소프트웨어 부품의 재사용을 위한 개선된 패싯 분류 방법과 의미 유사도 측정

강 문 설†

요 약

본 논문에서는 재사용가능한 소프트웨어 부품의 분류 과정을 자동화하여, 소프트웨어 부품 라이브러리에 구조적으로 저장하는 방안을 제안한다. 효율적이고 자동화된 소프트웨어 부품의 분류를 위하여 자연어로 기술된 소프트웨어 부품 설명서로부터 의미 정보와 문장 구성 정보 등의 특징을 획득하여 소프트웨어 부품의 특성을 표현하는 패싯을 결정하고, 각각의 패싯에 해당하는 항목들을 자동으로 추출하여 소프트웨어 부품 식별자를 구성하였다. 그리고 분류된 소프트웨어 부품들 사이의 의미 유사도를 측정하여 비슷한 특성을 갖는 소프트웨어 부품들을 인접한 장소에 저장시켜 구조화된 소프트웨어 부품 라이브러리를 구축하였다. 제안한 방법은 소프트웨어 부품의 분류 과정이 간단하고, 유사한 소프트웨어 부품을 쉽게 식별할 수 있었으며, 또한 소프트웨어 부품을 라이브러리에 구조적으로 저장할 수 있다.

Advanced Faceted Classification Scheme and Semantic Similarity Measure for Reuse of Software Components

Moon Seol Kang†

ABSTRACT

In this paper, we propose a automation of the classification process for reusable software component and construction method of structured software components library. In order to efficient and automatic classification of software component, we decide the facets to represent characteristics of software component by acquiring semantic and syntactic information from software components descriptions in natural language, and compose the software component identifier or automatic extract terms corresponds to each facets. And then, in order to construct the structured software components library, we store in the near location with software components of similar characteristic according to semantic similarity of the classified software components. As the result of applying proposed method, we can easily identify similar software components, the classification process of software components become simple, and the software components store in the structured software components library.

1. 서 론

소프트웨어 재사용은 새로운 소프트웨어 시스템을 구축하기 위해 이미 존재하는 소프트웨어 부품을 재사용하기 위한 기술적 및 조직적인 배경 등과 매우 밀접한 관련이 있다. 소프트웨어 재사용은 이미 개발된 부품들을 새로운 소프트웨어를 개발하는 과정에서 재사용함으로써 개발 시간을 단축시키고, 동일한

† 종신회원:광주대학교 전자계산학과 전임강사
논문접수:1995년 10월 30일, 심사완료:1996년 3월 27일

부품이 다수의 다른 소프트웨어 개발 과정에서 테스트되어 재사용되기 때문에 소프트웨어 개발자의 생산성 향상과 소프트웨어의 품질 개선을 보장하고 있다. 따라서 소프트웨어 재사용을 위해서는 소프트웨어 개발자들이 재사용가능한 부품을 효율적으로 활용할 수 있도록 부품을 포함하는 소프트웨어 라이브러리가 잘 구축되어야 한다[3].

소프트웨어 부품을 포함하는 라이브러리의 가장 중요한 요구사항은 첫째, 소프트웨어 라이브러리가 다양한 응용 영역에 다수의 재사용가능한 부품을 제공해야 한다. 특히, 라이브러리에 포함된 부품을 현재 응용의 필요성에 따라 쉽게 적용가능하거나 재사용할 수 있어야 한다. 둘째, 소프트웨어 라이브러리는 실제 사용자들이 가장 적합한 부품을 쉽게 찾을 수 있도록 구축되어야 한다. 특히, 라이브러리는 어떤 특수한 기능을 만족하는 부품도 찾을 수 있도록 사용자에게 다양한 기능을 제공해야 한다[2, 5, 19].

소프트웨어 부품을 재사용하기 위해서는 기존에 개발된 재사용가능한 부품을 효율적으로 분류하여 라이브러리에 구조적으로 저장하는 방법, 사용자의 요구사항을 만족하는 부품을 신속, 정확하게 검색하는 방법, 사용자의 요구사항에 따라 검색한 부품을 이해하기 위한 부품의 표현 방법, 새로운 소프트웨어 시스템에 결합시키는 방법 및 이들 기법을 평가하기 위한 기술 등이 개발되어야 한다[4, 16, 21].

한편, 소프트웨어 부품의 분류 방법과 검색 방법은 소프트웨어 재사용을 실용화시키기 위한 핵심 요소들 중에서 가장 중요한 역할을 한다. 부품을 분류하는 방법에 따라 사용자의 요구사항과 일치하는 부품을 검색하는 방법, 부품을 포함하는 라이브러리의 확장성 등이 결정되기 때문에 소프트웨어 재사용에서는 부품을 분류하는 방법이 매우 중요하다. 지금까지 널리 알려진 소프트웨어 부품의 분류 방법으로는 열거형 분류(enumerative classification) 방법과 패싯 분류(faceted classification) 방법, 그리고 이 두 방법의 장점을 이용한 혼합형 분류(hybrid classification) 방법 및 기존 문서의 유사성을 계산하는 방법 등이 있다. 이 분류 방법들은 분류자에 의해서 수동적으로 분류되기 때문에 분류자의 주관적인 특성이 반영되어 동일한 부품에 대하여 분류자 마다 각각 서로 다른 특성을 갖는 부품으로 분류할 수 있다. 그리고 수동으로 분류한 부품은 라이브러리의 임의의 장소에 저장되어 라이브러리 구축 효율이 떨어질 뿐만 아니라 검색 과정에서 검색 시간이 많이 걸리고, 유사한 부품들의 검색을 어렵게 하는 등 전체적인 검색 효율이 저하된다. 따라서 부품을 분류하는 과정의 반자동화 및 자동화가 절실히 필요하다[2, 8, 10, 11, 12, 13].

