한국정보전자통신기술학회논문지(jkiiect)'16-02, Vol.9 No.1
논문 15-09-01-001 http://dx.doi.org/10.17661/jkiiect.2016.9.1.001

소프트웨어 재사용을 위한 정보검색시스템 구축

김영길*

The Information Retrieval System for Software Reuse

Young-Geil Kim*

요 약 본 논문에서는 전반적으로 소프트웨어 재사용과정에서 지적되고 있는 문제점을 재사용 단계별로 정리하고, 지적되어온 문제점 중 라이브러리 구축과정에서 효과적인 부품의 인덱싱과 기능적으로 관련된 부품들끼리의 분류로 검색의 성능 증진을 꾀할 수 있는 방법을 제안한다. 객체지향 소프트웨어 라이브러리를 구성하는 부품은 클래스가 가지고 이있는 객체지향 모델의 특성과 클래스가 지니고 있는 책임으로 기능을 모두 고려하여 객체지향 라이브러리를 구축하고 객체지향 지향 라이브러리는 상속구조를 바탕으로 라이브러리내의 계층구조를 형성한다. 그러나 이러한 계층구조는 구문적인 정보를 바탕으로 이루어지는 것으로 클래스 부품의 기능성과 일치하지 않을 수도 있다. 이를 해결하기 위하여 본 논문에서는 클래스 계층구조와 같은 지기기반 접근법과 클래스 부품의 기능성에 기반을 둔 소프트웨어 부품의 인덱싱과 분류에 필요한 정보검색 방법을 혼용한 라이브러리 과닐 기법을 제안한다. 특히 본 논문에서는 최근에 많은 분야에 걸쳐 연구가 진행 중인 객체지향 방법론을 바탕으로 개발된 소프트웨어 라이브러리의 기능 향상을 위한 구조와 구축 방법을 제안하고 지원 시스템을 개발한다.

Abstract In this paper, several problems functioning as the obstacles against software reuse were summarized. Among them, the issues dealt with in this paper include the effective method for constructing the library, the proper structure of the library, and the efficient retrieval technique. The knowledge-based approach and the information retrieval approach were integrated to construct and manage the library. The former is on the object- oriented model. Basically the object-oriented library is based on the classes and organized by inheritance. Because inheritance hierarchy is based on syntactical information, it dose not present the relationship of functionality. Using the information retrieval approach, the index file which characterizes the component and similarity among the components can be analyzed. Especially, we focused on the reusable library for the object-oriented programming environments.

Keywords : software reuse, indexing, software library, component, retrieval system

1. 서 론

사물 인터넷 시대를 맞이하여 다양한 형태의 소프트웨어 재사용이 필한 시대에 와있다. 이러한 시대에 소프트웨어 재사용이 소프트웨어 생산성과 신뢰도를 증진하는 중요한 가장 확실한 방법으로 등장하면서 소프트웨어 공학의 중요한 관심사로 대두되었다. 이러한 현상의 원인으로는 재사용 가능한 소프트웨어 부품들을 가지고 있는 라이브러리가 충분하지 못하다는 점이 가장 큰 요인으로 지적되었다.[1-3] 또한 기존의 라이브러리들에 있어서는 재사용 가능한 부품들을 저장하고 이해하는데 필요한 기능들의 지원이 부족한 것도 하나의 원인이 된다.

부품이라고 하는 것은 소프트웨어 재사용을 위하여 필요한 것들을 의미하는 것으로 예를 들면 함

This paper received the fund of Korea National University of Transportation in 2015.
* Corresponding Author : Department of Computer Information Engineering, Korea National University of Transportation (ygkim@ut.ac.kr)
Received January 8, 2016 Revised January 15, 2016 Accepted January 22, 2016

수 라이브러리 같은 것들이 부품에 속한다.

소프트웨어 재사용 라이브러리가 재사용에 적합하려면 다음과 같은 점들을 충족 시켜야 한다

첫째, 다양한 도메인에 걸쳐 충분한 양의 부품들을 지님으로써 이들 부품이 기능의 단순한 재사용뿐 아니라 간단한 수정을 통한 재사용도 지원해야 한다.

