소프트웨어 부품의 검색을 위한 의미 유사도 측정

김 태 희[†] · 강 문 설[††]

요 약

본 논문에서는 재사용가능한 소프트웨어 부품의 분류 과정을 자동화하여 라이브러리에 구조적으로 저장하고, 사용자의 요구사항을 만족하는 부품을 효율적으로 검색하기 위하여 부품들 사이의 의미 유사도를 측정하는 방법을 제안한다. 자연어로 기술된 부품 설명서로부터 정보를 획득하여 부품의 특성을 표현하는 패싯을 결정하고, 각 패싯에 해당하는 항목을 자동으로 추출하여 부품 식별자를 구성하며, 분류된 부품들의 유사성에 따라 비슷한 특성을 갖는 부품들을 인접한 위치에 저장한다. 그리고 사용자의 요구사항을 만족하는 부품들을 검색하기 위하여 질의와 소프트웨어 라이브러리에 저장된 부품들 사이의 의미 유사도를 측정한다. 재사용가능한 부품의 검색을 위하여 의미 유사도를 이용함으로써 단순히 사용자의 질의를 만족하는 부품들의 집합을 검색할 뿐만 아니라 질의를 만족하는 정도에 따라 검색된 부품들의 상관순위를 부여하여 사용자들이 요구하는 부품의 검색 시간이 줄어들고, 전체적인 검색 효율이 개선되었다.

A Semantic Similarity Measure for Retrieving Software Components

Tae Hee Kim[†] · Moon Seol Kang[††]

ABSTRACT

In this paper, we propose a semantic similarity measure for reusable software components, which aims to provide the automatic classification process of reusable components to be stored in the structure of a software library, and to provide an efficient retrieval method of the software components satisfying the user's requirements. We have identified the facets to represent component characteristics by extracting information from the component descriptions written in a natural language, composed the software component identifiers from the automatically extracted terms corresponding to each facets, and stored them which the components in the nearest locations according to the semantic similarity of the classified components. In order to retrieve components satisfying user's requirements, we measured a semantic similarity between the queries and the stored components in the software library. As a result of using the semantic similarity to retrieve reusable components, we could not only retrieve the set of components satisfying user's queries, but also reduce the retrieval time of components of user's request. And we further improve the overall retrieval efficiency by assigning relevance ranking to the retrieved components according to the degree of query satisfaction.

1. 서 론

소프트웨어 재사용은 소프트웨어 위기(software crisis) 현상을 극복하기 위한 방안으로 널리 연구되고 있는 소프트웨어 공학의 여러 분야 중에서 관심이 상대적으로 증가하고 있는 한 분야이다. 소프트웨어 재사용은 소프트웨어 개발 과정에서 생성된 명세서, 설계 정

† 정 회 원:전남대학교 전산학과 박사과정
†† 종신회원:광주대학교 전자계산학과 조교수
논문접수:1996년 2월 8일, 심사완료:1996년 9월 16일

보, 프로그램 및 문서화 등과 같은 소프트웨어 부품을 새로운 소프트웨어 개발 과정에서 재사용 하는 것이다. 따라서, 소프트웨어 재사용은 이미 개발된 부품들을 새로운 소프트웨어를 개발하는 과정에서 재사용함으로써 개발 시간을 단축시키고, 동일한 부품이 다수의 다른 소프트웨어 개발 과정에서 테스트되어 재사용되기 때문에 소프트웨어 개발자의 생산성과 소프트웨어의 품질 개선을 보장하고 있다[2, 4].

소프트웨어 재사용을 위해서는 소프트웨어 개발자들이 재사용가능한 소프트웨어 부품을 효율적으로 활용할 수 있도록 지원하는 3개의 오퍼레이션, 즉 재사용가능한 소프트웨어 부품을 인식하고, 사용자의 요구사항과 일치하는 부품의 선택, 그리고 선택한 부품을 소프트웨어 개발 과정에 효율적으로 적용하기 위한 오퍼레이션이 필요하다. 한편, 마지막 오퍼레이션인 선택한 부품의 적용은 자동화가 매우 어려운 단계이기 때문에 부품의 재사용을 위해서 부품의 인식과 선택 단계에 대한 자동화 방안이 많은 연구 대상이 되고 있다[2, 4, 15].

소프트웨어 부품을 재사용하기 위한 분류 방식과 검색 모델은 소프트웨어 재사용을 실용화시키기 위해 매우 중요한 역할을 한다. 부품을 분류하는 방식에 따라 분류된 부품을 포함하는 소프트웨어 라이브러리의 구조와 사용자의 요구사항에 일치하는 부품을 검색하는 검색 모델의 선택이 결정되기 때문에 분류 방식과 검색 모델은 서로 밀접한 관계를 유지한다. 소프트웨어 부품을 분류하기 위해 널리 알려진 분류 방식에는 열거형 분류 방식(enumerative classification scheme), 패싯 분류 방식(faceted classification scheme), 그리고 이 두 방식의 장점을 이용한 혼합형 분류 방식(hybrid classification scheme) 및 기존 문서의 유사성(similarity)을 계산하는 방법 등이 있다[1, 8, 11, 12, 13]. 검색 모델에 관한 연구는 일반적으로 부품의 검색 결과로서 얻어진 부품들이 사용자가 원하는 모든 부품을 포함하지 못하고, 또한 검색된 모든 부품들이 사용자가 원하는 부품이 되지 못하기 때문에 매우 중요하며, 효율적인 정보검색 시스템의 구현을 위해서는 검색 모델에 관한 연구가 반드시 선행되어야 한다. 지금까지 널리 알려진 검색 모델로는 불리안 모델(boolean model), 퍼지 집합 모델(fuzzy set model), 확장된 불리안 모델(extended boolean model), 벡터 공간 모델(vector space model) 및 확률 모델(probability model) 등이 있다[3, 6, 10, 14].

