컴포넌트 검색을 지원하는 퍼지 기반 시소러스 구축

김 귀 정[†] · 한 정 수[††] · 송 영 재[†††]

요 약

컴포넌트 검색을 위한 많은 방법론이 제시되고 있고 그 중 유사 컴포넌트 검색을 위한 시소러스 개념이 도입되고 있다. 본 논문은 컴포넌트의 효율적인 검색을 위하여 컴포넌트를 구성하는 클래스들을 상속관계에 따라 개념적으로 분류하였고, 시소러스 방법에 퍼지 논리를 적용하여 객체지향 시소러스를 구축하였다. 제안한 방법은 개념들 사이의 범주를 자동으로 표현할 수 있으며, 각 클래스와 범주에 대한 매칭도와 비매칭도를 비교함으로서 클래스 사이의 퍼지 정도를 계산하여 시소러스를 구축하였다. 컴포넌트 검색은 컴포넌트를 구성하는 클래스들을 이용하여 유의어 테이블을 기반으로 후보 컴포넌트들을 검색한 후 퍼지 유사도 측정 방법을 이용하여 우선순위로 검색한다. 또한 시뮬레이션을 통하여 최적의 질의 확장 임계치를 설정함으로서 검색 성능을 크게 향상시켰다.

Fuzzy based Thesaurus Construction Supporting Component Retrieval

Gui Jung Kim[†] · Jung Soo Han[††] · Young Jae Song[†††]

ABSTRACT

Many Methodologies have proposed for component retrieval. Among them, thesaurus concept has introduced for similar component retrieval. This paper classified classes by concept according to inheritance relation for efficient retrieval of component, and applied fuzzy logic to thesaurus method and constructed object-oriented thesaurus. Proposed method could express category between concepts automatically, and calculate fuzzy degree between classes by comparing matching and mismatching degree to each class and category and construct thesaurus. Component retrieval is that using classes of component, candidate components are retrieved according to priority order using fuzzy similarity. Also, we improved retrieval performance by thesaurus greatly, setting critical of most suitable through simulation.

키워드 : 컴포넌트검색(Component Retrieval), 시소러스(Thesaurus), 질의확장(Query Expansion), 시뮬레이션(Simulation)

1. 서 론

객체 지향 방법론(Object-Oriented Methodology)의 영향으로 객체의 효율적 관리와 사용에 관한 관심이 높아지고 있으며 이를 활용하기 위해서는 실제적인 개념과 상황에 따른 올바른 객체의 검색, 선택, 적용 등이 필요하다[1-3]. 따라서 컴포넌트 검색을 위한 많은 방법론들이 제시되고 있는데 그 중 시소러스 개념이 많이 도입되고 있다[4]. 그러나 기존의 시소러스 연구에는 다음과 같은 몇 가지 문제점이 있다. 첫째 질의나 문제 질의로부터 개념적인 연상 과정의 명확성이 모호하여 개념적 용어의 정의가 어려운 점[5], 둘째 컴포넌트가 증가할수록 노이즈가 많이 발생하는 문제, 셋째 추출된 특성에 대해 도메인 전문가의 시간과 노력이 너무 많이 요구된다[6]. 이처럼 시소러스를 사용하는 일은 많은 검색 시간을 필요로 하며, 심지어 검색 과정에서 원하는 정보를 찾지 못하는 경우도 발생한다[7, 8].

본 논문에서는 컴포넌트의 효율적인 검색을 위하여 컴포넌트를 구성하는 클래스들의 상속관계를 이용하여 개념적으로 분류하였다. 또한 기존 시소러스가 갖는 고정된 형태의 관계구조를 상황에 따라 적절한 의미를 갖는 구조로 확장할 수 있으며, 구축자의 주관에 의해서만 규정되었던 개념 그룹화 작업이 상속관계에 의해 자동적으로 분류됨으로서 시소러스의 일관성과 객관성이 보장될 수 있도록 하였다. 이를 위해 본 연구에서는 클래스 분류를 위한 클래스 개념 범주(Class Concept Category : CCC)를 생성하였다. 시소러스는 CCC 관련 값(CRV)을 이용하여 각 클래스와 CCC에 대한 매칭도와 비매칭도를 비교하고 이들 사이의 퍼지 정도를 계산하여 구축한 유의어 테이블이다. 이 유의어 테이블은 질의 확장을 위해서 사용되며, 질의에 대한 컴포넌트가 존재하지 않을 경우에 질의와 유사한 컴포넌트를 검색한다. 또한 검색 노이즈의 감소를 위해서 질의 확장의 임

† 정 회 원 : 건양대학교 IT학부 교수
†† 정 회 원 : 천안대학교 정보통신학부학부 교수
††† 종신회원 : 경희대학교 전자계산공학과 교수
논문접수 : 2003년 5월 21일, 심사완료 : 2003년 7월 30일

계치를 시뮬레이션하여 최적의 임계치를 설정하였다.

본 연구에서 컴포넌트를 구성하는 클래스는 MFC 표준 라이브러리에 한정하였고, 컴포넌트 정보는 이러한 클래스들의 정보만을 가지고 실험하였다. 이를 바탕으로 컴포넌트의 검색을 효율적으로 수행할 수 있는 퍼지 기반 시소러스를 구축하였고, 시소러스에 의한 질의 확장을 통해 후보 컴포넌트까지 검색할 수 있으며, 질의와 컴포넌트 사이의 유사도에 의해 컴포넌트들이 우선순위로 검색될 수 있도록 하였다.

