논문 2015-1-4

벨론 레퍼런스 코퍼스를 기준으로 exEyes의 재현율 평가

최성하*, 도경구*†

Assessment of exEyes' Recall using Bellon Reference Corpus as a Benchmark

Sungha Choi*, Kyung-Goo Doh*†

요 약

소프트웨어는 개발자의 전문적인 기술과 경험이 들어가 만들기 때문에 개발자에게 저작권이 주어진다. 한국저작권위원회에서는 소스코드 유사도 탐지도구인 exEyes로 소프트웨어의 표절 여부를 감정한다. 이 논문에서는 코드 클론 탐지 프로그램의 성능평가 벤치마크인 벨론 레퍼런스 코퍼스를 기준으로 exEyes의 재현율을 평가한다. 벨론 레퍼런스 코퍼스에서 4개의 오픈소스(C언어 기반 : cook, weltab, Java 기반 : eclipse-ant, netbeans-javadoc, 10,055개 클론)를 선택하여 클론 타입 별 재현율을 측정한다. 그 결과 클론 타입 1은 100%, 타입 2는 63%, 타입 3은 34% 재현하는 것으로 나타났다. exEyes의 미탐 원인은 토큰의 의미를 고려하지 않고 토큰의 유사성을 판정하고, 줄 단위로 비교를 함으로써 줄 바꿈 및 문장의 추가/삭제/변경에 적절히 대응하지 못하기 때문으로 나타났다.

Abstract

Copyrights for software source codes are given to developers. Korea Copyright Commission utilizes a clone-detection tool, exEyes, to find code clones that can be used to assess software plagiarism. This paper evaluates the recall of exEyes using Bellon Reference Corpus as a benchmark. Four open sources(cook and weltab in C, eclipse-ant and netbean-javadoc in Java) in Bellon Reference Corpus are selected as the benchmark. Among 10,055 clones in the corpus, exEyes' recall rate is 100% in clone type 1, 63% in clone type 2, and 34% in cone type 3. False negatives turn out to be mainly caused by ignoring the meaning of tokens when the comparison is made, and by setting the comparison be made line-by-line.

한글키워드 : 소프트웨어감정, 표절탐지, 코드클론, 재현율

keywords : software evaluation, piracy detect, code clone, reproduce rate

* 한양대학교 컴퓨터공학과
† 교신저자:도경구
(email: honggd@hankook.re.kr)
접수일자: 2015.5.22. 심사완료: 2015.6.3.
게재확정: 2015.6.20.

1. 서 론

소프트웨어는 스마트폰, 의료, 국방, 교통, 은

행, 조선, 항공, 자동차 등 다양한 분야의 산업에 접목되어 사용된다. 기계를 정밀하게 다뤄야하거나 보안을 필요로 하는 일 등에 소프트웨어가 필수적으로 사용되기 때문에 소프트웨어의 수요는 점차 늘어나고 있다. 그만큼 소프트웨어 개발 업체들끼리 치열한 경쟁을 통하여 공급이 이루어지고 있다. 경쟁력있는 소프트웨어를 만들려면 개발자의 전문적인 지식과 다양한 개발 경험을 필요로 한다. 개발 노하우(know-how)는 소프트웨어 소스코드에 녹아 들어간다. 소프트웨어의 소스코드는 가격과 경쟁력, 독창성을 나타내기 때문에 저작권을 주장할 수 있으며 보호 받도록 되어있다.

매년 한국저작권위원회의 소프트웨어에 대한 저작권 분쟁조정은 전체 저작권 분쟁 조정 건수의 20%를 차지한다[2]. 그래서 한국저작권위원회에서는 소스코드의 무단복제와 표절 여부를 감정하는 작업을 돕기 위해 자체 외주 개발한 코드클론 탐지 프로그램인 exEyes[3]를 사용하고 있다.

