컴포넌트 분류를 위한 복합 클러스터 분석 방법

이 성 구†

요 약

컴포넌트 재사용을 위해 다양한 분류 방법들이 개발되어 왔다. 이러한 분류 방법들은 사용자가 필요로 하는 컴포넌트들을 쉽고 빠르게 접근하는 것을 돕는다. 전통적인 분류 방법들은 분류 구조 생성을 위한 도메인 분석 노력, 컴포넌트 사이의 관계 표현, 도메인 진화에 따른 분류 구조 유지 보수의 어려움, 그리고 한정된 도메인 적용과 같은 문제들을 포함한다.

본 논문은 이러한 문제들을 언급하기 위해 복합 클러스터 분석 기반의 컴포넌트 분류 방법에 대해 묘사한다. 안정적인 분류 구조 자동 생성을 위해 계층 클러스터 분석 방법과 새로운 컴포넌트의 자동 분류에 대해 비계층 클러스터 분석 개념은 결합된다. 제안된 방법에 의해 생성된 클러스터 정보는 관련 컴포넌트들에 대한 도메인 분석 과정을 지원할 수 있다.

키워드 : 분류, 클러스터, 컴포넌트

A Composite Cluster Analysis Approach for Component Classification

Sung-Koo Lee†

ABSTRACT

Various classification methods have been developed to reuse components. These classification methods enable the user to access the needed components quickly and easily. Conventional classification approaches include the following problems: a labor-intensive domain analysis effort to build a classification structure, the representation of the inter-component relationships, difficult to maintain as the domain evolves, and applied to a limited domain.

In order to solve these problems, this paper describes a composite cluster analysis approach for component classification. The cluster analysis approach is a combination of a hierarchical cluster analysis method, which generates a stable clustering structure automatically, and a non-hierarchical cluster analysis concept, which classifies new components automatically. The clustering information generated from the proposed approach can support the domain analysis process.

Key Words : Classification, Cluster, Component

1. 서 론

컴포넌트(특히, 소프트웨어 재사용 컴포넌트)에 대한 분류는 컴포넌트를 재사용하려는 개발자들에게 그들의 필요에 가장 적합한 후보 컴포넌트들을 발견할 수 있도록 한다. 컴포넌트들의 수가 증가할 때, 모든 컴포넌트를 개별적으로 조사하는 것은 불가능하다. 이를 위해, 컴포넌트의 특징을 가장 효과적으로 표현하는 정보에 기초하여 유사한 정보를 포함하는 컴포넌트들을 같은 그룹에 속하도록 하는 다양한 분류 방법들이 제안되었다[1-5].

그러나, 전통적인 분류 방법들은 도메인 분석을 통해 생성된 미리 정의된 분류 구조에 의해 컴포넌트 특징을 표현한다. 이러한 전통적인 분류 방법들은 도메인 분석을 통한 분류 구조 생성에 많은 노력과 비용을 필요로 한다. 또한, 생성된 분류 구조의 엄격함은 컴포넌트의 삽입 혹은 제거에 의해 해당 도메인이 진화할 때, 유지 보수의 어려움을 야기한다. 컴포넌트들에 대한 의미 정보는 물론, 컴포넌트 사이의 관계를 표현할 수 있는 지식 기반 분류 방법들 역시, 영역 지식 획득하기 위해 상당한 노력을 필요로 하며, 컴포넌트들의 표현과 질의 구성을 어렵게 한다. 도메인 분석의 어려움으로 인하여 이러한 방법들은 한정된 도메인에서 적용된다.

이들 문제들을 부분적으로 해결하기 위해 클러스터 분석 방법들이 제안되었다[6, 7]. 이들 방법들은 컴포넌트의 특징을 표현하기 위해 정형 명세 언어를 사용한다. 이러한 방법들은 본 논문의 2장에서 언급되는 컴포넌트 분류(표현) 방법에서 명세 기반 방법들로 구분될 수 있다. 정형 명세 방법은 의미 정보와 같은 컴포넌트의 상세한 특징을 표현할 수 있는 반면, 도메인 분석을 필요로 하여 분류 구조 생성에 많은 노력이 필요 하다. 또한, 이들 방법들은 미리 정의된 (혹은 참조되는) 클러스터 시드들을 기초로 하여 컴포넌

※ 이 논문은 2006년도 한신대학교 학술연구비 지원에 의하여 연구되었음.
† 정 회 원 : 한신대학교 컴퓨터정보소프트웨어학부 부교수
논문접수 : 2006년 7월 27일, 심사완료 : 2006년 12월 9일

트들을 분류하므로 본 논문에서 제안되는 텍스트 기반의 복합 클러스터 분류 방법의 클러스터 시드 생성에 대한 자동화 분류와는 차이가 있다.

본 논문은 재사용 컴포넌트들에 대한 복합 클러스터 기반의 분류 방법을 제안한다. 통계 처리에 의해 유사 객체들을 그룹하는 클러스터 분석 분류 방법은 생물학, 의학, 마케팅과 같은 다양한 분야에 적용되어 왔다[8]. 특히, 정보 과학 분야에서 클러스터 분석 기술은 문서 사이의 유사성을 측정하여 관련된 문서들의 그룹을 자동으로 생성한다. 제안된 방법은 도메인에 대한 전체적인 컴포넌트들에 대한 안정적이고 정확한 기본 구조를 얻기 위해 계층(hierarchical)과 변형된 비계층(non-hierachical) 클러스터 분석 방법을 적용하였다. 이렇게 생성된 기본 클러스터 구조에 기초하여 새로운 컴포넌트들에 대한 자동화된 분류는 비계층(nonhierarchical) 방법에 의해 수행된다. 컴포넌트들은 재사용을 위해 컴포넌트의 기능을 표현하는 유용한 문서들을 포함한다. 이러한 문서들은 대상 컴포넌트 라이브러리에 제안된 자동화된 클러스터 분석 방법을 적용할 수 있는 좋은 후보들이 된다.

