HACM을 사용한 객체지향 재사용 부품의 분류와 검색

배 제 민[†] · 김 상 근[††] · 이 경 환[†††]

요 약

재사용을 지원하는 라이브러리 구축을 위해서는 다양한 응용영역에 적용할 수 있는 분류스킴과 검색방법이 필요하다. 본 논문에서는 재사용 단계의 접근성의 핵심을 이루는 분류스킴을 클러스터를 이용한 계층적인 구조를 통해 정의하였다. 또한 검색시스템의 기능과 정확도를 결정하는 라이브러리 구조에 클러스터링 정보를 첨가하여 부품의 표현방법과 클래스들간의 유사관계를 기술, 관리하는 방법을 제안하였다. 이에 따라 개발자에게 소프트웨어 부품의 인덱싱 및 스테밍 등을 통한 분류 및 검색 방법을 제공함으로써 재사용부품에 대한 탐색가능성을 높이고 재사용의 효과를 증진시키려한다. 그 결과로 재사용 라이브러리의 구축과정을 자동화하였고 기존의 문제점인 확장성과 관련성 모두를 고려한 분류스킴을 통하여 재사용라이브러리와 검색시스템을 구축하였으며 관련연구를 클러스터 계층도를 통해 시각화함으로써 탐색가능성에 대한 효과를 높였다. 또한 검색결과는 재사용시스템 CARS 2.1에 통합되었다.

Classification and Retrieval of Object-Oriented Reuse Components with HACM

Je Min Bae[†] · Sang Geun Kim[††] · Kyung Whan Lee[†††]

ABSTRACT

In this paper, we propose the classification scheme and retrieval mechanism which can apply to many application domains in order to construct the software reuse library. Classification scheme which is the core of the accessibility in the reusability, is defined by the hierarchical structure using the agglomerative clusters. Agglomerative cluster means the group of the reuse component by the functional relationships. Functional relationships are measured by the HACM which is the representation method about software components to calculate the similarities among the classes in the particular domain. And clustering informations are added to the library structure which determines the functionality and accuracy of the retrieval system. And the system stores the classification results such as the index information with the weights, the similarity matrix, the hierarchical structure. Therefore users can retrieve the software component using the query which is the natural language. The thesis is studied to focus on the findability of software components in the reuse library. As a result, the part of the construction process of the reuse library was automated, and we can construct the object-oriented reuse library with the extendibility and relationship about the reuse components. Also the our process is visualized through the browse hierarchy of the retrieval environment, and the retrieval system is integrated to the reuse system CARS 2.1.

† 종신회원:중앙대학교 컴퓨터공학과
†† 정 회 원:성결대학교 전산통계학과
††† 정 회 원:중앙대학교 컴퓨터공학과
논문접수:1995년 9월 17일, 심사완료:1997년 5월 2일

1. 서 론

소프트웨어의 재사용은 소프트웨어 품질과 생산성 향상을 위해 가장 유용한 기법중의 하나로 활용되고 있지만 이를 저해하는 몇가지 요소가 있다[6]. 첫 번째는, 과학적 또는 공학적 분야로서의 소프트웨어 개발이 부족하다는 점이고, 두 번째는, 소프트웨어 개발 특히 재사용에 있어서 훈련이 부족하다는 점이다. 세 번째는, 관리체계와 경험이 부족하며, 마지막으로, 방법론과 도구의 결핍이라고 할 수 있다. 지금 재사용에 대한 연구는 패턴과 프레임워크에 걸쳐 활발히 진행되고 있지만, 소프트웨어를 재사용하는데 있어서는 이들을 효과적으로 저장할 수 있는 분류메커니즘과 이를 적절히 관리하여 검색할 수 있는 방법이 중요하다고 생각된다.

재사용 시스템 구축에 있어 제기되는 문제점은 라이브러리의 대형화, 부품의 기술, 부품의 검색, 부품의 이해, 부품의 선택과 수정을 통한 통합 등이라 할 수 있다[14]. 본 논문에서는 객체지향 환경에서 정립된 객체의 개념적 모델에 기반한 부품모델을 바탕으로 문헌정보검색에서 사용되어진 HACM(Hierarchical Agglomerative Clustering Method)에 의해 영역내의 부품을 분류함으로써 소프트웨어의 분류와 검색에 대한 해결책을 제시하려 한다. 또한 라이브러리의 확장성과 부품간의 관련성 모두를 개선시킬 수 있는 방법을 제안하고 브라우징 기능을 제공하여 사용자가 재사용 클래스 부품 라이브러리 내에 기능적으로 관계된 후보 부품들을 시각적인 검색을 통해 보다 더 용이하게 선택할 수 있도록 하였다. 본 방법론은 재사용 대상을 클래스부품으로 한정하였지만, 소프트웨어 자체나 처리과정 즉, 소프트웨어 자산에 확대하여 적용할 수 있다.

본 논문의 구성은 다음과 같다. 2장은 기존의 재사용 라이브러리의 구조와 분류방법에 대한 연구를 살펴보고, 3장에서는 제안된 객체지향 라이브러리가 구축되는 과정과 관련된 기술을 설명하며, 4장에서는 검색환경과 이를 통한 시험 및 평가과정을 설명했으며, 마지막으로 5장에서는 결론을 기술하였다.

2. 관련 연구

일반적으로 재사용 시스템에서 재사용 효과를 증진시키기 위해서는 재사용 시스템의 구성 요소를 정의하고 소프트웨어 라이브러리의 구축에 필요한 여러 가지 분류방법, 부품기술 방법들을 적용하게 된다[14]. 본 논문에서는 이러한 문제점에 대한 해결책으로 객체지향 환경하의 지식기반 접근법과 정보검색방법 및 HACM을 통하여 라이브러리를 구축해 나가게 될 것이다. 우선 기존의 재사용 시스템에서 사용된 분류방법은 다음과 같다.

2.1 정보검색

기존의 분류방법 중 계층적인 분류방법[9]은 지식 전체가 모두 합성 가능한 클래스들을 포함할 수 있는 것으로 클래스들 사이의 계층적인 관련성 표현에 그 특성을 두고 있다. 이 방법은 부품의 수가 증가하게 되면 계층 구조가 복잡해지고 새로운 부품의 추가가 어렵다는 문제점이 있다. 반면에 패싯 분류방법[10]은 어떤 모듈들의 공통적인 측면의 속성들을 하나의 패싯으로 모아서 여러 패싯들 중 찾으려 하는 모듈에 알맞은 속성을 나타내는 원소들을 선택, 합성함으로써 특정 모듈을 검색하게 된다. 이 방법은 지속적으로 모듈이 확장되는 경우에 쉽게 대처할 수 있지만 각 부품들을 객체지향 특성에 부합되게 표현할 수 없고 부품들 간의 관련성을 쉽게 표현할 수 없다는 단점을 지니게 된다.

RSL(Reusable Software Library)[2]은 재사용 부품의 속성으로서의 재사용 주석을 몇 가지 정의하고 각각에 속성이름과 기능설명을 가진 레이블을 할당한다. 정보추출기는 적절하게 주석이 달린 부품의 PDL과 소스코드로부터 재사용에 필요한 정보를 추출하여 RSL 데이터베이스에 저장한다. 메뉴방식으로 처리되는 사용자 질의는 레이블에 있는 재사용에 필요한 정보를 바탕으로 작성되는데, RSL은 정보검색방법을 기반으로 하고 있으므로 자연어 형식의 질의가 가능하다는 장점을 지니고 있다. 검색 메커니즘은 사용자가 입력한 질의어의 키워드 집합을 받아들여 저장되어 있는 키워드와 비교한 후 일치되는 부품들을 찾아낸다. 검색 시스템에 의해 검색된 부품들은 사용자가 제공하는 선정 기준 즉, 자신이 적용할 응용 분야에 중요한 의미를 지니는 품질평가 항목의 중요도에 따라 다시 평가된 후 가장 적합한 부품을 제공한

다. 단점으로는 저장 시와 검색 시에 사용되는 키워드에 따라 시스템의 효율이 결정되게 된다는 점이다. 즉, 검색 시에 사용되는 키워드의 의미를 파악하기 어렵기 때문에 질의를 재구성한다던가 브라우징을 지원하도록 부품을 관리하기가 어렵다. 또한 순수한 정보검색방법으로 질의에 사용된 특별한 키워드의 의미를 추론하기도 어렵다.

CATALOG[3]는 비구조화된 자료를 처리할 수 있으며 구축된 데이터베이스 내의 각 모듈의 헤더기술서로부터 인덱싱에 필요한 정보를 얻어낸다. 검색은 초보자를 위한 메뉴 인터페이스와 전문가를 위한 명령어 위주의 탐색 인터페이스를 제공하고, 부울린 질의와 부분적인 스트링매칭 기법을 허용한다. 또한 의미없는 단어들을 정지리스트(stoplist)에 의해 처리하지만, 라이브러리의 확장성, 부품의 이해, 부품의 선택과 수정을 통한 통합 등의 문제를 해결하지는 못하였다.

