반 구조적 Web 문서를 위한 정보수집기 개발

정종석* · 오동익* · 이경호** · 김중배**
* 순천향대학교 공과대학 정보기술공학부
** 한국 전자통신 연구원 전자거래 연구부

Development of an Information Gatherer for Semi-Structured Web Documents

Jong-Seok Jeong*, Dong-Ik Oh*, Kyeong-Ho Lee**, and Joong-Bae Kim**
* Division of Information Technology Engineering, Soonchunhyang University
** Electronic Commerce Department, ETRI

요 약

SECOS는 분리개발 된 객체 컴포넌트를 통합함으로써 새로운 응용 시스템을 작성할 수 있는 CBSE(Component Based Software Engineering)기법을 바탕으로 한, 웹 기반 정보 저장/검색의 전형적 모델을 제공하는 소프트웨어 시스템이다. 현재 본 연구팀은 SECOS시스템의 적용 사례로서 e-business를 위한 쇼핑몰을 구축하고자 검색, 지불, 응용분야의 모듈들을 S/W 컴포넌트로 개발 중에 있다. 본 논문에서는 그 중, SECOS의 검색분야에서 상품정보를 보다 효율적으로 수집하고 사용자에게 양질의 정보를 제공하기 위해 필수적으로 요구되는 정보 수집 서비스의 구조에 대하여 설명하고, 특히 비 정형화 된 XML 웹 소스에서의 정보수집 방법에 있어 현재 진행중인 연구에 대하여 설명하고자 한다.

1. 서론

SECOS[1] 시스템[1]은 다양한 웹 지향 정보검색 시스템 구축을 위한 정형화된 하나의 모델을 제공하고자 개발되어지고 있는 통합형 웹 기반 정보 저장/검색 시스템이다. SECOS는 CBSE (Component Based Software Engineering) 기법[1]에 바탕을 두고 설계되었으며, 기존의 다양한 소프트웨어 컴포넌트들을 통합하고 새로운 컴포넌트를 개발함으로써 구현되어지고 있다. 그림 1은 이러한 SECOS시스템의 전체적인 구조를 보여주고 있다.

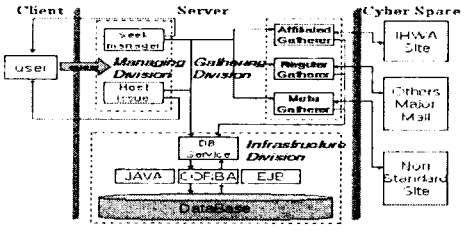

그림 1. SECOS 시스템 구성도

SECOS시스템에서는 6개의 서버측 컴포넌트들이 Managing, Gathering, Infrastructure의 3가지 Division으로 분리되어 제공된다. Managing Division은 시스템에서의 동작을 컨트롤하는 역할을 담당한다. Gathering Division은 3가지의 정보수집기로 구성되어 있고 이들은 분산된 환경에서의 정보를 수집하는 역할을 담당한다. 마지막으로 Infrastructure Division은 모든 시스템자원과 관련된 데이터 및 EJB와 CORBA 객체에 대한 관리와 정보 저장 역할을 담당한다.

SECOS시스템의 Gathering Division에서 제공하는 정보수집기는 Affiliated Gatherer와 Regular Gatherer 및 Meta Gatherer의 세 가지인데, 이들을 통해 SECOS 시스템은 분산된 환경에서 정보 수집 활동을 할 수 있게 된다. Affiliated Gatherer와 Regular Gatherer는 각각 동일한 SECOS시스템과 DTD가 공개된 XML문서에 대한 정보교환 및 수집을 담당한다. 본 연구진에서는 이들 Gatherer를 CORBA 미들웨어 및 인터넷 프로토콜을 사용하여 이미 구현하고 이를 보고 한 바 있다[1]. Meta Gatherer는 웹 상에서 표준화 되어있지 않은 정보, 즉, 일반적인 웹 문서(HTML)나 DTD가 정의되어 있지 않은 XML문서에

1) 본 연구는 정통부 지원 ITRC사업에 의해 수행되었음

서 제공되는 정보를 추출하기 위하여 필요한 Gatherer로서, 아직까지 웹에서 제공되는 대부분이 이러한 범주의 표현방식으로 제공되고 있는 것을 감안할 때 [3], 정확하고 포괄적인 정보를 제공해야 하는 검색 서비스에 반드시 필요한 모듈이라 할 수 있다.

