소프트웨어 소스 코드의 저작권 관리를 위한 디지털 라이센스 프로토타입

Digital License Prototype for Copyright Management of Software Source Code

차 병 래* 정 종 근** 오 수 열***
ByungRae Cha JongGeun Jeong SooLyul Oh

요 약

국가 경쟁력 제고를 위해서도 디지털콘텐츠에서 확대하여 소프트웨어 소스 코드에 대한 지적재산권 제도와 기술의 정비는 매우 중요한 의미를 지닌다. 이러한 지적재산권 중에서 특히 소프트웨어 보호에 대한 인지는 매우 낮은 편이다. 소프트웨어 소스코드의 소유권 분쟁이 발생 시 소유권을 증명하기 위해서는 원본의 소프트웨어 소스코드를 판별해야만 하는 문제점을 갖고 있다. 또한 소프트웨어가 복제되어도 복잡성과 독해 능력 부족으로 정확한 판정을 내리기가 쉽지 않다.

본 논문에서는 이러한 소프트웨어 복제에 대한 판별을 하나의 개별 코드 단위로 시행하지 않고, 전체 소스가 가지는 구조적 일치성을 기반으로 복제를 판별할 수 있는 XML 타입의 디지털 라이센스 프로토타입을 개발하였다. 소프트웨어는 구조적으로 Context Free Grammar 기반이며, 그러므로 BNF 표기 형태로 표현할 수 있고, 이는 다시 계층 구조로 표현할 수 있기 때문에 가능한 것이다. 그러므로 소프트웨어 소스코드의 구조적 일치성을 비교하기 위한 계층구조를 갖는 소스코드의 아키텍처를 표현할 수 있다.

Abstract

The digital contents expand into software source code and maintenance of technology and IPR about source code have a very important meaning to international competition. The recognition about software security is very low specially among these Intellectual Property Rights. On occurring disputation property, we have to prove the fact, there is a problem to discriminate the original source code. Also, it is hard to accurate decision that is correct to complexity and the lack of read and understand ability even if software is reproduced.

In this paper, we don't enforce distinction about software reproduction by one individual code unit. And we developed digital license prototype of XML that can distinguish reproduction based on structural conformability of whole source codes. Software has Context Free Grammar in structure and presents BNF notation type, it is apt to present hierarchical structure. Then, we can express architecture of software source code by hierarchical structure to discriminate structural conformability.

In this paper, we make a study of the digital licence prototype for discriminate the original source code. Reserved words of software source code by parsing express to XML file that have hierarchical structure. Then, we can express architecture of software source code by tree structure form instead of complex source code.

☞ Keyword : S/W Source Code, Digital License, Copyright Management

1. 서 론

전통적으로 저작권은 주로 법적인 관점에서 다루어지고 연구되어 왔다. 저작권의 본질, 그것의 보호의 범위, 강제와 침해 등이 일반적인 연구의 대상이었다. 우선, 저작권은 여러 가지 창조적인 저작물 시장을 위한 안전하고 안정적인 환경을

* 정 회 원 : 호남대학 컴퓨터공학과 교수
chabr@honam.ac.kr
** 정 회 원 : 호남대학교 컴퓨터공학과 초빙교수
jkjeong@honam.ac.kr
*** 정 회 원 : 목포대학 정보공학부 교수
syoh@mokpo.ac.kr
[2006/05/24 투고 - 2006/06/21 심사 - 2006/08/31 심사완료]

제공하는 법적 시스템이다. 오늘날 비즈니스의 영역에서 라이센싱, 투자, 거래 및 이전의 문제와 관련하여 저작권에 대한 관심이 크게 증대되고 있지만, 이것도 근본적으로는 저작권이 영향력 있는 법적 시스템이라는 속성에서 비롯되는 것이다. 최근 들어서 저작권의 경제적 영향에 대한 학계의 관심과 저작권의 국민경제적 역할에 대한 계량적 측정에 대한 정책담당자들의 관심이 세계적으로 증대되는 것은 일상적인 경제생활의 생산, 유통, 소비의 전반에 걸쳐 지속적으로 확대되어 온 저작권의 영향력이 이러한 수준을 넘어섰음을 의미한다. 이러한 저작권의 경제적 가치의 증대에는 여러 가지 이유가 있다. 첫째, 저작권에 대한 관심은 부분적으로는 비물질적인 생산요소가 보다 주목을 받는 지식기반사회에서 증가하고 있는 지적재산(Intellectual Property : IP)의 역할에 대한 인식이 증대된 결과이다. 저작권은 창의성과 정보에 기반을 둔 산업의 성장과 산업의 생산성, 고용 및 투자를 위하여 중요하다. 둘째는, 디지털기술의 발달로 인하여 저작권보호의 대상물의 범위가 크게 증대되었다는 것이다. 소프트웨어, 멀티미디어 및 기타 기술기반 산물로부터 경제적 이득은 막대해지고 있다. 셋째는 디지털 혁명의 결과, 저작권 보호 대상들이 전자상거래의 주요한 요소가 되었다는 점이다. 과거 산업사회의 태동과 발전과정에서 뿐만 아니라, 21세기 정보화 사회의 발전에 있어서도 이러한 지적재산권 제도는 중요한 역할을 하고 있다. 우리나라의 국가 경쟁력 제고를 위해서도 디지털콘텐츠에서 확대된 소프트웨어의 소스 코드에 대한 지적재산권 제도와 기술의 정비는 매우 중요한 의미를 지닌다.

디지털 콘텐츠의 저작권 보호를 위한 연구에 비해서 소프트웨어의 소스코드에 대한 저작권을 관리하기 위한 기술은 아직도 초기 연구단계에 있다. 소프트웨어 소스코드의 소유권 분쟁이 발생시 소유권을 증명하기 위해서는 원본의 소프트웨어 소스코드를 공개해야만 하는 문제점을 갖고 있다. 본 논문에서는 소프트웨어 소스코드의 원본 공개로 인한 저작권 침해와 기술 유출을 막기 위한 소프트웨어 소스코드의 디지털 라이센스 방식의 프로토타입을 설계 및 제안한다.

2. 관련 연구

디지털 컨텐츠란 디지털로 되어 있는 문자, 소리, 화상, 영상 등의 형태로 이루어진 정보 내용물이라고 할 수 있다. 본 연구에서는 디지털 컨텐츠의 영역을 확장하여 소프트웨어의 소스코드를 포함하고자 한다. 소프트웨어의 소스코드는 논리적 사고과정을 컴퓨터가 처리할 수 있는 고급 언어 또는 그에 준하는 프로그래밍 언어를 통해 기술해 놓은 일종의 디지털 컨텐츠이다. 디지털 컨텐츠가 갖는 특징은 디지털 컨텐츠의 비손실 복제 가능, 수정과 복원이 용이, 저렴한 복사비용, 저작권을 표현할 수 있으며, 디지털 경제(digital economy) 활동인 전자상거래가 가능하다는 점이다. 그러나 디지털 컨텐츠는 무한히 복사 가능하다는 장점을 갖는 반면에, 무한히 복사가능함으로서 컨텐츠 저작권이 보호되지 않을 수도 있다는 단점을 갖는다. 이를 예방하기 위해서는 DRM (Digital Rights Management) 기술이 필수적으로 필요하다.