Diaz[9, 10]의 분류 메커니즘은 모듈들이 갖는 여러 속성들을 각 패싯별로 나누어 이해하므로 모든 모듈들은 각 패싯에 해당하는 원소들을 조합시켜서 원하는 기능을 정의할 수 있고 분류대상에 따라 적절히 적응하기 쉬울 뿐만 아니라 지속적으로 모듈의 집합이 확장되는 경우에도 계층적 방법보다 쉽게 대처할 수 있다. 검색과 탐색을 지원하는 방법으로 개발된 프로토타입 시스템이 질의어 구성, 검색, 순서를 결정하는 시스템으로 구성되어 있고 원하는 부품에 대한 질의는 제시되는 분류 스킴을 구성하는 각 패싯에서 항을 선택하여 부품기술서(descriptor)를 형성한다. 선택된 항으로 구성된 질의는 검색에 사용되기도 하고 수정도 가능하게 되며, 이러한 수정은 새로운 기술서의 추가와 수정을 통하여 이루어지게 되고 이를 질의의 확장으로 정의하게 된다. 주어진 질의로 검색에 실패할 경우 기능이 비슷한 부품의 추출을 위해 동의어 관리 방법을 이용하여 새롭게 질의가 작성되도록 하여 사용자들이 비슷한 기능을 수행하는 부품중 하나를 선택하도록 한다. 결론적으로, Diaz에 의해 제안된 분류와 검색의 방법은 영역 모델이 설정되어 질의의 구성과 재구성에 사용되게 된다. 그러나 이는 한번 분류가 설계되고 나면 고정적이 된다는 제한성을 지닌다.

재사용에 대해 완전한 방법론을 제공하기 위하여 REBOOT(REuse Based on Object Oriented Techniques) 환경[7]은 재사용부품들이 저장된 데이터베이스와 재사용부품들을 구축·품질보증·삽입·검색·평가하기 위한 도구들을 통합하였다. 분류스킴으로는 패싯방법을 사용하였고, 전체구조는 재사용도구들, 데이터베이스 인터페이스, 데이터베이스 플랫폼, 통신관리자, 통합도구들과 같이 5개부분으로 나누어진다. 재사용도구 부분에는 재사용을 보조하는 도구들로 검색도구, 적용도구, 평가도구가 있고, 부품구축을 위해서는 품질평가도구, 재공학도구, 분류도구가 있다. 또한 탐색자(Navigator)를 통해 데이터베이스내의 재사용부품들을 브라우징할 수 있도록 하였다. 분류도구는 사용자가 분류트리를 브라우징함으로써 적당한 분류용어를 선택하도록 하고 부품과 선택된 용어사이의 관계를 데이터베이스에 설정한다. 분류트리의 각 노드는 패싯을 의미하고 부품을 분류하기 위해선 그 부품을 특성 짓는 용어를 찾아서 해당 노드에 첨가시킨다. 검색도구 역시 브라우저 기능을 통해 이루어진다. 사용자는 현재 소프트웨어 응용영역에서 유용한 분류기준을 사용하여 복잡한 요청을 할 수 있다. 부울린 연산을 통해 용어들은 기본연산을 형성할 수 있고 결과 값은 주어진 상태를 만족하는 부품들의 리스트 형태로 보여준다. 결과적으로 소프트웨어 재사용에 대해 통합된 환경을 제시하고 있으나 분류 및 검색 메커니즘에 있어서는 기존의 패싯방법을 사용하므로 재사용부품들의 객체지향 특성을 반영하기 어렵고 그 관련성을 표현할 수 없다는 단점이 있다.

2.2 정보분류방법

도큐먼트 클러스터링은 검색의 효율성과 효과성을 증진시키는데 있어 그 잠재적인 능력으로 인해 연구가 되어왔고 부울린 검색이나 최상매치 검색의 대안책을 제공했다. 처음에는 효율에 중점을 두어 도큐먼트 집합이 분할되었으며 이는 질의와 도큐먼트 사이의 비교 수를 감소시킬 수 있었다. 효율에 대한 문제가 어느 정도 해결되자 검색 효과가 저하되기 시작했다. 이에 검색 효과를 증진시키고자 클러스터설에 기초한 계층적인 방법에 대한 연구에 집중되었다. 문헌정보검색방법에서는 다양한 형태의 클러스터 분석방법이 소개되었다. 이는 단일패스나 재할당방법과 같이 계산 량이 적고 본질적으로 경험적인 요소가 많

이 담긴 비계층적인 방법과 단일링크, 완전링크, 그룹평균, 워드의 방법과 같이 공간과 시간은 많이 요구하나 클러스터를 기반으로 하는 도큐먼트 검색에 더 적합한 계층적인 방법으로 나누어진다. 클러스터 분석에 대한 대부분의 초창기 연구는 단순한 계층적인 방법에 대해 이루어졌지만 컴퓨터 자원이 증가하고 클러스터 분석에 대한 소프트웨어 펙키지의 쉬운 유용성과 향상된 알고리즘으로 인해 정보검색에 있어서 클러스터링에 대한 연구는 HACM(Hierarchical Agglomerative Clutering Method)에 집중되었다.

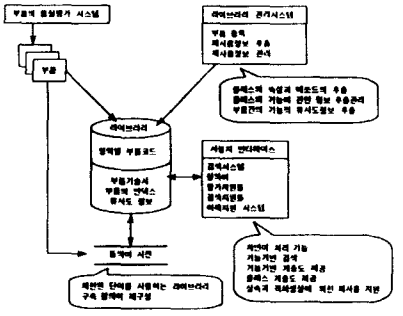
HACM을 사용한 여러 방법들은 덴드로그램 형태의 클러스터 구조를 그 결과로 얻게된다. 덴드로그램이란 간단한 트리형태의 다이어그램을 의미하며 이는 자료집합내의 객체들 중 가장 유사한 짝을 보여주며 유사도 함수의 값은 각각의 결합때 발생하게 된다. 형성되는 그룹은 같은 그룹의 멤버일 경우 높은 관련성을 가져야 하고 다른 그룹의 멤버사이에는 낮은 관련성을 가져야 한다. 이러한 덴드로그램은 도큐먼트들의 클러스터화된 집합으로부터 검색시 유용한 표기이다. 왜냐하면 검색 과정이 따라야 할 경로를 알려주기 때문이다. 자료집합들을 덴드로그램으로 나타내기 위해서는 자료집합내의 아이템들을 클러스터화할 수 있는 측정방법 즉, 거리측정 또는 유사도 측정방법[13]이 필요하다. 여러 가지 측정 방법들 중 선택된 측정 방법에 따라서 얻어지는 클러스터링 결과에 영향을 미치게 된다. 모든 측정방법들의 결과인 유사도 매트릭스는 좌우대칭형이므로 클러스터링 정보를 얻기에는 하삼각행렬의 값만으로 충분하다. 유사도 매트릭스는 가장 가까운 이웃(nearest neighbor, NN)을 나타내는데 기초가 될 수 있다. 즉, N개의 다차원벡터들의 집합으로부터 주어진 벡터에 가장 근접한 벡터를 찾는데 사용될 수 있다. NN의 표기는 여러 클러스터 알고리즘에서 나타나는데 커다란 자료집합에 대한 계산량에 의미 있게 사용된다.

기존의 방법들[4, 11]을 설명하면 단일링크는 가장 널리 사용되는 방법으로 새로운 클러스터를 이루는 부품들은 클러스터내의 가장 높은 유사도를 가진 도큐먼트를 기준으로 선택한다. 이 방법은 한 클러스터내의 최소한 하나의 멤버가 또다른 클러스터내의 모든 멤버들과 비교해서 결합이 이루어지는 클러스터에대해 높은 유사성을 갖고 있게 된다. 특징은 내부적으로 응집력이 약하고 느슨하게 결합된 클러스터를 형성하는 경향이 있다. 반면 완전링크는 단일링크와는 달리 클러스터내의 가장 유사도가 낮은 도큐먼트를 기준으로 하여 새로운 클러스터를 구성하는 방법이다. 즉 한 클러스터내의 한 멤버와 다른 클러스터내의 한 멤버사이의 가장 낮은 유사도가 적어도 또 다른 클러스터내의 멤버와의 최저 유사도보다는 높은 클러스터들을 결합시키는 방법이다. 이 방법을 사용했을 경우 적은 수의 강하게 결합된 클러스터를 형성하게 된다. 그룹평균은 이름이 말하듯이 한 클러스터내의 가장 유사한 링크의 평균값이 사용된다. 모든 객체들이 클러스터 내부의 유사도를 결정짓는데 참여하고 단일링크와 완전링크 사이의 중간 형태의 구조를 그 결과로 보여준다.

