프로그램 표절 감정 툴에 대한 비교,분석 및 개발 툴에 대한 방향제시

조동욱*, 소정**, 김진용***, 최병갑****, 김선영*****,
김지영******
*충북과학대 **한국전자통신연구원
혜천대학 ****목원대학교 *****충북대학교***서원대학교
e-mail:ducho@ctech.ac.kr

A Comparison and Analysis of Program Plagiarism Inspection Tools & Proposal of Developing Tools

Dong Uk Cho*, Jung Soh**, Jin Yong Kim***,
Byung Kap Choi****, Sun Young Kim*****,
Ji Yeong Kim******
*Chungbuk Provincial University of Science & Technology
**Electronics and Telecommunications Research Institute
***Hyecheon College, ****Mokwon University
*****Chungbuk University,******Seowon University

요 약

컴퓨터 소프트웨어, 디지털 콘텐츠등 디지털 정보 재산권의 보호는 현재 뿐 아니라 향후 국가의 국력을 좌우할 수 있을 정도로 대단히 중요한 과제가 아닐 수 없다. 본 논문에서는 소프트웨어 표절의 자동검출을 위하여 제작된 각종 소프트웨어 감정 도구들을 여러 각도에서 비교·분석하여 각 도구의 유용성, 제한성, 주요 적용환경 및 분야, 사용방법 등을 제시함으로써 향후 소프트웨어 복제 감정에 효과적으로 활용할 수 있도록 하고자 한다.

1. 서론

최근 인터넷의 급속한 발전과 더불어 소프트웨어분야도 많은 발전을 가져왔다. 그러나 이러한 소프트웨어는 무분별한 불법 복제 및 도용으로 소프트웨어발전에 저해를 가져오게 되었다. 이러한 이유를 근절하고자 많은 소프트웨어 표절을 검출하는 감정 도구들은 인터넷상에 배포 및 요청에 의하여 구할 수 있게 되었다. 그러나 프로그램 감정 툴들을 실제 프로그램 감정 업무에 직접 활용하려면, 각 도구의 설치 환경 및 방법, 사용법, 유용한 분야, 제한점, 기대할 수 있는 결과, 결과의 활용 방안 등을 자세히 파악하여야 한다. 프로그램 감정을 수행할 때 대부분의 감정인들은 새로운 도구를 처음부터 파악하여 사용하는데 추가적으로 시간을 할애할 수 없기 때문에, 자신에게 익숙한 1-2개 정도의 도구를 반복적으로 사용하는 경향이 뚜렷하다. 이로 인하여 각 감정 사안의 특이성에 비추어 가장 적절하고 효과적인 도구가 존재 하더라도 이를 활용하지 못하고 있는 상황이다.

본 연구는 이러한 상황을 타개하기 위하여 기획된 것으로, 현재 일반적으로 쓰이고 있는 감정 도구들을 상호 비교하고 분석하는 것을 주요 연구 내용으로 한다. 연구 결과는 감정인들이 쉽게 자신의 감정 업무에 적합한 소프트웨어 감정 도구를 선택하고, 또한 선택한 도구로부터 기대할 수 있는 결과의 성격과 그 결과의 활용 방안에 대한 가이드라인의 역할을 할 수 있도록 할 것이다.

2. 프로그램 표절 감정법

소프트웨어 감정 도구의 핵심 부분인 표절 자동 검출에 사용되어 온 방법론은 크게 두 가지로 분류할 수 있다. 첫째 방법론은 속성 계수(attribute counting)[1] 또는 순위 측정(ranking measure)[2]이다. 둘째 방법론은 제어 흐름(control flow) 또는 구조 측정(structure metric) 방법이다. 속성 계수 방법론에 근거한 프로그램들은 1970년대에 나오기 시작했는데, 그들은 Halstead의 소프트웨어 과학 측정값(software science

metric)[3] 또는 McCabe의 cyclomatic 복잡도(cyclomatic complexity)[4] 또는 범위값(scope number)[5]을 사용하였다.

