효과적인 소프트웨어 컴포넌트 분류 및 검색에 관한 연구

A Study for the Effective Classification and Retrieval of Software Component

조 병 호*
Byung-ho Cho

요 약

컴포넌트 재사용을 통한 소프트웨어의 개발은 소프트웨어 생산비용을 절감할 수 있는 유용한 방법이다. 그러나 컴포넌트 재사용에 있어 키워드나 카테고리 분류에 의한 검색 방법은 컴포넌트 개체의 복잡성으로 인하여 정확한 컴포넌트 검색이 어렵다. 따라서 기존의 다른 연구 방법을 조사/분석하여 XML 명세를 이용한 컴포넌트 분류 및 검색에 효과적인 방법 및 이를 기반으로 한 컴포넌트 통합관리 시스템 구조를 제시하고자 한다.

컴포넌트 검색에 있어 많은 일치하지 않은 컴포넌트 메타 표현인 DTD 항목이 존재한다. 이를 보완하기 위하여 정확도 및 간결도 측정을 사용한 검색 방법은 우선적으로 고려해야할 컴포넌트를 찾는데 효과적인 하나의 방법이다. 이 방법은 기존의 키워드 검색으로 어려운 유사하게 일치하는 항목의 컴포넌트를 찾음으로써 보다 나은 우선순위를 갖는 적합한 컴포넌트 검색이 가능하게 한다.

Abstract

A software development using components reuse is an useful method to reduce the software development cost. But a retrieval method by the keyword and category classifications is difficult to search an exact matching component due to components complexity in component reuse. Therefore, after different existing methods are examined and analyzed, an effective classification and retrieval method using XML specifications and the system architecture of components integrated management based on it are presented.

Many discording elements of DTD which is component meta-expression exist in components retrieval. To compensate it, this retrieval method using estimations of precision and concision is an effective one to catch considerable matching preference components. This method makes possible to retrieve suitable components having better priority due to searching similar matching components that are difficult in an existing keyword matching method.

☞ Keyword : component reuse, component classfication, componet retrieval

1. 서 론

컴포넌트는 재사용성과 상호 운용성이 뛰어난 미리 구현된 블록으로 응용시스템 개발자는 마치 하드웨어를 조립하듯이 원하는 기능이나 성능을 가진 소프트웨어 부품인 컴포넌트를 이용하여 시스템을 구성할 수 있다. 이렇게 개발된 컴포넌트 기반 시스템은 일부의 부품을 개선된 부품인 컴포넌트로 대체함으로써 품질이나 성능을 개선할 수 있고 새로운 운용환경에 쉽게 이식할 수 있다.

따라서 최근에 컴포넌트 기반 소프트웨어 개발은 가장 효과적이고 생산성이 뛰어난 방법으로 업계에서도 점차 많이 채택되어 사용되고 있다. 기업체에서 사용하는 많은 정보 시스템이 컴포넌트 형태로 개발이 되어 다른 시스템 개발에도 컴포넌트의 효과적인 재사용을 할 수 있다면 소프트웨어 생산비용을 많이 줄일 수 있을 것이다. 그러나 적절한 컴포넌트의 저장/검색이 가능한

* 정회원 : 관동대학교 컴퓨터학부 정교수
bhcho@kwandong.ac.kr
[2006/02/22 투고 - 2006/03/16 심사 - 2006/08/21 심사완료]

통합관리 체계가 갖추어져있지 않으면 적합한 컴포넌트를 찾는데 많은 시간이 소모되고 무분별한 컴포넌트 사용으로 인한 소프트웨어 품질의 저하를 가져올 수 있다.

그러나 현재 상용화된 컴포넌트 검색엔진은 키워드 검색이나 카테고리 분류에 의한 검색 기법이 많이 사용되고 있지만, 이는 컴포넌트 개체의 복잡성으로 인하여 단순히 키워드 검색으로는 원하는 컴포넌트를 얻기가 쉽지 않다. 따라서 효과적인 컴포넌트 통합관리 시스템 구축을 위한 컴포넌트의 분류 및 검색 방법에 대한 연구가 이루어져야 한다. 본 논문에서는 기존의 소프트웨어 재사용을 위한 분류 및 검색 방법을 분석하여 컴포넌트 분류 및 검색에 적합한 기법을 제안하고 이를 기반으로 컴포넌트 저장소 구축 및 효율적인 컴포넌트 통합관리 시스템 설계를 위한 전체적인 구조를 제안하고자 한다.