본 논문에서는 재사용가능한 소프트웨어 부품의 분류 과정을 자동화하여 소프트웨어 라이브러리에 구조적으로 저장하고, 사용자의 요구를 만족하는 부품을 효율적으로 검색하기 위하여 부품들 사이의 의미 유사도(semantic similarity)를 이용하는 방법을 제안한다. 재사용가능한 부품을 분류하기 위해서 패싯 분류 방식의 분류 과정을 자동화한다. 즉, 자연어로 기술된 부품 설명서를 분석하여 부품의 특성을 표현하는 패싯을 결정하고, 각 패싯에 해당하는 항목(term)을 자동으로 추출하여 부품 식별자를 구성하며, 분류된 부품들의 유사성을 조사하여 비슷한 특성을 갖는 부품들을 인접한 위치에 저장한다. 그리고 사용자의 요구사항을 만족하는 부품들을 검색하기 위하여 필요한 항목들의 벡터로 질의를 구성하고, 구성된 질의와 라이브러리에 저장되어 있는 부품들 사이의 의미 유사도를 측정하여 사용자의 요구사항과 정확하게 일치하는 부품, 또는 비슷한 특성을 갖는 부품을 효율적으로 검색한다. 재사용가능한 부품의 검색을 위하여 의미 유사도를 이용함으로써 단순히 사용자의 질의를 만족하는 부품들의 집합을 검색할 뿐만 아니라 질의를 만족하는 정도에 따라 검색된 부품들의 상관순위를 부여하여 사용자들이 요구하는 부품의 검색 시간을 줄이고, 전체적인 검색 효율을 개선할 수 있다.

본 논문은 다음과 같이 구성한다. 2장에서는 사용자의 요구사항에 따라 부품을 검색하는 검색 모델의 특징 및 연구 방향에 대해 설명하고, 3장에서는 패싯 분류 방식에서 항목 추출 과정을 자동화시킨 개선된 패싯 분류 방식과 유사도를 이용하는 검색 모델, 부품의 분류 및 검색 과정에서 사용되는 의미 유사도의 측정 방법을 설명하며, 4장에서는 부품들 사이의 의미 유사도 측정 결과를 기반으로 한 검색 모델을 이용한 검색 실험과 부품들의 상관 순위, 정확도 및 재현율을 기준으로 실험 결과를 검토하고, 5장에서는 결론 및 향후 연구방향을 기술한다.

2. 관련연구

대부분의 검색 모델은 사용자의 요구사항과의 유사성(similarity)이 일정한 기준을 넘고, 순위화된 재

사용가능한 후보들의 집합을 검색한다. 그러나 사용자들은 부품을 선택하기 위해 많은 노력을 투자하는 것을 원하지 않으며, 대부분의 경우 검색된 후보 부품들 뿐만 아니라 높은 순위에 있는 부품들조차 완전하게 분석하지 않는다. 사용자는 가장 적합한 부품이 검색된 후보 리스트의 정상에 있다고 가정하고, 첫번째 부품에 대해 관련된 정보만 시험한다. 만약 사용자의 요구사항을 만족하는 부품이 없으면 질의를 수정하거나 재 작성하며, 대부분의 경우에는 검색을 포기한다. 따라서 검색 모델은 사용자들의 요구사항을 만족할 수 있도록 후보 부품들의 집합으로부터 요구하지 않는 부품을 제외시키고, 사용자의 보다 정확한 요구사항을 만족하는 하나의 부품만을 검색하여 높은 정확성을 보장해야 한다[1, 2, 6, 10].

재사용을 위한 부품의 검색 모델[2, 3, 6, 10, 15]로는 불리안 검색 모델과 벡터 검색 모델이 가장 널리 사용되고 있다. 이것은 불리안 검색 모델이 사용자의 요구사항을 정확하고 간단하게 표현할 수 있고, 질의 작성과 구현이 용이하며, 구현된 시스템이 짧은 검색 시간을 제공하기 때문이다. 그러나 불리안 검색 모델은 각 항목(terms)의 상대적인 중요도를 표시할 수 없으며, 질의를 만족하는 정도를 표현하는 부품값(component value)을 계산할 수 없기 때문에 검색된 부품들을 질의를 만족하는 정도에 따라 정렬할 수 없다.

벡터 공간 검색 모델은 불리안 검색 모델의 단점을 개선하기 위한 방법으로 사용자의 요구 사항을 표현하기 위해 여러 개의 항목을 벡터로 표현하여 질의를 작성한다. 이 모델은 질의 항목의 중요도를 나타내기 위해 부품 항목과 질의 항목의 가중치를 할당 할 수 있고, 여러 가지의 질의 평가 함수에 따라 선택된 부품들의 상관 순위(relevance ranking)를 이용하여 출력될 객체의 수와 등급을 조절할 수 있다. 그러나 질의 항목들간의 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.

퍼지집합 모델은 불리안 모델과 동일한 형식의 질의를 사용하지만, 각 부품이 가지고 있는 항목에 가중치를 부여하고 불리안 연산자를 처리하기 위한 함수를 정의하여 사용한다. 이 모델은 사용자가 원하는 정보를 보다 정확하게 표현할 수 있고, 불리안 모델보다 융통성 있으며 검색된 부품들은 서로 다른 함수 값을 가지므로 순위를 부여할 수 있다. 그러나 부품의 내용을 정확히 알지 못하면 올바른 가중치를 부여할 수 없고, 질의 전체에 대한 부품의 함수 값이 일부분의 항목에 의해 결정될 수도 있다.