Halstead의 측정값은 아래와 같은 양들을 사용하였다.

n1 = 유일한 연산자의 수(number of unique or distinct operators)

n2 = 유일한 연산수의 수(number of unique or distinct operands)

N1 = 모든 연산자의 사용 회수(total usage of all the operators)

N2 = 모든 연산수의 사용 회수(total usage of all the operand)

그런 다음 Halstead는 vocabulary를

$$n = n1 + n2$$

로 계산하고, 구현 length를

$$N = N1 + N2$$

로 계산하였다. 그리고 나서 이러한 측정값들로부터 프로그램의 크기 측정값인 volume을 아래와 같이 계산하였다.

$$V = N \log2 (n)$$

McCabe의 cyclomatic 복잡도는 실행 경로(execution path)의 수를 계산하여 프로그램의 제어 흐름을 측정한다. 측정값 V(G)는 프로그램의 흐름 그래프(flow graph) 표현으로부터 도출할 수 있다. 흐름 그래프의 각 노드(node)는 프로그램에서의 일련의 명령문 블록에 해당하고, 에지(edge)는 프로그램의 처리 경로(processing path)에 해당한다.

$$V(G) = e - n + 2p$$

위 식에서

e = 그래프에 존재하는 에지의 수(the number of edges in the graph)

n = 그래프에 존재하는 노드의 수(the number of nodes in the graph)

p = 그래프에 존재하는 컴포넌트의 수(the number of components in the graph), 모듈이 한 개면 p = 1

을 나타낸다.

중첩 수준(nesting level) 측정값은 한 프로그램이나 모듈의 평균 중첩 깊이(nesting depth)를 나타내는데, 이는 각 코드 줄에 깊이 수치를 배정함으로써 가능하게 된다. 각 명령문에 주어진 깊이 값을 모두 더하면 전체 중첩 깊이가 되고, 이 값을 프로그램이나 모듈에 있는 명령문의 수로 나누면 평균 중첩 깊이가 된다.

Halstead의 측정값을 처음으로 사용한 사람은 Ottenstein[6]으로, 그는 FORTRAN 프로그램에서 표절을 검출하기 위한 프로그램을 개발하였다. 그러나 이 기법은 길이가 아주 짧은 프로그램에서 표절을 검출하는 데는 효과적이지 못했다. 이에 따라, 첫째 방법론인 속성 계수에서 둘째 방법론인 제어 흐름(control flow) 또는 구조 측정(structure metric) 기법으로의 전환이 시작되었다.

3. 프로그램 표절 감정 툴에 대한 비교, 분석

현재까지 일반에게 알려진 감정 도구, 특히 소프트웨어 표절 검출 도구는 제공되는 형태에 따라서 크게 세 가지로 나눌 수 있다.

(1) 상업용 소프트웨어: 유료로 판매되고 사용하기 위해서는 라이센스를 받아야 하는 소프트웨어로 McCabe와 Windiff가 여기에 해당한다.

(2) 인터넷 서비스: 인터넷을 이용하여 표절 검출 기능을 무료로 서비스하는 소프트웨어로, Moss와 JPlag이 이 부류에 해당한다.

(3) 공개 소프트웨어: 인터넷을 이용하여 소스코드 또는 실행파일을 다운로드한 후 자신의 컴퓨터에 설치하여 사용하는 소프트웨어로, 이 부류에는 SIM, YAP, CloneChecker가 있다.

3.1. 상업용 소프트웨어

유료로 판매되고 사용하기 위해서는 라이센스를 받아야 하는 소프트웨어로 Windiff 와 McCabe가 여기에 해당된다.

3.1.1. Windiff

Microsoft Visual Studio 안에서 제공하는 도구로서 디렉토리, 파일 비교가 가능하며 윈도우 환경에서 바로 사용이 가능하다. 사용자 인터페이스가 GUI를 제공하여 편리성을 제공해주며 소스코드 비교가 가능하다. 표 1은 Windiff를 이용하여 계산한 파일 구조 복제도와 소스코드, 자료구조 복제도를 정리하여 전체 프로그램의 복제도를 산출한 결과의 예를 보여준다.