본 논문의 구성은 다음과 같다. 2장에서는 전통적인 컴포넌트 분류 방법과 함께 그들의 장/단점에 대해 살펴본다. 3장에서는 정보 과학 분야에서 이용되는 클러스터 분석 방법에 대해 서술한다. 4장에서는 제안된 복합 클러스터 분석 방법에 대한 상세 설명과 함께 윈도우 도메인에 적용된 결과를 보인다. 5장에서는 기존 분류 방법과 제안된 방법을 비교 분석한다. 6장에서는 결론 및 향후 과제에 대해 알아본다.

2. 전통적인 분류 방법

컴포넌트들의 특징을 표현하고, 유사 특징을 갖는 컴포넌트들을 분류하는 방법들은 일반적으로 열거(enumeration), 특징(attribute), 패싯(facet), 명세(specification), 지식(knowledge), 그리고 텍스트(text) 기반 분류 방법들로 구분될 수 있다. 이들 중 열거, 특징,그리고 패싯 기반 분류 방법들은 모두 컴포넌트 분류를 위해 도메인 분석에 의해 정의된 한정된 어휘를 사용한다. 그러므로, 본 논문에서는 이들 3개의 분류 방법을 어휘 기반 방법으로 구분한다.

2.1 어휘 기반

열거 기반 방법은 도메인 분석을 통해 해당 도메인을 상호 배타적이고 계층적인 클래스들로 나눈다. 컴포넌트 분류는 정의된 클래스 계층 구조로부터 가장 바람직하게 해당 컴포넌트를 묘사하는 클래스를 선택함으로써 분류된다. 정확하고 이해하기 쉬운 계층 구조가 정의된다면 컴포넌트들 사이의 관계는 쉽게 이해될 수 있다.

특징 기반 분류 방법은 컴포넌트 도메인에 대한 특징들을 미리 정의한다. 그러나, 각 특징에 주어지는 값들은 결정되지 않는다. 컴포넌트는 하나 이상의 특징과, 각 특징에 주어지는 값에 의해 분류된다. 특징 분류 방법은 가장 쉬운 분류 메커니즘을 제공한다. 그러나, 유사 컴포넌트 검색은 복잡하다.

패싯 분류 방법에서 패싯이라 부르는 다양한 관점에 의해 컴포넌트는 분류된다. 패싯은 기본적인 어구(term)들로 구성된다. 이러한 어구들은 소속된 패싯(관점)의 기본적인 값을 나타낸다. 패싯과 어구들은 도메인 분석에 의해 정의된다. 컴포넌트는 서로 다른 패싯으로부터 어구들을 결합함으로써 표현된다. 이 방법은 패싯에 대한 값이 정의된 어구로부터 선택된다는 점을 제외하고 특징 기반 분류 방법과 유사하다.

대부분의 어휘 기반 분류 방법들은 분류의 용이함 때문에 Prieto-Diaz에 의해 제안된 패싯 기반 방법을[9] 사용하며, 이러한 패싯 기반 방법들은 적용된 도메인에 대하여 정의된 패싯의 차이에 의해 구분된다.

어휘 기반 분류 방법들은 표준화된 어휘를 사용하도록 강요한다. 그러므로, 분류 구조를 완전히 이해하는 사람이라면 컴포넌트들의 분류는 용이하다. 그러나, 도메인 분석에 의한 분류 구조를 개발하는 것은 많은 노력을 필요로 한다.

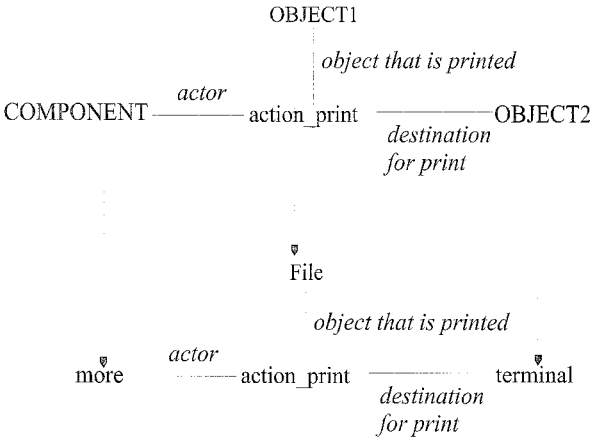
2.2 명세 기반

명세 기반 분류 방법에서 컴포넌트의 구문적인 특성과 의미 정보를 표현하기 위해 정형 명세 언어를 사용한다. 명세 언어는 컴포넌트들의 행위에 대한 상세한 묘사를 가능하게 한다. Nelson에 의해 제안된 CSRS(Class Storage and Retrieval System)은[10] 객체 지향 컴포넌트(클래스)들의 구문 구조(인터페이스와 기능)를 명세하기 위해 PDSL(Prototyping System Description Language) 명세 언어를, 그리고 컴포넌트들에 대한 의미 정보 표현을 위해 OBJ3 대수 명세 언어를 사용한다. 명세 기반 방법에 의해 컴포넌트의 행위와 같은 특징 정보를 상세히 표현 가능하지만, 복잡한 정형 명세 언어를 이해하는 어려움은 물론, 시스템에 의한 분류 자동화를 어렵게 한다.

2.3 지식 기반

지식 기반 방법들은 컴포넌트에 대한 의미 정보와 함께, 컴포넌트들 사이에 존재하는 구조적인(관계) 정보 표현이 가능 하다. 이러한 방법들은 적용되는 도메인에 대한 지식 정보를 코드화 하여 컴포넌트들을 표현하고, 후보 컴포넌트 검색 시 추론 능력을 제공한다. 지식 기반 방법들은 컴포넌트 분류를 위해 의미망(semantic network), 개념그래프(conceptual graph), 술어논리(predicate logic), 프레임(frame) 개념을 이용한다.

