# Web 소프트웨어 컴포넌트 재사용을 위한 라이브러리 관리와 서비스

## (Library Management and Services for Software Component Reuse on the Web)

이 성 구 †

(Sung-Koo Lee)

**요 약** 사용자들은 웹브라우저를 통해 웹에 존재하는 소프트웨어 컴포넌트 라이브러리를 탐색한다. 그러나, 웹 라이브러리에 포함된 컴포넌트들의 수가 빠르게 증가하는 상황에서 우리는 이러한 라이브러리들을 효과적으로 구성하고 관리하기 위한 방법을 필요로 한다. 전통적인 웹 라이브러리 검색 시스템들은 컴포넌트들을 저장하고 검색하기 위해 다양한 분류방법을 이용한 검색 서비스를 제공한다. 이러한 전통적인 시스템들은 사용자들이 검색 초기 단계에서 라이브러리의 전체적인 내용의 이해를 바탕으로 한 다양한 검색 서비스를 준비하지 못한다.

본 논문은 전통적인 시스템들의 단순한 컴포넌트 저장과 검색이상의 다양한 서비스와 객체지향 컴포넌트들의 효율적인 관리를 제공하는 웹 라이브러리 시스템에 대해 토론한다. 이러한 서비스들은 역공학 프로세스를 통한 컴포넌트 이해서비스, 라이브러리 요약내용 자동생성서비스, 이해기반 검색서비스이다. 또한, 본 논문에서 적용된 자동화된 클러스터 기반 분류체계 방법의 성능은 전통적인 분류방법을 이용하는 2개의 다른 시스템들의 성능과 비교, 평가된다.

**키워드** : 소프트웨어 재사용, 라이브러리, 검색, 분류, 클러스터링, 웹, 데이타베이스, 역공학

***Abstract*** In searching and locating a collection of components on the Web, users require a Web browser. Since the Web libraries tend to grow rapidly, there needs to be an effective way to organize and manage such large libraries. Traditional Web-based library(retrieval) systems provide various classification scheme and retrieval services to store and retrieve components. However, these systems do not include invaluable services, for example, enabling users to grasp the overall contents of the library at the beginning of retrieval.

This paper discusses a Web-based library system, which provides the efficient management of object-oriented components and a set of services beyond simple component store and retrieval. These services consist of component comprehension through a reverse engineering process, automated summary extraction, and comprehension-based retrieval. Also, The performance of an automated cluster-based classification scheme adopted on the system is evaluated and compared with the performance of two other systems using traditional classification scheme.

**Kye words** : Software rense, library, classification, clustering, web, database, reengineering

## 1. 서 론

과거 30년간 소프트웨어 공학분야는 소프트웨어 생산성을 높이고, 질을 향상시키기 위해 소프트웨어 재사용의 기술적인 문제들을 해결하려고 노력해왔다[1-2]. 이러한 기술적인 문제를 해결하기 위한 2가지 접근방법이 있다. 첫째, 'for reuse'에 의한 소프트웨어 개발 방법으로써 사용자들에 의해 이용될 수 있는 재사용 컴포넌트들을 개발하려는 접근이다. 둘째, 'with reuse'에 의한 소프트웨어 개발 방법으로써 재사용할 수 있는 컴포넌트들로 구성된 라이브러리로부터 소프트웨어 시스템을 개발하는 것이다. 두 번째 접근 방법에서 개발되기 위한

· 이 논문은 2000년도 한신대학교 학술연구비 지원에 의하여 연구되었음.

† 정 회 원 : 한신대학교 정보시스템공학과 교수

sklee@hanshin.ac.kr

논문접수 : 2001년 3월 10일

심사완료 : 2001년 10월 9일

소프트웨어를 위해 사용자들의 질의 요구를 충족하는 후보 컴포넌트들을 제공하기 위해 다양한 분류방법에 의한 검색기능을 제공하는 소프트웨어 재사용 라이브러리 시스템을 필요로 한다.

전통적인 재사용 라이브러리 시스템들의 대부분은 컴포넌트들을 구별하기 위해 텍스트기반 혹은 패싯(facet) 기반의 분류 방법을 이용한다[3-7]. 그러나, 이러한 시스템들은 컴포넌트사이의 중요한 관계에 대한 정보 손실의 가능성, 혹은 영역전문가(domain expert)에 의한 분류체계 구축을 위해 상당한 영역분석(domain analysis) 노력을 필요로 한다. 텍스트기반의 분류방법은 컴포넌트들에 상응하는 텍스트 자원들(소스코드 혹은 매뉴얼)로부터 인덱싱 과정을 통해 컴포넌트들을 분류하는 것이다. 이러한 인덱싱 과정은 자동화될 수 있으므로 이 방법의 주된 장점은 컴포넌트 분류비용의 최소화이다. 그러나 이 방법은 컴포넌트 사이의 중요한 정보가 처리과정에서 손실될 수 있다. 패싯기반 분류방법은 영역 전문가는 패싯이라고 불려지는 컴포넌트의 다양한 관점(viewpoint)과 각 패싯에 속하는 어구(term)를 결정함으로써 기본적인 분류체계를 준비하고 이들을 결합하여 영역 전문가는 컴포넌트들을 분류한다. 그러나 패싯기반 라이브러리 시스템들은 분류체계를 구축하기 위해 많은 분석 노력을 필요로 한다. 일반적으로, 전통적인 라이브러리 시스템들은 다음 5가지 문제들을 포함한다.

