XML 컴포넌트 명세서 기반의 컴포넌트 검색 기법

(A Search Method for Components Based-on XML Component Specification)

박 서 영 † 신 영 길 †† 우 치 수 ††

(Seoyoung Park) (Yoeng-Gil Shin) (Chisu Wu)

요 약 최근 들어 컴포넌트는 소프트웨어 재사용의 핵심 기술로 인식되고 있다. 컴포넌트는 인터페이스 부분만을 이용하여 개발되는 소프트웨어에 바로 바인딩시켜 재사용될 수 있는 독립적인 바이너리 코드이다. 많은 컴포넌트 사용자들이 인터넷 상에서 적합한 컴포넌트를 검색하고 사용할 수 있도록, 컴포넌트 명세서는 웹 문서 형태를 사용하는 것이 바람직하다. 기존의 HTML 웹 문서 기반의 검색 엔진을 통하여 컴포넌트를 검색할 때 정확한 의미 검색이 불가능하다는 문제점이 있다. 본 논문에서는 정확한 의미 검색을 수행할 수 있도록 컴포넌트 명세서를 HTML 문서 대신 XML 문서로 사용할 것을 제안한다. 또한, XML 컴포넌트 명세서를 통하여 사용자가 원하는 컴포넌트를 정확하게 검색할 수 있는 XML 문맥 기반 검색(context-based search)을 제안한다. 문맥 기반 검색은 컴포넌트의 특성을 나타내는 문맥(context)과 컴포넌트 특성의 실제 값인 용어(term)를 사용하여 사용자가 원하는 컴포넌트의 특성을 정확하게 질의하고 검색할 수 있는 방법이다. 이 검색 방법은 용어-문맥-컴포넌트 명세서 순으로 된 역화일 인덱싱(Inverted File Indexing) 방법을 사용한다. 이와 함께 사용자의 편의를 위하여, 기존의 소프트웨어 재사용 라이브러리에서 사용되던 검색 방법인 키워드 검색, 퍼싯 검색, 브라우징 검색 방법 등을 지원한다. 이들 다양한 검색 방법들은 인터페이스 레이어, 질의 확장 레이어, XML 검색 엔진 레이어 등 3-레이어 검색 엔진 구조를 통한 효율적인 인덱스 스킴에 의해 지원된다. 본 논문에서는 컴포넌트 사용자들이 원하는 컴포넌트를 정확하게 검색할 수 있도록 하기 위하여 컴포넌트 명세서를 대한 XML DTD(Document Type Definition)를 정의하고, HTML 기반 검색 방법과 XML 기반 검색 방법에 대한 컴포넌트 검색 성능을 비교한다.

Abstract Recently, the component technology has played a main role in software reuse. It has changed the code-based reuse into the binary code-based reuse, because components can be easily combined into the developing software only through component interfaces. Since components and component users have increased rapidly, it is necessary that the users of components search for the most proper components for HTML among the enormous number of components on the Internet. It is desirable to use web-document-typed specifications for component specifications on the Internet. This paper proposes to use XML component specifications instead of HTML specifications, because it is impossible to represent the semantics of contexts using HTML. We also propose the XML context-search method based on XML component specifications. Component users use the contexts for the component properties and the terms for the values of component properties in their queries for searching components. The index structure for the context-based search method is the inverted file indexing structure of term-context-component specification. Not only an XML context-based search method but also a variety of search methods based on context-based search, such as keyword, search, faceted search, and browsing search method, are provided for the convenience of users. We use the 3-layer architecture, with an interface layer, a query expansion layer, and an XML search engine layer, of the search engine for the efficient index scheme. In this paper, an XML DTD(Document Type Definition) for component specification is defined and the experimental results of comparing search performance of XML with HTML are discussed.

· 본 논문은 1998년도 한국과학재단 핵심전문연구(과제번호: 981- 0923-119-2) 연구비 지원에 의하여 연구되었음

† 비 회 원 : 서울대학교 전산과학과
psy@selab.snu.ac.kr
†† 종신회원 : 서울대학교 전산과학과 교수
yshin@cglab.snu.ac.kr
wuchisu@selab.snu.ac.kr

논문접수 : 1999년 7월 28일
심사완료 : 1999년 12월 13일

1. 서 론

1968년, Doug McIlroy[23, 5]가 공유 가능한 소프트웨어 부품들의 라이브러리를 처음으로 제안한 이래, 소프트웨어 재사용은 소프트웨어 생산성 향상을 위한 중요한 방법으로 인식되어 왔다.[30] 소프트웨어 부품 라이브러리나 소프트웨어 저장소(repository)는 품질 높은 소프트웨어 부품을 쉽고 많은 사람들이 이용할 수 있도록 해주기 때문에, 소프트웨어 재사용의 중추 역할을 해왔다[4]. 더욱이, 인터넷과 웹에 대한 사용이 폭발적으로 증가하면서 소프트웨어 재사용 부품들은 기업이나 조직체 내에서만 이용하는 인트라넷 기반에서 전 세계가 공유할 수 있는 인터넷 수준으로 급속히 개방되어 왔다. Netlib[33]이나, GAGS[14], ASSET[3], GAMS [15], DARPA STARS [10], alphaBeans[1] 등은 인터넷 상에 개방된 대표적인 소프트웨어 라이브러리나 소프트웨어 저장소들이다.

그러나 1990년 대 초반 이후 소프트웨어 재사용은 소스 코드(Source Code) 기반이라는 한계점 때문에 실용화되는 데에는 어려움을 많이 겪게 되었는데, 컴포넌트(Component) 기술은 침체기에 있던 소프트웨어 재사용에 대한 연구에 다시 활력을 불어넣는 역할을 하였다. 컴포넌트는 바이너리 코드 형태로서 개발되고 있는 소프트웨어에 쉽게 바인딩되어 수행될 수 있는 독립적으로 재사용 가능한 소프트웨어 단위를 말한다. 최근, 컴포넌트 분야의 기반을 이루고 있는 세 가지 주류는 CORBA[27] 컴포넌트와 COM[6] 컴포넌트, 그리고 JavaBeans[26] 컴포넌트들이다. CORBA 컴포넌트는 주로 엔터프라이즈 응용 분야를 기반으로, COM 컴포넌트는 데스크탑 분야를 기반으로, 그리고 JavaBeans 컴포넌트는 인터넷 분야를 위한 컴포넌트로서 활발히 연구, 개발되고 있다. [18] [21] [30]

컴포넌트의 개발은 세계 곳곳에서 이루어지고 있으며, 이들 컴포넌트의 숫자가 기하 급수적으로 증가하고 있는 추세이다. 또한 컴포넌트의 사용자 측면에서도, 소프트웨어 개발 과정 동안 컴포넌트를 사용하려는 개발자들의 숫자도 빠르게 증가하고 있기 때문에 [30], 막대한 숫자의 컴포넌트들 사이에서 컴포넌트 사용자가 가장 적합한 컴포넌트를 검색할 수 있도록 해주는 컴포넌트 검색 엔진이 필수적이다. 사용자가 자신의 요구사항을 만족하는 컴포넌트를 선택하고 사용하기 위해서는 컴포넌트에 대한 특성을 상세히 기술하는 컴포넌트 명세서를 이용해야 하는데, 컴포넌트의 검색은 주로 인터넷 상에서 이루어지기 때문에 웹 브라우저에서도 사용될 수 있는 웹 문서 형태를 컴포넌트 명세서로 이용하는 것이 바람직하다. 현재 인터넷 상에서 웹 브라우저를 통하여 검색에 이용되는 문서의 형태는 HTML (HyperText Markup Language) 문서이다. 그런데, HTML 문서는 정보의 의미를 정확히 표현하는 기능이 거의 없기 때문에 사람이 직접 읽어보고 의미를 파악해야 한다.[9] HTML 검색 엔진인 AltaVista[2]나, WebCrawler[31], InfoSeek[19], HotBot[17] 등은 HTML 문서를 기반으로 검색하기 때문에 문맥의 의미 검색까지 수행할 수 없어 정확도나 신뢰성이 떨어진다.