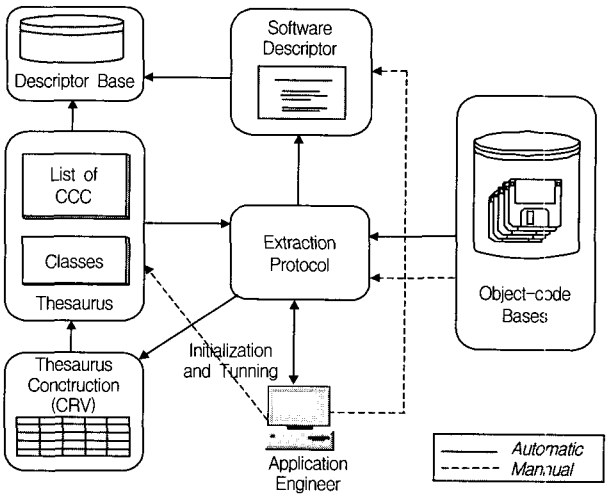
2. 관련 연구

객체 기반 시소러스[9]는 시소러스에 개념 표현 레벨과 인스턴스 표현 레벨로 구성된 객체지향 패러다임을 적용함으로서 객체들 사이에 존재하는 복잡한 관계성들의 표현에 자동 구축전략을 제공한다. 그러나 이 방법은 효과적인 질의 재형성 과정이 필요하며 실제적으로 응용될 수 있는 검색 시스템의 개발이 필요하다. 적응형 시소러스[10]는 스프레이딩 액티베이션 기반의 유사도 측정 방법을 기반으로 신경망을 이용하여 학습 가능한 시소러스를 제안하였다. 제안한 시소러스는 가중치를 효과적으로 조절할 수 있는 장점이 있지만, 표준화된 형태의 시소러스를 자동으로 추출하는 방법과 보다 적절한 활성화 함수를 개발하는 방법이 요구된다. 계층적 시소러스 시스템[6]은 코드에서 계층적 분류를 위한 범주를 설정하고, 컴포넌트가 행위적 특성에 따라 분류되는 방법을 제안하였다. 컴포넌트 특성은 용어의 쌍으로 구성되며, 소프트웨어 디스크립터(software descriptor : SD)로부터 추출된다[11, 12]. 특성에 따라 계층적으로 분류된 용어의 쌍은 퍼지 시소러스를 이용하여 유의어 사전을 구축하게 된다. 그러나 컴포넌트의 검색이 아닌 멤버함수와 파라미터를 이용한 클래스 검색이기 때문에 클래스가 증가할수록 멤버함수가 기하급수적으로 증가하여 노이즈가 많아진다는 단점이 있다.

3. 시소러스 구축

3.1 제안한 시소러스 모델

본 논문에서 제안한 시소러스 기반 검색은 컴포넌트를 구성하는 클래스들을 서로 관련 있는 여러 개의 그룹으로 구성하는 방식이다. 제안한 시소러스 구축 모델은 (그림 1)과 같다. 시소러스 구축을 위해서 각 클래스 그룹을 표현할 수 있는 개념을 선정하였고, 공통된 특징이나 기능별로 분류하는 과정은 패싯 분류 방법을 사용하였다. 각 패싯은 컴포넌트의 특성을 나타내며, 패싯 아이템이 하나의 개념으로서의 역할을 수행한다. 따라서 하나의 컴포넌트는 구성 클래스에 따라 여러 개의 패싯에 속할 수 있으며, 다중 상속과 기능에 따라 여러 개의 개념에 할당될 수 있다. 클래스들은 분류된 그룹에 따라 CCC 관련 값(CRV)이 계산되는데, 이 값은 클래스 사이의 유의 값(synonym value)을 계산하여 초기 시소러스를 구축할 수 있도록 해준다.



(그림 1) 시소러스 모델

3.1.1 CRV

본 연구에서는 각 컴포넌트의 기능적 분류인 CCC를 이용하였다. 본 연구의 기본적인 가정은 「한 컴포넌트의 소프트웨어 디스크립터를 구성하는 CCC에 대하여, i번째 CCC에 속한 클래스의 수는 컴포넌트와 i번째 CCC와의 관련성을 암시해준다.」이다. 이를 바탕으로 *'class-CCC 테이블'*을 정의하였다. 이 방법은 범주별 가중치 계산 방법에 기본을 두고 있으며[6], 이 테이블은 CCC 관련 값(CCC Relevance Value : CRV)에 의해 계산된다.

$$CRV_{i,j} = \frac{CCC\ j\text{에서 } i\ class\text{의 발생횟수}}{CCC\ j\text{에서 모든 } class\text{의 발생횟수}} \times \frac{CCC\ j\text{에서 } i\ class\text{의 발생횟수}}{\text{모든 } CCC\text{에서 } i\ class\text{의 전체발생횟수}}$$

이 식에서 'class 발생횟수'는 CCC 상속레벨에 따라 클래스가 포함된 CCC일 경우에는 1이 되고, 상위 클래스로부터 상속받은 CCC인 경우에는 상속레벨 h에 따라 $\left(\frac{1}{2}\right)^h$가 된다. 이는 클래스 자신이 포함된 CCC에 가장 큰 값을 부여하고, 상속받은 CCC에는 차등적인 값을 부여함으로서 객체지향 상속성을 자연스럽게 표현하기 위함이다. j번째 CCC에 있는 i번째 클래스의 CCC 관련성은 j번째 CCC에 나타나는 i번째 클래스의 백분율에 의존한다. 이 정의는 $CRV_{i,j}$의 값이 CCC의 상대적 크기에 상관없이 클래스 발생 빈도에 따라 좌우됨을 알 수 있다. 모든 CCC와 클래스에 대해서 CRV는 0에서 1사이의 값을 가지며, 한 CCC(j)에 어떤

클래스(i)가 전혀 나타나지 않았다면 $CRV_{i,j}$는 0의 값을 갖는다.

3.1.2 Extraction Protocol

클래스 정보 추출은 구문 분석에 의해 이루어지고, 추출 과정은 소스 코드를 읽어 각 클래스를 추출하고 클래스 사이의 관계 정보를 저장한다. 이때, 클래스는 자신이 가지고 있는 클래스 정보와 함께 추출된다. 클래스 정보는 클래스 관계 정보와 외부 정보로 나눌 수 있는데, 클래스의 상속 관계를 나타내는 상위 클래스와 하위 클래스가 클래스 관계 정보에 해당하고 root 클래스, 컴포넌트, CCC, 도메인 정보 등이 외부 정보에 해당한다.

3.1.3 SD

소프트웨어 디스크립터(SD)는 컴포넌트의 행위적 특성을 전체적인 관점에서 제공해주는 역할을 한다. 소프트웨어 디스크립터는 컴포넌트와 문서 정보로부터 구축되며, 구성단위는 컴포넌트의 기능을 포괄적이고 모듈별로 제공해주는 클래스 정보가 된다. 각 클래스는 컴포넌트에 대한 가중치를 가지고 있으며, 같은 클래스가 여러 컴포넌트에 포함될 수 있으므로 컴포넌트별로 클래스 가중치를 계산하게 된다. 소프트웨어 디스크립터를 구성하기 위해서는 클래스 정보로부터 CCC 분류가 이루어져야 한다.

