소프트웨어 재사용을 지원하는 확장된 패싯 분류 방식과 혼합형 검색 모델

강 문 설† 김 병 기††

요 약

본 논문에서는 소프트웨어 부품을 분류하여 라이브러리에 저장하고, 사용자의 요구에 따라 효율적으로 검색할 수 있도록 지원하는 확장된 패싯 분류 방식과 혼합형 검색 모델을 제안하고, 프로토타입 시스템을 설계하여 구현하였다. 분류 방식의 설계를 위하여 부품들의 기본적인 클래스를 분석하여 필요한 항목을 식별한 다음, 항목들의 특성을 분석하고 패싯을 결정하여 부품 식별자를 구성한다. 그리고 부품의 기본적인 특성을 기준으로 응용 영역별로 클러스터링시켜 라이브러리에 저장하고, 부품의 특성을 표현하기 위하여 패싯과 항목들에 가중치를 할당하였다. 부품의 검색을 위하여, 질의에 의한 검색 모델 및 유사한 부품들을 쉽게 검색할 수 있도록 가중치와 유사도를 이용하였다. 제안한 분류 방식과 검색 모델은 분류 과정이 간단하고, 유사한 부품을 쉽게 식별할 수 있었으며, 또한 질의 작성이 간단해지고, 출력될 부품들의 크기와 순서의 조절이 가능하여 검색 효율이 개선되었다.

An Extended Faceted Classification Scheme and Hybrid Retrieval Model to Support Software Reuse

Moon Seol KANG† and Byung Ki KIM††

ABSTRACT

In this paper, we design and implement the prototype system, and propose the Extended Faceted Classification Scheme and the Hybrid Retrieval Method that support classifying the software components, storing in library, and efficient retrieval according to user's request. In order to designs the classification scheme, we identify several necessary items by analyzing basic classes of software components that are to be classified. Then, we classify the items by their characteristics, decide the facets, and compose the component descriptors. According to their basic characteristics, we store software components in the library by clustering their application domains and are assign weights to the facets and its items to describe the component characteristics. In order to retrieve the software components, we use the retrieval-by-query model, and the weights and similarity for easy retrieval of simialr software components. As the result of applying proposed classification scheme and retrieval model, we can easily identify similar components and the process of classification become simple. Also, the construction of queries becomes simple, the control of the size and order of the components to be retrieved is possible, and the retrieval effectiveness is improved.

1. 서 론

소프트웨어의 생산성 향상과 품질 개선을 위한 방안으로 소프트웨어 재사용에 관한 많은 연구가 진행되고 있다. 소프트웨어 시스템을 개발하는 과정에서 기존의 소프트웨어 부품을 효율적으로 재사용하기 위해서는 재사용 가능한 소프트웨어 부품을 식별하고, 분류하여 저장한 후 사용자의 요구 사항과 일치하는 부품의 검색 방법, 그리고 검색한 부

† 정 회 원 : 전남대학교 전산학과 강사
†† 종신회원 : 전남대학교 전산학과 교수
논문접수 : 1994년 2월 25일 , 심사완료 : 1994년 4월 6일

품을 사용자가 쉽게 수정하여 합성할 수 있도록 부품을 이해할 수 있는 부품의 효율적인 표현 방법이 제공되어야 한다[8, 14].

부품을 재사용하는 과정에서 직면하게 되는 문제들은 재사용 가능한 부품의 분류, 검색, 그리고 표현을 위한 도구와 방법론 등 재사용을 지원하는 기술이 부족하다는 것이다. 특히 기존의 재사용 가능한 부품을 재사용하는 과정에서 많은 문제에 직면하게 되는 데, 이 문제점들을 다음과 같이 요약할 수 있다. 첫째, 재사용 가능한 부품을 식별하고, 분류하여 체계적으로 기술할 수 있는 적합한 부품의 분류 방식이 부족하다. 둘째, 사용자의 요구 사항을 질의로 구성하는 방법 및 사용자와 시스템간의 상호작용을 지원할 수 있는 적합한 부품의 검색 모델이 부족하다. 셋째, 부품의 수정과 합성을 위한 부품의 이해 방법이 대부분 제공되지 않는다. 넷째, 재사용 가능한 부품의 분류 방식과 검색 모델을 위한 기술의 개발이 부족하다[7, 15].

본 논문에서는 재사용 가능한 부품을 분류하여 라이브러리에 저장하고, 사용자의 요구에 따라 효율적으로 검색할 수 있도록 지원하는 확장된 패싯(facets) 분류 방식과 혼합형 검색 모델을 제안하여, 프로토타입 시스템을 설계하고 구현하였다. 첫째, 부품들을 효율적으로 분류하기 위한 방법으로 부품들의 특성을 분석하여 5개의 패싯을 결정하고, 각 부품들의 특성을 표현하기 위하여 패싯과 항목들에 가중치를 부여하는 확장된 패싯 분류 방식을 제안하여 부품의 분류를 간단하게 하고, 검색 효율을 개선하였다. 둘째, 부품의 효율적인 검색을 위하여 질의에 의한 방법 및 유사한 부품을 쉽게 검색할 수 있도록 가중치를 할당하고 유사도를 계산하여 검색하는 혼합형 검색 모델을 제안하여 질의 구성을 간단하게 하고, 검색 효율을 개선하였다. 셋째, 부품을 재사용하기 위하여 제안한 분류 방식과 검색 모델을 바탕으로 프로토타입 시스템을 IBM-PC/486에서 C 언어로 구현하여 실험한 결과를 이론적인 성능 평가 모델을 이용하여 평가하였다.

본 논문은 2장에서 부품의 분류 방식 및 검색 모델의 개발에 관한 연구 동향을 조사 분석하여 문제점을 기술하고, 3장에서는 확장된 패싯 분류 방식에 대하여, 그리고 4장에서는 사용자의 요구 사항에 일치하는 부품을 검색할 수 있도록 제안한 혼합형 검색 모델에 대하여 기술하였다. 5장에서는 제안한 분류 방식과 검색 모델을 바탕으로 프로토타입 시스템을 구현하여, 실험을 통한 평가 결과를 기술하였며, 6장에서는 결론 및 향후 연구 방향을 약술하였다.

2. 관련 연구

소프트웨어의 생산성 향상과 품질 개선은 소프트웨어 공학의 가장 중요한 연구 목표이다. 수십년 동안 이러한 목표를 달성하기 위하여 많은 연구가 진행되어 왔으나 획기적인 방법은 개발되지 않고 있다. 많은 연구 결과에 의하면 재사용 기술이 소프트웨어 생산성의 향상과 품질을 개선하기 위해서 가장 적절한 방법중의 하나로 밝혀졌다. 재사용은 이미 개발된 소프트웨어에서 공통적으로 이용된 부분들을 표준화하고, 이들을 새로운 소프트웨어 개발 과정에서 재사용함으로써 소프트웨어 생산성의 향상 및 품질의 개선 뿐만 아니라 유지 보수 비용과 테스트 비용을 줄일 수 있다[8, 14].

재사용에 관한 최근의 연구 동향은 부품을 재사용하기 위한 필요한 활동이나 절차의 형식화 및 표준화를 추구하고 있는데, 현재까지의 재사용에 관한 연구 결과를 다음의 세 가지 범주로 분류하여 요약할 수 있다. 첫째, 소프트웨어 개발 과정에서 생산되는 부품들을 다음 개발 과정에서 재사용할 수 있도록 지원하기 위한 부품의 구축에 필요한 이론 및 방법론에 관한 연구가 진행되고 있다[2, 4, 5]. 둘째, 부품을 효율적으로 재사용할 수 있도록 지원하는 재사용 라이브러리 시스템을 위한 프레임워크(framework)에 관한 연구가 진행되고 있다[3, 10, 16]. 셋째, 효율적인 재사용을 성취하기 위하여 부품의 라이브러리를 재구성하기 위한 분류 방식과 검색 모델에 관한 연구이다[9, 11, 12, 15, 17].