이 논문에서는 exEyes의 재현율(recall)을 평가한다. 재현율이란 코드클론 탐지도구가 클론을 실제로 얼마나 빠짐없이 찾는지 알려주는 척도로 백분율로 나타낸다. 벨론 레퍼런스 코퍼스(Bellon Reference Corpus)[4]는 코드 클론 탐지 도구의 재현율을 평가하기 위한 목적으로 모아놓은 클론 집합이다. 2002년에 처음 발표되어 지금까지 코드 클론 탐지 프로그램의 재현율 평가용 벤치마크로 널리 사용하고 있다.

본 논문의 구성은 다음과 같다. 2장에서 클론의 타입별 분류를 정의하고, 3장에서 이 논문에서 벤치마크로 사용하는 벨론 레퍼런스 코퍼스를 소개한다. 4장에서는 exEyes의 클론 탐지 설정값과 탐지 알고리즘을 설명하고, 5장에서는 exEyes의 재현율을 평가하고, 6장에서 마무리한다.

2. 클론의 분류

코드 클론을 탐지하는 목적은 매우 다양하다. 구조적으로 비슷한 클론을 찾을 때도 있고 의미적으로 비슷한 클론을 찾아야 될 때도 있다. 그래서 일반적으로 클론을 다음과 같이 3가지 타입으로 분류한다[5,6,7,8].

* 타입 1 : 공백과 주석, 레이아웃을 제외하고 동일
* 타입 2 : 타입, 변수이름, 표현식을 제외하고 구조적/외형적으로 동일
* 타입 3 : 명령어의 변경 또는 삽입과 삭제를 제외하고 동일

표 1. 예제 - 코드 A
Table 1. Example - code A

```
1: void sumProd(int n) {
2:    float sum = 0.0; // 주석입니다.
3:    float prod = 1.0;
4:    for(int i = 1; i<=n; i++)
5:       { sum = sum + i;
6:         prod = prod * i;
7:         foo(sum, prod); }}
```

표 2. 예제 - 코드 B
Table 2. Example - code B

```
1: void sumProd(int n) {
2:    float sum = 0.0; // 다른 주석입니다.
3:    float prod = 1.0; // 새 주석입니다.
4:    for(int i = 1; i<=n; i++) {
5:             sum = sum + i;
6:             prod = prod * i;
7:             foo(sum, prod); }}
```

표 2의 코드 B를 표 1의 코드 A와 비교해보자. 코드 B의 2번 줄에서는 주석이 변경되었고, 3

번 줄에서는 주석이 새로 삽입되었다. 코드 A의 5번 줄 맨 앞에 있는 여는 중괄호는 코드 B에서는 4번 줄의 맨 뒤에 있는데 이는 레이아웃 변경에 해당한다. 그리고 5, 6, 7번 줄에서는 앞부분에 공백이 삽입되었다. 이는 모두 타입 1 클론을 결정하는데 무시하는 요소이므로 코드 A와 코드 B는 타입 1 클론이라고 할 수 있다.

표 3. 예제 - 코드 C
Table 3. Example - code C

```
1: void sumProd(int n) {
2:    int sum = 0;  // 주석입니다.
3:    float p = 1.0;
4:    for(int i = 1; i<=n; i++)
5:      { sum = sum + (i*i);
6:        prod = prod * (i*i);
7:        foo(sum, p); }}
```

표 3의 코드 C를 코드 A와 비교해보자. 코드 C의 2번 줄에서는 오른쪽 상수식의 타입이 정수로 바뀌었고, 3번과 7번 줄에서는 변수이름이 p로 바뀌었다. 그리고 5번, 6번 줄에서는 오른쪽 표현식의 일부가 변경되었다. 이 경우 타입, 변수이름, 표현식이 다르므로 코드 A와 코드 C를 타입 1 클론이라고 할 수 없지만, 나머지 부분은 같으므로 타입 2 클론이라고 할 수 있다.

