SW복제도 감정을 위한 유사성 탐지도구의 설계 및 구현

방효근*, 차태원*
*컴퓨터프로그램보호위원회
revoice@socop.or.kr · twcha@socop.or.kr

Design and Implementation of the Detection Tool for Calculating the Similarity Degree between Two Computer Programs

Hyo-Keun Bahng*, Tea-Won Cha*
*Korea Software Copyright Committee

요 약

디지털 시대의 도래와 함께 국내외적으로 SW 및 디지털 콘텐츠로 확대되고 있는 표절과 불법복제 문제의 심각성은 날로 더해가고 있으며, 이에 따른 사회·경제적인 폐해 규모도 급격히 증가하고 있다. 따라서 SW표절과 불법복제로부터 저작권 보호를 위한 적극적 대응 방안으로 SW복제도 감정에 적합하고 유용한 SW시스템 개발의 필요성을 인식하게 되었다. 본 논문에서는 SW복제도 감정, 즉 두 프로그램 사이의 동일·유사성 정도를 판단하기 위해 제안된 유사성 탐지도구의 핵심 설계구조 및 기반 기술 등 전반적인 구현 메커니즘에 관하여 논한다.

1. 서론

컴퓨터 및 SW산업과 인터넷의 발전은 수많은 종류의 디지털 콘텐츠를 양산화 시키며 첨단 지식 정보화 시대의 도래를 앞당기는 촉진제 역할을 수행하였다. 반면에 컴퓨터를 이용한 다양한 매체들이 디지털화 됨에 따라 누구나 쉽게 디지털 콘텐츠의 불법복제 및 표절을 행하는 등 다양한 부작용을 초래하였다. 특히 대학가에서의 표절 행위는 심각한 문제로 대두된 지 오래이다. 대다수의 학생들이 인터넷 서핑을 통해 단순한 '복사하기(copy) & 붙여넣기(paste)'와 같은 기능을 이용하여 손쉽게 보고서 및 연구논문 등을 불법 복제한다는 것은 이미 누구나 인정하는 공공연한 사실이다. 비단 이러한 문제는 학계에서의 보고서 표절에만 국한된 것이 아니다. 즉, 많은 기업 활동을 포함한 다양한 수요자 계층에서 컴퓨터 소프트웨어(SW)에 대한 권리보호 인식부족과 함께 손쉽고 은밀하게 복제할 수 있다는 특성을 이용하여 무의식적으로 불법복제[1]를 자행하고 무분별하게 사용하고 있다. 더욱이 SW의 창작적 개발과 이에 따른 지적재산권(Intellectual Property Rights)에 대한 낮은 인식 수준으로 인하여 많은 법적분쟁을 일으키고 있는 것 또한 현실이다.

따라서 국내에서는 SW산업의 지속적 발전과 더불어 다양한 유형의 SW관련 분쟁이 증가하고 있는 만큼, SW감정[2]이 분쟁의 공정한 해결수단으로서 그 비중과 적용영역은 점차 확대되고 있다.[1] 또한, 다양한 유형의 SW감정의뢰 건수와 SW규모 및 복잡도의 증가로 인해 프로그램 소스코드를 감정인의 수작업만으로 감정할 경우 대량의 시간과 노력이 소요될 뿐만 아니라 감정결과에 대한 신뢰성에도 의문이 제기될 소지를 떠안게 된다. 이에 따라 SW감정의 객관성 및 신뢰성을 유지하고, 계획된 수행기간 내에 효과적으로 대응하기 위하여 자동화된 SW감정도구의 필요성이 대두되었다.

이러한 점에서 SW저작권 보호와 관련된 다양한 분쟁해결의 기술적인 주요 수단으로서 SW감정도구의 역할 및 중요성은 점차 커지고 있다. 또한 SW감정도구는 감정인으로 하여금 과학적인 감정수행을 통해서 밝혀낸 정확한 사실과 정량적인 분석결과를 바탕으로 감정의견을 객관화하는데 활용하는 필수적인 수단임과 동시에 감정업무 수행에 필요한 인프라로서 인식되고 있다.