- 빠르게 성장하는 Web 라이브러리 관리의 어려움
- 라이브러리의 전체적인 내용 이해의 어려움
- 분류체계를 구현하기 위한 영역분석의 어려움
- 사용자들로부터 다양한 질의 요청을 지원하는 검색능력 부족
- 웹 사용자들 사이에 컴포넌트들의 공유능력 부족

본 논문은 이러한 문제들을 해결하기 위해 다양한 서비스와 객체지향 컴포넌트들의 효율적인 관리를 제공하는 SOORLS(Summary-based Object Oriented Library System)라는 웹 라이브러리 시스템을 구현하였다[8]. SOORLS의 클러스터 분류방법에 의해 자동으로 생성되는 클러스터 구조는 후보 컴포넌트들의 자동화된 분류서비스는 물론, 시스템이 제공하는 독특하고 다양한 서비스들의 기초가 된다. 이러한 서비스들은 리엔지니어링 프로세스를 통한 컴포넌트 이해서비스, 라이브러리 요약내용 자동생성서비스, 이해기반 검색서비스이다. 소프트웨어 프로그램을 분해하는 역공학 프로세스는 검색된 후보 컴포넌트들의 이해과정을 돕는다. 클러스터 구조에 의해 자동으로 생성된 라이브러리 요약내용 서비스는 사용자들이 검색 초기에 해당 라이브러리의 전체적인 내용을 이해시키는 것을 돕는다. 라이브러리 이해를 통한 검색 서비스는 사용자의 다양한 질의 요청을 만족시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장은 라이브러리 관리를 위한 데이타베이스 스키마 디자인을 보인다. 3장은 역공학 프로세스를 통한 컴포넌트 이해 서비스에 대해 토론한다. 4장은 자동으로 생성된 분류체계에 의해 라이브러리 요약내용을 추출하는 것을 보인다. 5장은 시스템이 다양한 사용자 요청을 어떻게 만족하는가를 보이고, 6장에서 구현된 시스템의 성능을 평가한다. 마지막으로, 본 논문의 결론과 향후 과제에 대해 언급한다.

## 2. 라이브러리 관리

소프트웨어 생명주기의 각 단계동안 발생하는 어떤 결과물도 미래 프로젝트에 대한 재사용 컴포넌트가 될 수 있다. 그러나 객체 지향 프로그래밍에서 클래스는 재사용 컴포넌트의 기본 단위가 된다. 본 논문은 SOORLS에서 제공하는 다양한 서비스를 구현하고, GUI(Graphical User Interface) 영역을 자동으로 분석하기 위해 Java AWT(Abstract Window Toolkit) 라이브러리 패키지의 62 컴포넌트들(클래스 혹은 인터페이스)이 시스템 라이브러리로 선택되었다. 그림1은 Java 언어 모델과 라이브러리 서비스들을 위한 SOORLS 데이타베이스 E-R(Entity-Relationship) 다이어그램을 묘사한다.

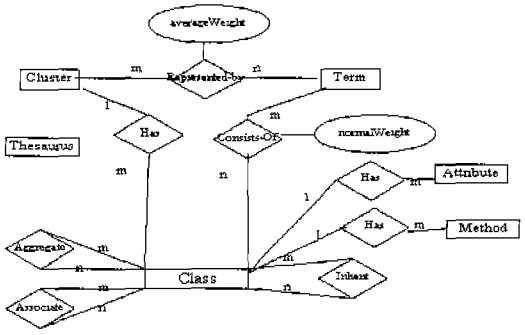


그림 1 시스템 데이타베이스 E-R 다이어그램

Java 언어 모델에서 구별될 수 있는 여러 엔터티(entity), 관계(relationship)형이 있다. 'Class' 엔터티형은 Java 컴포넌트(클래스 및 인터페이스)를 표현한다. 'Attribute' 와 'Method' 엔터티형들은 컴포넌트에 포함되는 속성과 행위를 표현한다. 'Associate' 와 'Aggregate' 관계형은 두 개의 컴포넌트들 사이의 상호

작용에 의한 관계를 표현한다. 'Inheritance' 관계형은 어떤 컴포넌트의 속성과 행위는 다른 컴포넌트에서 이용될 수 있다는 것을 표현한다. 'Has' 관계형은 어떤 컴포넌트가 다른 컴포넌트를 포함한다는 것을 표현한다. 본 논문에서 그림 1의 E-R 다이어그램은 단지 Java 언어 모델을 위해 설계되었지만 본 다이어그램은 대부분의 객체 지향 언어 모델에 적용될 수 있다.

또한, 그림 1은 라이브러리 서비스들을 제공하기 위한 엔터티 와 관계형들을 포함한다. 'Term'은 단일 인덱싱 단어를 표현한다. 'Cluster'는 유사한 컴포넌트들의 집합을 표현한다. 'Consists-of'는 컴포넌트들과 단어들 사이의 관계를 표시하며 여기에서 M:N은 한 컴포넌트는 여러 인덱싱 단어들로 구성하며 한 단어는 여러 컴포넌트들에 나타날 수 있다는 것을 가리킨다. 'Represented-by'는 단어들과 클러스터들 사이의 관계를 표현한다. 여기에서 M:N은 어떤 클러스터가 여러 단어들에 의해 표현되고 또한 하나의 단어는 여러 클러스터들에 나타날 수 있다는 것을 의미한다. 'Has'는 클러스터들과 컴포넌트들 사이의 관계를 표현한다. 여기에서 1:N은 어떤 클러스터가 여러 개의 컴포넌트들을 포함한다는 것을 의미한다. 'Thesaurus', 'Input', 'Intermediate', 'Similarity' 엔터티형들은 SOORLS의 이해 기반 검색 서비스를 위한 임시 저장을 위해 설계된다. 표 1은 그림 1의 E-R 다이어그램으로부터 변형된 관계 데이타베이스 스키마이다.

표 1 시스템 데이타베이스 릴레이셔널 스키마

| 관계형이름 | 속성 |
|---|---|
| Class | (class#, name, decl., example, version, author, package, type, visibility, modifier, URL, property) |
| Aggregation | (class#, aggreClass#) |
| Association | (class#, assocClass#) |
| Inherit | (class#, superClass#) |
| Method | (name, modifier, decl , desc., visibility, returnType, class#) |
| Attribute | (name, modifier, decl., desc., visibility, class#) |
| Term | (term#, name, assignedClass, IDF) |
| Consists-Of | (class#, term#, weight, frequency, normalWeight) |
| Represented-By | (cluster#, term#) |
| Thesaurus | (*termName*, *termSynonym*) |
| Similarity | (className, simWeight) |
| Input | (name, term#, frequency, normalWeight) |
| Intermediate | (term#, interAverageWeight) |

decl :declaration (클래스 정의), desc.:description (클래스 설명)

## 3. 컴포넌트 역공학

SOORLS에서 제공되는 라이브러리 서비스를 준비하기 위해 본 논문은 컴포넌트 역공학 프로세스를 적용한다. 재사용을 위한 역공학은 컴포넌트의 정보와 다른 컴포넌트들과의 구조적인 정보를 분석함으로써 사용자가 검색된 후보 컴포넌트들의 이해를 돕도록 한다. 수많은 SOORLS 도구중의 하나인 RIA(Reuse Information Analyzer)는 입력된 컴포넌트의 선언 부분을 분석함으로써 역공학 프로세스를 시작한다. 이를 위해 RIA는 컴포넌트와 그것의 멤버들에 대한 선언부분을 받아들이는 입력 폼을 사용자에게 준비한다. RIA는 컴포넌트 선언 부분을 Java 문법에 기초하여 분석함으로써 컴포넌트 이름, 반환 타입, 접근 제한자와 같은 정보는 물론 다른 컴포넌트들과의 구조적인 관계를 자동으로 추출할 수 있다. 표 2는 현재 RIA로부터 추출되는 정보를 보인다.