재사용 가능한 부품과 라이브러리를 분류하여 재

구성하고 효율적으로 검색하기 위한 부품의 분류 방식과 검색 모델에 관련된 연구에서 나타난 문제점을 다음과 같이 요약할 수 있다. 첫째, 부품의 분류가 패싯 분류 방식을 사용하여 수동적으로 수행되고 있다. 따라서 분류된 부품의 분류 방식에 대한 많은 정보를 사용자가 알고 있어야 하며, 라이브러리의 크기가 커지면 검색 시간이 길어지고 라이브러리의 관리가 어려워진다. 둘째, 부품을 검색하기 위해서 전통적인 텍스트 검색 방법을 채택하고 있기 때문에 부품을 표현하는 데 필요한 정보를 제공하기 어렵고, 검색 시간이 길어지며, 전체적인 검색 효율이 떨어지게 된다. 또한 질의를 작성할 때 어휘 통제 기능이 부족하기 때문에 항목 동의어 사전나 항목들 간의 개념적인 관계를 반드시 이용해야 한다. 셋째, 사용자의 요구 사항과 일치하는 부품의 검색에 실패했을 경우에 다수의 유사한 부품들을 검색하여 사용자가 선택할 수 있어야 하는데, 유사한 부품의 검색이 어렵게 구성되었다. 넷째, 검색한 부품은 적절한 수정을 거쳐서 개발 중인 시스템에 합성된다. 따라서 검색한 부품을 수정하기 위해서는 우선 부품을 이해할 수 있어야 한다. 그러나 관련 연구에서는 검색한 부품의 이해를 위한 필요한 정보의 표현 방법이 대부분 제공되지 않는다.

3. 확장된 패싯 분류 방식

3.1 소프트웨어 부품을 위한 분류 방식의 설계 목적

부품을 위한 분류 방식의 목적은 재사용의 효율을 증진시키기 위하여 부품의 라이브러리를 효율적으로 구성하는 것이다. 재사용 환경에서 부품을 효율적으로 분류할 수 있도록 지원하는 분류 방식의 설계 목적을 다음의 가정을 기초로 정의한다.

가정 1. 대규모의 부품을 포함하는 라이브러리가 있다.

가정 2. 실제적으로 재사용이 이루어지고 있다.

가정 3. 부품의 재사용 대상이 코드 수준에서 다른 부품으로 확장되고 있다.

위의 가정을 전제로 자료 분류의 목적[18]을 수정하여 부품을 분류하기 위한 분류 방식의 설계 목적과 고려 사항을 다음과 같이 정의하였다.

1. 부품을 위한 분류 방식의 주요 목적은 재사용 가능한 부품의 특정 집합을 식별하고 찾아냄으로써 사용자를 지원하는 것이다.
2. 분류 방식에 사용된 구조와 개념이 특정 응용 영역에서 작업하는 소프트웨어 개발자들에 의해서 사용되는 구조 및 개념과 비슷하게 해야 한다.
3. 분류 방식과 기술된 속성들은 유사한 부품을 식별하기 위해 정확성을 제공해야 한다.
4. 분류 방식은 검색된 부품의 상관 순위를 제공하는 자동화된 검색 메카니즘을 지원해야 한다.
5. 부품의 나열 순서는 다른 사용자의 요구에 적합하도록 조정할 수 있어야 한다. 다시 말하면, 중요도의 변화에 따라 클래스들을 재배열할 수 있어야 한다.
6. 전산학은 변화 속도가 빠른 분야이므로 유연성과 확장성은 중요한 고려 사항이 된다. 분류방식은 재분류 문제가 발생하지 않도록 수정 및 확장이 용이해야 한다.

위에서 정의한 분류 방식의 설계 목적 및 고려 사항은 기존의 분류 방식을 비교하고 평가하기 위하여 사용된다. 그리고 선택된 분류 방식의 구조와 특징은 부품의 분류를 위하여 새롭게 제안하는 분류 방식의 기초를 제공한다.

3.2 확장된 패싯 분류 방식의 설계

분류 방식의 설계 목적과 고려 사항을 기준으로 열거형 분류 방식과 패싯 분류 방식을 비교 분석한 결과, 패싯 분류 방식이 기호법을 제외한 다른 모든 측면에서 좋게 나타났다[11]. 따라서 대부분의 부품을 위한 분류 방식이 패싯 분류 방식을 채택하고 있다. 그러나 부품의 분류와 검색은 매우 밀접한 관계를 가지고 있기 때문에 다음과 같은 문제점이 검색 과정에서 야기된다.

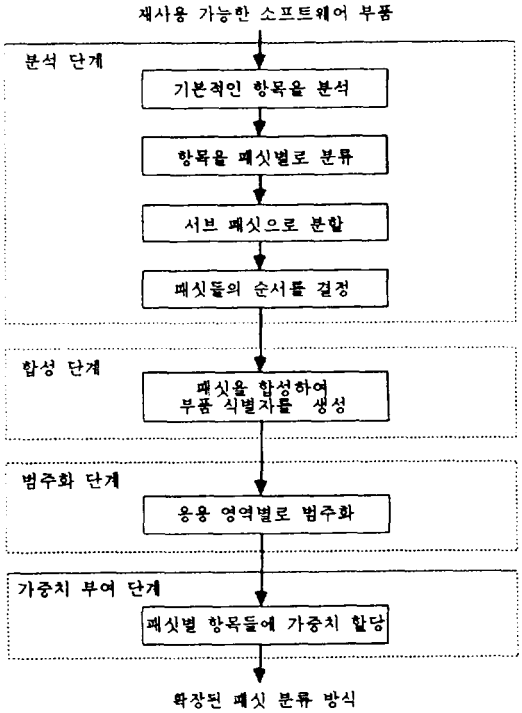
첫째, 부품을 표현하기 위한 적절한 패싯의 수를 결정하는 것이 어렵다. 일반적으로 패싯의 수가 많

으면 부품을 정확하게 표현할 수 있으나 질의 구성과 처리가 복잡해진다. 한편 패싯의 수가 적으면 질의 구성과 처리는 간단하지만 부품을 정확하게 표현할 수 없다.

둘째, 일반적으로 라이브러리의 크기는 계속해서 증가되기 때문에 라이브러리의 관리가 어렵고 검색 효율이 떨어진다. 특히 재사용 가능한 부품의 규모가 계속해서 증가되고 있기 때문에 이 문제는 더욱 심각하게 나타난다.

셋째, 패싯 분류 방식을 이용하여 구축한 라이브러리에서는 유사한 부품의 검색이 어렵다. 사용자가 요구하는 부품의 검색에 실패했을 경우 다수의 유사한 부품을 검색할 수 있도록 지원되어야 한다.

제안한 확장된 패싯 분류 방식은 부품의 기본적인 클래스를 분석하여 패싯과 패싯의 나열 순서를 결정하는 분석 단계, 분석 단계에서 결정된 패싯과 패싯의 나열 순서에 따라 패싯들을 합성하여 부품 식별자를 생성하는 합성 단계, 부품의 기본적인 특



(그림 1) 확장된 패싯 분류 방식의 설계 과정
(Fig. 1) Design process of extended faceted classification scheme

성에 따라 응용 영역으로 범주화시키는 범주화 단계 및 각 패싯과 패싯별 항목에 가중치를 할당하는 가중치 부여 단계로 구성하였다. 제안한 확장된 패싯 분류 방식의 설계 과정을 (그림 1)에 나타내었다.

3.2.1 패싯 분석 단계

패싯 분석 단계에서는 부품의 기본적인 클래스들을 분석하여 패싯을 결정하고, 결정된 패싯의 순서를 결정하는 단계이다. 패싯 분류 방식을 구축하기 위한 일반적인 지침은 도서관학에서 사용하는 분류 지침에 기초를 두고 있다[11]. 그러나 이 지침은 문헌 및 도서 분류 시스템을 위해 설계되었기 때문에 부품의 분류 방식에 적용하기 위하여 다음과 같이 패싯을 분석하는 알고리즘을 제안하였다.

〈패싯 분석 알고리즘〉

단계 1. 기본적인 클래스들을 분류하기 위하여 부품의 대표적인 예제들을 분석한다.

단계 2. 기본적인 클래스들을 논리적인 분할의 단일 특성 원칙을 사용하여 패싯으로 통합한다.

단계 3. 하나의 패싯을 분할의 특성을 적용하여 서브패싯을 생성한다. 불필요한 경우에는 생략한다.

단계 4. 사용자들의 필요성에 따라 패싯과 서브패싯 내의 항목들을 위한 적합한 순서를 선택한다.