반면, 최근에 관심과 기대가 커지고 있는 XML (eXtensible Markup Language)은 의미 정보를 기술할 수 있는 장점이 있다. XML은 기업체의 데이타베이스나 응용 프로그램에서 생기는 많은 데이타에 대해 정보 교환을 위한 표준 데이타 형식을 제공해줄 것으로서 기대를 모으고 있으며[22], 상세한 정보와 그 의미를 정의하고 기술할 필요가 있는 컴포넌트 명세서에도 매우 적합하다. 현재 XML은 HTML을 대체하거나 HTML을 보완할 언어로서, 또한 미래의 웹 문서의 표준으로서 기대되고 있다.[16]

본 논문에서는 컴포넌트 사용자가 컴포넌트를 검색하고 컴포넌트를 이해하는데 필요한 정보들을 컴포넌트 제공자들이 기술할 수 있도록 컴포넌트 명세서에 대한 XML DTD (Document Type Definition)를 정의한다. 컴포넌트 제공자들이 본 논문에서 제안한 컴포넌트 명세서에 대한 XML DTD를 따라 컴포넌트 명세서를 작성하면 이들 명세서의 의미 정보를 이용하여 컴포넌트를 정확하게 검색해 줄 수 있는 검색 기법을 제안한다. 컴포넌트 검색 방법으로 XML 컴포넌트 명세서를 기반으로 검색할 수 있는 XML 문맥 기반 검색(context-based search)을 제안한다. 문맥 기반 검색은 컴포넌트의 특성을 나타내는 문맥(context)과 컴포넌트 특성의 실제 값인 용어(term)를 사용하여 사용자가 원하는 컴포넌트의 특성을 정확하게 질의하고 검색할 수 있는 방법이다. 이 검색 방법은 용어-문맥-컴포넌트 명세서 순으로 된 역화일 인덱싱(Inverted File Indexing) 방법을 사용한다. 그리고, 검색 효율은 XML 문맥 기반 검색보다 다소 낮지만, 사용자의 편의를 위하여 비구문 키워드 방법, 브라우징 검색 방법, 퍼싯 검색 방법 (faceted method) 등을 제공한다. 기본 검색 방법인 XML 문맥 기반 검색을 기반으로 다른 검색 방법들이 수행될 수 있도록, 검색 엔진의 구조를 인터페이스 레이어, 질의 확장 레이어, XML 검색 엔진 레이어의 3-레이어(layer)로 나누어 구성한다. 이 구조를 통하여 검색

엔진의 기본인 XML 검색 엔진을 사용하면서도 인덱스 구조나 검색 방법을 크게 변경시키지 않고 검색의 효율도 감소시키지 않으면서 다른 방법들을 효율적으로 수행시킬 수 있다. 또한 컴포넌트 사용자에게 넓은 선택의 폭을 제공하기 위하여, 스스로 인터넷을 돌아다니며 본 검색 시스템의 표준이 아닌 다른 XML DTD에 기반을 둔 컴포넌트 명세서를 검색하고 등록할 수 있는 기능을 제공한다. 다양한 형태의 컴포넌트 명세서를 등록하고 이들에 대한 검색이 가능하도록 하는 인덱스 스킴을 제안한다.

검색 엔진에서는 검색 용어와 문맥에 대한 유의어 사전, 상·하위 개념 사전을 퍼지 테이블의 형태로 제공하여 검색 시에 이용한다. 퍼지 개념의 사용은, 값의 존재 유무와 관련성의 존재 유무만을 검사하는 불리언 검색 방법 대신에 불확실한 관련성 정도까지 나타낼 수 있기 때문에 검색의 성능과 사용자의 만족도를 높이는데 목적을 두고 있다.

본 논문은 다음과 같이 구성된다.

제 2장에서는 컴포넌트의 검색을 위한 관련 연구에 대해 살펴보고, 제 3 장에서는 컴포넌트에 대한 정보를 기술하기 위하여 컴포넌트 명세서에 포함되어야 할 내용과 XML 컴포넌트 명세서 사례를 기술한다. 제 4장에서는 본 검색 엔진의 구조 및 인덱싱 방법, 퍼지 개념을 이용한 질의 확장 방법, 그리고 검색 방법 등에 대해 기술한다. 제 5 장에서는 본 검색 엔진을 통한 검색 방법과 HTML 검색 방법의 검색 예제를 통하여 비교해 보고, 이들의 검색 성능을 비교해 보기 위한 실험과 실험 결과가 기술된다. 마지막으로 제 6장에서는 결론과 향후 연구에 대하여 기술하며 부록에 컴포넌트 명세서에 대한 XML DTD가 나와 있다.

2. 관련 연구

2.1 기존의 소프트웨어 재사용 라이브러리의 검색 방법

기존의 소프트웨어 재사용 라이브러리들에서 사용하던 소프트웨어 부품의 분류, 검색 방법으로는 비구문 키워드(Free Text Keyword) 방식[13], 십진 분류 방법[20], 퍼싯 분류 방법(Faceted Method)[28], 시맨틱 넷(Semantic Net) 분류 방법[32] 등이 있다. 비구문 키워드 방법[13]의 가장 큰 장점은 기법의 단순함과 인덱싱을 자동적으로 처리할 수 있다는 점이다. 그러나, 각 키워드에 시맨틱(semantic)이 없기 때문에 키워드들이 갖는 의미를 유추해 낼 수 없다는 단점을 갖는다. 십진 분류 방식 및 브라우징 검색 방법[20]은 재사용 라이브러리 사용자들이 가장 많이 의존하고 있는 방법이지만, 계층 구조상의 속성을 표현할 수 없고 새로운 부품들을 등록하기 어렵기 때문에 확장성이 좋지 않고 정확한 조건을 이용한 검색은 불가능하다. 퍼싯 방법(Faceted Method)[28]은 인덱스 전체 구조를 바꿀 필요 없이 새로운 퍼싯을 추가할 수 있기 때문에 확장성은 좋지만, 정해진 퍼싯 이외의 상세한 정보는 제공할 수 없고, 어떤 항목이 퍼싯으로 만들어져야 하는지를 결정하기 위해서는 정확한 도메인 분석이 요구된다. 시맨틱 넷 방법[32]은 적절한 부품을 효율적으로 찾을 수 있는 장점이 있지만 자세한 도메인 분석에 따른 많은 노력이 요구된다는 단점이 있다. 이들 방법들은 각각의 장단점을 가지고 있기 때문에, 사용자들이 자신이 선호하는 방법으로 컴포넌트를 검색할 수 있도록 다양한 검색 방법을 지원할 필요가 있다.

2.2 기존 웹 검색 엔진을 통한 컴포넌트 검색 및 제한점

현재, 인터넷 상에서 컴포넌트를 검색하는 가장 쉽고 일반적인 방법은 AltaVista[2], WebCrawler[31], InfoSeek[19], HotBot[17] 같은 기존의 웹 검색 엔진들을 사용하는 것이다. 이들은 HTML 문서를 기반으로 검색을 수행하기 때문에 주로 키워드 검색 방법만이 지원된다. 이 검색 방법은 검색하는 키워드의 존재 유무만으로 검색하기 때문에 키워드에 의미를 부여하여 검색할 수가 없다. 이러한 문제점 때문에, 검색 결과에 대한 정확도(precision)가 매우 낮다. 그러므로 정확성을 요구하는 컴포넌트 검색에 일반 웹 문서 검색 엔진을 이용하는 것은 한계가 있다.

카네기 멜론 대학에서 컴포넌트만을 검색해 주는 검색 에이전트인 Agora [29]가 연구, 개발되고 있다. Agora는 JavaBeans 컴포넌트, CORBA 컴포넌트, COM 컴포넌트를 검색해 주는 전문적인 컴포넌트 검색 엔진이다. 그러나, Agora도 HTML 기반의 AltaVista의 검색 엔진인 SDK를 사용하고 있기 때문에, 일반 문서 검색 엔진보다는 컴포넌트에 대한 정보를 추가하여 검색할 수는 있지만, 키워드가 갖는 의미를 정확하게 파악하여 검색하지는 못하기 때문에 검색 성능을 크게 향상시키지 못한다.

2.3 XML 검색 방법을 통한 컴포넌트 검색

XML은 HTML과 SGML (Standard Generalized Markup Language)과 함께 일반 마크업 언어의 일종으로, HTML에서의 간결한 데이타 표현 형식과 웹 브라우저에서의 뛰어난 표현 능력뿐만 아니라 SGML에서의 뛰어난 데이타 표현 능력을 동시 갖도록 설계되었다.