둘째, 효과적인 부품의 인덱싱과 기능적으로 관련된 부품을 관리하여 사용자의 요구에 가장 근접한 부품들을 쉽게 찾도록 관리되어야 한다. 기능적으로 상호 관련성이 있는 부품의 제공은 검색 기능을 개선하며, 이해 과정을 직접 지원한다. 상호 관련된 부품이라면 하나의 부품에 대한 이해가 다음 부품의 이해에 기반이 되기 때문이다.

셋째, 효과적인 부품의 관리를 위하여 필요한 것이 분류 방법이다. 부품의 분류는 유사 정도를 측정하여 모든 부품들을 분류하여 저장을 해야 재사용이 원활하게 이루어질 수 있다.

2. 정보검색방법에 의한 부품의 인덱싱 과정

인덱싱 작업의 첫 단계는 부품의 기능을 식별할 수 있는 인덱스 항을 할당/선정하는 것이고, 다음으로는 기능을 식별하는 데 있어서 중요도에 따라 가중치를 결정하는 것이다.[4]

2.1. 인덱싱 항목의 추출

부품의 특성을 대표할 수 있는 인덱스 항목의 추출을 위해 부품의 기능을 기술한 문서와 주석의 구문적 분석을 통해 각 인덱스 항을 결정하고 빈도수를 파악한다. 객체지향 라이브러리를 구성하는 클래스 부품은 그 고유의 특성에 의해 자료 부분과 자료 관리에 필요한 오퍼레이션 부분으로 구성되므로 클래스를 특성 짓는 인덱스를 그림1 과 같이 정의한다.

Compi=(Domain-indexi , Operation-indexi)

Compi : 임의의 클러스터에 저장된 부품 I의 프로화일명

Domain-indexi = ($Term_1$, $Term_2$, ... , $Term_n$)

Operation-indexi =($Term_1$, $Term_2$, ... , $Term_t$)

$Term_i$: 인덱스 항

그림 1. 클래스를 정의하는 인덱스 형태
Fig. 1. Index Form to define Class

각 인덱스 항은 문서의 구문적 분석을 통해서 의미 있는 단어, 즉, 명사, 동사, 형용사만을 추출하고 전치사, 접속사, 조사 등이 의미 없는 단어는 구분은 통해 추출된다. 또한 인덱스는 단어의 원형을 취하여 추출한다.

각 인덱스 별로 클러스터 전체에 걸쳐 나타난 빈도수 Fk 는 그림 2 와 같이 얻는다.

$$F_k = \sum_{i=1}^{n} f_{ik}$$

F_k : 하나의 클러스터 내에서 인덱스 k의 빈도수

f_{ik} : 부품 I에서 인덱스 k의 빈도수

n : 클러스터 내의 부품 개수

그림 2. 인덱스의 전체 빈도수
Fig. 2 Total Frequency Number of Index

이렇게 하여 그림 1에 따라 인덱스를 결정하였다. 추출된 인덱스와 빈도수를 표현하고 저장하며, 추후 분류와 검색의 과정에 필요한 자료로 관리하기 위하여 인덱스들의 적절한 표현 방법이 필요하다. 본 논문에서는 이를 벡터 모델(vector model)로 표현한다.

2.2. 가중치 할당

부품에서 추출된 인덱스 항들을 모으면 이러한 인덱스들이 부품이 가지고 있는 특성을 나타내기 때문에 부품의 기능을 대표하게 된다. 기존의 문헌 검색 시스템은 항의 중요도에 따라 항에 가중치를 부여하지 않고, 항이 부품에 해당하는 것인지 아닌지에 대한 사항만을 결정할 뿐이다. 그림 3과 같이 정보에 속할 경우의 가중치는 1이고, 그렇지 않은 경우는 0으로 간주되었다.[5]

$$W_{ik} = \begin{cases} 1 & \text{if } f_{ik} > 0 \\ 0 & \text{otherwise} \end{cases}$$

W_{ik}: 부품 i에서 인덱스 k의 가중치

그림 3. 이진 검색에서의 가중치
Fig. 3 Weight Value in Binary Tree

이를 보완하기 위한 방법으로 가장 간단한 방법은 그림 4와 같이 임의의 부품을 특성 짓는 부품의 인덱스 항들의 빈도수를 인덱스의 가중치로 파악하는 것이다.