표 2 RIA로부터 추출되는 정보

| 속성 | 설명 | 값 |
|---|---|---|
| Name | 컴포넌트, 메소드, 속성 이름 | |
| Modifier | 부분적으로 구현된 컴포넌트 | Abstract |
| | 서브클래스될 수 없는 컴포넌트, 서브클래스에서 오버라이드될 수 없는 메소드, 상수 값을 갖는 속성 | Final |
| | 컴포넌트의 모든 인스턴스들에게 같은 메소드 혹은 속성 | Static |
| Visibility | 컴포넌트 멤버들과 외부세계 사이에서의 접근 정도 | Default, Public, Protected, Private |
| Component Type | 완전히 구현되지 않은 컴포넌트 | Interface |
| | 모든 오브젝트들에게 공통적인 멤버들을 정의하는 프로토타입 컴포넌트 | Class |
| Return Type | 메소드의 리턴 타입 | |
| Structural Relationships | 컴포넌트들 사이의 구조적인 관계 | |

RIA에 의해 컴포넌트들 사이의 구조적인 의존관계를 분석하고 분석된 정보를 데이타베이스에 저장함으로써, SOORLS는 사용자에게 검색된 후보 컴포넌트들을 이해하기 위한 전형적인 질의 생성을 가능하게 한다. SOORLS에 의해 제공되는 컴포넌트 이해를 위한 데이타베이스 질의는 다음과 같다.

- 어떤 컴포넌트에서 접근 가능한 모든 멤버들을 보여라.

- 어떤 컴포넌트와 관련된 모든 슈퍼 컴포넌트들을 보여라.
- 어떤 컴포넌트와 의존관계(association, aggregation)를 가진 모든 컴포넌트들을 보여라.
- 어떤 멤버이름을 포함하고 있는 모든 컴포넌트를 보여라.

위 언급된 질의와 별도로 SOORLS 데이타베이스는 컴포넌트와 관련된 정보(예, 이름, version, URL, 저자, 패키지이름)를 저장한다.

## 4. 요약내용 자동생성 서비스

SOORLS에 의해 제공되는 독특한 서비스중 하나인 요약내용 자동 추출 서비스는 사용자가 웹서버에 존재하는 라이브러리의 전체적인 내용을 한번에 파악할 수 있게 한다. 이러한 서비스를 준비하기 위해 컴포넌트 인덱싱 프로세스와 클러스터링 프로세스가 필요로 된다.

### 4.1 컴포넌트 인덱싱

서로 관련된 컴포넌트들을 그룹화 하기 전에 각 컴포넌트를 표현하는 인덱싱 과정이 필요하다. SOORLS에 적용된 인덱싱 프로세스는 컴포넌트로부터 추출된 인덱싱 단어들로 구성된 벡터에 의해 해당 컴포넌트를 표현하는 'vector space model'을 적용한다[4][9]. 이 모델에서 벡터는 인덱싱 단어들로 구성되며 각 단어는 해당 컴포넌트를 대표하는 정도에 따라 무게(weight)가 주어진다. 본 논문은 SOORLS의 성능 향상을 위해 2개의 인덱싱 프로세스를 고용했다: stop words, stemming. 영어에서 가장 빈번히 사용되는 단어들은 컴포넌트들을 표현하는데 전혀 의미를 갖지 못한다. SOORLS는 Frakes[3]에 의해 언급된 425 단어들을 stop words에 포함 시켰다. 불필요한 단어들이 제거된 후, 인덱싱 단어의 접미사(suffix) 부분을 제거하는 stemming 과정을 적용시켰다. 본 논문에서 적용된 stemming 알고리즘은 매우 단순하다. 이러한 단순한 알고리즘을 사용한 중요한 이유는 궁극적으로 인덱싱 단어들에 의해 자동으로 생성된 요약내용이 사용자에게 명백히 이해되어야 하기 때문이다. 단어들로부터 지나치게 많은 문자를 제거하는 복잡한 stemming 알고리즘을 이용한다면 결론적인 요약내용은 사용자에게 어떤 의미도 없을 것이다.

다음 단계는 컴포넌트를 더욱 잘 표현하는(특징짓는) 단어에게 무게를 주는 것이다. SOORLS는 TF(Term Frequency), ICF(Inverse Component Frequency), NTW(Normal Term Weight)의 3가지 요소를 고려하여 각 단어에 대한 무게를 계산한다. 컴포넌트에서 발생하는 단어의 빈도를 고려하는 TF가 해당 컴포넌트의 내용을 적절히 대표한다 해도, TF는 해당 컴포넌트와 다른 컴포넌트들과의 차이를 고려하는데 불충분하다. 그러므로 라이브러리에 있는 모든 컴포넌트들에서 해당 인덱싱 단어의 분산 효과를 고려하는 것이 필요하다. ICF는 어떤 컴포넌트에서 빈번히 발생하면서 다른 컴포넌트에서 드물게 발생하는 단어에게 더욱 많은 무게를 준다. TF와 ICF요소 외에 NTF는 컴포넌트의 길이를 고려한다. 즉, 검색을 위해 모든 관련된 컴포넌트들은 똑같이 중요한 것으로 고려되기 위해 컴포넌트의 길이를 표준화한다. 이러한 과정을 통해 각 컴포넌트는 무게가 주어진 단어들의 벡터에 의해 표현되고, 이에 따라 무게 값을 많이 갖는 특정 단어들은 컴포넌트들을 표현하는데 있어서 개념적으로 매우 중요한 역할을 한다.