확장된 불리안 모델은 퍼지집합 모델의 단점을 개선하기 위하여 질의와 부품의 유사성을 거리 함수를 이용하여 계산하는 방법을 사용한다. 이 모델은 퍼지집합 모델의 장점을 모두 유지하면서 질의 전체에 대한 부품의 함수 값이 일부분의 항목에 의해 결정될 수 있는 단점을 해결했다. 즉, 색인어로 가지고 있는 항목의 개수를 고려하여 부품과 질의 사이의 유사도를 계산하므로 많은 항목을 색인어로 가질수록 증가된 함수를 갖게된다. 그러나 가중치를 부여하는 과정이 어렵고, 불리안 연산에 대한 처리 법칙 중 교환법칙, 결합법칙 등 상당수가 유효하지 않으며, 부품에 대한 함수 결과 값을 계산하는 과정이 복잡하다.

확률 모델은 어떤 질의에 대하여 각 부품의 적합할 확률과 부적합할 확률을 각각 계산하여 적합할 확률이 부적합할 확률보다 큰 부품들을 검색하는 기법이다. 이 모델은 적합할 확률이 높은 부품들만을 검색하여 검색 효율을 향상시킬 수 있고, 이용자 피이드백을 적용하기 쉬우며, 일정 기준치를 넘는 함수 값을 갖는 부품들이 검색되므로 검색된 부품들에 순위를 부여할 수 있다. 그러나 질의가 항목들의 벡터들로 표현되므로 각 항목들간의 논리적 관계성을 표현할 수 없으며, 적합한 부품과 부적합한 부품에 관한 정보가 검색 과정 이전에 미리 확보되어야 하는 어려움이 있다.

3. 분류 방식과 검색 모델

소프트웨어 부품의 분류 방식과 검색 모델은 소프트웨어 재사용을 실용화시키기 위한 핵심 요소들 중에서 가장 중요한 역할을 한다. 부품을 분류하는 방식에 따라 사용자의 요구사항과 일치하는 부품을 검색하는 방법, 부품을 포함하는 라이브러리의 확장성 등이 결정되기 때문에 소프트웨어 재사용에서는 부품을 분류하는 방식과 검색 모델은 매우 밀접한 관계를 유지한다. 지금까지 널리 알려진 소프트웨어 부품의 분류 방식으로는 열거형 분류 방식과 패싯 분류

방식, 그리고 이 두 방식의 장점을 이용한 혼합형 분류 방식 등이 있다[10, 12, 13].

3.1 개선된 패싯 분류 방식

패싯 분류 방식은 소프트웨어 부품들이 갖는 공통적인 측면의 특성을 합성하여 하나의 패싯으로 표현하고, 하나의 부품은 여러 개의 패싯으로 나타낼 수 있다. 소프트웨어 부품을 기능(function), 객체(object), 매체(medium), 시스템 유형(system style), 기능 영역(function area) 및 설정(setting) 등의 패싯으로 분류하고, 각 패싯은 그를 구성하는 요소인 항목(terms)을 포함시켰다. 이 분류 방법은 부품들이 갖는 기본적인 클래스만 표현하므로 분류가 간단하며 이해하기 쉽고, 확장이 용이하다. 그러나 패싯의 항목이 많아지면 그들 사이에 관련성의 명세와 동의어 처리가 어렵고, 검색 시간이 길어진다[12, 13].

개선된 패싯 분류 방식(advanced faceted classification scheme)은 패싯 분류 방식에서 패싯을 결정하고, 각 패싯에 해당하는 항목을 추출하는 과정을 자동화시킨 분류 방식이다. 즉, 소프트웨어 부품의 설명서를 자연어 처리 기법[5, 7]을 이용하여 대표적인 소프트웨어 부품들의 특성을 분석하여 획득한 정보들로부터 부품들이 가지고 있는 공통의 특성을 표현하는 패싯을 결정하고, 결정된 각 패싯에 해당하는 항목들을 설명서로부터 자동 추출하여 부품 식별자를 구성한다[9].

패싯(facets)은 소프트웨어 부품 설명서의 핵심 문장에서 명사구나 전치사구가 동사와 의미적으로 관계되는 방법을 표현하고 있다. 따라서 부품 설명서로부터 핵심 동사, 명사구, 전치사구 등을 분석하여 각 패싯을 결정하기 위한 속성 값들을 추출하고, 중복되지 않으면서 부품의 특성을 표현할 수 있는 패싯을 결정할 수 있다. 패싯은 부품의 기본적인 기능과 부품을 구현하고 실행하기 위한 환경을 기술하기 위하여 부품이 수행하는 기능, 기능을 수행하는 대상, 매체 및 기능을 수행하기 위한 환경 등을 포함한다.

〈정의 1〉 패싯 속성(facets attribute)

소프트웨어 부품 설명서의 핵심 문장에서 특정 패싯의 존재를 표현하는 요소들을 패싯 속성(facets attribute)으로 정의하며, 주로 전치사가 패싯 속성에 해당한다.

패싯 속성은 부품 설명서의 문장 표현으로부터 패싯을 결정할 수 있는 성질을 지닌 단어(주로 전치사)들을 의미하고, 이 패싯 속성에 관련되는 단어들을 이용하여 패싯 값(항목)을 추출할 수 있다.

본 논문에서 실험 대상으로 선정한 운영체제(MS-DOS, Linux) 명령어의 설명서에서 핵심 문장을 분석하여 정의한 패싯의 기본 집합 일부와 설명서를 분석하여 식별한 패싯 속성들의 예를 〈표 1〉에 나타내고 있다. 부품 설명서의 문장에서 전치사의 의미와 정의된 패싯을 갖는 전치사의 관계를 분석함으로서 식별이 가능하다. 한편, 일부의 패싯 속성들은 하나 이상의 패싯을 유도할 수 있으나 문장에서 관련된 전치사구에 대해 일련의 경험을 적용함으로써 다른 해석을 하지 않도록 한다.