$$W_{ik} = f_{ik}$$

그림. 4. 인덱스의 가중치를 구하는 방법
Fig. 4. Method to Seek for Weight Value of Index

이런 현상을 줄이기 위해 또 다른 요소로 한 인덱스가 속한 부품의 개수를 고려하여 그림 5와 같은 방법으로 부품의 개수를 측정하고, 각 인덱스의 가중치는 그림 6과 같이 구한다.

$$C_k = \sum_{i}^{n} c_{ik}$$

C_k: 인덱스 k가 나타나는 클러스터내 부품의 갯수

$$C_{ik} = \begin{cases} 1 & \text{if } f_{ik} > 0 \\ 0 & \text{otherwis} \end{cases}$$

n : 클러스터를 구성하는 부품의 개수

그림 5. 인덱스가 속한 부품의 개수 측정 방법
Fig. 5. Method to Measure Number of Component in Index

$$W_{ik} = F_{ik}/C_i$$

그림 6.인덱스의가중치를 구하는 보완된 방법
Fig. 6. Weight Value of Index

이는 높은 가중치를 갖는 항은 어느 특정 부품에서 높은 등장횟수를 갖지만 전체 클러스터에서의 빈도는 낮게 나타난다. 높은 등장횟수를 갖는 항목은 그 부품에서의 높은 중요도를 의미하는 반면, 낮은 전체 빈도나 문서빈도는 문서 집합에 속하는 모든 문서에서 동일한 중요도를 갖는다는 것을 의미한다. 다른 문서에서 와의 중요도가 작아서 그 항은 해당 문서를 대표하는 항으로 간주할 수 있다.

이와 같이 만들어진 각 인덱스 항에 대한 가중치를 바탕으로 인덱스를 표현한 벡터 모델의 내용이 변경된다.

3. 객체지향 환경에서 재사용 모델

기존의 부품 재사용 모델에는 함수집합체[6], 모듈결합[7], Diaz에 의한 부품 모델이[8] 있으나, 본 논문에서는 다음과 같은 모델을 재사용 모델로 제안하는데 기존의 연구에서는 질의어 습득을 위해서 특별한 노력을 필요로 하지만 본 논문에서는 재사용자의 경험과 지식에 따른 질의용어의 자유로운 선정이 가능한 자연어를 사용하므로 질의를 위한 특별한 노력이 필요 없다는 장점을 가지고 있다

3.1. 재사용 모델

객체지향에서의 재사용 역시 기능 중심 환경에서의 재사용과 비슷한 점을 지니고 있다. 즉, 라이브러리에 저장된 함수를 새로 구성하는 응용 프로그램에서 호출하여 사용하는 형태 역시 객체지향 환경에서도 가능하다. 그러나 객체지향 환경에서의 재사용은 이와 다른 새로운 방법의 재사용을 제공하고 있다. 이는 크게 두 가지로 나누어 볼 수 있다.

첫 번째, 객체 생성에 의한 재사용으로 동일한 속성과 행위를 갖는 객체의 생성을 통해 클래스를

재사용하게 되며, 이는 클래스들 간의 컴포넌트 관계에 서로 동일한 재사용 형태를 갖게 된다.
그림 7은 컴포넌트 관계에 의한 재사용을 보여 주고 있다.

```
#include <lib/Company/employee.h>
#include <lib/Company/project.h>
class Department
{
        private :
        char department_name[20];
        Employee OurEmp[100];
        Project OurProject[10];
        }
```

그림 7. 클래스의 속성으로서 이미 정의된 클래스의 재사용
Fig. 7. Reuse of Class Predefined with Attribute of Class

두 번째 방법으로는 상속에 의한 재사용 구조로써 이는 모델링 단계의 동일 속성을 모아 상위 클래스를 만드는 일반화/상세화를 통한 재사용의 관계나 프로그래밍 단계에서 상속 구조를 통해 상위 클래스의 구현 사항을 재사용하는 방법이다. 이 경우 상위 클래스의 단순한 인터페이스의 재사용과 상위 클래스의 확장을 위한 구현 사항의 재사용으로 구분하여 볼 수 있다.