Wood에 의해 제안된 분류 방법[11]은 주로 자연어 이해 시스템에 이용되는 개념그래프 개념에 기초된다. 개념그래프는 적용되는 도메인에 대한 근본적인 개념들과 의미를 표현하기 위해 이용된다. 개념그래프를 컴포넌트 분류에 적용하기 위해 컴포넌트 서술자 프레임(CDF : Component Descriptor Frame) 구조를 개발했다. 기본 개념들은 각 컴포넌트가 수행하는 함수들과, 컴포넌트에 의해 다루어 지는 객체들로 구성된다. 개념적으로 유사한 기능들은 기본 함수로 분류된



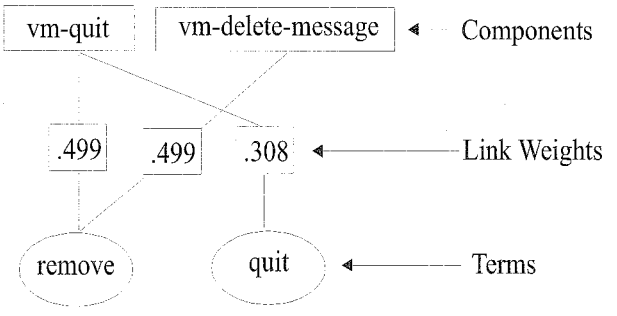
(그림 1) Component Descriptor Frame (CDF)

다. CDF는 이러한 기본 함수들에 대해 개발된다. 그림1은 UNIX 컴포넌트 도메인에서 개발된 25개의 기본 함수들에 대한 프레임에서 'print' 함수에 대한 CDF를 보인다. 그림의 상단 부분은 'print' 함수에 대한 CDF를 보인다. 그림의 하단 부분은 'more' 함수가 'print' 기본 함수로 분류되는 것을 보인다.

지식 기반 분류 방법이 컴포넌트에 대한 의미와 구조에 대한 정보 표현 능력을 제공한다 해도, 제한된 도메인에 대한 적용 문제, 지식 획득에 필요한 상당한 노력을 필요로하는 문제를 갖는다.

2.4 텍스트 기반

텍스트 기반 분류 방법은 컴포넌트에 대한 관련 소스 코드, 매뉴얼, 문서와 같은 텍스트 소스로부터 색인어를 자동 추출한다. 이러한 색인어는 컴포넌트들에 표현하기 위해 이용된다. 그러나, 언급된 방법들과 텍스트 기반 분류 방법은 인덱싱 과정에서 유사 컴포넌트들을 그룹화하기 위한 분류 구조는 생성되지 않는다. 오히려, 검색 과정에서 사용자 질의와 유사한 후보 컴포넌트들을 선택하기 위한 기법들이 제안된다. Henninger에 의해 제안된 CodeFinder 시스템은[12] 텍스트 기반 인덱싱과 함께, 유사 컴포넌트들에 대한 구별을 위해 spreading activation 기술을 이용하여 컴포넌트들 사이의 연관 네트워크를 구성한다. 그림2는 CodeFinder에 의해 구성된 Emacs 컴포넌트들에 대한 연관 네트워크를 보인다. 연관 네트워크는 단어들과 컴포넌트들을 위한 2 계층 노드들로 구성된다. 노드들 사이의 link weight는 컴포넌트들의 텍스트 파일로부터 각 컴포넌트에 포함된 단어들의 빈도에 의해 결정된다. 구성된 연관 네트워크에 기초하여 사용자로부터 주어진 컴포넌트와 유사한(연관된)컴포넌트들은 구별된다. 텍스트 기반 방법의 주된 장점은 컴포넌트에 대한 정보 표현과 유사 컴포넌트들의 검색 과정에 대한 시스템 자동화에 있다. 즉, 다른 방법에서 보여지는 영역 분석에 의한 분류 구조 생성 노력은 필요하지 않다. 그러나, 이 방법은 컴포넌트의 특징과 컴포넌트 사이의 관계 정보는 손실될 수 있다.



(그림 2) 연관 네트워크

3. 클러스터 분석 방법

본 논문에서 제안되는 분류 방법은 정보 과학(information science) 분야에서 문서 분류를 위해 사용되는 클러스터 분석 방법에 기초한다. 통계 처리에 의해 문서 사이의 유사성을 측정하여 관련된 문서들을 자동으로 그룹화하는 클러스터 분석 방법은 생물학, 의학, 마케팅과 같은 다양한 분야에 적용되어 왔다. 클러스터 분석 방법은 생성된 클러스터 구조 형태에 따라 계층(hierachical) 그리고 비계층(non-hierachical) 방법으로 분류된다[8, 13].

계층 방법은 관련된 문서를 파악하기 위해 모든 문서에 대한 유사성을 측정한다. 유사성 측정에 기초하여 모든 문서들이 최상위 클러스터에 소속될 때까지 점차적으로 중간 클러스터들로 그룹하여 2진 트리 구조를 생성한다. 계층적 방법들은 각 단계에서 어떤 객체(문서 혹은 클러스터)들이 결합되는가에 대한 기준에 따라 단일 링크(single link 혹은 nearest neighbor), 완전 링크(complete link), 그리고 그룹평균 링크(group average link) 방법으로 분류된다. 단일 링크 방법은 문서들 사이에서 가장 유사한 2개의 문서를 결합한다. 완전 링크방법은 서로 다른 클러스터에 할당된 문서들 중 가장 유사하지 않은 한 쌍의 문서를 결합한다. 그룹평균 링크 방법은 한 클러스터 내의 모든 객체 사이에서 한쌍의 링크들에 대한 평균 값에 기초하여 결합한다.