단계 5. 패싯 내의 순서에 따라 서브패싯을 배치한다.

단계 6. 패싯들을 합성하기 위해 사용할 수 있도록 패싯들을 위한 조합 순서를 결정한다.

(그림 2) 패싯을 분석하는 알고리즘
(Fig. 2) Facets analysis algorithms

패싯 분석 알고리즘에 따라 제안한 분류 방식은 부품을 효율적으로 분류하고, 사용자가 요구하는 부품의 표현을 쉽게하기 위하여 부품이 수행하는

기본적인 기능과 부품의 구현 및 실행 환경에 관한 정보를 패싯으로 선택하였다. 또한 사용자의 요구 사항에 일치하는 부품이 없을 때, 다수의 유사한 부품을 검색할 수 있도록 5개의 패싯 즉, 기능, 객체, 매체, 구현 언어 및 운영체제를 채택하였다.

(1) 부품의 기본적인 수행 기능

사용자의 요구 사항을 어느 정도 정확하게 표현할 수 있는가에 따라 부품의 검색 효율이 결정된다. 사용자가 요구하는 부품을 신속하고 정확하게 검색하기 위해서는 재사용 가능한 부품이 수행하는 기능과 기능이 수행되는 대상들이 커다란 영향을 미친다. 이들 정보를 다음과 같은 패싯으로 정의한다.

〈정의 3.1〉 기능 패싯(functions facet)

부품이 수행하는 기본적인 기능을 기능 패싯이라 한다.

〈정의 3.2〉 객체 패싯(objects facet)

기능을 수행하기 위해 적용되는 대상을 객체 패싯이라 한다.

〈정의 3.3〉 매체 패싯(medium facet)

객체가 저장된 장소나 실제 기능이 수행되는 장소에 관한 정보를 표현하는 패싯을 매체패싯이라 한다.

(2) 부품의 구현 및 실행 환경

부품은 같은 방법으로 동일한 기능을 수행하더라도 구현 정도, 표현 방법, 개발된 환경 및 실행 환경 등에 따라 재사용에 영향을 줄 수 있다. 이들 정보를 다음과 같은 패싯으로 정의한다.

〈정의 3.4〉 언어 패싯(language facet)

부품을 기술하는 프로그래밍 언어를 언어 패싯이라 한다. 부품의 원시 코드를 작성하는데 사용된 프로그래밍 언어를 기술한다.

〈정의 3.5〉 운영체제 패싯(operating system facet)

부품을 개발한 시스템이나 부품이 수행될 수 있는 시스템의 운영체제를 운영체제 패싯이라 한다.

3.2.2 합성 단계

합성 단계에서는 분석 단계에서 결정된 패싯들과 그들의 순서에 따라 패싯들을 합성하여 하나의 부품 식별자를 결정한다.

즉 부품 식별자는 분류하고자 하는 부품을 표현하기 위해서 패싯 단위로 분류된 기본적인 클래스들을 모아서 구성한다. 따라서 패싯들을 합성하는 과정은 일관성 있는 부품의 분류를 위하여 패싯들의 합성 순서를 반드시 고려해야 한다.

〈정의 3.6〉 부품 식별자(CI : component identifier)

분석 단계에서 결정된 각 패싯들로부터 선택한 항목들의 집합을 이용하여 부품 식별자를 합성하고, 다음과 같이 구성한다.

$$CI = \langle T_{i1}, T_{i2}, T_{i3}, \cdots, T_{ij}, \cdots, T_{in} \rangle$$

여기서, T_{ij} : i번째 부품의 j번째 패싯의 항목, n : 패싯의 수

3.2.3 범주화 단계

부품을 5개의 패싯에 따라 분류하고, 항목들을 합성하여 부품 식별자를 구성한 다음 한 개의 데이타 베이스에 저장한다. 그러나 부품의 라이브러리는 상대적으로 그 규모가 크고, 급속히 확장되어 라이브러리의 크기가 커지면 관리하기 어렵고 검색 효율이 떨어진다.

따라서 범주화 단계에서는 열거형 분류 방식의 개념을 적용하여 분류 대상의 부품을 N개의 영역으로 분류한다. 이렇게 부품을 응용 영역별로 분류하는 이유는 검색 과정에서 검색 시간을 줄이고, 부품의 특성을 쉽게 표현하여 검색 효율을 개선시킬 수 있도록 사용자에게 편의성을 지원하는 것이

다. 대부분의 사용자들은 부품을 재사용하기 위해서 해결하고자 하는 문제 영역에 대한 부품에만 관심을 가지게 된다. 따라서 본 논문에서는 부품의 라이브러리가 계속해서 확장된다고 가정하고, 사용자가 관심을 갖고 있는 영역만 탐색하여 요구하는 부품을 신속히 검색할 수 있도록〈표 1〉과 같이 실험적으로 5개의 응용 영역으로 분류하였다.

〈표 1〉 소프트웨어 부품의 응용 영역별 범주
〈Table 1〉 Domain categories of software components

응 용 영 역	분류 대상의 소프트웨어 부품
수 학 및 통 계	수학과 통계 분야에 관한 이론 및 응용 분야에 관련된 소프트웨어 부품
자 료 구 조	스택, 큐, 리스트, 정렬 및 탐색 등의 자료구조 관리에 관련된 소프트웨어 부품
윈 도 우 관 리	윈도우 및 텍스트 화면의 관리에 관련된 소프트웨어 부품
그 래 픽 처 리	그래픽 처리에 필요한 소프트웨어 부품
입출력 및 기타	입출력, 마우스 조작 및 기타 시스템의 관리에 관련된 소프트웨어 부품

3.2.4 가중치 할당 단계

가중치 할당 단계에서는 기본 패싯에 대하여 가중치를 할당하고, 응용 영역별로 항목들에 가중치를 할당하는 단계이다. 패싯 가중치는 부품의 특성을 설명하는 정도에 따라 각 패싯에 가중치를 할당하고, 항목 가중치는 각 패싯별로 분류되는 항목들의 중요도에 따라 가중치를 할당한다. 본 논문에서는 패싯 가중치와 항목 가중치를 다음과 같이 정의하였다.

〈정의 3.7〉 패싯 가중치(f_j : facet weight)

패싯들이 부품의 특성을 어느 정도 설명하고, 표현할 수 있는 가에 따라 각각의 패싯에 가중치를 할당하여 패싯 가중치라고 정의하고, f_j로 표현한다.

패싯 가중치는 검색 과정에서 사용자가 요구하는 부품의 검색에 실패했을 때, 다수의 비슷한 부품을 검색할 수 있도록 하기 위해서 사용된다. 따라서 이 패싯 가중치는 사용자들의 요구 사항에 따라 결정되는데, 사용자가 요구하는 부품의 특성을 표현할 수 있는 정도, 즉 어떤 패싯이 부품의 특성을 얼마나 설명하고 있는가를 가중치로 표현하기 위해〈표 2〉와 같이 기준치(threshold weight, $0 < \lambda_i \leq 1.0$)를 할당할 수 있도록 구성하였다.

〈표 2〉 패싯 가중치 값
〈Table 2〉 Facets weights values

패싯명	기 능	객 체	매 체	언 어	운영체제
가중치	λ_1	λ_2	λ_3	λ_4	λ_5

일반적으로 사용자가 부품을 재사용하기 위해서 그 부품이 어떤 기능을 수행하고 있는 가를 가장 먼저 고려한다. 본 논문에서는 가중치 선택 기준을 참조하여[13], 사용자가 요구하는 부품을 검색할 경우에 가장 먼저 그 부품의 기능을 고려한다는 가정에서 실험적으로 패싯 가중치 λ_1를 결정하였다

〈정의 3.8〉 항목 가중치(w_{ij} : term weight)

패싯별로 포함된 각 항목들에 가중치를 할당하여 항목 가중치라 정의하고, w_{ij}로 표현한다.