기존의 연구에서는 객체지향 라이브러리의 분류는 상속 구조에 기반을 둔 계층구조를 가지고 있으나 본 연구에서는 객체지향 모델을 기반으로 하는 라이브러리 구조에 대한 지식 기반과 함께 부품의 기능성에 관한 유용한 재사용 정보를 문서로부터 추출하기 위해 정보검색 기법을 적용하여 소프트웨어 부품을 자동으로 분류, 저장, 검색이 가능하도록 하였다.

3.2. 부품 모델과 재사용 모델을 기반으로 한 지식 표현

모델에 따른 컴포넌트의 재사용 정보를 지식으로 표현하는 방법을 제시한 CRL(Component Representation Language)은 C++ 원시 코드로 부터 컴포넌트 표현에 필요한 추출 정보를 입력으로, 컴포넌트 표현 파일을 생성하는 기본이 된다. CRL의 기본 문법은 그림 8와 같다.

```
#representation
#cluster Cluster-name
#class Class-name
  inherit : Superclass-name
            componen:
            ComponentClass-name
  depend : ServerClass-name
  friend : FriendClass-name
#method Method-name
#end cluster
#end representation
```

그림 8. CRL의 기본 문법
Fig. 8. Basic Grammar of CRL

이를 BNF(Backus-Naur Form)형태로 정형화하면 그림 9와 같다.

컴포넌트를 표현하는데 있어서 각 구성원 사이를 구분 짓기 위하여 그림 10에 정의된 특수 문자를 사용한다.

```
# :  컴포넌트 어트리뷰트 구분
:  :컴포넌트 지시자와 기술의 구분
,  :  기술 항목들을 구분
```

그림 10. CRL에 정의된 구분자
Fig. 10. Separator Defined in CRL

```
<CRL> ::= <REP><CLU_REP>
<REP> ::= #representation
<CLU_REP> ::=<CLUS><EN_REP>
<CLUS> ::= <CLU>CLU_body>
<CLU> ::= #cluster<NAME>
<CLU_body> ::= <CLA><EN_CLU>
<CLA> ::=<CLA_body><EN_CLA>
<CLA_body> ::= <CLA_NA><RELA_MET>
<CLA_NA> ::= <RELATED><METHOD>
<RELA_MET> ::= <RELATED><METHOD>
<RELATED> ::= {inherit:(<NAME><)|
                component:(<NAME>,)|
                depend:(<NAME>,)|
                document:(<NAME>,)|
                friend:(<NAME>,)}
<METHOD>::= #method<NAME>
<EN_CLA> ::= #end class
<EN_CLU> ::= #end cluster
<EN_REP> ::= #end representation
<NAME>  ::= a|b|…|y|z|A|B|…|Y|Z|_
```

그림 9. CRL의 BNF 기술형식
Fig. 9. BNF Form of CRL

CRL의 시작과 끝, 라이브러리를 구성하는 클러스터의 구분과 각 클러스터에 속한 부품에 관한 정보의 시작과 끝을 알리는 지시자로 그림 11에 정의된 예약어를 사용한다

```
#representation  : CRL정의의 머리부 표기
#cluster  : 도메인의 머리부 표기
#end cluster : 도메인의 끝을 표기
#class : 클래스의 머리부 표기
#end class : 클래스의 끝을 표기
#method : 메소드의 머리부 표기
#end representation : CRL화일의 끝을 표기
```

그림 11. CRL의 지시자로 정의된 예약어
Fig. 11. Reserved Word defined with Indicator of CRL

3.3. 적용 예

III장에서 제시한 컴포넌트 모델과 객체지향 라이브러리 구조에 따른 컴포넌트 표현의 예제로 윈도우 도메인에 속한 window.cpp를 CRL로 기술한다.

윈도우 도메인의 Browser 클래스는 Buffer Window를 상속받고 있으며, Browser 클래스에 정의된 메소드중의 하나인 dispatcher()를 구현하는데 있어서 Keystroke이라는 클래스를 depend-on의 관계로서 재사용하고 있다. 이를 CRL에 따라 기술한 예는 그림 12과 같다.

윈도우 도메인에 대한 부품 표현 화일은 분석기에 의하여, 제시한 컴포넌트 모델과 CRL의 구문에 의하여 생성하게 된다.

```
#representation
#cluster window
#class browser
        inherit : BufferWindow
        component :
        depend : keystrok
        friend :
#method dispatcher
#method process_층
#method up
#method down
#method page_up
#method page_down
#method top_of_buffer
#method bottom_of_buffer
#method redisplay_window
#end class
```

그림 12. 컴포넌트 표현화일
Fig. 12. Represent File of Component

4. 검색 시스템 구축

본 장에서는 객체지향 기법에 기반한 지식 구조와 정보 검색을 통한 부품의 인덱싱 방법을 바탕으로 본 논문에서 구축한 재이용 시스템을 기술한다.