비계층 방법은 모든 문서에 대한 유사성을 측정하는 계층 방법과 달리, 클러스터 과정 초기에 기본 그룹으로써 클러스터 시드(seed)를 이용한다. 문서들은 이러한 시드들에 대한 유사성 측정에 의해 그룹 된다. 비계층 분석은 초기 시드 결정에 대한 방법에 의해 단일 패스(single-pass), 재배치(relocation 혹은 k-means) 방법으로 분류된다. 단일 패스에 의한 시드 결정은 입력되는 첫째 문서 자체가 하나의 클러스터 시드가 된다. 입력된 문서와 존재하는 클러스터 시드 사이의 유사성 측정에 기초하여 입력된 문서는 유사성 기존 클러스터에 포함되거나 혹은 새로운 클러스터를 생성한다. 이 방법에서 대상 문서들은 한번에 하나씩 클러스터 된다. 재배치 방법은 기존 문서 집합으로부터 임의의 문서들을 선택한다. 이러한 문서는 클러스터 시드가 된다. 결정된 초기 클러스터 시드로부터 개선된 클러스터 구조를 획득하기 위해, 클러스터와 클러스터 사이에 문서들은 옮겨진다.

4. 복합 클러스터 분석

본 논문에서 제안된 복합 클러스터 분석에 의한 분류 방법은 3장에서 언급된 계층 그리고 변형된 비계층 클러스터 분석 방법을 결합한다. 제안된 방법은 도메인에 대한 전체적인 컴포넌트들에 대한 안정적이고 정확한 기본 구조를 얻기 위해 계층(hierarchical)과 변형된 비계층(non-hierachical) 클러스터 분석 방법을 적용한다. 이렇게 생성된 기본 클러스터 구조에 기초하여 새로운 컴포넌트들에 대한 자동화된 분류는 비계층(nonhierarchical) 방법에 의해 수행된다.

복합 클러스터 분석의 첫번째 단계는 컴포넌트를 표현하기 위한 인덱싱 과정을 필요로 한다. 제안된 방법을 테스트하기 위해 윈도우 프로그래밍 컴포넌트들로 구성된 자바 AWT 라이브러리(API)에 적용되었다. 자바 컴포넌트에 대한 API 문서들은 컴포넌트 재사용을 위한 의미 있는 정보를 포함할 뿐 아니라 자동화된 클러스터 분석 방법에 필요한 다량의 문서로 구성된다. 인덱싱 과정을 통해 컴포넌트는 관련 API 문서들로부터 추출된 색인어에 의해 표현된다.

4.1 유사성 측정

인덱싱 과정 후, 문서 사이의 유사성 측정이 수행된다. 정보 과학 분야에서 문서 사이의유사성을 측정하는 가장 일반적인 방법은 'Overlab', 'Cosine', 'Dice', 그리로 'Jaccard' 방법이 있다[14]. 이들 중 'Overlab' 방법은 가장 간단하다.

$$SIM(D_j, D_k) = \sum_{i=1}^{n} (TW_{i,j} \times TW_{i,k})$$

여기에서, 문서 D_j 와 D_K는 공유된 색인어의 가중치(weighted value)이다. TW_{ij}는 문서 j의 어구 i에 대한 가중치이며, n은 문서 집합에서 독특한 어구들의 수를 표현한다. 그러나, 'Overlab' 측정 방법은 문서 크기 요소를 고려하지 않는다. 다른 3가지 측정 방법은 문서 사이의 유사성 계산에 문서 크기를 고려하여 정규화한다. 본 논문에서는 컴포넌트 사이의 유사성 계산을 위해 'cosign' 측정 방법이 이용된다.

Cosine 측정 :

$$SIM(D_j, D_k) = \sum_{i=1}^{n} (TW_{i,j} \times TW_{i,k}) \Big/ \sqrt{\sum_{i=1}^{n} TW_{i,j}^2 \times \sum_{i=1}^{n} TW_{i,k}^2}$$

Dice 측정 :

$$SIM(D_j, D_k) = 2\sum_{i=1}^{n} (TW_{i,j} \times TW_{i,k}) \Big/ \sum_{i=1}^{n} TW_{i,j}^2 + \sum_{i=1}^{n} TW_{i,k}^2$$

Jaccard 측정 :

$$SIM(D_j, D_k) = \sum_{i=1}^{n} (TW_{i,j} \times TW_{i,k}) \Big/ \sum_{i=1}^{n} TW_{i,j}^2 + \sum_{i=1}^{n} TW_{i,k}^2 - \sum_{i=1}^{n} TW_{i,j} \times TW_{i,k}$$

4.2 기본 클러스터 생성

컴포넌트 사이의 유사성 측정 벡터가 생성된 후, 도메인 구성 컴포넌트들에 대한 기본클러스터 구조가 생성된다. 기본 클러스터들을 결정하기 위해 계층 클러스터 분석 방법이 이용된다. 계층 방법은 모든 문서들의 유사성을 측정하므로 비계층 방법보다 더욱 많은 컴퓨터 자원을 소비한다. 그러나, 비계층 방법 보다 더욱 정확하고 안정적인 클러스터 구조를 생성한다. 제안된 방법에 의해 생성되는 기본 클러스터 구조는 도메인에서 컴포넌트 사이의 기본적인 관계 구조를 나타내는 것은 물론, 새로운 컴포넌트 분류를 위한 기초가 되므로 정확하고 안정적이어야 한다.

기본 클러스터 구조 생성을 위해 SPSS(Statistical Package for Social Science)통계 패키지의[15] 그룹평균 클러스터 방법을 이용하였다. SPSS 클러스터 분석 프로그램의 결과는 'agglomeration schedule'과 'dendrogam' 그래프로 묘사된다. 클러스터 과정에서 결합되는 컴포넌트 (혹은 클러스터)들을 구분하는 'agglomeration schedule' 그래프 그리고 그룹되는 컴포넌트의 구조를 시각적으로 표현하여 용이한 해석을 돕는 'dendrogram' 그래프는 초기 클러스터들의 결정을 돕는다. 결정된 기본 클러스터들의 수가 지나치게 적다면, 낮은 유사 측정값을 갖는 컴포넌트가 관련 클러스터에 소속될 수 있다. 이러한 경우, 선택된 클러스터들은 도메인에 대한 효과적인 클러스터 후보일 수 없다. 다양한 실험을 통해 초기 클러스터들의 수는 관련 도메인 컴포넌트들의 1/4인 것이 발견되었다. 보기를 들면, 62개 컴포넌트로 구성되는 자바 윈도우 도메인에 대해 16개의 초기 클러스터가 선택된다. <표 1>과 <표 2>는 각각 SPSS로 부터 그룹 평균 클러스터 방법에 의해 생성된 'agglomeration schedule'을, 그리고 'dendrogram' 그래프 분석에 의해 16개 기본 클러스터에 소속된 컴포넌트를 보인다.