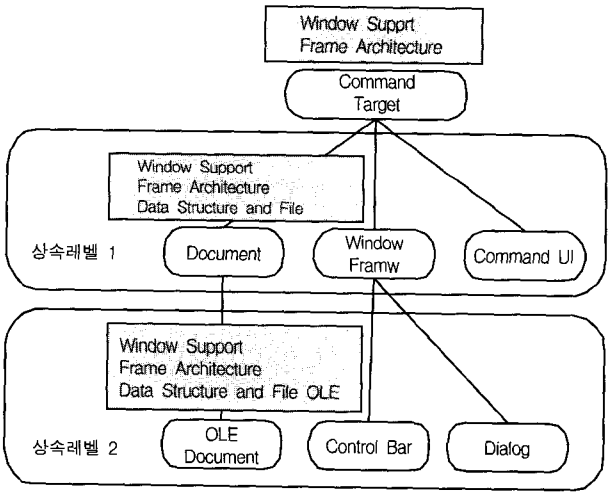
3.2 클래스 개념 범주 생성

클래스의 개념은 패싯 항목으로 분류되는데 본 연구에서는 클래스 분류를 위해 클래스 개념 범주(Class Concept Category : CCC)를 생성하였다. <표 1>은 클래스에 대한 패싯 항목(CCC)을 나타낸 것이다. 「Class Type」은 마이크로소프트 윈도우 애플리케이션 개발을 위한 MFC 클래스 라이브러리 분류를 바탕으로 클래스 타입을 5개의 CCC로 분류하였고, 「Component Type」, 「User Interaction Style」, 「Using Scope」은 기존의 객체지향 컴포넌트의 패싯 분류 방법을 이용하여 분류하였다[6].

〈표 1〉 패싯 분류에 의한 CCC

패 싯	패싯 항목(CCC)
Component Type	• File System and Interprocess Communication Service • User Interface Development
Class Type	• Window Support • Data Structure and File • Internet and Network • Exception and Debugging • OLE
Using Scope	• Frame Architecture • Application Processing
User Interaction Style	• Graphic and Drawing Support • Interactive and Dialog

분류된 CCC는 컴포넌트의 특성에 부합하는 클래스 상속 관계를 표현할 수 있다. 상속받은 하위 클래스는 상위 클래스와 같은 범주에 분류될 수도 있고, 다른 범주에 분류될 수도 있다. 그러나 다른 범주에 분류되어도 하위 클래스는 상위 클래스의 특성을 포함하고 있기 때문에 상위 클래스가 가지고 있는 CCC가 하위 클래스로 상속될 수 있도록 하였다. 그러나 상속이 거듭될수록 상위 클래스들이 포함된 특성은 일정하게 유지되지 못할 것이므로 CCC의 관련정도를 상속레벨에 따라 감소시켜야 한다. 따라서 본 연구는 상속레벨 h에 따라 CCC 관련정도를 $(1/2)^h$만큼 감소시켰다. (그림 2)는 CCC 상속을 보여준다. 'Document'는 자체적으로 'Data Structure and File' CCC에 포함되며, 'Command_Target' 클래스에서 상속되기 때문에 'Window Support'와 'Frame Architecture' CCC를 상속받을 수 있다. 이때, CCC 관계값은 CCC별 클래스 발생횟수를 상속레벨 h에 따라 $(1/2)^h$로 계산한다. 즉, 'Document'에 대한 3개의 CCC별 발생횟수가 'Data Structure and File'은 1, 'Window Support'는 $(1/2)^1$, 그리고 'Frame Architecture'는 $(1/2)^1$의 CCC 발생횟수를 가지게 된다.



(그림 2) CCC 상속

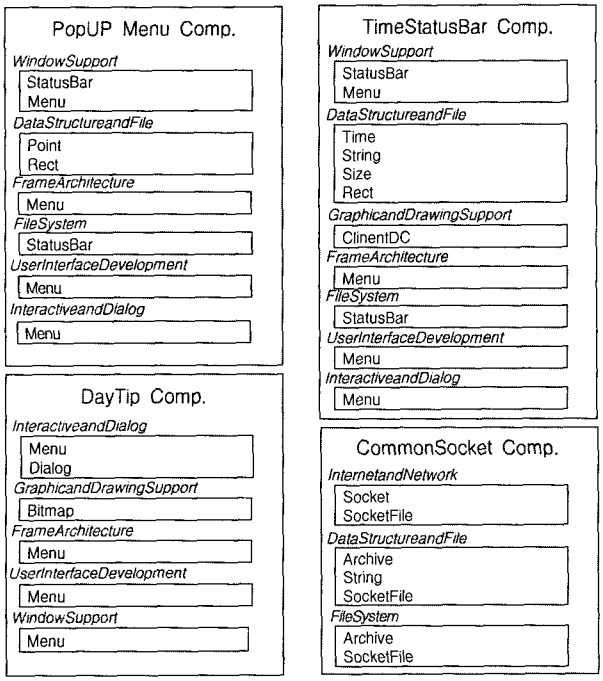
3.3 시소러스 구축

본 연구는 퍼지 시소러스[6]의 구축방법을 이용하여 CCC와 클래스 사이의 관련성과 상속에 초점을 두어 다음과 같이 시소러스를 구축하였다.

3.3.1 CCC 발생횟수 계산

클래스가 여러 CCC에 중복되는 경우는 클래스가 여러 특성을 가지고 있기 때문에 하나 이상의 패싯 항목에 해당함을 의미한다. 이때 클래스가 포함된 CCC뿐 아니라 상속받은 CCC도 해당 클래스와 관련이 있으므로 *class-CCC* 테이블은 직접 포함된 CCC와 상속받은 CCC 모두를 표현할 수 있어야 한다. (그림 3)은 CCC 별로 분류된 각 컴포

넌트의 클래스 리스트를 보여준다. <표 2>는 (그림 3)의 클래스에 대한 CCC 관련을 숫자로 표현한 것이다. '1'은 클래스가 해당 CCC에 포함된 경우이며, '1/2'은 상위 클래스가 포함된 CCC를 상속받은 것이며, '1/4'은 두 단계 상속받은 CCC를 나타낸다. 이 값들은 CRV를 계산할 때 클래스 발생횟수로서 사용된다.



(그림 3) CCC별로 분류된 컴포넌트

〈표 2〉 CCC 상속 표현

CCC \ Component	PopUpMenu	TimeStatus -Bar	Common -Socket	DayTip
Window Support	StatusBar : 1 Menu : 1 Rect : 1/2	StatusBar : 1 Menu : 1 Rect : 1/4 ClientDC : 1/2	—	Menu : 1 Dialog : 1/2
Data Structure	Point : 1 Rect : 1	Time : 1 String : 1 Size : 1 Rect : 1	SocketFile : 1 Archive : 1 String : 1	—
Graphic	Point : 1/2 Rect : 1/2	Rect : 1/2 ClientDC : 1	—	Bitmap : 1
Internet	—	—	Socket : 1 SocketFile : 1	—
Interactive	Menu : 1	Menu : 1	—	Menu : 1 Dialog : 1
Fram Arch.	Menu : 1 StatusBar : 1/2	Menu : 1 StatusBar : 1/2	—	Menu : 1 Dialog : 1/2
File System	StatusBar : 1	StatusBar : 1	SocketFile : 1 Archive : 1	—
Application Processing	—	—	SocketFile : 1/2 Archive : 1/2 String : 1/2	—
User Interaction	Menu : 1	Menu : 1	—	Menu : 1 Dialog : 1

3.3.2 j번째 CCC와 l번째 클래스에 대한 $CRV_{l,j}$ 계산

class-CCC 테이블에 CCC 관련 값(CCC Relevance Value : *CRV*)을 계산한다. CRV는 범주별 가중치 계산 방법에 기본을 두고 있으며, CRV 계산의 가정은 「한 컴포넌트를 구성하는 CCC에 대하여, i번째 CCC에 속한 클래스의 수는 컴포넌트와 i번째 CCC와의 관련성을 암시해준다.」와 같다. (그림 3)의 컴포넌트들의 CRV가 <표 3>과 같다.