재이용 시스템에 있어 가장 기본이 되는 것은 라이브러리, 라이브러리 분류방법, 지원 시스템이다. 본 장에서는 II장과 III장에서 제시한 두 방법에 따라 구축된 라이브러리의 재이용 환경을 지원하는 재이용 시스템의 일부로 구현된 검색 시스템을 설명하고 검색의 실제 결과를 기술한다.

4.1. 시스템의 전체 구성

본 연구의 시스템 개발 환경은 기존의 윈도우 환경 하에서 개발하였고, 시스템은 크게 라이브러리와 검색 시스템으로 구분되며 전체 구성은 그림 13과 같다.

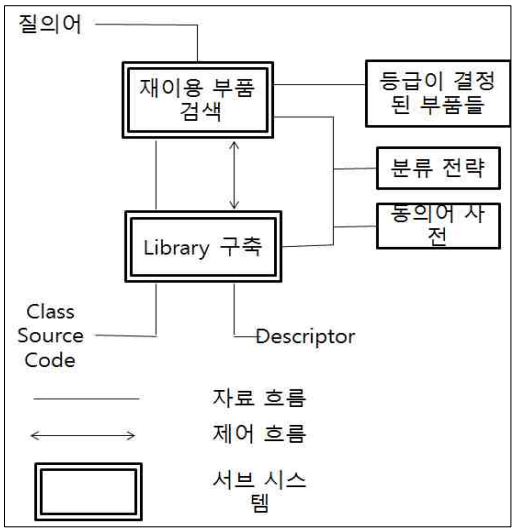

그림 13. 재이용 시스템의 전체 구성도
Fig. 13. Total Composition of Reuse System

4.2. 검색의 예와 결과

검색 환경은 다음과 같이 크게 4개의 작업영역으로 구분된다.

(1) Query Window : 재이용자가 필요한 부품의 기능을 기술하는 질의 작성 윈도우

(2) Component Window : 주어진 질의를 바탕으로 질의와 부품 인덱스 간의 유사도 측정 방식에 의해 가장 높은 부합도를 갖는 부품의 일반적 정보나 상세한 사항을 보여주는 윈도우 (3) Browse Window : Component Window에 제시된 부품과 기능적으로 유사성을 지니는 부품을 그 유사도에 따라 정렬하여 유사도와 함께 나타낸다.

(4) Class Window : 기능성과는 달리 객체지향 모델에 따라 라이브러리내의 클래스 계층 구조도를 표현한다.

초기에 작성된 질의는 (질의 1)과 같으며, Query Window에 작성된다.

“draw a polygon on the screen set the border color of polygon red display a regular polygon on the screen” ····· (질의 1)

작성된 질의어는 질의어 수정과 확장을 하여 그 결과로 Query Window에 (질의 1‘)과 같이 재구성된 질의어를 얻게 된다.

“display a polygon on the screen set the attribute attribute of polygon red display a regular polygon on the screen“ ····· (질의 1‘)

이 질의로부터 질의를 구성하는 인덱스들의 가중치를 빈도수에 의해 구한 질의는 (가중치를 포함한 질의 1“)과 같으며, 이를 바탕으로 하여 라이브러리에 대한 선형 검색을 한 후의 결과로 Component Window에 검색 결과(Regular_ polygon)를 표현하게 된다.

“display 2 polygon 3 set 1 attribute 2 red regular 1”····· (가중치를 포함한 질의 1“)

이 결과에 만족하지 않거나 다른 부품을 살펴보기 위하여 Browse Window와 Class Window에 제시된 부품을 선택할 수 있는데, 이렇게 선택된 부품을 바탕으로 새로운 클래스 계층 구조와 브라우즈 구조가 만들어 질 수도 있다. 그 결과는 Component Window에 검색된 부품과 가장 높은 유사도를 가지는 부품을 보여준다. (가중치를 포함한 질의 1“)에 의해 선택된 Reg_polygon과 유사한 부품은 그 유사도에 따라 내림차순으로 정렬한 후 Browse Window에 보여 주고 있다. 이들 부품중 하나를 선택할 경우 선택된 부품과의 유사도에 따라 Component Window의 내용과 Browse Window와 Class Window의 계층 구조가 바뀌어 제공되게 된다.