Indexer는 앞서 언급된 과정을 구현하고 데이타베이스에 추론된 인덱싱 정보를 저장하는 SOORLS 도구이다. 표 3은 Indexer에 의해 AWT 컴포넌트중의 하나인 'Menu' GUI 컴포넌트의 인덱싱 정보를 보인다. 표의 결과는 NTW값의 내림차순에 의해 정렬되었고, 'Menu' 컴포넌트의 52개의 인덱싱 단어들 사이에서 위, 아래 7개의 단어들에 대한 결과를 보인다. 이 테이블로부터 우리는 다음과 같은 사실을 알 수 있다.

- 'menu'는 가장 높은 NTW값에 의해 'Menu' 컴포넌트를 가장 잘 대표하는 단어이다.
- 'return'은 'Menu' 컴포넌트에서 빈번하게(4번) 발생하지만 다른 컴포넌트에서도 빈번히 발생하므로 유용한 인덱싱 단어가 아니다.

표 3 'Menu' Java 컴포넌트의 인덱싱 정보

| 단어 | 소속 클래스 | ICF | TW | TF | NTW |
|---|---|---|---|---|---|
| menu | 13 | 1.5621 | 46.8655 | 30 | 0.1497 |
| tear | 1 | 4.1271 | 28.8899 | 7 | 0.0923 |
| insert | 3 | 3.0285 | 18.1711 | 6 | 0.0580 |
| item | 11 | 1.7292 | 17.2923 | 10 | 0.0552 |
| .. | ... | ... | ... | ... | ... |
| return | 50 | 0.2151 | 0.8604 | 4 | 0.0027 |
| get | 46 | 0.2984 | 0.5969 | 2 | 0.0019 |
| create | 35 | 0.5717 | 0.5717 | 1 | 0.0018 |
| component | 38 | 0.4895 | 0.4895 | 1 | 0.0015 |

### 4.2 컴포넌트 클러스터링

요약내용 자동 생성 서비스를 준비하기 위한 다음 단계로써 개념적으로 관련된 컴포넌트들은 그룹(클러스터)

화 되어야한다. 이를 위해 SOORLS는 통계적 처리 방법에 의해 문서들을 그룹화 시키는 클러스터링 기술의 한 변형을 적용했다. 이렇게 응용된 클러스터링 프로세스에 의해 생성된 클러스터링 구조는 결국 라이브러리의 요약 내용을 생성하는데 이용될 것이고 또한 새로운 컴포넌트들의 자동화된 분류에 이용된다.

클러스터링 기술은 2개의 문서사이의 유사성(similarity)을 계산하여 자동으로 관련된 문서들을 그룹화 하는 것이다. 생성된 클러스터 구조 형태에 따라 클러스터링은 계층적(hierarchical) 그리고 비계층적(non-hierarchical) 방법으로 분류된다[10-11]. SOORLS에 의해 응용된 클러스터링 방법은 계층적 클러스터링 분석과 비계층적 클러스터링 개념을 결합한다. 계층적 방법은 모든 문서 사이의 유사성을 계산하는 것이 필요하다. 계층적 방법에서, 모든 문서들이 하나의 클러스터에 속할 때까지, 문서들을 클러스터들로 점진적인 그룹화를 시도함으로써 2진 트리와 같은 분류구조를 생성한다. 반면에, 비계층 방법은 클러스터링 과정의 초기화로써 클러스터 시드들(cluster seeds)을 이용한다. 이때, 문서들은 이러한 시드들과의 유사성에 따라 그룹 된다.

현재의 디스플레이 장치들은 클러스터링 구조에 따른 요약 내용을 표시하는데 있어 한정된 공간을 갖는다. 이러한 이유 때문에 디스플레이 장치를 고려한 적당한 수의 클러스터들이 결정되어야 한다. 클러스터 수는 디스플레이 장치의 공간의 한계와 사용자가 라이브러리의 전체적인 내용에 대한 일반적인 생각을 줄 수 있을 만큼 충분해야 한다. 또한, 계층적인 방법이 컴퓨터 자원을 많이 소비한다 해도, 비계층적 방법보다 더욱 정확하고 견고한 클러스터 구조를 준비한다는 사실이 다양한 논문에서 언급되었다 [3, 12-13]. 위에 언급된 요소들을 고려할 때, 본 논문은 궁극적으로 라이브러리의 클러스터 시드들을 위한 초기 클러스터들을 결정하기 위해 계층적 방법이 이용된다. 4.1장에서 세워진 컴포넌트 벡터들로부터 컴포넌트들 사이의 유사성이 계산된 후, 계층적 클러스터링 방법이 이용된다. SOORLS에서 이용된 계층적 클러스터링 방법은 SPSS(Statistical Package for Social Science)의 'group-average' 방법이 이용되었다. SPSS 분석 프로그램의 결과는 초기 클러스터들의 수를 결정하는 것을 돕는다. 이 수가 낮다면, 클러스터와 유사성의 정도가 낮은 컴포넌트들이 해당 클러스터에 속할 수 있다. 이런 경우, 선택된 초기 클러스터들은 클러스터 시드에 대한 효과적인 후보일 수 없다. 여러 실험을 통해, 초기 클러스터들의 적절한 수는 라이브러리 컴포넌트 수의 1/4인 것으로 발견되었다. 그러므로, 62개의 AWT 라이브러리 컴포넌트에서 16개의 초기 클러스터들이 선택되었다.

초기 클러스터들로부터 클러스터 시드들을 선택할 때 2가지 요소는 고려되어야 한다. 첫째, SOORLS에서 라이브러리 요약 내용은 클러스터 시드들에 기초해서 생성되기 때문에, 디스플레이 장치의 공간 한계를 고려해야 한다. 둘째, 계층 방법에 의해 생성된 클러스터링 구조는 비계층 방법보다 더욱 안정적이므로, 가능한 한 초기 클러스터에 포함된 컴포넌트들의 구조는 유지 되어야한다. 이러한 2가지 문제를 고려할 때, 초기 클러스터에 포함된 컴포넌트들의 수에 따라 8개의 클러스터 시드들이 선택되었다. 표 4는 위 과정을 통해 결정된 클러스터 시드들을 보인다.