본 논문에서는 SW복제도 감정, 즉 두 프로그램 사이의 동일·유사성 정도를 판단하기 위해 제안된 유사성 탐지도구의 핵심 설계구조 및 기반 기술 등 전반적인 구현 메커니즘에 관하여 논한다. 본 논문의 구성은 다음과 같다. 2장에서는 프로그램 소스코드의 표절 탐지 및 분석 방법을 살펴본다. 3장은 핵심 모듈의 설계구조 및 기능을 제

1) 특정 저작물을 동일하게 부착하여 복제물을 작성(i.e. 특정프로그램을 디스켓이나 CD-ROM 등에 원형과 동일하게 기계적으로 옮겨서 수록한 경우)하는 협의의 복제(Dead-Copy)와 다른 표현양식을 이용하여 제 3자가 감지할 수 없도록 다소의 변경, 수정, 증감을 행하여 다시 작성하는 행위(i.e. 프로그램의 언어변경, 기록매체의 변경, 프로그램 중 일부의 변경 또는 증감 등의 행위)로 개변시키되 전혀 작성자의 학술적 사상을 창작적으로 표현하지 아니하는 광의의 복제가 급격히 증가하고 있음

2) 정부는 SW감정제도를 '컴퓨터프로그램보호법 제38조의2'에 규정하고, 컴퓨터프로그램보호위원회(SOCOP)에서 시행하고 있음

시한다. 4장에서는 ALC 및 MbT 비교분석 알고리즘과 GUI를 기반으로 구현된 유사성 탐지도구를 소개한다. 마지막으로 5장에서는 결론 및 향후 연구방향에 대하여 언급한다.

2. 관련연구

2.1 프로그램 소스코드의 표절

프로그램 소스코드에 있어서 표절의 정의는 Parker & Hamblen[2] 에 의해 제안된 "A program which has been produced from another program with a small number of routine transformation." 이라는 정의가 보편적으로 사용된다. 정의 내의 "transformation" 이란 단순한 요소(i.e. 필요 없는 주석(Comments)의 삽입 및 삭제, 변수(Variable)의 형(Type) 및 이름 변경 등)로부터 복잡한 요소(i.e.루틴(routine) 및 모듈(module), 라이브러리(library)의 변경) 들의 변환을 의미한다. 즉, 프로그램 소스 코드의 표절은 원본 소스 코드를 논리적인 흐름에 영향을 주지 않고 단순히 구조를 변경하거나 식별자의 이름, 데이터 타입(Data type)의 변경, 함수의 호출 변경 등을 포함한다. 다음은 프로그램 소스 코드의 표절 방법의 실례이다. [3]

- 필요 없는 주석(Comments)의 삽입 및 삭제
- 변수(Variable)의 이름 및 형 변경
- 프로그램에 필요하지 않은 코드의 삽입
- 함수의 위치 및 호출 변경
- 부분적인 소스 코드의 변경
- 기본 루틴은 동일하나 특정 라이브러리(Library)의 삽입
- 여러 프로그램 소스로부터 코드들을 조합하여 새로운 코드 생성
- 특정 모듈 혹은 함수를 몇 개로 나누거나 또는 반대로 하나의 모듈로 병합하여 새로운 코드 생성
- 결과물의 동일 및 유사

2.2 프로그램 소스코드의 표절 탐지 및 분석

2.2.1 지문(Fingerprint)을 이용한 탐지 방법

일반 문서에서 표절 유무를 탐지하기 위해서 사용되는 방법으로 원본과 사본 사이에서 사용된 단어들의 유사성 및 단어의 빈도수 등을 비교 분석한다. 또한, 문서의 길이에 영향을 받지 않고 가장 쉽게 지문을 얻기 위해 통계적인 방법을 사용한다. 즉, 문서 내에서 사용된 단어의 개수, 빈도수, 특정 주제어의 위치 및 사용 회수 등을 조사하여 탐지에 적용한다. 이 방법은 물론, 소프트웨어 프로그램 소스코드의 표절 탐지에 적용하여 토큰 비교 분석을 통해 유사도를 평가할 수 있으나, 코드의 구조(Structure) 및 실행 루틴(Routine)을 파악하기 곤란하다.