4.3. 검색 결과의 평가

라이브러리 검색 결과는 대개 “recall”과 “precision”에 의해 평가되는데, “recall”이란 관련없는 부품의 배제 기 부품의 검색 능력에 관한 평가이다.[9] 이를 그림으능에 관련된 평가 기준이며, “precision”이란 관련된로 설명하면 그림 14와 같다.

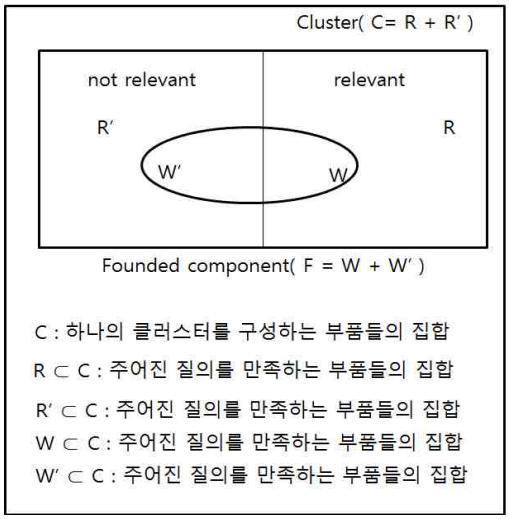

그림 14. “재호출"과 정확성"
Fig. 14. “recall" and ”precision"

재호출과 정확성은 각각 다음과 같이 표현된다.

재호출 = W / F

정확성 = W / R

검색의 예에서 보았듯이 본 시스템에 (질의 1)과 같은 질의를 주었을 때 가장 근접한 기능을 수행하는 부품으로 Reg_polygon이 검색되어졌으며 또한 클러스터 기반 부품 검색을 통해 Rectangle, Square, Polygon, Figure, Circle, Close_figure등의 부품을 유사한 기능을 갖는 부품으로 검색되었다. 주어진 질의를 바탕으로 검색한 라이브러리는 Graphic/Window 라이브러리로 라이브러리를 구성하는 클래스 부품의 개수는 87개이다.. 또한 각 부품을 구성하는 실제 메소드를 포함하면 전체 600여개의 함수를 갖고 있다.

그러나 Graphic/Window 라이브러리에는 질의에 부합되는 기능을 수행하는 부품이 Rectangle, Square, Polygon, Triangle, Close_figure, Circle, Ellipse 등이 존재했다. 이는 저장된 부품들 중 사각형과 유사한 그림을 그리거나 그려진 물체에 색을 지정하는 기능을 담당하는 부품이 7개 존재한다는 것을 의미한다. 이 경우 다음과 같은 결과를 얻을 수 있다.

재호출 = W / F = 4 / 6 = 0.66

정확성 = W / R = 4 / 7 = 0.57

검색의 결과로 얻어진 Rectangle, Square, Polygon, Figure, Circle, Close_figure중 Rectangle, Square, Polygon, Ccolse_figure의 4개만이 재사용자의 요구에 부응하는 것이며, 실제 라이브러리내에 저장된 Rectangle, Square, Polygon, Triangle, Close_Figure, Circle, Ellipse중 4개만이 검색된 것이다.

임의로 주어진 질의 30개에 대해 각각의 recall과 precision을 평가한 결과는 <표 5-2>과 같다.