본 논문에서는 소프트웨어 부품을 분류하기 위해서 널리 사용되고 있는 수동 분류 방법의 문제점을 분석하고, 그 해결책으로서 부품을 자동으로 분류하는 분류 방법을 제안한다. 이미 개발된 부품들의 설명서(Descriptions of Software Components)를 분석하여 부품이 수행하는 기본적인 기능(연산), 연산이 적용되는 대상(객체), 객체가 저장된 장소나 실제 연산이 수행되는 장소에 관한 정보(매체), 부품의 구현 및 실행 조건(환경) 등의 패싯을 결정하고, 각 패싯에 해당하는 항목들을 추출하여 부품의 식별자(Identifier of Software Components)를 생성한다. 그리고 소프트웨어 설명서를 이용하여 자동으로 분류된 부품들간의 의미 유사도를 측정하여 비슷한 특성을 갖는 부품을 인접한 장소에 저장시켜 구조화된 소프트웨어 라이브러리를 효율적으로 구축한다.

본 논문은 다음과 같이 구성한다. 2장에서는 지금까지 소프트웨어 부품의 재사용을 위해 사용된 열거형 분류 방법, 패싯 분류 방법, 혼합형 분류 방법 등의 수동 분류 방법과 자동 분류 방법에 대해서 설명하고, 3장에서는 부품의 분류를 자동화시키는 과정 및 분류된 부품을 구조적으로 라이브러리에 저장하는 방법을 설명한다. 4장에서는 제안한 분류 방법을 이용하여 부품을 분류하고 라이브러리에 저장하는 실험 및 실험 결과를 설명하고, 5장에서는 결론 및 향후 연구 방향을 기술한다.

2. 관련연구

이 장에서는 소프트웨어 부품을 수동 분류 절차에 의해 소프트웨어를 기반으로 목록화시키는 기법을 따르는 수동 분류 방법과 부품의 자연어 설명서에서 가능한 어휘, 구문, 의미 정보를 활용한 자동 분류 방법에 대해 간단히 살펴본다.

2.1 수동 분류 방법

소프트웨어 부품을 분류하기 위하여 널리 사용된 방법으로는 열거형 분류 방법과 패싯 분류 방법이 있다. 열거형 분류 방법은 예상되는 모든 소프트웨어 부품을 클래스로 분할하면서 계층적 관계로 정의한 후에 각 부품들을 계층 구조상의 클래스에 사용자가 직접 할당하는 방법이다. 이 분류 방법은 부품들간의 상호관계를 잘 나타내기 때문에 다른 분류 방법에 비해 원하는 부품을 신속하게 찾을 수 있다는 장점이 있지만, 분류된 부품들과 특성을 달리하는 새로운 부품을 추구할 경우 분류 계층을 다시 구성해야 하므로 확장이 어렵다[1].

패싯 분류 방법은 열거형분류 방법의 단점을 개선하기 위해서 부품들이 갖는 공통적인 측면의 특성을 합성하여 하나의 패싯으로 표현하며, 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[17, 18].

이밖에도 열거형 분류 방법과 패싯 분류 방법의 특징을 결합시킨 혼합형 분류 방법과 부품이 실행하는 기본 기능을 위한 프레임(frame)과 부품에 의해 조작되는 객체를 위한 슬롯(slot)으로 목록을 구성하는 소프트웨어 부품의 의미를 획득하기 위한 목적으로 사용된 프레임 기반 부품의 목록화 방법이 있으며, NLH/E(Natural Language Help/English)는 영어로 질의 작성을 채택한 질의 응답 시스템으로 case 문법을 사용하여 부품을 위한 프레임 기반 부품의 목록화 방법이다. 목록은 수동적으로 작성되고, 작성된 결의는 case 문법을 사용하여 분석한다. Proteus는 소프트웨어 부품을 분류하기 위하여 4가지의 다른 분류 방법, 즉, 키워드 기반(keyword-based), 패싯(faceted), 열거 그래프(enumerated graph) 및 속성값(attribute value)을 이용한 분류 방법의 실험적 평가를 지원하는 소프트웨어 재사용 시스템이다[10, 11, 12, 14, 20, 22].

위의 수동 분류 방법은 소프트웨어 부품을 분류하기 위한 분류 방법에 대한 많은 정보를 사용자(분류자)가 알고 있어야 하며, 부품 라이브러리의 크기가 커지면 검색 시간이 길어지고 라이브러리의 관리가 어려워진다.

2.2 자동 분류 방법

소프트웨어 부품의 분류 과정을 자동화시키려는 분류 방법은 2개의 범주로 분류할 수 있다. 어휘 분석의 수준만을 포함하는 분류 방법과 소프트웨어 설명서의 구문 분석과 의미 분석을 포함하는 분류 방법으로 나눌 수 있다[9].