3. 객체지향 라이브러리의 구축 방법론

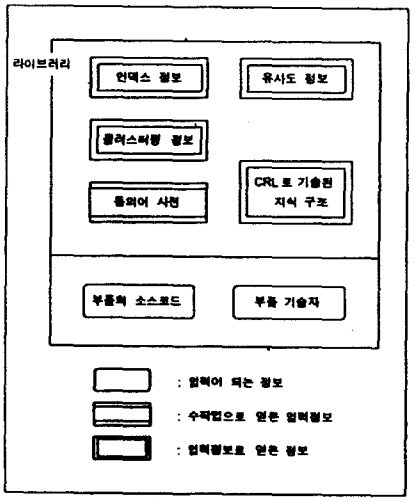
CARS(Computer Aided Reuse System) 2.1은 윈도우환경에서 구축된 재사용시스템으로, 라이브러리 관리시스템, 검색시스템, 사용자 인터페이스, 그리고 품질평가시스템의 4가지 서브시스템으로 구성된다. 응용영역은 그래픽스, 자료구조, 윈도우 및 커널 라이브러리가 구축되었고 그 구조는 (그림 1)과 같다. 라이브러리 관리시스템은 재사용 라이브러리 관리자와 품질보증 시스템이 라이브러리를 관리하는데 도움을 주는 도구들의 집합으로 구성된다. 라이브러리 관리시스템의 기능은 부품과 관련된 재사용 정보를 추출하고 품질평가시스템을 통해서 라이브러리에 저장될 부품의 품질이 정해진 기준에 적합한 지를 확인한 후, 품질이 보증된 부품을 저장, 관리한다. 사용자 인터페이스와 검색시스템을 통해 사용자가 요구하는 기능을 만족하는 부품을 식별하고 다른 요구에 대한 비교를 통해 적절한 소프트웨어 부품을 선정하도록 도와준다. 즉, 브라우저를 통한 소프트웨어 분류계층과 함께 부품을 단계별로 이해하도록 도와줌으로써 원하는 부품을 찾을 수 있도록 해준다. 정확히 원하는 부품이 라이브러리에 없는 경우 분류 계층의 상위 클래스를 선정할 수 있도록 지원하고 있다. CARS 2.1은 객체지향 모델을 기반으로 하고 있으므로 라이브러리의 구조에 대한 지식을 얻을 수 있다. 즉 객체지향 방법론으로 개발된 부품의 분석을 통해 부품들 간

의 관련성을 자동으로 추출한다. 하지만 라이브러리 구축과 검색시스템은 기본적으로 지식기반 접근법을 따르고 있기 때문에 본 논문에서는 정보검색방법을 혼용한 새로운 방법을 제시하려 한다. 그 결과로 CARS 2.1의 라이브러리 구축과 검색과정의 자동화와 기능의 확장이 이루어졌다.



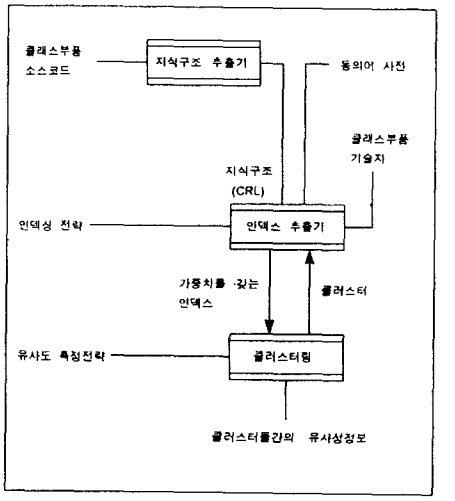
(그림 1) CARS 2.1의 구조
(Fig. 1) Architecture of CARS 2.1

CARS 2.1의 지식기반 접근법에 의해 구축된 객체지향 클래스 라이브러리는 Eiffel 라이브러리[5]와 유사한 상속관계를 이용한 검색 메커니즘을 제공하고 있다. (그림 2)는 CARS 2.1에 새로이 제안한 정보가 추가되어 확장된 라이브러리 구조도를 보여주고 있



(그림 2) 확장된 라이브러리 구조
(Fig. 2) Extended library structure

고 (그림 3)은 이러한 정보를 이용하여 재사용 라이브러리를 구축하는 과정이 나타나 있다. 라이브러리 구조도에서 추가된 정보는 클러스터링 정보이고 구축과정에서는 본 논문에서 제안한 유사도 측정전략에 의해 클러스터링하는 과정과 클러스터들 간의 유사도 정보가 추가되었다. 그러면 이러한 정보들을 얻어내는 과정을 살펴보도록 하겠다.



(그림 3) 라이브러리 구축과정
(Fig. 3) Construction process of library

3.1 부품의 인덱싱

정보검색 시스템은 그 적용영역에 따라 다양한 검색환경이 설정되는데 그 요인을 알아보면, 첫째 인덱싱의 자동화 여부이고, 둘째 정규적인 인덱스항의 사용 여부이며, 셋째 단일 항으로 구성된 인덱스인지 구문으로 표현되는 인덱스인지를 결정해야 한다. 본 논문에서는 제한된 단일 인덱스 항으로 인덱싱의 자동화를 이루어나갔으며 전체 인덱싱 과정은 각 문서마다 부품의 프로화일을 구성하게 될 인덱스들의 집합이 생성되고 부품의 기능을 식별할 수 있는 인덱스 항을 할당, 선정한 후 기능을 식별하는데 있어서 중요도에 따라 가중치를 결정하는 것이다.

(1) 인덱스 항목의 추출

이 과정에서는 부품의 특성을 나타낼 수 있는 인덱스 항목의 추출을 위해 부품의 기능을 기술한 문서와

주석의 구문적 분석을 통해 각 인덱스 항을 결정하고 그 빈도수를 파악한다. 우선 객체지향 라이브러리를 구성하는 클래스 부품은 그 고유의 특성으로 인해 자료부분과 자료관리에 필요한 오퍼레이션 부분으로 구성되므로 클래스를 특성짓는 인덱스는 다음과 같은 형태로 표현된다.

$Comp_i = (Domain\text{-}index_i, Operation\text{-}index_i)$

$Comp_i$: 임의의 클러스터에 저장된 부품 i의 프로화일명

$Domain\text{-}index_i = (Term_1, ..., Term_n)$

$Operation\text{-}index_i = (Term_1, ..., Term_n)$

$Term_i$: 인덱스항

영역 인덱스(Domain-index)는 클래스의 정의된 자료구조에 대한 특성을 기술한 인덱스들의 집합이고, 오퍼레이션 인덱스(Operation-index)는 클래스에 정의된 오퍼레이션의 특성을 기술한 인덱스들의 집합을 의미한다.

```
For each Component Ci stored in Cluster CL
  For each sentence S in domain document Dd of Component Ci
    t <-- lemma (t)
    If t is an opened-class word
    then
      check the presence of t in domain index set
      If (frequency count of t == 0)
      then
        store t into domain-index set
        increase the frequency count of t
      else
        increase the frequency count of t
      EndIf
    EndIf
  EndFor
  For each sentence S in operation document Dd of Component Ci
    t <-- lemma (t)
    If t is an opened-class word
    then
      check the presence of t in operation index set
      If (frequency count of t == 0)
      then
        store t into operation-index set
        increase the frequency count of t
      else
        increase the frequency count of t
      EndIf
    EndIf
  EndFor
  For total index set of component in cluster CL
  ( Ci : 0 < i <= n )
    get the total frequency of each index in cluster CL
  EndFor
EndFor
```

(그림 4) 인덱스 추출 절차

(Fig. 4) Procedure of extracting index

각 인덱스 항은 문서의 구문적 분석(lemma 함수)을 통해서 의미있는 단어(명사, 동사, 형용사)와 단어의 원형을 취하여 추출한다. 예를 들면, boxes는 box로 deleted는 delete로 해당 단어의 원형을 인덱스로 선정하게 된다. 인덱스 선정과정에서 각 인덱스의 빈도수, 단일항(opened-class word)의 빈도수 f_{ik} (i:부품명, k:인덱스 항)를 구하는 이유는 부품 기능 설명서에 자주 나타나는 단어가 그 부품의 기능을 특성 짓는 중요한 인덱스 항으로 간주한다는 가정 때문이다. 결국 (그림 4)의 인덱스 추출 절차에 의해 얻어진 각 부품에 대한 인덱스는 다음과 같은 형태로 표현이 된다.

$$Comp_i = (Term_{i1}, Term_{i2}, ..., Term_{in})$$

또한 한 라이브러리 영역내의 모든 클래스 부품에 대해 추출된 인덱스와 그 빈도수는 다음과 같은 벡터 모델로 표현이 된다.

$$\begin{array}{c} \\ Comp_1 \\ Comp_2 \\ \cdot \\ \cdot \\ Comp_m \end{array} \begin{array}{c} \begin{array}{cccc} Term_1 & Term_2 & \cdots & Term_n \end{array} \\ \begin{bmatrix} Term_{11} & Term_{12} & \cdots & Term_{1n} \\ Term_{21} & Term_{22} & \cdots & Term_{2n} \\ \cdot\cdot & \cdot\cdot & \cdots & \cdot\cdot \\ \cdot\cdot & \cdot\cdot & \cdots & \cdot\cdot \\ Term_{m1} & Term_{m2} & \cdots & Term_{mn} \end{bmatrix} \end{array}$$

지금까지 설명한 과정은 (그림 4)와 같은 절차에 의해 이루어지게 된다.