본 논문에서는 이러한 Meta Gatherer의 기능 중, DTD가 정의되어 있지 않은 XML문서에서의 정보수집 및 추출기능의 구현과 그의 활용방법에 대해 설명하고자 한다.

2. 관련연구

비 정형화 된 문서에서 상품정보를 추출하기 위해 필요한 가장 핵심적인 기술은 문서내의 정보를 계층구조로 표현하고 이를 통해 정보를 추출하는 기술이다. 비 정형화된 데이터 소스에서 정보 추출을 위해서는 Data Mining분야에서 활발한 연구가 진행되고 있고, 특히 우리의 연구와 직접적인 연관을 가진 HTML 문서에서의 정보추출에 관한 작업도 Wrapper[4]를 활용한 방식 등을 통하여 활발히 진행되고 있으나 아직까지 만족 할 만한 정보 추출이 이루어지고 있지는 못하다.

특히 우리가 구현한 DTD가 제공되지 않는 XML문서 (반정형화 된 문서)에서 DTD를 추출하는 연구는 Bell 연구소의 XTRACT[2]와 Singapore의 Nanyang Technological University[3], 그리고 IBM Alphaworks의 DDbE등에서도 이루어졌으며, 3가지 연구 방법은 모두 반 정형화된 데이터 소스에서 DTD를 추출하기 위하여 트리나 그래프를 이용하여 계층적인 구조를 생성하고 이 구조에 보다 효율적이고 함축적인 DTD를 생성하기 위한 알고리즘을 적용하는 방법을 채택하고 있다. 이들은 모두 반 정형화된 데이터 소스들에서 유효한 정보를 추출할 수 있는 계층적인 데이터 구조를 생성하는 연구로서 우리가 필요로 하는 정보추출기와 직접적인 연관성을 가지고 있으나 전체적인 DTD 문법을 활용할 수 없다거나, 다른 프로그램 모듈에서 인터페이스가 될 수 없다는 제약점을 가지고 있다. 또한 이들은 XML문서들의 공통된 구조를 추출하기 위하여 먼저 XML문서 각각의 구조를 트리나 그래프를 이용하여 표현하고 이들을 통합하는 2단계 알고리즘을 채택하고 있다. 이에 본 연구에서는 독립된 모듈로 구현되지만 인터페이스를 통해 다른 모듈에서 활용이 가능하고 기존의 2단계 DTD추출 알고리즘에서 각 XML문서의 구조를 트리나 그래프로 변환하는 단계를 삭제한 향상된 알고리즘을 적용한 DTD추출기를 구현 할 필요성을 느끼게 되었다.

3. 본론

SECOS시스템의 Gathering Division은 분산환경에서 다양한 정보를 수집하여 사용자에게 제공한다. 많은 웹 상품정보들이 HTML 문서형식으로 혹은 DTD가 제공되지 않는 XML문서들과 같이 비 정형화 된 형태로 정보를 표현하는 것을 감안 할 때, 이러한 데이터 소스들로부터 상품정보를 수집, 추출하는 기능을 담당하는 Meta Gatherer의 구현은 완벽한 상품정보 수집기능을 제공하기 위하여 필수적이다. 우리가 현재 구현한 반 정형화 된 문서에서의 정보 추출기능은 이러한 Meta Gatherer의 기능 중의 하나이며, 이를 통해 Meta Gatherer는 DTD가 제공되지 않는 XML문서에서의 정보 추출을 수행한다.

