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

조동욱*, 소정**, 김진용***, 최병갑****, 김선영*****
김지영******
*충북과학대 **한국전자통신연구원
***혜천대 ****독원대학 *****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 콘텐츠 등 디지털 정보 제작된 보호는 현재 뿐 아니라 향후 국가의 극적
을 좌우할 수 있을 정도로 대단히 중요한 과제가 아닐 수 없다. 본 논문에서는 소프트웨어 표절의 자
동검출을 위하여 제작된 각종 소프트웨어 감정 도구들을 여러 각도에서 비교, 분석하여 각 도구의 유
용성, 제한성, 주요 적용경향 및 분야, 사용방법 등을 제시함으로써 향후 소프트웨어 복제 감정에 효과
적으로 활용할 수 있도록 하고자 한다.

1. 서론

최근 인터넷의 급속한 발전과 더불어 소프트웨어분
야도 많은 발전을 가져왔다. 그러나 이러한 소프트웨어
는 무분별한 불법 복제 및 도용으로 소프트웨어발전에
저해를 가져오게 되었다. 이러한 이유를 근절하고자 많
은 소프트웨어 표절을 검출하는 감정 도구들은 인터넷
상에 배포 및 요청에 의하여 구할 수 있게 되었다. 그
러나 프로그램 감정 틀들을 실제 프로그램 감정 업무
에 직접 활용하려면, 각 도구의 설치 환경 및 방법, 사
용법, 유용한 범야, 제한점, 기대할 수 있는 결과, 결과
의 활용 방안 등을 자체히 파악하는 것이 필요하다. 프로그램
감정을 수행할 때 대부분의 감정인들은 새로운 도구를
차용하여 파악하여 사용하는 데 추가적으로 시간을 함
애할 수 없기 때문에, 자신에게 익숙한 1-2개 정도의
도구를 반복적으로 사용하는 경향이 두려하다. 이로 인
하여 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 한
다. 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 검
출에 사용되어 온 방법론은 크게 두 가지로 분류할 수
있다. 첫째 방법론은 속성 계수(attribute counting)[1]
또는 순위 측정(ranking measure)[2]이다. 둘째 방법론
은 제어 흐름(control flow) 또는 구조 측정(structure
metric) 방법이다. 속성 계수 방법론에 근거한 프로그
램들은 1970년대에 나오기 시작했는데, 그들은
Halstead의 소프트웨어 과학 측정값(software science

Halstead의 측정값은 아래와 같은 방식을 사용하였다.
\[ n_1 = \text{유일한 연산자의 수 (number of unique or distinct operators)} \]
\[ n_2 = \text{유일한 연산자의 수 (number of unique or distinct operands)} \]
\[ N_1 = \text{모든 연산자의 사용 횟수 (total usage of all the operators)} \]
\[ N_2 = \text{모든 연산자의 사용 횟수 (total usage of all the operands)} \]

그린 다음 Halstead는 vocabulary를
\[ n = n_1 + n_2 \]
로 계산하고, 구현 length를
\[ N = N_1 + N_2 \]
로 계산하였다. 그리고 나서 이러한 측정값들로부터 프로그램의 크기 측정값인 volume을 아래와 같이 계산하였다.
\[ V = \log_2 (n) \]
McCabe의 cyclomatic 복잡도는 실행 경로(execution path)의 수를 계산하여 프로그램의 제어 화로를 측정한다. 즉, 간단한 V(G)는 프로그램의 호름 그래프(flow graph) 표현으로부터 도출할 수 있다. 호름 그래프의 각 노드(node)는 프로그램에서의 일련의 명령문 별점에 해당하고, 에지(edge)는 프로그램의 처리 경로 (processing path)에 해당한다.
\[ V(G) = e - n + 2p \]

위에서
\[ e = \text{그래프에 존재하는 에지의 수 (the number of edges in the graph)} \]
\[ n = \text{그래프에 존재하는 노드의 수 (the number of nodes in the graph)} \]
\[ p = \text{그래프에 존재하는 컴포넌트의 수 (the number of components in the graph)} \]
모듈이 한 개인
\[ p = 1 \]

을 나타낸다.

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


3. 프로그램 표절 감정 용에 대한 비교, 분석

현재까지 일반에게 알려진 감정 도구, 특히 소프트웨어
3.2. 인터넷 서비스

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

3.2.1. Moss


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

3.2.2. JPlag


3.3. 공개 소프트웨어

인터넷을 통해서 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 허스트리그(EH)는 의심되는 프로그램 제출물에 포함된 분야에 대한 정보를 제공한다.

표 4: SIM의 기본 알고리즘

SIM는 메모 방식이 아닌 명령어 방식으로 사용하도록 되어 있어 몇 가지 병렬기를 힙기하거나 찾아야 하는 단점이 있다.

3.3.2. YAP

YAP의 목적은 이전의 속성 개수 방법이나 구조 기반 방법들보다 공정적이었던 Plague를 기반으로 하여, Plague의 문제점을 극복할 수 있게 했다. 몇몇 YAP 시스템들은 같은 방식으로 작동하며 두 단계로 구성된다.