정확하게 일치하지 않는 유사한 컴포넌트의 검색결과에서 원하는 컴포넌트를 선택할 때는 패싯 분류/검색 방법이 적합하다. 이는 검색을 위해 사용하는 방법이 수작업에 의한 것으로 자동화하기 어려운 단점이 있고, 서명매칭 방법은 컴포넌트가 클래스 위주의 소스 레벨의 타입 매칭으로 정의하면 가장 정확한 검색이 가능한 우수한 방법이나 컴포넌트가 분석/설계 단계에서의 산출물을 포함하므로 타입형식으로 컴포넌트 분류가 불가능해 부적합하다. 따라서 본 논문에서는 최근의 문서표준 언어인 XML(eXtensible Markup Language)을 명세언어로 사용한 컴포넌트 분류/검색 기법을 제안한다

이 때 XML 명세로 된 컴포넌트를 검색하기 위한 방법으로 단순히 키워드 검색에 의한 컴포넌트의 XML 명세를 찾는 것은 AtaVista, Web-Craweler, Infoseek 와 같은 인터넷 검색엔진 같은 것으로 만들 수 있다. 그러나 원하는 컴포넌트의 재사용을 위해서 키워드에 의한 검색은 컴포넌트 특성 항목이 많기 때문에 단순히 키워드에 의해 매칭되는 것을 찾는 것이 어려울 때가 많고 비슷한 결과가 검색되는 경우에 어느 것이 더 적합한지를 판단하여야 하므로 더 효율적인 검색엔진의 구축방법이 필요하다.

따라서, 본 논문에서는 컴포넌트의 정확성 혹은 유사성 판단을 위해 해밍 거리(Hamming Distance) 측정 방식과 순위(ranking) 알고리즘에 의한 모듈을 추가함으로써 보다 효율적인 컴포넌트 검색엔진을 설계하는 방법을 제시하고자 한다.

본 논문의 2장에서는 관련 연구로서 기존에 재사용을 위한 소프트웨어 산출물의 효과적인 저장/검색을 위한 문서 분류 및 검색기법을 기술하고, 이들 기법의 특징 및 장단점을 알아본다. 3장에서는 컴포넌트 표현을 위한 속성 및 XML을 이용한 컴포넌트의 명세작성에 대하여 기술한다. 4장에서는 효과적인 컴포넌트 검색방법 및 컴포넌트 통합관리 시스템 구조를, 그리고 5장에서는 본 제안방법의 우수성을 제시하고, 6장은 결론 및 향후 연구방향에 대하여 기술한다.

2. 관련연구

소프트웨어 재사용을 위한 분류/검색 방법으로는 키워드, 지식기반 및 패싯(facet) 인덱싱에 의한 기법 등에 관한 연구가 많이 이루어지고 있다. 지식기반 시스템은 소프트웨어의 자연어 검색을 위한 문법 및 의미적인 분석을 수행하는 방법으로 세만틱-넷(semantic-net)이나 프레임(frame) 기반 시스템으로 구성된다. 세만틱-넷은 추론에 의한 구조적 지식 표현방법을 사용하고 프레임기반 분류는 소프트웨어 산출물의 의미를 획득하는 것으로 객체의 구조적 표현 방법을 사용한다. 대표적인 예로는 LaSSIE(Large Software System Information Environment) 및 RLF(The Reuseability Library Framework)가 있다[9].

키워드에 의한 소프트웨어 산출물의 분류방법은 문서로부터 키워드를 자동으로 추출하는 것으로 확률적인 시스템과 문법 및 의미론적인 시스템으로 나뉘는데 확률적 시스템은 문서에서 단어

의 빈도로 키워드를 추출하는 것이고 문법 및 의미론적인 시스템은 자연어의 문법적 및 의미론적 분석에 의한 것이다. 이는 자연어에 대한 키워드 쿼리를 수행하여 저장소에서 매칭하는 소프트웨어 산출물을 검색하는 시스템으로서 예로는 CATALOG와 GURU가 있다[4].

패싯(facet) 인덱싱은 가장 많이 사용하는 소프트웨어 분류법으로 Prieto-Diaze가 처음 제안한 방법으로 기능중심의 소프트웨어 코드의 재사용을 위해 소프트웨어를 <함수, 객체, 매개체, 시스템 타입, 함수영역, 셋팅>의 6가지 facets으로 나누었는데 앞의 3가지 요소는 프로그램의 함수에 관련된 부분이고 뒤의 3가지는 프로그램의 운영환경에 관련된 것이다. 이 방법에 있어 검색은 산출물의 유사값을 이 6가지 요소로서 개념적 거리 그래프와 퍼지 로직에 의해 측정하여 가장 근접한 산출물을 원하는 것으로 정하게 된다. 또 다른 방법으로는 REBOOT(Reuse Based on Object-Oriented Techniques)가 있는데 이는 객체지향 소프트웨어 컴포넌트의 재사용을 위한 것으로 클래스에 중점을 두어 <추상화, 운용, 오퍼레이터 On, 상관성)의 4가지 facets을 사용한다[14].