어휘 처리(lexical processing)를 이용한 대표적인 예제는 GURU 시스템으로, 어휘 친화력(lexical affinity)의 개념과 정보의 양에 기초한 분류 방법을 사용하여 자연어 문서로부터 자동으로 추출된 속성에 따라 부품을 분류한다. 검색 시스템은 자유스런 유형의 자연이 질의를 채택하고, 부품을 분류하기 위하여 사용된 방법과 동일한 방법을 질의 작성에 적용한다. 설명서에 대한 이해 및 어떤 종류의 구문과 의미 지식을 사용하지 않더라도 시스템은 단일 키워드 기반 시스템보다 좋은 정확도를 유지한다[23].

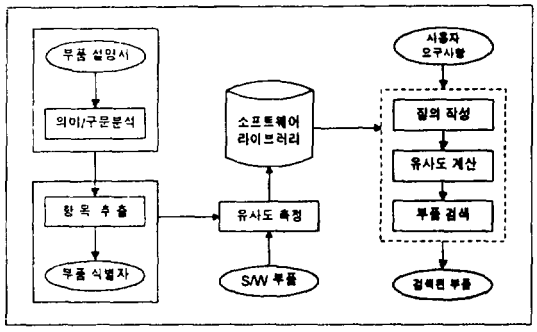
구문 및 의미 처리(syntactic and semantic processing)를 이용한 분류 방법은 소프트웨어 설명서의 완전한 이해와는 상관없이 자연어 명세서의 구문 분석과 의미 분석을 할 수 있도록 지원한다. 이 방법은 응용 영역과 자연어에 대한 의미 정보를 저장하는 지식베이스에 기초하기 때문에 전통적인 검색 시스템보다 강력하지만 일반적인 지식베이스가 각 응용 영역을 수동적으로 생성되기 때문에 많은 자원을 요구한다. 대표적인 예로는 LaSSIE(Large Software System Information Environment)로 식베이스, 형식 추론을 기반으로 한 의미 분석 알고리즘, 그래픽 브라우저 및 자연어 분석을 통합하는 사용자 인터페이스로 구성되었다. 대형 소프트웨어 시스템에서 유용한 정보를 탐색하는 과정에서 프로그래머에게 도움을 제공하고, 재사용을 위한 부품의 검색을 지원한다. 현재 지식베이스의 구축은 수작업으로 처리하고 있으며, 향후 구축 과정의 자동화를 위한 지식 습득 메커니즘에 대한 연구가 진행되고 있다[7, 15].

3. 소프트웨어 부품의 분류 방법과 의미 유사

도 측정

이 장에서는 소프트웨어 부품을 분류하는 과정에서 각 패싯에 해당하는 항목을 자동으로 추출하여 부품 식별자를 구성하고, 분류된 부품들 사이의 의미 유사도를 측정하여 유사한 부품들을 인접한 위치에 저장시켜 라이브러리를 구축하는 방법을 설명한다.

본 논문에서는 (그림 1)과 같이 먼저 분류 대상 부품들의 설명서를 분석하여 부품의 특성을 표현할 수 있는 속성을 추출하여 필요한 패싯(facets)을 결정한다. 이렇게 결정된 각 패싯에 필요한 항목을 자동으로 추출하여 부품 식별자를 구성하고, 분류된 부품들의 유사성에 따라 비슷한 특성을 포함하는 부품들을 인접한 위치에 저장하는 구조화된 라이브러리 구축 방안을 제안한다.



(그림 1) 소프트웨어 부품의 분류 및 라이브러리 구축 과정
(Fig. 1) Classification and library construction process of software components

3.1 패싯 결정을 위한 설명서 분석

사용자가 원하는 소프트웨어 부품을 신속하게 검색할 수 있도록 부품을 분류하기 위한 분류 방법을 결정할 때 분류 대상이 되는 객체의 성질과 분류 목적이 무엇인가를 고려해야 한다. 즉, 원하는 부품의 검색이 용이하고, 새로운 특성을 갖는 부품이 추가될 때 확장이 용이하며, 원하는 부품을 찾지 못했을 경우 유사한 부품을 검색할 수 있도록 지원하는 분류 방법을 선택해야 한다.

본 논문에서는 자연어로 기술된 부품의 설명서를 자연어 처리의 구문분석 기법[6]을 적용하여 대표적인 부품들을 분석하여 획득한 정보들로부터 부품들이 가지고 있는 공통의 특성을 표현하는 패싯을 결정하고, 결정된 각 패싯에 해당하는 항목들을 설명서로부터 자동 추출하여 부품 식별자를 구성한다.

〈정의 1〉 패싯(*facets*)

소프트웨어 부품들이 갖는 공통적인 측면의 특성을 합성하여 항목(*terms*)이라는 기본적인 클래스들로 표현할 때, 이 기본적인 클래스를 패싯(*facets*)이라 한다. 하나의 패싯은 소프트웨어 부품을 표현하는 같은 종류의 특성을 갖는 모든 항목들을 포함한다.

패싯은 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. 〈표 1〉은 본 논문에서 실험 대상으로 선정한 MS-DOS 명령어의 설명서에서 핵심 문장을 분석하여 정의한 패싯의 기본 집합 일부를 나타내었다.

〈표 1〉 부품을 설명하기 위한 패싯의 기본 집합
〈Table 1〉 Basic set of facets for describing software components

패싯(facets)	특 성
기능(function)	부품이 수행하는 기본 기능
객체(objects)	기능 수행을 위해 적용되는 대상
매체(medium)	객체가 저장되거나 실제 기능이 수행되는 장소
언어(language)	부품의 원시코드를 작성한 프로그래밍 언어
운영체제(OS)	부품의 개발 및 실행을 위한 시스템의 운영체제
조건(condition)	기능을 실행시키기 위한 전제 조건

〈정의 2〉 패싯 속성(*facets attribute*)

소프트웨어 부품 설명서의 핵심 문장에서 특정 패싯의 존재를 표현하는 요소들을 패싯 속성(*facts attribute*)으로 정의하며, 주로 전치사가 패싯 속성에 해당

한다.