표 4 AWT 라이브러리에 대한 클러스터 시드

| 초기 클러스터 번호 | 포함된 클러스터들의 수 | 클러스터 시드 번호(8) |
|---|---|---|
| 1 | 2 | 없음 |
| 2 | 2 | 없음 |
| 3 | 14 | 1 |
| 4 | 3 | 7 |
| 5 | 9 | 2 |
| 6 | 7 | 3 |
| 7 | 5 | 4 |
| 8 | 1 | 8 |
| 9 | 3 | 없음 |
| 10 | 2 | 없음 |
| 11 | 1 | 없음 |
| 12 | 2 | 없음 |
| 13 | 1 | 없음 |
| 14 | 4 | 5 |
| 15 | 4 | 6 |
| 16 | 2 | 없음 |

표 4에서 초기 클러스터 1, 2, 9, 10, 11, 12, 13, 16에 포함된 15개의 컴포넌트들은 어떤 클러스터 시드에도 포함되지 않는다. AWT 컴포넌트들을 클러스터링 하는 목적은 AWT 라이브러리 전체의 요약 내용을 생산하고 또한 GUI 영역에 대한 기본적인 분류체계를 생성하는 것이기 때문에 15개의 소속이 없는 컴포넌트들이 포함될 클러스터 시드들이 결정되어야 한다. PCP(Post Clustering Processor)는 위에 언급된 비계층 클러스터링 분석 방법을 구현한 도구이다. PCP는 각 클러스터 시드에 포함되는 모든 컴포넌트들을 대표하는 클러스터

센터들을 생성한다. 클러스터 센터는 상응하는 클러스터 시드에 포함된 컴포넌트들과 연관된 단어들의 평균 무게이다. 이때, 클러스터 시드에 소속되지 않은 컴포넌트들은 각 클러스터 시드와 유사성의 정도가 가장 높은 시드에 배정된다. 어떤 컴포넌트가 시드에 배정된 후, PCP는 해당 컴포넌트의 포함을 반영하기 위해 다시 계산된다. 표 5는 이러한 과정을 통해 모든 AWT 컴포넌트들이 각 클러스터 시드에 포함된 후의 결과를 보인다.

표 5 모든 AWT 컴포넌트들에 대한 클러스터 시드 배정

| 클러스터 | 컴포넌트(총수) |
|---|---|
| 1 | Checkbox, CheckboxGroup, CheckboxMenuItem, Choice, ItemSelectable, List, MenuBar, MenuComponent, MenuContainer, MenuItem, Menu Shortcut, PopupMenu, Menu (13) |
| 2 | BorderLayout, CardLayout, FlowLayout, GridBag Constraints, GridBagLayout, GridLayout, Insets, LayoutManager, LayoutManager2, Panel (10) |
| 3 | AWTError, AWTEvent, AWTEventMulticaster, Component, Container, Event, EventDispatch Thread, EventQueue, MediaTracker (9) |
| 4 | Color, Graphics, Image, PrintGraphics, PrintJob, System Color, Toolkit (7) |
| 5 | Button, Label, TextArea, TextComponent, Text Field (5) |
| 6 | AWTException, Canvas, Cursor, Dialog, FileDialog, Frame, IllegalComponentStateException, Window (8) |
| 7 | Adjustable, Polygon, Scrollbar, ScrollPane, Shape (5) |
| 8 | Dimension, Font, FontMetrics, Point, Rectangle (5) |

SH(Summary Handler)는 라이브러리 요약 내용을 생성하는 도구이다. 이 프로그램은 각 클러스터 센터로부터 사용자가 입력한 값을 초과하는 무게 값을 갖는 단어들을 추출함으로써 요약내용을 생성한다. 결과로써, SOORLS는 8개의 미리 정의된 클러스터 시드들에 상응하는 8개의 그룹들로부터 요약내용을 보일 수 있다. 그림 2는 0.03의 디폴트 무게 값에 의해 추출된 라이브러리 내용을 보인다.

SOORLS의 요약내용 서비스가 어떻게 사용자들을 도울 수 있는지 보이기 위해 그림2의 내용은 사용자에 의해 다음과 같이 해석될 수 있다.

- cluster1: 클러스터1에 포함된 컴포넌트들은 'menus'와 관련된다. 이러한 컴포넌트들은 'checkbox'와 같이 사용자가 여러 'items'들 중 선택할 수 있는 옵션들을 다룬다. -> menu 관련 클러스터.
- cluster2: 클러스터2에 포함된 컴포넌트들은 'container'에 컴포넌트들의 'lay out'과 관련된다. 이러한 컴포넌트들은 스크린의 'left', 'right', 'bottom' 위치에 컴포넌트들을 놓는 것을 가능하게 한다. -> layout 관련 클러스터.
- cluster3: 클러스터3에 포함된 컴포넌트들은 'AWT'로부터 일어나는 'errors' 그리고 'images'의 'loading'과 관련된다. -> image 관련 클러스터.
- cluster4: 클러스터4에 포함된 컴포넌트들은 'graphics', 'printed page'의 페인팅, 스크린 'resolution'의 검색과 관련된다. -> graphics 관련 클러스터.
- cluster5: 클러스터5에 포함된 컴포넌트들은 'buttons', 'text fields', 'text areas', 'labels'와 같은 사용자 인터페이스와 관련된다. -> UI 관련 클러스터.
- cluster6: 클러스터6에 포함된 컴포넌트들은 Java 프로그램이 AWT에서 'illegal' 동작을 만날 때 'exceptions'의 'signaling'과 관련된다. -> exception 관련 클러스터.
- cluster7: 클러스터7에 포함된 컴포넌트들은 'polygon'과 같은 기하학적인 형태와 관련된다. -> shape 관련 클러스터.
- cluster8: 클러스터8에 포함된 컴포넌트들은 'characters'의 'width'을 결정하는 'fonts'들과 관련된다. -> font 관련 클러스터.