〈표 3〉 CRV

CCC class	Window Support	Data Struc.	Grap -hic	Inter -net	Interac -tive	Fram Arch.	File System	App. Process -ing	User Inter -face
StatusBar	0.123	—	—	—	—	0.044	0.2	—	—
Menu	0.115	—	—	—	0.188	0.167	—	—	0.188
Point	—	0.074	0.048	—	—	—	—	—	—
Rect	0.011	0.127	0.082	—	—	—	—	—	—
Time	—	0.111	—	—	—	—	—	—	—
String	—	0.148	—	—	—	—	—	0.056	—
Size	—	0.111	—	—	—	—	—	—	—
ClientDC	0.026	—	0.190	—	—	—	—	—	—
Dialog	0.013	—	—	—	0.083	0.019	—	—	0.083
Bitmap	—	—	0.286	—	—	—	—	—	—
Socket	—	—	—	0.5	—	—	—	—	—
Socket -File	—	0.032	—	0.154	—	—	0.071	0.048	—
Archive	—	0.444	—	—	—	—	0.1	0.067	—

3.3.3 클래스와 클래스 사이의 매칭도, 비매칭도 계산

이 과정은 CRV를 이용하여 클래스 사이의 매칭도, 비매칭도를 계산하는 과정이다. 매칭도와 비매칭도는 퍼지논리를 이용한 방법으로 두 클래스 A, B가 한 CCC에 대하여 CRV가 서로 비슷할수록 유사함이 높고 즉 매칭도가 크고, CRV의 차가 클수록 비매칭도가 크다. 여기서 클래스 A와 B에 대해 CRV 테이블의 행에 해당하는 두 개의 벡터 (a_1, a_2, …, a_j, …, a_c)와 (b_1, b_2, …, b_j, …, b_c)를 얻을 수 있다. 벡터 값 a_j와 b_j는 j번째 CCC에 대한 A와 B의 CRV를 나타낸다. A와 B의 j번째 CCC에 대한 매칭도는 다음과 같이 계산된다.

j 번째 CCC 에 대한 매칭도

$$\begin{cases} m_j = \dfrac{1}{1+|a_j - b_j|} & , a_j > 0 \text{ and } b_j > 0 \\ 0 & , a_j = 0 \text{ or } b_j = 0 \\ & \quad a_j = 0 \text{ and } b_j = 0 \end{cases}$$

이는 A, B가 한 CCC에 동시에 나타나는 경우에만 클래

스 매칭도를 구할 수 있으며, 한 CCC에 대한 CRV가 서로 비슷할수록 매칭도는 1에 가까워지므로 A, B는 유사함을 뜻한다. 이러한 과정을 모든 CCC($1 \le j \le C_{ccc}$)에 대해 수행함으로서 두 클래스 A, B의 매칭도를 구할 수 있다.

$$\text{클래스 } A, B\text{의 매칭도 : } M = \sum_{j=1}^{C} m_j$$

또한, 클래스 A와 B의 j번째 CCC에 대한 비매칭도는 다음과 같이 계산된다.

$$j \text{ 번째 } CCC\text{에 대한 비매칭도}$$

$$\begin{cases} m_j^m = \dfrac{|a_j - b_j|}{1+|a_j - b_j|} & , either\ a_j = 0\ or\ b_j = 0 \\ 0 & , a_j = 0 \text{ and } b_j = 0 \\ & \quad a_j \neq 0 \text{ and } b_j \neq 0 \end{cases}$$

A, B가 어느 한 CCC에 나타나는 경우에만 클래스 비매칭도를 구할 수 있으며, 한 CCC에 대한 CRV의 차가 클수록 비매칭도는 1에 근접하며 A, B가 서로 상이함을 뜻한다. 이 과정을 모든 CCC($1 \le j \le C_{ccc}$)에 대해 수행함으로서 A, B의 비매칭도를 구할 수 있다.

$$\text{클래스 } A, B\text{의 비매칭도 : } M^m = \sum_{j=1}^{C} m_j^m$$

3.3.2에서 만들어진 CRV 테이블을 이용하여 클래스 'StatusBar'와 'Menu'의 매칭도와 비매칭도를 구해보면 다음과 같다.

'StatusBar'와 'Menu'의 매칭도 :

$$\frac{1}{1+|0.123-0.115|} + \frac{1}{1+|0.044-0.167|} = 1.882$$

'StatusBar'와 'Menu'의 비매칭도 :

$$\frac{0.188}{1+|0-0.188|} + \frac{0.188}{1+|0-0.188|} + \frac{0.2}{1+|0-0.2|} = 0.483$$

3.3.4 두 클래스와 CCC 사이의 매칭도, 비매칭도 계산

두 클래스에 대한 CCC 사이의 매칭도, 비매칭도 계산은 클래스 사이의 매칭도, 비매칭도 계산과 유사하다. 한 CCC에 대해 두 클래스가 동시에 나타나거나, 나타나지 않거나 하는 정도를 퍼지 집합을 사용하여 계산하는 방법이다. 따라서 CCC 매칭도는 한 CCC에 대해 동시에 나타나거나 동시에 나타나지 않은 두 클래스의 CRV만을 계산하며, 비매칭도는 두 클래스 중 하나의 클래스만이 나타난 CCC의 CRV를 선택한다. 그러므로 계산 방법은 3.3.3과 비슷하며 CCC 매칭도를 M'로 표시하고, CCC 비매칭도를 $M^{m\prime}$로 나타내도록 한다. 클래스 'StatusBar'와 'Menu'의 CCC 매칭도와 비매칭도를 구해보면 다음과 같다.