DRM을 이루는 기술은 크게 두 가지로 나누어 볼 수 있다. 첫 번째는 저작권을 관리해주는 기술이고, 또 다른 하나는 저작권을 보호해 주는 기술이다. 대부분의 상업화된 DRM 기술은 저작권 보호기술이 주를 이루고 있다. 저작권 보호 기술은 저작권 관리 기술에서 정의하는 일련의 원칙과 시나리오를 강제화(enforcement)하는 기술로 이해할 수 있다. 저작권 보호 기술에는 암호화, TRM, 워터마킹 그리고 핑거프린팅 기술 등이 있다. 저작권 관리 기술은 범세계적으로 통일된 일련의 디지털 저작물 관리 체계를 마련하기 위한 것으로 여러 저작권 단체들을 중심으로 컨텐츠의 식

별자, 컨텐츠의 메타데이터, 컨텐츠의 메타데이터 기술 언어에 대한 표준화 작업이 진행되고 있다. 저작권 관리 기술에는 DOI, MARC, DC, INDECS, ONIX, XrML, ODRL 등이 있다. 컨텐츠 식별자 DOI(Digital Object Identifier)[1]는 모든 도서에 부여되는 ISBN 번호 체계처럼 인터넷의 디지털 컨텐츠에 적용해 인터넷 상거래를 자동화하고, 저작권을 보호하기 위한 새로운 형태의 식별자 시스템으로 미국출판협회(AAP)에 의해 최초 제안되었다. DOI 체제는 출판물 등 텍스트 정보 분야에서 출발하였지만 현재는 음악·영화·애니메이션과 같은 모든 디지털 컨텐츠, 더 나아가 각종 제품의 인터넷 유통에도 적용할 수 있게 그 응용 분야가 확대되었다. MARC(MAchine Readable Catalog)[2]는 전자책 분야의 오래된 표준으로서 각국의 주요 도서관을 중심으로 발전해오고 있다. DC (Dublin Core)[3]는 네트워크 환경에서 상거래를 수반하는 모든 정보 자원을 기술하기 위한 메타데이터이다. INDECS(INteroperability of Data in E-Commerce Systems)[4]는 전통적인 메타데이터인 DC와 같은 자원 기술을 가지고 있으나, 인간과 지적재산권 계약 그리고 이들 사이에 연결 요소를 추가적으로 포함하고 있다. INDECS의 특징으로는 첫째는 논리적 측면에서 IFLA FRBR 모델을 사용해 모든 정보자원을 4가지 형태 중 하나로 규정한다. 둘째는 저작권(IPR:Intellectual Property Rights) 처리이며, 마지막으로는 상거래 트랜잭션에 대한 투명한 정보를 제공한다. INDECS는 기존 단체에서 제안하는 식별자 또는 메타데이터와 호환성을 갖게 설계되었으며, 특히 교환을 위한 RDF(Resource Description Framework) 모델을 채택하고 있다. 또한 현재 이슈화되고 있는 MPEG-7, MPEG-21의 권리와 상거래에 대한 보완적 요소를 제공하고 있다. ONIX(ONline Information eXchange)[5]는 AAP와 아마존 인터넷 서점의 주도로 개발된 도서의 전자상거래를 지원하기 위한 메타데이터이다. ONIX는 INDECS를 기반으로 하고 있다. XrML은 (eXtensoble rights Markup Language)[6]은 현재 가장 많이 사용되는 XML 기반의 저작권 표현 언어이다. XrML은 디지털 컨텐츠 및 웹 서비스와 관련된 권리와 조건들을 표현하고, 저작권자, 컨텐츠 제공자, 사용자 간에 권리 항목들의 표준을 제정하기 위한 목적에서 시작되었다. 컨텐츠 제공자는 XrML을 이용하여 사용자에게 특정 권한을 부여할 수 있으며 이러한 모든 권한에 대해 사용기간과 조건을 명시할 수 있다. XrML은 MPEG-21 REL(Rights Expression Language)의 기반으로 채택되었다. MPEG REL은 국제표준(ISO)으로 채택되었으며, 최근에 나오는 DRM 솔루션은 XrML을 바탕으로 하고 있다. ODRL(Open Digital Rights Language)[7]은 컨텐츠에 대한 권리 정보를 표현하기 위하여 정의된 표준 언어 및 어휘이다. 권리 언어를 표현하는 ODRL Expression Language와 데이터 사전에 들어가는 요소들을 정의하는 ODRL Data Dictionary가 표준화의 대상이며, 모두 XML 스키마를 사용하여 표현된다.

그리고 소프트웨어 소스코드의 패턴 매칭에 대한 연구는 Rieger[8]과 Yang[9]에 의해서 연구되어졌다. Rieger는 소프트웨어의 소스코드가 중복되거나 복제된 부분을 시각화하여 탐지하는 연구를 수행하였고, Yang은 두 개의 다른 프로그램을 구문론적으로 동일한 코드를 찾는 연구를 수행하였다. 프로그램 표절 검출 S/W로는 국외는 SIM[10], Dup[11], Plague, YAP[12], YAP3, MOSS[13] 등이 있다. 국내는 KAIST의 clonechecker와 부산대의 LOFC가 있으며[14], 프로그램 심의위원회의 exEyesLight[15]가 있다.

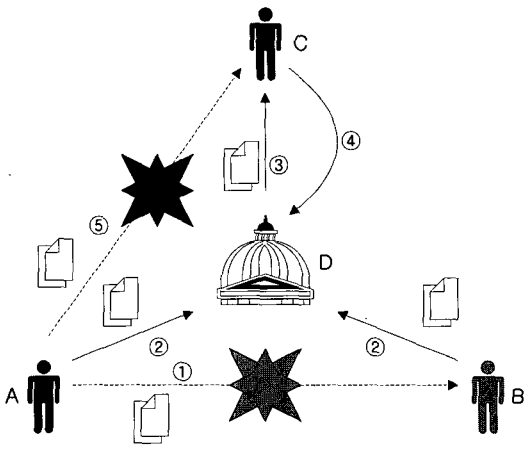
본 연구는 소프트웨어의 소스코드에 대해서 DRM을 지원하기 위한 기초 연구이다, DRM의 저작권 보호 기술보다는 저작권 관리 기술을 제안하며, DRM의 영역을 소프트웨어의 소스코드까지 확장한다. 소프트웨어 소스코드에 대한 DRM 기술은 부재하며, 저작권 보호기술로는 1:1 관계의 거래에서만 임의의 효력을 갖은 암호화 기술뿐이며, 초기연구단계에 머물러 있다. 소프트웨어 소스코드는 제품을 생산하는 공장의 설계도면을 뛰어

넘어서 제품생산라인과 같은 아주 중요한 디지털 콘텐츠 자원이다. 그러나 정작 이 자원들을 지원하는 DRM 기술은 너무나도 부족한 상태이다.

3. 소프트웨어 불법 유출에 의한 분쟁

소프트웨어 개발자 A, B, C와 법원 D의 소프트웨어 불법 유출에 의한 문제 사항을 (그림 1)과 같이 제기한다. 소스코드의 원본의 소유자 A와 불법 유출자 B에 의해서 분쟁은 발생한다(그림 1의 ①). 법원은 이 문제를 해결하기 위해서 개발자 A와 불법 유출자 B의 소스 코드를 제출할 것을 요구한다(그림 1의 ②). 그러나 소프트웨어의 소스코드는 전문가가 감별해야 하기 때문에 법원 D는 소프트웨어 전문 개발자 C에게 소프트웨어 소스코드의 원본과 불법 유출본과의 차이점을 의뢰하게 된다(그림 1의 ③). 소프트웨어 전문 개발자 C는 원본과 불법 유출본을 비교하기 위해서 소스코드를 이해해야 한다. C에 의해서 원본과 불법 유출본을 감별하는 것은 전적으로 C의 주관성 및 소스코드의 이해 능력에 전적으로 의지하게 되어 객관성이 부족하게 된다(그림 1의 ④). 여기서 제 2의 소스코드 불법유출 문제가 발생할 수도 있게 된다(그림 1의 ⑤). 의도하지 않은 소스 코드의 지적재산권이 침해되는 문제가 발생하게 된다. 즉 소소코드의 원본과 불법 유출본을 감별하기 위해서 C가 소스코드를 이해하다보니 자동적으로 C에게 제 2의 소스코드가 불법 유출되는 문제가 발생하며, 제 2의 분쟁 소지를 갖게된다.