XML 문서는 태그(tag), 요소(element), 속성(attribute), 처리 명령, 주석문 등으로 구성된다. 태그는 문맥을 나타내는 것으로 문서의 의미를 표현할 수 있게 해 준다. 요소는 태그와 그 사이의 텍스트를 일컫는다. 태그는 속성을 가질 수 있으며 처리 명령은 파싱을 위하여 파서에 전달되는 정보이며 주석문은 문서에 대한 설명을 나타낸다. XML 문서의 형태는 요소(element)로 구성되어 있으며, 요소는 <tag> text </tag> 형태를 갖는다. 여기서 text는 요소의 형태를 반복적으로 취할 수 있기 때문에, XML 문서는 트리 형태의 구조적 표현이 가능하다.

기존의 XML 검색 엔진은 거의 개발 완성된 것이 없으나, 초기 단계의 XML 검색 엔진으로서 Milner가 개발한 SCOOBS[24, 25]가 있다. 이것은 각 용어들을 문맥인 요소(element)들의 링크와 연결시켜 나타내고 이들과 관련된 문서를 연결시키는 역 인덱스 화일(Inversed Index File) 구조를 가진다. 이 검색 엔진은 HTML 검색 방법보다 검색 성능이 좋지만, 컴포넌트 검색을 지원하는 검색 엔진이 아니기 때문에 컴포넌트의 특성에 대한 정보 및 검색 방법은 지원하고 있지 않다. 또한 문맥이나 용어에 대한 유의어 사용이 미흡하기 때문에, 정확하게 일치하는 문맥이나 용어를 사용하지 않는 경우에는 검색 결과가 좋지 못하다.

3. 컴포넌트 검색을 위한 XML 컴포넌트 명세서

3.1 컴포넌트 명세서

컴포넌트 명세서에는 컴포넌트를 인터넷 상에 개방하면서 그 컴포넌트에 대한 검색을 하거나 컴포넌트 이해하고자 하는 사용자들에게 도움이 되는 정보들이 상세하게 기술되어야 한다. 본 논문에서 제안된 컴포넌트 검색 엔진은 컴포넌트 공급 업체(vender)들이 본 검색 시스템에 컴포넌트를 등록할 때 등록 정보를 입력하여 생성되는 XML 표준 컴포넌트 명세서와, 컴포넌트 공급 업체들이 직접 작성한 XML 컴포넌트 명세서나, 등록 엔진이 스스로 돌아다니면서 등록하는 XML 일반 컴포넌트 명세서를 통해 컴포넌트 검색을 수행할 수 있다. 현재까지 컴포넌트 명세서를 위한 XML DTD 표준이 정의되어 있지 않기 때문에, 본 검색 엔진에서 정의한 컴포넌트 XML DTD를 따르는 명세서와 다른 컴포넌트 공급 업체에서 정의한 다양한 종류의 XML 컴포넌트 명세서를 모두 등록하고 검색에 사용할 수 있도록 지원한다.

Lassing은 컴포넌트의 사용자와 공급자가 컴포넌트에 대한 거래(deal)를 하기 위하여 서로 의사 소통할 수 있는 계약서(software contracts)를 제안했다. 소프트웨어 계약서에는 컴포넌트의 기능(functionality), 운영체제, 언어, 시스템과 같은 환경(environment), 메소드(method) 및 입력/출력 매개변수(input/output parameter), 성능(performance), 제약점(limitations), 데이타베이스 사용 등과 같은 서비스 수준 등이 포함되어야 한다고 제안했다.[21] 본 검색 시스템에서는 Lassing이 제안한 소프트웨어 계약서에 필요한 정보와 사용자들이 컴포넌트를 검색하고 사용할 때 필요한 항목들을 합하여, 현재의 컴포넌트 기술이 컴포넌트 검색에 미치는 영향에 대하여 최대한 반영할 수 있도록 컴포넌트의 특성을 정의한다. 컴포넌트의 검색을 위하여 컴포넌트 명세서에 기술되어야 할 컴포넌트의 특성을 다음과 같다.

컴포넌트 명세서에는 공급 업체 별로 컴포넌트 검색을 할 수 있도록 컴포넌트 공급 회사 정보를 제공되며, 컴포넌트의 이름이나 버전, 도메인, 특징 등에 관한 컴포넌트의 일반 정보, 컴포넌트의 개발 환경 및 사용 환경, 사용 도구 및 사용되는 데이타베이스, 컴포넌트의 종류 등 현재 기술적인 측면을 검색에 이용할 수 있도록 하기 위한 기술적 정보(technical information)가 제공된다. 또한 컴포넌트의 인터페이스 정보로서 컴포넌트의 기능과 특성을 가장 상세하게 기술한 정보, 즉 인터페이스의 메소드(method), 성질(property), 이벤트(event) 등의 상세한 정보를 알 수 있으며 이에 대한 검색도 가능하다. 시스템 요구 사양 정보는 컴포넌트를 수행하는 시스템에 대한 최소 사양을 나타내며, 컴포넌트의 가격이나 라이센스에 관한 정보로도 검색할 수 있도록 제공된다. 컴포넌트의 버전 정보 및 개선점 등에 관한 정보와, 컴포넌트들 사이의 구성 관계를 나타내는 형상 관리 정보가 제공된다.

3.2 컴포넌트 명세서를 위한 XML DTD(Document Type Definition)

XML 표준 컴포넌트 명세서를 검색 시스템에서 이용하기 위하여 검색에 필요한 정보를 잘 기술하는 컴포넌트 명세서를 제공하기 위해서는 XML DTD(Document Type Definition)를 잘 정의하여야 한다. 앞 절에서 정의한 컴포넌트 명세서가 가져야 할 정보들을 의미적 표현이 가능한 XML의 요소(element)로 정의한다. 본 시스템에서 정의한 표준 컴포넌트 명세서에 대한 XML DTD의 일부가 부록에 나와 있다.

부록의 컴포넌트 명세서를 위한 XML DTD의 내용 중에서, <!ELEMENT Component (Vendor, Information, Requirements, License, Price, Config-

uration)>는 컴포넌트가 가져야 하는 정보들로서 컴포넌트 공급 회사 정보, 컴포넌트에 대한 일반 정보와 기술적 정보, 시스템 요구사양, 라이센스 정보 및 가격 정보, 컴포넌트의 버전이나 구성 정보 등을 요소(element)로 정의한 것이다. 이 중 인터페이스에 대한 좀 더 상세한 정보는 <!ELEMENT Interface (InterfaceName, Method+, Property*, DataType*, Exception*, Event*)>에서 살펴 볼 수 있다. 인터페이스의 이름, 메소드, 특성, 데이타 타입, 예외, 이벤트 등의 부분을 정의하고 있으며, XML에서는 구조적(structured)인 트리 형태로 세부 정보를 더 상세하게 정의해 줄 수 있다. 컴포넌트 명세서를 위한 XML DTD는 이런 형식으로 컴포넌트 사용자가 검색을 할 때나 컴포넌트에 대한 정보를 기수하기 위하여 3.1 절에서 언급된 정보들을 정의한다.