패싯 속성은 부품 설명서의 문장 표현으로부터 패싯을 결정할 수 있는 성질을 지닌 언어(주로 전치사)들을 의미하고, 이 패싯 속성에 관련되는 단어들을 이용하여 패싯 값을 추출할 수 있다. 〈표 2〉는 MS-DOS 명령어의 설명서를 분석하여 식별한 패싯 속성들의 예를 나타내고 있다. 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 2〉 패싯을 추출하기 위한 속성
〈Table 2〉 Attributes for extract facets

패싯(facets)	패 싯 속 성
기능(function)	필수 문장에 있는 핵심 동사
객체(objects)	'by using', with, using
매체(medium)	at, in into, within, through
언어(language)	C, C++. PASCAL, ASSEMBLER
운영체제(OS)	MS-DOS, UNIX, Windows95
조건(condition)	as, at, except, how, unless, until

〈정의 3〉 패싯 결정을 위한 제약 조건 및 경험 (*constratints 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.

위의 정의 1, 2, 3을 이용하여 부품들이 갖는 공통적인 측면의 특성을 표현하는 패싯을 결정하는 알고리즘을 다음과 같이 정의한다.

〈패싯 결정 절차〉

단계 1. 기본적인 클래스를 결정하기 위하여 대표적인 소프트웨어 부품을 설명서를 분석한다.

단계 2. 결정된 클래스를 이용하여 부품을 표현하기 위한 패싯들을 결정한다.

단계 3. 부품의 특성을 표현할 수 있도록 결정된 패싯별로 속성을 추출한다.

단계 4. 기본적인 클래스들을 논리적인 분할의 단일 특성 원칙과 추출된 속성을 이용하여 패싯으로 통합한다.

단계 5. 패싯들을 합하여 부품 식별자를 구성할 수 있도록 패싯들을 위한 조합 순서를 결정한다.

(그림 2) 패싯을 결정하는 알고리즘
(Fig. 2) Facets decision algorithms

소프트웨어 부품을 효율적으로 분류하고, 사용자들에게 필요한 부품의 표현을 쉽게하기 위하여 부품이 수행하는 기본적인 기능, 부품의 구현 및 부품을 실행시키는데 필요한 환경등에 관한 정보를 패싯으로 선택한다. 또한 사용자의 요구사항에 따라 질의 작성과 부품의 검색을 쉽게하고, 요구사항과 일치하는 부품이 없을 경우에도 다수의 유사한 부품의 검색을 지원할 수 있도록 기능, 객체, 매체, 언어, 및 운영체제 등의 패싯으로 부품을 표현한다.

3.2 소프트웨어 부품의 식별자 구성

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 4〉 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.

〈정의 5〉 패싯 가중치(f_j : *facets weight*)

각각의 패싯들이 소프트웨어 부품의 특성을 어느 정도 설명하고, 표현할 수 있는 가에 따라 각 패싯에 가중치를 할당하여, 이 가중치를 패싯 가중치(facets weight)라 정의하고 f_j로 표현한다.

패싯 가중치는 부품을 라이브러리에 저장하는 과정에서 비슷한 부품들을 식별하기 위해서 유사도를 계산하는데 이용한다. 따라서 이 패싯 가중치는 사용자들의 요구사항에 따라 결정되는데, 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 먼저 고려한다고 가정하고, 본 논문에서는 실험적으로 패싯 가중치 λ_j를 결정한다.

3.3 라이브러리 구축을 위한 부품의 유사도 측정

3.3.1 유사도 측정 방법

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.

따라서, 본 논문에서는 제안한 분류 방법에 따라 분류되어 라이브러리에 저장되는 부품들의 유사도를 〈정의 6〉과 식 (3)을 이용하여 측정하고, 의미적으로 기능이 유사한 부품들을 서로 가까운 위치에 저장시켜 소프트웨어 부품 라이브러리를 구조적으로 구축한다. 라이브러리에 저장되는 부품들 사이에서 두 부품간의 떨어진 정도(거리 : distance)나 서로간의 의미적 유사성(proximity)의 크기를 측정하기 위하여 부품들 사이의 유사도(similarity)를 다음과 같이 정의한다.

〈정의 6〉 부품간의 유사도(S_{ij} : *similarity between components*)

라이브러리에 저장될 임의의 부품들 사이의 유사성 또는 관련성을 측정하는 측도, 즉 임의의 부품 i와 부품 j 사이의 함수 $S_{ij} = f(C_{ij}, C_j)$를 유사도로 정의하며, 다음의 조건 (1), (2), (3)을 만족해야 한다.

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

한편, 라이브러리에 저장되는 소프트웨어 부품들은 n개 패싯의 항목을 합성하여 부품 식별자를 구성하기 때문에 유사도는 k번째 패싯에 대하여 적절한 스코어(score)를 배정하는 방법으로 정의할 수 있다. 즉 패싯별 스코어 함수 $r_k(t_{ik}, t_{jk})$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($0 < x_k \leq 1$), s_i는 항목 t_{ik}와의 동의어, α는 항목 t_{ik}와 동의어 s_i 사이의 거리($0 < \alpha < 1.0$)를 의미한다. 따라서 라이브러리에 저장되는 임의의 부품 i와 부품 j 사이의 유사도는 다음과 같이 정의된다.