(그림 1)의 저작권 침해 문제의 발생을 소프트웨어 소스코드의 디지털 라이센스에 의해서 제 2의 분쟁을 사전에 제거할 수 있다. 소프트웨어 개발자 A, B, C와 법원 D의 소프트웨어 불법 유출에 의한 문제 사항을 소프트웨어의 소스코드 대신에 디지털 라이센스로 대처하므로서 제 2의 분쟁을 사전에 예방할 수 있다. 또한 C에 의한 주관적 감별을 디지털 라이센스의 패턴 분류 프로그램으로 대처함으로서 어느 정도 객관성을 갖출 수 있다.

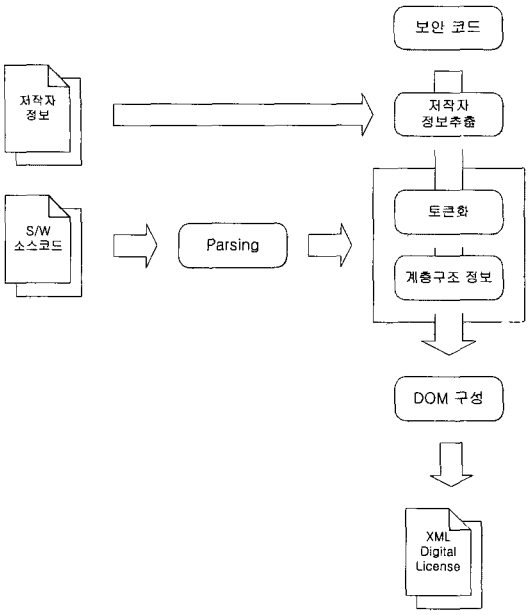


(그림 1) 소프트웨어 분쟁에 의한 지적재산권 침해 문제 발생의 다이어그램

4. 소프트웨어 소스 코드의 디지털 라이센스 설계

일반적으로 소프트웨어 등록 협회에 소프트웨어를 등록하면 하드 카피인 등록증 하나가 배달된다. 최근에는 좀 더 개선되어 웹을 통해서 소프트웨어 등록증을 출력할 수 있다. 그렇지만, 정작 분쟁이 생기면 이를 해결하기 위한 절차가 복잡해진다. 단지, 소프트웨어를 보관하고 있다가 분쟁이 발생하면 복사본을 제공받으며, 먼저 등록한 사람에게 법정에서 우선권을 주장할 수 있는 근거를 제공할 뿐 등록한 소프트웨어에 대한 좀 더 많은 정보를 제공하지는 못하고 있다. 본 논문에서는 하드 카피의 단순한 등록증에서 탈피하여 하드 카피의 내용에 부가적으로 소프트웨어의 소스코드에 대한 많은 정보를 제공할 수 있는 XML 형태의 디지털 라이센스 설계에 대해 연구를 하였다.

소프트웨어의 소스코드에 대해 진정으로 이해하기 위해서는 코드와 아키텍처를 보다 면밀하게 구분할 필요가 있다. 코드 또는 소프트웨어는 컴퓨터에 의해서 수행되어질 소프트웨어를 구성하는 벽돌과 시멘트에 해당한다. 한편 아키텍처는

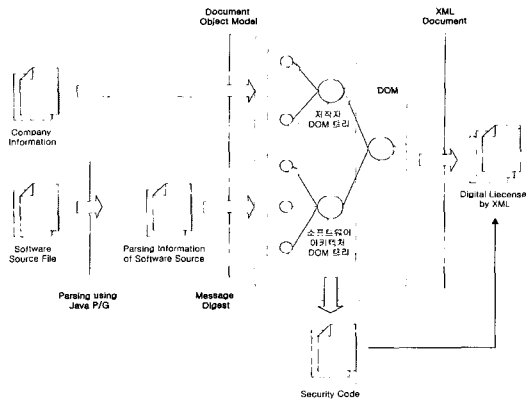


(그림 2) 소프트웨어 소스코드의 디지털 라이센스 생성 블록 다이어그램

코드를 벽돌 삼아 쌓아올린 구조물을 의미하는 것이다. 코드는 행위에 제약을 가할 수도 있고 구조에 영향을 미치기도 한다. 그렇지만 모든 것이 오로지 코드에 의해서만 결정되는 것은 아니다. 본 논문에서는 소프트웨어 소스코드의 디지털 라이센스를 생성하는 과정의 블록 다이어그램(그림 2)와 같이 제안하고 세부사항은 (그림 3)에 나타내었다.

(그림 2)에서 디지털 라이센스를 발급하기 위해서는 먼저, 저작자 정보와 소프트웨어의 소스코드가 입력되어야 한다. 특히 소프트웨어의 소스코드는 파싱을 통해서 예약어와 계층 관계의 정보를 추출한다. 추출된 정보로 DOM 트리를 구성하고 XML 파일로 디지털 라이센스를 발행한다.

(그림 3)에서 소프트웨어 소스코드의 디지털 라이센스의 생성과정은 크게 두 개의 정보 영역인 저작자 DOM 트리 영역과 소프트웨어 아키텍처 DOM 트리 영역으로 구분된다. 저작자 DOM 트리 영역은 하드 카피인 소프트웨어 등록증과 같



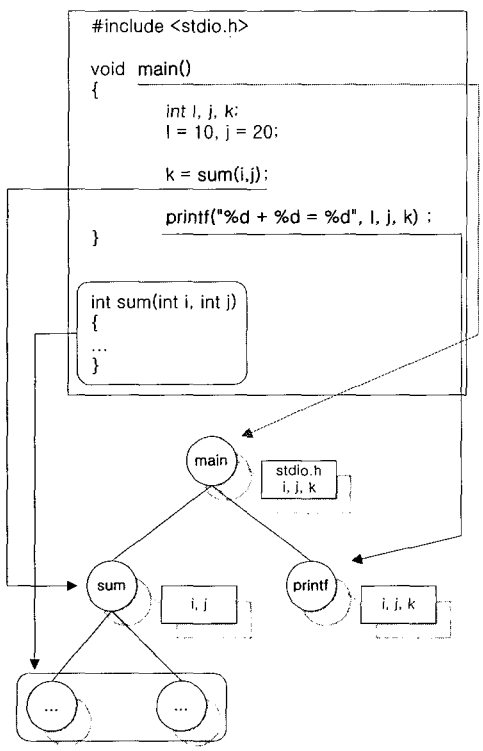
(그림 3) 소프트웨어 소스코드의 디지털 라이센스를 생성 과정

은 정보를 갖고 있으며, 동일한 역할을 수행한다. 하드 카피인 소프트웨어 등록증과 디지털 라이센스의 차이점은 소프트웨어 아키텍처 DOM 트리의 정보를 더 갖는다는 점이다. 소프트웨어 아키텍처 DOM 트리는 소프트웨어 소스코드를 이용해서 정보를 생성한다. 소프트웨어 소스코드를 프로그래밍 언어의 예약어에 의해서 파싱 및 토큰화를 수행한다. 이때 생성된 계층구조 정보와 입·출력 데이터 타입, 서브 함수 정보 등을 이용하여 소프트웨어 아키텍처의 DOM 트리를 구성한다. 추가적으로 저작자 DOM 트리와 소프트웨어 아키텍처 DOM 트리의 무결성과 인증을 제공하기 위한 보안 코드를 생성하고, 이 모든 것을 XML 형식으로 소프트웨어 소스코드의 디지털 라이센스를 생성한다.