3.3 XML 컴포넌트 명세서의 사례

그림 1은 부록에 있는 컴포넌트 명세서를 정의하는 XML DTD를 따르는 컴포넌트 명세서의 하나의 예로서 XML 문서 검증 도구인 Internet Explore 5.0의 DOM에 의하여 검증된 문서(valid document)이다. 태그는 컴포넌트가 가지는 특성들을 의미적으로 나타내기 위하여 사용되며 시작 태그와 닫는 태그 사이에 있는 내용이 태그가 나타내는 문맥 의미를 갖는 정보이다.

```
<Component id="cjb0001">
 <Vendor href="http://www.alphaworks.ibm.com">
  IBM Alphaworks</Vendor>
 <Information>
  <General_Info>
   <ComponentNamehref= "http://www.alphaworks.ibm.com/
alphabeans/XMLTreeDiff1.2"> XMLTreeDiff </ComponentName>
    <ComponentVersion>1.2</ComponentVersion>
   <ComponentDomain>XMLTreeDiff is a set of Java beans
designed to perform <Function id= "f01">fast differentiation
</Function> and <Function id="f02">update</Function> of
<Object id="o01"> XML documents</Object>
   </ComponentDomain>
            ...
  </General_Info>
  <Technical_Info>
          ...
    <Language type="Java">Java</Language>
   <OS type="undefined">Any Java 1.1 (or greater) enabled
platform </OS>
   <ComponentType type="javabeans">javabeans suite</Com-
ponentType>
     ...
  </Technical_Info>
  <Interface id="i01">
   <InterfaceName>XMLTreeDiff
   </InterfaceName>
   <Method id="m03">
    <MethodName>diff</MethodName>
    <Declaration>public static Document diff(Child root1, Child
     root2, boolean use XUL)
    </Declaration>
     <ReturnType>Document</ReturnType>
            ...
     </Method>
     <Property id="p01">
        <PropertyName>factory
     </PropertyName>
    <Declaration>public static Document factory
   </Declaration>
            ...
  </Property>
 </Interface>
</Information>
 ...
<Configuration>
 <VersionInfo>
    <OldVersion >
     <ComponentName href="http://www.alphaworks.ibm.com/
XMLTreeDiff1.1"> XMLTreeDiff1.1</ComponentName>
        ...
    </OldVersion>
 </VersionInfo>
 <RelatedComponents relative_type= used_by >
   <ComponentName href="http://www.alphaworks.ibm.com/
alphabeans/DOMGenerator">DOMGenerator
   </ComponentName>
 </RelatedComponents>
</Configuration>
</Component>
```

그림 1 XML 컴포넌트 명세서의 예

본 검색 엔진의 기반이 되는 문맥 기반 검색에서는 문맥(context)와 용어(term)에 의하여 검색을 수행하는데, 문맥은 용어가 가지는 의미를 나타내는 태그(tag)이고, 용어는 태그들 사이의 단어를 나타낸다. 예를 들어, 그림 1에서 XMLTreeDiff1.1의 새로운 버전을 찾고자 할 때 사용자는 {<context> Old version, <term> XMLTreeDiff1.1} 라는 질의를 줄 수 있다. XML 문맥 기반 검색에서는 <OldVersion>이라는 태그가 있기 때문에 옛날 버전에 대한 새로운 버전의 컴포넌트를 검색해 줄 수 있는 반면, HTML 기반의 검색 방법에서는 'Old Version'이라는 문맥은 줄 수 없고, 단지 'XMLTreeDiff1.1'이라는 키워드만으로 질의할 수 있기 때문에 이에 대한 검색 결과는 'XMLTreeDiff1.1'라는 키워드가 들어가는 것은 새로운 버전이든 본 버전이든 이것을 사용하고 있는 다른 컴포넌트이든 상관없이 모

두 검색해 주게 된다. 그러므로 검색의 정확도가 상당히 떨어지는 결과를 가져온다.

4. XML 기반 컴포넌트 등록 및 검색

4.1 컴포넌트 검색을 위한 검색 엔진의 구조

컴포넌트 검색을 위한 컴포넌트 검색 엔진은 3-레이어(layer)로 구성된다. 가장 상위의 레이어는 컴포넌트를 검색하는 사용자에게 제공되는 다양한 컴포넌트 검색 방법을 위한 인터페이스 레이어(interface layer)이다. 인터페이스 레이어에서는 키워드 검색, 퍼싯 검색, 계층구조에 의한 브라우징 검색, XML 문맥 기반 검색 등 다양한 검색 방법에 대한 인터페이스를 제공한다. 인터페이스 레이어의 하위 레이어는 컴포넌트 검색을 위한 질의 확장 레이어이다. 질의 확장 레이어에서는 컴포넌트 검색을 위하여 본 검색 엔진의 기반이 되는 XML 검색 엔진과의 중간 역할을 수행한다. 문맥(context)과 용어(term)에 대한 유의어나 상·하위 개념으로 확장하는 기능과, 키워드 검색, 브라우징 검색, 퍼싯 검색의 검색 질의를 XML 문맥 기반 검색 질의로 변환하는 기능을 수행한다. 질의 확장 레이어에서는 각 검색 방법들이 XML 검색 엔진 위에서 수행될 때, 정보의 손실을 최소화하면서 검색을 수행할 수 있도록 지원한다. 가장 하위의 레이어가 실제 검색 기능을 수행하는 XML 검색 엔진 레이어이다. 이 레이어는 XML 문맥 기반 검색을 수행하는 검색 엔진이다. XML 검색 엔진으로 들어오는 질의(query)는, 인터페이스 레이어에서 지원하는 다양한 검색 방법에 대한 사용자 질의를 받아들여 질의 확장 레이어를 통하여 확장된 후, 문맥 기반 검색 질의 형태로 들어온다.

문맥 기반 검색

```
<C_Query> := <C_Query> AND <Context Query> |
             <C_Query> OR <Context Query> |
             <Context Query>
<Context Query> := <Context> <Term> <Weight>
<Context> := "Reserved Character String"
<Weight> := Real value between 0 and 1
<Term>  := "Character String"
```

키워드 검색

```
< K_Query> := <K_Query> AND <Term> |
              <K_Query> OR <Term>|<Term>
<Term>  := "Character String"
```

브라우징 검색

```
<B_Query> := <B_Query> AND <Browsing_Query>|
             <B_Query> OR <Browsing_Query>|
             <Browsing_Query >
<Browsing_Query> := <Attribute><Category>
<Attribute> := "Reserved Character String"
<Category> := "Reserved Character String"
```

퍼싯 검색

```
<F_Query> := <F_Query> AND <Facet Query>|
             <F_Query> OR <Facet Query>|
             <Facet Query>
<Facet Query> := <Facet> <Term> <Weight>
<Facet> := "Reserved Character String"
<Weight> := Real value between 0 and 1
<Term>  := "Character String"
```

그림 2 각 검색 방법의 질의 형식

그림 2는 각 검색 방법의 질의 형식을 보여주고 있다. 문맥 기반 검색은 문맥(context)과 검색 용어(term)를 질의로 주고 컴포넌트를 검색한다. 문맥(context)이란 태그들을 일컬으며, 용어(term)는 태그 사이의 키워드를 일컫는다. 컴포넌트를 검색할 때, 컴포넌트의 특성을 나타내는 문맥(context)과 그 문맥 내의 값을 나타내는 용어(term)를 사용하여 검색하면 키워드 검색 방법을 이용하는 것보다 검색 성능이 좋아진다. 각 문맥과 검색 용어 쌍에 대하여 언어로 표현된 중요도 가중치(linguistic weight)를 줄 수 있다.

키워드 검색 방법은 HTML 기반의 검색 방법과 같은 방법으로서 XML 문맥 기반 검색에서 문맥에 해당하는 부분이 빠진 형태의 질의이다. 검색 성능은 떨어지지만 사용자가 컴포넌트 검색 시에 문맥을 사용하는데 익숙하지 못한 경우에 사용할 수 있다.

계층 구조에 의한 브라우징 검색은 미리 정의된 카테고리에 의하여 원하는 검색을 수행하는 십진 분류에 의한 검색 방법이다. 본 검색 시스템에서는 기존의 십진 분류 방식을 확장하여 여러 가지 특성(Attribute)에 대한 계층 분류와 검색을 할 수 있도록 지원한다. 전문가에 의해 정의된 특성과 그 계층 분류를 이용하여 사용

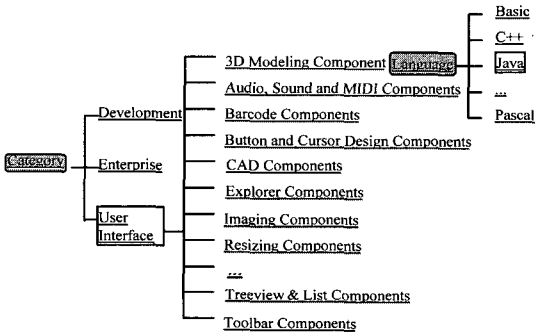

그림 3 계층구조에 의한 브라우징 검색 방법

자는 원하는 특성을 따라 계층 구조를 따라 브라우징 한다. 그 조건 상에서 다른 브라우징 속성을 선택하면 현재의 검색된 결과들을 대상으로 두 번째로 선택된 요소(속성)에 대한 브라우징을 수행한다.(그림 3)

퍼싯 검색 방법에서는 다양한 퍼싯(facet)을 정의하여 사용자가 정확한 검색을 수행할 수 있도록 지원하며, 정의되지 않은 퍼싯에 대한 검색도 수행할 수 있도록 문맥 기반 검색 방법을 동시에 제공하여 퍼싯 방법이 갖는 검색 범위에 대한 제한을 해결한다. 정의된 퍼싯은 다음과 같다.