2.2.2 구조 기반(Structure-related)의 탐지 방법

일반 문서와는 달리 프로그램의 소스 코드는 특정한 문법 구조와 제어 흐름의 규칙이 사전에 정의 되어 있기 때문에 구조적인 특징을 내포하고 있다. 따라서 엄격한 프로그래밍 문법의 적용으로 표현되는 구조적 특성을 파악한 후, 서로 밀접한 관계를 형성하는 부분을 탐지함으로써, 속성 계수[4] 방법론의 취약점을 보완하고, 보다 향상된 프로그램 소스코드 표절 탐지를 위해 제안되었다. 가장 최근의 표절 탐지 도구에 적용된 구조 매트릭스는 프로그램 구조의 스트링 표현을 비교하여 유사도를 측정한다. 그리고 이 매트릭스를 사용한 도구들은 연속되는 토큰으로 구성되는 스트링의 유사성을 평가하므로, 정확한 매치(exact match)를 요구하지 않는다.[5, 6]

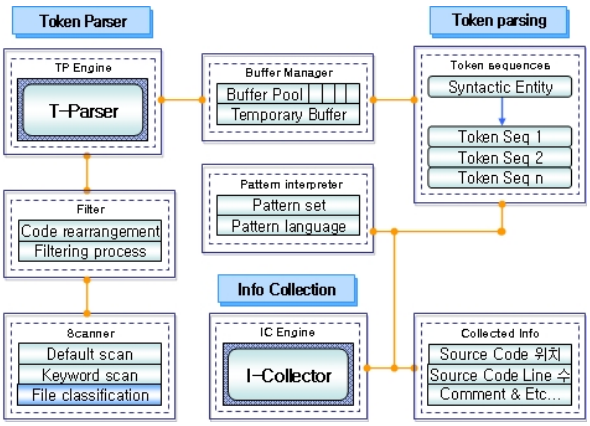
대표적인 표절 탐지 시스템으로는 YAP3[7], MOSS[8], JPlag[9], Plague[10] 등이 있으며, 이중 MOSS와 JPlag는 온라인 서비스로 제공된다. MOSS는 C, C++, Java, Pascal, Ada, ML, Lisp, Scheme programs 등의 다양한 프로그래밍 언어를 지원하며, JPlag는 Java로 작성된 소스코드를 비교하여 시각적으로 그 결과를 표현해 준다.

3. 유사성 탐지도구의 구성모듈 설계

유사성 탐지도구는 크게 언어 선택기(Language Selector), 기본 스캐너(Simple Scanner), 어휘 분석기(Lexical Analyser), 패턴 인터프리터(Pattern Interpreter), 비교분석 엔진(Comparison & Analysis Engine), 보고서 생성기(Report Generator) 등 7부분의 주요 모듈로 이루어진다. 본 장에서는 유사성 탐지도구를 구성하고 있는 핵심 모듈의 구조도와 그 기능을 간단히 제시한다.

3.1 어휘 분석기

다음 (그림 1)은 어휘 분석기의 구조도를 보여준다.

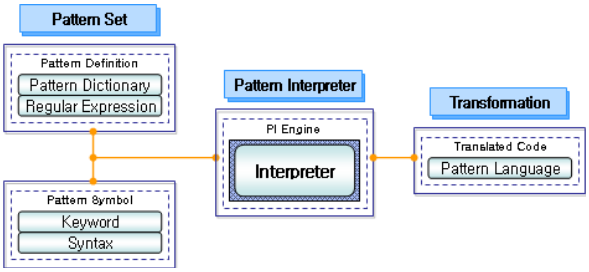


(그림 1) 어휘 분석기의 구조도

원본과 비교본 소스파일을 토큰(token)으로 분류하는 어휘 분석을 수행한다. 그리고 어휘 분석이 진행되는 동시에 각 토큰들의 기본속성 정보(ex. keyword, identifier, constant, special character 등), 변수정보, 함수정보 등이 등록되어 지며, 패턴 비교(pattern matching)를 위한 패턴 변환 정보 역시 생성된다.