그 밖의 방법으로는 서명매칭(signature matching), 명세매칭(specification matching) 기법이 있는데, 서명매칭은 타입(type) 매칭이나 변환에 의해 소프트웨어 산출물을 분류하는 방법으로 클래스는 다중집합의 서명(signature)으로 표현되는데 가능한 타입은 단순타입(정수, 실수 etc), 구조적 타입 등이 있다. 명세매칭은 소프트웨어를 전제 조건 및 사후 조건 등과 같은 논리적인 조건들로 표현하여 분류한다. 이는 Z, B, OCL과 같은 명세언어로 소프트웨어 산출물을 명세화 하여 표현한다. 검색에는 명세 라이브러리와 질의어가 매칭하는 기법으로 수학적 증명기법을 사용한다[16].

3. 소프트웨어 컴포넌트 분류

Rational Software의 Philipe Krutchen는 소프트웨어 컴포넌트는 "잘 정의된 아키텍쳐 상에서 어떠한 기능을 수행하는 시스템에 독립적이면서 대치 가능한 부분이다. 컴포넌트는 인터페이스들의 집합에 대한 물리적인 구현을 제공한다."와 같이 컴포넌트를 정의한다. 그 밖에 여러 가지 정의가 있지만 좁은 의미에서는 CCM(CORBA Component Model), COM(Component Object Model)/DCOM(Distributed Component Object Model) 및 EJB(Enterprise Java Beans) 플랫폼에서의 컴포넌트가 있고 넓은 의미에서는 분석/설계 과정의 산출물을 포함한 넓은 의미의 컴포넌트가 있다.

본 논문에서는 Catalysis Method나 Sterling Enterprise Model 및 Rational사의 CBD(Component Based Developlment)를 이용하든 간에 넓은 의미의 각 소프트웨어 개발 단계의 산출물을 포함한 컴포넌트의 문서를 분류하고 검색하기에 효과적인 방법을 제시한다.

2장의 관련연구의 분석을 통해 컴포넌트 분류/검색 방법으로 적합한 방법을 고려해 볼 때, 컴포넌트를 검색할 때 정확하게 일치하지 않은 유사한 컴포넌트에 대하여는 원하는 컴포넌트를 선택할 때 패싯 분류/검색 방법이 적합하지만 아직 검색을 위해 사용하는 방법이 수작업에 의한 것으로 자동화하기 어려운 단점이 있고, 서명매칭 방법은 컴포넌트가 클래스 위주의 소스 레벨의 타입 매칭으로 정의하면 가장 정확한 검색이 가능한 우수한 방법이나 컴포넌트가 분석/설계 단계에서의 산출물을 포함하므로 타입형식으로 컴포넌트 분류가 불가능해 부적합하다. 따라서 본 논문에서는 최근의 문서표준 언어인 XML(eXtensible Markup Language)을 명세언어로 사용한 컴포넌트 분류/검색 기법을 제안한다[2,3,6,7,8,11]. 기존의 명세언어인 Z, B 등은 수학적인 표기법으로 이해하기도 어렵고 컴포넌트 저장소 구축하기가 어려운 단점이 있지만 XML은 컴포넌트를 구조적인 형태로 명세화하기 쉽고 XML 언어로 명세화한 컴포넌트 명세는 메타데이터로

컴포넌트 저장소 구축이 용이하고 데이터를 효과적으로 관리할 수 있으며 상용화된 DBMS(Database Management System)를 이용한 컴포넌트 구축이 가능하고 최근의 웹 환경의 분산환경에서의 접근성이 용이한 컴포넌트 관리 시스템 제작이 가능한 장점을 가지고 있다[1,5,12,13].

3.1 컴포넌트의 표기

컴포넌트의 효과적인 관리를 위한 통합시스템 구축을 위해서는 컴포넌트를 어떻게 표현하는 것이 중요한데 컴포넌트를 표현하기 위한 특성들은 여러 연구논문에서 착안하여 얻은 여러 가지 속성들을 컴포넌트 표현 인자로서 사용한다. 이들 컴포넌트를 표현하기 위한 속성들을 기술하면 아래와 같다[10,15].

(1) 컴포넌트 식별자(idendification)

- 컴포넌트를 구분하는 유일한 식별자인 id, 컴포넌트 이름(name), 컴포넌트 버전숫자 등으로 구성된다.

(2) 컴포넌트 영역(domain)

- 컴포넌트를 재사용하기 위해서 우선 도메인 분석을 통하여 컴포넌트가 어떤 분야(예를 들면 제조업, 금융 etc)에 응용되는 컴포넌트인지를 파악하는 것이 중요하다.