'StatusBar'와 'Menu'의 CCC 매칭도 :

$$\frac{1}{1+|0.123-0.115|} + \frac{1}{1+|0.044-0.167|} = 1.882$$

'StatusBar'와 'Menu'의 CCC 비매칭도 :

$$\frac{0.188}{1+|0-0.188|} + \frac{0.188}{1+|0-0.188|} + \frac{0.2}{1+|0-0.2|} = 0.483$$

3.3.5 두 클래스의 퍼지 유의값 계산

클래스 사이의 매칭도, 비매칭도, 그리고 두 클래스와 CCC 사이의 매칭도, 비매칭도를 이용하여 두 클래스의 최종적인 퍼지 유의값을 생성할 수 있다.

$$f = \max\left(0, \frac{1}{2} \times \left(\frac{M - M^m}{M + M^m} + \frac{M' - M^{m\prime}}{M' + M^{m\prime}}\right)\right) \quad 0 \le f \le 1$$

$\frac{1}{2} \times \left(\frac{M - M^m}{M + M^m} + \frac{M' - M^{m\prime}}{M' + M^{m\prime}}\right)$이 0보다 작은 값을 가질 경우에는 매칭도보다 비매칭도가 더 크다는 것을 의미하므로 유의성이 전혀 없음을 나타내는 0으로 설정한다. 따라서 'StatusBar'와 'Menu'의 최종적인 유의값은 다음과 같다.

$$f_{'StatusBar','Menu'} = \frac{1}{2} \times \left(\frac{1.882 - 0.483}{1.882 + 0.483} + \frac{1.882 - 0.483}{1.882 + 0.483}\right) = 0.592$$

3.3.6 모든 클래스에 대한 퍼지 유의어 테이블 생성

3.3.1에서부터 3.3.5까지의 과정을 시소러스 내의 모든 클래스에 대해 수행함으로서 모든 클래스들 간의 유의어 값을 생성할 수 있다. 이와 같은 방법으로 클래스 'SoketFile'와 'Archive'의 최종적인 유의값은 0.905로 계산된다. 이는 <표 3>에서 직감적으로 'StatusBar'와 'Menu' 클래스보다 'SoketFile'와 'Archive'의 관련도가 더 깊음을 알 수 있다.

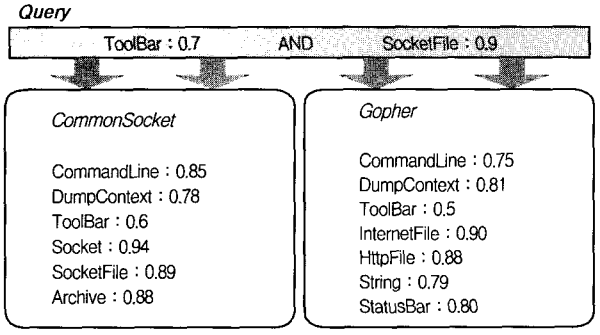
4. 컴포넌트 검색

4.1 시소러스에 의한 질의 확장과 컴포넌트 검색

본 연구의 컴포넌트 검색은 퍼지 불리언 형태로 표현된 질의를 시소러스를 통해 확장함으로서 이루어진다. (그림 4)는 질의가 'ToolBar : 0.7 AND SocketFile : 0.9' 일 때 두 컴포넌트 'CommonSocket'과 'Gopher'의 질의 확장과 컴포넌트 검색을 보여주기 위한 예이다. 각 색인어에 대한 가중치는 코사인 정규화 방법을 이용한 클래스 가중치 계산방법에 의해 얻어진다[13].

'CommonSocket'에 대한 질의 평가와 검색 여부 평가 방법은 다음과 같다. 질의가 q1 = [ToolBar : 0.7]이고 q2 = [SocketFile : 0.9]라면, 'CommonSocket'는 q1을 0.6(min(0.6, 0.7)) 정도로 만족하고 q2를 0.89(min(0.89, 0.9)) 정도로 만

족하고 있다. 따라서 q1과 q2를 동시에 만족하는 정도는 min(0.6, 0.89)로 해석되므로, 질의에 대한 컴포넌트의 만족 정도는 0.6으로 평가될 수 있다. 만족도가 0.5 이상이면 질의에 대해 컴포넌트가 어느 정도 연관성이 높다고 인정되므로 'CommonSocket'을 검색될 후보 컴포넌트에 포함시킨다[5]. 그러나 'Gopher'는 의미적으로 질의와 상당히 관련이 있지만, 색인 집합에 'SocketFile'이 포함되지 않기 때문에 q2를 0.0(min(0.0, 0.9)) 정도로 만족하고 있다. 즉, 'Gopher'가 q1과 q2를 동시에 만족하는 정도는 0.0(min(0.5, 0.0))으로 해석되므로, 'Gopher'는 검색되지 않는다. 이 단점은 검색의 재현율을 감소시키는 주된 원인으로 작용하기 때문에 효과적인 검색을 저해하게 된다. 본 논문에서는 이 단점을 해결하기 위해 퍼지 시소러스를 통한 질의 확장 방법을 이용하였다.

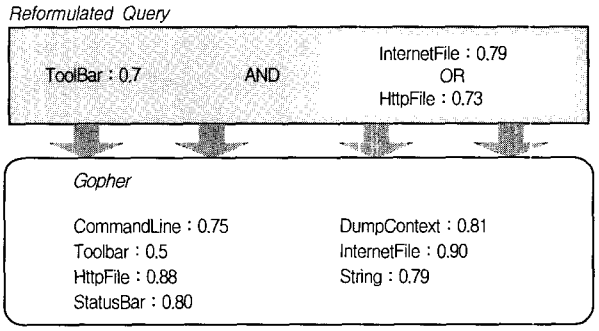


(그림 4) 질의 확장과 컴포넌트 검색

〈표 4〉 'SocketFile' 시소러스 유의어 값

class명 \ class명	...	InternetFile	HttpFile	Archivet	Menu	...
...	...	...	...	...	...	...
SocketFile	...	0.79	0.73	0.65	0.4	...
...	...	...	...	...	...	...

'Gopher' 검색을 위한 질의 확장은 다음과 같다. 3.3절에서 기술한 시소러스에 과정에 의해 "q2 = [SocketFile : 0.9]"는 <표 4>와 같은 유의값을 갖는다. 'SocketFile'은 모든 클래스에 대한 유의값을 가지고 있으며, 이중 유의값이 0.7이상의 클래스들만이 확장의 대상이 된다. 이는 정확도를 적절히 유지하면서도 재현율이 높게 나타나는 임계치 범위를 시뮬레이션 통하여 설정한 것으로, 5.1절에 자세히 기술하고 있다. 이에 따라 퍼지 시소러스에서 q2 = 'SocketFile'의 질의 확장 집합 $Exp_{(SocketFile)}$은 {SocketFile : 1.0, InternetFile : 0.79, HttpFile : 0.73}과 같이 구성된다. 여기서 q2의 'SocketFile' 중요도가 0.9로 설정되었기 때문에 질의 확장 집합 $Exp_{(SocketFile)}$의 확장 질의에 대한 유의값이 조절되어야 하는데, 질의에 주어진 중요도와 각 확장 질의의 유의값 중 적은 값을 선택한다. 그러므로 질의 q2 = [SocketFile : 0.9]에 대한 최종적인 질의 확장 집합 $Exp_{(SocketFile)}$은 {SocketFile : 0.9, InternetFile : 0.79, HttpFile : 0.73}과 같이 확장된다.