3.2 패턴 인터프리터

다음 (그림 2)는 패턴 인터프리터의 구조도를 보여준다.

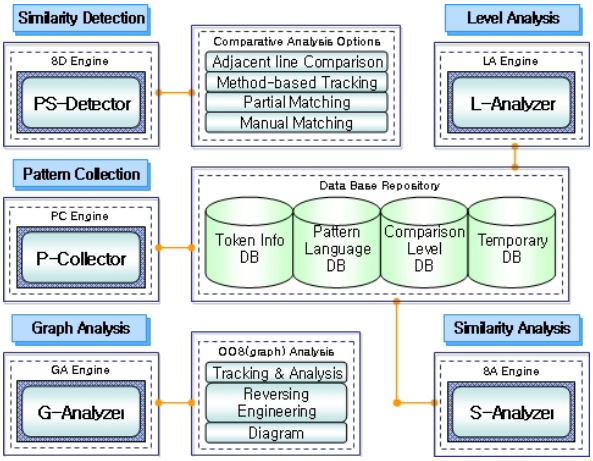


(그림 2) 패턴 인터프리터의 구조도

사전에 정의된 패턴 심볼(정규표현)을 사용하여 소스코드를 구조화된 패턴 언어(Pattern Language)로 변환하는 역할을 수행한다. 구조적이고 추상적인 패턴 언어를 대상으로 하는 패턴 매칭은 유사성 비교분석의 효율을 극대화할 수 있으며, 또한 변수 및 함수명 변경, 데이터 타입 변경, 특정 소스코드 블록의 위치 변경 등과 같은 개작 및 복제 유형을 쉽게 탐지할 수 있는 장점을 가지고 있다.

3.3 비교분석 엔진

다음 (그림 3)은 비교분석 엔진의 구조도를 보여준다.



(그림 3) 비교분석 엔진의 구조도

두 프로그램간의 물리적·논리적 비교분석을 수행한다. 물리적 비교분석은 소스코드의 유사도를 정량적으로 측정하는 것을 의미하고, 반면에 논리적 비교분석은 프로그램이 실행되는 과정에서 호출되는 함수와 명령문 실행에 따른 제어흐름관계, 자료구조 및 기능상의 유사성을 포함한 알고리즘 측면에 대한 분석을 말한다.

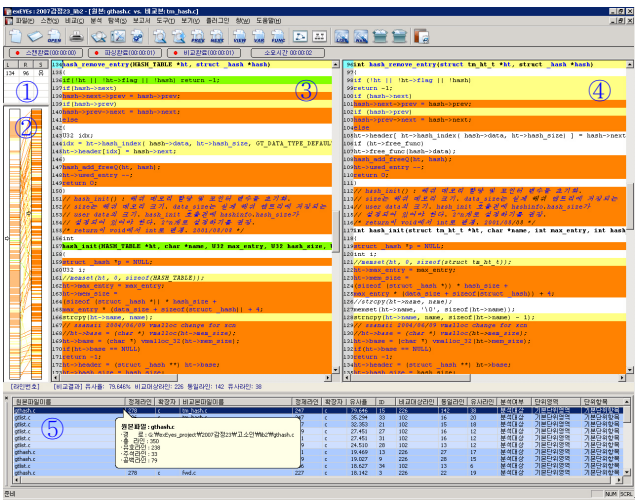
소스코드의 유사성 비교에는 GST를 기반으로 한 인접라인비교(ALC: Adjacent Line Comparison)알고리즘과 함수기반추적(MbT: Method-based Tracking) 알고리즘을 적용하였다. 전자는 파일 단위로 원본과 비교본의 토큰들을 서로 비교한 후, 소스코드의 각 라인별 동일 또는 유사여부를 판단한다. 그리고 전체 유효라인수 대비 동일·유사 라인수의 비율을 측정하여 라인수 기반의 유사도를 산정한다. 즉, 여러 개의 동일·유사 라인이 소스코드에 분포해 있는 경우를 고려하여 가장 커다란 동일·유사 블록을 구성하는 라인을 탐지하는 방법이다. 후자는 전체 프로그램을 구성하는 함수(메소드) 단위의 비교를 수행하여 동일 또는 유사 여부를 판단하고 각 함수별 유사도를 산정한다. 즉, 함수 호출 흐름을 추적해가며 동일 또는 유사함수를 탐지하는 방법이다.