- 컴포넌트 도메인/기능/객체/컴포넌트 타입/컴포넌트 인터페이스(메소드/성질/이벤트)/컴포넌트 개발 언어/컴포넌트 공급 업체/컴포넌트 가격/컴포넌트 개발 플랫폼/사용 도구

퍼싯 방법은 검색 용어에 대한 문맥을 지원하지만 시스템에서 미리 정의한 퍼싯에 한하여 질의할 수 있다는 단점이 있다. 그러므로, XML 문맥 기반 검색 방법보다 검색할 수 있는 문맥의 범위가 제한된다. 하지만, 정의된 퍼싯을 사용하여 컴포넌트를 검색할 수 있다는 편리함을 제공하므로, 문맥(context)과 용어(term)를 사용자가 직접 정의하고 검색하는 것을 부담스럽게 생각하는 사용자에게는 퍼싯 검색 방법이 사용하기 쉽다는 장점이 있다.

4.2 검색 질의에 대한 퍼지 확장

불리언 모델(Boolean Model)을 사용하는 기존의 검색 엔진들은 키워드의 존재 유무, 또는 그 키워드의 빈도수를 기반으로 검색 결과의 순위를 결정한다. 그러나 검색 대상의 특징을 의미 있게 기술하고 이를 명확하게 검색할 수만 있다면, 검색하는 키워드의 빈도수는 중요하지 않으며, 빈도수에 상관없이 얼마나 관련성이 있는지를 찾아주는 것이 더 효율적인 방법이다.[7, 11] 본 인덱스 스킴에서는 검색하는 값의 존재 유무만을 가지고 검색을 수행하는 불리언 모델(BOOLEAN Model)의 단점을 보완하기 위하여 퍼지 모델(Fuzzy Model) 기반의 인덱스를 사용한다. 퍼지 모델은 모든 용어에 대한 관련성을 0과 1 사이의 값으로 정의해 줄 수 있는 퍼지 관련성 테이블(Fuzzy Relevance Table)로 정의한다. 이것은 퍼지 개념 네트워크(Fuzzy concept network)[8]에 기반을 둔 기법으로서, 존재 유무나 관련성 유무만을 판단해 주는 불리언 모델의 단점을 보완해 주어 검색 성능을 향상시킬 수 있다. 표 1은 퍼지 관련성 테이블의 인덱스 구조이다. 용어(term)와 문맥에 대하여 각각 퍼지 유의어 테이블과 퍼지 상·하위 개념 테이블을 제공한다.

표 1 퍼지 관련성 테이블

	t_1	t_2	...	t_n
t_1	w_{11}	w_{12}	...	w_{1n}
t_2	w_{21}	w_{22}	...	w_{2n}
...	...	...	...	...
t_n	w_{n1}	w_{n2}	...	w_{nn}

4.3 XML 컴포넌트 명세서 등록 및 인덱스 스킴

컴포넌트들이 검색 시스템에 등록되는 방법은 세 가지가 있다. 컴포넌트 제공자가 등록 인터페이스를 통하여 정보를 입력하면 본 검색 시스템에서 정의한 컴포넌트 명세서의 표준 형식을 갖게 되며, 자신들이 정의한 컴포넌트 명세서를 DTD와 함께 위치 정보로 등록하거나 검색 시스템이 돌아다니면서 컴포넌트에 대한 명세서를 수집하여 등록할 때는 각 컴포넌트들은 독자적인 XML 컴포넌트 문서의 형태를 갖는다. 컴포넌트 검색 시스템에서는 다양한 형태의 XML 컴포넌트 명세서를 다루어야 하기 때문에 이들을 잘 분석하여 인덱싱 하여야 한다. 키워드 방식은 문맥이 없는 문맥 검색 방식으로, 브라우징 방식은 분류 특성(attribute)은 문맥으로, 그 특성 내의 선택된 카테고리는 용어(term)로 변경시켜 문맥 기반 검색의 형태로 인덱스의 추가나 변경 없이 쉽게 변환시킬 수 있다. 본 절에서는 본 시스템의 기본 검색 방법인 문맥 기반 검색 방법을 위한 인덱스 스킴과 인덱스의 확장이 필요한 퍼싯 방법에 대해서 기술한다.

1) XML 문맥 기반 검색(Context-based Search)을 위한 인덱스

여러 형태의 XML DTD를 기반으로 한 다양한 형태의 XML 컴포넌트 명세서가 등록되기 때문에, 실제로 등록되는 컴포넌트 명세서에서 나타나는 문맥(context)과 용어들은 상당히 많은 종류가 생성된다. XML 문맥 기반 검색은 문맥(context)과 검색 용어(term)를 모두 검색하여 두 가지가 동시에 만족되는 컴포넌트를 검색한다. 용어, 문맥, 그리고 컴포넌트 명세서인 문서에 대한 인덱싱을 위하여 역화일(inverted file)[13] 인덱스 스킴을 사용한다. 문맥 기반 검색을 위하여 기본적으로 제공되는 용어-문맥-컴포넌트 명세서의 역 인덱스 화일 형태는 그림 4의 예제와 같다. 예를 들어 컴포넌트의 종류가 'CORBA'인 경우 맨 앞에 'CORBA'라는 용어가

삽입되고, 이에 대한 문맥을 나타내는 <type>, <component> 리스트가 용어 뒤에 삽입된다. 마지막으로 이 용어와 문맥을 포함하는 컴포넌트 명세서 ID를 저장한다. 그림 2에 있는 XML 문맥 기반 검색의 질의 형식처럼 사용자가 문맥(context), 용어(term), 중요도 가중치(weight)를 질의로 주어 검색을 수행하면 그 문맥과 용어를 동시에 만족하는 컴포넌트가 존재하는지를 검색한 후 관련된 컴포넌트 명세서 리스트를 결과로 돌려준다.

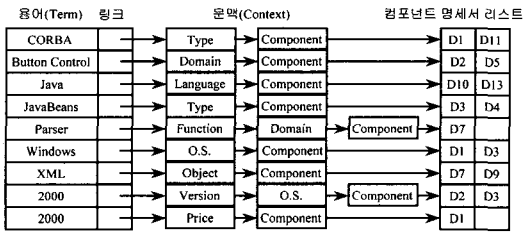

그림 4 용어-문맥-컴포넌트 명세서 역 화인 인덱스

2) 퍼싯 검색을 위한 인덱스 확장

퍼싯 검색은 시스템에서 미리 정의한 퍼싯들에 대한 검색을 수행하기 때문에 미리 정의된 퍼싯과 XML 문서에서의 다양한 문맥에 대하여 관련성이 먼저 정의되어야 한다. 검색 시스템에서 정의된 퍼싯을 표준 퍼싯이라 하면, 표준 퍼싯은 XML 문맥 기반 검색의 문맥(context)에 해당하며, 퍼싯의 값(value)은 용어(term)에 해당한다. 형태가 다른 XML 컴포넌트 명세서를 모두 검색하기 때문에 모든 문맥(context)은 표준 퍼싯과 일치하지 않는다. 그러므로, 검색 시스템은 인터넷 상에서 찾아진 컴포넌트 명세서와 그에 따른 XML DTD를 분석하여 새로운 문맥과 퍼싯의 관련성 정보를 전문가가 정의하도록 한다. DTD 마다 생성되는 문맥과 퍼싯의 관련성 정도를 나타내는 정보로 {<c_i, f_i, w_i>}의 형태의 정보를 갖는데, c_i는 문맥(contex)을, fi는 퍼싯을, wi는 문맥과 퍼싯 사이의 관련성 정도를 나타낸다. 이 정보를 기반으로 퍼싯-문맥 그룹 테이블에 새로운 문맥과 관련성 정보가 추가되며, 그 예가 표 2에 나타나 있다. 본 검색 시스템에서는 퍼싯 검색 방법에 대해, 표준 퍼싯에 대한 검색과 함께 문맥 기반 검색을 추가로 제공하여 사용자가 컴포넌트의 특성에 대한 검색 조건을 쉽게 질의하면서도 퍼싯 방법만 지원하는 기존의 재사용 라이브러리에서 한정된 정보에 대한 검색만 수행할 수 있는 단점을 보완하여 더욱 다양하고 정확한 검색 조건으로 컴포넌트 검색을 할 수 있도록 지원한다.