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

그리고 각 패싯에 대하여 사전의 중요성이나 신뢰성을 반영하기 위하여 패싯 가중치 f_k를 사용 하였으므로 패싯의 가중 함수 $f_k(t_{ik}, t_{jk})$를 k번째 성분에 대응시키면, 임의의 부품 i와 부품 j 사이의 유사도 S_{ij}는 다음과 같다.

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

3.3.2 부품들 사이의 유사도 측정

4개의 소프트웨어 부품에 대한 부품 식별자가 다음과 같이 구성되었고, 스코어 함수 $x_k=1$, 패싯 가중치 $f_k=\langle 0.85, 0.55, 0.30, 0.15, 0.15\rangle$를 실험적으로 할당하고, 식 (3)을 적용하여 유사도를 계산하면 다음과 같다.

$C_{dir}=\langle$ display, file, directory, C, MS-DOS $\rangle$,
$C_{tree}=\langle$ display, structure, directory, C, MS-DOS $\rangle$
$C_{format}=\langle$ create, directory, disk, C, MS-DOS $\rangle$
$C_{unformat}=\langle$ restore, file, disk, C, MS-DOS $\rangle$

$$S_{dir \cdot tree}=\frac{0.85*1.0+0.55*0+0.3*1.0+0.15*1.0+0.15*1.0}{0.85*1.0+0.55*1.0+0.3*1.0+0.15*1.0+0.15*1.0}=0.7250$$

$$S_{for \cdot unfor}=\frac{0.85*0.9+0.55*0.7+0.30*1.0+0.15*1.0+0.15*1.0}{0.85*1.0+0.55*1.0+0.30*1.0+0.15*1.0+0.15*1.0}=0.8750$$

소프트웨어 부품을 제안한 분류 방법에 따라 분류한 후 부품들 사이의 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 검색할 수 있도록 지원한다.

〈부품의 분류 및 라이브러리 구축 절차〉

단계 1. 각 패싯의 패싯 속성을 이용하여 각각의 패싯에 해당하는 항목을 소프트웨어 설명서로부터 추출한다.

단계 2. 패싯의 조합 순서에 따라 추출된 항목들을 이용하여 부품 식별자를 구성한다.

단계 3. 부품 식별자를 구성하는 항목들을 이용하여 부품들의 유사도를 계산한다.

단계 4. 계산된 부품들의 유사도에 따라 유사한 부품들을 가까운 위치에 저장하여 라이브러리를 구축한다.

(그림 3) 부품 분류 및 라이브러리 구축 알고리즘
(Fig. 3) Components classification and library construction algorithms

4. 부품의 분류 및 라이브러리 구축 실험

소프트웨어 부품의 분류 과정을 자동화시키기 위하여 제안된 분류 방법과 라이브러리 구축 방법의 타당성 평가를 위하여 실험을 실시한다. 실험에서는 분류 대상 부품의 정확한 분류 및 수동분류 방법의 결과와 비교, 분류된 부품들간의 유사도 측정을 실험한다.

4.1 실험 환경 및 방법

소프트웨어 부품을 효율적으로 분류하기 위해 제안한 분류 방법과 라이브러리 구축 방법의 타당성을 평가하기 위하여 펜티엄(IBM-PC)의 MS-DOS 환경에서 부품의 분류 실험을 실시하였다. 본 논문에서는 재사용가능한 소프트웨어 부품의 예제로서 운영체제(MS-DOS) 명령어를 선택한다. 운영체제 명령어는 여러가지 이유에서 소프트웨어 부품을 분류하기 위한 분류 방법의 효율성 평가에 적합하고, 부품의 설명서는 다양한 명령어의 설명(메뉴얼)에서 쉽게 획득이 가능하며, 운영 체제 메뉴얼을 소프트웨어 문서화의 실세계 예제로 간주할 수 있다. 그리고 운영체제의 다양한 명령어들의 기능 또는 행위는 잘 알려져 있기 때문에 분류 방법의 평가와 소프트웨어 라이브러리의 구조적인 구축이 용이하다.

약 100개의 운영체제 명령어를 대상으로 운영체제 메뉴얼의 설명서를 분석하여 부품의 기본적인 특성을 결정하는 패싯을 선정하고, 각 패싯에 해당되는 항목을 〈표 2〉의 패싯 속성을 이용하여 자동으로 추출하여 부품 식별자를 구성한다. 그리고 유사한 부품을 식별하기 위하여 부품 식별자를 이용하여 유사도를 측정하고 구조화된 라이브러리를 구축한다. 이 과정에서 부품 분류의 정확성, 유사한 부품의 식별 및 구축된 라이브러리의 구조성 등을 분석한다.

4.2 부품 분류 및 유사도 측정 실험

일반적으로 소프트웨어 개발과정에서 재사용가능

한 부품을 검색하여 재사용하기 위해서는 사용자의 요구사항을 어느 정도 정확하게 표현할 수 있는가에 따라 부품의 검색 효율이 결정된다. 사용자가 요구하는 부품을 신속하고 정확하게 검색하기 위해서는 재사용가능한 부품이 수행하는 기능과 기능이 수행되는 대상들이 커다란 영향을 미치게 된다.