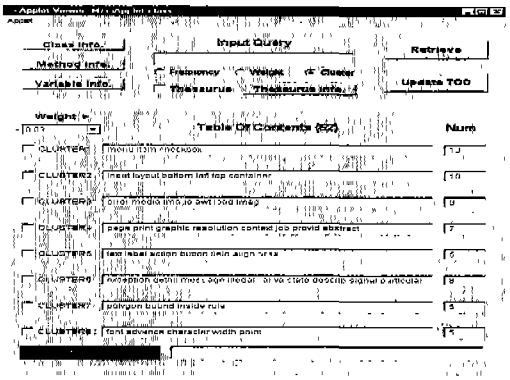

그림 2 AWT 라이브러리 요약 내용

그러나, 사용자들의 이러한 해석은 잘못될 수 있다. 보기를 들면, 클러스터3에 포함되는 대부분의 컴포넌트들은 AWT 컴포넌트들 사이에서 발생하는 이벤트들과 관련된다. 또한, 클러스터6에 포함되는 대부분의 컴포넌트들은 drop-down 메뉴들을 준비하는 윈도우와 관련된

다. 이러한 잘못된 해석은 SOORLS의 요약내용 자동생성 서비스가 컴포넌트 개발자가 준비한 문서로부터 추출된 정보에 의존하기 때문이다. 컴포넌트와 관련된 문서가 해당 컴포넌트를 정확히(충분히) 정의(묘사)되지 않는다면 사용자는 검색 초기에 라이브러리 내용을 이해하는데 어려움이 있을 것이다. 이러한 잘못된 해석을 방지하기 위해 SOORLS는 사용자와 상호 작용을 통해 더욱 상세한 라이브러리 내용을 볼 수 있는 능력을 준비한다. 예를들면, 사용자는 0.03보다 약간 작은 무게 값을 선택할 수 있다. SOORLS는 각 클러스터에서 더욱 상세한 단어들의 리스트를 보인다. 결과로써, 클러스터3과 4를 묘사하는 전형적인 단어들(예, 'window', 'event')이 요약내용에 더해진다. 사용자와의 상호작용에 의한 요약내용의 조절은 각 클러스터들을 올바르게 해석하는 것을 돕는다. 반면, 사용자가 빠른 시간에 가장 효과적으로 각 클러스터를 대표하는 단어들에 의해 라이브러리 내용을 조사하기 원한다면 높은 무게 값을 선택할 수 있다. 따라서 사용자들이 해당 라이브러리 내용을 더욱 잘 이해하기 위해 높고 낮은 무게 값을 번갈아 가며 선택하는 것이 필요하다.

## 5. 이해기반 검색 서비스

SOORLS에 의해 준비되는 이해기반 검색 서비스는 사용자의 다양한 질의 요청을 만족할 수 있다. 보기를 들면, 다음 4가지 질의 요청에 따라 후보 컴포넌트들을 검색할 수 있다.

- 요청1: 사용자는 어떤 클러스터도 선택하지 않고 질의도 입력하지 않는다.
- 요청2: 사용자는 어떤 클러스터도 선택하지 않고 질의를 입력한다.
- 요청3: 사용자는 약간의 클러스터들을 선택하고 질의를 입력하지 않는다.
- 요청4: 사용자는 약간의 클러스터들을 선택하고 질의를 입력한다.

요청1과 3에서, 사용자는 검색되어질 후보 컴포넌트들에 대한 특별한 요구 조건을 갖지 않는다. SOORLS는 이런 경우에도 라이브러리에 포함된 모든 컴포넌트들 혹은 선택된 클러스터들에 포함된 컴포넌트들의 순서화된 리스트를 표시한다. 이러한 요청에 대한 질의를 지원하기 위해 SOORLS는 모든(혹은 선택된) 클러스터들을 대표하는 일시적인 클러스터를 준비한다. 그때, 후보 컴포넌트들은 일시적인 클러스터 벡터와 모든 컴포넌트들 (혹은, 선택된 클러스터의 모든 컴포넌트) 사이의 유사성 정도에 따라 검색될 수 있다. 요청2 와 4는 사용자가 특별한 요구 조건을 갖는다. 질의는 자연어로 입력된다. SOORLS는 이러한 자연어 질의를 컴포넌트로써 처리한다. 그러므로, 4장에서 언급된 Indexer는 자연어로 구성된 질의를 처리하기 위해 이용되며, 이 결과에 의해 생성되는 질의에 해당하는 컴포넌트 벡터는 일시적인 클러스터 벡터와의 유사성 정도에 따라 후보 컴포넌트들은 검색된다.

요청1과 2의 경우, 모든 컴포넌트들은 잠재적인 후보 컴포넌트들이 된다. 이때, 요청1은 라이브러리를 표현하는 일시적인 클러스터와 각 컴포넌트 사이의 유사 정도에 따라 결과를 표시한다. 요청2는 우선적으로 입력 질의와 충분히 높은 유사성 값(=0.05)을 갖는 클러스터들을 구별한다. 입력 질의와 구별된 클러스터들에 포함된 컴포넌트들 사이의 유사 정도에 따라 결과를 표시한다. 요청 3과 4의 경우, 사용자는 흥미 있는 클러스터들로 탐색 공간을 한정시킨다. 이때, 요청3은 선택된 클러스터들을 표현하는 일시적인 클러스터와 선택된 클러스터에 포함된 각 컴포넌트 사이의 유사 정도에 따라 결과를 표시한다. 요청4는 입력 질의와 선택된 클러스터에 포함된 컴포넌트들 사이의 유사 정도에 따라 결과를 표시한다.

## 6. 성능 평가

SOORLS와 같은 라이브러리 시스템의 성능은 정보과학(Information Science)분야에서 라이브러리 시스템들의 성능을 평가할 때 빈번히 이용되는 다음과 같은 4가지 기준(criteria)에 의해 평가되었다: 사용자노력(user effort), 응답시간(response time), 회상도(recall), 정확도(precision)[14].

### 6.1 사용자 노력

라이브러리 시스템에서 사용자들이 얼마나 쉽게 질의를 구성할 수 있는가 혹은 라이브러리 시스템을 이용하기 위해 얼마나 많은 노력이 필요한 가를 판단함으로써 시스템의 성능을 평가할 수 있다. SOORLS는 자연어에 의해 질의를 구성할 수 있는 단순한 방법을 사용자에게 제공한다. 또한 사용자는 질의 생성에서 SOORLS 라이브러리의 전체적인 요약 내용으로부터 직접 단어들을 이용할 수 있고, 라이브러리 내용을 이해함으로써 간접적으로 추론된 새로운 단어들에 의해 질의를 구성할 수 있는 사용자 친화적인 환경을 준비한다.