표 4. 예제 - 코드 D
Table 4. Example - code D

```
1: void sumProd(int n) {
2:    float sum = 0.0;  // 주석입니다.
3:    float prod = 1.0;
4:    for(int i = 1; i<=n; i++)
5:      {  if(i/2) sum += i;
6:        foo(sum, prod, n); }}
```

표 4의 코드 D를 코드 A와 비교해보자. 코드 D의 5번 줄은 문장이 변경되었고, 코드 D의 6번 줄은 인수가 하나 추가되었다. 코드 A의 6번 줄은 코드 D에서는 삭제되었다. 그런데 이와 같은 추가, 삭제, 변경 때문에 코드 A와 코드 D는 타입 1, 타입 2 클론은 아니지만, 타입 3 클론이라고는 할 수 있다.

3. 벨론 레퍼런스 코퍼스

표 5. 벨론 레퍼런스 코퍼스 구축에 사용한 클론 탐지 도구
Table 5. Bellon Reference Corpus tool

문자열기반	트리기반	계량치기반
Duploc	CloneDR	CLAN
토큰기반		의존그래프기반
Dup, CCFinder		Duplix

표 6. 벨론 레퍼런스 코퍼스 구축에 사용한 오픈소스
Table 6. Bellon Reference Corpus open source

언어	오픈소스 이름	파일 크기
C	Cook	2.8MB
	Weltab	450KB
	snns	5.5MB
	postgrsql	10MB
Java	eclipse-ant	1.5MB
	netbeans-javadoc	709KB
	eclipse-jdtcore	7.2MB
	j2sdk1.4.0-javax-swing	8.8MB

벨론 레퍼런스 코퍼스(Bellon Reference Corpus)[4]는 코드클론 탐지도구의 재현율을 평가하기 위한 벤치마크이다. 재현율이란 실제 클론을 얼마나 많이 찾는지를 나타내주는 척도로 백분율로 표현한다. 재현율을 계산하기 위해서는

분모 역할을 할 벤치마크가 필요한데 벨론 레퍼런스 코퍼스가 바로 그 벤치마크 역할을 할 목적으로 구축되었다. 표 6에 나열한 오픈소스를 대상으로 표 5에 나열한 코드클론 탐지도구를 동원하여 탐지한 코드클론을 모두 모은 뒤, 결과의 일부(2%)를 무작위로 골라 맨눈으로 분석하여 진짜 코드클론을 타입 별로 가려내는 방식으로 구축하였다. 벨론 레퍼런스 코퍼스에는 총 325,935개의 클론이 있으며, 각 클론 마다 파일의 경로 정보, 클론의 시작 줄번호, 끝 줄번호, 클론의 타입, 탐지한 도구 정보를 가지고 있다. 벨론 레퍼런스 코퍼스는 새로 개발한 코드클론 탐지도구의 재현율을 평가하는 기준으로 학계에서 주로 많이 사용해왔다.

4. exEyes

exEyes는 한국저작권위원회에서 소스코드 표절여부 감정평가용으로 개발한 코드클론 탐지 도구이다. 원칙적으로 프로그래밍언어에 상관없이 비교할 수 있으며, 대상 소스코드의 성질 및 감정 목적에 따라 다양하게 설정 값을 지정하여 탐지의 정밀도를 조절할 수 있다.

표 7. exEyes 설정 값
Table 7. exEyes initialize

설정 항목	단위
유효 동일/유사 블록 최소 크기	줄
유사라인 판정 동일 토큰수 비율	%
유사라인 판정 최소 토큰수	개
유사라인 판정 최대 토큰비	배
토크나이징 선행 여부	O / X

exEyes는 유사여부를 줄 단위로 판정하며, 비교 단위는 문자 또는 토큰 중 하나를 선택할 수 있다. 문자를 선택하는 경우는 문자열을 문자단위로 비교하고, 토큰을 선택하는 경우는 일반적인 방법으로 토큰을 인식하여 구분한 뒤 토큰단위로 비교한다. 줄 별로 어느 정도 유사해야 유사한 줄로 판정하는지에 대한 기준도 표 7과 같이 사용자가 설정하게 하고 있다.