(3) 컴포넌트 용도(usage)

- 컴포넌트가 사용자 인터페이스, 비즈니스 로직, 데이터접근 등 어떤 용도로 사용되는지를 나타낸다.

(4) 컴포넌트 기능(function)

- 컴포넌트가 수행하고자 하는 기능을 표시하는 것으로 여러 개의 매칭 값이 존재한다. 예를 들면 인터넷 쇼핑몰 제작을 컴포넌트로 구축할 경우 카타고리 컴포넌트는 상품 분류정보를 등록, 검색, 수정, 삭제 등의 기능을 가지는데 이들 기능이 컴포넌트의 기능이 된다.

(4) 컴포넌트 운용환경(os environment)

- 컴포넌트가 실제 응용 시스템에서 사용될 때의 컴퓨터 하드웨어, 운용체제(OS), DBMS 등을 나타낸다.

(5) 컴포넌트 구현 언어(implementation language)

- 컴포넌트 구현을 위한 소스레벨에서의 프로그래밍 언어의 종류 표시

(6) 컴포넌트 Lifecycle 단계

- 컴포넌트가 소프트웨어 생명주기인 분석, 설계, 코딩, 테스트 중의 어느 단계의 산출물인지를 표현.

(7) 컴포넌트 history

- 컴포넌트의 개발자, 제작일자, 컴포넌트 크기, 컴포넌트 style 등의 표현

(8) 컴포넌트 설명(description)

- 컴포넌트를 기술하는 설명 파일

(9) 컴포넌트 위치(location)

- 컴포넌트 문서의 디스크상의 파일 위치

3.2 XML를 이용한 컴포넌트 명세작성

XML으로 컴포넌트 명세를 기술하기 위해서는 DTD(Document Type Definition)를 정의하

여야 하는데 위에서 컴포넌트를 표현하기 위한 요소를 가지고 DTD를 작성하면 <그림 1>과 같다.

```
<?xml version = "1.0"?>
<DOCTYPE components SYSTEM
  "component.dtd">
<!ELEMENT Components(Identification+,
  Domain, Usage, Function, Os_environment*,
  Implementation_language, Lifecycle_level,
  Description, Location)>
<!ELEMENT Identification(Id, Name,
  Version_no)>
<!ELEMENT Id(#PCDATA)>
<!ELEMENT Name(#PCDATA)>
<!ELEMENT Version_no(#PCDATA)>
<!ELEMENT Domain(#PCDATA)>
<!ELEMENT Usage(#PCDATA)>
<!ELEMENT Function(#PCDATA)>
<!ATTLIST Function type (en1| en2| ..)>
<!ELEMENT Os_envrionment(Hardware, OS,
  DBMS)>
<!ELEMENT Hardware(#PCDATA)>
<!ELEMENT OS(#PCDATA)>
<!ELEMENT DBMS(#PCDATA)>
<!ELEMENT Implementation_language
  (#PCDATA)>
<!ELEMENT Lifecycle_level(#PCDATA)>
<!ELEMENT Description(#PCDATA)>
<!ELEMENT Location(#PCDATA)>
```

〈그림 1〉 컴포넌트의 DTD 표기

4. 컴포넌트 XML 명세 문서의 검색 엔진 설계

단순히 키워드 검색에 의한 컴포넌트의 XML 명세를 찾는 것은 AtaVista, WebCraweler, Infoseek 와 같은 인터넷 검색엔진 같은 것으로 만들 수 있다. 그러나 원하는 컴포넌트의 재사용을 위해서 키워드에 의한 검색은 위에서 컴포넌트 특성에 따른 분류에 알 수 있듯이 컴포넌트 특성 항목이 많기 때문에 단순히 키워드에 의해 매칭되는 것을 찾는 것이 어려울 때가 많고 비슷한 결과가 검색되는 경우에 어느 것이 더 적합한지를 판단하여야 하므로 더 효율적인 검색엔진의 구축방법이 필요하다.

따라서, 본 논문에서는 컴포넌트의 **정확도(P)** 판단을 위해 해밍 거리(Hamming Distance) 측정 방식과 **간결도(C)** 측정을 위해 순위(ranking) 알고리즘에 의한 모듈을 추가함으로써 보다 효율적인 컴포넌트 검색엔진을 설계한다. 검색의 대상이 되는 것은 XML 컴포넌트의 스키마타(schemata)인 DTD 명세가 되며 이들 컴포넌트 검색엔진의 설계를 위한 두 가지 방식을 설명하면 아래와 같다.