1단계: 자 제출물에 대해서 하나의 톤과 파일을 생성한다.

2단계: 톤과 파일의 생성을 비교한다.

YAP에 의한 검출 방법은 복제 행위에는 표적사 <표 5>와 같은 것들이 있다. 또한 <표 6>은 YAP를 이용한 검정도 수준을 나타낸다.

표 5: YAP에 의한 검출 가능한 복제

<table>
<thead>
<tr>
<th>주석이나 출력 포맷 변경</th>
<th>식별자 변경</th>
<th>연산수의 변경</th>
<th>데이터 타입 변경</th>
<th>식별 동등한 다른 것으로 변경</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>YAP1</th>
<th>YAP2</th>
<th>YAP3</th>
<th>YAP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>97</td>
<td>98</td>
<td>100</td>
</tr>
<tr>
<td>55</td>
<td>69</td>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>65</td>
<td>69</td>
<td>62</td>
<td>75</td>
</tr>
</tbody>
</table>

표 6: YAP 1, 2, 3의 검정도 수준

3.3.3. CloneChecker

이 도구는 두 프로그램을 비교하여 유효기간 0~1 사이의 인수값으로 알려준다. 단순히 두 프로그램의 테스트만 비교하여 결과가 달라, 프로그램에서 사용된 이름들을 생략하고 구문 구조만 비교함으로써 변수 이
.SQL 메소드, 함수 순서 바꾸기 등에 영향을 받지 않는다. C, Java, S초본, nML로 바꾸어 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

\[
\text{similarity} = \frac{S_a + S_b}{T_a + T_b}
\]

\[
\begin{align*}
S_a &= \text{number of sub-tree of a that appears} \\
& \text{identically in b} \\
S_b &= \text{number of sub-tree of b that appears} \\
& \text{identically in a} \\
T_a &= \text{number of total nodes(sub-trees) in tree a} \\
T_b &= \text{number of total nodes(sub-trees) in tree b}
\end{align*}
\]

세 번째 단계에서에서는 유사성에 따라 프로그램을 기본적으로 <표 8>과 같이 그룹을 정한다. 그룹 안의 임의의 두 프로그램의 유사성은 0과 1사이의 실수인 전역

유사성보다 큰 값을 갖게 된다.

<표 8> 유사성에 의한 그룹 관리

<table>
<thead>
<tr>
<th align="left">정의 1: 전역유사성을 g에 의해 정의된 그룹 G는</th>
<th align="left">정의 2: 지역유사성 1에 의해 정의된 그룹 G는</th>
</tr>
</thead>
<tbody>
<tr>
<td align="left">(\forall a \in G, \forall b \in G, \text{similarity}(ab) \geq g)</td>
<td align="left">(\forall a \in G, 3 \exists b \in G, \text{similarity}(ab) \geq 1)</td>
</tr>
</tbody>
</table>

4. 개발 통에 대한 방향 제시

최근의 소프트웨어 개발 방향은 비주얼 환경에서 계체적, 기법의 인터페이스를 사용하고 있으며 Web과 GUI를 활용하고 있다. 이를 위해서는 개인용 컴퓨터의 화면 환경에서 사용할 수 있는 감정도구가 필요하며, 파일구조와 소스코드에 대한 많은 정보를 제공해 주어야 할 것이다.

<표 9> 표절 검출 감정 도구의 개발 방향

<table>
<thead>
<tr>
<th>감정도구</th>
<th>개발방향</th>
</tr>
</thead>
<tbody>
<tr>
<td>-unix, 도스환경</td>
<td>-개인용 컴퓨터, 웹상 환경</td>
</tr>
<tr>
<td>-명령어 형태</td>
<td></td>
</tr>
<tr>
<td>-구조적언어기반</td>
<td>-GUI 형태, 웹 제공</td>
</tr>
<tr>
<td>-스트림 메시지</td>
<td>-비주얼 언어기반</td>
</tr>
<tr>
<td>-단일 파일 비교</td>
<td>-여러 파일 비교기능</td>
</tr>
<tr>
<td>-사용에 폐쇄적</td>
<td>-공개소프트, 참조파일 등</td>
</tr>
<tr>
<td>-문법적</td>
<td>-런리형 코드 분석</td>
</tr>
<tr>
<td>-콘텐츠 형태 분석</td>
<td>-콘텐츠 감정기법</td>
</tr>
<tr>
<td></td>
<td>-언어의 변형에 대처</td>
</tr>
</tbody>
</table>

5. 결론

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

소프트웨어 감정 도구 현황조사

- 현재 나아지는 감정 도구에서 주로 사용되는 방법론, 크게 속성 계수 방법과 구조 특성 방법으로 분류하고, 각각에 대하여 기술하였으며, 각 방법론을 적용한 도구에는 어떠한 것들이 있는지 살펴보았다.

- 소프트웨어 감정 도구 비교, 분석
  - 국내 감정 도구로서는 McCabe, Plague, JFlag, MOSS, YAP, SIM, Windiff의 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 외에도 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 감정에 적합한 도구를 선택하는 데 도움이 될 것이다.

참고문헌