〈표 1〉 패싯의 기본 집합과 패싯 속성
〈Table 1〉 basic set of facets and facet

패싯(facets)	특 성	패 싯 속 성
기능(function)	부품이 수행하는 기본 기능	필수 문장에 있는 핵심 동사
객체(objects)	기능 수행을 위해 적용되는 대상	'by using', with, using
매체(medium)	객체가 저장되거나 실제 기능이 수행되는 장소	at, in, into, within, through
언어(language)	부품의 원시코드를 작성한 프로그래밍 언어	C, C++, PASCAL, VISUAL BASIC
운영체제(OS)	부품의 개발 및 실행을 위한 시스템의 운영체제	MS-DOS, Linux, Windows95
조건(condition)	기능을 실행시키기 위한 전제 조건	as, at, except, how, unless, until

〈정의 2〉 패싯 결정을 위한 제약 조건 및 경험(constraints and heuristics)

다음의 제약 조건 및 경험은 소프트웨어 설명서의 핵심 문장에서 패싯을 식별하기 위해 구문 및 의미 규칙을 표현한다.

1. 하나의 패싯이 두번 나타날 수 없다.
2. '기능' 패싯은 문장에서 핵심 동사로부터 추출한다.
3. '매체' 패싯은 패싯을 위한 속성을 포함하는 문장에서 전치사구가 없다면 문장에 있는 객체로부터 직접 생성한다.
4. 기타 패싯은 패싯을 위한 속성을 포함하는 문장의 전치사구로부터 추출한다.

일반적으로 소프트웨어 부품은 부품의 특성을 표현하는 부품 식별자(component identifier)와 부품 자체에 관한 관련 정보로 구성된 부품 기술자(component descriptor)로 표현할 수 있다. 즉, 부품 식별자는 분류하고자 하는 부품을 표현하기 위해서 패싯 단위로 분류된 기본적인 클래스들을 모아서 구성한다. 따라서 패싯을 합성하는 과정은 일관성 있는 부품의 분류와 분류 방법의 확장성을 위하여 패싯들의 합성 순서를 반드시 고려해야 한다.

〈정의 3〉 부품 식별자(C_i : component identifier)
소프트웨어 부품의 특성을 표현하기 위해서 패싯 결정 단계에서 결정된 각 패싯들에 해당하는 항목들의 집합을 부품 설명서에서 추출하여 부품 식별자를 합성하고 다음과 같이 구성한다.

$$C_i = \langle t_{i1}, t_{i2}, t_{i3}, \cdots, t_{ij}, \cdots, t_{in} \rangle$$

여기서, C_i는 라이브러리에 저장될 i번째 소프트웨어 부품이고, t_{ij}는 i번째 소프트웨어 부품의 j번째 패싯의 항목(terms)이며, n는 패싯의 수를 의미한다.

3.2 의미 유사도 측정

소프트웨어 라이브러리의 구조적인 구축은 소프트웨어 재사용의 성공을 위해서 매우 중요한 요소가 된다. 즉, 구축된 소프트웨어 라이브러리의 구조는 사용자가 요구하는 부품을 쉽고 효율적으로 검색이 가능하도록 비슷한 특성을 갖는 부품들을 서로 가까운 위치에 저장시키고, 유사한 부품들의 식별이 가능해야 한다.

개선된 패싯 분류 방식에 따라 분류되어 라이브러리에 저장되는 부품들간의 유사도를 측정하여 의미적으로 기능이 유사한 부품들을 서로 가까운 위치에 저장하여 라이브러리를 구조적으로 구축하고, 사용자가 요구하는 부품을 효율적으로 검색하기 위하여 부품들간의 의미 유사도를 이용한다. 따라서, 라이브러리에 저장되는 부품들 사이 또는 사용자의 요구사항을 나타내는 질의와 라이브러리에 저장된 부품들에 대해 두 부품간의 떨어진 정도(거리:distance)나 서로간의 의미적 유사성(proximity)의 크기를 측정하기 위하여 부품들 사이의 의미 유사도를 다음과 같이 정의한다.

〈정의 4〉 의미 유사도(S_{ij} : semantic similarity)
임의의 소프트웨어 부품들 사이의 유사성 또는 관련성을 측정하는 측도, 즉 임의의 부품 C_i와 부품 C_j 사이의 함수 $S_{ij}=f(C_i, C_j)$를 의미 유사도로 정의하며, 다음의 조건 (1), (2), (3)을 만족해야 한다.

조건 (1)$0 \le S_{ij} \le 1$, (2)$S_{ij}=S_{ji}$, (3)$C_i=C_j \Rightarrow S_{ij}=1$

개선된 패싯 분류 방식에 따라 분류되어 라이브러리에 저장되는 부품은 n개 패싯의 항목을 합성하여 부품 식별자를 〈정의 3〉과 같이 구성하기 때문에 유사도는 k번째 패싯에 대하여 적절한 스코어(score)를 배정하는 방법으로 정의할 수 있다. 즉 패싯별 스코어 함수 $r_k(t_{ik}, t_{jk})$를 다음과 같이 정의할 수 있다.

$$r_k(t_{ik}, t_{jk}) = \begin{cases} x_k, & t_{ik}, t_{jk} \\ x_k \times \alpha, & \exists s_i \in \{t_{ik}\text{'s synonym}\}\,[t_{ik} \neq t_{jk} \wedge t_{jk} = s_i] \\ 0, & \forall s_i \in \{t_{ik}\text{'s synonym}\}\,[t_{ik} \neq t_{jk} \wedge t_{jk} = s_i] \end{cases} \quad (1)$$

여기서 $k=1, 2, \cdots, n$이고, x_k는 스코어 함수의 값($x_k=1$), s_i는 항목 t_{ik}와의 동의어, α는 항목 tik와 동의어 s_i 사이의 거리($0<\alpha<1.0$)를 의미한다. 따라서 라이브러리에 저장되는 임의의 부품 C_i와 부품 C_j 사이의 유사도 및 질의 Q_i와 라이브러리에 저장된 부품 j 사이의 유사도는 다음과 같이 정의된다.

$$S_{ij} = \frac{\sum_{k=1}^{n} r_k(t_{ik}, t_{jk})}{\sum_{k=1}^{n} x_k} \quad (2)$$

〈정의 5〉 패싯의 가중치 함수(f_k : weight function of facets)
개선된 패싯 분류 방식에서 각각의 패싯들이 소프트웨어 부품의 특성을 어느 정도 설명하고, 표현할 수 있는 가에 따라 각 패싯에 가중치를 할당하여($f_k=\lambda_j$), 이 가중치를 패싯의 가중치 함수(weight function of facets)라 정의하고 f_k로 표현한다.