정확도(P)을 측정을 위해 만약에 사용자가 찾고자 하는 컴포넌트가 U라고 하고 DTD 저장소(repository)의 컴포넌트 명세가 DSi이면 U와 DSi의 차이를 f(U, DSi)라 한다. 만약에 f(U, DSi)= 0이라 하면 U와 DSi는 동일한 것으로 판단할 수 있다. 그러나 U와 DSi의 컴포넌트 특성 비교 항목이 동일하게 일치하는 경우가 아닌 경우에 해밍 거리(Hamming Distance) 측정기법에 의해 판단하도록 한다. 예를 들면 저장소의 컴포넌트 명세에서의 특성 항목이 약간씩 차이가 있는 세 개의 문서가 있다고 했을 때, 즉 DS1:(a, b), DS2:(a, b, c), DS3:(a, b, c, d)라고 하고 찾고자 하는 컴포넌트 항목이 (a, b, c)인 U:(a,b,c)는 f(U, DS2)= 0 이므로 컴포넌트 명세 U와 정확히 일치한다. 그러나 U가 U:(b, c, d)라 하면 정확하게 DSi와 일치하는 것이 없으므로 **정확도** 측정값인 **P**을 도입한다. **P** 값은 사용자가 컴포넌트 특성 항목이 정확하게 일치하지 않을 경우 유사한 것을 선택하기 허용 가능한 값으로서, 일치항목 개수를 의미한다. 예를 들면, **P**가 2라고 하면 위에서 사용자가 원하는 컴포넌트 U의 재사용가능한 컴포넌트 DS2와 DS3가 된다. 좀 더 체계적인 유사성 측정을 위해 벡터 x와 y의 거리를 측정하는 해밍거리 HD(x, y)를 적용한다. 그러면 U:(0,1,1,1)이고 DS1:(1,1,0,0), DS2:(1,1,1,0), DS3:(1,1,1,1)이며 HD(U, DSi)는 각각 3, 2, 1 이다. 따라서 가장 U와 적합한 컴포넌트로는 해밍거리가 가장 적은 값인 DS3를 선택하게 된다.

다음으로는 DTD에서 중복되는 요소가 있는데, 예를 들면 항목(element) a, b가 {ab, abab, ababab}와 같은 경우 DTD에서는 (ab)*, (ab|ab) 등으로 표시 하는데, 이때 간결한 표현이 컴포넌트 검색의 우선 조건이 될 수 있는데 이를 위해 간결성 순위를 정하는 랭킹모듈을 설치하는데, 이는 다음과 같은 규칙에 의하여 순위를 정하는 값으로 **간결도(C)**을 표기하며 검색결과의 순서를 정하는데 사용한다.

(1) 각 항목은 하나(one)로 계산하고 루트 항목은 계산하지 않는다.
(2) 각 |, *, ?와 같은 메타(meta) 캐릭터는 하나로 계산한다.
(3) 부구조(substructure)의 길이를 계산한다.

```
<Purchase_Order>
  <Form_No>24233598</Form_No>
  <Date><20060212></Date>
  <Vendor>Raincom Co.</Vendor>
  <Detail>
    <Product>MP3 Player</Product>
    <Qty unit="sets">5</Qty>
    <Cost>$2,350</Cost>
  </Detail>
  <Detail>
    <Product>LCD TV</Product>
    <Qty unit="sets">2</Qty>
    <Cost>$5,350</Cost>
  </Detail>
<Purchase_Order>
```

〈그림 2〉 purchase order에 대한 XML 표현 간단한 예

```
<!ELEMENT Purchase_Order(Form_No,
  Date, Vendor, Detail*)>
<!ELEMENT Form_No(#PCDATA)>
<!ELEMENT Date(#PCDATA)>
<!ELEMENT Vendor(#PCDATA)>
<!ELEMENT Detail(Product, Qty, Cost)>
<!ELEMENT Product(#PCDATA)>
<!ELEMENT Qty(#PCDATA)>
<!ELEMENT Qty unit CDATA #REQUIRED>
<!ELEMENT Cost(#PCDATA)>
```