표 1. 검색 시스템 성능 평가
Tab. 1. Efficiency Estimate of Retrieval

질의의 성격	recall	precision
일반적인 질의	0.73	0.6
재구성되지 않은 질의	0.45	0.39
애매한 질의	0.33	0.23
원하는 기능의 항의빈도를 높게 작성한 질의	0.53	0.83

동의어 사전의 유용도 확인을 위하여 동일한 질의 30개에 대하여 동의어 사전에 의한 질의의 확장/수정의 단계를 제외한 검색을 수행해 보았다. 그 결과로, 동일한 (질의 1)에 대하여 얻어진 결과는 Reg_Polygon으로 동일하였으나 이와 유사한 기능을 수행하는 부품으로 선정된 부품들이 (Figure, Segment, Point, Close-Figure, Line)등으로 실제 원하는 기능과 관련성이 떨어지는 부품이 검색되거나 실제 유사한 기능을 수행하는 부품이 검색되지 않은 결과를 얻었다. 즉 재호출l과 정확성이 떨어지는 결과를 얻게 되었다. 양 실험을 통해 얻은 또 한 가지의 경험은 애매한 질의, 예를 들어 “display”와 같

이 구체적인 요구 사항 없이 광범위한 부품인 Figure가 검색되었는데, 실제 원하는 부품을 발견하기 위하여 여러 번의 브라우즈 계층구조의 재구성이 요구되었다. 그러나 질의에 자신이 원하는 기능들 중 가장 중요한 것을 자주 사용한 경우, 예를 들어 Polygon에 속하기는 하지만 일정한 Box모양에 대한 여러 기능을 제공하는 부품을 검색하고자 한다면 질의에 Box를 자주 등장 시킬 경우 Polygon이나 Reg_Polygon보다는 Rectangle을 가장 유사한 부품으로 찾고 클러스터 기반 검색의 결과에는 Triangle이나 Circle등은 빠지게 된다. 즉, 더 좋은 precision을 얻을 수 있다는 것을 알 수 있었다.

5. 결 론

라이브러리 구축 시 라이브러리를 구성하는 부품의 특성을 잘 표현할 수 있는 부품의 조직화 방법과 부품이 수행하는 기능성에 관한 정보를 관리하고 저장된 부품들 간의 유사성을 파악함으로써 가능해진다.

본 논문에서는 기존의 소프트웨어 라이브러리 구축의 전 단계에 소요되는 라이브러리 관리자의 노력과 비용의 절감과 각 관리자의 경험과 지식에 기반한 방법보다는 더 표준화되고 기술적인 방법으로 소프트웨어를 분류하고 인덱싱하는 방법을 제안하였다.

다음 연구로는 최근에 부각되고 있는 시각적 프로그래밍 환경이나 그래픽 사용자 인터페이스의 많은 연구 결과를 이용하여 본 논문의 유사도 측정 결과를 상속관계로 나타내는 상속 그래프와 같은 형태로 시각화함으로써 사용자가 부품에 대한 유사도 정도를 쉽게 파악할 수 있도록 하는 방안에 관한 연구도 필요하다

REFERENCE

[1] R.E. Johnson snd B. Foote "Designing Reusable Classes" JOOP, 1983, pp. 6-7

[2] Horowitz, E. "An Expansive View of Reuseable Software" In Alan Perlis, editor, Workshop on Reusability in Programming, ITT Programming, Newport, RI, Setember, 1983, pp. 250-261

[3] Standish, T.A. "Software Reuse.", In Alin Perlis, editor, Workshop on Reusability in Programming, ITT Programming, Newport, RI, September, 1983

[4] Young Geil, Kim, "A Study on the Indexing and the Classification of the Component in the Information Retrieval Method", Korea National University of Transportation, 2013, pp.529-532

[5] Gerard Salton, Michel J. McGill, Introduction of Morden Information Retrieval, McGraw-Hill Book Company, 1983.,

[6] Neighbors, J. M. "Software Construction Using Components", Ph. D. thesis, University of California, Irvine, 1980.

[7] R. Prieto-Diaz "A Software Classification Scheme", Ph. D. thesis, University of California, Irvine, 1985.

[8] R. Prieto-Diaz and Neighbors, J. "Module Interconnection Language: A Survey", ICS Tech, Report 198, Dept. of Information and Computer Science, University of California, Irvine, August, 1982.

[9] A.S. Pollitt, Information Storage Retrieval System, EllisHorword, 1989.

저자약력

김 영 길(Young-Geil Kim) **[종신회원]**

1980년 2월 : 동국대학교 전자계산학과 졸업
1984년 8월 : 중앙대학교 대학원 컴퓨터공학과 졸업
1992년 8월 : 중앙대학교 대학원 컴퓨터공학과 졸업

<관심분야> 소프트웨어 공학, 임베디드 소프트웨어, 소프트웨어 보안, iTS 등