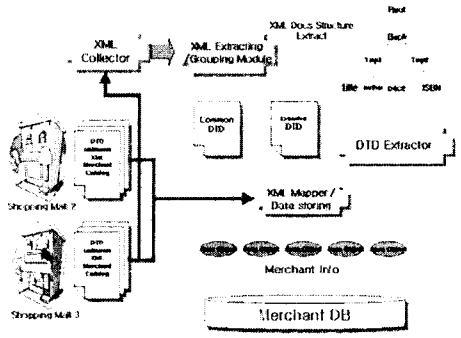

그림 2. Meta Gatherer의 시스템 구조

Meta Gatherer는 "문서수집기", "XML 정보추출 및 그룹화 모듈", "DTD Extractor", "XML Mapper and Data Storing Module"로 구성된다. 그림2는 Meta Gatherer의 이와 같은 기능을 담당하는 모듈의 전체적인 구성도이다.

"문서 수집기"는 웹 사이트들을 정기적 또는 비정기적으로 돌아다니며 XML문서를 수집하는 역할을 담당한다. 이렇게 수집된 XML 정보는 "XML 정보추출 및 그룹화 모듈"을 통해 트리로 표현되고, 이 트리는 계속적인 XML Input에 의해 확장되고 그룹화 되어진다. 모든 XML Input에 의해 트리가 완성되면 "DTD Extractor"가 이 트리를 읽어 문서들에 적합한 DTD를 생성해 내고, 이 DTD에 기초하여 "XML Mapper and Data Storing Module"은 시스템에서 제공하는 상품정보를 표현하기 위해 미리 정의되어진 공통 DTD와의 차이점을 파악하고, 수집되어진 XML 문서들을 공통 DTD에 유효하도록 수정한 후, 하부 DB에 Oracle XDK 유틸리티를 사용하여 이 자료를 저장한다.

3.1 XML 정보추출 및 그룹화 모듈

여러 XML데이터를 위한 단일 DTD를 생성하기 위해서는 입력되는 XML정보를 계층적 자료구조(Tree)로 표현하는 것이 필요하다. XML데이터는 중첩된 구조로 구성되고 (well-formed XML데이터) 이는 n-ary 트리로서 잘 표현 될 수 있으며, 이러한 트리에서 문서의 구조를 파악하기가 용이하기 때문이다. 이렇게 XML문서를 트리로 표현하기 위해 본 연구에서는 다음과 같은 데이터 구조 및 트리구축 알고리즘을 사용하였다.

3.1.1 트리노드

트리는 XML 문서의 각 element를 가리키는 노드로 구성되어진다. 또한 이러한 트리노드는 XML문서에서 각 element가 가질 수 있는 관계들, 즉 Mandatory, Optional, OR, Repetition을 가리키는 속성들로 구성되어져 있다. 또한 element의 이름을 위한 속성과 문서내에서 element의 처리여부를 판별할 수 있는 isCheck속성을 보유하게 된다. 그리고 해당 트리노드가 현재 처리중인 XML 문서의 부모 element로 둘러싸인 단락을 처리하는 도중에 새로이 생성되었는지 아니면 이미 존재하는 트리노드인지를 나타내기 위한 isNewCreate 속성을 가지며, 마지막으로 각 트리 노드가 포함하고 있는 자식노드의 구조를 중복해서 정의하는 것을 피하기 위해 refnodeId속성을 갖는다. OR, Optional, Repetition관계를 나타내기 위해서는 XML element와 직접적으로 연관되지는 않는 임시노드를 활용하였다. 표 1은 트리노드 속성의 종류와 각 속성의 역할을 설명하고 있다.

