二进制文件相似性检测技术对比分析

陈 慧[1,2],郭 涛[3],崔宝江[2],王建新[4]
CHEN Hui[1,2], GUO Tao[3], CUI Baojiang[2], WANG Jianxin[4]
1.山东英才学院 计算机学院,济南 250104
2.北京邮电大学 计算机学院,北京 100876
3.中国信息安全测评中心,北京 100085
4.北京林业大学 信息学院,北京 100083
1.School of Computer Science and Technology, Shandong Yingcai University, Jinan 250104, China
2.School of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
3.China Information Technology Security Evaluation Center, Beijing 100085, China
4.School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

CHEN Hui, GUO Tao, CUI Baojiang, et al. Comparison and analysis on binary file similarity detection technique. Computer Engineering and Applications, 2012, 48(33):79-84.

Abstract: The traditional file similarity detection technique is generally based on source code. In the case of source code unavailable, binary comparison technique is proposed for clone detection. Four binary file similarity detection techniques and the main detection tools are summarized and analyzed. Based on the evaluation method of binary file clone comparison, the experiment test has been carried on. This method provides a review of binary file clone types, detection approaches and the similarity calculation standard. Experiments show that for continuous clone, division clone which doesn't affect the call relations, equivalent replacement clone which doesn't affect basic block number and the call relations, the techniques similarity detection with the binary files gets more accurate result than the ones token-based similarity detection with the source code files.

Key words: homologous software; binary file comparison; clone detection

摘 要: 传统的文件相似性检测技术是基于源代码的,针对源代码难以获取的情况,二进制文件比对技术被提出并受到越来越多的关注。总结和分析了四种二进制文件相似性检测技术和主流的检测工具。在提出了二进制文件克隆比对的评价方法的基础上进行了实验测试。该方法针对二进制文件克隆的分类方式,设计了实验流程和相似度的计算标准。结果表明对于连续克隆,不影响调用关系的分割克隆,不影响基本块数量和调用关系的等价替换克隆,采用二进制文件相似性检测比采用基于token的源代码文件相似性检测能得到更准确的检测结果。

关键词: 软件同源性; 二进制文件比对; 克隆检测
文献标识码: A **中图分类号:** TP311 doi:10.3778/j.issn.1002-8331.1112-0242

1 引言

软件相似性检测也称为抄袭检测,或软件剽窃检测,是计算机编程分析非常重要的一部分[1]。基于源代码的相似性检测技术已经比较成熟,被用于检测软件剽窃、软件侵权,以及软件重构,检测错误等方面。但某些情况下无法得到源代码,如大部分商

基金项目: 国家自然科学基金(No.61170268,No.90818021)。
作者简介: 陈慧(1974—),女,讲师,主要研究领域为信息安全;郭涛(1974—),男,博士,副研究员;崔宝江(1973—),男,博士,副教授;王建新(1972—),男,博士,副教授。 E-mail:chh2@163.com
收稿日期: 2011-12-13 **修回日期:** 2012-04-23 **文章编号:** 1002-8331(2012)33-0079-06
CNKI出版日期:2012-07-05 http://www.cnki.net/kcms/detail/11.2127.TP.20120705.1538.003.html

业软件不会发布源代码,这时使用二进制文件进行相似性检测就显得尤为重要。二进制文件相似性检测技术常用的有四大类,分别是基于二进制字节比较、基于二进制文件反汇编后的文本比较、基于指令相似性的图形化比较和基于结构化签名的比较方法。不同的检测技术针对不同的抄袭手段有何优势,现有的检测工具是否能够满足二进制文件相似性检测的需要,目前还没有相关数据予以证明。因此需要对现有的检测技术进行对比分析,以便未来能够设计出针对多种克隆类型有良好检测效果的二进制文件检测工具。针对二进制文件相似性检测的这种现状,本文提出了适用于二进制文件克隆分类的方式,并根据该分类准则设计了二进制文件相似性检测比对实验流程和相似度计算标准,通过对四款常用的二进制相似性检测工具测试分析,归纳出二进制相似性检测所适用的克隆类型。

2 二进制文件相似性检测比对技术及工具

2.1 二进制文件相似性检测比对技术

二进制字节比较是直接读入二进制数据进行比较。该方法较为直接,但缺少对语义和程序整体逻辑的理解,只适用于若干字节变化的比较。基于汇编指令的比对方法[2]以汇编指令作为分析对象,将二进制文件反汇编后进行对比。可分别提取汇编指令中的操作码和操作数进行比对,提高比对的准确度。该方法也缺乏对程序逻辑的理解,只适用于小文件和少量变化。

基于指令相似性的图形化比较[3]是对指令图形的同形匹配,该方法将整个函数结构进行同构匹配,从可执行文件的入口地址开始分析,对图中的每一条指令进行比较。该算法根据操作码的相似性以及操作数比较来判断两个函数是否发生变化。最后利用比较的结果对控制流图(CFG)进行化简合成,得到最大可能的相似图。该方法能发现非结构化变动,但比对精度会受编译器优化干扰,相似图形的化简也存在不完全的问题。

结构化签名的比较方法[4]是将整个可执行文件视为一个图,函数作为基本逻辑单位。对每个函数分配一个结构化的签名,然后利用签名对函数进行比较,标识函数之间的关系。该方法可忽略指令重排和寄存器重分配,部分抵抗编译器优化的干扰,但不能发现非结构的变化,并且可能存在多个函数签名相同的情况,使部分函数不能进行比较。Dullien等人改进了该方法[5],引进了属性的概念,属性相同的节点才比较结构化签名,从而提高比较效率。