일반적으로 문헌 검색에서 항목의 빈도와 항목의 중요성의 관계가 검색에 미치는 영향을 빈도가 많은 항목은 유용하지 않고, 빈도가 중간 정도의 항목이 가장 유용하며, 그리고 빈도가 매우 적은 항목은 별로 유용하지 않다고 정의하였다[13]. 그러나 본 논문에서는 분류 대상이 부품이고, 사용되는 항목들은 부품의 특성을 나타내고 있기 때문에 빈도가 많은 항목은 다수의 부품에서 매우 유용하게 사용된다고 할 수 있다. 따라서 항목 가중치 w_{ij}를 다음의 수식으로 정의한다.

$$w_{ij} = \frac{Freq_{ij}}{MaxFreq_j}$$

여기서, $Freq_{ij}$: j번째 패싯에서 항목 i의 빈도수, $MaxFreq_j$: 각 패싯에서 어떤 항목의 최대 빈도수, i : 각 패싯에서 i번째 항목, j : 해당 패싯

3.3.6 항목 동의어 사전

일반적으로 동의어 사전는 주어진 영역에 관련된 지식을 나타내는 항목들과 그 항목들 간의 통제어, 동의어 및 개념적 관계를 기술하는 정보 구조로써 동의어 사전에 나타난 항목들은 부품을 기술하고 사용자 질의에 사용될 항목들에 대한 표준 어휘 역할을 한다.

〈정의 3.9〉 **항목 동의어 사전(term synonym dictionary)**

패싯에 속한 각 항목들에 대한 동의어 사전을 항목 동의어 사전이라 정의하고, 다음과 같은 형식으로 구성한다.

$[\langle s_0 \rangle \mid \langle s_1, s_2, s_3, \cdots, s_n \rangle]$

여기서, s_0 : 항목, s_i : 동의어($i = 1, 2, \cdots, 10$)

본 논문에서는 항목 동의어 사전을 〈정의 3.9〉와 같이 각 패싯에 속한 항목들에 대한 동의어 사전을 만들어 다양한 사용자 구성원들 사이의 항목에 대한 개념의 차이를 해결하고자 할 때나 저장된 정보와 사용자 질의 사이의 모호성 또는 항목에 대한 개념의 차이를 제거하기 위하여 질의 작성 과정에서 사용된다. 본 논문에서 사용한 항목 동의어 사전의 일부를 (그림 3)에 나타내었다.

항 목(terms)	동의어(synonyms)
add	increment/total/sum/plus
append	attach/concatenate/join
compare	test/relate/match/check/verify
copy	duplicate/facsimile/image
delete	remove/erase
measure	estimate/valuate/enumerative/size
output	print/echo/write/list
search	find/locate/retrieval
sort	order/rank/arrange/discriminate/category

(그림 3) 항목 동의어 사전의 예
(Fig. 3) Examples of term synonym dictionary

3.3.6 제안한 분류 방식에 따른 분류의 예

제안한 분류 방식에 따라 부품을 분류하면 응용 영역별로 각 부품의 특성에 대한 패싯의 정보가 〈항목(T_{ij}) : 가중치(w_{ij})〉의 쌍으로 부품 식별자에 저장된다. 〈표 3〉은 제안한 분류 방식에 따라 분류한 결과의 일부를 나타내었다.

〈표 3〉 제안한 분류 방식에 의한 부품의 분류 예
〈Table 3〉 Examples of component classification

부품 이름	기능 패싯		재질 패싯		매체 패싯		언어 패싯		운영체제 패싯	
atoi	convert	1.00	string	1.00	buffer	1.00	C++	1.00	MS-DOS	1.00
strcpy	copy	0.99	string	1.00	buffer	1.00	C++	1.00	MS-DOS	1.00
strcmp	comapre	0.98	string	1.00	buffer	1.00	C++	1.00	MS-DOS	1.00
strcat	append	0.92	string	1.00	buffer	1.00	C++	1.00	MS-DOS	1.00
fseek	change	0.95	pointer	0.78	file	0.77	C++	1.00	MS-DOS	1.00
lseek	move	0.91	pointer	0.78	file	0.77	C++	1.00	MS-DOS	1.00
strchr	find	0.91	character	0.76	string	0.64	C++	1.00	MS-DOS	1.00
lsearch1	search	0.96	element	0.71	array	0.51	C++	1.00	MS-DOS	1.00
bsearch1	search	0.98	element	0.71	array	0.51	C++	1.00	MS-DOS	1.00
qsort1	sort	0.91	element	0.71	array	0.51	C++	1.00	MS-DOS	1.00
lsearch2	search	0.98	element	0.71	array	0.51	C++	1.00	UNIX	0.44
bsearch2	search	0.98	element	0.71	array	0.51	C	0.37	UNIX	0.44
qsort2	sort	0.91	element	0.71	array	0.51	C	0.37	UNIX	0.44

4. 소프트웨어 부품의 검색을 위한 혼합형 검색 모델

4.1 정보 검색 모델의 요구 사항

부품을 재사용하기 위한 검색 시스템의 주요 목적은 사용자가 요구하는 부품의 특성에 대한 주어진 표현을 만족하는 일련의 재사용 가능한 부품을 검색하는 것이다. 이러한 목적을 만족하기 위해서 요구되는 검색 시스템의 필수적인 요구 사항은 질의 구성, 검색된 부품의 상관 순위, 어휘 통제를 위한 방법 및 사용자가 소프트웨어 라이브러리를 재배열하기 위한 방법 등이 포함된다[1, 13]. 부품의 재사용을 위한 검색 모델은 질의 구성이 간결하고 쉬워야 하며, 사용자가 요구하는 부품을 신속하고 정확하게 검색할 수 있어야 한다. 또한 사용자가 요구하는 부품의 검색에 실패할 경우에 다수의 유사한 부품을 검색할 수 있어야 하며, 라이브러리 내의 부품들의 배열 순서를 조정할 수 있어야 한다.

4.2 검색 모델의 설계 배경

정보 검색 시스템의 중요한 특성은 사용자의 요구 사항을 만족하는 부품을 라이브러리에서 짧은 시간에 효율적으로 찾아낼 수 있도록 지원하는 검색 모델을 선택하는 것이다. 정보 검색 시스템에서 사용자의 요구 사항을 질의로 표현하는 방법에 따라 검색 모델을 세 가지 모델, 즉 부울리언 모델, 백터 모델, 확률 모델로 분류하고 있다. 확률 검색 모델은 검색 효율이 나쁘고, 연구 단계에 있

기 때문에 대부분의 검색 시스템에서 사용하지 않고 있다.

부울리언 검색 모델은 사용자가 요구하는 요소를 복수의 색인어로 표시하고, 각 색인어의 관계를 파악하여 질의 항목을 AND, OR, NOT 등의 부울리언 연산자로 조합하여 질의를 작성한다. 이 모델은 간단하고 유연성이 있는 질의 구성과 역화일 조작을 기초로 한 내부 처리 구조가 매우 효율적이고, 항목들 간의 동의어 관계와 구의 효과를 질의로 표현하는 것이 쉽다. 이러한 특징 때문에 큰 규모의 라이브러리에 효율적으로 적용할 수 있다. 그러나 부울리언 모델은 중요한 의미를 갖는 질의 항목에 중요성을 나타내는 가중치를 할당 할 수 없다. 또한 상관 순위를 위한 유사성 측도를 사용할 수 없기 때문에 출력될 객체의 수를 조절할 수 없으며, 객체의 등급을 정할 수 없다는 단점이 있다.

벡터 모델은 부울리언 모델의 단점을 개선하기 위한 방법으로 사용자의 요구 사항을 표현하기 위해 여러 개의 항목을 벡터로 표현하여 질의를 작성한다. 이 모델은 질의 항목의 중요도를 나타내기 위해 색인과 질의 항목의 가중치를 할당 할 수 있고, 여러 가지의 질의 평가 함수에 따라 선택된 부품들의 상관 순위를 이용하여 출력될 객체의 수와 등급을 조절할 수 있다. 그러나 질의 항목들 간의 동의어 관계나 구의 효과를 표현할 수 없다. 또한 이 모델은 질의 평가를 위해 요구되는 많은 양의 추가적인 계산 때문에 적은 규모의 라이브러리에서 효율적으로 적용되고 있다.

4.3 혼합형 검색 모델

기존 검색 모델의 분석 결과[6], 부울리언 검색 모델과 벡터 검색 모델은 부품을 위한 정보 검색 모델의 요구 사항의 전부를 지원하지 못한 것으로 분석되었다. 따라서 본 논문에서는 확장된 패싯 분류 방식에 의해서 제공되는 라이브러리 구조를 지원하고, 검색 효율을 개선시키기 위하여 새로운 혼합형 검색 모델을 제안하였다.