### 6.2 응답시간

SOORLS의 응답시간은 사용자가 질의를 입력한 후 그림 1의 'retrieve' 버튼을 클릭 했을 때부터 시스템이

스크린에 후보 컴포넌트들을 보일 때까지의 경과된 시간으로 정의된다. 표 6은 5장에서 설명된 다양한 사용자 검색 요청에 따른 각각의 응답시간을 보인다. 테스팅 환경으로써, 웹브라우저, RMI서버, 그리고 데이타베이스서버는 한 컴퓨터(Pentium 200Mhz)에서 실행되었다.

표 6 사용자 요청에 의한 응답시간

| 질의입력 | 선택클러스터 | 응답시간(초) | 평균응답시간(초) |
|---|---|---|---|
| 없음 | 없음 | 36.426 | 36.426 |
| 있음 | 1, 2 | 20.476 | 19.815 |
| | 7, 8 | 16.987 | |
| | 3, 4 | 21.982 | |
| 없음 | 없음 | 3.901 | 3.901 |
| 있음 | 1, 2 | 2.230 | 1.828 |
| | 7, 8 | 1.515 | |
| | 3, 4 | 1.739 | |

표 6에서 사용자가 검색하기 위한 흥미 있는 그룹이 존재하는 경우의 테스트 케이스로써, 다수의 컴포넌트들을 포함하는 그룹으로 1, 2 클러스터, 소수의 컴포넌트들을 포함하는 7, 8 클러스터, 중간 크기 그룹으로 3, 4 클러스터를 선택했다. 사용자가 질의를 입력하지 않은 경우 SOORLS는 상당히 많은 응답시간을 필요로 한다(예, 36.426, 19.815). SOORLS의 독특한 이해 기반 검색 메커니즘은 임시 클러스터들의 계산에 많은 시간을 요구한다. 그러므로 특별한 질의 요구가 없는 사용자에게 순차적인 후보 컴포넌트들을 제공하는 기능을 이용하기 위해 사용자는 더욱 많은 시간을 기다리는 것이 필요하다. 반면, 사용자가 질의를 입력한 경우 SOORLS는 적당한 응답시간을 준다. 다른 경우로써, 사용자가 흥미 있는 클러스터들을 선택한 경우와 그렇지 않은 경우에 응답시간은 거의 2배 차이가 난다(예, 36.426: 19.815, 3.901:1.828). 이러한 차이는 클러스터들을 선택함으로써 탐색공간을 한정하기 때문이다.

### 6.3 회상도 와 정확도

라이브러리 시스템은 사용자 질의에 대한 결과로써 잠재적인 후보 컴포넌트들을 보일 때 효과적이어야 한다. 시스템의 효과를 측정하기 위해 라이브러리 시스템들의 탐색 효과를 평가할 때 자주 사용되는 회상도 와 정확도를 이용했다. 회상도는 라이브러리에 존재하는 질의와 관련된 모든 컴포넌트들 중 검색된 질의와 관련된 컴포넌트들의 비율로써 정의될 수 있다. 반면, 정확도는 검색된 컴포넌트들 중 질의와 관련된 컴포넌트들의 비율로써 정의된다. 이들의 측정을 위해, 우선 각 컴포넌트가 입력 질의와 관련되는지의 여부가 결정되어야 한다. 이러한 판단은 해당 영역의 전문가에 의해 수행함으로써 가능한 한 객관적이어야 한다. 어떤 질의에 대한 각 컴포넌트의 관련 여부와 이 질의에 대한 결과 컴포넌트들이 주어졌을 때, 우리는 다음과 같이 회상도와 정확도를 측정할 수 있다.

- 회상도 = 검색된 컴포넌트들 사이에서 질의 관련 컴포넌트들의 수 / 해당 라이브러리에서 질의 관련 컴포넌트들의 수
- 정확도 = 검색된 컴포넌트들 사이에서 질의 관련 컴포넌트들의 수 / 검색된 컴포넌트들의 수

회상도와 정확도 사이에는 역관계가 존재한다. 즉, 시스템의 회상도를 높이기 위해 노력할 때 정확도는 낮아지고, 정확도를 높일 때 회상도는 낮아진다. 그러므로, 이상적인 라이브러리 시스템은 회상도와 정확도를 동시에 최대화하기 위해 노력해야 한다. 또한 우리는 어떤 라이브러리 시스템의 회상도와 정확도의 측정은 절대적이 아니고 상대적이라는 것을 고려해야 한다. 즉, SOORLS의 성능을 다른 라이브러리 시스템들과 비교하기 위해 그들은 같은 환경(같은 라이브러리 와 같은 질의 환경)에서 비교되어야 한다. 본 논문은 이러한 환경을 만족시키기 위해 두 개의 실험적인 라이브러리 시스템들이 구현되었다: FOORLS(Frequency-based OORLS)와 WOORLS (Weight-based OORLS).

SOORLS를 포함하는 3개의 시스템들은 단지 그들의 검색 방법에서 차이를 가진다. FLOORS는 입력 질의와 컴포넌트에서 나타나는 인덱싱 단어의 매칭 빈도 수에 기초한 단순한 라이브러리 시스템이다. WOORLS는 SOORLS와 같은 인덱싱 과정을 통하여 계산된 컴포넌트 인덱싱 단어의 무게 값과 입력 질의에서의 단어들의 매칭 결과에 따라 후보 컴포넌트들을 보이는 라이브러리 시스템이다. 이에 반해, SOORLS는 이해 기반 검색 방법을 제공한다. 본 논문에서 질의와 관련된 컴포넌트들에 대한 정의는 해당 질의에 상응하는 Java 응용 프로그램을 위해 이용되는 컴포넌트들로써 결정되었다. 이런 상황에서 질의와 관련된 컴포넌트들은 Java AWT 패키지 이외의 다른 패키지 컴포넌트들을 포함할 것이다. 현재, SOORLS 라이브러리는 AWT 패키지 컴포넌트들로 구성되므로 우리는 AWT 이외의 패키지에 포함된 컴포넌트들은 고려하지 않는다. SOORLS 성능 평가에 대한 합법적인 테스팅 환경을 위해 자연어로 표현된 5개의 질의가 상응하는 응용 프로그램들에 대한 사용자 요구로써 입력되었다. 각각의 질의에 대한 회상도-정확도 그래프는 각 시스템의 평균 성능을 반영하는 하나의

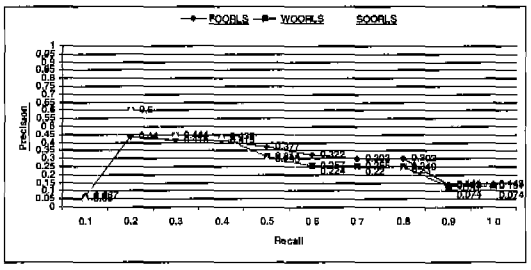


그림 3 평균 회상도-정확도 그래프

그래프를 생성하기 위해 결합되었다.