4.3 클러스터 시드 생성

4.2절에서 선택된 기본 클러스터는 도메인에 포함된 컴포넌트들의 수에 따라 많은 클러스터들을 양산한다. 이러한 커다란 클러스터 구조는 해당 도메인에 대한 전반적인 관련 컴포넌트를 이해하는 것을 어렵게 한다. 또한, 기본 클러스터 구조는 관련 도메인에 대한 컴포넌트들의 근본적인 분류 구조를 생성하는 것이기 때문에 안정된 구조를 필요로 한다. 이러한 문제들을 고려하여 본 논문에서는 16개의 초기 클러스터로부터 8개의 클러스터 시드들이 선택된다. 선택된 시드들은 초기 클러스터들에 할당된 컴포넌트들이 많은 것부터 순서대로 선택된다. 클러스터 시드를 결정하는 근본 목적은 도메인에 대한 근본 분류 구조를 생성하는 것이기 때문에 선택된 시드에 할당되지 않은 모든 컴포넌트들은 선택된 시드 중 하나에 포함되어야 한다. 이를 위해 본 논문은 변형된 비계층 클러스터 분석 방법을 적용한다.

클러스터 시드에 소속된 모든 컴포넌트들을 대표하기 위한 클러스터 센터가 계산된다. 클러스터 센터는 클러스터 시드에 할당된 모든 컴포넌트와 관련된 색인어 가중치 평균

〈표 1〉 Agglomeration Schedule

Stage	Cluster1	Cluster2	Coef.	Next Stage
1	56	57	.828650	41
2	9	30	.791211	4
	51	52	.744774	61
4	9	50	.655075	33
5	53	55	.649146	10
6	10	26	.637473	43
7	3	31	.634077	11
8	1	24	.620301	52
9	19	20	.557027	48
10	7	53	.555210	15
11	3	34	.536640	33
12	46	48	.527234	16
13	40	49	.525350	16
14	36	37	.517067	23
15	6	7	.511854	20
16	40	46	.501017	24
17	35	41	.482068	37
18	17	25	.464620	30
19	44	45	.459324	45
20	6	54	.444539	21
21	6	58	.425908	39
22	27	62	.392181	40
23	36	38	.380140	34
24	21	40	.379016	32
25	8	60	.349296	36
26	4	32	.346685	38
27	42	47	.344692	45
28	12	22	.302710	35
29	14	15	.297962	49
30	16	17	.297093	43
31	59	61	.290482	36
32	21	43	.270920	38
33	3	9	.263134	44
34	33	36	.231711	50
35	12	13	.216357	49
36	8	59	.213528	42
37	5	35	.204411	50
38	4	21	.185466	44
39	6	39	.183577	41
40	27	28	.164312	42
41	6	56	.156984	51
42	8	27	.142071	47
43	10	16	.132480	48
44	3	4	.116480	47
45	42	44	.111314	54
46	2	23	.085828	55
47	3	8	.077348	53
48	10	19	.071548	52
49	12	14	.058084	55
50	5	33	.056141	53
51	6	18	.049249	58
52	1	10	.043052	57
53	3	5	.038545	56
54	11	42	.036909	56
55	2	12	.031210	57
56	3	11	.026781	58
57	1	2	.025106	59
58	3	6	.023790	59
59	1	3	.015397	60
60	1	29	.008733	61
61	1	51	.005645	0

〈표 2〉 클러스터 할당 컴포넌트

Label	Case	16
PrintGraphics	1	1
Shape	2	2
ItemSelectable	3	3
MenuContainer	4	3
Adjustable	5	4
LayoutManager	6	5
LayoutManager2	7	5
AWTEvent	8	6
CheckboxGroup	9	3
Color	10	7
Cursor	11	8
Rectangle	12	9
Dimension	13	9
Font	14	10
FontMetrics	15	10
Graphics	16	7
Image	17	7
Insets	18	11
AWTError	19	12
MediaTracker	20	12
MenuShortcut	21	3
Point	22	9
Polygon	23	2
PrintJob	24	1
Toolkit	25	7
SystemColor	26	7
Component	27	6
Button	28	6
Canvas	29	13
Checkbox	30	3
Choice	31	3
Container	32	3
Label	33	14
List	34	3
Scrollbar	35	4
TextComponent	36	14
TextArea	37	14
TextField	38	14
Panel	39	5
Menu	40	3
ScrollPane	41	4
Window	42	15
PopupMenu	43	3
Dialog	44	15
FileDialog	45	15
MenuItem	46	3
Frame	47	15
MenuComponent	48	3
MenuBar	49	3
CheckboxMenuItem	50	3
AWTException	51	16
IllegalComponent	52	16
BorderLayout	53	5
CardLayout	54	5
FlowLayout	55	5
GridBagConstrain	56	5
GridBagLayout	57	5
GridLayout	58	5
EventDispatchThr	59	6
Event	60	6
EventQueue	61	6

이다. 임의의 도메인에서 n이 도메인에 포함된 구별되는 어구들의 총 개수라고 가정하면, 컴포넌트 j는 다음과 같은 어구 가중치 벡터에 의해 표현될 수 있다.

$$C_j = (NTW_{1j}, NTW_{2j}, \cdots\cdots, NTW_{nj}),$$

여기서 NTW_{ij} 는 컴포넌트 j에서 어구 i에 대한 정규화된 가중치를 표현한다. 이때, 클러스터 센터(CC_k)는 다음 형태의 벡터로 표현될 수 있다.