(2) 동의어 사전(Term Dictionary)의 작성

추출된 인덱스들을 살펴보면 동일한 의미를 지니는 인덱스가 부품을 작성한 프로그래머의 경험에 따라 서로 다른 형태의 단어들로 표현된 경우를 볼 수 있다. 이러한 단어들은 하나의 대표어로 바뀌게 되는데 이는 라이브러리 사용의 관점에서 또 라이브러리 확장의 관점에서 어느 특정 영역에 국한된 이름의 사용보다는 광범위한 영역에 적용될 수 있는 기준을 마련하여 제한된 이름의 사용을 유도하는 것이 바람직하기 때문이다. 결국 본 논문의 검색환경에서는 정규표현의 사용을 전제로 하므로 동일한 의미를 지니는 여러 단어들은 제한된 단어로 대체된다. 이를 지원하기 위하여 인덱스 추출과정의 결과를 바탕으로 동일한 의미의 인덱스, 더 넓은 의미를 지닌 인덱스, 더 좁은 범위의 인덱스들을 대표할 수 있는 인덱스로 동의어 사전을 구축하였다. 이는 자동으로 추출된 인덱스를 바탕으로 라이브러리 관리자에 의해 이루어지게 된다. 객체지향 파라다임이 지닌 특성 중의 하나인 클래스 이름이나 클래스를 구성하는 메쏘드 이름들의 정의시 비정규적인 이름의 사용보다는 범용 적이고 제한된 단어의 사용을 유도한다는 점에서 라이브러리 관리자에 의한 동의어 사전의 작성은 자연스러운 작업으로 볼 수 있다. 그래픽 라이브러리에 대해 구축된 동의어 사전은 (그림 5)와 같은 과정을 통해 추출된 인덱스중 관련 인덱스에 속하는 인덱스를 대표 인덱스로 대체하고 관련 인덱스의 빈도수를 대표 인덱스의 빈도수로 합산하게 된다.

```
For each index file of Component Ci
  For each index Termj in index file
    If Termj is related term
      replace the Termj with prime term
      accumulate the frequency of prime term
    EndIf
  EndFor
EndFor
```

(그림 5) 동의어사전 구축 절차
(Fig. 5) Procedure of constructing term dictionary

(3) 가중치 할당

지금까지는 부품에서 추출된 인덱스 항들과 그 빈도수로 각 부품의 기능을 표현하였다. 기존의 문헌검색 시스템에서는 항의 중요도에 따라 가중치를 부여하지 않는 이진 인덱싱 방법을 사용하였고 가중치가 없는 항의 사용은 인덱싱 과정에서 간단하다는 장점을 지니지만 검색의 단계에서는 복잡함을 지니게 되고 질의에 의해 검색된 문서들 사이의 차이점을 발견하기 어렵다. 따라서 본 논문에서는 가중치를 부여하는 방법을 채택함으로써 검색된 정보들 간의 차이점을 나타내려고 한다. 항에 가중치를 부여하는 방법에는 해당 응용분야의 전문가들이 수작업으로 항을 추출하고 가중치를 부여하는 방법이 있으나 이러한 작업은 정보의 양이 방대해질수록 오류가 많아지고 어려운 작업이 되므로 이를 보완하기 위한 방법으로

부품의 인덱스 항들의 빈도수를 인덱스의 가중치로 파악하는 방법을 택할 수 있다. 즉 등장횟수 f_{ik}를 그대로 가중치로 보는 것인데 이것은 모든 인덱스 항들이 거의 모든 부품에서 같은 빈도수를 갖고 있을 경우 정량적 기준에 의해서 인덱스 항들 간의 중요도 차이를 파악하기 어렵고 각 클래스 부품마다 같은 인덱스 항의 빈도수가 높게 나타날 경우 그 인덱스가 각 부품을 특성 짓는 항이라고 보기 어렵다는 문제점을 갖고 있다. 이런 현상을 줄이기 위한 또 다른 요소로 한 인덱스가 속한 부품의 개수 C_k를 고려하였다. C_k와 각 인덱스의 가중치는 다음과 같은 방법으로 측정이 된다.

$$C_{ik} = \begin{cases} 1 \text{ (if } f_{ik} > 0) \\ 0 \text{ (otherwise)} \end{cases}$$

$$C_k = \sum_{k=1}^{n} C_{ik}$$

(n : 클러스터를 구성하는 부품의 개수,
C_k : 인덱스 k가 나타나는 클러스터내 부품의 개수)

$$W_{ik} = f_{ik}/C_k$$

이는 높은 가중치를 갖는 항은 어느 특정 부품에서는 높은 등장횟수를 갖지만 전체 클러스터에서의 빈도는 낮게 나타남을 의미한다. 이와 같이 만들어진 각 인덱스 항에 대한 가중치를 바탕으로 각 부품을 표현한 벡터모델의 내용이 변경된다. 즉 각 부품에 대한 인덱스 표현은 아래와 같이 각 인덱스에 대해 가중치를 갖는 형태로 바뀌게 된다. 이렇게 가중치를 갖는 부품의 인덱스 벡터모델을 바탕으로 라이브러리 영역내의 클래스 부품들을 분류해 나가게 되는 것이다.

$Comp_i = (Term_{i1}, Term_{i2}, ..., Term_{in})$
$Term_{ij}$: 부품i에 할당된 인덱스j의 가중치 값

3.2 부품의 분류

본 논문에서는 라이브러리의 확장성과 검색의 효과를 증진시키기 위해 재사용 부품의 탐색가능성을 증가시킬 수 있는 HACM을 적용한 라이브러리 구축에 초점을 맞추었다. 이를 위해 우선 일반적인 클러스터링 방법들은 보통 그들이 제공하는 클러스터 구조의 형태에 따라 구분되어진다. 그리고 분리적인 방법으로 알려져 있는 간단한 비계층적인 방법들은 N개의 객체로 구성된 자료집합을 M개의 클러스터로 나누며 중복은 허용하지 않는다. 반면 좀더 복잡한 계층적인 방법들은 아이템들이나 클러스터들의 쌍이 자료집합내의 모든 아이템들이 연결될 때까지 연속적으로 링크 되어지는 중첩된 자료집합을 제공하게 된다. 계층적인 방법들은 클러스터화가 이루어지지 않은 자료집합으로부터 시작해서 N−1번의 결합이 이루어지는 응집적(agglomerative)인 방법과 하나의 클러스터내의 모든 객체들로부터 시작해서 더 작은 클러스터로 N−1번의 분할이 진행되는 분할적(divisive)인 방법이 있다. 분할 적인 방법은 잘 이용되지 않고 유용한 알고리즘도 거의 없는 형편이며 본 논문에서는 단지 응집 적인 방법들만이 사용되었다.

일반적으로 NN을 사용하여 계층적인 분류를 하기 위한 단순 과정[4]은 첫 번째로 내부 도큐먼트들 사이의 유사도 값을 계산하고 두 번째로 각 도큐먼트들을 자신의 클러스터에 포함시키며 세 번째로는 현재 구성된 클러스터들 중에서 가장 유사한 짝을 합침으로써 새로운 클러스터를 형성하는 것으로 이는 클러스터의 레벨화를 의미하며 마지막으로 클러스터의 수가 하나가 될 때까지 세 번째까지의 과정을 반복하게 된다. 다양한 HACM이 위의 과정 중에서 세 번째 단계가 수행되어지는 방식에 따라 구분되어진다. 다시 말해 가장 유사한 클러스터쌍을 선택하는 기준과 하나의 클러스터를 표현하는 방법에 따라 달라지는 것이다.