(그림 5) 확장된 질의

(그림 5)은 이를 바탕으로 재형성된 질의를 나타내며, 'Gopher'의 검색은 질의가 q1 = [ToolBar : 0.7] 이고 q2 = [InternetFile : 0.79 OR HttpFile : 0.73]이므로, 'Gopher'는 q1을 0.5(min(0.5, 0.7)) 정도로 만족한다. 또한 q2에 대해 확장된 각 질의 중요도와 컴포넌트 가중치 중 적은 값을 선택한 후, 퍼지 OR 연산을 수행한다. 즉, max(min(0.79, 0.90), min(0.73, 0.88)) = 0.79의 값을 얻을 수 있다. 이는 'Gopher'가 q2를 0.79 정도로 만족함의 의미한다. 따라서 'Gopher'가 q1과 q2를 동시에 만족하는 정도는 min(0.5, 0.79)으로 해석되므로, 질의에 대한 컴포넌트의 만족 정도는 0.5로 평가되어 최종적으로 'Gopher'는 후보 컴포넌트에 포함되어 검색된다.

4.2 후보 컴포넌트의 유사도 계산

유의어 테이블에 의해 질의가 확장된 후, 이 질의를 만족하는 후보 컴포넌트들이 검색된다. 검색된 컴포넌트의 우선순위를 결정하기 위하여 질의와 컴포넌트들의 유사도를 계산한다[14]. 'Gopher'를 이용하여 유사도를 계산하는 방법은 다음과 같다.

Query
ToolBar : 0.7 AND SocketFile : 0.9
Gopher Component
CommandLine : 0.95, DumpContext : 0.81,
ToolBar : 0.50, InternetFile : 0.90,
HttpFile : 0.88, String : 0.79,
StatusBar : 0.80

4.2.1 질의와 컴포넌트간의 동치관계 계산

유사도를 계산하기 위한 첫 번째 단계는 각 질의와 컴포넌트를 구성하는 클래스 사이의 유의정도 즉 동치관계(Equivalence)를 유의어 테이블로부터 찾아내는 과정이다. 이 식은 질의에 나타난 클래스와 컴포넌트에 있는 각 클래스간의 유의값을 반환한다.

$Eq(Query(u), Comp(v)) = SYNON(Query(u), Comp(v))$

u : 질의에 있는 질의어 갯수

v : 컴포넌트에 있는 클래스 갯수

〈표 5〉 질의에 대한 "Gopher" 컴포넌트의 동치관계 값

	Command -Line	Dump -Context	ToolBar	Internet -File	Http -File	String	Status -Bar
ToolBar	0.769	0.894	1.000	0.654	0.580	0.753	0.912
SocketFile	0.752	0.874	0.942	0.901	0.930	0.872	0.890

4.2.2 질의와 컴포넌트간의 함축관계 계산

함축관계(Implication)는 질의 중요도와 컴포넌트 가중치 중 큰 값을 반환하여 4.2.1에서 얻어진 동치관계와 곱한다. 함축관계 식에 의해서 질의 중요도가 함축관계에 의해 계산된 값보다 작거나 같을 경우에 질의와 컴포넌트의 각 클래스에 대한 교환이 가능함을 의미한다. 함축관계 계산식은 다음과 같다.

$$Imp(Query(u), Comp(v)) = \max w(Query(u)), w(Comp(v))[Eq(u, v)]$$

〈표 6〉 질의에 대한 "Gopher" 컴포넌트의 함축관계 값

	Command -Line	Dump -Context	ToolBar	Internet -File	Http -File	String	Status -Bar
ToolBar	0.731	0.724	0.700	0.589	0.510	0.595	0.730
SocketFile	0.714	0.787	0.848	0.811	0.837	0.785	0.801

4.2.3 질의와 컴포넌트 클래스의 만족도 계산

만족도(satisfaction value)는 질의와 컴포넌트의 각 클래스가 얼마나 호환성이 있는가를 의미한다. 이는 각 컴포넌트 클래스에 대하여 질의 함축관계와 동치관계 값을 곱한 후 그들의 합으로 얻어질 수 있다.

$$Sat(Query, Comp(v)) = \frac{\left[\sum_{u=1}^{U} Imp(u, v) \times Eq(u, v)\right]}{U}$$

$\sum_{u=1}^{U} Imp(u, v) \times Eq(u, v)$는 질의에 대한 한 클래스의 만족정도를 나타내는 값으로서, 함축관계와 동치관계 값이 1일 경우 최대 질의 수만큼의 값 U를 가질 수 있다. 따라서 이 값을 질의 수(U)로 나누어 0과 1사이의 값을 나타낼 수 있도록 하였다.

$$Sat(Query, Comp(v)) = \{0.550, 0.668, 0.725, 0.558, 0.537, 0.567, 0.690\}$$

4.2.4 질의와 컴포넌트간의 유사도 계산

4.2.3의 만족도에 컴포넌트의 가중치 벡터를 적용함으로서 최종적인 질의와 컴포넌트간의 유사도를 계산한다. 가중치 벡터의 역할은 컴포넌트내에 있는 클래스들의 매칭 효과를 높이고자 하는데 있다. 같은 클래스들로 구성된 컴포넌트가 여러 개 존재할 경우, 같은 질의에 대해서 구성 클래스의 가중치가 큰 것이 우선으로 검색되어야 하기 때문에 클래스 가중치 벡터를 이용함으로서 가중치가 큰 클래스의 컴포넌트가 더 높은 값을 가질 수 있도록 하였다.

다음은 'Gopher' 컴포넌트의 가중치 벡터이다.

$$W = \frac{w(Comp(k))}{\sum_{i=1}^{V} w(Comp(i))} = \left\{\frac{0.95}{5.63}, \frac{0.81}{5.63}, \frac{0.5}{5.63}, \frac{0.9}{5.63}, \frac{0.88}{5.63}, \frac{0.79}{5.63}, \frac{0.8}{5.63}\right\}$$

'Gopher'의 질의에 대한 유사도는 다음과 같다.

$$\begin{aligned} Sim(Query, Comp) &= \sum (Sat \times W) \\ &= \sum(\{0.550, 0.668, 0.725, 0.558, 0.537, 0.567, 0.690\} \\ &\quad \times \{0.169, 0.144, 0.089, 0.160, 0.156, 0.140, 0.142\}) \\ &= 0.604 \end{aligned}$$

질의에 대해 'Gopher'는 0.604의 유사도를 가지고 있음을 알 수 있다. 위와 같은 방법으로 검색된 모든 후보 컴포넌트에 대해서 유사도를 계산한 후 가장 높은 값을 가진 컴포넌트 순서로 출력하도록 하였다.