$$CC_k = \left(\frac{\sum_{i=1}^{f} NTW_{1i}}{f}, \frac{\sum_{i=1}^{f} NTW_{2i}}{f}, \cdots\cdots, \frac{\sum_{i=1}^{f} NTW_{mi}}{f} \right)$$

여기서, f는 클러스터 시드에 포함된 $C_{1..f}$ 로 구성하는 컴포넌트들의 개수이다. 이때, 할당되지 않은 컴포넌트는 각 시드 센터들과 가장 높은 유사 측정값 결과에 의해 할당된다. 유사성 측정은 이미 정규화된 두개의 벡터들 사이의 'overlab' 방법에 의해 계산된다. 클러스터 시드에 컴포넌트 포함을 반영하기 위해 센터는 다시 계산된다. 표3에서 가중치1은 모든 62개의 기본 컴포넌트들이 8개의 클러스터 시드들에 할당된 후, 각 시드 센터에서 4개의 가장 높은 유사성 가중치를 갖는 색인어를 보인다. 62개의 자바 AWT 컴포넌트들은 클러스터 시드1에 메뉴관련 13개, 시드2에 레이아웃관련 10개, 시드3에 이미지관련 9개, 시드4에 그래픽관련 7개, 시드5에 사용자인터페이스관련 5개, 시드6에 예외관련 8개, 시드7에 'polygon'과 같은 기하 형태관련 5개, 시드8에 폰트관련 5개의 컴포넌트들이 클러스터링 되었다.

4.4 컴포넌트 분류

도메인을 구성하는 기본 컴포넌트들로부터 생성된 클러스터 시드들은 새로운 컴포넌트들을 분류하는 기초가 된다. 새로운 컴포넌트들은 4.3절에서 언급된 할당되지 않은 기본 컴포넌트들에 대한 비계층 클러스터 분석 과정이 적용된다. 'ColorPicker' 라고 불리우는 자바 GUI 응용 프로그램을 구성하는 4개의 컴포넌트들(ColorEvent, ColorListener, ColorPicker, ColorPickerMulticaster)이 클러스터 시드들에 적용되었고, 이들은 모두 클러스터 시드 4에 할당되었다. <표 3>에서 가중치1은 기본 컴포넌트들에 대한 클러스터 센터를 보이고, 가중치2는 4개의 새로운 컴포넌트들이 시드4에 할당된 후 클러스터 센터 결과를 보인다.

5. 시스템 평가

본 논문에서 제안된 복합 클러스터 분류에 대한 평가는 회상도(precision)와 정확도(precision)를 이용한 성능 평가와 함께, 전통적인 분류 방법들과의 특징 비교를 통해 수행된다.

5.1 성능 평가

일반적으로 검색 시스템들의 성능 평가를 위해 많이 사용

〈표 3〉 새로운 컴포넌트 분류에 의한 클러스터 센터

클러스터 시드	클러스터 센터		
	색인어	가중치1	가중치2
1	menu	0.1047	0.1089
	item	0.0866	0.0897
	ckeckbox	0.0557	0.0568
	select	0.0288	0.0267
2	inset	0.2050	0.1080
	layout	0.0661	0.0469
	bottom	0.0511	0.0464
	left	0.0501	0.0452
3	error	0.2752	0.1107
	media	0.0452	0.0507
	image	0.0409	0.0264
	awt	0.0398	0.0236
4	page	0.1073	0.0843
	print	0.1058	0.0319
	graphic	0.0946	0.0171
	resolution	0.0449	0.0148
5	text	0.1254	0.1297
	label	0.0767	0.0787
	action	0.0498	0.0513
	button	0.0402	0.0412
6	exception	0.1641	0.0611
	detail	0.1009	0.0587
	message	0.1009	0.0586
	illegal	0.0952	0.0559
7	polygon	0.0870	0.0532
	bound	0.0461	0.0499
	inside	0.0435	0.0493
	rule	0.0348	0.0354
8	font	0.0796	0.0549
	advance	0.0395	0.0505
	character	0.0367	0.0489
	width	0.0334	0.0436

되는 회상도(precision)와 정확도(precision)에 대한 측정은 다음과 같이 계산된다. N(Rel)은 질의 관련 컴포넌트들의 수, N(Ret)는 검색된 컴포넌트들의 수를 나타낸다. N(Rel & Ret)는 검색된 컴포넌트 중 질의와 관련된 컴포넌트들의 수이다. 즉, 회상도는 해당 도메인에서 사용자 요구(질의) 관련 컴포넌트들이 검색 결과에 나타나는 정도를 나타내며, 정확도는 검색 결과 중 사용자 요구 관련 컴포넌트들이 포함되어있는 정도를 나타낸다.

- 회상도 : $N(Rel \cap Ret)/N(Rel)$
- 정확도 : $N(Rel \cap Ret)/N(Ret)$

이들을 측정하기 위해 다음 2가지 문제에 대한 결정이 필요하다. 첫 번째 문제는 질의와 컴포넌트 사이의 관련 여부에 대한 결정이다. 각 컴포넌트가 해당 질의와 관련되는지 아닌지에 대한 판단은 주관적일 수 있으므로 가능한 객관적인 판단 방법에 의해 컴포넌트 관련성이 결정되어야 한다. 두 번째 문제는 회상도와 정확도에 의한 측정 결과는 상대적인 평가 방법이므로 같은 조건하에서 비교 대상이 되는 시스템이 존재해야 한다는 것이다.