여러 클러스터링 방법들중 하나를 선택한다는 것은 그 결과를 결정하는데 영향을 미칠 것이고 이에 따른 알고리즘의 선택은 그 효율을 결정하게 될 것이다. 클러스터링 방법이나 그에 따른 알고리즘을 선택할 때 두 가지 의문점이 생기게 된다. 하나는 선택된 클러스터링 방법이 특정 자료집합에 적당한 것인가 하는 점이고 또 하나는 나온 결과가 정말로 그 자료를 특성 지었다고 할 수 있는 가 하는 점이다. 첫 번째 의문점은 클러스터링 방법들에 대한 평가 연구를 통해서, 두 번째 의문점은 클러스터링 방법에 의해 나타난 결과의 확인검사를 통해서 풀릴 수 있다. 이들 방법에 대한 평가는 원래의 자료를 교란시킬 여러 가

지 에러상황에서 안정성을 유지하고 복구인덱스를 사용하여 얼마만큼 진정한 구조로 회복될 수 있냐는 정도로 결정하였다. 많은 정보 검색 환경에서의 자료 집합들은 동적 특성을 갖는다. 즉, 새로운 아이템들이 기본 자료들에 추가되고 드문 경우지만 기존의 아이템들이 삭제된다. 거대한 자료를 클러스터화하는 것은 많은 자원을 요구하기 때문에 전체 자료집합을 재클러스터화 하지 않고 클러스터 구조를 수정할 수 있는 메커니즘이 필요하지만 전반적으로 클러스터 구조의 유지에 관한 연구는 아직까지는 별로 이루어지지 않은 상태이다. 단일링크 방법이 여러 가지 기준을 만족하고 알고리즘이 간단하여 받아들이기에는 적합하지만 제안된 기준들이 너무 엄격하고 실제 응용영역에서는 부적합하다고 판명되었다. 한편 시뮬레이션 결과는 그룹 평균 방법이 가장 좋은 복구성능을 나타냈지만 각각의 방법들이 나름대로의 장단점을 가지고 있다고 한다. 이와 같이 기술적으로 만들어진 자료구조를 사용하고 그 분야의 전문가에 의해 만들어진 분류와 여러 클러스터링 방법을 적용하여 나온 결과들을 비교하는 평가 연구가 이루어졌지만 현재 일반적인 연구실 상태에서는 각 방법들을 평가하기란 어렵고 이러한 연구의 결과 역시 단지 하나의 방법만이 좋다고 평가하고 있진 않다.

이와 같은 분류의 결과로 나온 계층도 상의 도큐먼트들이 질의와 부합되는 방법은 여러 가지가 있다. 그 중에 상향식 탐색[12]은 트리의 루트로부터 각 노드의 클러스터와 부합시켜 유사도가 가장 크게 나타나는 경로를 따라 내려가게 된다. 탐색은 일정 기준에 도달하면 끝나는데, 예를 들면 클러스터 크기가 일정 도큐먼트 수 이하로 떨어진다든지 질의 대 클러스터간의 유사도가 감소하기 시작할 경우에 끝내는 것이다. 탐색의 결과는 하나의 클러스터가 택해지게 되며 상향식 탐색은 완전링크 방법에 잘 나타나 있다. 하향식 탐색은 트리의 기초가 되는 도큐먼트나 클러스터로 시작하여 검색의 한계를 만족할 때까지 이동을 하게 된다. 시작 도큐먼트는 탐색에 앞서 관련이 있다고 알려진 도큐먼트일 수도 있고 도큐먼트들이나 가장 하위의 레벨에 있는 클러스터들을 최상 매치 탐색방법으로 얻은 것일 수도 있다. 비교연구[13] 결과 완전링크 방법을 제외하곤 하향식 탐색이 좋은 결과가 나타났으며 특히 탐색이 하위레벨의 클러스터들에 제한되어 있을 경우는 더욱 그렇다고 한다. 각 방법에 대한 간단한 검색 메커니즘은 NN 클러스터들에 기초하고 있다. 또한 클러스터 표기는 하나의 클러스터내의 도큐먼트들의 특성을 나타내는데 사용되어진 기록이라 할 수 있다. 이는 검색시 필요한데 그 이유는 질의와 클러스터 간의 유사 정도를 결정할 수 있기 때문이다. 또한 비계층적인 방법에서 필요한데 도큐먼트와 클러스터간의 유사도가 도큐먼트를 가장 유사한 클러스터에 첨가하기 위해서 결정되야 하기 때문이다.

사용자가 라이브러리 내에 원하는 부품으로 접근시 사용자의 요구에 완전히 부합되는 부품이 없을 시 데이타베이스관리시스템은 사용자를 지원하지 못하지만 라이브러리 분류단계에서 비슷한 속성을 지니는 부품들을 하나의 클러스터로 모아서 사용자 요구에 가장 근접하고 가장 수정이 용이한 부품의 제공을 위해 기능이 비슷한 부품끼리의 브라우징 기능을 제공했을 경우 이러한 문제점은 어느 정도까지 해결될 수 있다. 또한 본 연구에서는 브라우징 계층과 클래스 계층이 일치할 필요는 없는 것으로 정의한다. 즉 클래스 계층은 클래스 구현의 결과로 간주하고 브라우징 계층은 기능성의 유사관계를 나타내는 것으로 정의한다. 이를 위해서는 우선 브라우즈 계층도상의 부품들 사이의 유사관계를 측정할 수 있는 유사도 측정 방법이 필요하다. 문헌정보기법에서는 분류를 위해 다양한 유사도 측정방법이 제안되었으나 본 논문에서는 인덱스를 벡터모델로 표현하고 표현된 인덱스의 가중치를 바탕으로한 벡터연산을 사용한다. 다음 식은 부품들 사이의 기능적인 유사관계를 측정하는 방법을 나타낸다.

$$\frac{\sum_{k=1}^{n} (\mathrm{Term}_{ik} \times \mathrm{Term}_{jk})}{\sum_{k=1}^{n} \mathrm{Term}_{ik} + \sum_{k=1}^{n} \mathrm{Term}_{jk} - \sum_{k=1}^{n} (\mathrm{Term}_{ik} \times \mathrm{Term}_{jk})}$$

종전에는 이렇게 나온 부품들간의 유사도 관계가 벡터모델로 표현되고 이것이 부품 분류의 결과가 되었으나 본 논문에서는 이 벡터모델을 이용하여 부품들간의 클러스터화를 이루어 나가게 되고 그 결과로 나오는 클러스터 계층도가 부품 분류의 결과가 되며 이를 바탕으로 검색이 이루어지게 된다. 클러스터링

이란 관련이 있는 도큐먼트들을 그룹핑해서 하나의 집합으로 만드는 것으로, 본 논문에서 사용한 클러스터링 방법은 부품들간의 유사도 측정방법에 의해 하위 클러스터들이 구축되면 하위 클러스터들에 대한 새로운 인덱스들이 생성되어 이들 인덱스간의 유사도를 측정하여 가장 밀접한 관계를 가진 클러스터들을 합침으로써 새로운 클러스터를 만들어가게 된다. 즉, 클러스터내의 두 원소에 유사도 행렬의 i와 j에 해당되는 행과 열을 없애고 새로운 클러스터 i +j에 해당되는 행과 열을 계산하여 생성시킴으로써 유사도 행렬을 갱신하게 된다. 이 응집 적인 클러스터화 전략은 N개의 도큐먼트 집합에 대해 총 N−1의 결합이 이루어지며 상호연관성이 높은 도큐먼트들로 이루어진 작은 클러스터들이 상호연관성이 적은 도큐먼트들로 이루어진 그보다 더 큰 클러스터들에게 연속적으로 중첩된 형태의 계층적인 구조가 나타나게된다. 이는 처음 하나의 클러스터가 연속적으로 더 작은 도큐먼트의 그룹으로 나누어지는 분할 적(divisive) 클러스터화 방법(procedure)과 비교가 된다. 클러스터 계층도를 만들어 가는 과정은 (그림 6)에 나타나 있다. 이와 같은 절차를 통해 재사용 부품은 분류가 이루어지고 검색시 (그림 10)과 같은 트리형태로 사용자에게 보여지게 되는 것이다.

4. 검색 시스템의 구축과 시험

본 장에서는 기존의 지식기반 접근법과 3장에서 제시한 정보검색방법에 따라 구축된 객체지향 라이브

For each component C_i in Library Domain
  make the index term
EndFor
compute the weight for each index
// C_i= ($Term_{i1}$, $Term_{i2}$, $Term_{i3}$, ... , $Term_{in}$)
For each component C_i in Library Domain
  compute simularity among component C_i & C_j

$$SIM\ (Comp_i\ ,\ Comp_j) = \frac{\sum_{k=1}^{n}(Term_{ik} \times Term_{jk})}{\sum_{k=1}^{n} Term_{ik} + \sum_{k=1}^{n} Term_{jk} - \sum_{k=1}^{n}(Term_{ik} \times Term_{jk})}$$

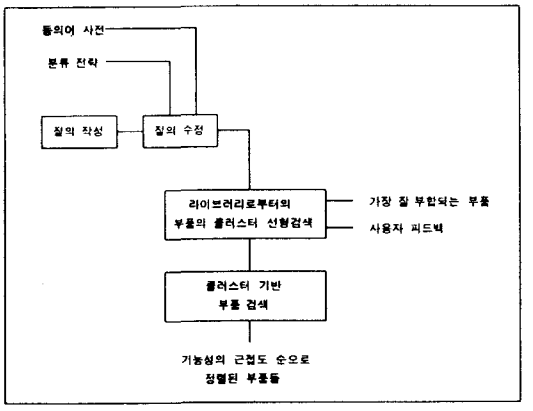
EndFor
S := set of clusters;
n := number of clusters
For |S| <> 1
  choose two clusters CL_i , CL_j which have highest similarity
  build new Tree(T) with root CL_{n+1} which has two child CL_i , CLj ;
  S := S - {CL_i , CL_j} ;
  compute the weight for the index of new cluster CL_{n+1}
  // CL_{n+1} = ($Term_{i1}$, $Term_{i2}$, ... , $Term_{in+1}$)
  compute the similarity between CL_{n+1} and clusters in S
  delete the rows and columns corresponding to CL_i and CL_j
  S := S ∪ CL_{n+1} ;
  n := n+1 ;
EndFor

(그림 6) 클러스터 계층도 생성 알고리즘
(Fig. 6) Procedure of generating cluster hierarchy

러리의 재사용 환경을 지원하는 재사용 시스템의 일부로 구현된 검색 시스템을 설명하고 검색의 실제 결과를 기술한다.

