto repair
exchange	Files	List	file handler	bookkeeping	barbershop
expand	functions	Stack	line editor	budgeting	catalog
format	instructions	Table		customer	sales
input		Tape		DB manage-	
insert		Tree		ment	

이 GTE 재사용 시스템도 동의어 관리를 두어 사용의 편리성을 도모하고 확장성이 좋지만 객체지향 개념인 컴포넌트의 일반화/상세화의 상속개념 및 집단화의 포함개념을 표현하지 못하므로 객체지향 클래스 라이브러리 구성에는 알맞지 않다.

4.3 제안된 객체지향 클래스 라이브러리 설계 방법

제안된 설계 방법론은 개념적으로 Embedded Three-tired 계층 구조를 지원하고 있어 재사용자가 찾는 컴포넌트를 제일 상위 계층인 실세계 지식 기반 분류 영역인 Category 영역에서 적용할 도메인 영역을 선택하고, 선택된 Category 하위 계층인 유사 기능을 하는 컴포넌트 그룹의 클러스터를 선택하면 재사용할 후보 컴포넌트들이 브라우징되어 컴포넌트 계층구조가 보여지게 할 수 있다. 이는 Category, 클러스터, 컴포넌트들이 경로탐색 방법을 지원할 수 있는 정보를 가지고 있기 때문이다. 이러한 경로 탐색 방법의 개념적인 예가 (그림 9)와 같다.

다음은 대표적인 재사용 시스템의 클래스 라이브러리 구성방법 및 객체지향 C++ 클래스 컴포넌트 라이브러리로서 갖추어야 할 요소들로 비교 항목을 정하여 Intermetrics사의 RSL시스템과 GTE의 재사용 시스템 및 본 논문의 객체지향 클래스 라이브러리 설계 방법을 비교 검토 하였다. <표 3>의 비교 테이블을 살펴보면 제안된 설계 방법론이 객체지향 클래스 라이브러에 제일 합당한 조건을 지니고 있으며 단지 검색 방법에서 자연어 검색을 지원하지 못하는 점이 단점이다. 그러나 Intermetrics사의 RSL시스템과 GTE의 재사용 시스템은 객체지향 클래스 라이브러리로서 핵심 기능인 일반화/상세화의 상속개념 및 집단화의 포함개념을 표현하지 못하므로 객체지향 클래스 라이브러리 구성에는 알맞지 않다.



(그림 9) 클래스 라이브러리 경로탐색 개념 모델

또한 영역 분류 및 컴포넌트 분류 방법에 있어서도 각기 하나의 핵심 분류 방법에 의존하고 있는 반면 제안된 객체지향 클래스 라이브러리 설계 방법은 몇 개의 핵심 분류 방법을 통해서 재사용자에게 개념적으로 명확한 분류 체계를 지원하고 있고 객체지향 클래스 상속관계와 C++ 클래스 코드 재사용시 부품을 이해하기 위한 부속된 메타 정보를 제공해 주므로서 재사용 효율을 높여 재사용 시간을 단축 시킬 수 있어서 제안된 설계 방법이 객체지향 클래스 라이브러리로서 합당하다.

〈표 3〉 클래스 라이브러리 구성 방법 비교

시스템 / 방법	제안된 설계방법	Intermetrics RSL System	GTE Reuse System
Enumerative	Category	Category	X
Faceted	5개Facet term	X	6개Facet term
Clustering	Cluster of family grouping	X	X
Generalization	Inheritance	X	X
Aggregation	Reference	X	X
객체지향 개념	Ok	X	X
검색 방법	경로 탐색 Facet Keyword	Natural-language Keyword	개념 그래프 Facet Index

5. 결 론

본 논문에서는 객체지향 C++ 프로그램 코드를 재사용하기 위해서 객체지향 클래스 라이브러리 설계 방법

분에 대하여 분류하였다. 제안된 방법으로는 재사용 컴포넌트로서 갖추어야 할 정보의 종류를 정의하여 재사용시 이해정보로서 유용한 정보를 제공하고 재사용 시간을 단축 시키려고 하였다. 또한 도메인 분류 방법으로서 Category 분류 방법을 제안 하여 실세계 지식을 분류할 수 있게 구성하였다. 그리고 Category 영역에서 유사한 기능을 하는 부품들을 그룹핑하는 방법으로서 클러스터링에 의한 클러스터 구성방법을 제안하고 이 클러스터를 구성하기 위하여 클래스 유사도, 데이터 유사도 함수 유사도의 계산 방법과 그 임계값을 제안 하였다. 또한 한 클러스터내에서 유사도 관계와 상속관계를 표현하는 방법을 제안하였다. 이와 같이 제안된 분류 방법론을 통해서 소프트웨어 분류 방법의 대표적인 Enumerative 방법과 Facet분류 방법의 객체지향 분류 단점을 극복하였다.

앞으로 좀더 연구가 진행되어할 것은 부품 등록시 등록되는 컴포넌트의 품질 평가부분과 모든 프로그램 Language로 코딩된 부품을 등록할 수 있는 재사용 S/W 라이브러리 설계 방법에 대한 연구가 좀더 이루어져야 할 것이다.

참 고 문 헌

[1] T. Biggerstaff, A.J. Perlis, Software Reusability. Vol.1:Conecpt and Models, Addison-Wesley, 1989.

[2] P. A. V. Hall, Overview of Reverse Engineering and Reuse Research, Information and software technology 34, pp.239-349, 1992.