따라서, 본 논문에서는 부품을 효율적으로 분류하고, 사용자들에게 필요한 부품의 표현을 쉽게하기 위하여 부품이 수행하는 기본적인 기능, 부품의 구현 및 실행시키는데 필요한 환경 등에 관한 정보를 패싯으로 선택한다. 즉, 〈표 1〉과 같이 부품이 수행하는 기본 기능, 기능을 수행하기 위해 적용되는 대상으로 나타내는 객체, 객체가 저장되거나 실제 기능이 수행되는 장소를 나타내는 매체, 부품의 원시코드를 작성한 프로그래밍 언어, 부품의 개발 및 실행을 위한 시스템의 운영체제를 패싯으로 선택한다. 그리고 적용 분야나 소프트웨어 개발자들의 요구에 따라 필요한 패싯을 추가 또는 삭제시킬 수 있다. 이렇게 패싯이 결정되면 제안한 방법에 따라 다음과 같은 순서로 부품을 분류하여 라이브러리에 저장한다.

첫째, 소프트웨어 부품 설명서로부터 부품의 기본적인 특성을 표현하는 기본적인 클래스를 식별하여, 각 패싯에 해당하는 항목을 자동으로 추출한다. 예를 들면, 〈그림 4〉의 운영체제 명령어 'DIR'의 부품 설명서로부터 〈표 2〉의 패싯 속성을 이용하여, 각 패싯에 해당하는 항목을 자동으로 추출한 결과는 〈표 3〉과 같다. 기능 패싯은 '필수 문장에 있는 핵심 동사'를 추출하므로 항목 '*display*'를 추출하며, 객체 패싯은 전치사 '*of*'와 관련되는 문장 '*a list of the files*‥'

> Displays a list of the files and subdirectories that are in the directory you specify. When you use DIR without parameters or switches, it displays the disk's volume label and serial number, one directory or filename per line, including the filename extension, the file size in bytes, and the date and time the file was last modified; and the total number of files listed, their cumulative size, and the free space remaining on the disk. . . . . .
> (중략) . . . . .

〈그림 4〉 'DIR'의 부품 설명서
(Fig. 4) Component description of 'DIR'

에서 항목 '*file*', 그리고 매체 패싯은 전치사 '*in*'과 관련되는 문장 '*in the directory*'에서 항목 '*directory*'를 추출한다.

〈표 3〉 부품 설명서로부터 추출한 패싯별 항목
〈Table 3〉 facets' terms extracted from component description

패싯	기 능	객 체	매 체	언 어	운영체제
항목	display	file	directory	C	MS-DOS

둘째, 소프트웨어 부품 설명서로부터 자동으로 추출한 항목들을 〈정의 4〉를 이용하여 부품 식별자로 합성한다. 즉, 기능 패싯의 항목은 '*disply*', 객체 패싯의 항목은 '*file*', 매체 패싯의 항목은 '*directory*', 언어 패싯의 항목은 '*C*', 그리고 운영체제 패싯의 항목으로 '*MS-DOS*'를 할당하여 부품 '*DIR*'의 식별자 C_{dir} = 〈*display, file, directory, C, MS-DOS*〉를 구성한다. 한편, 본 연구에서는 부품 설명서에 언어 및 운영체제 정보가 표현되지 않은 경우에는 C 언어와 MS-DOS로 결정한다. 분류 대상의 다른 모든 부품들에 대해서도 동일하게 패싯별로 항목을 추출하여 부품 식별자를 구성한다.

셋째, 부품의 분류가 끝나면 분류된 부품들간의 유사성을 식별하기 위하여 〈정의 6〉과 식 (3)을 이용하여 부품들간의 유사도를 측정한다. 〈표 4〉는 분류된

〈표 4〉 부품들간의 유사도 측정 결과
〈Table 4〉 Result of similarity measure between components

부품이름	패 싯 이 름					유사도
	기능	객체	매체	언어	운영체제	
dir	display	file	directory	C	MS-DOS	
tree	display	structure	directory	C	MS-DOS	0.7250
copy	copy	file	directory	C	MS-DOS	
xcopy	copy	directory	directory	C	MS-DOS	0.9175
format	create	directory	disk	C	MS-DOS	
unformat	restore	file	disk	C	MS-DOS	0.8750
comp	compare	contents	file	C	MS-DOS	
diskcomp	compare	contensts	disk	C	MS-DOS	0.9250
fc	compare	file	directory	C	MS-DOS	0.7625

부품들간의 유사도 측정 결과의 일부이며, 유사도는 〈정의 6〉의 식 (3)에서 스코어 함수 $x_k=1$, 패싯 가중치 $f_{k,1}=\langle 0.85, 0.55, 0.30, 0.15, 0.15\rangle$을 실험적으로 할당하여 계산한 결과이다. 부품 '*dir*'과 '*tree*'의 유사도는 객체 패셋의 항목 '*file*'과 '*structure*'가 동의어로 처리할 수 없으므로, 스코어 함수를 계산할 때, 두 항목사이의 거리 $\alpha=0.0$을 할당하여 계산하였으며, 부품 '*copy*'와 부품 '*xcopy*'의 유사도 계산에서 객체 패싯의 항목 '*file*'과 '*directory*'를 동의어로 처리하고, 두 항목사이의 거리 $\alpha=0.7$을 할당하여 계산하였으며, 부품 '*format*'과 '*unformat*'의 유사도 계산에서도 기능 패싯의 항목 '*create*'와 '*restore*'를 동의어로 처리하고, 거리 $\alpha=0.9$를 할당하여 유사도를 계산한다.