4.1 클러스터 기반 부품의 검색 과정

클러스터를 기반으로 재사용 부품을 검색해 나가는 전체적인 과정은 (그림 7)과 같으며 부품의 검색은 재사용자로부터 원하는 부품에 대한 질의를 받아 이를 바탕으로 라이브러리를 검색하고 그 결과를 제시하게 된다.



(그림 7) 부품의 검색과정
(Fig. 7) Retrieval process of component

사용자가 명시한 질의를 부품기술서와 동일한 형태로 간주하고 부품의 인덱싱 과정을 그대로 적용하여 질의로부터 빈도수를 갖는 인덱스를 추출한 후 동의어 사전의 대표인덱스로 인덱스 항의 대체가 이루어지며 이를 바탕으로 가중치를 갖는 인덱스를 얻게 된다. 얻어진 인덱스는 다음과 같은 형태로 재구성된다.

$Query = (Q-Term_1, Q-Term_2, ..., Q-Term_n)$
$Q-Term_i$: 질의를 구성하는 인덱스의 가중치

위와 같은 방법으로 가중치가 할당된 인덱스 항이 추출되면 다음과 같은 질의와 부품간의 유사도 측정 방법에 의해 해당 라이브러리 영역내의 정의된 클러스터에 대한 인덱스 집합과의 유사도를 측정하여 기능적인 유사도를 파악하게 된다.

$$SIM(CL_i, Query) = \frac{\sum_{k=1}^{n} (Term_{ik} \times Q-Term_{jk})}{\sum_{k=1}^{n} Term_{ik} + \sum_{k=1}^{n} Q-Term_{jk} - \sum_{k=1}^{n} (Term_{ik} \times Q-Term_{jk})}$$

CL_i: 라이브러리 도메인내의 i번째 클러스터

부품의 검색은 (그림 8)과 같이 질의에 대하여 클러스터를 기반으로한 선형검색으로 이루어지게 되며 선형검색의 결과를 바탕으로 분류단계에서 정의된 클러스터들간의 기능적 유사관계를 이용하여 가장 높은 등급을 받은 클러스터와 부품의 기능적 유사관계에 따른 계층도를 재사용자에게 제시한다.

```
get the query from the reuser
For Query Q
    make the index term
EndFor
compute the weight for each index
//Query = (Q-Term1 , .... , Q-Termn)
For each cluster CLi in Library Domain
compute similarity among query & CLi
find the components with the highest similarity to the Query
```

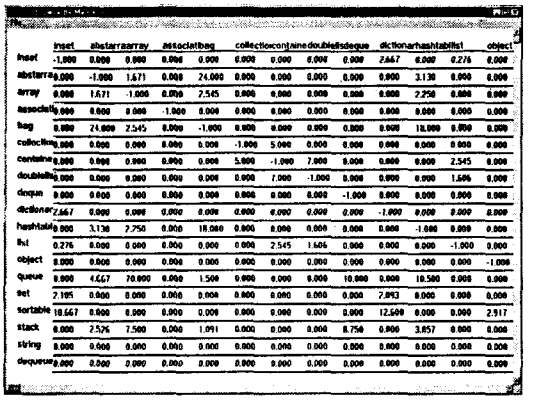
(그림 8) 클러스터 기반 선형검색 방법
(Fig. 8) Linear retrieval procedure based on cluster

재사용 라이브러리의 부품을 재사용하려는 자는 이러한 브라우즈 계층도를 이용하여 또다른 부품으로 접근을 할 수 있게 된다. 실제 구현된 검색시스템을 통하여 검색과정을 살펴보면 다음과 같다.

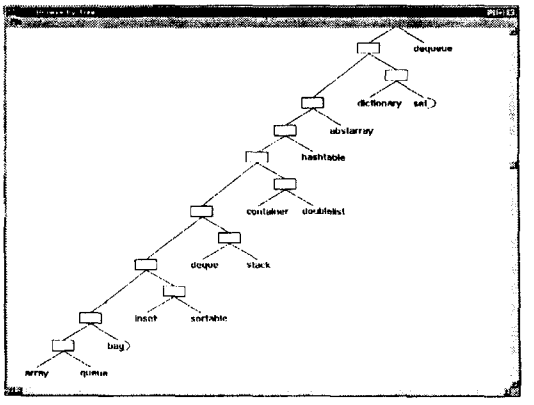
4.2 검색환경

본 논문의 검색환경은 재사용시스템 CARS 2.1로 통합되었고 검색시스템은 3가지 방법을 통해 사용자에게 부품간의 유사도 정보를 보여준다. 유사도란 사용자가 입력한 인덱스와 가중치를 의미하며, 프로화일을 이용하여 각 부품간의 비슷한 정도를 행렬과 트리형태로 보여준다. 3번째 항목은 사용자가 선택한 라이브러리, 즉 응용영역에서 사용자가 정의한 질의에 맞게 부품들간의 유사도를 보여준다. 이때 사용자의 질의는 정해진 구문을 갖는 질의어보다는 사용자의 사용편의성을 증진하기 위해서 이미 한 라이브러

리내의 정의되어진 인덱스와 가중치를 이용하여 질의를 작성한다. 즉 특별한 질의어에 대한 구문의 습득이 없어도 자신이 원하는 기능과 관련이 있는 인덱스를 선택하여 그의 중요도를 지정하면 되는 것이다. 행렬식 검색은 대상 라이브러리를 선택하면 (그림 9)와 같이 부품간의 유사도를 행렬로 보여준다.



(그림 9) 유사도 행렬
(Fig. 9) Similarity matrix



(그림 10) 클러스터 계층도
(Fig. 10) Cluster hierarchy

이 행렬은 자료구조 라이브러리의 각 부품들 간의 유사도 정보를 나타내고 있다. 행렬의 행과 열은 각각 부품을 나타내며 행렬 값은 각 행과 열에 해당하는 부품들 간의 유사도를 나타낸다. 위 그림에서는 array와 queue 부품의 유사도가 70.000으로 가장 크다. 따라서 자료구조 라이브러리에서는 array와 queue가 기능적으로 가장 유사한 부품임을 나타낸다.

트리식 검색은 한 라이브러리내의 부품들 간의 유사도를 트리형태로 보여주는 것으로 라이브러리를 선택하면 (그림 10)과 같이 부품들 간의 유사도를 트리형태로 보여주게 된다. 트리형태의 검색에서 단말노드는 부품과 클러스터 모두를 나타내고 널노드는 자식노드를 포함하는 클러스터의 개념으로 사용되고 있다. 깊이가 가장 깊은 자식노드인 array와 queue가 가장 유사도가 큰 부품임을 나타내고 이 두 부품으로 구성되는 클러스터와 가장 유사한 부품은 이 자식노드들의 부모노드의 형제노드인 bag이 되며 다시 이 세 부품으로 구성된 클러스터와 가장 유사한 부품은 이 노드의 부모노드의 형제노드가 된다. 다시 말해 형제노드사이가 가장 큰 관련성을 갖고 있고 다음으로는 부모노드의 형제노드순으로 그 유사도를 표현하고 있는 것이다.

사용자 질의에 의한 검색은 한 라이브러리내의 부품들 간의 유사도가 사용자가 정의한 질의를 기준으로 계산되어지는 방법으로 이러한 질의는 특정한 형식을 가지는 것이 아니라 사용자가 선택한 라이브러리내의 인덱스와 그 중요도, 즉 가중치를 입력하는 것이다. 여기서 입력하는 인덱스는 사용자가 원하는 부품과 관련 있는 인덱스를 입력한다. 그러나 사용자 질의에 의한 검색의 결과는 (그림 10)과 같은 트리형태로 나타나며 단지 사용자가 입력한 질의가 user라는 노드로 트리의 가장 아래쪽 자식노드에 나타난다.