일반적으로 부품은 부품 식별자와 부품 자체에 관한 정보로 구성된 부품 기술자로 표현할 수 있다. 따라서 제안된 혼합형 검색 모델에서는 벡터 모델을 사용하여 각각의 부품을 일련의 항목 속성 튜플로 표현하고, 3장에서 설명한 확장된 패싯 분류 방식에 따라 분류된 각각의 부품들을 각 패싯별로 항목들에 대하여 가중치를 할당하여, 〈정의 3. 6〉에서 정의된 부품 식별자를 다음과 같이 확장하여 정의하였다.

〈정의 4.1〉 **부품 식별자(CI : component identifier)**

확장된 패싯 분류 방식에 따라 분류된 각각의 부품은 각 패싯별 항목과 가중치의 쌍으로 부품 식별자를 구성하며, 다음과 같은 구조로 라이브러리에 저장되어 있다.

$$CI = \langle T_{i1} : w_{i1},\ T_{i2} : w_{i2},\ \cdots,\ T_{ij} : w_{ij},\ \cdots,\ T_{in} : w_{in} \rangle$$

여기서, CI : 어떤 응용 영역의 i번째 부품, T_{ij} : i번째 부품에 대한 패싯 f_j의 항목, w_{ij} : i번째 부품에 대한 패싯 f_j의 항목 T_{ij}의 가중치, n : 패싯의 수

부품 기술자는 사용자의 요구 사항을 만족하는 부품을 검색하여 재사용하기 위해 필요한 부품의 수정 및 이해를 위한 유의 사항, 부품의 기능 및 상속 관계 등을 설명하는 문서화 및 관련 있는 부품들에 대한 정보들로 구성하였다. 사용자가 원하는 부품을 검색하기 위한 질의 구성은 검색하고자 하는 부품의 속성을 포함하고 있는 각각의 패싯별 항목들이 제시되면, 사용자는 가장 적합한 항목을 선택하여 질의를 구성할 수 있도록 다음과 같이 정의하였다.

〈정의 4.2〉 **질의 구성(Q : query formulation)**

사용자가 요구하는 부품에 대한 요구 사항을 질의로 표현하기 위하여, 각 패싯별로 제시되는 항목을 이용하여 구성한다. 질의의 구성 형식은 다음과 같다.

$$Q = \langle t_1,\ t_2,\ \cdots,\ t_i,\ \cdots,\ t_n \rangle$$

여기서, t_i : 각각의 패싯별로 해당하는 항목, n : 패싯의 수

사용자는 확장된 패싯 분류 방식에 따라 분류되어 있는 각 부품의 패싯별 항목을 참조하여 질의를 작성한다. 따라서 작성된 질의는 다음과 같이 확장되어 표현되며, 분류 과정에서 패싯별로 각 항목에 할당되어 있는 가중치를 이용할 수 있도록 구성되었다.

$$Q = \langle t_1 : v_{i1},\ t_2 : v_{i2},\ \cdots,\ t_i : v_{ij},\ \cdots,\ t_n : v_{in} \rangle$$

여기서, v_{ij} : 질의 패싯 f_j의 항목 t_i의 가중치

부울리언 모델의 가장 중요한 장점 중의 하나는 효율적인 매칭 함수를 제공하고 있다는 점이다. 따라서 속성들을 정렬하고, 항목의 중요성을 반영하기 위하여 부울리언 모델의 매칭 함수를 수정하여 다음과 같이 매칭 함수 M을 정의하였다.

〈정의 4.3〉 매칭 함수(M : matching function)

$$M(Q) = \{T \mid T_1 = t_1 \wedge T_2 = t_2 \wedge \cdots \wedge T_n = t_n\}$$

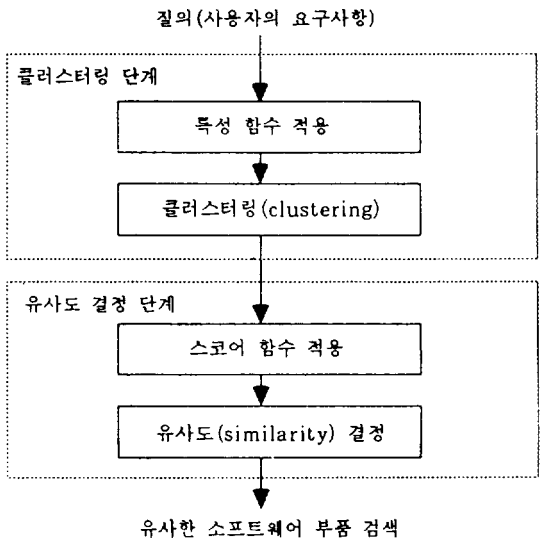
여기서, T_i : 부품의 i번째 패싯의 항목, t_i : 질의 항목의 i번째 패싯의 항목, n : 부품과 질의 패싯의 수

매칭 함수는 어떤 부품의 속성 항목 T_1과 질의 항목 t_1이 일치되고, 부품의 속성 항목 T_2와 질의 항목 t_2가 일치하며, 그리고 부품의 속성 항목 T_n과 질의 항목 t_n가 일치하는 것을 의미한다.

본 논문에서는 매칭 함수 M이 만족되지 않아 사용자가 요구하는 부품의 검색에 실패할 경우, 확장된 패싯 분류 방식에 따라 할당된 가중치를 이용하여 사용자가 요구하는 부품과 유사한 부품을 검색할 수 있도록 구성하였다. 특히 사용자가 요구하는 유사한 부품을 효율적으로 검색, 즉 재현율과 정확도를 높이기 위하여 (그림 4)와 같이 2단계로 나누어 검색을 실시한다.

4.3.1 단계 1 : 특성 함수를 이용한 클러스터링

사용자의 요구 사항과 일치하는 부품을 효율적으로 검색하기 위하여 확장된 패싯 분류 방식에 따라



(그림 4) 유사한 소프트웨어 부품의 검색 과정
(Fig. 4) Retrieval process of similar software components

분류하는 과정에서 결정된 패싯 가중치 f_j와 항목 가중치 w_{ij}를 이용하여 부품과 질의의 특성 함수를 다음과 같이 정의한다.

〈정의 4.4〉 부품 특성 함수(c_i : component characteristic function)

라이브러리에 저장되어 있는 부품의 특성을 표현하기 위하여 함수 $c_i = f(f_j, w_{ij})$를 부품의 특성 함수라 정의하고, 패싯 가중치 f_j와 항목 가중치 w_{ij}를 이용하여 다음과 같이 정의한다.

$$c_i = \sum_{j=1}^{5} (f_j \times w_{ij})$$

여기서, c_i : 각 응용 영역별 i번째 부품의 특성 함수, f_j : j번째 패싯에 할당된 패싯 가중치, w_{ij} : i번째 부품의 j번째 패싯의 항목 가중치

〈정의 4.5〉 질의 특성 함수(q_w : query characteristic function)

사용자의 요구 사항을 표현하기 위하여, 각 패싯별로 제시된 항목들을 참조하여 질의를 작성하고, 작성된 질의의 특성을 나타내는 함수 q_w

: $f(f_j, v_{ij})$를 질의 특성 함수라 정의한다. 그리고 패싯 가중치 f_j와 사용자가 작성한 질의의 항목 가중치 v_{ij}를 이용하여 다음과 같이 정의한다.

$$q_w = \sum_{j=1}^{5} (f_j \times v_{ij})$$

여기서, q_w : 사용자가 작성한 질의의 특성 함수, f_j : 질의(q_w)의 j번째 패싯의 항목에 할당된 패싯 가중치, v_{ij} : i번째 부품에 해당하는 j번째 패싯의 항목 가중치

따라서 라이브러리에 저장되어 있는 부품들을 대상으로 부품의 특성 함수를 이용하여 서로 관련 있는 부품들을 물리적으로 인접한 장소에 위치시키고, 사용자가 요구하는 부품을 검색하기 위해서 각 패싯별로 제시된 항목들을 이용하여 질의를 구성하였을 때의 질의의 특성 함수를 이용하여 관련 있는 부품들의 집합을 검색한다.