클론의 최소 줄 수를 지정하여, 클론의 줄 수가 지정한 수 미만이면 클론에서 제외한다. 유사한 문자열을 만났을 경우 줄의 토큰유사도를 기준으로 탐지한다. 클론을 탐지한 후 유사한 부분은 웹페이지나 자체 프로그램 내에서 하나씩 확인 할 수 있다.

벨론 레퍼런스 코퍼스 벤치마크는 주석을 모두 무시하고 실제 코드의 대한 클론만 탐지하지 하여 구축하였다. 그런데 exEyes는 주석을 무시하지 않고 코드에 포함하여 탐지한다.

5. 재현율 평가

exEyes의 코드 클론 탐지의 재현율을 벨론 레퍼런스 코퍼스 벤치마크를 기준으로 평가한다. 모든 탐지 결과를 맨눈으로 최종 확인하여 판정하기 때문에 벨론 레퍼런스 코퍼스 제작에 사용된 오픈소스 8개 중에서 비교적 규모가 작은 C언어 기반의 cook과 weltab, Java기반의 eclipse-ant와 netbeans-javadoc의 언어별로 두 개씩 총 4개를 대상으로 선택하였다.

클론 탐지 설정 값은 벨론 레퍼런스 코퍼스의 모든 클론 중 제일 작은 클론의 크기가 5줄이라서 유효 동일/유사 블록 크기 값을 “5”로 설정하였고, 나머지 설정 값은 기본 설정 값으로 유사라인 판정 동일 토큰 수 비율은 80%, 유사라인 판정 최소 토큰 수는 “3”, 유사라인 판정 최대 토큰 비는 “2”로 하였다.

5.1 주석 제거

벨론 레퍼런스 코퍼스는 주석을 무시하고 수집한 클론이다. 그런데 exEyes는 주석도 모두 고려하여 클론을 탐지하므로, 사전에 소스코드에서 주석을 모두 제거하고 exEyes를 실행하였다.

표 8. 원본 코드와 주석제거 된 코드의 예
Table 8. Original and comment delete code

원본 코드	주석제거 된 코드
01: int a = 10; 02: **//this is comment** 03: char b = 'b'; **//c1** 04: **/*this is** 05: **comment */** 06: float c = 1.0;	01: int a = 10; 03: char b = 'b'; 06: float c = 1.0;

표 8과 같이 주석은 원본 코드에서 모두 제거하되 기존의 줄 정보를 유지하기 위하여 기존의 원본 코드의 줄 정보를 참고하여 복원시키는 작업을 exEyes 성능평가하기 전에 처리하였다.

5.2 재현율

exEyes의 재현율은 벨론 레퍼런스 코퍼스의 클론 중 exEyes가 탐지한 벨론 레퍼런스 코퍼스 클론으로 다음 식으로 계산한다.

$$\frac{exEyes\text{가 찾은 벨론 레퍼런스 코퍼스의 클론의 수}}{\text{벨론 레퍼런스 코퍼스의 클론 수}}$$

exEyes가 탐지한 클론의 시작 줄번호와 끝 줄번호를 가지고 벨론 레퍼런스 코퍼스에 있는 클론 정보 중 일치하는 클론이 있다면 재현이 된 것으로 보았다.

5.3 exEyes의 재현율

벨론 레퍼런스 코퍼스에는 동일 파일 내부에 존재하는 클론이 포함되어 있는데, exEyes는 파일을 자체 비교하여 파일 내부에 있는 클론은 찾지 않는다. 따라서 벨론 레퍼런스 코퍼스에서 동일 파일 내부에 존재하는 클론은 재현율 평가의 모수에서 제외하였다.