4.1 디지털 라이센스를 위한 DOM 트리의 구성

소프트웨어 소스 코드에 대한 암호화 기법은 단지 패키징 기법 중의 하나일 뿐이다. 복호화된 소스코드에 대해서는 어떠한 해결책도 제공하지는 못한다. 소프트웨어 소스코드의 저작권을 표현하는 디지털 라이센스의 프로토타입을 생성하기 위해 필요한 제반기술인 DOM(Document Object Model)은 HTML 문서 구조를 정의하는 모델을

언급할 때 사용한 것이다. W3C[16]는 HTML 뿐만 아니라 XML에 대해서도 상호간에 동작할 수 있는 공통적인 문서 객체 모델을 만들 목적으로 DOM Working Group 이라는 워킹그룹을 만들었다. DOM은 객체들의 표준 집합을 통하여 응용 소프트웨어들에 대한 플랫폼 중립적이며 언어 중립적인 인터페이스를 정의한다. DOM의 처리과정은 XML 문서를 파싱하는 단계와 DOM 트리를 접근하는 단계로 구성된다. DOM에서는 XML 문서가 하나의 트리로 표현되며, 전체 문서는 일종의 중첩된 트리로서 표현된다[17]. 본 연구에서는 소프트웨어 소스 코드의 디지털 라이센스 생성에 DOM 모델을 적용하며, 디지털 라이센스를 바이너리 파일이 아닌 XML 파일로 기술 및 생성하며, XML 문서를 DOM 기술로 처리할 수 있다는 장점을 갖는다.



(그림 4) 프로그램 소스 코드의 소프트웨어 아키텍처 DOM 트리의 변환 과정

본 연구에서는 소프트웨어 소스코드의 디지털 라이센스는 루트 노드(Root Node)인 <DL_PGSCode> 태그로 구성되어 그 아래에 두 개의 자노드(Child Node)를 갖는다. 자노드는 소프트웨어 소스코드의 저작자에 대한 정보를 갖는 저작자 DOM 트리와 소프트웨어 소스코드의 아키텍처를 나타내는 소프트웨어 아키텍처 DOM 트리로 (그림 3)과 같이 이루어진다. 소프트웨어 아키텍처 DOM 트리는 소프트웨어 소스 코드를 파싱하여 예약어, 라이브러리 그리고 함수 등을 DOM 트리로 구축하는 방법을 (그림 4)와 같이 제안한다. C언어 프로그램 소스 코드의 예를 들어, 모든 C 언어의 프로그램은 main() 함수와 서브 함수로 구성된다. 그러므로, main() 함수가 디지털 라이센스의 소프트웨어 아키텍처 DOM 트리의 루트 노드가 되며, 서브 함수는 main() 함수의 자노드로 구성된다. 각 노드에는 사용된 라이브러리와 변수들 자료형, 연산자들의 패턴 정보들을 갖는다.

4.2 소프트웨어 소스코드 분석

대부분의 DRM 기술은 바이너리 파일에 대한 DRM 기술이다. 원래의 디지털 저작물인 바이너리 코드를 패키징을 수행하여 디지털 저작권 관리가 이루어진다. 그러나 디지털 저작물의 원천 요소인 소프트웨어 소스 코드에 대해서는 암호화 이외에는 별다른 방법이 존재하지 않으며, 아직은 연구 초기 단계이다. 암호화 방법 역시 소프트웨어 소스 코드 개발자와 소프트웨어 소스 코드 구입자간의 1차적인 거래에만 저작권 관리를 지원할 뿐 그 이후에 발생하는 거래에 대한 저작권을 보호해 주지 못하고 있다.

소프트웨어 소스 코드에 대한 DRM 기술이 연구 초기 단계인 이유는 소스 코드 자체가 아스키(ASCII) 코드나 유니코드(Unicode)로 구성되어 있기 때문이다. 이런 이유로 인하여, 암호화를 통한 1차적 거래에만이 1:1 관계의 디지털 저작권 관리가 이루어지고 있다.

소프트웨어 소스 코드에 대한 디지털 라이센스를 생성하기 위해서는 먼저 소프트웨어 소스 코드를 구성하는 프로그래밍 언어의 예약어, 라이브러리 그리고 함수 등을 이용한다. ANSI 표준 C 언어인 경우는 32개의 예약어를 정의하고 있다. 또한 많은 C 컴파일러들은 컴파일러 환경의 효율성, 다른 언어와 상호 프로그래밍, 인터럽트, 메모리 관리 등을 지원하기 위해서 다른 예약어 11개를 추가하고 있다. Java 언어의 경우는 48개의 예약어를 정의하고 있다.

소프트웨어의 소스코드를 분석하기 위해서는 패턴 매칭(pattern matching)과 파싱(parsing) 기술이 필요하다. 소프트웨어 소스코드를 파싱하여 의미를 부여할 수 있는 구분된 컴포넌트를 토큰(token)이라 한다. 그리고 토큰은 각각 독립적인 유닛(unit)을 나타낸다. 소프트웨어 소스코드가 갖는 추상적인 표현을 토큰화(tokenizing)를 거쳐야 하며, 구성된 토큰을 이용하여 디지털 라이센스의 정보 및 아키텍쳐를 구성하게 된다. 토큰은 두 가지 형식을 갖게 되는데, 외부 형식(external format)과 내부 형식(internal format)이다. 외부 형식은 소프트웨어 소스코드를 작성할 때 사용되는 텍스트 형식이다. 예를 들어, PRINT라는 예약어는 외부 형식이다. PRINT를 1로 재정의하는 것이 내부 표현이다. 외부 형식을 효율적인 처리를 위해서는 내부 형식으로 전환한다. 소프트웨어의 디지털 라이센스의 정보 표현을 생성할 때 내부 형식으로 표현하면 문자열보다는 정수를 사용하는 것이 훨씬 빠르게 수행되기 때문에 내부 형식으로 표현하면 소프트웨어의 디지털 라이센스 생성처리를 위한 성능상의 이점이 있다. 향후 색인 생성기와 검색기에 사용되는 텀(term)은 토큰을 이용한다.

5. 소프트웨어 소스 코드의 디지털 라이센스를 위한 자노드 패턴의 생성과정

소프트웨어 소스코드의 디지털 라이센스를 생성하면 XML 문서의 최상의 루트 노드는 <DL_PGSCode> 태드이다. 루트 노드는 소프트웨어 저작자 DOM 트리 노드와 소프트웨어 아키텍처 DOM 트리 노드라는 두 개의 자노드를 갖는데, 왼쪽 자노드는 저작자에 대한 정보를 표현하는 DOM 트리로 구성된다. 일반적인 소프트웨어 등록증의 내용과 정보의 무결성을 보장하기 위한 메시지 축약(MD : Message Digest)[18] 코드가 추가된다.