넷째, 분류된 소프트웨어 부품들의 유사도 측정 결과에 따라 비슷한 특성을 갖는 부품들을 인접한 위치에 저장하여 구조화된 소프트웨어 라이브러리를 구축한다. 〈표 4〉에 나타난 유사도에 따라 부품 '*dir*'과 '*tree*', '*copy*'와 '*xcopy*', '*format*'와 '*unformat*', 그리고 '*comp*', '*diskcomp*' 및 '*fc*'를 비슷한 특성을 갖는 부품들로 식별하여 서로 인접한 위치에 저장한다.

4.3 실험 결과 및 검토

실험 대상으로 선정한 100개의 재사용가능한 부품을 대상으로 분류 실험을 실시한 결과 87%의 정확성, 즉 87개의 부품에 대해서는 각 패싯에 해당하는 항목들을 정확하게 자동으로 추출하여 부품 식별자를 구성하고 부품들 사이의 의미 유사도를 측정하여 비슷한 특성을 갖는 부품들을 서로 인접한 위치에 저장하여 라이브러리를 구조적으로 구성하였다. 한편, 소프트웨어 부품 설명서에 사용된 단어들이 어휘 사전에 없는 일반 단어이거나 컴퓨터 영역에서만 사용하는 일반 단어 및 복합 수식어들로 구성되었기 때문에 13개의 부품에 대해서는 각 패싯에 해당하는 항목들을 정확하게 추출하지 못하고, 만들어진 부품 식별자를 수정하였다. 따라서 부품 설명서의 효율적인 문장 구성과 동의어 사전의 구축이 요구된다.

소프트웨어 부품을 분류하기 위해 제안한 분류 방법과 라이브러리 구축 방법은 재사용가능한 부품들의 일관성있는 분류와 분류 과정의 자동화가 가능했으며, 분류된 부품들의 유사도를 측정하여 비슷한 부품들의 식별이 가능하였다. 또한 구조화된 라이브러리의 구축을 위해 분류된 부품들 사이의 의미 유사도를 측정하여 비슷한 특성을 갖는 부품들을 인접한 위치에 저장할 수 있었다. 따라서 기존의 부품 분류 과정에서 발생할 수 있는 일관성 없는 부품의 분류와 비효율적인 라이브러리 구축 문제를 해결할 수 있으며, 분류 방법의 확장, 라이브러리의 효율적인 관리 및 확장이 가능하였다.

4.4 기존 연구와 비교

소프트웨어 부품을 위한 분류 방법은 사용자가 관심을 가지고 있는 부품의 신속한 검색과 유사한 부품을 검색할 수 있도록 분류되어 라이브러리가 구성되었는가를 평가한다. 첫째, 패싯 분류 방법[8, 17, 18]과 혼합형 분류 방법[10, 12]에서는 각 패싯에 해당하는 항목들을 수동으로 추출하여 부품 식별자를 구성하기 때문에 동일한 부품에 대해 분류자 마다 서로 다른 부품 식별자를 구성할 수 있다. 본 논문에서는 각 패싯에 해당하는 항목을 부품 설명서로부터 자동으로 추출하여 부품 식별자를 구성하기 때문에 부품 분류의 일관성을 유지하고 항목 추출 과정을 자동화하였다.

둘째, 사용자가 요구하는 부품을 정확하게 표현하지 못하거나 부품이 라이브러리에 존재하지 않은 경우에 유사한 부품의 식별 문제는 매우 중요하다. 패싯 분류 방법[8, 17, 18]에서는 유사한 부품을 식별할 수 있는 기능을 제공하지 않으며, 혼합형 분류 방법[8, 12]에서는 항목 가중치와 패싯 가중치를 부여하여 유사한 부품을 식별할 수 있는 기능을 제공하고 있다. 본 논문에서도 분류되어 라이브러리에 저장되는 부품들 사이의 의미 유사도를 이용하여 유사한 부품을 식별할 수 있도록 하였다.

셋째, 분류 과정의 마지막 단계인 라이브러리 구성 방법은 검색 효율에 커다란 영향을 미치게 된다. 기존 연구에서는 부품을 응용 영역별로 집단화(clustering)하여 라이브러리에 저장하는 방법[8, 10, 12]과 부품이 분류되는 순서대로 라이브러리에 저장하는 방법[17, 18]을 채택하였다. 본 논문에서는 부품들 사이의 의미 유사도를 측정하여 기능적으로 유사한 특성을 갖는 부품들을 서로 인접한 거리에 저장하여 라이브러리를 구성하기 때문에 검색 시간을 단축시키고 유사한

부품을 신속히 검색할 수 있다.

5. 결 론

본 논문에서는 소프트웨어 부품을 분류하기 위하여 자연어로 기술된 부품 설명으로부터 의미 및 문장 구성 정보를 획득하여 부품의 특성을 표현하는 패싯을 결정하고, 각각의 패싯에 해당하는 항목들을 자동으로 추출하여 부품 식별자를 구성한다. 그리고 분류된 부품들 사이의 의미 유사도 측정 결과에 따라 비슷한 특성을 갖는 부품을 인접한 장소에 저장하여 구조화된 소프트웨어 라이브러리를 구축하였다.