패싯의 가중치 함수는 분류된 부품을 라이브러리에 저장하는 과정에서 비슷한 부품들을 식별하기 위해서 의미 유사도를 계산하는 데 이용한다. 따라서 패싯의 가중치 함수는 사용자들의 요구사항에 따라 결정되는 데, 사용자가 요구하는 부품의 특성을 표현할 수 있는 정도, 즉 어떤 패싯이 부품의 특성을 얼마나 설명하고 있는가를 가중치로 표현하기 위해 임계값 (threshold weight, $0 < \lambda_j \leq 1.0$)을 할당할 수 있도록 구성한다. 일반적으로 사용자가 부품을 재사용하기 위해서는 그 부품이 어떤 기능을 수행하고 있는가를 가장 먼저 고려한다고 가정하고, 본 논문에서는 실험적으로 가중치 $\lambda_j = \langle 0.85, 0.55, 0.30, 0.15, 0.15 \rangle$로 결정하여 실험한다.

그리고 각 패싯에 대하여 사전의 중요성이나 신뢰성을 반영하기 위하여, 〈정의 7〉의 패싯 가중치 f_k를 사용할 수 있으므로, 패싯의 가중 함수 $f_k(t_{ik}, t_{jk})$를 k 번째 성분에 대응시키면, 임의의 부품 i와 부품 j 사이의 유사도 S_{ij}는 다음과 같이 정의된다.

$$S_{ij} = \frac{\sum_{k=1}^{n} f_k(t_{ik}, t_{jk}) \cdot r_k(t_{ik}, t_{jk})}{\sum_{k=1}^{n} f_k(t_{ik}, t_{jk}) \cdot x_k} \quad (3)$$

소프트웨어 부품들 사이의 의미 유사도를 식 (3)을 이용하여 계산하고, 유사한 부품들을 가까운 위치에 저장하여 소프트웨어 라이브러리는 구조적으로 구축된다. 이렇게 구조화된 라이브러리는 소프트웨어 부품의 재사용성을 최대화하고, 사용자의 요구사항과 일치한 부품이나 유사한 부품들을 효율적으로 식별하여 검색할 수 있도록 지원한다.

개선된 패싯 분류 방식에서 소프트웨어 부품을 분류하는 과정을 〈정의 1, 2, 3〉과 〈정의 4, 5〉을 이용하여 다음과 같이 알고리즘으로 정의한다.

단계 1 〈패싯과 패싯의 속성을 결정〉
- 1.1 대표적인 부품 설명서를 분석하여 전체 부품의 특성을 표현하기 위한 패싯을 결정한다.
- 1.2 부품의 특성을 표현하기 위한 패싯의 합성 순서를 결정한다.
- 1.3 특정의 패싯의 존재여부를 표현하는 요소들인 패싯 속성을 추출한다.

For i = 1, 2, …, until end-of-components Do

단계 2 〈부품의 분류와 항목의 자동 추출〉
- 2.1 패싯 속성을 이용하여 각 패싯에 해당하는 항목들을 부품 설명서로부터 추출한다.
- 2.2 추출된 항목들이 각 패싯의 특성에 적합한지 검토하고 수정한다.

단계 3 〈부품 식별자의 구성〉
- 3.1 각 패싯별로 추출된 항목들을 합성하여 부품 식별자를 구성한다.

단계 4 〈부품들의 의미 유사도 측정 및 라이브러리에 저장〉
- 4.1 부품들의 유사성을 결정하기 위하여 식 (3)을 이용하여 의미 유사도를 계산한다.
- 4.2 의미 유사도에 따라 부품들을 클러스터링시켜 라이브러리에 저장한다.

End

3.3 검색 모델

사용자가 원하는 소프트웨어 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 6〉 질의 구성(Q : query formulation)
사용자가 요구하는 소프트웨어 부품에 대한 요구사항을 질의로 표현하기 위하여, 각 패싯별로 필요한 항목을 요구사항으로부터 추출하여 구성하며, 질의 구성 형식은 다음과 같다.

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

여기서, t_i: 각각의 패싯별로 해당하는 항목, n: 패싯의 수이다. 사용자는 개선된 패싯 분류 방식에 따라 분류되어 라이브러리에 저장되어 있는 각 부품의 패싯별 항목을 참조하여 질의를 작성한다. 불리안 모델의 가장 중요한 장점 중의 하나는 효율적인 매칭 함수를 제공하고 있다는 점이다. 따라서 속성들을 정렬하고, 항목의 중요성을 반영하기 위하여 불리안 모델의 매칭 함수를 수정하여 다음과 같이 매칭 함수 M을 정의하였다.

〈정의 7〉 매칭 함수(M : matching function)

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

여기서, T_i: 부품의 i번째 패싯의 항목, t_i: 질의 항목의 i번째 패싯의 항목, n: 부품과 질의 패싯의 수이다. 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. 즉, 부품 튜플의 속성 항목들과 질의 튜플의 항목들이 정확하게 일대일(1:1)로 일치할 때, 사용자가 원하는 부품을 검색할 수 있음을 의미한다. 따라서 이 매칭 함수가 정확하게 일치되지 않을 때는 부품을 검색하지 못하게 된다. 사용자의 요구사항에 따라 소프트웨어 부품을 검색하는 알고리즘은 다음과 같다.