소프트웨어 소스코드의 디지털 라이센스의 핵심 부분은 오른쪽 자노드인 소프트웨어 아키텍처 DOM 트리이다. 이 자노드는 소프트웨어 소스코드를 파싱하여 생성된 예약어 토큰과 소프트웨어의 구조에 의해서 트리 형태의 아키텍처로 (그림 4)와 같은 형태로 생성된다.

5.1 프로그래밍 언어의 예약어

모든 프로그래밍 언어는 예약어를 갖고 있다. 예약어를 코드라하며, 코드를 이용하여 토큰을 생성하고, 토큰과 프로그램의 구조인 토큰의 위치를 이용하여 소프트웨어의 아키텍처를 구축한다. 예를 들어, ANSI 표준 C 언어의 경우는 32개의 예약어와 ANSI 표준 C 언어에 등록되지 않은 11개의 보조 예약어를 갖는다. C 언어로 작성된 프로그램의 호환적 측면을 위해서는 ANSI 표준 C 언어의 예약어만을 사용한다.

5.2 함수를 이용한 DOM 자노드 구성

소프트웨어 아키텍처 노드의 루트는 main() 함수가 된다. main() 함수의 루트 노드 아래에 여러 개의 자노드를 포함한다. 자노드는 소프트웨어의 소스코드의 블록과 제어문 그리고 함수 호출로 구성된다.

(1) 블록을 이용한 자노드 생성

소프트웨어의 소스 코드에는 중괄호("{", "}")로 나타내어지는 블록을 포함하고 있다. 블록은 수행될 코드의 묶음으로서 이를 자노드로 생성한

다. 블록은 서브 함수와 연속으로 수행될 명령 문장 그리고 블록을 내포할 수 있다. 그러므로 블록을 하나의 자노드로 생성한다.

(2) 제어문에 의한 자노드 생성

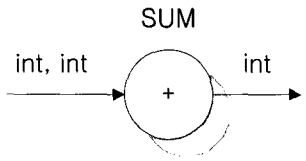
프로그래밍 언어의 제어문에는 조건문, 반복문 그리고 분기문이 있다. 제어문은 주어진 조건에 의해서 프로그래밍된 임무의 진행 순서나 절차가 바뀌어 진다. 동일한 프로그램을 수행하더라도 주어진 조건의 상황에 따라 프로그램의 수행은 달라지게 된다. 그러므로 제어문을 자노드로 생성한다.

(3) 함수의 호출에 의한 자노드 생성

C 언어는 최소한 한 개 이상의 함수로 구성된다. 크게 main() 함수와 서브 함수로 구분되며, 서브 함수를 자노드로 생성한다. 서브 함수는 또한 다른 서브 함수나 자신을 다시 호출할 수 있으며, 이를 자노드로 생성한다. 함수 호출에서 무한 루프와 재귀적인 호출에 대해서는 동일한 함수가 3번 이상 호출이 발생하면 연속적인 자노드를 2개만 생성한다.

5.3 자노드의 연산자와 입출력 자료형 패턴 생성

함수의 코드 중에 연산자가 존재하지 않는 경우에는 자노드를 생성하지 않는다. 연산자를 포함하는 함수만이 입력에 대해 새로운 출력을 생성할 수 있다. 즉, 연산자를 포함하는 함수이어야만 임의의 프로세싱으로 간주하고, 연산자를 포함하지 않는 코드는 단지 화면의 출력이거나, 악의적인 경우의 원본 코드를 속이기 위한 빈 코드의 삽입으로 간주할 수 있다. 그러므로, 이런 코드에 대해 대처할 수 있는 트리의 축약 과정이 필요하다. 모든 노드에는 노드만이 갖는 패턴을 생성하게 된다. 노드에 나타내는 패턴은 입력된 자료형과 수, 연산자 그리고 출력된 자료형과 수로 노드의 패턴을 내부 형식(internal format)으로 생성한다.



(그림 5) sum 노드의 표현

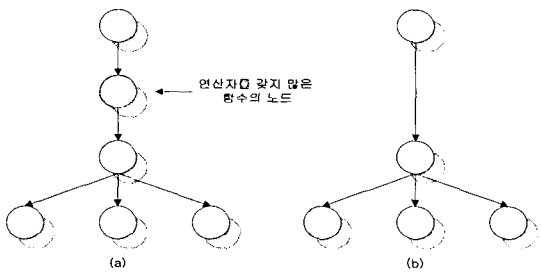
(표 1) sum 노드의 연산자와 입출력 패턴 데이터

```
<Node>
<Node>int</Node><Node>int</Node>
= +
<Node>int</Node>
</Node>
```

노드의 패턴 데이터를 이용하게 되면, 소프트웨어의 아키텍처가 동일하더라도 (그림 5)와 (표 1)과 같이 노드의 패턴 데이터로 프로그램의 기능이 다름을 구분할 수 있게 된다. 즉, sum 노드를 입력 패턴은 두 개의 정수형 데이터, 연산자 패턴은 +, 출력 패턴은 정수형 데이터로 표현되는 노드를 표현함으로 소프트웨어의 아키텍처만을 비교하는 단점을 극복할 수 있게 되며, 프로그램의 구별 능력을 증대시키게 된다.

5.4 빈 노드의 트리 축약

연산자를 포함하지 않은 함수 노드는 DOM 트리의 축소와 실제 소스코드의 분석과정에서 차별화를 위한 어떠한 의미를 제공하지 않는다. 그러므로 연산자 패턴을 포함하지 않은 함수 노드를



(그림 6) 연산자를 갖지 않은 함수 노드의 제거

제거해도 소스코드를 같은 의미로 볼 수 있다. 그러므로 (그림 6)과 같이 연산자 패턴을 갖지 않은 함수 노드를 제거할 수 있으며, 더불어 DOM 트리의 축소를 수행할 수 있다.

5.5 중복 노드 제거와 트리 순회를 이용한 인덱스 패턴 생성

소프트웨어 소스코드의 디지털 라이센스에는 인덱스 패턴을 갖고 있다. 인덱스 패턴에 의해서 디지털 라이센스에 기록된 소프트웨어에 대한 순차패턴과 클러스터링 정보를 제공한다. 만약, 임의의 두 소프트웨어의 인덱스 패턴을 비교해서 동일한 인덱스 패턴으로 매칭되면, 소프트웨어의 아키텍처와 노드의 패턴에 의해서 두 소프트웨어는 정밀한 분석이 이루어진다. 인덱스 패턴의 생성 과정은 DOM 트리의 순회 방법인 중순위(In-order) 방법에 의해서 패턴을 나열하게 되며, 나열된 순차 패턴에서 처음 나타난 다음의 중복된 패턴을 제거하여 인덱스 패턴을 생성하게 된다.

5.6 소프트웨어 버전 관리

인덱스 패턴은 소프트웨어를 구분 및 분류하는 데도 이용되지만, 소프트웨어의 버전(Version) 관리에도 필요한 정보를 제공하는데 사용된다. A라는 소프트웨어가 개발되어 유지보수되면 버전업 과정이 이루어진다. 버전업 과정을 위해서는 최소한 인덱스 패턴은 동일하더라도 소프트웨어 아키텍처나 노드의 패턴이 동일해서는 안된다. 소프트웨어의 버전업은 과거의 소프트웨어 버전과는 많은 부분은 동일하더라도 일부분에서는 노드 패턴의 변경, 노드 위치의 변동, 새로운 노드의 추가와 구 노드의 제거 등의 과정이 나타나게 될 것이다. 이러한 정보에 의해서 소프트웨어의 버전업 관리가 이루어지게 된다.