〈정의 4.6〉 **특성 함수의 차(Difference)**

검색 효율을 개선하기 위하여 서로 관련 있는 부품을 논리적으로 인접한 장소에 집합시키고, 사용자가 작성한 질의와 관련 있는 부품들의 집합을 검색하기 위한 측도로, 함수 $D_{iq} = f(c_i, q_w)$를 특성함수의 차라 정의한다. 그리고 D_{iq}는 부품의 특성 함 수 c_i와 질의의 특성 함수 q_w를 이용하여 정의한다.

$$D_{iq} = c_i - q_w,\ i = 1, \cdots, n$$
$$= \sum_{j=1}^{5} (f_j \times (w_{ij} - v_{ij}))$$

따라서 부품과 질의의 특성 함수 값의 차이 $|D_{iq}|$가 λ(임계치 : threshold value)이하인 부품들의 집합을 검색하여 부품의 특성 함수 값의 크기 순으로 출력하여, 단계 2의 입력으로 사용한다.

4.3.2 단계 2 : 유사도를 이용한 유사한 부품의 검색

단계 2에서는 사용자가 작성한 질의와 단계 1에서 검색된 부품들 사이의 유사도를 계산하여 사용자의 요구 사항과 비슷한 부품을 검색한다. 확장된 패싯 분류 방식에 따라 n개의 패싯으로 구성된 부품을 벡터 $CI_i = (T_{i1}:w_{i1}, T_{i2}:w_{i2}, \cdots, T_{in}:w_{in})$로 표현하였다. 주어진 부품들 사이에서 두 부품간의 떨어진 정도(distance)나 서로간의 유사성(proximity)의 크기를 측정하기 위하여 〈정의 4.1〉과 〈정의 4.2〉를 이용하여, 사용자가 작성한 질의와 라이브러리에 저장되어 있는 부품들 사이의 유사도를 다음과 같이 정의한다.

〈정의 4.7〉 **유사도(similarity)**

사용자가 작성한 질의와 임의의 부품들 사이의 유사성 또는 관련성을 표현하는 측도, 즉 임의의 부품과 질의 사이의 함수 $S_{iq} = f(CI, Q)$를 유사도로 정의하며, 다음의 조건 i), ii), iii)을 만족해야 한다.

조건 i) $0 \le S_{iq} \le 1$ ii) $S_{iq} = S_{qi}$
iii) $CI_i = Q \quad S_{iq} = 1$

라이브러리에 저장되어 있는 부품은 5개 패싯의 항목을 합성하여 부품 식별자를 구성하기 때문에 유사도는 k번째 패싯에 대하여 적절한 스코어(score) 를 배정하는 방법으로 정의할 수 있다. 즉 패싯별 스코어 함수 $r_k(T_{ik}, t_k)$를 다음과 같이 정의할 수 있다.

$$r_k(T_{ik}, t_k) = \begin{cases} v_{ik}, & T_{ik} = t_k \\ v_{ik} \times 0.9, & T_{ik} \neq t_k \wedge t_k = s_i \\ 0, & T_{ik} \neq t_k \wedge t_k \neq s_i \end{cases}$$

여기서 $k = 1, 2, \cdots, n$이고, s_i는 항목 T_{ik}와 동의어를 의미한다. 따라서 단계 1에서 검색된 임의의 부품 i와 사용자가 작성한 질의 q 사이의 유사도는 다음과 같이 정의된다.

$$S_{iq} = \frac{\sum_{k=1}^{n} r_k(T_{ik}, t_k)}{\sum_{k=1}^{n} v_{ik}}$$

그리고 각 패싯에 대하여 사전의 중요성이나 신

뢰성을 반영하기 위하여, 패싯 가중치 f_k를 사용하였으므로 패싯의 가중 함수 $f_k(T_{ik}, t_k)$를 k번째 성분에 대응시키면, 임의의 부품 i와 질의 q 사이의 유사도 S_{iq}는 다음과 같이 정의된다.

$$S_{iq} = \frac{\sum_{k=1}^{n} f_k(T_{ik}, t_k) \cdot r_k(T_{ik}, t_k)}{\sum_{k=1}^{n} f_k(T_{ik}, t_k) \cdot v_{ik}}$$

따라서 유사도 S_{iq}가 μ(임의의 기준치, $0 \leq \mu \leq 1$) 이상인 유사한 부품들을 검색하여 사용자의 요구 사항과 가장 적합한 부품을 검색하여 재사용 할 수 있도록 구성하였다. 따라서 제안한 검색 모델은 유사한 부품을 효율적으로 검색할 수 있고, 백터 모델에서 문제점으로 지적된 검색 시간을 단축할 수 있다. 사용자가 요구하는 부품을 검색하고, 다수의 유사한 부품을 검색하여 출력할 수 있도록 제안한 검색 모델의 검색 알고리즘을 (그림 5)에 나타내었다.

〈검색 알고리즘〉

단계 1. 확장된 패싯 분류 방식에 따라 사용자가 요구하는 부품의 특성에 따른 응용 영역을 선택한다.

단계 2. 각 패싯별로 제공되는 표준 항목들을 참조하여 질의를 작성한다.

단계 3. 작성된 질의를 매칭 함수 M에 따라 평가하여, 결과가 참이면 검색한 부품을 출력하고, 거짓이면 단계 4로 이동한다.

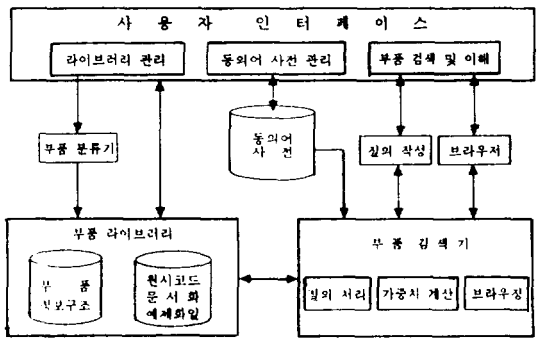
단계 4. 질의의 특성 함수 q_w를 계산하여, 부품의 특성 함수와 차이 $|D_{iq}|$가 λ이하인 소프트웨어 부품들을 검색한다.

단계 5. 유사도 S_{iq}를 계산하고 부품의 출력 순서를 결정하여, 유사한 부품들을 사용자가 선택할 수 있도록 출력한다.

(그림 5) 소프트웨어 부품의 검색 알고리즘
(Fig. 5) Retrieval algorithms of software components

5. 시스템의 구현 및 평가

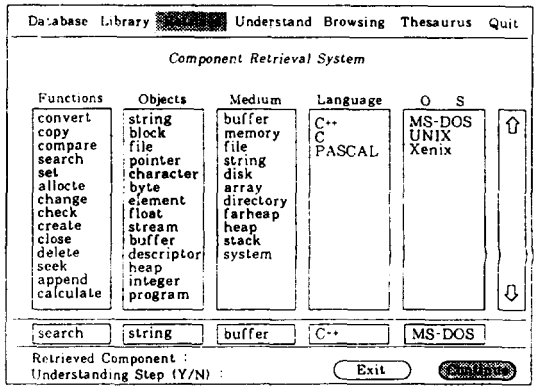
부품을 재사용하기 위하여확장된 패싯 분류 방식과 혼합형 검색 모델을 제안하여 설계하고, 프로토타입 시스템을 IBM-PC/486의 MS-DOS 환경에서 C 언어로 구현하였다. 따라서 이 프로토타입 시스템은 제안한 부품의 분류 방식 및 검색 모델의 효율을 평가하는 데 그 목적이 있다.



(그림 6) 프로토타입 시스템의 구성도
(Fig. 6) Architectures of prototype system

5.1 시스템의 구성도

부품을 분류하고 검색하기 위한 프로토타입 시스템의 전체적인 구성도는 (그림 6)과 같이 사용자 인터페이스, 부품 분류기, 부품 라이브러리, 부품 검색기, 브라우저 및 항목 동의어 사전으로 구성되었으며, 사용자가 요구하는 부품의 검색을 위한 화면은 (그림 7)과 같다. 사용자가 "Retrieve"을 선택하면, 시스템은 질의 작성을 쉽게하기 위하여 라이브러리에 저장되어 있는 항목들을 패싯별로 제시해준다. 그러면 사용자는 제시된 항목중에서 선택하여 질의를 작성한다.