5. 성능 평가

5.1 질의 확장 임계치 평가

시뮬레이션에 사용된 컴포넌트는 Visual C++ Class Library를 이용하였다. 이 클래스들은 범용 라이브러리이며 특정 범주에 치우치지 않으므로 다양한 클래스를 제공해 준다. 시뮬레이션 환경은 131개의 컴포넌트가 11개의 CCC에 포함되어 있으며, 각 CCC는 컴포넌트에 따라 최소 1개에서 최대 22개까지의 클래스로 구성되어 있다. 평균 클래스 수는 약 8개이고, 모든 CCC에는 중복을 포함하여 총 1023개의 클래스가 존재하며 독립된 303개의 클래스로 구성하였다.

시소러스 효율성은 질의에 대한 효과적인 확장에 달려 있다. 따라서 본 연구에서는 컴포넌트 검색의 효율을 최대한 보장해줄 수 있는 최적의 질의 확장 임계치를 시뮬레이션을 통하여 설정하였다. 임계치를 위해 임의의 클래스 10개를 선택하고, 선택된 클래스에 대한 유사 확장 집합(similar expended sets)을 수동으로 선정하였다. 유사 확장 집합이란 한 클래스에 대해 유사하거나 기능적으로 함께 사용될 수 있는 클래스들을 의미하며, 유사 확장 집합에 속한 클래스들은 시소러스에 있는 클래스로 한정한다. 그 후, 질의로

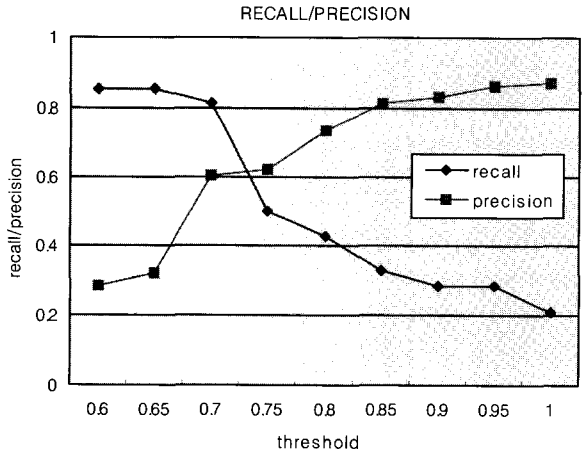
주어진 10개의 클래스에 대해 본 연구에서 제안한 시소러스에 의해 확장된 질의와 유사 확장 집합과의 비교를 통하여 유의값을 0.6에서부터 1.0까지 0.05의 간격으로 각각의 정확도와 재현율을 측정하였다[6].

$$\text{재현률} = \frac{\text{확장된 유의어 중 유사확장 집합에 속한 유의어의 수}}{\text{유사확장 집합의 수}}$$

$$\text{정확도} = \frac{\text{확장된 유의어 중 유사확장 집합에 속한 유의어의 수}}{\text{확장된 유의어의 수}}$$

〈표 7〉 임계값 별 검색 효율

임계치	검색 효율	
	정확도	재현율
0.60	0.287	0.853
0.65	0.323	0.853
0.70	0.603	0.811
0.75	0.620	0.500
0.80	0.734	0.428
0.85	0.812	0.329
0.90	0.829	0.285
0.95	0.860	0.285
1.00	0.870	0.210



(그림 6) 임계값 별 정확도/재현율 비교

<표 7>과 (그림 6)은 임의의 10개 클래스에 대한 임계값 별 검색 효율에 따른 정확도와 재현율의 평균을 보여주고 있다. 임계값 0.7미만이 되면 정확도가 0에 가깝게 되어 검색 효율이 떨어지고, 임계값 0.8부터는 0.7과 정확도에 별 차이가 없지만 재현율은 상당히 떨어짐을 알 수가 있다. 따라서 본 연구에서는 정확도를 유지하면서 재현율을 최대한 보장할 수 있는 범위인 임계값 0.7을 질의 확장의 범위로 설정하여 검색 효율을 최대한 보장할 수 있도록 하였다.

5.2 제안한 시소러스 평가

본 연구에서는 시소러스의 효율성을 평가하기 위하여 검색 효율성의 기준이 되는 재현율과 정확도를 측정하였다. 질의는 임의로 20개를 선정하였고, No-시소러스와 본 연구에서 제안한 방법으로 시소러스를 사용한 경우의 재현율을 0.1 단위로 변화시키면서 정확도의 비율을 <표 8>과 같이 비교하였다[15]. 이 실험에서 본 연구에서 제안한 시소러스에 의한 검색은 No-시소러스에 비해 효율성에 있어서 22.3% 정도 크게 향상되었음을 보여주고 있다.

〈표 8〉 재현율과 정확도의 비율

Recall	No-시소러스의 Precision	제안한 시소러스의 Precision
0.1	0.68	1.00
0.2	0.70	0.95
0.3	0.78	0.92
0.4	0.75	0.82
0.5	0.72	0.86
0.6	0.63	0.72
0.7	0.59	0.71
0.8	0.54	0.65
0.9	0.48	0.57
1.0	0.42	0.49
평균 Precision	0.629	0.769
향상된 평균 비율	-	22.3%

〈표 9〉 시소러스 비교 분석

항목 / 시스템	검색대상	구성항목	범주분류	질의연산 방법	시소러스 구축방법
No Thesaurus	함수 클래스	함수형 데이터형	단순 category	string matching	×
객체기반 시소러스[9]	객체	개념	단순 category	불리언 연산	관계확장
신경망 시소러스[10]	문서	용어	단순 category	확장 불리언 연산	스프레딩 액티베이션 + 신경망
계층적 시소러스[6]	클래스	함수명 인자명	context	context 조합	퍼지시소러스 + 통계
제안한 시소러스	컴포넌트	클래스	CCC	불리언 연산	퍼지시소러스 + 통계 + CCC상속

<표 9>의 시소러스 비교 분석을 보면 타 시스템은 검색 대상이 함수, 문서 또는 클래스로 국한되기 때문에 전정한 의미의 객체지향 컴포넌트의 검색이라고 볼 수 없다. 또한, 범주 분류에 있어서도 대부분의 경우 수작업을 통하여 항목을 분류해야 하지만, 본 연구에서 제안한 CCC는 클래스 상속관계를 기본으로 분류되기 때문에 그 의미가 자연스럽고 클래스간의 기능 분류가 자동으로 이루어 질 수 있다는 점을 가지고 있다. 특히 계층적 시소러스는 상속관계를 고려한 객체지향 시소러스이긴 하지만, 코드에서 추출한 함수

명과 인수명을 가지고 시소러스를 구축하기 때문에 데이터 양이 너무 방대해져 구축과 관리, 갱신에 어려움이 따르고, 숙련된 전문가도 사용하기 어려운 단점이 있다.