그림 3은 5개의 질의가 3개의 라이브러리 시스템들에 각각 입력되었을 때, 고정된 회상도 값(0.1, 0.2, ..., 1.0)에 따른 평균 정확도 값을 보인다. 그림3에서, SOORLS는 회상도 0.4이상에서 WOORLS와 성능 면에서 거의 같은 반면, FOORLS보다 더욱 좋은 성능을 보인다. 반면, 회상도 0.4이하에서 SOORLS는 다른 2개의 시스템보다 더욱 나쁜 성능을 보인다. 이러한 결과에 따라, SOORLS는 사용자가 높은 정확도를 요구하는 경우에 더욱 좋은 반면, 다른 두 시스템들은 높은 회상도에 흥미가 있는 사용자에게 더욱 좋은 결과를 준다. 회상도가 증가할 때 SOORLS의 나쁜 성능은 SOORLS의 독특한 특성(탐색 영역을 제한하는)에 기인한다. FOORLS 와 WOORLS가 성능 면에서 SOORLS보다 약간 높다 할지라도 SOORLS에서 제공되는 라이브러리 이해 기반의 검색 기능과 같은 사용자의 심리적인 요소가 고려되어야 한다.

## 7. 결론과 향후 연구과제

본 논문은 전통적인 재사용 라이브러리 시스템들의 단순한 검색 서비스 이상의 가치 있는 서비스들의 제공은 물론, 효과적인 라이브러리 관리를 위한 웹 라이브러리 재사용 시스템을 보였다. 특히 사용자들에게 흥미 있는 독특한 서비스로써 라이브러리 내용의 자동 생성, 이해 기반의 검색 서비스는 본 논문에서 제안된 계층 클러스터링 방법과 비계층 클러스터링 방법의 응용에 의해 가능했다. 이러한 서비스들 외에 사용자가 입력 질의 구성을 위한 적절한 어휘를 생각하지 못할 때 이 시스템은 사용자에게 요약 내용을 보임으로써 질의 생성을 도울 수 있다. 또한 제안된 시스템은 전통적인 패싯 기반의 라이브러리 시스템들에서, 영역 전문가가 분류체계 구현을 위해 주어진 영역을 분석하는 것을 도울 수 있다.

본 논문에서 제안된 시스템은 다른 검색 메커니즘을 사용하는 전통적인 라이브러리 시스템들 사이의 회상도-정확도에 의한 성능이 평가되었다. 이러한 성능 평가가 단순한 사용자노력, 응답시간, 회상도, 정확도에 의해 평가되었다 해도 웹에 존재하는 수많은 재사용 컴포넌트들로 구성된 라이브러리의 전체적인 내용의 이해로부터 파생되는 효과적인 질의 생성, 사용자의 다양한 요구 만족과 같은 사용자의 심리적인 요소를 고려하는 객관적인 평가 방법이 개발되어야 한다. 또한, 제안된 시스템이 완전한 라이브러리 시스템이 되기 위해 컴포넌트들의 갱신과 변화를 사용자에게 통지하기 위해 이용할 수 있는 SCM(Software Configuration Management) 서비스들이 구현되어야 한다.

## 참 고 문 헌

[1] Hooper, J. W. and Chester, R. O., *Software Reuse: Guidelines and Methods*, Plenum Press, N.Y., 1991.

[2] Sutcliffe, A., *Software Reuse: State of the Art and Survey of Technical Approaches* (ed. P. Walton and N. Maiden), UNICOM Seminars Ltd., pp. 51-75, 1993.

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

[4] Salton, G., *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley Publishing Company, 1989.

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

[6] Sindre, G., Conradi, R. and Karlsson, A., The REBOOT Approach to Software Reuse, *Journal of Systems Software*, Vol. 30, No. 3, pp. 201-212, Sep. 1995.

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

[8] Lee, S. K. and Urban, J. E., "SOORLS: A Software Reuse Approach On The Web," International Journal of Software Engineering and Knowledge Engineering, Vol. 9, No. 3, pp. 279-296, 1999.

[9] Lee, D. L., Chuang, H. and Seamons, K., "Document Ranking and the Vector-Space Model," *IEEE Software*, pp. 67-75, March/April 1997.

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

[11] Jain, A. K. and Dubes, R. C., *Algorithms for Clustering Data*, Prentice Hall, 1988.

[12] Salton, G. and Buckley, C., "Term-Weighting Approaches in Automatic Text Retrieval," *Information and Processing and Management*, Vol. 24, No. 5, pp. 513-523, 1988.
[13] Willett, P., Recent Trends in Hierarchic Document Clustering: A Critical Review, *Information Processing & Management*, Vol. 24, No. 5, pp. 577-597, 1988.
[14] Salton, G. and McGill, J., *Introduction to Modern Information Retrieval*, McGraw-Hill Book Company, 1983.
[15] H. Mili, F. Mili, and A. Mili, Reusing Software: Issues and Research Directions, *IEEE Transactions on Software Engineering*, Vol. 21, No. 6, pp. 528-561, June 1995.
[16] 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.

이 성 구

1980년 중앙대학교 전자계산학과 학사. 1989년 중앙대학교 전자계산학과 석사. 1993년 Arizona 주립대학 전자계산학과 석사. 1998년 Arizona 주립대학 전자계산학과 박사. 1999년 ~ 현재 한신대학교 정보시스템 공학과 조교수. 관심분야는 소프트웨어 공학, 인터넷 프로그래밍, 및 객체지향 언어