4.3 시험 및 평가

본 논문에서는 검색성능에 관한 문제는 고려하지 않았다. 그보다는 탐색가능성과 같이 재사용자가 재사용 클래스 부품 라이브러리 내에 원하는 재사용 부품이 있다면 반드시 찾을 수 있고, 이를 3가지 방법을 통해 특히, 시각적인 검색을 통해 보다 더 용이하게 찾을 수 있다는 검색의 효과에 주안점을 두었다. 다만 제안한 클러스터링 방법과 알고리즘이 특정 라이브러리 영역에서 적당한 것인지, 또한 나타난 결과가 각각의 영역을 효과적으로 표현한 것인지에 대한 평가를 위해 서는 검색의 효율성과 검색 성능을 기준으로 평가해 본 후, 문서검색의 평가기준으로 사용되는 정밀도와 재현도를 본 연구의 검색결과를 평가하는

데 이용하도록 하겠다.

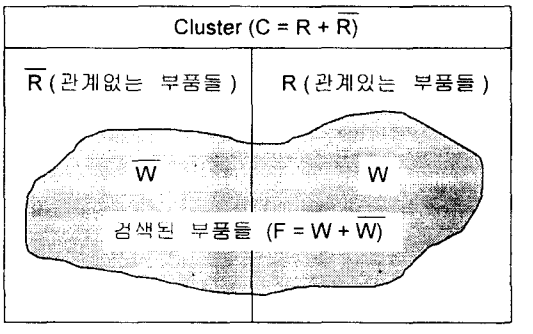
우선 검색의 효율성은 Blair[1]가 제시한 두 가지 관점, 즉 질의어 작성의 용이성과 제한된 단어로의 교체가 바람직한 가를 기준으로 평가해 본다. 이를 평가하기 위해서 검색과정을 다시 한번 살펴보면 다음과 같다. 질의는 재사용자가 필요한 기능을 의미하고 있는 자연어를 인덱스 형태로 기술하면 된다. 여러 인덱스중 중요한 의미를 갖고 있다고 생각한다면 높은 가중치를 주고 반면 상대적으로 중요도가 떨어지는 인덱스의 가중치는 낮게 설정한다. 예를 들어 재사용자가 "윈도우를 스크린 상에 그리고 그 윈도우의 배경색을 빨강 색으로 변경"하는 기능을 수행하는 부품을 라이브러리에서 찾고자 할 경우 window, screen, draw, background, color, red, change의 인덱스를 그 중요도에 따라 상대적인 가중치 값을 설정하여 검색시스템에 요청하면 검색시스템은 사용자 질의와 가장 유사한 부품을 여러 형태로 제시해준다. 이는 질의를 작성하는 재사용자의 경험과 지식에 따른 용어의 자유로운 선정이 가능한 자연어를 사용하게 한 것으로 특별한 질의어 습득의 노력이 필요 없다는 장점을 지닌다. Diaz[9]가 제안한 패싯 단위의 질의어 작성법과 비교하기 위하여 Diaz가 제안한 질의어 작성과정을 살펴보면 다음과 같다. Diaz 방식에 따르면 라이브러리가 기능, 응용영역, 매개물, 시스템 타입, 언어, 세팅의 6개 패싯단위로 구성된 함수들의 집합이므로 패싯에 해당하는 질의를 입력해야 한다. 또한 각 패싯에 대해서는 하나의 질의어만을 입력해야 한다. 예를 들어 "다중라인으로 backspace 변경"의 기능을 담당하는 부품을 찾으려면 (substitutue/backspace/file/text_formatter/program-development)의 질의를 작성해야 한다. 이 방법은 다양한 기능을 지니고 있는 객체지향 라이브러리의 부품을 검색하기 위해서는 여러 번의 질의를 작성한 후 부울린 연산을 수행해야 하는 단점을 지닌다. 두 번째로 고려할 점은 재사용자의 경험을 토대로 작성된 질의의 대체 필요성과 효율에 대한 평가이다. 앞의 질의를 구성하는 인덱스들은 미리 구축된 동의어 사전에 정의된 대표인덱스로 교체되면 window, screen, display, attribute, attribute, red, set으로 재구성된다. 질의의 재구성 단계는 더 광범위한 영역으로의 확장과 질의의 수정을 수행하는 것이다. 재구성을 통해 "draw"와 "change"가 "display"와 "set"으로 수정되었고 "background", "color"는 "attribute"의 확장된 용어로 대체되었다. 이는 라이브러리를 구성하는 부품에 대한 인덱스가 제한된 언어로 사용되어 있으므로 사용자의 지식을 라이브러리 시스템이 이해 가능한 형태로의 변형과정이다. 이런 재구성의 과정을 통해 수정/확장된 질의를 바탕으로 검색을 수정하므로써 보다 광범위한 질의처리가 가능해지는 장점을 지닌다. 그러나 단점으로는 본 논문에서는 Diaz 방법에서 채택한 질의의 축소/상세화 과정은 고려하지 않았으므로 보다 상세한 정보의 이용을 통해 얻을 수 있는 검색의 정확도 효과는 얻을 수 없다는 점이다.

또한 라이브러리의 검색결과는 정밀도(precision)와 재현도(recall)에 의해 평가될 수 있다. 정밀도는 관련된 부품의 검색 능력에 과한 평가기준이고, 재현도란 관련이 없는 부품의 배제 기능에 관한 평가기준이다[8]. 이를 수식으로 나타내면,

$$\text{정밀도} = \frac{\text{검색된 부품중 관련된 부품의 합}}{\text{검색된 부품의 합}}$$

$$\text{재현도} = \frac{\text{검색된 문서중 관련된 문서의 합}}{\text{전체문서중 관련된 문서의 합}}$$

이라 할 수 있다. 본 논문의 검색대상인 클러스터와 부품에 관점에서 이 기준을 설명한다면 다음 그림과 같다.



C는 하나의 클러스터를 구성하는 부품들의 집합이고, R과 $\overline{R}$는 C의 부분집합으로 R은 주어진 질의를 만족하는 부품들의 집합이고, $\overline{R}$는 질의를 만족하지 않는 부품이다. W는 R의 부분집합으로 검색된 부품

중 질의를 만족하는 부품이고, $\bar{W}$는 $\bar{R}$의 부분집합으로 검색된 부품중 질의를 만족하지 않는 부품을 의미한다.

그러므로 앞의 수식은 다음과 같이 수정된다.

$$재현도 = \frac{W}{F}, \quad 정밀도 = \frac{W}{R}$$

검색환경에서 사용자 질의를 display 2, polygon 3, screen 2, set 1, attribute 2, red 1, regular 1으로 주었을 때 가장 근접한 기능을 수행하는 부품으로 Reg_Polygon이 검색되어졌다. 또한 클러스터 기반 부품 검색을 통해 Rectangle, Square, Polygon, Figure, Circle, Close_Figure 등의 부품이 유사한 기능을 갖는 부품으로 검색되었다. 주어진 질의를 바탕으로 검색한 라이브러리는 Graphic/Window 라이브러리로 라이브러리를 구성하는 클래스 부품의 개수는 87개이고, 각 부품을 구성하는 실제 메소드를 포함하면 전체 600여개의 함수를 갖고 있다. 그러나 Graphic/Window 라이브러리에는 질의에 부합되는 기능을 수행하는 부품이 Rectangle, Square, Polygon, Triangle, Close_Figure, Circle, Ellipse 등이 존재했다. 이는 저장된 부품들 중 사각형과 유사한 그림을 그리거나 그려진 물체에 색을 지정하는 기능을 담당하는 부품이 7개 존재한다는 것을 의미한다. 즉, 검색의 결과로 얻어진 Rectangle, Square, Polygon, Figure, Circle, Close_Figure 중 Rectangle, Square, Polygon, Close_Figure의 4개만이 재사용자의 요구에 부응된 것이며, 실제 라이브러리 내에 저장된 Rectangle, Square, Polygon, Triangle, Close_Figure, Circle, Ellipse 중 4개만이 검색된 것이다. 이 경우 재현도와 정밀도 값은 다음과 같다.

$$재현도 = \frac{W}{F} = \frac{4}{6} = 0.66$$

$$정밀도 = \frac{W}{R} = \frac{4}{7} = 0.57$$

임의로 주어진 질의 50개에 대해 각각의 재현도와 정밀도를 평가한 결과는 다음과 같다.