(그림 7) 사용자가 요구하는 부품의 검색을 위한 화면
(Fig. 7) Retrieval screen of software components

이 프로토타입 시스템은 사용자가 요구하는 부품을 신속하게 검색하여 쉽게 이해할 수 있도록 효율적인 표현 방법을 제공하기 위하여, 그래픽 사용자 인터페이스를 이용하여 부품, 라이브러리, 그리고 항목 동의어 사전을 관리하며, 부품을 검색하기 위한 질의 작성과 브라우저를 관리한다. 각각의 부품에 대한 부품 식별자들은 관계형 데이타 베이스로 저장하며, 원시 코드, 문서화 및 예제는 각각 하나의 라이브러리를 구성하여 서로 다른 디렉토리에 저장하였다. 그리고 항목 동의어 사전은 하나의 독립된 화일로 구성하여 저장하였다.

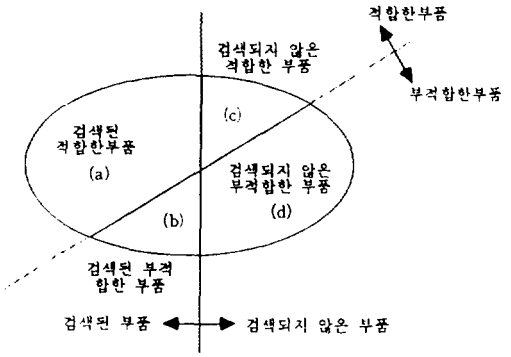
5.2 실험 및 결과 분석

검색 시스템의 성능을 평가하는 기준으로 검색 효율, 경제성, 신속성을 사용하고 있지만, 경제성과 신속성은 객관적인 기준이 필요하고, 정보 검색 시스템을 평가하는 데 큰 영향을 미치지 않기 때문에 평가 기준으로 대부분 검색 효율만을 고려하고 있다[6, 13]. 따라서 검색 시스템을 평가하는 기준으로 널리 사용하는 검색 효율과 검색 효율의 매개변수로 사용하고 있는 정확도(Precision)와 재현율(Recall)을 다음과 같이 정의하였다.

〈정의 5.1〉 검색 효율(Retrieval Effectiveness)

사용자가 작성한 질의에 의해서 검색된 부품이 사용자의 요구 사항에 일치하는 비율을 정보 검색 시스템의 검색 효율이라 정의한다. 그리고 검색 효율을 평가하는 매개변수로 정확도와 재현율을 사용한다.

검색 효율은 사용자의 요구 사항에 적합한 부품을 검색하는 검색 시스템의 능력을 의미하는 것으로 검색된 적합한 부품과 부적합한 부품, 검색되지 않은 적합한 부품과 부적합한 부품 사이의 비율로 측정된다. 라이브러리에 저장되어 있는 부품을 대상으로 검색을 수행하면 전체 라이브러리는 (그림 8)과 같이 네개의 영역으로 구분된다. (그림 8)을 이용하여 〈정의 5.2〉의 정확도(P)와 〈정의 5.3〉의 재현율(R)을 다음과 같은 공식으로 정의할 수 있다[6, 13].



(그림 8) 검색 결과에 따른 부품 라이브러리의 구분
(Fig. 8) Divisions of component library according to retrieval results

〈정의 5.2〉 정확도(Precision)

검색한 부품들 중에서 사용자의 요구 사항을 만족하는 부품이 얼마나 검색되었는 가를 나타내는 비율을 정확도라고 정의하고, 다음 공식을 이용하여 계산한다.

$$\text{정확도 } P = \frac{a}{a+b} \times 100$$

〈정의 5.3〉 재현율(Recall)

라이브러리에 저장되어 있는 부품들 중에서 검색한 부품이 사용자의 요구사항과 관련된 부품이 어느 정도 추출되었는 가를 나타내는 비율을 재현율이라 정의하고, 다음 공식을 이용하여 계산한다.

$$\text{재현율 } R = \frac{a}{a+c} \times 100$$

라이브러리에서 부품을 검색할 때, 사용자가 요구하는 부품이 빠짐없이 전부 검색되고, 또한 불필요한 부품이 하나도 나오지 않는다면 가장 이상적이라 할 것이다. 정보 검색 시스템에서 검색의 목표는 정확도와 재현율을 모두 100%의 성과를 얻기 위함이나, 실제로는 불가능하다고 밝혀졌다. 따라서 대부분의 검색 시스템에서는

그 근사치에 접근할 수 있도록 노력할 뿐이다. 많은 연구 결과에 따르면 정확도와 재현율은 한 쪽이 높으면 다른 쪽은 저하되는 상반 관계에 있다고 알려져 있다[13].

5.2.1 실험 환경 및 방법

제안한 분류 방식과 검색 모델의 효율성을 검증하기 위하여 다음과 같은 성능 평가 실험을 하였다. 재사용 지향 소프트웨어 개발에 가장 널리 사용되고 있는 C++ 언어로 작성된 부품, 기존의 절차적 프로그래밍 환경에 익숙한 소프트웨어 개발자들을 위해 C 언어로 작성된 부품, 그리고 파스칼로 작성된 함수 및 모듈을 재사용 가능한 부품으로 선정하였다.

약 500개의 부품을 제안한 분류 방식에 따라 분류하여 부품 식별자를 5개의 관계형 데이타 베이스에 저장하고, 부품 기술자를 부품 라이브러리에 저장하였다. 그리고 검색 효율을 평가하기 위하여 5개의 응용 영역별로 각각 10회씩 사용자의 요구사항을 질의로 작성하여 검색 실험을 실시하고, 정확도와 재현율을 계산하였다.

5.2.2 실험 결과

사용자가 (그림 7)을 이용하여 작성한 질의와 일치하는 부품이 라이브러리에 있으면 그 부품을 검색하여 출력하고, 일치하는 부품의 검색에 실패할 경우, 사용자의 요구와 일치하는 부품을 선택할 수 있도록 (그림 9)와 같이 다수의 유사한 부품을 검색한다. 이때 검색된 부품들을 대상으로 정확도와 재현율을 계산한다.

〈예제 1〉 사용자가 작성한 질의와 일치하는 부품의 검색에 성공

Query : search | string | buffer | C++ | MS-DOS

결 과 : strstr.cpp 해당 부품의 이름을 출력

〈예제 2〉 사용자가 작성한 질의와 일치하는 부품의 검색에 실패

Query : compare | block | buffer | C | UNIX

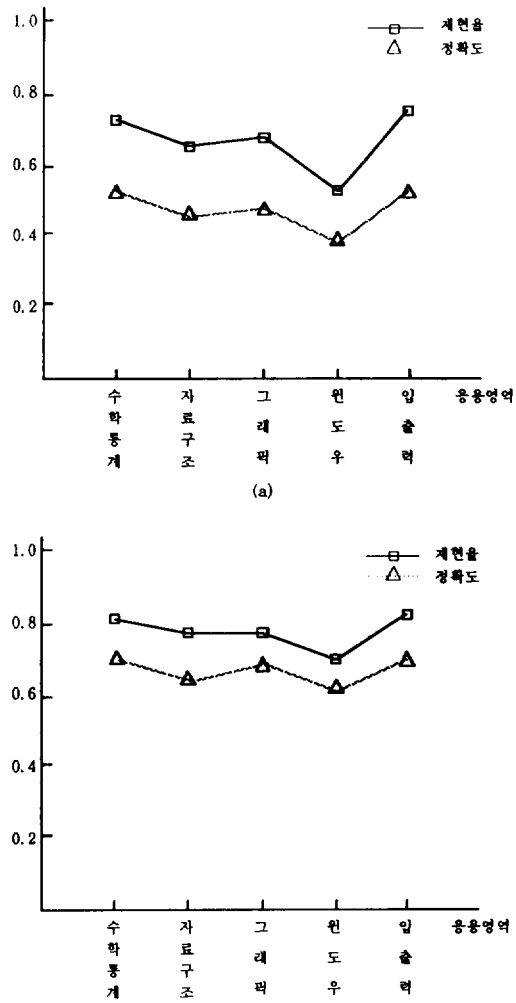
결 과 : 다음과 같이 유사한 부품을 검색하여 출력

Database Library Retrieval Understand Browsing Thesaurus Quit

Component Retrieval System

Com_name	Functions	Objects	Medium	Language	O S
bsearch1	search	element	array	C++	MS-DOS
lfind	search	element	array	C++	MS-DOS
lsearch1	search	element	array	C++	MS-DOS
memchr1	search	byte	memory	C++	MS-DOS
memcmp1	compare	block	memory	C++	MS-DOS
memicmp	compare	block	memory	C	UNIX
memcmp2	compare	byte	memory	C++	MS-DOS
strcmp1	compare	byte	memory	C++	UNIX
strcmp2	compare	string	buffer	C++	MS-DOS
strcmpi	compare	string	buffer	C	UNIX
strcoll	compare	string	buffer	C++	MS-DOS
stricmp	compare	string	buffer	C++	MS-DOS
strncmp	compare	string	buffer	C++	MS-DOS
strncmpi	compare	string	buffer	C++	MS-DOS
strnicmp	compare	string	buffer	C++	MS-DOS
strpbrk	search	character	string	C++	MS-DOS