표 10. exEyes의 소스별/타입별 재현율
Table 10. exEyes reproduce rate

오픈 소스	타입1	타입2	타입3	총 합
cook	100% ($\frac{1,454}{1,454}$)	56% ($\frac{3,605}{6,483}$)	29% ($\frac{611}{2,118}$)	56% ($\frac{5,670}{10,055}$)
weltab	100% ($\frac{573}{573}$)	81% ($\frac{1,825}{2,246}$)	43% ($\frac{524}{1,226}$)	72% ($\frac{2,922}{4,045}$)
eclipse-ant	100% ($\frac{34}{34}$)	78% ($\frac{101}{130}$)	13% ($\frac{1}{8}$)	79% ($\frac{136}{172}$)
netbeans-javadoc	100% ($\frac{73}{73}$)	84% ($\frac{202}{241}$)	44% ($\frac{8}{18}$)	85% ($\frac{283}{332}$)
총 합	**100%** ($\frac{2,134}{2,134}$)	**63%** ($\frac{5,733}{9,100}$)	**34%** ($\frac{1,144}{3,371}$)	**62%** ($\frac{9,011}{14,604}$)

exEyes의 소스별/타입별 재현율은 표 10에 정리되어 있다. 타입 1 클론은 하나도 빠짐없이 재현 하여 100%의 총 재현율을 나타냈다. 타입 2 클론은 모든 오픈소스 전체의 63%를 재현하였다. 하지만 타입 3 클론은 재현율이 34%로 나타났다.

exEyes의 타입 2 클론 재현율이 낮은 이유는 상이한 변수나 리터럴의 비중이 큰 라인의 경우 프로그램의 구조가 유사함에도 불구하고 다르게 판정하기 때문이다. 토큰의 의미를 구분하지 않

는 exEyes의 단점이 드러나는 대목이다.

타입 3 클론의 재현율이 낮게 나오는 이유는 exEyes가 소스코드의 구조 또는 의미를 전혀 고려하지 않으므로 당연한 결과이다. 하지만 일부 문장이 삽입 또는 제거된 클론의 경우 exEyes의 결과를 맨눈으로 검사하여 수동으로 탐지 가능한 경우가 있을 수는 있다.

5.4 exEyes의 미탐 유형

exEyes은 타입 1의 클론의 경우 완벽하게 재현한다. 하지만 타입 2 클론과 타입 3 클론은 완벽하게 재현하지 못한다. exEyes가 빠트리고 찾아내지 못한 클론을 맨눈으로 확인한 결과 미탐 유형을 원인별로 다음과 같이 세 가지로 분류할 수 있었다.

- 토큰의 의미를 고려하지 않음
- 줄 바꿈
- 문장 삽입 / 삭제 / 변경

유형 1 : 토큰의 의미를 고려하지 않음

다음은 exEyes가 찾지 못하는 클론의 대표적인 사례이다. eclipse-ant의 javac.java와 rmic.java에 존재하는 클론에서 exEyes가 찾지 못하는 일부분을 각각 차례로 발췌한 것이다.

```
321: public String getTarget() {
322:     return targer; }
                    :
```

```
222: public Vector getFileList() {
223:     return complieList; }
                    :
```

exEyes는 클론 탐지 작업을 할 때 소스코드 라인별로 미리 정한 구분자를 기준으로 토큰을 식별하고, 식별한 토큰을 가지고 라인별로 유사도를 계산한다. 그런데 토큰 분류시 토큰의 종류를 전혀 구분하지 않기 때문에 이 사례와 같이 유사하다고 분류할 가능성을 놓친다. 위의 사례의 경우 타입과 식별자가 상이하지만 구조가 일치하므로 유사하다고 볼 수 있는 여지가 있음에도, 라인별 동일 토큰이 차지하는 비율이 일정 기준을 넘지 않아 원천적으로 배제되었다.