질의의 성격	재 현 도	정 밀 도
· 일반적인 질의	0.75	0.62
· 재구성되지 않은 질의	0.42	0.39
· 애매한 질의	0.33	0.21
· 원하는 기능의 항에 대한 빈도를 높게 작성한 질의	0.58	0.85

동의어 사전의 유용도 확인을 위하여 동일한 질의 50개에 대하여 동의어사전에 의한 질의의 확장/수정의 단계를 제외한 검색을 수행해 보았다. 그 결과로 동일한 위의 질의에 대하여 얻어진 결과는 Reg_Polygon으로 동일하였으나 이와 유사한 기능을 수행하는 부품으로 선정된 부품들이 Figure, Segment, Point, Close_Figure, Line 등으로 실제 원하는 기능과 관련성이 떨어지는 부품이 검색되거나 실제 유사한 기능을 수행하는 부품이 검색되지 않는 결과를 얻었다. 즉 재현도와 정밀도가 떨어지는 결과를 얻게 되었다. 두 가지 실험을 통해 얻은 또 한가지의 경험은 애매한 질의, 예를 들어 display와 같이 구체적인 요구사항없이 광범위한 질의를 주었을 때, 찾아진 부품은 해당 라이브러리 중 가장 광범위한 부품인 Figure가 검색되었는데 실제 원하는 부품을 발견하기 위하여 여러 번의 브라우즈 계층구조의 재구성이 요구되었다. 그러나 질의에 자신이 원하는 기능중 가장 중요한 것에 대해 가중치를 높게 준 경우, 예를 들어 Polygon에 속하기는 하지만 일정한 Box 모양에 대한 여러 기능을 제공하는 부품을 검색하고자 한다면 Box란 항목의 가중치를 높게 주면 Polygon이나 Reg_Polygon보다는 Rectangle을 가장 유사한 부품으로 찾고 클러스터 기반의 검색결과에서는 Triangle이나 Circle 등은 빠지게 된다. 즉, 적절한 인덱스와 가중치가 정밀도의 향상에 영향을 미치게 된다.

5. 결 론

재사용 라이브러리가 효율적이기 위해서는 다양한 응용영역에 필요한 부품들이 효과적인 분류를 바탕으로 저장되고 재사용 부품들이 원하는 응용분야에서 적절하게 사용되어져야 한다.

본 논문에서는 재사용 단계의 접근성의 핵심을 이루는 분류스킴을 클러스터를 기반으로한 계층적인 구조로 정의하였고 검색시스템의 기능과 정확도를

결정하는 중요한 요소인 라이브러리 구조에 부품의 표현방법과 클러스터 정보가 추가된 클래스 부품들 간의 기능적인 관련성을 기술함으로써 이들 부품들을 관리하는 방법을 제안하였다. 결국 재사용 부품들은 객체지향 환경에서의 상속관계를 바탕으로한 지식기반 접근법과 부품들 간의 기능적인 관련성을 부품기술자로부터 얻어내는 정보검색방법을 모두 적용하여 재사용 라이브러리에 분류, 저장되어진다.

정보검색방법에 의한 재사용 부품들의 분류는 각 부품의 기술자로부터 그 부품의 기능을 나타낼 수 있는 의미 있는 단어 즉, 인덱스를 추출하고 이 인덱스가 각 부품에 나타난 빈도수를 구한 후 동의어 사전에 의해 관련된 인덱스들을 이들을 대표할 수 있는 대표 인덱스로 대체하게 된다. 각 부품들은 이들의 기능을 특성 짓는 인덱스와 그 빈도수로 표현되고 여기에 해당 인덱스가 라이브러리 영역내의 모든 부품들에 나타난 빈도수를 구하게 된다. 이러한 정보를 이용하여 얻어진 가중치를 바탕으로 한 인덱스 벡터 모델로 각 부품들을 다시 한번 표현하게 된다. 이렇게 가중치를 가진 인덱스 벡터모델을 바탕으로 라이브러리 영역내의 부품들 간의 유사도를 구하여 이를 매트릭스 형태로 표현하고 이 정보를 HACM에 적용하여 각 부품을 클러스터 계층으로 표현하게 된다. 본 논문에서는 이러한 클러스터 계층이 부품 분류의 결과가 되었고 이를 기반으로 사용자의 질의와 가장 부합되는 부품을 찾게되며 사용자에게는 CARS 2.1의 서브시스템으로 검색환경을 제공하였다. 기능적으로 관련된 다른 부품들은 이미 구해진 클러스터 계층에 나타난 정보를 바탕으로 사용자에게 정렬하여 보여주고 사용자가 다른 부품을 선정시 선정된 부품과 또 다른 부품들간의 기능적인 관련성은 다시 계산할 필요없이 분류 단계에서 얻어진 클러스터 계층도를 사용자에게 직접 제시함으로써 사용자가 원하는 부품을 시각적인 검색을 통하여 접근할 수 있도록 하였다.

본 논문에서는 재사용 라이브러리의 구축과정중 일부를 자동화하였고 패싯 분류방법의 장점인 확장성과 계층적인 분류방법의 장점인 관련성 모두를 고려한 라이브러리를 HACM을 기반으로 하여 구축하였다. 연구방향은 원하는 재사용부품이 라이브러리 내에 존재한다면 반드시 찾을 수 있어야한다는 탐색 가능성에 초점을 맞추었다.

향후 연구과제로는 라이브러리의 확장성에 대한 검증이 필요하고 동의어 사전의 자동화와 검색의 정확도를 증진시킬 필요가 있다. 또한 재사용부품의 기술서로부터 추출된 인덱스들간의 의미적 유사성을 좀 더 정확하게 추출하기 위하여, 단어들간의 개념적 거리(conceptual distance)를 측정할 수 있는 방법을 도입할 필요가 있다.

참 고 문 헌

[1] Blair, D.C. and Maron, M.E., "An evoluation of retrieval effectiveness for a full-text document-retrieval system.", Communication of the ACM, pp. 289-299, 28(3):March, 1985.

[2] B.A. Burton, R.W. Aragon, S.A. Bailey, K.D. Koehler and L.A. Mayes, "The reusable software library", IEEE software, pp. 25-33, July 1987.

[3] W.B. Frakes and B.A. Nejmeh, "An information system for software reuse", Proceedings of the Tenth Minnowbrook Workshop on Software Reuse, 1987.

[4] A. Griffiths, L.A. Robinson and P. Willett, "Hierarchic agglomerative clustering methods for automatic document classification", The Journal of Documentation, Vol. 40, No. 3, Sep 1984.

[5] B. Meyer, "Lesson from the design of the Eiffel libraries", Communications of the ACM/September, Vol. 33, No. 9, pp. 90-103, 1990.

[6] Hafedh Mili, Fatma Mili, and Ali Mili "Reusing software:issues and research directions", IEEE transactions on software engineering, Vol. 21, No. 6, June 1995.

[7] J.-M. Morel & J. Faget, "The REBOOT environment", 2nd International Workshop on Software Reusability:Advances in Software Reuse, pp. 80-88, 1993.

[8] A. S. Pollitt, "Information storage retrieval system", Ellis Horwood, 1989.

[9] R. Prieto-Diaz and P. Freeman, "Classifying software for reusability", IEEE software Vol. 4, No

pp. 6-16, Jan 1987.

[10] R. Prieto-Diaz and G.A. Janes "Breathing new life" GTE Journal of Science and Technology, pp. 23-31, Spring, 1987.

[11] E. Rasmussen, "Clustering algorithms", Information retrieval:data structure and algorithms, W.B. Frakes & R. Baeza-Yates, Prentice Hall, pp. 419-442, 1992.

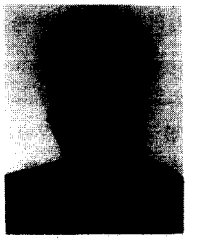
[12] Voorhees, E.M. "The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval." Ph.D. thesis, Cornell University, 1986.

[13] Willett, "Recent trends in hierarchic document clustering:a critical review.", Information Processing & Management, 24(5), pp. 577-97, 1988.

[14] 이 경환, "소프트웨어 재이용을 위한 연구", 과기처 특정과제 연구보고서, 1991.

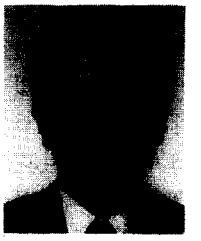
배 제 민

1991년 중앙대학교 전자계산학과 (이학사)
1993년 중앙대학교 전자계산학과 (이학석사)
1994년~현재 중앙대학교 컴퓨터공학과 박사과정
관심분야:소프트웨어공학, 객체지향 방법론, 소프트웨어 재사용



김 상 근

1987년 중앙대학교 전자계산학과 (이학사)
1989년 중앙대학교 전자계산학과 (이학석사)
1996년 중앙대학교 컴퓨터공학과 (공학박사)
1996년~현재 성결대학교 전산통계학과 전임강사
관심분야:소프트웨어공학, 객체지향 방법론, 소프트웨어 재사용



이 경 환

1980년 중앙대학교 대학원 응용수학 전공(이학박사)
1982년~1983년 미국 Auburn대학 객원교수
1986년 서독 Bonn대학 객원교수
1971년~현재 중앙대학교 컴퓨터공학과 교수
관심분야:소프트웨어 공학, 객체지향 모델링, 소프트웨어 재사용.