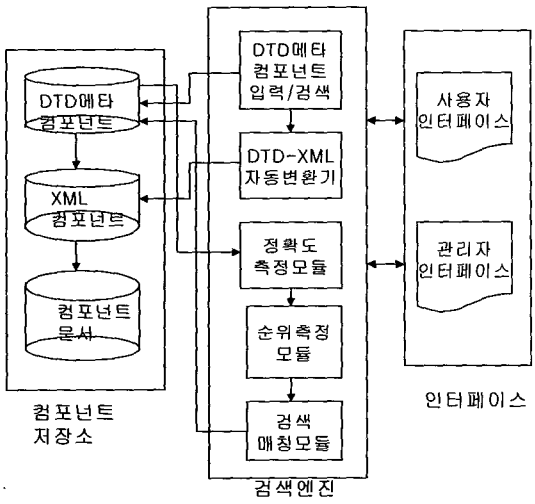
〈그림 3〉 purchase_order의 XML 표현에 대한 DTD

이와 같은 규칙으로 <그림 2>의 간단한 purchase_order XML 예에 대한 DTD 표현인 <그림 3>을 가지고 설명하면 DTD의 루트 항목인 purchase_order는 규칙1에서 계산하지 않는다. 그리고, 이는 4개의 항목이 있으므로 규칙1에 의해 C는 4로 표시한다. Detail은 메타 캐릭터인 * 를 가지고 있으므로 2번째 규칙에 의해 간결도 C는 5가 된다. DTD에서 Detail은 부구조를 가지고 있으므로 그 길이가 3이므로 전체 순위도는 8이 된다. 사용자가 재사용 컴포넌트 검색에 특별한 제한사항을 두지 않으면 MDL 값이 적을수록 높은 순위의 컴포넌트 후보가 된다. 그러나 DTD 문서의 구조상에 제약사항이 주어지면 |C(U) - C(DS i)|가 가장 적은 순으로 정렬하고 이 순서가 적합한 재사용 가능한 컴포넌트가 된다. 예를 들면, 3개의 DTD 후보가 있고 이는 DTD1: (a,b,c), DTD2:(a,b*,c), DT3:(a,b,c*)이며 U는 (a,b*,c)라면 |C(U) - C(DTi)|는 각각 1, 0, 0이므로 검색 결과 재사용 가능한 컴포넌트의 적합한 후보는 DS2, DS3, DS1의 순이다.

위에서 제시한 재사용 컴포넌트의 검색 분류방법과 정확하게 일치하지 않는 검색조건의 보다 효과적인 컴포넌트 검색을 위해 정확도 및 간결도 측정을 수행하는 모듈을 검색엔진에 추가함으로써 기존의 단순 키워드 검색에 의한 방식보다 보다 진전된 검색엔진의 설계가 가능하다. 다음 <그림 4>는 이와 같은 이론적인 배경을 바탕으로 설계한 전체 컴포넌트 통합관리 시스템 구조이다.

이 시스템의 동작원리를 설명하면 다음과 같다. 우선, 컴포넌트의 메타명세를 2장에서 제시한 컴포넌트 분류 방법에 의해 작성하여 입력한다. 다음으로, 사용자가 원하는 재사용 컴포넌트 검색은 일차적으로 정확도 측정 모듈에서 DTD 항목이 가장 근접한 유사한 항목을 갖는 컴포넌트를 선택하고 그 선택한 후보들 중에 중복되는 항목들을 갖는 경우에 순위측정 모듈(간결도 측정)에서 순위를 매겨 가장 적은 값을 갖는 것을 DTD 컴포넌트로 선정한다. 최종적인 컴포넌트

문서는 DTD와 매칭되는 XML 명세형태의 컴포넌트를 저장소에서 검색하며, 이는 XML 명세에서 연결된 외부 컴포넌트 문서 파일을 검색하여 사용자 인터페이스 화면에서 보여줌으로써 원하는 재사용 컴포넌트를 찾게 된다.



〈그림 4〉 컴포넌트 통합관리 시스템 구조

5. 제안 방법의 우수성 제시

본 논문에서 제시한 XML 명세를 이용한 컴포넌트의 분류 및 검색 방법이 가장 효과적인 방법임을 증명하기 위해서는 각 비교 대상 방법과 본 제안 방법을 구현하여 비교 결과치를 제시하는 것이 가장 효율적이고 명확한 방법이나 이를 구현하기 위해서는 너무 많은 시간이 요하므로 향후 연구를 지속적으로 수행하여 다음 과제로 수행하기로 한다. 그러나 논문의 제안 방법의 우수성을 보여주기 위하여 <표 1>과 시뮬레이션 결과를 통해 이를 제시하고자 한다.

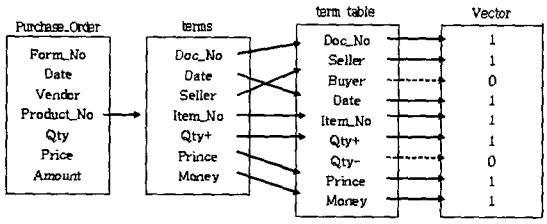
<표 1>에서 보는 봐와 같이 본 제안 방법은 모든 항목에서 우수함을 보여준다. 가장 자동화하여 구현하기 쉬운 방법은 키워드에 의한 분류/검색 방법이나 이는 검색 항목이 정확하게 일치하지 않은 경우에 검색정확도 많이 떨어지는 방법이므로 본 논문은 정확도 및 간결도 모듈을 보완하여 검색 정확도를 높인 방법으로서 효과적이라 할 수 있다.