3.1.2 원본 트리노드 테이블

XML문서를 트리로 구축하는 과정에서는 여러 개의 임시노드를 사용하여 각 XML문서에 포함되어진 element간의 순서관계와 Repetition 및 Optional관계를 표현하고 있다. 이러한 방법 때문에 트리상에서 하나의 element에 대한 복수개의 트리노드가 존재하고 또한 그러한 각각의 트리노드는 자식노드를 가질 수 있다. 이러한 중복정의 문제를 해결하기 위해서 본 연구에서는 트리상에서 실제 해당 트리노드의 구조를 표현하는 원본 트리노드에 대한 경로를 refnodeID라는 이름으로 각 트리노드가 필드로서 저장하도록 설계하였다. 그리고 원본 트리노드들의 리스트를 테이블로서 관리하여 새로운 노드가 트리에 추가되어질 때나 트리노드의 isCheck속성을 갱신할 때 원본 트리노드 테이블에서 원본 트리노드의 경로정보를 추출하여 currentPath와 insertPointer의 위치를 설정하는데 사용할 수 있도록 하였다.

표 1. 트리노드의 속성과 역할

속성명	역할	비고
ele_name	해당 노드의 XML Element이름	트리노드의 공통적인 속성
isMandatory	해당 노드의 관계 중 Mandatory와 Optional을 나타냄	
isRepeat	해당 노드의 관계 중 Repetition을 나타냄	
isCheck	해당 노드의 Repetition관계 확인을 위해, 또한 트리의 구성중 해당 노드의 방문 여부 및 후후의 트리 수정 작업에서 사용	
isNewCreate	해당 노드가 XML 문서를 처리하는 중에 새로이 생성되었는지 아니면 이미 생성되어진 트리노드인지를 나타냄	
refnodeId	해당 노드의 원본트리노드에 대한 링크정보	일반 트리노드 속성
chain	트리노드가 임성 패턴으로 반복되는 것을 표현하는 것으로 다음에 나타나는 트리노드의 트리상의 경로정보를 저장하는 필드	트리의 반복처리를 위한 속성
linked	chain속성과 같이 element의 패턴 반복을 나타내기 위한 것으로 이전의 트리노드를 가리키는 심보를 저장하는 필드.	
isTemp	해당 노드가 임시노드임을 나타냄.	임시노드에 관련된 속성
isOR	임시노드에 사용되는 속성으로 해당 임시노드가 OR관계임을 나타냄	
isOptional	임시노드에 사용되는 속성으로 해당 임시노드가 Optional관계임을 나타냄	
isRepetition	임시노드에 사용되는 속성으로 해당 임시노드가 Repetition관계임을 나타냄	

3.1.2 currentPath 와 insertPointer

XML문서에 포함되어진 각 XML element를 트리노드화 하여 트리에 추가/갱신하기 위해서는 해당 트리노드가 삽입되어질 위치를 가리키고 있는 포인터가 필요하게 된다. 본 논문에서는 이러한 정보를 currentPath와 insertPointer라는 포인터를 통해 표현한다. currentPath는 트리노드가 삽입되거나 검색되어야할 부모노드를 가리키고 insertPointer는 currentPath의 자식노드들 중 검색의 시작점을 지시하는 역할을 담당한다.

3.1.3 트리노드의 chain속성과 linked속성 그리고 startPointer

element의 반복되어지는 패턴을 추출하기 위해서는 정규 표현식을 그래프 형태로 표현한 상태전이도와 같은 형태의 수평적 관계를 나타내는 속성이 동일 레벨상의 트리노드을 위해 필요하다. 트리노드의 chain 속성은 이러한 동일 레벨상의 트리노드들 간의 반복관계 및 연속 관계를 나타낸다. 즉 서로 반복되어지는 패턴을 chain속성에 의해 연결하여 사이클을 추출함으로써 동일 패턴의 반복 여부를 판별하도록 하였다. startPointer는 chain속성에 의해 사이클을 추출할 때

사이클의 시작점을 가리키는 지시자의 역할을 한다. 트리노드의 linked속성은 chain속성과 반대의 개념으로 사용되어진다. chain속성은 형성된 사이클의 추출을 위해 사용되지만, linked속성은 사이클이 형성되어지지 않았을 때 그동안 처리되어진 element이름을 추출하는데 사용된다. 즉, chain속성은 연속되어지는 다음 sibling 노드를 가리키고, linked속성은 연속되는 이전 sibling노드를 가리킨다.