표 2 표준 퍼싯 'Price'와 문맥(태그) 관련도 테이블

문맥(태그)	퍼싯-문맥 관련도	DTD id
Component_Price	1.0	1
Description/Cost	1.0	5
Rate	0.9	3
Figure	0.7	2
Charge	0.8	7
Expense	0.8	9

4.4 XML 문맥 기반 검색을 통한 컴포넌트 검색

XML 문서는 문맥을 중심으로 기술되어 있기 때문에 XML 문서 내에 나타난 용어의 빈도수에 의해 문서와 용어와 관련성 정도를 결정하는 것은 타당하지 않다. 문맥과 용어를 함께 검색하여 문맥과 용어를 동시에 만족하면 용어와 문맥은 문서와 관련성이 있고 그렇지 않으면 용어와 문맥은 문서와 관련성이 없다고 결정할 수 있다. [11]에서 문서를 구조적인 부분과 비구조적인 부분으로 나누어 가중치를 계산한 방법이 연구되었다. 그러나 XML 문서는 모두 구조화된 형태를 갖고 있기 때문에 용어의 빈도수를 적용해서 관련성을 계산한다는 것은 의미가 없다. 그러므로 본 검색 방법에서는 문서와 검색 용어는 검색 용어가 속한 문맥 속에서 검색되기 문서와 관련성을 1로 부여하고 검색되지 않으면 0으로 부여하는 방식을 사용한다. 그러나 앞 절에서 언급한 바와 같이 사용자가 질의에서 사용하는 용어와 문맥에 대한 유의어나 상·하위 관계에 대해서는 관련성 정도를 0과 1 사이의 퍼지 값으로 부여하여, 관련성 정도를 더 정확하게 나타낼 수 있도록 한다. 사용자는 자신의 질의 각 부분에 대해 언어로서 중요도(linguistic fuzzy importance)를 지정할 수 있으며, 이런 조건을 기반으로 하여 결과의 순위를 결정할 때도 불리언 방법이 아닌 퍼지 방법을 사용한다. 결과의 순위를 결정하기 위하여 Fox에 의하여 제안된 P-norm 모델[12]을 사용한다.

검색 과정에서 P-norm 모델을 사용하기 위하여 P-norm에 대하여 간략히 기술한다.

문서 D와 용어 A1,A1,...,An 과의 관련성을 각각 $d_{A1}, d_{A2}, ..., d_{An}$ 이라고 하자. 그리고, 사용자의 질의에서 사용된 각각의 용어 대해 사용자가 중요도를 가중치로 준 질의 형태가 다음과 같다고 하자. a1, a2, ..., an은 각 용어 A1, A2, ..., An에 대해 사용자가 주는 용어에 대한 중요도 가중치이다.

Q_{OR} = (A1, a1) OR (A2,a2) OR ... OR (An, an)이고

Q_{AND} =(A1,a1) AND (A2,a2) AND ... AND (An, an) 일 때,

$$SIM(Q_{OR},D)=\sqrt[P]{\frac{a_a^P d_{A1}^P + a_a^P d_{A1}^P + ...a_a^P d_{A1}^P}{a_1^P + a_2^P + ... + a_{n1}^P}}$$

$$SIM(Q_{AND},D)$$

$$=1-\sqrt[P]{\frac{a_1^P(1-d_{A1})^P + a_2^P(1-d_{A2})^P + ... + a_n^P(1-d_{An})^P}{a_1^P + a_2^P + ... + a_{n1}^P}},$$

$$SIM(Q_{AND},D)=1-SIM(Q,D)$$

이다.

사용자의 질의에 대해 문맥 기반 검색 방법으로 검색하는 과정을 다음 알고리즘에서 기술한다.

(알고리즘) XML 문맥 기반 검색 과정

단계1: 사용자 질의 확장

단계 1.1: 퍼지 용어 유의어 테이블에서 임계치보다 큰 유의어를 선택하여 용어를 확장한 다.

단계 1.2: 퍼지 문맥 유의어 테이블에서 임계치보다 큰 유의어를 선택하여 문맥을 확장한다.

단계2: 확장된 문맥과 용어 조건들에 대해 각각 곱집합을 한다.

단계3: 곱집합 된 (문맥, 용어)의 각 쌍을 만족하는 문서를 검색한다.

단계 3.1: 각 (문맥, 용어) 조건의 쌍을 동시에 만족하는 컴포넌트 명세서를 용어-문맥-컴포넌트 명세서 인덱스에서 검색한다.

단계 3.2: 각 질의 조건을 만족하는 컴포넌트 명세서를 검색하고 각 컴포넌트 명세서와 질의와의 관련성을 계산한다.

단계4: 모든 세부 질의에 대하여 생성된 결과에 대한 순위(ranking)를 계산한다.

단계 4.1: 각 컴포넌트 명세서마다 생성된 컴포넌트 명세서와 질의의 관련성 가중치와 사용자가 주는 질의의 중요성 가중치를 질의 문서의 관련성 정도를 P-norm 방법에 의하여 계산한다.

5. 검색 사례 및 성능 평가

5.1 HTML과 XML 기반의 검색 사례 및 비교

다양한 형태의 XML 컴포넌트 명세서를 통하여 의미를 갖는 태그로 인식하여 검색해 주는 경우의 사례와, HTML 문서 기반의 검색에서 정확히 검색해 주지 못하는 것을 XML 기반 문맥 기반 검색에서 본 검색 엔진이 검색해 주는 경우 등에 대한 사례를 통하여 XML 검색이 HTML 검색보다 검색 성능이 우수함을 보여주고 있다.

1) 다양한 형태의 컴포넌트 명세서를 통한 검색 사례

컴포넌트의 명세서가 표준 컴포넌트 명세서인 경우와 일반 컴포넌트 명세서를 사용하여 다른 태그 이름을 가지고 정의되어 있을 경우에 다음과 같이 검색이 가능함을 보여준다. 표준 컴포넌트 명세서로부터 해당 컴포넌트의 도메인에 대한 검색을 수행할 때 컴포넌트 명세서 중에 <Function> 태그가 정의되어 있지 않은 경우에, 인터페이스 명세서의 <Method> 태그와 그 하위 태그들을 검색하여 관련 정보를 찾아낼 수 있음을 보여준다. 뿐만 아니라 <Function> 태그와 관련된 다른 태그들을 통해서도 검색할 수 있다. 이는 전문가에 의해 정의된 퍼지 문맥 유의어 테이블을 이용하여 검색이 가능하다. 문맥 <Domain>과 <Function> 사이의 연관성이 퍼지 문맥 유의어 테이블에 정의되어 있는 관계임을 가정하고 있다.

사용자 질의가 (<context>Function, <terms> sort AND int , <weight> 0.3)이고, SortBean이라는 컴포넌트의 인터페이스 명세서의 내용의 일부가 다음과 같을 때 예제의 명세서에는 <Function> 태그가 기술되어 있지 않기 때문에 <Method/Description> 태그를 검색하여 컴포넌트의 Function을 검색해 줄 수 있다.

```
<Method>
  <MethodName> sortIntArray
  </MethodName>
  <Declaration>public int[] sortIntArray(int[]
arrayOfIntegers)</Declaration>
  <ReturnType>int[]</RuturnType>
  <ParameterType order= 1> int[]
  </ParameterType>
  <Description>This is the method which sorts integer
    array.</Description>
</Method>
```

2) HTML 키워드 기반 검색과 문맥 기반 검색의 성능 비교 사례

가격이 free인 component를 검색하는 질의로서, (<context> Price, <terms> free , <weight>, Important) 를 수행할 때, HTML 명세서에서 가격이 아닌 플랫폼에 대한 다음 내용을 포함한다고 하자.