소프트웨어 부품 설명서로부터 항목을 자동으로 추출하여 부품 식별자의 구성에 있어서는 87%의 정확성을 얻었으며, 나머지 13%는 추출된 항목들을 수정하여 부품 식별자를 재구성하였다. 그리고 구성된 부품 식별자를 이용한 의미 유사도의 계산은 비교적 정확하게 계산되었으나 부품 '*format*'과 부품 '*unformat*'은 기능적으로 반대의 기능이라 할 수 있는데, 유사한 기능을 포함하고 있는 부품 '*dir*'과 부품 '*tree*'보다 유사도가 높게 계산되었다. 이는 부품 설명서로부터 항목을 자동으로 추출하여 부품 식별자를 구성하기 때문에 나타나는 문제점이다.

소프트웨어 부품을 분류하기 위한 분류 방법과 의미 유사도 측정 방법의 특징은 다음과 같다. 첫째, 동일한 부품에 대하여 분류자 마다 각각 서로 다른 특성을 갖는 부품으로 분류할 수 있는 수동 분류 방법의 문제점을 해결하여 부품 분류의 자동화와 일관성을 제공한다. 둘째, 패싯을 이용하여 부품의 공통적인 측면의 특성을 표현하기 때문에 부품의 분류가 간단하고, 유사한 부품의 식별이 용이하며, 또한 분류 방법의 확장이 용이하다. 셋째, 분류된 부품들의 유사도 측정 결과에 따라 유사한 특성을 갖는 부품들을 서로 인접한 거리에 저장하여 라이브러리를 구축하였기 때문에 검색 시간의 단축과 유사한 부품들의 검색이 가능하다.

그리고 소프트웨어 개발과정에서 빈번하게 재사용 가능하고, 개발자들이 일반적인 응용 분야에 적용할 수 있는 소프트웨어 부품을 대상으로 한 실험, 효율적인 동의어 사전 및 검색 모델의 개발, 그리고 개발한 부품의 분류 방법 및 검색 모델을 기반으로 한 재사용 도구에 관한 연구를 진행하고 있다.

참 고 문 헌

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

[2] C. McClure, *The Three Rs of Software Automation: Reengineering, Repository, Reusability*, Prentice-Hall, 1993.

[3] D. Merkl, A.M. Tjoa, and G. Kappel, "Learning the Semantic Similarity of Reusable Software Components," 3rd International Conference on Software Reuse, pp.33~41, November 1994.

[4] E. Ostertag, J. Hendler, R. Prieto-Diaz, and C. Braun, "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.

[5] H. Mili, F. Mili, A. Mili, "Reusing Software: Issues and Research Directions" IEEE Trans. on Software Engineering 21(6) pp. 528~561, June 1995.

[6] J. Allen, *Natural Language Understanding*, 2nd Ed., The Benjamin/Cummings Publishing Company, Inc., 1995.

[7] J. Nie, F. Paradis and J. Vaucher, "Using Information Retrieval for Software Reuse," Proceeding of 5th International Conference on Computing and Information(ICCI'93), pp.448~452, May 1993.

[8] K.W. Lee., "CARS 1.0:Software Reuse System to Support Object-Oriented Technology," Jongang University, Seoul, 1991.

[9] M.R. Girardi and B.Ibrahim, "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.

[10] M.S. Chang and Y.R. Gwon., "A Classification

Scheme and Retrieval Method of Software Components for Reuse," Journal of the Korea Information Science Society 19(6), pp.602~613, November 1992.

[11] M.S. Kang and B.K. Kim, "An Extended Faceted Classification Scheme and Hybrid Retrieval Model to support Software Reuse," The Transactions of the Korea Information Processing Society 1(1), pp.23~37, May 1994.

[12] M.S. Kang and B.K. Kim., "Retrieval and Understanding of Reusable Software Components," Journal of the Korea Information Science Society 20(10), pp.1519~1529, October 1993.

[13] 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.

[14] M. Wood and I. Sommerville, "An Information Retrieval System for Software Components," ACM SIGIR Fourm 22(3/4), pp.11~28, 1988.

[15] P. Devanbu, R. Brachman, P. Selfridge and B. Ballard, "LaSSIE:A Knowledge-base Software Information System," CACM 34(5), pp.34~49, May 1991.

[16] R.D, Banker, R.J. Kauffman, and D. Zweig, "Repository Evaluation of Software Reuse," IEEE Trans. on Software Engineering 20(8), pp. 163~172, July 1994.

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

[18] R. Prieto-Diaz, "Implementing Faceted Classification for Software Reuse," Proceedings of the 2th International Conference on Software Engineering, pp.300~304, March 1990.

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

[20] 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.

[21] W.B. Frakes and C.J. Fox, "Sixteen Questions about Software Reuse," CACM 38(6), pp.75~87, June 1995.

[22] W. Tichy, R. Adams and L. Holter, "NLH/E:A Natural Language Help System," Proceedings of 11th International Conference Software Engineering, pp.364~374, May 1989.

[23] Y.S. Marrek, D.N. Berry, and G.E. Kaiser, "An Information Retrieval Approach for Automatically Constructing Software Library," IEEE Trans. on Software Engineering 17(8) pp.800 ~813, August 1991.

강 문 설

1986년 전남대학교 계산통계학과(이학사)

1989년 전남대학교 대학원 전산통계학과(이학석사)

1994년 전남대학교 대학원 전산통계학과(이학박사)

1983년 9월~1985년 12월 제1군 수지원단 전산실

1989년 4월~1994년 8월 전남대학교 전산학과 조교 및 강사

1994년 9월~현재 광주대학교 전자계산학과 전임강사

관심분야:소프트웨어공학(재사용, 재공학, 역공학), 객체지향시스템, 정보검색