3.1.4 트리의 구성 방법

앞서 설명한 트리 노드 및 데이터 구조를 사용하여 XML 정보추출 및 그룹화 모듈은 XML 문서내의 중첩된 구조를 트리로 표현하게 된다. 크게 3가지 단계로 구분되어질 수 있으며 각 단계에 대한 설명은 다음과 같다.

Step1. 초기화

우선 모든 XML element의 부모인 Root 트리 노드를 생성한다. 그리고 트리내에서 검색 및 삽입될 위치를 가리키는 currentPath와 insertPointer를 생성하고 Root노드를 가리키도록 초기화한다.

Step2. 트리 구축

XML Tokenizer에 의해서 XML문서의 각 element를 추출하고 추출된 element에 대하여 첫 번째로 전달된 element와 같은 이름을 가지는 트리 노드를 검색한다. 두 번째로 그 결과에 따라서 새로운 노드를 추가하거나 혹은 존재하는 노드의 속성을 변경한다. 우선 본 단계에서는 먼저 주어진 element의 이름을 바탕으로 트리에서 검색을 실시하게 된다. 표 2는 이러한 검색의 종류와 그 결과에 따른 동작을 설명하고 있다.

추가되어질 위치를 나타내는 currentPath와 insertPointer의 설정정보와 startPointer의 설정정보를 바탕으로 본 모듈은 트리노드를 새로이 생성하여 트리에 추가하거나 아니면 기존의 존재하는 트리노드의 속성들을 갱신한다. 참고적으로 일곱번째 비교인 next sibling과의 비교에서 next sibling이 임시노드인 경우에는 임시노드의 첫 번째 자식 노드와의 비교가 수행된다.

표 2의 비교 결과에 대한 설명은 다음과 같다.

표 2. 트리구축 알고리즘

currentPath의 자식유무	currentPath가 가리키는 노드의 isNewCreate필드	currentPath가 가리키는 노드가 임시노드 여부	임시노드 종류	insertPointer가 가리키는 노드와 비교	startPointer의 설정 유무	insertPointer의 next sibling과의 비교	insertPointer의 이전 sibling들과의 비교	결과
단말노드	true							①
	false							②
비 단말노드	true	임시노드	OR					③
			Optional					③
			Repetition		설정되어짐	동일		④
						다름		⑤
					초기값유지	없음	동일노드 존재	⑥
							존재하지 않음	⑦
		비 임시노드		동일	설정되어짐			⑤
					초기값유지			⑧
				다름	설정되어짐	동일		④
						다름		⑤
					초기값유지	없음	동일노드존재	⑥
							존재하지 않음	⑦
	false	임시노드	OR					⑨
			Optional					⑨
			Repetition		설정되어짐	동일		④
						다름		⑤
					초기값유지	동일		⑩
						다름		⑪
						없음	동일노드 존재	⑥
							존재하지 않음	⑦
		비 임시노드		동일	설정되어짐			⑤
					초기값유지			⑧
				다름	설정되어짐	동일		④
						다름		⑤
					초기값유지	동일		⑩
						다름	동일노드존재	⑥
							존재하지 않음	⑪
						없음	동일노드존재	⑥
							존재하지 않음	⑦

① 새로운 트리노드를 currentPath 자식노드로 추가.
② 이러한 경우는 PCDATA를 표현하는 트리노드에서만 나타날 수 있음.
③ 임시노드가 아닌 경우와 같음.
④ insertPointer가 가리키는 노드의 chain속성과 next sibling노드의 linked속성을 상호 연결시킴.
⑤ linked 속성에 의해 연결되어진 트리노드를 추출하여 currentPath의 자식노드로 추가. currentPath가 임시노드이면 currentPath의 부모의 자식노드로 추가.
⑥ startPointer에 현재 insertPointer의 값을 저장하고 현재 insertPointer가 가리키는 노드의 chain속성과 검색되어진 노드의 linked속성을 연결시킴.
⑦ isOptional속성이 true인 임시노드를 생성하여 추출되어진 element에 대한 트리노드를 추가.
⑧ insertPointer가 가리키는 트리노드의 isRepeat 속성을 true로 설정.
⑨ 임시노드가 아닌 경우와 같음.
⑩ 다음 sibling노드의 isCheck 속성을 true로 설정.
⑪ isOR 속성이 true인 두 개의 임시노드를 생성하여 currentPath의 자식노드들 중 이미 처리되어진 노드를 제외한 트리노드들과 현재 추출되어지는 element들을 분리.