6. 소프트웨어 소스코드의 분류를 위한 패턴 정보 생성

소프트웨어 소스코드의 디지털 라이센스 정보를 구축하기 위해선 단 하나의 정보를 생성하기 보다는 많은 정보를 제공하기 위한 다양한 패턴 정보를 생성하여야 한다. 소프트웨어 소스코드의 분쟁 발생 시 소프트웨어의 소스코드를 제공하지 않는 대신에 그에 해당하는 충분한 정보를 제공하여야만 디지털 라이센스의 기능을 수행할 수 있기 때문이다.

본 논문에서는 소프트웨어의 소스코드를 이용해서 인덱스 패턴, 소프트웨어 아키텍처, 노드의 패턴 정보 그리고 메시지 축약과 암호화의 다양한 패턴 정보를 생성한다.

6.1 인덱스 패턴

프로그래밍 언어의 예약어에 의한 예약어의 발생 순차 패턴과 중복 제거에 의한 인덱스 패턴 정보를 생성한다. 기본적으로 이 인덱스 패턴에 의해서 프로그램을 구분 및 분류 그리고 비교를 수행한다. C 언어의 예약어는 32개이므로 main() 함수를 루트 노드로 설정한 32!(2.631308369336935301672180121 6e+35)의 가능한 인덱스 패턴이 존재할 수 있다.

6.2 소프트웨어 아키텍처

인덱스 패턴이 동일한 경우에는 어느 정도 유사한 소프트웨어로 간주할 수 있다. 그래서 노드로 구성된 소프트웨어 아키텍처에 의해서 소프트웨어를 분류할 수 있으며 (그림 4)와 같이 DOM을 이용한 추상 코드 아키텍처(ACA: Abstract Code Architecture)를 생성한다. 코드는 컴퓨터에 의해서 수행되어질 프로그램을 구성하는 기본적인 구성 요소에 해당한다. 한편 아키텍처는 코드라는 기본 구성 요소로 이루어진 구조물을 의미한다. 코드는 행위에 제약을 가할 수도 있고 구조

에 영향을 미치기도 한다. 그렇지만 모든 것이 오로지 코드에 의해서만 결정되지 않으며, 아키텍처에 의해서도 분류정보를 제공할 수 있다.

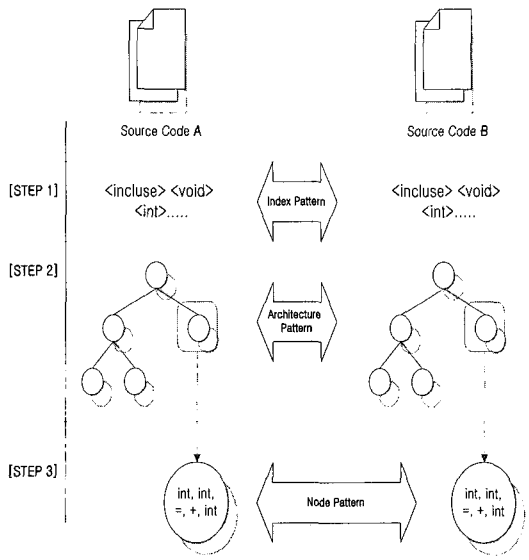
6.3 노드의 패턴 정보

노드의 패턴 정보는 블록이나 함수로 구성된 자 노드에 대해 입력된 자료형, 연산자, 출력되는 자료형 등으로 패턴이 구성된다. 인덱스 패턴과 소프트웨어 아키텍처가 동일한 경우에는 거의 동일한 소프트웨어라고 할 수 있다. 그러나 소프트웨어도 생명주기(Life-Cycle)를 갖기 때문에 과거의 소프트웨어의 패턴을 상속되거나 임의의 변경이 될 수 있다. 이런 경우를 버전업되었다고 하는데, 노드의 패턴 정보에 의해서 이러한 정보를 제공할 수 있다.

6.4 메시지 축약과 암호화

모든 데이터는 메시지 축약(MD: Message Digest)에 의해서 데이터 무결성을 보장받는다. 악의의 행위에 대해 데이터의 무결성을 보장하고, 개인 정보의 유출을 막기 위해서는 임의의 항목들은 암호화가 수행된다. MD의 특성은 임의의 입력된 대용량 데이터 집합을 짧고 축약된 식별자 데이터를 생성한다[18]. 또한 메시지 축약은 일 방향 함수의 특성을 갖으며, 이는 이상적으로 주어진 축약 값을 생성하는 입력이 무엇인지를 계산하는 것이 불가능하다는 것을 의미한다.

인덱스 패턴, 소프트웨어 아키텍처, 노드의 패턴 정보 그리고 메시지 축약과 암호화 등의 다양한 이러한 패턴 정보는 소프트웨어의 소스코드를 분류하거나, 클러스터링할 수 있는 정보는 제공한다. 또한 소프트웨어 버전 관리를 위한 정보도 제공할 수 있다. 소프트웨어 소스코드의 분류 단계는 (그림 7)과 같이 3단계로 구성된다. 분류를 위한 단계를 진행 할수록 좀 더 세밀한 분류가 이루어진다.



(그림 7) 디지털 라이센스에 의한 분류 3단계 과정

[단계 1] 인덱스 패턴 정보에 의한 분류
[단계 2] 소프트웨어 아키텍처 패턴에 의한 분류
[단계 3] 노드의 패턴 정보에 의한 분류

단계 1의 인덱스 패턴은 소프트웨어 소스코드에 사용된 예약어의 종류와 순차 정보로 구성된다. 인덱스 패턴 정보에 의해서 동일한 예약어를 갖는 소프트웨어 소스코드를 크러스터링에 의한 분류가 가능하다. 그러나 인덱스 패턴 정보도 임의의 정도에서는 한계를 드러낼 것이다. 이에 부가적으로 단계 2와 3에 의해서 거시적인 관점과 미시적인 관점의 소프트웨어 소스코드의 분류 기준을 제공한다. 단계 2의 소프트웨어 아키텍처 패턴 정보를 이용하면 거시적인 관점에서 같은 예약어 그룹으로 구성된 소프트웨어 소스코드도 계층구조의 소프트웨어 아키텍처에 의해서 그래픽 형태의 분류 정보를 제공한다. 즉, 소프트웨어의 전체 소스코드에 의한 조감도를 한눈에 확인할 수 있는 기능을 제공한다. 한 단계 더 나아가서, 단계 3에서는 미시적인 관점에서 같은 트리 구조의 소프트웨어의 아키텍처를 갖더라도 아키텍처를 이루는 노드의 패턴 정보인 입력 데이터의 수

와 타입, 연산자의 순차나열 그리고 출력 데이터의 수와 타입에 의해서 더 세밀하게 분류 패턴 정보를 제공할 수 있다.