4. 유사성 탐지도구의 구현

본 장에서는 ALC 및 MbT 비교분석 알고리즘과 GUI를 기반으로 구현된 유사성 탐지도구를 소개한다.

유사성 탐지도구는 소스코드를 비교하는 주요기능 외에 함수 호출 그래프 보기, 클래스 다이어그램 보기, 변수 및 함수 리스트 보기 등과 같은 다양한 기능을 가지고 있어 두 프로그램간의 실질적 유사성을 판단하는데 유용하다. 각각의 파일을 비교하기 위하여 유사성 탐지도구는 우선 비교대상 소스코드 파일에서 의미 없는 공백을 제거하고[3], 소스코드를 라인단위로 병합 또는 분리한 후 토큰으로 파싱 한다. 그리고 토큰의 패턴정보를 이용하여 동일·유사여부를 비교할 수 있기 때문에 함수명이나 변수명을 단순히 바꾼 경우에도 유사한 코드 검출이 가능하다. 또한 모든 소스코드에 대한 비교쌍을 생성하여 비교하므로 파일명이 다르거나 일부 소스코드가 서로 다른 파일에 존재하는 경우에도 정확히 탐지할 수 있다.

다음 (그림 4)는 두 프로그램간의 유사도 측정을 위한 비교·분석 결과를 보여준다.



(그림 4) 비교·분석 결과

비교·분석 결과 화면은 크게 5부분으로 구성되어 지며, 각각의 사용자 인터페이스 기능은 다음과 같다.

3) 유사성 탐지도구는 감정대상 프로그램에 포함되어있는 소스파일을 스캐닝(예비검사)하는 과정에서 소스코드의 라인수를 전체라인수, 공백라인수, 주석라인수, 유효코드라인수 등으로 구분하여 실질적인 비교대상라인을 산정하는 기능을 가지고 있음

① 동일・유사라인 목록 화면
- 원본과 비교본 파일의 동일 및 유사 라인 정보를 리스트로 관리
- 'L' 컬럼은 원본에서 선택되어진 라인 번호를 의미하고, 'R' 컬럼은 비교본에서 동일 및 유사로 판단된 라인 번호를 의미하며, 'S' 컬럼은 '동일' 또는 '유사'를 구분하여 표시

② 원본(Source)과 비교본(Target) 파일의 비교・분석 결과 그래프
- 원본과 비교본 파일의 비교・분석 결과를 각각의 막대 그래프로 표현하는 퀵 뷰어(Quick viewer) 모드를 제공
- 즉, 동일하거나 유사한 것으로 나타난 라인을 각기 다른 색으로 표시하고, 서로 매칭(matching) 되는 부분을 실선으로 연결하여 직관적인 위치파악을 할 수 있도록 함
- 또한, 마우스의 포인터를 사용하여 막대 그래프의 특정 위치를 클릭하면, 하단에 현재의 라인 번호를 표시하고, "원본(Source) 파일의 비교・분석 결과 화면"과 "비교본(Target) 파일의 비교・분석 결과 화면"을 동시에 자동으로 스크롤 시킴으로서 선택된 라인 번호의 위치로 이동

③ 원본(Source) 파일의 비교・분석 결과 화면
- 비교본 파일과 서로 동일하거나 유사한 것으로 매칭되는 라인을 각기 다른 색으로 표시(주황색 : 동일라인, 노랑색 : 유사라인, 연두색 : 사용자가 변경한 유사라인)
- 마우스의 포인터를 사용하여 동일하거나 유사한 것으로 나타난 라인을 클릭하면, 선택된 라인번호를 마킹하고, "비교본(Target) 파일의 비교・분석 결과 화면"을 동시에 자동으로 스크롤 시킴으로서 원본과 매칭되는 라인 번호의 위치로 이동