본 제안 방법의 구체적인 수치를 통한 우수성 제시를 위하여 검색 시간을 측정하는 시뮬레이션 수행 첫 번째 단계로서 <그림 3>의 purchase_order에 대한 DTD 표기를 벡터 값으로 변경하는 방법을 보여준다. 정확도 측정을 위해서 HD(Hamming Distance)는 두개의 DTD 사이의 차이를 측정하는 값이고, Us와 DTDi를 포함하는 DTDs는 term 테이블의 term 갯수를 나타내는 이진 벡터 값으로 변경된다. 이와 같이 term 테이블의 각 term 들은 이진 벡터 값인 0과 1로 변경된다. DTD 원소 값이 벡터의 i번째 요소와 연관이 있으면 1 값을 아니면 0 값을 갖게 된다. 예를 들면, term 테이블의 길이가 9이고 pur-

〈표 1〉 각 컴포넌트 분류/검색 방법의 비교

항목 방법	키워드 검색	패싯분류/검색	세만틱넷	서명매칭	명세매칭	XML(본논문)
이해 용이성	good	good	good	good	poor	good
명세화 용이성	good	good	good	good	good	good
정보저장소 구축	very good	poor	poor	poor	poor	very good
자동화	very good	poor	poor	poor	poor	very good
구현가능성	very good	good	good	good	poor	good
검색 정확도	poor	good	good	good	good	very good

chase_order가 6개 term 값을 가지면 <그림 5>와 같이 된다. 정확도를 이용한 검색 후의 결과는 HD(U, DTDi) < r 값이 되며 r=0이면 100% 정확한 값을 갖는 경우가 된다.



〈그림 5〉 DTD를 벡터 값으로 변경하는 구조

각 DTD 테이블의 DTDs의 숫자를 m이라 하고 DTD의 term 숫자를 p라고 하면 HD를 계산하기 위한 알고리즘은 <그림 6>과 같이 기술할 수 있다. 이 알고리즘은 정확도(P)를 측정하기 위해서 4장에서 기술하였던 것처럼 HD 값을 구하기 위해 사용된다. 검색 시간 측정을 위해서 term 테이블의 term 갯수가 n=100개이고 컴포넌트 XML의 DTD의 term 수는 p=30개인 경우에 검색하고자 하는 DTD 표현 갯수를 1,000에서 10,000까지 랜덤하게 생성하여 검색시간을 시뮬레이션을 통하여 측정하여 보았다. 그 결과는 <표 2>와 같다. 표에서 알 수 있듯이 아주 빠른 검색 시간 내에 컴포넌트를 검색한다. 따라서 본 제안 방법을 이용한 컴포넌트 분류/검색 기법은 키워드 방식에 의한 것처럼 검색 항목의 불일치로 인한 검색 실패의 경우도 없으며 검색 시간이 빠르므로 매우 우수한 방법임을 알 수 있다.

〈표 2〉 HD 검색시간 결과(n=100, p=30)

No. of DTDs	HD 검색시간 결과치(ms)
1,000	771
2,000	1,552
3,000	2,323
4,000	3,094
5,000	3,865
6,000	4,646
7,000	5,417
8,000	6,179
9,000	6,970
10,000	7,751

6. 결론 및 향후과제

컴포넌트 재사용을 통한 소프트웨어의 개발은 소프트웨어 생산비용을 절감할 수 있는 효율적인 방법이지만 단순히 키워드나 카테고리 분류만으로 재사용 컴포넌트를 검색하는데 한계가 있다. 따라서 이에 대한 다른 연구방법으로 지식기반, 패싯(facet), 서명매칭(signature matching), 명세

```
X be the input data; Oj be the j known data
Y be the output data; Pj be the number of terms in patterns
n=length of vector; m=number of patterns
for j-1 to m
 if(Pj is in the range of (i+r) and (i-r)) then
 s=0;
 for i=1 to n
 if Xi<>Oji then
 s++;
 end
 if(s<=r) then Yj=s;
 else Yj=0;
 else
 Yj=0;
end
```

〈그림 6〉 HD 알고리즘

매칭 방법 등이 제시되고 있는데 본 논문에선 XML 명세를 이용한 컴포넌트 분류/검색 방법 및 컴포넌트 통합관리 시스템 구조를 제시하였다.