<표 1> Windiff를 이용한 복제도 분석표

기준	소스코드화일		자료구조		화일구조	
	복사율	복제도	복사율	복제도	복사율	복제도
서버	32.2%	16.1%	33%	9.9%	83%	16.6%
클라이언트	83.0%	41.5%	82.3%	24.7%	90%	18.0%
캡처	98.5%	49.3%	99.5%	29.8%	100%	20.0%
맵툴	84.7%	42.4%	92.5%	27.8%	100%	20.0%

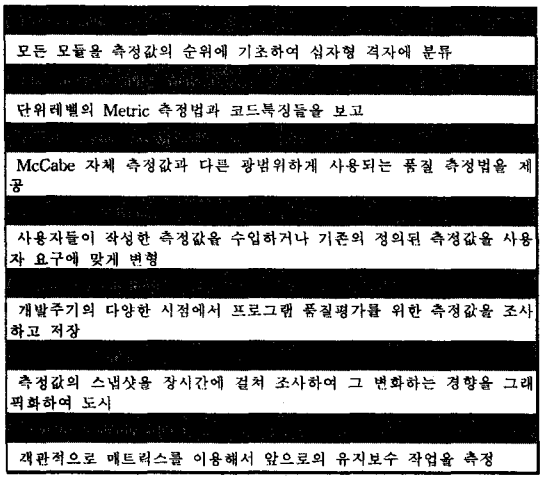
3.1.2. McCabe

McCabe는 모듈의 사이즈와 구조를 결정하는 하나의 모듈을 통해서 독립적인 실행 경로의 수를 계산함으로써 소프트웨어 품질을 측정한다. McCabe는 <표 2>와 같은 많은 기능을 제공한다. McCabe는 표절 검출을 위한 도구가 아니므로 정확하게 이 목적으로 사용할 수 있는 도구를 제공하지 않으며, 상업적 소프트웨어로서 고가로 널리 유통되지는 않는다.

<표 2>는 McCabe의 기능을 도시하였다.

<표 2> McCabe의 기능

사용자가 열거한 측정값에 기초하여 시스템의 칼라 코드화된 구조도를 제공
각 모듈에 대한 흐름 그래프와 소스코드를 함께 제공하고 있으며, 각 모듈의 제어 구조를 파악할 수 있게 해준다.
시스템을 데이터 측면에서 분석하는 기능
모듈간의 인터페이스를 시험할 수 있는 통합 시험의 시나리오 제공
단위 시험 계획을 제공



3.2.. 인터넷 서비스

인터넷을 이용하여 표절 검출 기능을 무료로 서비스하는 소프트웨어로 Moss와 JPlag가 이 부류에 해당한다.

3.2.1. Moss

이 도구의 목적은 C, C++, Java, Pascal, Ada, ML, Lisp 또는 Scheme으로 작성된 소스코드의 유사성을 측정하는 것이다. 현재까지 Moss의 주요 적용분야는 프로그래밍 수업에서 표절을 검출하는 것이며, 이 목적에 있어서는 매우 효과적이었다.

이 도구는 인터넷 서비스 형태로 제공되며 인터넷을 통해서 소스 파일 묶음이 주어지면 유사한 소스코드를 가진 프로그램의 쌍들을 나열한 HTML 페이지를 출력한다. 또한, 프로그램 안의 개별적인 경로들을 하이라이트로 표시하여 쉽게 추적, 비교할 수 있게 해준다. 그리고, 공유하는 것이 당연한 소스 코드는 유사성 검사에서 제외시켜준다. Window나 UNIX 시스템에서 모두 사용할 수 있다.

3.2.2. JPlag

이 도구는 C, C++, Java, Scheme 그리고 자연어로 작성된 여러 세트의 프로그램 소스 코드의 유사성을 검출한다. 사용방법은 문장 및 프로그램 구조를 함께 비교하여 준다. 이렇게 함으로써 표절을 위장하는 여러 가지 방식에 대처할 수 있게 해준다. <표 3>은 Moss와 JPlag에서 검출 가능한 언어별로 비교[7]한 것이다.

<표 3> Moss와 JPlag의 검출 언어 비교

	JAVA	C++	C	Pascal	Ada	ML	Lisp	Scheme
MOSS	0	0	0	0	0	0	0	0
JPlag	0	0	0					0

3.3. 공개 소프트웨어

인터넷을 통하여 소스코드 또는 실행파일을 다운받아 자신의 컴퓨터에 설치하여 사용하는 소프트웨어로 SIM, YAP, CloneChecker가 있다.