유형 2 : 줄 바꿈

다음은 eclipse-ant 코퍼스 클론 중에서 copy.java와 move.java에 있는 클론의 일부를 각각 차례로 발췌한 것이다.

```
                    :
298: if (count > 0) {
299:     log("Copied " + count +
300:     " empty director" +
301:     (count==1?"y":"ies") +
302:     " to " + destDir.getAbsolutePath()); } } } }
```

```
                    :
133: if (count > 0) {
134:     log("Moved " + count + " empty directories
to " + destDir.getAbsolutePath()); } }
```

copy.java의 줄번호 299번부터 줄번호 302번까지는 의미적으로 한 문장이다. 그리고 move.java의 줄번호 134번 줄과 클론이다. 하지만 exEyes는 줄 단위로 끊어서 유사도를 검사하기 때문에 탐지하지 못한다.

유형 3 : 문장 삽입 / 삭제 / 변경

다음은 cook 코퍼스 클론 중에서 flag.c와 run.c에 있는 클론의 일부를 각각 차례로 발췌한 것이다.

```
231:   scp = sub_context_new();
232:   sub_var_set(scp, "Name", "%S", name);
233:   sub_var_set(scp, "Guess", "%S", other);
234:   error_with_position
235:     (
236:         pp,
237:         scp,
238: i18n("flag \"$name\" not understood, closest is
      the \"$guess\" flag")
239:     );
240:     sub_context_delete(scp);
241:     /* DO NOT str_free guess */
242:     goto set_it; }
```

```
254:   scp = sub_context_new();
255:   sub_var_set(scp, "File_Name", "%S", target1);
256:   error_with_position
257:         (
258:               &grp->rp->pos,
259:               scp,
260:    i18n("\"$filename\" is out of date because the
      ingredients changed (reason)")
261:               );
262:   sub_context_delete(scp); } }
```

이 두 클론을 보면 몇 줄씩 서로 다른 부분이 있지만 전체적으로 유사한 모양을 가지므로 클론이라고 할 수 있는 전형적인 타입 3의 클론이다. 그러나 이러한 유형의 클론을 exEyes는 특별한 몇 가지의 경우를 제외하고는 찾지 못했다.

클론 타입 3의 클론 중에서 exEyes가 찾은 클론은 주로 문장 삽입이나 삭제가 되지 않고 문장의 변경만 일어난 다음(weltab의 wellib.c와 weltab.c)과 같은 클론이다.

```
17:       FILE *fileid; {
18:             int trimlen;
19:             if (trace) fprintf(stderr,"> cprint\n");
20:             report[80] = '\0';
21:             trimlen = itrim(80,report);
22:            /* printf("trimlen = %d\n",trimlen); */
23:             report[trimlen] = '\0';
24:             fprintf(fileid,"%s\n",report);
25:             blkbuf(80,report);
26:             return; }
```

```
237:     char buf[]; {
238:           int trimlen;
239:           if (trace) fprintf(stderr,"> lhprint\n");
240:           buf[130] = '\0';
241:           trimlen = itrim(130,buf);
242:             /* printf("trimlen = %d\n",trimlen); */
243:           buf[trimlen] = '\0';
244:           fprintf(fileid,"%s\n",buf);
245:           return; }
```

변경된 부분이 일정 기준 이하이기 때문에 exEyes가 탐지한 것으로 보인다.

5.5 유형별 미탐 비율

표 11. exEyes의 유형별 미탐 개수
Table 11. exEyes detection error

오픈 소스	유형 1	유형 2	유형 3	합계
cook	2,609	269	1,507	4,385
weltab	346	75	702	1,123
eclipse -ant	26	3	7	36
netbeans -javadoc	33	6	10	49
전체	**3014**	**353**	**2,226**	**5,593**
비율	**54%**	**6%**	**40%**	100%

exEyes가 재현을 못한 세 가지 유형의 비율은 표 11과 같다. 클론 타입 2의 미탐은 주로 유형 1과 유형 2에서 찾아볼 수 있고, 클론 타입 3의 미탐은 주로 유형 3에서 찾아 볼 수 있다.