단계 1 사용자가 요구하는 부품에 대한 요구사항을 수집하여 정형화시킨다.
단계 2 정형화된 사용자의 요구사항으로부터 필요한 항목을 추출하여 질의를 구성한다.
단계 3 매칭함수를 이용하여 질의를 평가하고,
  3.1 평가 결과가 참이면, 검색한 부품을 출력시키고 단계 1로 이동
  3.2 평가 결과가 거짓이면, 단계 4로 이동
단계 4 질의와 라이브러리에 저장된 부품들 사이의 의미 유사도를 계산한다.
단계 5 의미 유사도 값에 따라 부품들의 출력순서를 결정하여 적합한 부품을 사용자가 선택할 수 있도록 출력한다.

본 논문에서는 매칭 함수 M이 만족되지 않아 사용자가 요구하는 부품의 검색에 실패할 경우, 3.2절에서 소개되는 의미 유사도를 이용하여 사용자가 요구하는 부품과 유사한 부품을 검색할 수 있도록 구성하였다. 특히, 의미 유사도는 사용자가 요구하는 유사한 부품을 효율적으로 검색, 즉 재현율과 정확도를 높이기 위하여 사용된다.

4. 실험 및 분석

이 장에서는 사용자의 요구사항을 질의로 표현하고, 부품들 사이의 의미 유사도 측정 결과를 이용하는 검색 모델을 설명하고, 개발된 검색 모델을 이용하여 부품을 검색하는 실험과 부품들의 상관 순위, 정확도 및 재현율을 기준으로 실험 결과를 비교 분석한다.

4.1 실험 환경 및 방법

사용자가 요구하는 소프트웨어 부품의 효율적인 검색을 위해 제안한 검색 모델의 타당성을 평가하기 위하여 펜티엄(IBM-PC)의 MS-DOS 환경에서 부품의 검색 실험을 실시하였다. 본 논문에서는 재사용가능한 소프트웨어 부품의 예제로서 운영체제(MS-DOS, Linux) 명령어를 선택한다. 운영체제 명령어는 소프트웨어 부품을 분류하고 검색하는 방법의 효율성 평가에 적합하고, 부품의 설명서는 다양한 명령어의 설명(메뉴얼)에서 쉽게 획득이 가능하며, 운영체제 메뉴얼을 소프트웨어 문서화의 실세계 예제로 간주할 수 있다. 그리고 운영체제의 다양한 명령어들의 기능 또는 행위는 잘 알려져 있기 때문에 분류 방식과 검색 모델의 평가와 소프트웨어 라이브러리의 구조적인 구축이 용이하다.

약 325개의 운영체제 명령어를 대상으로 개선된 패싯 분류 방식의 자동화된 분류 절차에 따라 운영체제 메뉴얼의 설명서를 분석하여 부품의 기본적인 특성을 결정하는 패싯을 선정하고, 항목을 자동으로 추출하여 부품 식별자를 구성한다. 그리고 유사한 부품을 식별하기 위하여 부품 식별자를 이용하여 의미 유사도를 측정하고 구조화된 라이브러리를 구축하며, 사용자가 요구하는 부품을 검색할 수 있도록 요구사항을 질의로 작성하여 검색 실험을 실시한다.

일반적으로 검색 모델의 효율성을 평가하기 위하여 부품의 상관순위(relevance ranking), 재현율(recall) 및 정확도(precision)를 이용한다. 검색 효율은 사용자의 요구 사항과 관련 있는 부품을 검색하는 검색 시스템의 능력을 의미하는 것으로 검색된 관련 있는 부품과 관련 없는 부품, 검색되지 않은 관련 있는 부품과 관련 없는 부품 사이의 비율로 측정된다. 상관순위는 질의를 만족하여 검색된 관련된 부품들의 순서화된 나열 상태이고, 재현율은 라이브러리에 저장되어 있는 관련 있는 부품중에서 검색된 관련 있는 부품의 비율, 그리고 정확도는 검색된 부품 중에서 관련 있

는 부품의 비율을 말한다. 예를 들어 150개의 부품으로 구성된 라이브러리에 질의에 관련 있는 부품이 8개가 있다고 가정하자. 사용자가 질의를 작성하여 10개의 부품을 검색하였을 때, 검색된 부품 중에서 질의에 관련된 부품이 6개라 하면 재현율과 정확도는 각각 0.75, 0.60이 된다. 한편, 재현율과 정확도는 각각 1.00을 목표로 하고 있으나 실제로는 불가능하다고 밝혀졌으며, 대부분의 검색 시스템은 0.25(낮은 재현율, 낮은 정확도), 0.50(중간 재현율, 중간 정확도), 0.75(높은 재현율, 높은 정확도)로 나타내고 있다[3, 10, 14].

4.2 검색 실험

일반적으로 사용자는 재사용하기 위한 부품을 검색하기 위해 요구사항을 질의로 표현하고, 검색 시스템은 사용자의 질의를 만족하는 부품을 라이브러리에서 검색하여 사용자에게 제공한다. 따라서 비정형적으로 표현된 사용자의 요구사항을 정형화시켜 질의로 구성하는 방법과 질의를 만족하는 부품을 효율적으로 검색하기 위한 검색 모델은 검색 효율을 개선시키기 위해 매우 중요한 요소로 작용한다.

제안한 검색 모델에서는 〈정의 6〉를 이용하여, 사용자의 요구사항으로부터 개선된 패싯 분류 방식에서 결정된 패싯별로 항목을 추출하여 간단하게 질의를 구성한다. 예를 들면, '하나의 플로피 디스크 전체 내용을 다른 플로피 디스크로 복사하라'는 사용자의 요구사항에 대한 질의는 다음과 같이 구성할 수 있다.