6. 결 론

본 연구는 컴포넌트의 효율적인 검색과 유동적인 시스템 구조의 구축을 위하여 클래스의 상속관계를 이용한 개념 분류와, 퍼지 논리를 적용한 객체지향 시소러스를 구축하였다. 제안한 방법은 클래스 분류를 위한 클래스 개념 범주를 생성하고 모든 클래스에 대한 CCC별 클래스 관련값을 이용하여 퍼지 기반 시소러스 유의어 테이블을 구성하였다. 클래스와 CCC에 대한 매칭도와 비매칭도를 비교하고 이들 사이의 퍼지 정도를 계산하여 퍼지 시소러스를 구축하였다. 검색 노이즈의 감소를 위해서 본 연구에서는 질의 확장의 임계치를 조절하여 효율성을 시뮬레이션한 결과, 유의값 0.7 이상인 경우에 재현율과 정확도에 있어서 평균 81.1%, 60.3% 이상을 보였으며, 제안한 시소러스의 효율성이 No-시소러스에 비해 22.3% 정도 크게 향상되었음을 보여주었다. 따라서 본 연구는 시소러스에 의한 질의 확장을 통하여 후보 컴포넌트까지 검색하여 컴포넌트 선택의 범위를 넓힐 수 있었으며, 질의와 컴포넌트 사이의 유사도에 의해 컴포넌트들이 우선순위로 검색될 수 있는 효율적인 객체지향 컴포넌트 기반 시소러스를 구축하였다. 따라서 본 연구는 클래스 라이브러리에 대한 개념을 이용하여 컴포넌트들을 검색하고 우선순위로 표현하기 때문에 컴포넌트 조립시 보다 효율적이다.

그러나 본 연구는 CCC 별 출현빈도에 따라서 시소러스를 구축하였기 때문에 클래스의 수가 적은 도메인이나 특정 범주에 클래스가 몰려있는 환경에서는 효율성이 떨어지는 경향이 있다. 또한 검색결과는 유사도에 의존하였기 때문에 컴포넌트들에 대한 정확한 기능 정의에만 한정되어 있어서 더 많은 컴포넌트의 이해를 위한 정보가 요구된다.

참 고 문 헌

[1] Frankes and B. William, "Information Retrieval-Data Structure and Algorithms," Prentice-Hall, 1992.

[2] R. F. Simmons, "Probability and Fuzzy-Set Application to Information Retrieval," Annual Review of Information Science and Technologyp, pp.117-151, 1985.

[3] C. Danilowicz, "Modelling of User Preference and Needs in Boolean Retrieval System," Information Processing and Management, Vol.30, No.3, pp.363-378, 1994.

[4] E. Damiani and M. G. Fugini, "Automatic thesaurus construction supporting fuzzy retrieval of reusable components," *Proceeding of ACM SIG-APP Conference on Applied Computing*, Feb., 1995.

[5] 최재훈, 김지숙, 조기환, "문제 은행에서 연상학습 을 지원하는 퍼지 검색 시스템", 한국정보과학회논문지, Vol.29, No.4, pp.278-288, Apr., 2002.

[6] E. Damiani, M. G. Fugini and C. Bellettini, "Aware Approach to Faceted Classification of Object-Oriented Component," ACM Transaction on Software Engineering and Methodology, Vol.8, No.4, pp.425-472, Oct., 1999.

[7] R. Rada, H. Mili, E. Bickenell and M. Blettner, "Development and Application of a Metric on Semantic Nets," IEEE Transaction on System, Mand Cybernetics, Vol.19, No.1, pp. 17-30, 1989.

[8] H. Chen, T. Tim and D. Fye, "Automatic Thesaurus Generation for an Electronic Community System," Journal of the American Society for Information Science, Vol.46, No.3, pp.175-193, 1995.

[9] 최재훈, 한종진, 박종진, 양재동, "구조적인 시소러 스 구축을 지원하는 객체 기반 정보 검색 모델", 한국정보과학회논문지, Vol.24, No.11, pp.1244-1256, Nov., 1997.

[10] 최종필, 최명복, 김민구, "신경망을 이용한 적응 형 시소러스", 한국정보과학회논문지, Vol.27, No.12, pp.1211-1218, Dec., 2000.

[11] Y. S. Maarek, D. M. Berry and G. E. Kaiser, "An information retrieval approach for automatically constructing software library," IEEE Transaction on Software Engineering, Vol.18, No.8, pp.800-813, Aug., 1996.

[12] D. Batory and S. O'Malley, "The design and implementation of hierarchical software systems with reusable components," ACM Transaction on Software Engineering and Methodology, Vol.1, No.4, pp.355-398, Oct., 1992.

[13] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing and Management, Vol.24, No.5, pp.513-523, 1988.

[14] M. Moormann Zaremski and J. M. Wing, "Signature matching : A tool using software librarys," ACM Transaction on Software Engineering and Methodology, Vol.4, No.2, pp.146-170, Apr., 1995.

[15] B. Y. Ricardo and R. N. Berthier, "Modern Information Retrieval," Addison-Wesley, 2000.

김 귀 정

e-mail : gjkim@konyang.ac.kr

1994년 한남대학교 전자계산공학과(공학사)

1996년 한남대학교 전자계산공학과(공학석사)

2003년 경희대학교 전자계산공학과(공학박사)

2001년~현재 건양대학교 IT학부 교수

관심분야 : S/W 재사용, CASE 도구, 컴포넌트 검색

한 정 수

e-mail : jshan@cheonan.ac.kr

1990년 경희대학교 전자계산공학과(공학사)

1992년 경희대학교 전자계산공학과(공학석사)

2000년 경희대학교 전자계산공학과(공학박사)

2001년~현재 천안대학교 정보통신학부학부 교수

관심분야 : CBD, 컴포넌트 관리, CASE 도구

송 영 재

e-mail : yjsong@khu.ac.kr

1969년 인하대학교 전기공학과(공학사)

1976년 일본 Keio University 전산학과(공학석사)

1979년 명지대학교 대학원(공학박사)

1971년~1973년 일본 Toyo Seiko 연구원

1982년~1983년 미국 Univ. of Maryland 전산학과 연구교수

1984년~1989년 경희대학교 전자계산소장

1985년~1989년 IEEE Computer Society 한국지회부 회장

1993년~1995년 경희대학교 교무처장

1996년~1998년 경희대학교 공과대학장

1999년~2000년 경희대학교 기획조정실장

2001년~2002년 경희대학교 산업정보대학원장

1976년~현재 경희대학교 전자계산공학과 교수

관심분야 : CBD, CASE 도구, S/W 개발도구론