(그림 9) 질의와 유사한 부품의 출력 결과
(Fig. 9) Retrieval results of similar components



(그림 10) 정확도와 재현율의 실험 결과
(Fig. 10) Simulation result of precision and recall

(그림 10)의 (a)는 라이브러리에 저장되어 있는 부품들과 사용자가 작성한 질의 사이의 특성 함수 값의 차를 이용한 검색 실험의 결과이고, (그림 10)의 (b)는 특성 함수 값의 차이와 유사도를 측정하여 사용자의 질의와 유사한 부품을 검색한 실험 결과이다.

각 응용 영역별로 나타난 정확도와 재현율을 살펴보면, 이론적인 평가 모델에서 제시된 값보다 상대적으로 높게 나타났다. 따라서 제안한 분류 방식으로 부품을 분류하여 라이브러리에 저장하고, 혼합형 검색 모델을 이용하여 검색 실험을 실시한 결과 검색 시간이 단축되고 검색 효율이 개선되었으며, 유사한 부품을 신속 정확하고 쉽게 검색할 수 있다는 결론을 얻을 수 있었다. 그리고 재현율이 정확도보다 상대적으로 높게 평가되어 사용자의 요구 사항을 만족하는 다수의 유사한 부품이 검색되었음을 알 수 있었다.

6. 결론 및 연구 방향

본 논문에서는 부품을 재사용하기 위한 부품의 분류 방식을 제안하고, 이 분류 방식에 따라 부품을 분류하여 라이브러리에 저장한 후, 사용자의 요구에 따라 검색하는 검색 모델을 제안하였다. 제안한 분류 방식과 검색 모델을 바탕으로 프로토타입 시스템을 설계하고 구현하였으며, 실험을 이용한 성능 평가와 기존 연구와의 비교 분석을 실시하였다.

첫째, 제안한 분류 방식은 부품의 분류가 간단하고, 유사한 부품들의 식별이 가능하였으며, 패싯 분류 방식의 특징을 이용하였기 때문에 확장성이 높다. 그리고 검색 단계에서 유사한 부품들의 검색이 가능하고, 검색 효율이 개선됨을 알 수 있었다.

둘째, 제안한 검색 모델은 질의를 작성하기 위해서 사용자에게 부품 식별자의 각 패싯별로 사용 가능한 항목들을 제시하여 사용자가 선택할 수 있도록 구성하였기 때문에 질의 작성이 간단하였다. 질의 처리 과정에서 가중치와 유사도를 이용하기 때문에 유사한 부품의 효율적인 검색이 가능했으며, 정확도와 재현율이 이론적인 모델에서 제시하는 값보다 높게 평가되어 검색 효율이 개선되었음을 알 수 있었다.

본 논문에서 제안한 분류 방식과 검색 모델이 실제 프로젝트에 적용되기 위해서는 분류 과정의 자동화, 가중치 부여를 위한 객관적인 기준의 설정 및 부품들 간의 거리를 측정할 수 있는 수리 모델의 개발에 대한 연구가 추가로 진행되어야 한다.

참 고 문 헌

[1] Bartschi, M., An Overview of Information Retrieval Subjects, IEEE Computer 18(5), pp.67~84, May 1985.

[2] Basili, V.R., Caldiera, G., and Cantone, G., A Reference Architecture for the Component Factory, ACM Transactions on Software Engineering and Methodology 1(1), pp.53~80, January 1992.

[3] Burton, B.A., Aragon, R.W., Bailey, S. A., Koehler, K.D, and Mayes, L.A., The Reusable Soft-ware Library, IEEE Software 4(4), pp.25~33, July 1987.

[4] Caldiera, G., and Basil, V.R., Identifying and Qualifying Reusable Software Components, IEEE Computer 24(2), pp.61~70, February 1991.

[5] Chen, D.J., and Lee, P.J., On the Study of Software Reuse Using Reusable C++ Components, The Journal of Systems and Software 20(1), pp.19~36, January 1993.

[6] Frakes, W.B., Baeza-Yates, W., *Information Retrieval: Data Structures & Algorithms*, Prentice-Hall, 1992.

[7] Hall, P.A.V., *Software Reuse and Reverse Engineering in Practice*, Chapman & Hall, 1992.

[8] Lanergan, R.O., and Grasso, C.A., Software Engineering with Reusable Designs and Code, IEEE Transactions on Software Engineering SE-10(5), pp.498~

501, September 1984.

[9] Liao, L.H., & Wang, F.J., Software Reuse on a Large Object-Oriented Library, ACM SIGSOFT Software Engineering Notes 18(1), pp.74~81, January 1993.

[10] Ostertag, E., Hendler, J., Prieto-Diaz, R., & Braun, C., Computing Similarity in a Reuse Library System: An AI-Based Approach, ACM Transactions on Software Engineering and Methodology 1(3), pp.205~228, July 1992.

[11] Prieto-Diaz, R., and Freeman, P., Classifying Software for Reusability, IEEE Software 4(1), pp.6~16, January 1987.

[12] Prieto-Diaz, R., Implementing Faceted Classification for Software Reuse, Communications of the ACM 34(5), pp.89~97, May 1991.

[13] Salton, G. & McGill, M.J., *Introduction to Modern Information Retrieval*, McGraw-Hill Book Company, 1983.

[14] Standish, T., An Essay on Software Reuse, IEEE Transactions on Software Engineering SE-10(5), pp.494~497, September 1984.

[15] 강문설, 김병기, 재사용 가능한 소프트웨어 부품의 검색 및 이해 방법, 한국정보과학회 논문지 20(10), pp.1519~1529, October 1993.

[16] 이경환, 객체지향 기술에 의한 소프트웨어 재사용 시스템: CARS 1.0 메뉴얼, 중앙대학교 전자계산학과 소프트웨어공학연구실, 서울, 1991.

[17] 장명섭, 정인상, 권용래, 재사용을 위한 소프트웨어 부품의 분류 및 검색 방법, 한국정보과학회 논문지 19(6), pp.602~613, Novemver 1992.

[18] 정필모, 오동근, 문헌 분류 이론(B. Buchanan저), 구미무역(주) 출판부, 서울, 1989.

강 문 설

1986년 전남대학교 전산통계학과 졸업(이학사).
1989년 전남대학교 대학원 전산통계학과 졸업(이학석사).
1994년 전남대학교 대학원 전산통계학과 졸업(이학박사).
1983년 9월~1985년 12월 제1 군수지원단 전산실(프로그래머).
1989년 4월~1992년 8월 전남대학교 전산학과 조교.
1992년 9월~현재 전남대학교 전산학과 시간강사.
관심분야 : 소프트웨어공학, 객체지향시스템, 정보검색

김 병 기

1978년 전남대학교 수학과 졸업(이학사).
1980년 전남대학교 대학원 수학과(응용수학) 졸업(이학석사).
1987년~1990년 한국정보과학회 학회지 편집위원.
1994년~현재 한국정보과학회 이사.
1981년~현재 전남대학교 전산학과 교수.
관심분야 : 소프트웨어공학, 뉴럴네트워크, 병렬처리 프로그래밍.