3.3.1. SIM

SIM은 소프트웨어 프로젝트에서의 표절과 대형 소프트웨어 프로젝트에서 복제가 의심되는 코드 조각을 검출할 수 있다. SIM의 출력은 히스토그램 EH는 의심되는 프로그램 제출물의 목록을 작성하기 위한 쉘 스크립트로 처리할 수 있으며, 토큰을 기본 단위로 하여, 연속되는 여러 개의 토큰으로 구성되는 런을 비교 단위로 하여 두 파일을 비교한다. SIM의 기본적인 알고리즘은 표 4와 같다.

<표 4> SIM의 기본 알고리즘

1. 프로그램 파일을 읽어들인다.
2. 전향 참조표를 작성한다. 최소 크기를 만족하는 각 텍스트 서브스트링을 자신의 오른쪽에 있는 다른 모든 서브스트링과 비교한다. 이 결과가 전향 참조 표인데, 동일한 위치로 해쉬되는 다음 서브스트링의 인덱스를 저장한다. 유사한 다음 서브스트링이 발견되지 않았으면 인덱스는 0이다.
3. 의미있는 런들의 집합을 결정한다.
4. 의미있는 런들의 줄 번호를 결정한다.
5. 런들의 내용을 순서대로 출력한다.

SIM는 메뉴 방식이 아닌 명령줄 방식으로 사용하도록 되어 있어 몇 가지 명령어를 암기하거나 찾아봐야 하는 단점이 있다.

3.3.2. YAP

YAP의 목적은 이전의 속성 계수 방법이나 구조 기반 방법들보다 성공적이었던 Plague를 기반으로 하되, Plague의 문제점을 극복할 수 있게 하는 것이었다. 모든 YAP 시스템들은 같은 방식으로 작동하며 두 단계로 구성된다.

1단계 : 각 제출물에 대해서 하나의 토큰 파일을 생성한다.

2단계 : 토큰 파일의 쌍들을 비교한다.

YAP에 의해서 검출될 수 있는 복제 행위에는 <표 5>와 같은 것들이 있다. 또한 <표 6>은 YAP를 이용한 정밀도 수준을 나타낸다.

<표 5> YAP에 의한 검출 가능한 복제

- 주석이나 출력 포맷 변경
- 식별자 변경
- 연산수의 변경
- 데이터 타입 변경
- 식을 동등한 다른 것으로 변경

<표 6> YAP 1, 2, 3의 정밀도 수준

	Q/Q1	Q/Q2	Q1/Q2
YAP1	73	55	65
YAP2	98	56	62
YAP33	97	69	69
YAP32	100	75	75

3.3.3. CloneChecker

이 도구는 두 프로그램을 비교해서 유사도를 0~1 사이의 실수값으로 알려준다. 단순히 두 프로그램의 텍스트만을 비교하는 것과 달리, 프로그램에서 사용된 이름등을 생략하고 구문 구조만을 비교함으로써 변수 이

름 바꾸기, 함수 순서 바꾸기 등에 영향을 받지 않는다. C, Java, S초듣, nML로 짜여진 프로그램 등에 대하여 사용할 수 있다. CloneChecker의 유사성 검사 방법은 요약, 유사성 비교, 그룹짓기의 세 과정을 거친다. 두 번째 단계인 유사성은 <표 7>에서와 같이 구해진다.

<표 7> CloneChecker에서 유사성의 정의

$$\text{simility} = \frac{S_a + S_b}{T_a + T_b}$$

Sa = number of sub-tree of a that appears identically in b
Sb = number of sub-tree of b that appears identically in a
Ta = number of total nodes(sub-trees) in tree a
Tb = number of total nodes(sub-trees) in tree b

세 번째 단계에서는 유사성에 따라 프로그램을 기본적으로 <표 8>과 같이 그룹 짓는다. 그룹 안의 임의의 두 프로그램의 유사성은 0과 1사이의 실수인 전역유사성보다 큰 값을 갖게 된다.