PLATFORM : This component is ***free*** from platform, because it is written with java.

대응되는 내용으로 XML 명세서에는 다음의 태그와 내용을 포함한다고 하자.

<Technical_Info>
<Platform type= JVM > This component is ***free*** from platform. </Platform>·
</Technical_Info>

퍼지 문맥 유의어 테이블에서 'price'와 'platform' 사이에는 관련성이 거의 없는 것으로 정의되어 있다고 가정하자. 앞의 질의에 대하여 HTML 기반의 검색 엔진에서는 가격이 free로 잘못 인식되어 검색 결과로 찾아질 수 있는데 반하여, XML 명세서에 대한 문맥 기반 검색의 경우에는 태그 내용이 platform에 대한 free를 나타내기 때문에 가격이 free인 경우로는 검색되지 않는다.

5.2 XML과 HTML의 검색성능 평가실험

본 절에서는 XML 검색 엔진을 사용할 때와 HTML 기반 검색 엔진을 사용할 때의 검색 성능을 비교하기 위하여, 검색 효율에 대한 실험 결과를 보인다. 일반적으로 검색 엔진의 성능을 비교하기 위하여 정확도(Precision)와 리콜 (Recall)을 사용한다. 정확도는 관련 있는 문서 중에서 추출된 문서의 비율을 나타내며, 리콜은 추출된 문서 중에 관련 있는 문서의 비율을 나타낸다.

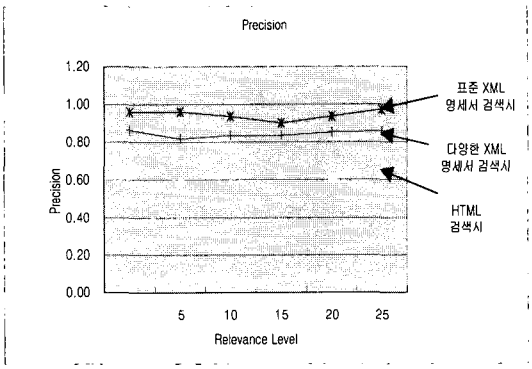

그림 5 HTML 문서 검색과 표준 XML 문서 검색 및 다양한 XML 문서 검색의 정확도 비교

본 실험에서는 138 개의 컴포넌트에 대한 HTML 명세서와 이에 대응하는 XML 명세서를 데이타로 하여 정확도에 대한 검색 성능을 실험하였다. 실험은 일단 세 가지 검색 방법으로 수행한다. 첫번째는 HTML 컴포넌트 명세서를 기반으로 HTML 검색 방식인 비구문 키워드 방식(HTML)을 사용하였으며, 두 번째로는 검색 엔진을 통하여 컴포넌트 정보를 입력하여 본 시스템에서 정의한 표준 XML 컴포넌트 명세서의 형식으로 등록한 경우에 대해서만 검색을 수행한 결과이다. 마지막으로 표준 XML 컴포넌트 명세서뿐만 아니라 일반적인 XML 컴포넌트 명세서에 대하여도 검색할 수 있도록 문맥을 확장 시켜 준 방법에 대한 검색 성능을 평가하였다. 첫번째 방식만 HTML 명세서에 대한 검색을 수행하며, 나머지 두 가지에 실험에서는 XML 명세서에 대한 검색을 수행한 것이다.

30개의 질의에 대하여 검색에 사용하여 나온 결과를 Relevant의 레벨 별로 나누어 5개씩의 질의 결과에 대한 정확도의 결과가 그림 5와 같다.

검색 결과는 표준 XML 을 사용하여 질의를 수행한 경우가 가장 정확도가 높았으며 그 다음이 문맥을 확장할 수 있도록 융통성을 준 경우로서 표준 XML 문서를 사용한 경우보다는 정확도가 떨어졌다. 그러나 역시 HTML의 키워드 검색 방법보다는 정확도가 높았다.

6. 결론 및 향후 연구

본 논문에서는 인터넷 상에 개방된 컴포넌트를 검색하기 위하여 XML 기반 컴포넌트 검색 기법에 대해 제안하였다. HTML 문서 기반의 검색 엔진들은 주로 키워드 기반의 검색을 수행하기 때문에 명세서에서 제공되는 정보들의 의미를 정확히 이해하지 못하고 찾는 키워드의 존재 유무나 빈도수에 의해서 검색 결과를 처리한다. 그러나 XML을 이용하여 정확한 의미 검색을 수행하고 컴포넌트에 대한 지능적인 검색을 수행해 주기 때문에 검색의 정확도가 뛰어나다.

본 연구에서는 컴포넌트 검색을 위하여 XML 컴포넌트 명세서를 사용할 것을 제안하며, 또한 이를 기반으로 한 문맥 기반 검색을 제안하였다. 문맥 기반 검색은 단순히 컴포넌트 명세서 내의 키워드만을 이용하는 것이 아니라 용어와 그 용어가 갖는 문맥의 의미를 동시에 사용하여 검색하므로 컴포넌트에 대한 정확한 검색을 수행할 수 있다.

먼저 본 논문에서는 컴포넌트의 검색을 위하여 필요한 컴포넌트에 필요한 명세서의 내용을 기술하고 이를 기반으로 한 XML 컴포넌트 명세서의 형태를 정의하기 위하여 컴포넌트 XML DTD를 정의하였다. 물론 현재까지 컴포넌트에 대한 표준 명세서가 정의되지 않았기 때문에 본 검색 엔진에서 정의된 명세서 이외의 다양한 컴포넌트 명세서를 수용할 수 있는 유연성(flexibility)을 가진 검색 방법을 사용하였다.

문맥 기반 검색에서는 용어-문맥-명세서 형태의 인덱스를 사용한다. 본 검색 엔진에서는 사용자의 편의를 위하여, 기존의 재사용 라이브러리 검색 방법들인 키워

드 검색, 퍼싯 검색, 브라우징 검색 방법도 지원한다. 이들을 효율적으로 지원하기 위하여 3-레이어의 검색 엔진의 구조를 설계하여 XML 검색 엔진 위에서 XML 문맥 기반 검색을 기반으로 하여 다른 검색 방법들이 지원될 수 있도록 하였다.

그리고, 사용자의 질의를 불리언 방식이 아닌 퍼싯 방법을 확장하여 검색 용어나 문맥에 대한 서로의 관련성을 명확하게 표현하고 사용자도 자신의 기호를 반영할 수 있도록 하였다.

용어와 문맥을 동시에 사용하여 검색하는 XML 문맥 기반 검색이 키워드 검색 방식의 HTML 문서 검색 방법보다 검색 성능이 좋아지는 것을 정확도 비교 실험을 통하여 입증하였다.

향후 연구 과제로는 컴포넌트를 검색함과 동시에 사용할 수 있는 환경의 제공, 더욱 상세한 컴포넌트에 대한 정보 검색 등에 대한 연구가 수행될 것이다.

참 고 문 헌

[1] AlphaBeans, http://www.alphaworks.ibm.com/alphabeans

[2] AltaVista, http://www.altavista.com/

[3] ASSET, http://source.asset.com/

[4] S. Browne, J. Dongarra, S. Green, K. Moore, T. Pepin, T. Rowan, and R.Wade, "Location-Independent Naming for Virtual Distributed Software Repositories," Proc. of the ACM SIGSOFT Symp. on Software Reusability, pp.179-185, 1995

[5] G. Caldiera and V. R. Basili, "Identifying and Qualifying Reusable S/W Component," IEEE S/W, Vol.8, No.2, pp61-72, Feb. 1991

[6] D. Chappell, *Understanding ActiveX and OLE*, Microsoft Press, 1996

[7] S. M. Chen and J.Y. Wang, "Document retrieval using knowledge-based fuzzy information retrieval techniques," IEEE Trans. Syst., Man, Cybern., Vol.25, No.5, pp.793-803, May 1995

[8] S. M. Chen and Y. J. Horng, "Fuzzy Query Processing for Document Retrieval Based on Extended Fuzzy Concept Networks," IEEE Trans. on Sys., Man, and Cybern., Vol. 29, No.1, Feb. 1999