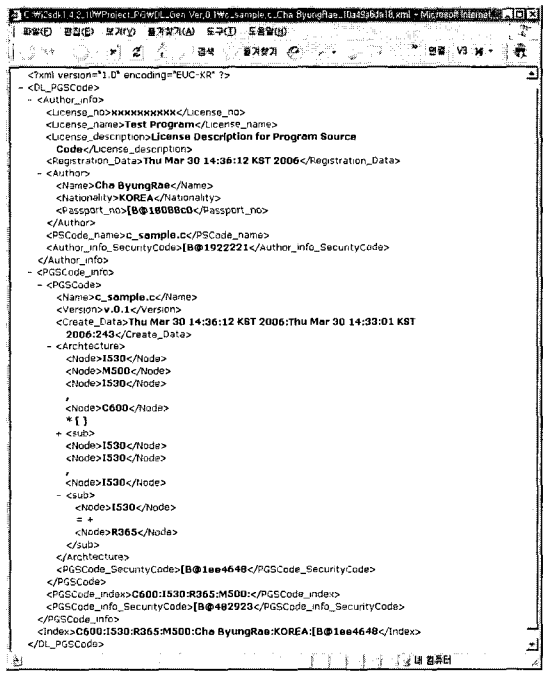
7. XML 기반의 디지털 라이센스 프로토타입 구현 및 보완사항

7.1 XML 기반의 디지털 라이센스 설계

과거에는 모든 정보를 바이너리 형태로 생성하였다. 그러나 최근에는 암호화 기술 및 통신의 발달은 바이너리 형식보다는 가독성이 높고 메타데이터를 기술할 수 있는 XML 형식으로 대부분 기술한다. 향후의 개선된 스키마로 기술되더라도 쉽게 변환이 가능하다는 장점을 갖는다. 소프트웨어 소스 코드의 디지털 라이센스는 XML 기반의 소프트웨어 소스 코드에 대한 소프트웨어 작성자, 프로그래밍 언어, 등록 날짜, 등록번호, 버전 번호, 소프트웨어 소스 코드의 DOM 정보 등이 기술되어 있다. 소프트웨어 소스 코드의 디지털 라이센스의 무결성과 비밀성을 보장하기 위해서 XML로 만들어진 디지털 라이센스 코드의 일부분을 암호화를 수행한다. 본 연구에서는 JCE[19]의 패키지를 이용하여 메시지 축약을 수행하며 (그림 8)과 같이 나타난다.

7.2 디지털 라이센스의 색인기, 검색기 그리고 디지털 라이센스의 비교

소프트웨어의 소스코드는 특별히 데이터베이스에 관리하지는 않으며, IDE 환경의 프로젝트 디렉터리에 보관하는 게 대부분이다. 유능한 개발자들은 CVS[20] 등을 이용하여 소스 코드를 관리하지만 이것 또한 디렉터리에 관리하게 된다. 그래서 향후 소스 코드와 디지털 라이센스는 같이 디렉터리에 존재하는 경우가 많이 발생할 것이다. 디지털 라이센스를 관리하기 위해서는 디렉터리를 탐색하여 디지털 라이센스의 색인하는 색인기

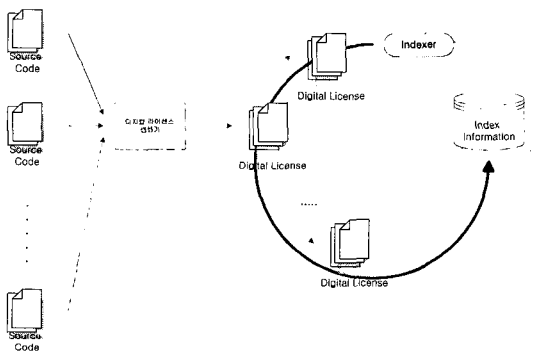


(그림 8) 소프트웨어 소스 코드의 디지털 라이센스 프로토타이핑

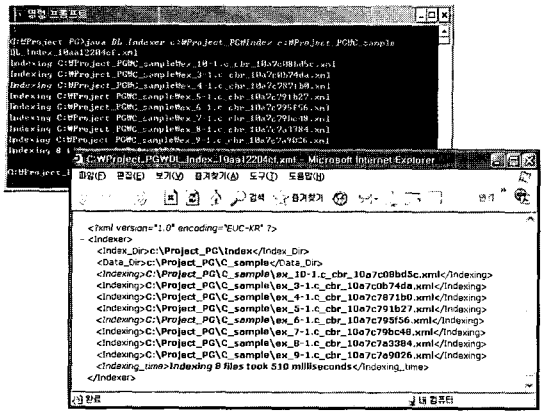
와 역파일로 색인된 색인 파일을 읽어서 텀(term)에 의해 검색하는 검색기를 설계((그림 9)과 (그림 11)) 및 구현하였다.

(1) 디지털 라이센스의 색인기

(그림 9)는 작업 디렉토리에 작성된 소스코드를 시스템에서 검색하여 소스코드에 대한 디지털



(그림 9) 디지털 라이센스의 색인 생성기의 개괄도



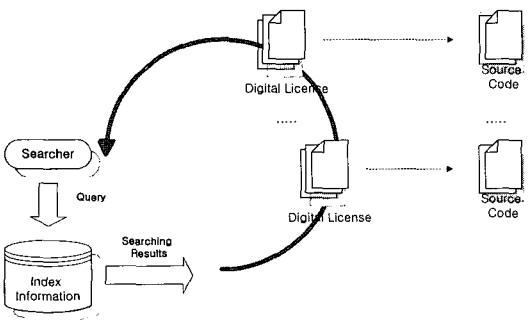
(그림 10) 디지털 라이센스의 검색기 시뮬레이션 결과

라이센스를 생성하고, 생성된 디지털 라이센스에 대한 색인 정보를 생성하는 과정을 나타낸다.

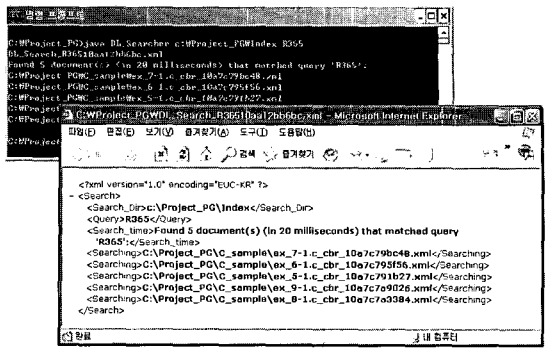
소프트웨어 소스코드를 대처하기 위한 디지털 라이센스를 이용하기 위해서는 먼저, 디지털 라이센스를 파일 시스템에서 검색하고 디지털 라이센스의 정보를 이용한 역파일 색인을 생성한다. 생성된 역파일은 검색기에 의해서 디지털 라이센스의 정보 검색 및 위치를 찾는데 정보로 활용하며 결과는 (그림 10)와 같다.

(2) 디지털 라이센스의 검색기

(그림 11)은 생성된 디지털 라이센스의 색인 정보를 이용하여 검색과정에서 검색된 디지털 라이센스와 이에 해당하는 소프트웨어 소스코드를 매핑하는 것을 나타낸다.



(그림 11) 디지털 라이센스의 검색기의 개괄도



(그림 12) 디지털 라이센스의 색인기 시뮬레이션 결과

사전에 생성한 디지털 라이센스의 색인 정보를 이용하여 검색어에 의한 일치되는 디지털 라이센스의 위치와 디지털 라이센스 파일 이름 정보를 생성하며 결과는 (그림 12)와 같다.