【예 1】 사용자의 요구사항에 대한 질의 구성

요구사항 : Copies the entire contents of on floppy disk to another floppy disk

질 의 : $Q_1 = \langle copy, contents, disk, C, MS\text{-}DOS \rangle$

검색결과 : 운영체제 명령어 "DISKCOPY"를 검색하여 출력

사용자가 작성한 질의는 〈정의 7〉의 매칭 함수를 이용하여 평가한다. 매칭 함수의 결과가 참이면 사용자의 요구사항과 일치하는 부품('DISKCOPY' 명령)이 검색되지만, 매칭 함수의 결과가 거짓이면 사용자가 요구하는 부품을 검색하지 못한다. 이 경우에 제안한 검색 모델에서는 사용자가 작성한 질의와 라이브러리에 저장된 부품들 사이의 의미 유사도를 계산하고, 부품들의 출력 순서를 결정하며, 유사한 부품들을 검색하여 사용자가 부품을 선택할 수 있도록 한다. 예를들어, [예 1]의 요구사항에 대하여 사용자가 질의를 $Q_2 = \langle copy, file, disk, C, MS\text{-}DOS \rangle$로 작성했다고 가정하면, 이 질의를 만족하는 부품은 라이브러리에서 검색할 수 없다. 따라서, 사용자의 요구사항과 유사한 부품을 검색할 수 있도록 [예 2]와 같이 질의와 라이브러리에 저장되어 있는 부품들 사이의 의미 유사도는 식 (3)을 이용하여 계산하고, 의미 유사도가 0.5보다 큰 부품중에서 질의와 부품 사이의 상관순위(의미 유사도 값이 큰 부품을 먼저 출력)에 따라 출력한다.

【예 2】 질의 Q_2와 라이브러리에 저장된 부품 C_i 사이의 의미 유사도 계산 결과

질 의 : $Q_2 = \langle copy, file, disk, C, MS\text{-}DOS \rangle$

검색 :

부품 이름	기능패싯	객체패싯	매체패싯	언어패싯	운영체제	유사도
질의 Q_2	copy	file	disk	C	MS-DOS	
copy	copy	file	directory	C	MS-DOS	0.9550
diskcopy	copy	contents	disk	C	MS-DOS	0.9175
cp	copy	file	directory	C	Linux	0.8800
xcopy	compare	directory	directory	C	MS-DOS	0.8725
cpio	copy	file	tape	C	Linux	0.8500
mv	move	file	directory	C	Linux	0.7525
fc	compare	file	directory	C	MS-DOS	0.7425
diskcomp	compare	contents	disk	C	MS-DOS	0.7050
comp	compare	contents	files	C	MS-DOS	0.6600
cmp	compare	contents	files	C	Linux	0.5850

질의 Q_2와 부품들 사이의 의미 유사도 계산은 식 (3)에서 스코어 함수 $x_k = 1$, 패싯 가중치 $f_k = \langle 0.85, 0.55, 0.30, 0.15, 0.15 \rangle$을 실험적으로 할당하여 계산한 결과이다. 질의 Q_2와 부품 copy 사이의 의미 유사도는 매체 패싯의 항목 'disk'와 'directory'를 동의어로 처리하고, 두 항목 사이의 거리 $\alpha = 0.7$을 할당하여 다음과 같이 계산한다.

【예 3】 질의 Q_2와 부품 copy 사이의 의미 유사도

$$C_{Q \cdot copy} = \frac{0.85*1.0 + 0.55*1.0 + 0.3*0.7 + 0.15*1.0 + 0.15*1.0}{0.85*1.0 + 0.55*1.0 + 0.3*1.0 + 0.15*1.0 + 0.15*1.0}$$

$$= 0.9550$$

4.3 실험 결과 및 검토

제안한 검색 모델의 검색 효율(재현율, 정확도)을 계산하기 위하여 20개의 질의를 작성하고, 325개의 일반 목적의 운영체제 명령어(MS-DOS, Linux)를 포함하는 라이브러리를 테스트 집합으로 선정하여 검색 실험을 2가지 방법으로 실시하였다. 첫 번째 실험은 20개의 질의를 작성하고 재현율을 각각 0.25, 0.50, 0.75로 설정하여 모든 질의를 평가한 후 검색된 소프트웨어 부품들을 대상으로 정확도의 평균을 계산한 결과 〈표 2〉와 같이 72%로 나타나 전통적인 검색 모델보다 개선되었음을 알 수 있었다.

〈표 2〉 평균 정확도
〈Table 2〉 average precision

재 현 율	0.25	0.50	0.75	평 균
정 확 도	0.78	0.71	0.67	0.72

두 번째 실험에서는 사용자가 작성한 20개의 질의와 325개의 일반적인 운영체제 명령어를 포함하는 라이브러리를 대상으로 계산한 검색 효율과 [10]에서 제안한 검색 모델의 검색 효율의 비교 결과가 〈표 3〉처럼 재현율이 93%, 정확도가 79%로 나타나 제안한 검색 모델의 검색 효율이 개선되었음을 알 수 있다. 한편, 라이브러리는 본 논문에서 제안한 개선된 패싯 분류 방식에 따라 부품들이 분류되고 비슷한 특성을 갖는 부품들이 인접한 장소에 저장되도록 구성되었다.