④ 비교본(Target) 파일의 비교・분석 결과 화면
- 원본 파일과 서로 동일하거나 유사한 것으로 매칭되는 라인을 각기 다른 색으로 표시
- "원본(Source) 파일의 비교・분석 결과 화면"에서 선택된 라인에 매칭되는 라인 번호를 마킹

⑤ 원본(Source)과 비교본(Target) 파일의 비교쌍 목록 화면
- 스캐닝 단계를 통해 비교대상폴더로부터 얻어진 파일의 비교쌍을 기준으로 동일라인, 유사라인 등의 정보와 함께 유사율 측정 결과[4]를 리스트로 관리
- 마우스의 포인터를 사용하여 하나의 비교쌍 행(row)을 기준으로 클릭하면, 선택되어진 비교쌍의 "원본(Source) 파일의 비교・분석 결과 화면"과 "비교본(Target) 파일의 비교・분석 결과 화면" 등이 자동으로 새롭게 생성

5. 결론 및 향후 연구방향

본 논문에서는 기존 표절탐지도구의 취약점과 문제점을 개선하여 좀더 SW복제도 감정에 적합하고 효과적으로 적용하기 위해 제안된 유사성 탐지도구를 소개하였다. 소스코드 기반의 물리적 비교・분석에는 인접라인비교(ALC)알고리즘을 적용하였다. 본 알고리즘의 기본 개념은, "프로그램의 소스 코드는 특정한 문법 구조와 제어 흐름 규칙이 사전에 정의 되어 있기 때문에 구조적인 특징을 내포하고 있다"라는 점에서 출발하였다. 따라서 구조기반(Structure-related)의 탐지 방법을 적용하여 서로 밀접한 관계를 형성하는 부분을 탐지함으로써 비교결과의 정확성을 높일 수 있었다. 현재 유사성 탐지도구는 다양한 프로그램 언어(C, C++, C#, Visual C++, Java, Visual Basic, Delphi, Power Builder, PHP, JSP, ASP)와 Text 문서의 동일・유사 여부 검출기능을 제공한다.

향후 연구방향은 프로그램 언어별 문법 구조와 제어 흐름을 좀더 정밀히 분석하고 알고리즘 개선을 통한 유사성 탐지의 정확성과 성능향상에 초점이 맞추어질 것이다.

4) 원본(Source) 파일의 비교・분석 결과 화면에서 사용자가 특정라인을 선택하여 유사라인으로 변경(육안식별)하면 유사율 역시 자동으로 재산출 되어짐

참고문헌

[1] 방효근, "소프트웨어 복제도 감정기법의 표준화 모델에 관한 연구", 한국정보처리학회지 제13권 제6호, pp.823-832, 2006.10.

[2] A. Parker and J. Hamblen, "Computer algorithms for Plagiarism Detection", IEEE Transactions on Education, Vol. 32, No. 2, May 1989.

[3] 조환규, "Genomic Sequence Aligment and its application fo Computing Linear Structure Similarity", 2002년 제 1차 한국 생물정보학회

[4] M. H. Halstead, "Elements of Software Science", North Holland, New York, 1977.

[5] 방효근, "소프트웨어 복제 감정을 위한 유사도 및 복제도 평가방법에 관한 연구", 한국소프트웨어감정평가학회 제2회 추계 학술발표대회 논문집, pp.9-15, 2003. 11.

[6] 방효근, "Chain hashing 기법을 적용한 소프트웨어 감정관리 시스템의 설계 및 구현에 관한 연구", 한국소프트웨어감정평가학회 춘계 학술발표대회 논문집, pp.49-55, 2003. 6.

[7] J. Wise, "YAP3: improved detection of similarities in computer programs and other texts", In proceedings of SIGCSE'96, pp. 13-134, 1996.

[8] http://ftp.cs.berkeley.edu/~aiken/moss.html

[9] http://wwwipd.ira.uka.de:2222/

[10] G. Whale. "Plague: plagiarism detection using program structure". Dept. of Computer Science TechnicalReport 8805, University of NSW, Kensington, Australia, 1988.