<표 8> 유사성에 의한 그룹 짓기

정의 1. 전역유사성 g에 의해 정의된 그룹 G는
$\forall a \in G, \forall b \in G, \text{simility}(a,b) \geq g$
정의 2. 지역유사성 1에 의해 정의된 그룹 G는
$\forall a \in G, \exists b \in G, \text{simility}(a,b) \geq 1$

4. 개발 틀에 대한 방향 제시

최근의 소프트웨어개발 방향은 비주얼 환경에서 객체지향 기법의 언어를 사용하고 있으며 Web과 GUI를 활용하고 있다. 이를 위해서는 개인용 컴퓨터의 윈도우 환경에서 사용할 수 있는 감정도구가 필요하며, 파일구조와 소스코드에 대한 많은 정보를 제공해 주어야 할 것이다.

<표 9> 표절 검출 감정 도구의 개발 방향

기존도구	개발방향
-UNIX, 도스환경 -명령어 형태 -구조적언어기반 -스트링 매칭 -단일 파일 비교 -사용에 폐쇄적 -문법적, 토큰 형태 분석	-개인용컴퓨터, 윈도우 환경 -GUI 형태, Web 제공 -비주얼 언어기반 -여러 파일 비교기능 공개소프트, 참조파일 등 -논리형 흐름 분석 -컨텐츠 감정기법 -언어의 변형에 대처 (C언어->C++, JAVA 등)

5. 결론

본 연구에서는 소프트웨어 감정에 주로 사용되어 온 도구들의 현황을 조사하고, 각 감정 도구를 비교 및 분석하고, 감정 도구의 활용 방법과 감정 도구 개발 방안을 제시하였다. 연구 결과를 요약하면 다음과 같다.

◉ 소프트웨어 감정 도구 현황조사
- 현재 나와 있는 감정 도구에서 주로 사용되는 방법론을 크게 속성 계수 방법과 구조 측정 방법으로 분류하고, 각각에 대하여 기술하였으며, 각 방법론을 적용한 도구에는 어떠한 것들이 있는지 살펴보았다.

◉ 소프트웨어 감정 도구 비교·분석
- 국외 감정 도구로서는 McCabe, Plague, JPlag, MOSS, YAP, SIM, Windiff 의 개요, 주요 기능, 적용 방법, 성능 등에 대하여 기술하였다.
- Windiff의 실제 코드에 적용 예를 설명하였다.
- 국내 감정 도구로서는 CloneChecker와 LOFC의 기능과 적용환경 및 사용법 등에 대하여 설명하였다.

◉ 소프트웨어 감정 도구 개발 방안
- 기존 감정도구의 제한점을 극복할 수 있는 소프트웨어 감정 도구의 개발 방안을 제시하였다.

소프트웨어 감정 도구에 대한 연구는 국외에서도 체계적으로 이루어진 바가 없는 새로운 분야이다. 따라서 체계적으로 참고할 만한 문헌이나 자료가 부족한 상태에서 연구가 수행되었다. 본 연구의 결과는 감정을 수행하는 사람들이 다양한 사용 가능한 도구들의 내용을 신속하게 파악하여 해당 감정 건에 적합한 도구를 선택하는 데 도움이 될 것이다.

참고문헌

[1]The University of Wisconsin Writing Centre, Writer's handbook: Quoting and paraphrasing, 1997(http://www.wisc.edu/writing/Handbook/QuoSampleParaphrases.html).

[2]G. Whale, Identification of program similarity in large populations, The Computer Journal, Vol. 33, No. 2, 1990.

[3]T. J. McCabe, A complexity measure, IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, pp. 308-320, December 1976.

[4]U. Manber and B. S. Baker, Deducing similarities in Java sources from bytecode, *USENIX Technical Conference*, New Orleans, June 1998.

[5]W. A. Harrison and K. L. Magel, A complexity measure based upon nesting level, ACM SIGPLAN Notices, Vol. 16, No. 3, pp. 63-74, March 1981.

[6]Nejmeh, NPATH: A measure of execution path complexity and its applications,
Communications of the ACM, Vol. 31, pp. 188-200, 1988.