〈표 3〉 정확도와 재현율의 비교 결과
〈Table 3〉 comparative results of recall and precision

검색효율 / 검색모델	평균 재현율	평균정확도
제안한 검색모델	0.93	0.79
[10]의 검색모델	0.75	0.62

실험 결과에 의하면 제안한 검색 모델은 개선된 패싯 분류 방식에 따라 부품을 분류하고, 부품들 사이의 의미 유사도를 측정하여 비슷한 특성을 갖는 부품들을 인접한 장소에 저장하여 라이브러리를 구축하기 때문에 질의와 관련 있는 부품들의 의미 유사도 계산이 간단하고 신속하게 이루어진다. 따라서 제안한 검색 모델은 사용자가 요구하는 부품의 검색 시간이 단축되고 검색 효율이 높게 나타났으며, 사용자의 요구사항을 만족하는 다수의 유사한 부품들이 효율적으로 검색됨을 알 수 있다. 또한 질의와 유사한 특성을 갖는 다수의 부품을 검색하여 상관순위에 따라 부품을 출력하므로 사용자의 부품 선택이 쉽다.

한편, 제안한 모델의 검색 효율은 이용가능한 동의어 사전의 품질에 많은 영향을 받는다. 실제 실험에서 많은 단어의 의미와 항목들 사이의 의미 관계가 발견되지 않아 수작업으로 처리하였다.

5. 결론 및 연구방향

본 논문에서는 사용자의 요구사항을 만족하는 소프트웨어 부품을 효율적으로 검색할 수 있도록 지원하는 검색 모델을 제안하고, 실험을 통하여 성능 평가를 실시하였다. 부품을 분류하고 부품들 사이의 의미 유사도를 측정하여 비슷한 특성을 갖는 부품들을 서로 인접한 장소에 저장시켜 라이브러리를 구성하였다. 그리고 부품의 효율적인 검색을 위하여 정형화된 사용자의 요구사항으로부터 필요한 항목을 추출하여 벡터형의 질의를 구성하고, 유사한 부품들을 효율적으로 검색할 수 있도록 질의와 부품들 사이의 의미 유사도를 이용하였다.

질의를 작성하기 위하여 사용자는 질의 작성에 필요한 항목을 요구사항으로부터 각 패싯별로 필요한 항목을 추출할 수 있으므로 질의 작성이 간단하다. 질의 처리 과정에서는 질의와 부품들 사이의 의미 유사도를 이용하기 때문에 유사한 부품의 효율적인 검색이 가능했으며, 정확도와 재현율이 [10]의 검색 모델보다 높게 평가되었고, 부품의 특성함수와 유사도를 계산하는 [10]의 모델보다 검색 시간도 단축되어 전체적인 검색 효율이 개선되었음을 알 수 있었다.

제안한 검색 모델이 실무 영역에 적용하기 위해서는 부품들 간의 거리를 효율적으로 측정할 수 있는 수리 모델의 개발과 단어의 의미와 항목들 사이의 의미 관계를 정확하게 설명할 수 있는 고품질의 동의어 사전에 관한 연구가 추가로 진행되어야 한다.

참 고 문 헌

[1] D. Merkl, A.M. Tjoa, and G. Kappel, "Learning

the Semantic Similarity of Reusable Software Components," Third International Conference on Software Reuse, pp.33~41, November 1994.

[2] E.A. Karlsson, *Software Reuse: A Holistic Approach*, John Wiley & Sons Ltd., 1995.

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

[4] H. Mili, F. Mili, and A. Mili, "Reusing Software : Issues and Research Directions," IEEE Trans. on Software Engineering 21(16), pp.528~561, June 1995.

[5] J. Allen, *Natural Language Understanding*, 2dn ED., The Benjamin/Cummings Publishing Company, Inc., 1995.

[6] J.A. Kim, C.R. Moon and S.T. Kim, "Object-Oriented Retrieval Framework to Construct the Reuse Supporting Systems," The Transactions of the Korea Information Processing Society 2(5), pp.711~720, September 1995.

[7] M.R. Girardi and B. Ibrhim, "A Software Reuse System based on Natural Language Specification," Proceedings of 5th International Conference on Computing and Information (ICCI'93), pp. 507~511, May 1993.

[8] M.R. Girardi and B. Ibrhim, "Automatic Indexing of Software Artifacts," Third International Conference on Software Reuse, pp.24~32, November 1994.

[9] M.S. Kang, "Advanced Faceted Classification Scheme and Semantic Similarity Measure for Reuse of Software Components," The Transactions of the Korea Information Processing Society 3(4), pp.855~865, July 1996.

[10] M.S. Kang and B.K. Kim, "A Retrieval Model of Software Components using Similarity and Weights," Journal of the Korea Information Science Society 22(1), pp.69~78, January 1995.

[11] R.J. Leach and T.L. Fuller, "An Illustration of the Domain Analysis Process," ACM SIGSOFT Software Engineering Notes 20(5), pp.78~82, December 1995.

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

[13] R. Prieto-Diaz, "Implementing Faceted Classification for Software Reuse," Proceedings of 12th International Conference on Software Engineering, pp.300~304, March 1990.

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

[15] W.B. Frakes and T.P. Pole, "An Empirical study of Representation Methods for Reusable Software Components," IEEE Trans. on Software Engineering 20(8), pp.617~630, August 1994.

김 태 희

1991년 동신대학교 전자계산학과(공학사)
1993년 전남대학교 대학원 전산통계학과(이학석사)
1996년 전남대학교 대학원 전산통계학과(박사과정수료)
1993년 3월~현재 동신대학교 전자계산학과 시간강사
관심분야: 소프트웨어공학, 객체지향시스템

강 문 설

1986년 전남대학교 계산통계학과(이학사)
1989년 전남대학교 대학원 전산통계학과(이학석사)
1994년 전남대학교 대학원 전산통계학과(이학박사)
1983년 9월~1985년 12월 제1군수지원단 전산실
1989년 4월~1994년 8월 전남대학교 전산학과 조교 및 강사
1994년 9월~현재 광주대학교 전자계산학과 조교수
관심분야: 소프트웨어공학(재사용, 재공학, 역공학), 객체지향시스템, 정보검색