6. 결 론

코드 클론 탐지는 프로그램 유지보수와 저작권 감정 등 다양한 분야에서 여러 가지 목적으로 사용되고 있다. 본 논문에서는 소스코드 감정평가 목적으로 한국저작권위원회에서 사용 중인 코드클론 탐지 프로그램 exEyes의 재현율 성능을 평가하였다. 벨론 레퍼런스 코퍼스 기준으로 exEyes의 클론 타입 별 재현율을 계산했다.

총 4개의 오픈소스(C언어 기반 2개, Java 기반 2개, 10,055개 클론)를 대상으로 성능평가를 해본 결과, 타입 1 클론은 100% 완벽하게 재현한 것으로 나타났으나, 타입 2는 63%, 타입 3은 34% 재현하는 선에 그쳐, 전체 평균 62%의 재현율을 보여주었다.

exEyes가 놓친 미탐 5,593개는 크게 세 가지 유형으로 나눌 수 있다. 첫째, 토큰을 분류하면서 토큰의 의미를 전혀 고려하지 않아서 발생하는 미탐으로 전제 미탐의 53%를 차지한다. 둘째, 한 줄의 코드가 두 줄 이상의 코드로 줄바꿈이 된 경우로, 전체 미탐의 6%를 차지한다. 셋째, 문장 추가/삭제/변경이 있는 경우로, 전체 미탐 클론의 41%를 차지한다.

exEyes는 줄 단위로 유사토큰의 비율이 일정 비율 이상이 되면 유사하다고 판단하는 방식을 사용하기 때문에 원천적으로 타입 3 클론을 정확하게 찾으리라 기대하는 건 무리이다. 하지만 유형 1과 유형 2의 미탐은 exEyes가 토큰의 의미를 토큰의 유사성 판정에 고려하고, 줄 단위로 끊어서 비교하는 방식을 배제하고 유연성을 확보하면 재현율을 충분히 높일 수 있을 것이다.

참 고 문 헌

[1] 한국저작권위원회, http://www.copyright.or.kr, 2015.

[2] 한국저작권위원회, “조정 통계”, 2014.12.

[3] exEyes, http://www.copyright.or.kr, 2015.

[4] Stefan Bellon, et al., “Comparison and Evaluation of Clone Detection Tools”, IEEE Transactions on Software Engineering, 2007.

[5] 이효섭, 도경구, “트리패턴 기반의 코드클론 자동 탐지”, 한양대학교 일반대학원 박사학위논문, 2013.

[6] Koschke R., et al., “Clone Detection Using Abstract Syntax Suffix Tree”, WCRE, 2008.

[7] Roy C., et al, “A survey on software clone detection reseach”, Queen’s school of computing, 2007.

[8] Roy C., et al., “Comparison and Evaluation of Code Clone Detection Techniques ant Tool: A Qualitative Approach”, Journal od Software Maintenance and Evolution, 2009.

저 자 소 개

최성하(Sungha Choi)

2014년 한라대학교 정보통신방송공학부 학사 졸업
2014 ~ 현재 한양대학교 대학원 컴퓨터공학과 석사과정

<주관심분야 : 프로그래밍언어, 소프트웨어 저작권, 코드 클론, 모바일 애플리케이션>

도경구(Kyung-Goo Doh)

1980년 한양대학교 산업공학과 학사 졸업.
1987년 아이오아주립대학교 컴퓨터과학 석사 졸업.
1992년 캔사스주립대학교 컴퓨터과학 박사 졸업
1993년 ~ 1995년 University of Aizu 교수
1995년 ~ 현재 한양대학교 ERICA 컴퓨터공학과 교수

<주관심분야 : 프로그래밍언어, 프로그램분석, 소프트웨어 보안, 소프트웨어 공학>