[3] Charles W.krueger, "Software reuse," ACM Computing Surveys, pp.131-183, June 1992.

[4] K. C. Kang, "A Reuse-Based Software Development Methodology," Tutorial on Software reuse : Emerging Technology, pp.194 196, 1990.

[5] Ruben Prieto-Diaz. A Software Classification Scheme. PhD thesis. University of California, Irvine, 1985.

[6] A. Gargaro, T. L. Pappas, "Reusability Issues and Ada," IEEE Software, pp.43-51, July 1987.

[7] B. Meyer, "Reusability : The Case for Object-oriented Design," IEEE Software, pp.50-64, March 1987.

[8] M. Lubars, "Domain Analysis and Domain Engineering in Ideal' Domain Analysis and Software System Modeling, IEEE CS Press, pp.167-178, 1991.

[9] L. Latour, "SEE : An Automated Tool to Facilitate Reuse in Ada Project Development," Proc. Of the Workshop on Software Reusability and Maintainability, pp.318-324, October 1987.

[10] W. Tracz, "Software Reuse : Motivators and Inhibitors," Proc. Of COMPCON, pp.358-363, Spring 1987.

[11] Tim Korson and jhon D. McGregor, "Understanding Object-Oriented : A Unifying Paradigm," Communications of the ACM Vol.33, No.9, pp.40-60, September, 1990.

[12] William B. Freakes and Ricardo Baeza-Yates, "Information Retrieval," Prentice Hall, pp.419-442, 1992.

[13] Michael S. Lewis-Beck and Roger K.Blashfield Quantitative Applications in the Social Sciences, Cluster Analysis, Vol.44. 1984.

[14] International Federation for Documentation. "Principles of the universal decimal classification," 1981.

[15] B. A. Burton, et al., "The Reusable Software Library," IEEE Software, Vol.4, pp.25-33, July 1987.

[16] S. R. Ranghanathan. Prolegomena to Library Classification. The Garden City Press Ltd., Letchworth, Hertfordshire, 1957.

[17] S. R. Ranghanathan. Prolegomena to Library Classification. Asia Publishing House, Bombay, India, 1967.

[18] Ruben prieto-Diaz and Peter Freeman. "Classifying software for reusability". IEEE Software, pp.6-16, January 1987.

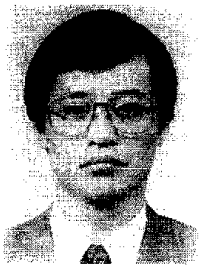
[19] R. Priteo-Diaz, "Implementing Faceted Classification for software reuse," Comm. ACM, Vol. 34, No.5, pp.88-97, May 1991.

[20] M. Dewey. "Decimal Classification and Relative Index," Frest Press Inc., Ny, 17 edition, 1965.

[21] L. S. Sorumgard, E. Tryggeseth, "Classification, Search and Retrieval of Reusable Software Com-

ponents," Norwegian Institute of Technology, May 22, 1992.

[22] Andy Podguski, Lynn Pierce, "Retrieving Reusable Software by Sampling Behavior," ACM transacation on Software Engineering and Methodology, Vol.2, No.3, pp.286-303, July, 1993.



이 해 원

e-mail : hewlee@etri.re.kr

1989년 대전산업대학 전자계산학과 졸업(학사)

1999년 대전대학교 대학원 컴퓨터공학과(공학석사)

1983년~현재 한국전자통신연구원 우정정보화팀 연구원

관심분야 : 소프트웨어공, 객체지향시스템(OOAD, 재사용), 분산시스템, 멀티미디어

김 진 석

e-mail : kimjs@etri.re.kr

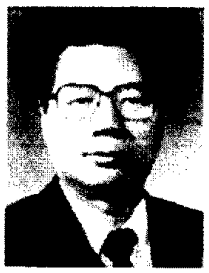
1982년 울산대학교 전자계산학과 졸업(공학과)

1988년 동국대학교 대학원 전자계산학과 졸업(공학석사)

1992년 정보처리 기술사

1982년~현재 한국전자통신연구원 우정정보화팀장(책임연구원)

관심분야 : CSCW, 멀티미디어 데이터베이스, 소프트웨어공학



김 혜 규

e-mail : hkkim@etri.re.kr

1973년 서울대학교 공과대학 응용물리학과(학사)

1985년 서강대학교 경영대학원 경영학과(석사)

1994년 서강대학교 공공정책대학원 정보처리(석사)

1979년~현재 한국전자통신연구원 우정기술연구부장(책임연구원)

관심분야 : 정보산업정책, 멀티미디어

하 수 철

e-mail : soocha@dragon.taejon.ac.kr

1981년 홍익대학교 컴퓨터공학과 졸업(학사)

1986년 홍익대학교 대학원 컴퓨터공학과(석사)

1990년 홍익대학교 대학원 컴퓨터공학과(박사)

1999년~현재 한국정보처리학회 논문지 편집위원

1998년~현재 한국정보처리학회 멀티미디어시스템연구회 부위원장

1997년~현재 소프트웨어연구센터(SOREC) 운영위원

1996년 한국전자통신연구원 초빙연구원

1991년~1992년 플로리다 주립대학교/텍사스 주립대학교 객원교수

1981년~1984년 Army Logistics Command. System Analyst

1987년~현재 대전대학교 컴퓨터공학과 교수

관심분야 : 소프트웨어공학(객체지향화), 멀티미디어응용, 게임공학, 시각언어, 프로토콜기술언어