[9] P. Ciancarini, F. Vitali, and C. Mascolo, "Managing Complex Documents Over the WWW:A Case Study for XML," IEEE Trans. on Knowledge and Data Engineering, Vol. 11, No. 4, pp. 926-638, July/Aug 1999

[10] DARPA STARS, http://www.asset.com/stars/

[11] D. Dubois, H. Prade, and R. R. Yager, Fuzzy Information Engineering, Wiley, 1997

[12] E.A. Fox, Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types., Cornell University, August 1983

[13] W. B. Frakes, R. Baeza-Yates, Information Retrieval Data Structures & Algorithms, Prentice Hall PTR, 1992

[14] GAGS, http://kal-el.ugr.es/GAGS/

[15] GAMS, http://gams.nist.gov/

[16] R. J. Glushko, J. M. Tenensaum, and B. Meltzer, "An XML FRAMEWORK For Agent-based E-Commerce," Communications of ACM, Vol.42, No.3, March, 1999

[17] HotBot, http://www.hotbot.com/

[18] P. Iglio and G. Attardi, "Software Components for Computer Algebra," Proc. of 1998 Int. Symp. on Symbolic and Algebraic Computation, pp.62-69, 1998

[19] InfoSeek, http://infoseek.go.com/

[20] T.C. Jones, "Reusability in Programming: A Survey of the State of the Art," IEEE Trans. on Software Engineering, Vol. SE10, No. 5, pp.488-494, September 1984

[21] N. H. Lassing, D.B.B. Rijsenbrij, and J.C. van Vliet, "A View on Components," Proc. of 9th Int. Workshop on Database and Expert Sys. Applications, pp.768-777, 1998

[22] A. C. Lear, "XML Seen as Integral to Application Integration", IT Pro, pp. 12-16, September/October 1999

[23] H. Mili, F. Mili, and A. Mili, "Reusing Software : Issues and Research Directions," IEEE Transactions on Software Engineering, Vol.21, No.6, pp528-562, June 1995

[24] T.D. Milner, Context Based Retrieval of Distributed Information Objects, Honours Thesis, Monash University, Caulfield, 1998

[25] T.D. Milner, SCOOBS, A Context Based Search Engine, Technical Report 1999/36, Monash University, Australia, 1999

[26] M. Morrison, presenting JAVABEANS, Sams net, 1997

[27] Robert Orfali and Dan Harkey, Client/Server Programmnng with JAVA and CORBA, John Wiely & Sons, 1998.

[28] Prieto-Diaz, R., Freeman, P., "Classifying software for reusability," IEEE Software, Vol.4, No.1, pp6-16, January, 1987.

[29] R. C. Seacord, S. A. Hissam, and K. C. Wallnau, "Agora: A Search Engine for Software Components," Technical Report, CMU/SEI-98-011, 1998

[30] C. Szyperski, *Component Software Beyond Object-oriented Programming*, Addison-Wesley, 1998

[31] WebCrawler, http://www.webcrawler.com/

[32] M. Wood, and I. Somerville, "An information system for S/W Components," SIGIR Forum Vol. 22, No. 3, pp.11-25, Spring/Summer 1988
[33] Netlib, http://www.netlib.org/

부 록 : 컴포넌트 명세서를 위한 XML DTD (Document Type Definition)

```
<!ELEMENT Component (Vendor, Information, Requirements, License, Price,Configuration)>
<!ATTLIST Component
        id CDATA #REQUIRED>
<!ELEMENT Vendor (#PCDATA)>
<!ATTLIST Vendor
        xml:link CDATA #FIXED "simple"
        href CDATA #REQUIRED>
<!ELEMENT Information (General_Info, Technical_Info, Interface+)>
<!ELEMENT General_Info (ComponentName, ComponentVersion?, ComponentDomain, Overview?, Advantages?, Awards*)>
<!ELEMENT ComponentName (#PCDATA)>
<!ATTLIST ComponentName
        xml:link CDATA #FIXED "simple"
        href CDATA #REQUIRED>
<!ELEMENT ComponentVersion (#PCDATA)>
<!ELEMENT ComponentDomain (#PCDATA|Function|Object|KeyTerm)*>
<!ELEMENT Function (#PCDATA)>
<!ATTLIST Function
        id CDATA #REQUIRED>
<!ELEMENT Object (#PCDATA)>
<!ATTLIST Object
        id CDATA #REQUIRED>
<!ELEMENT KeyTerm (#PCDATA)>
<!ELEMENT Overview (#PCDATA|KeyTerm)*>
<!ELEMENT Advantages (#PCDATA|KeyTerm)*>
<!ELEMENT Awards (#PCDATA)>
<!ELEMENT Technical_Info (Platform+, Language+, OS+, ComponentType+, Tools, Supported-Language*, Y2K*, TechnicalDescription*)>
...
<!ELEMENT Interface (InterfaceName, Method+, Property*, DataType*, Exception*, Event*)>
<!ATTLIST Interface
        id CDATA #REQUIRED>
<!ELEMENT InterfaceName (#PCDATA)>
<!ELEMENT Method (MethodName, Declaration, ReturnType, ParameterType*, ThrowsException*, Description?)>
<!ATTLIST Method
        id CDATA #REQUIRED>
<!ELEMENT MethodName (#PCDATA)>
<!ELEMENT ReturnType (#PCDATA)>
<!ELEMENT ParameterType (#PCDATA)>
<!ATTLIST ParameterType
        order CDATA #REQUIRED>
<!ELEMENT ThrowsException (#PCDATA)>
<!ELEMENT Property (PropertyName, Declaration, DataType, ValidValue*, DefaultValue?, Description?)>
<!ATTLIST Property
        id CDATA #REQUIRED>
<!ELEMENT PropertyName (#PCDATA)>
<!ELEMENT DataType (#PCDATA)>
<!ELEMENT ValidValue (#PCDATA)>
<!ELEMENT DefaultValue (#PCDATA)>
<!ELEMENT Event (EventName, Description?)>
<!ATTLIST Event
        id CDATA #REQUIRED>
<!ELEMENT EventName (#PCDATA)>
<!ELEMENT Exception (ExceptionName, Description?)>
<!ATTLIST Exception
        id CDATA #REQUIRED>
<!ELEMENT ExceptionName (#PCDATA)>
<!ELEMENT Declaration (#PCDATA)>
<!ELEMENT Description (#PCDATA)>

<!ELEMENT Requirements (Disk*, Memory*, Prerequistics*, Additional*)>
  ...
<!ELEMENT License (#PCDATA)>
  ...
<!ELEMENT Price (#PCDATA)>
  ...
<!ELEMENT Configuration (VersionInfo, UsedComponents*)>
<!ELEMENT VersionInfo (OldVersion)*>
<!ELEMENT OldVersion (ComponentName, Relationship, ComponentVersion, Improvement)>
<!ELEMENT Improvement (#PCDATA)>
<!ELEMENT RelatedComponents (ComponentName)*>
<!ATTLIST RelatedComponents
        relative_type (used_by | using | aggregation )
        "used_by">
```

박 서 영
1988년 서울대학교 계산통계학과 졸업. 1990년 서울대학교 계산통계학과 석사 졸업. 1990년 ~ 현재 서울대학교 전산과학과 박사과정. 1995년 ~ 1997년 미국 Colorado 대학 연구생. 관심분야는 소프트웨어 공학, 컴포넌트 소프트웨어, 소프트웨어 재사용

신 영 길

1982년 서울대학교 계산통계학과 졸업(학사). 1984년 서울대학교 계산통계학과 석사학위 취득. 1989년 University of Southern California 박사학위 취득. 1990년 ~ 1992년 경북대학교 전임강사. 1992년 ~ 현재 서울대학교 전산과학과 조교수. 관심분야는 컴퓨터그래픽스, CAD, 인공지능, 멀티미디어 등임.

우 치 수

1972년 서울대학교 공과대학 응용수학(공학사). 1977년 서울대학교 대학원 전산학(석사). 1982년 서울대학교 대학원 전산학(박사). 1978년 영국 라퍼러대학연구원. 1975년 ~ 1982년 울산대 전자계산학과 부교수. 1985년 미국 미시간대학 Post-Doc. 현재 서울대학교 전산과학과 교수. 관심분야는 소프트웨어 공학, 프로그래밍 언어.