트리노드의 추가나 갱신시에 본 모듈이 취하는 동작을 살펴보면 다음과 같다. 우선, 새로운 트리노드를 추가할 때 먼저 추출되어진 element이름으로 원본 트리노드 테이블에서 검색을 실시한다. 검색결과 이미 해당 이름이 등록되어 있다면 이름과 관련된 원본 트리노드의 경로정보를 추출하여 생성되어진 트리노드의 refnodeID필드를 설정한다. 그리고 currentPath와 insertPointer의 위치는 추출되어진 링크정보로 설정한다. 그렇지 않고 원본 트리노드 테이블에 해당 이름으로 저장되어진 정보가 없다면 생성된 트리노드의 이름과 트리상의 경로 정보를 원본 트리노드 테이블에 저장하고 currentPath와 insertPointer의 위치는 추가되어진 트리노드를 가리키도록 한다. 이미 존재하는 트리노드의 속성을 변경할 때는 해당 트리노드의 isCheck속성 및 필요한 속성을 변경한다. 변경한 후에 currentPath와 insertPointer의 위치를 설정해야 하는데, 추출되어진 element이름을 원본 트리노드 테이블에서 검색, 추출되어진 경로정보로 currentPath와 insertPointer의 위치를 조정한다.

표2의 알고리즘을 사용하여 표3의 XML문서의 데이터 구조를 트리로 구성하면 그림3과 같다.

표 3. 예제 XML문서

```
<?xml version="1.0" enc oding="EUC KR"?>
<books>
  <book>
    <title>JBuilder 4.0 </title>
    <author>
      <first name>JongSeok</first name>
      <last name>Jeong</last name>
      <first name>JongSeob</first name>
      <last name>Lee</last name>
    </author>
    <title>CBSE</title>
    <authoe>
      <name>Nam Kyu Cho</name>
    </author>
  </book>
</books>
```

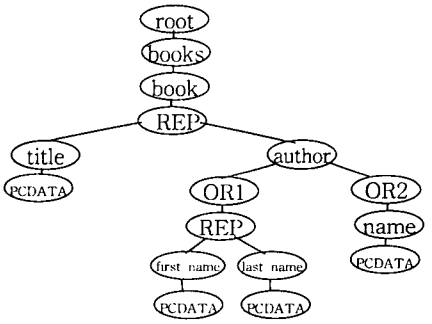

그림 3 트리구축 예

Step3. 트리 수정 및 보완

Step 2에 의해 구성된 트리는 상당한 복잡도를 갖는 형태가 될 것이다. 이 단계에서는 추출되어질 DTD의 가독성을 높이기 위해 구축되어진 트리를 보다 간략화 하는 작업이 수행된다.

3.2 DTD Extractor

DTD Extractor는 XML정보추출 및 그룹화 모듈에서 생성된 트리를 실제 XML DTD화일로 변환 시켜주는 모듈이다. 구축되어진 트리는 XML문서의 계층적 구조와 각각의 부모와 자식간의 관계를 그대로 포함하고 있기 때문에, DTD Extractor는 트리를 level별로 순회하면서 노드들의 순서와 부모 element와의 관계, 즉 각 노드에 표현된 Mandatory, Optional, OR, Repetition과 같은 정보를 활용하여 DTD화일을 생성하게된다.

표3의 XML문서에 대해 생성된 DTD는 표4와 같다.