이러한 문제와 함께 제안된 분류 방법에 대한 성능을 평가하기 위해, 본 논문은 두 개의 실험적인 검색 시스템(제안된 복합 클러스터 기반 검색 시스템, 매칭되는 단어 빈도

수에 의한 단순한 텍스트 기반 검색 시스템)을 구현하였다. 복합 클러스터 검색 시스템은 질의와 높은 유사성을 갖는 클러스터 구분 후 구분된 클러스터의 각 컴포넌트와의 유사 정도에 의해 후보 컴포넌트는 검색된다. 컴포넌트에 대한 질의 관련 여부는 질의에 의해 묘사되는 자바 프로그램에서 이용되는 컴포넌트들로 정의된다. 질의 표현은 자연어로 표현되며 인덱싱 과정을 통한다.

- 방법

보기를 들면 다음 자바 윈도우 프로그램에 대한 문제는 자연어 질의로 구성되어 실험적인 두 개의 검색 시스템에 입력된다.

"The program creates a window with a text area thathas a black background and white foreground. The white foreground causes the text to be painted in white color. Also, the text area is set with a large font [16]"

5개의 자바 AWT 컴포넌트(TextArea, Color, Font, BorderLayout, Frame)은 위 문제에 대한 프로그램에 포함되므로 질의 관련 컴포넌트라고 정의한다. 이러한 경우 표4는 해당 문제에 대한 클러스터 기반 검색 시스템의 회상도와 정확도 값을 보인다. 문제는 5개의 관련 컴포넌트들을 갖으므로 N(Rel)=5 이다. 검색된 'Color' 컴포넌트의 회상도 값은 N(Rel ∩ Ret) / N(Rel)에 의해 0.4 이고, 정확도 값은 N(Rel ∩ Ret) / N(Ret)에 의해 0.333 이다.

〈표 4〉 클러스터 기반 시스템에 대한 회상도/정확도 (부분 리스트)

컴포넌트 이름		회상도	정확도
TextArea (0.138715307277583)	*	0.200	1.000
TextComponent (0.1080616503423)		0.200	0.500
Label (0.0707205741229499)		0.200	0.333
TextField (0.0576061944623734)		0.200	0.250
Window (0.0437769344624042)		0.200	0.200
Color (0.0415829629607313)	*	0.400	0.333
SystemColor (0.0408257257986575)		0.400	0.286
Canvas (0.0292800710339185)		0.400	0.250
Font (0.0262330679148043)	*	0.600	0.333
FontMetrics (0.0261154464686138)		0.600	0.300
Frame (0.0184221011173103)	*	0.800	0.364
Graphics (0.0139959229023005)		0.800	0.333

* 질의 관련 컴포넌트

-결과

<표 5>는 5개의 자바 윈도우 문제 프로그램에 적용된 두 시스템에 대한 회상도/정확도 결과를 보인다. 회상도 0.4 이상에서 두 시스템의 정확도는 유사했다. 그러나, 회상도 0.4 이하에서 제안된 분류 방법에 의한 정확도는 낮았다. 이는 제안된 복합 클러스터 검색 방법에서 탐색 영역을 유사도가 높은 클러스터로 한정하는 것과 같은 독특한 특성으로부터 발생한다. 그러나, 제안된 클러스터 분석 방법의 낮은 성능평가 결과에도 불구하고, 탐색 공간의 한정에 의한 빠른 응답 시간 결과 제공, 클러스터 분석에 의한 도메인 분석 지원과 같은 부수적인 결과에 대하여 고려하는 것이 필요하다.

〈표 5〉 클러스터 기반 시스템에 대한 평균 회상도/정확도

		정확도											
		텍스트 기반						클러스터 기반					
		문제1	문제2	문제3	문제4	문제5	평균	문제1	문제2	문제3	문제4	문제5	평균
회상도	0.1	0	0	0.333	0	0	0.067	0	0	0.250	0	0	0.050
	0.2	0.250	0.333	0.333	1	0.286	0.440	1	1	0.250	0.500	0.250	0.600
	0.3	0.333	0.333	0.455	0.667	0.286	0.415	1	0.333	0.286	0.200	0.400	0.444
	0.4	0.333	0.333	0.455	0.667	0.286	0.415	1	0.333	0.240	0.200	0.400	0.435
	0.5	0.333	0.300	0.538	0.429	0.286	0.377	0.190	0.333	0.250	0.200	0.500	0.295
	0.6	0.294	0.300	0.300	0.429	0.286	0.322	0	0.333	0.276	0.200	0.313	0224
	0.7	0.294	0.300	0.300	0.333	0.286	0.303	0	0.333	0.256	0.200	0.313	0.220
	0.8	0.200	0.333	0.333	0.333	0.313	0.302	0	0.364	0.275	0.211	0.300	0.230
	0.9	0	0.139	0.317	0.106	0.143	0.141	0	0	0	0.179	0.189	0.074
	1.0	0	0.139	0.326	0.106	0.143	0.143	0	0	0	0.179	0.189	0.074

〈표 6〉 분류 방법 비교

	어휘기반	명세기반	지식기반	텍스트기반	
				기존 방법	복합 클러스터 방법
장점	-정의된 분류구조에 의한 컴포넌트 구분 용이 -표준화된 어휘 사용	-컴포넌트의 구문/의미 정보 표현 가능	-컴포넌트 행위 표현 -추론 기능	-분류과정 자동화	-분류구조 자동 생성 -도메인 분석 지원 -새로운 컴포넌트 자동 분류 -도메인에 한정되지 않음
단점	-도메인 분석 위한 노력 -컴포넌트 관계 표현 어려움	-정형 명세 언어 이용 어려움 -컴포넌트 명세 어려움	-도메인에 대한 지식 획득 노력 필요 -한정된 도메인 적용	-컴포넌트 특징 정보 손실 -컴포넌트 관련 문서 요구	-컴포넌트 특징 손실 -컴포넌트 관련 문서 요구

5.2 특징 비교

<표 6>은 2장에서 언급된 4가지 분류 방법들에 대한 장/단점과 함께 제안된 방법에 대한 특징을 보인다. 텍스트 기반 분류 방법을 제외한 나머지 방법들은 도메인 분석을 통한 분류 구조 생성을 위해 많은 노력을 필요로 한다. 반면, 텍스트 기반 방법은 분류 과정의 자동화에 의해 도메인에 대한 지식 획득 노력을 필요로 하지 않는다. 그러나, 정의된 분류 구조 없이 유사 컴포넌트들에 대한 관계와 특징 정보는 손실된다.