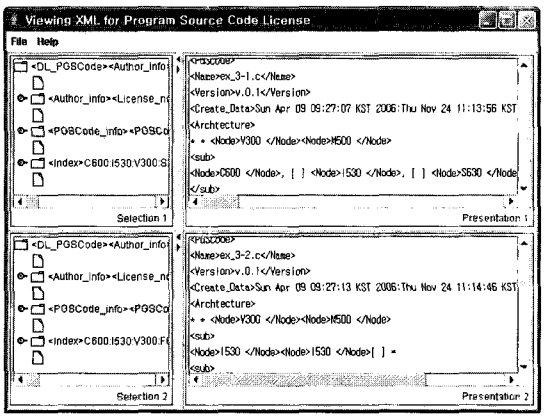
(3) 디지털 라이센스에 의한 소프트웨어 소스코드의 비교

먼저, 임의의 A와 B라는 두 개의 C 언어의 소스 코드를 이용하여 A와 B의 디지털 라이센스를 발행하였다. (그림 13)은 A와 B의 소스 코드를 이용하여 발행한 디지털 라이센스를 자체 개발한 XML TreeView에 의해서 나타낸 것이다. A와 B의 두 소스 코드는 거의 같은 트리 구조와 아키텍처 패턴을 가졌으나, 단말 노드 부분에서 두 소스 코드의 디지털 라이센스의 아키텍처가 다르게 표현되었으며, 더불어 노드의 패턴 정보도 다르게 표현되었다.

분쟁 해결을 위해서는 소소 코드의 패턴 매칭에 의한 방법도 있지만, 트리 구조로 표현된 소스 코드의 아키텍처를 보면 전반적인 소프트웨어 소스코드의 조감도를 비교할 수 있는 장점을 갖으며, 분쟁 해결에 도움이 될 것이다.

7.3 보완 사항

소프트웨어 소스코드가 상당히 길고, 복잡한



(그림 13) 두 개의 디지털 라이센스에 의한 소스 코드의 트리 구조 형태, 아키텍처 패턴과 노드의 패턴 정보 비교

특성을 갖고 있다. 또한. 서로 다른 소프트웨어가 유사한 제어 구조로 작동된다면, 즉 유사한 알고리즘으로 구성되어 작동되면서도 약간 또는 일부만을 수정하여 작성한 경우 이에 대한 소유권 검증 과정에서 유일성을 확인하기 어려울 것으로 판단된다. 또한 유일성을 명확히 판단하기 어려운 경우도 발생할 수 있을 것이다. 그러므로 유일성을 제공할 수 있는 패턴 생성 분야에 연구가 더 필요하다고 생각한다. 더불어, 소스코드와 디지털 라이센스의 파일 용량간의 오버헤드 문제와 디지털 라이센스를 그래픽으로 표현하기 위한 연구가 필요하다. 추가하여, 소프트웨어에 대한 보호 및 저작권 정보를 좀 더 명확하게 기술할 수 있는 향후 연구가 필요하다.

8. 결 론

과거 산업사회의 태동과 발전과정에서 뿐만 아니라, 21세기 정보화 사회의 발전에 있어서도 이러한 지적재산권 제도는 중요한 역할을 하고 있다. 우리나라의 국가 경쟁력 제고를 위해서도 디지털콘텐츠에서 확대된 소프트웨어 소스 코드에 대한 지적재산권 제도와 기술의 정비는 매우 중요한 의미를 지닌다. 디지털 저작물의 원천 요소인 소프트웨어 소스 코드에 대해서는 암호화이외에는 별다른 방법이 존재하지 않으며, 아직은 연구 초기 단계이다. 암호화 방법 역시 소프트웨어 소스 코드 개발자와 소프트웨어 소스 코드 구입자간의 1차적인 거래에만 저작권 관리를 지원할 뿐 그 이후에 발생하는 거래에 대한 저작권을 보호주지 못하고 있다. 소프트웨어 소스코드의 소유권 분쟁이 발생 시 소유권을 증명하기 위해서는 원본의 소프트웨어 소스코드를 공개해야만 하는 문제점을 갖고 있다. 본 논문에서는 소프트웨어 소스코드의 원본 공개로 인한 저작권 침해와 기술 유출을 막기 위한 소프트웨어 소스코드의 디지털 라이센스의 설계를 제안하였다. 분쟁 해결에 소소 코드의 패턴 매칭 방법과 트리 구조로 표현된 소스코드의 아키텍처와 노드 정보를 이용하여 거시적인 소프트웨어 소스코드의 조감도를 비교할 수 있으므로 분쟁 해결에 도움이 될 것이다. 향후 연구로는 구조적 프로그래밍 언어와 객체지향 프로그래밍 언어의 구분을 통합 및 소프트웨어의 소스 코드에 대한 정보를 포함 가능한 다양한 패턴 정보의 생성과 생성된 패턴 정보의 검색을 위한 색인기 및 검색기에 대한 연구가 필요하다.

참 고 문 헌

[1] http://www.doi.org
[2] http://www.loc.gov/marc/
[3] http://dublincore.org/
[4] http://dublincore.org/
[5] http://www.editeur.org/onix.html
[6] http://www.xrml.org
[7] http://odrl.net/
[8] Matthias Rieger, Stephane Ducasse, "Visual Detection of Duplicated Code", Proceedings ECOOP Workshop on Experiences in Object-Oriented Re-Engineering, 1988.
[9] Wuu Yang, "Identifying Syntactic Differences Between Two Programs", Software-Practice and

Experience, Vol. 21(7), 739-755. July 1991.
[10] http://www.few.vu.nl/~dick/sim.html
[11] http://glimpse.arizona.edu/javadup.html
[12] http://www.ccsr.cam.ac.uk/~mw263/YAP.html
[13] http://www.cs.berkeley.edu/~aikem/moss.html
[14] 조동욱, "디지털 재산권 보호를 위한 S/W 프로그램 표절 감정 기술과 툴의 분석", 한국콘텐츠학회 2003 춘계종합학술대회 논문집, p.177-184, 2003.
[15] http://www.pdmc.or.kr/
[16] http://www.w3.org/
[17] Hiroshi Matuyama, Kent Tamura and Naohiko Uramoto, "XML and Java Developing Web Applications", Addision Wesley, 1999.
[18] Jonathan Knudsen, "Java Cryptography", O'Reilly, 1998.
[19] Jess Garms and Daniel Somerfield, "Professional Java Security", Wrox, 2001.
[20] 데이비드 토머스, 실용주의 프로그래머를 위한 버전관리 using CVS, 인사이트, 2005.

◐ 저 자 소 개 ◑

차 병 래
1995년 호남대학교 수학과 졸업(학사)
1997년 호남대학교 대학원 컴퓨터공학과 졸업(석사)
2004년 목포대학교 대학원 컴퓨터공학과 졸업(박사)
2005년~현재 호남대학 컴퓨터공학과 교수
관심분야 : 정보보호, DRM, 데이터마이닝, 신경망
E-mail : chabr@honam.ac.kr

정 종 근
1995년 조선대학교 전자계산학과 졸업(학사)
1997년 조선대학교 대학원 전자계산학과 졸업(석사)
2002년 조선대학교 대학원 전자계산학과 졸업(박사)
2004년~현재 호남대학교 컴퓨터공학과 초빙교수
관심분야 : 인공지능, 검색엔진, 데이터베이스, 정보보안, 전자상거래, 바이러스, 유비쿼터스
E-mail : jkjeong@honam.ac.kr

오 수 열
1981년 전남대학교 재료공학과 졸업(학사)
1986년 조선대학교 대학원 전자계산학과 졸업(석사)
2001년 전남대학교 대학원 전산통계학과 수료(박사)
1988년~현재 목포대학 정보공학부 교수
관심분야 : 웹서비스, 유비쿼터스컴퓨팅, 소프트웨어 공학
E-mail : syoh@mokpo.ac.kr