표 4. 예제 XML문서의 DTD

```
<!ELEMENT root books>
<!ELEMENT books book>
<!ELEMENT book (title, author)+>
<!ELEMENT title #PCDATA>
<!ELEMENT author (first-name, last name)+ | name>
<!ELEMENT first name #PCDATA>
<!ELEMENT last-name #PCDATA>
<!ELEMENT name #PCDATA>
```

3.3 XML Mapper and Data Storing Module

XML 문서수집기가 수집한 문서에 포함된 데이터들은 추출되어 DB에 저장되어야 한다. 이러한 동작을 수행하기 위해서 본 연구에서는 추출 되어진 DTD와 시스템의 공통 DTD간의 상이점을 파악하여 수집되어진 문서들을 공통 DTD에 유효하도록 수정하는 모듈과, 이렇게 수정되어진 XML문서에서 상품 정보를 추출하여 DB에 저장할 수 있는 XML Mapper and Data Storing Module을 개발하였다. 그림 4는 XML Mapper and Data Storing Module의 구조 및 동작에 대하여 설명하고 있다.

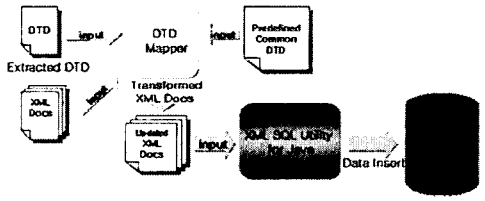

그림 4 XML Mapper and Data Storing Module의 구조 및 동작

4. 결론

SECOS시스템의 Gathering Division이 보다 광범위한 정보를 수집하도록 하기 위해서는 Meta Gatherer의 역할이 중요하다. 우리는 Meta Gatherer의 DTD생성 모듈과 DB저장 모듈의 구조를 설계하고 Java언어를 사용하여 이를 구현하였고, 이를 위해 개발된 알고리즘을 본 논문에서 설명하였다.

여러 가지 기존의 툴들도 전체적인 시스템을 위해 활용되었는데, XML문서의 토큰별 추출을 위해서는 Oracle의 xmlparserv2.0의 XMLTokenizer를 사용하였고, XML정보추출 및 그룹화 모듈의 트리를 구성하기 위하여서는 Java의 swing 컴포넌트를 사용하였다. DTD 추출모듈을 구현하여 현재 여러 종류의 XML문서들에 대해서 실험중이며, 성능향상을 위한 지속적인 수정과 갱신작업을 수행하되고 있다.

임시노드를 사용한 트리구조는 상당히 복잡하다. 이렇기 때문에 트리를 DTD로 변환하면 대단히 복잡하고 가독성이 떨어지는 DTD가 종종 생성된다. 생성된 트리를 순회하면서 트리의 구조를 단순화시킬 수 있는 연구가 앞으로 계속 연구되어야 할 것이다.

[참고문헌]

[1] Dong-Ik Oh and Jong-Suk Jung Effective Web-Based Information Gathering Services of IHWA. Proceedings of ICEIC'2000 International Conference, Shenyang, China, 2000.8 pp. 202-205.

[2] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: A System for Extracting Document Type Descriptors from XML Documents. Bell Labs Tech. Memorandum, 1999.

[3] Re-engineering Structures from Web Documents by C.-H. Moh, E.-P. Lim, W.-K. Ng. Proceedings of the 5th ACM International Conference on Digital Libraries (DL2000), San Antonio, Texas, USA, June 2-7, 2000.

[4] N. Ashish and C. A. Knoblock. Semi-automatic wrap-per generation for internet information sources. In Proceedings of Coopis Conference, 1997.

[5] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semi-structured data from the web. Proceedings of Workshop on Management of Semi-structured Data, pages 18-25, 1997.

[6] N. Kushmerick, D. Weil, and R. Doorenbos. Wrapper induction for information extraction. In Proceedings of Int. Joint Conference on Artificial Intelligence (IJCAI), 1997.

[7] Cay S. Horstmann, Gary Cornell, Core Java Volumn I - II, Sun Microsystems Press, 1998

[8] Tom Valesky, Enterprise JavaBeans - Developing Component-Based Distributed Applications, Addison-wesley, 1999