제안된 복합 클러스터 분석 방법은 기존 텍스트 기반 방법과 다르게 클러스터 시드에 의한 분류 구조를 생성한다. 생성된 구조는 대상 도메인에 대한 전체적인 컴포넌트의 유사 관계를 파악하는 것을 돕는다. 클러스터링 과정에서 생성되는 정보는 대상 도메인을 분석하는 기초가 될 수 있다. 보기를 들면, 패싯 기반 방법에서, 도메인 전문가는 패싯 그리고 패싯 관련 키워드(어구)들을 미리 정의하고 이들을 결합함으로써 컴포넌트를 표현한다. 이때, 제안된 방법에 의해 생성된 클러스터 정보에 대한 분석은 도메인 전문가들이 이러한 패싯과 키워드들을 결정하는 것을 도울 수 있다. 제안된 분류 방법은 텍스트 기반 분류 방법에 포함된다. 그러므로, 컴포넌트 관련 풍부한 텍스트 문서들이 존재 하는 다양한 도메인에 적용될 수 있다.

6. 결 론

본 논문에서는 컴포넌트의 재사용을 위한 접근을 가능하게 하는 기존의 다양한 분류 방법에 대해 언급하였다. 이러한 분류 방법들은 컴포넌트를 표현하는 방법에 의해 어휘, 명세, 지식, 텍스트 기반 방법들로 구분될 수 있다. 이들 분류 방법은 각각의 장/단점을 포함한다. 이들의 단점을 극복하기 위해 계층적 그리고 비계층적 클러스터 분석 개념을 결합하는 복합 클러스터 분석 방법이 제안되었다. 제안된 방법은 자동화된 분류 구조 생성은 물론, 새로운 컴포넌트의 자동 분류를 지원한다. 또한, 생성된 클러스터 구조는 해당 도메인에 대한 분석 과정을 지원할 수 있으며 다양한 도메인에 적용될 수 있다. 향후 연구에서 제안된 방법의 유용성을 테스트하기 위해 다양한 도메인에 대한 적용 결과를 보이는 것이 필요할 것이다. 특히, 컴포넌트의 기능에 대한 유용한 정보를 포함하는 디자인 패턴, XML 웹서비스, 테스트 케이스, Ejb와 같은 컴포넌트에 대한 적용이 요구될 것이다.

참 고 문 헌

[1] W. B. Frakes and T. P. Pole, "An Empirical Study of Representation Methods for Reusable Software Components," *IEEE Transactions on Software Engineering*, Vol.20, No.8, pp.617-630, Aug., 1994.

[2] A. Goldberg and K. S. Rubin, Succeeding with Objects: Decision Frameworks for Project Management, Addison Wesley, 1995.

[3] Lee, S. K., "Web 소프트웨어 컴포넌트 재사용을 위한 라이브러리 관리와 서비스," 정보과학회논문지, 제29권 제1호, pp.10-19, 2002.

[4] Bruce W. N. Lo and Huilin Ye, "Self-Organization of Software Asset Library for Reuse," *Journal of Systems Research and Information Systems*, Vol.10, pp.3-21, 2001.

[5] S. Poulin and P. Yglesias, "Experiences with a Faceted Classification Scheme in a Large Reusable Software Library (RSL)," *IEEE 17th Annual International Computer Software & Applications Conference (COMPSAC93)*, Phoenix, Arizona, pp.90-99, 1993.

[6] S. Nakkrasae, P. Sophatsathit, and W. R. Edwards, Jr., "Fuzzy Subtractive Clustering Based Indexing Approach for Software Components Classification", *International Journal of Computer & Information Science*, Vol.5, No.1, pp.63-72, 2004.

[7] K. S. Daudjee and A. A. Toptsis, "A Technique for Automatically Organizing Software Libraries for Software Reuse", *Proceedings of the 1994 Conference of the Centre for Advanced Studies on Collaborative Research*, 1994.

[8] P. Willett, "Recent Trends in Hierarchic Document Clustering: A Critical Review," *Information Processing & Management*, Vol.24, No.5, pp.577-597, 1988.

[9] R. Prieto-Diaz and P. Freeman, "Classifying Software for Reusability," *IEEE Software*, Vol.4, No.1, pp.6-16, Jan., 1987.

[10] M. L. Nelson and T. Poulis, "The Class Storage and Retrieval System: Enhancing Reusability in Object-Oriented Systems," *OOPS Messenger*, Vol.6, No.2, pp.28-36, Apr., 1995.

[11] M. Wood and I. Sommerville, "An Information Retrieval System for Software Components," *Software Engineering Journal*, Vol.3, No.5, Sep., 1988.

[12] S. Henninger, "Supporting the Construction and Evolution of Component Repositories," *Proceedings of 18th International Conference on Software Engineering (ICSE-18'96)*, pp.279-288, 1996.

[13] B. S. Everitt, *Cluster Analysis: Third Edition*, John Wiley & Sons Inc., 1993.

[14] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," *Information and Processing and Management*, Vol.24, No.5, pp.513-523, 1988.

[15] SPSS Inc., *SPSS Professional Statistics 6.1*, pp.83-142, 1994.

[16] P. Chan and R. Lee, *The Java Class Libraries: Second Edition*, Vol.2, Addison-Wesley Longman Inc., 1998.

이 성 구

1986년 중앙대학교 전자계산학과(학사)
1989년 중앙대학교 전자계산학과(석사)
1993년 Arizona 주립대학 컴퓨터공학과 (석사)
1998년 Arizona 주립대학 컴퓨터공학과 (박사)
1999년~현재 한신대학교 컴퓨터정보소프트웨어학부 부교수
관심분야 : 소프트웨어공학, 객체지향, 인터넷 프로그래밍