기존의 명세언어인 Z, B 등은 수학적인 표기법으로 이해하기도 어렵고 컴포넌트 저장소 구축하기가 어려운 단점이 있지만 XML은 컴포넌트를 구조적인 형태로 명세화 하기 쉽고 XML 언어로 명세화한 컴포넌트 명세는 메타데이터로 컴포넌트 저장소 구축이 용이하고 데이터를 효과적으로 관리할 수 있으며 상용화된 DBMS를 이용한 컴포넌트 구축이 가능하고 최근의 웹 분산 환경에서의 접근성이 용이한 컴포넌트 관리 시스템 제작이 가능한 장점을 가지고 있다.

본 논문의 주요 핵심은 컴포넌트 검색 시에 DTD 항목이 정확히 일치하지 않은 경우가 많으므로 이를 보완하기 위한 방법으로 정확도(P) 및 간결도(C) 측정 방법을 통하여 검색 시에 우선적으로 고려해야할 컴포넌트를 찾는 방법을 제시함으로써 기존의 키워드 검색보다 효과적으로 컴포넌트 재사용이 가능하도록 하였다.

앞으로는 추가적인 연구를 통하여 기존에 제시한 정확도 및 순위측정 방법을 구현하고 실험을 통하여 기존방식에 비해 우수함을 검증하고, 컴포넌트 통합관리 시스템 각 모듈을 제작하여 통합도구로서 구현이 필요할 것으로 본다.

참 고 문 헌

[1] Bourret., Bornhovd, C., & Buchmann. "A Genenric load/extract utility for data transfer between XML document relational databases". Procedings of the Second International workshop on Advanced Issues of E-Commerce and Web-based Information Systems, pp. 134-143, 2000.

[2] Decker, S., Melink, S., Van Harmelen, F., Fensel, "The semantic web: the roles of XML and RDF.", IEEE internet Computing, pp 63-74, 2000.

[3] Garofalakis, M., Gionis, A., Rastogi, R., Seshadri, "XRACT: a system for extracting document type descriptors from XML documents., SIGMOD, pp 165-176, 2000

[4] Gudivada, V., Rahavan, V., Grosky, "Information retrieval on the world wide web.", IEEE Internet computing, pp 58-68, 1997

[5] Kotsaki, "XSD: a hierachical access mehod for indexing XML schemeta.", Knowledge and information Systems, pp 168-201, 2000.

[6] Moh, C., Lim, "DTD-Miner: a tool for mining DTD from XML documents. Proceedings of Advanced Issues of E-Commerce and Webbased Information Systems", pp. 144-151, 2000

[7] Elisa Bertino, Giovanna Guerrini, Marco Mesiti, " A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications", Information Systems 29, pp 23-46, 2004.

[8] Shanzen Yi, Bo Huang, Weng Tat Chan, "XML application schema matching using similarity measure and relaxation labeling", Information Sciences, 2004

[9] Fauzi M. Ali, Weichang Du, "Towards reuse of object-oriented design models", Information and Software technology, pp 499-517, 2004

[10] Li Bin, Zhou Yun-fei, Tang Xiao-qi, " A Research on open CNC system based on architecture/component software reuse technology", Computers in Industry, pp 73-85, 2004

[11] 이정수, 정상혁, 주경수, "EJB 컴포넌트 기반의 XML 저장관리 시스템 설계 및 구현", 인터넷정보학회논문지, 제3권3호, 75-84, 2002
[12] 이상태, 주경수, "XML DTD의 JDBC 기반 SQL 스키마로의 변환", 인터넷정보학회 논문지, 3권 1호, pp 29-40, 2002.
[13] 김진성, 송행숙, 최동윤, "XML 기반의 소프트웨어 공동 작업의 위한 UDXF 저장소의 관리자", 3권 1호, pp 71-79, 2002
[14] 김행곤, 최화정, "인터넷 응용 시스템 개발을 위한 컴포넌트 기반 환경 연구", 소프트웨어공학회연구지 제3권 1호, pp 52-55, 2000.
[15] 조용호, 최윤석, 정기원, "XML을 이용한 컴포넌트 인터페이스 명세 기법", 정보처리학회 소프트웨어공학회논문지, 제 5권 2호, pp 3-16, 2002
[16] 조병호, "서명매칭 기법을 이용한 컴포넌트 저장/검색 엔진의 설계", 관동대학교산업기술연구소논문지 제19호, pp 75-79, 2004

◐ 저 자 소 개 ◑

조 병 호(Byung-ho Cho)
1983년 인하대학교 전자공학과 졸업(학사)
1990년 뉴욕공대 대학원 컴퓨터학과 졸업(석사)
2006년 숭실대학교 대학원 컴퓨터학과 졸업(박사)
2006년~현재 관동대학교 멀티미디어학과 교수
관심분야 : 소프트웨어공학, 데이터베이스, etc.
E-mail : bhcho@kwandong.ac.kr