2.2 二进制文件比对工具

伴随着二进制文件相似性检测比对技术的发展,检测工具也逐渐出现,目前流行的二进制相似性检测工具大多是基于指令相似性的图形化比较和基于结构化签名比较这两种技术。

IDACompare[6]是由iDefense发布的一款用VB编写的IDA[7](The Interactive Disassembler)插件。它是基于签名扫描的,用于将两个独立的反汇编文件进行比较,使用mdb数据库文件存储生成的结果,存放基准文件和目标文件的一些特征。该工具适用于分析500 KB左右的文件。

BinDiff[8]是由Flake[4]基于图像和指纹理论所开发的二进制文件比较工具,也是基于IDA的插件。BinDiff的函数匹配基于NEC值,即基本块数(Node)/边数(Edge)/调用数(Call),NEC是从控制流图中得到。

Patchdiff[9]是Tenable Security在发布的用C++编写的工具,它也是一个IDA插件,基于CG(Call Graph)同构比较技术。Patchdiff使用CRC值(Cyclic Redundancy Check)和函数的起始地址比较函数是否相同。

eEye Binary Diffing Suite[10]是Jeong Wook Oh开发的二进制文件比较工具。它包含了逆向工具Binary Diffing Starter(BDS)和代码分析工具DarunGrim。DarunGrim是用python写的,使用sqlite数据库来存储分析结果。该工具的算法经过逐步改进为DarunGrim2[11]的算法。

DarunGrim2是用C++编写的基于指纹哈希对比匹配[12],还采用了同构分析和符号名称匹配作为辅助。DarunGrim2的指纹哈希是采用指令序列作为特征值,使用指令和操作数生成指纹,并忽略内存和立即数,选择性地忽略一些寄存器差异。DarunGrim2用抽象的字节作为基本块的指纹,把生成的指纹串存入哈希表中进行匹配。

国内NCNIPC也发布了一款结构化比对工具NBD(NCNIPC Binary Differ)[13],通过对签名相似程度的强弱进行量化进行函数配对。

3 二进制文件相似性检测比对技术的评价方法

二进制文件是由源代码经过编译后得到,某些抄袭手段经编译后会被消除,不会影响生成的二进制文件,因此源代码克隆分类方式并不完全适用于二进制文件克隆,基于源代码的相似性检查技术的评价方法[14]同样不再适用于评价二进制文件的相似性。目前已经提出的二进制文件相似性检测比对技术和工具,针对哪些克隆类型有效,检测结果是否准确,目前还无定论。针对这一现状,本文提出了一种

二进制文件相似性检测的规范化测试方法，对现有的二进制文件相似性检测比对技术和工具进行评价，以期在未来的研究中能提高二进制比对技术的相似度检测的准确度。

3.1　二进制文件克隆分类

使用二进制文件进行相似度对比，在没能获得源代码的情况下，无法得知目标文件采用什么方式抄袭的基准文件。通过对基于源代码相似性检测技术的研究，可以发现和总结出一些常用的抄袭手段。文献[14]归纳出四类源代码克隆，从I型克隆到IV型克隆采用的抄袭手段逐渐复杂，进行源代码分析和检测时的难度也逐步增加。其中对于源代码中的I型克隆，目标文件与基准文件的代码完全相同，使用了增加删除空格，改变排版，增加删除修改注释的这类比较简单的抄袭手段。采用二进制文件比对的方式，目标文件经过编译后，I型克隆所做的修改都被消除，得到二进制代码与编译后的基准文件完全相同。对于这种克隆方式在进行二进制文件克隆分类时就不再列入。

为了能对二进制克隆检测正确评价，并构造合适的实验用例，本文把不同的抄袭手段重新分类，按照变换的复杂度将二进制文件克隆分为三类：

（1）连续克隆：克隆后仍保持代码段的语句连续且顺序不变。通常采用的抄袭手段：常量/变量/参数/函数重命名；常量替换；改变数据类型。

（2）分割克隆：代码段的语句顺序改变，重新调整或分割。通常采用的抄袭手段：增加冗余的变量或语句；代码块重排序；代码块内语句重排序。

（3）等价替换克隆：代码段使用不同的句法变体执行相同的功能。通常采用的抄袭手段：改变表达式中的操作符或者操作数顺序；表达式拆分；等价控制结构替换。

3.2　评价方法流程设计

二进制文件相似性比对与源代码比对的方式略有差别，源码比对可以直接用工具检测源代码文件，通过人工分析源代码发现采用的克隆方式。而二进制采用的克隆方式无法人工分析，如果直接用二进制文件检测相似度，无法得知不同克隆方式下二进制文件比对的效果。因此设计了如下步骤分析二进制文件相似度：

步骤1　根据第2章的二进制文件克隆分类构建测试用例。首先设计基准文件源代码，然后根据三种克隆分类设计了9类目标文件源代码。这9类采用顺序编号，对应表1到表4中的抄袭手段。

（1）连续克隆的目标文件2个，采用的抄袭手段有：

①变量声明位置和名称改变，函数名改变，常量替换，改变数组的长度（增加），改变自定义数据类型名字。

②改变数据类型，将int型改为char型。

（2）分割克隆的目标文件3个，采用的抄袭手段有：

③增加冗余的变量，增加冗余语句，增加函数中的冗余参数，增加自定义结构体的无用的成员变量。

④增加空函数，并在程序中调用该函数。

⑤改变头文件顺序，提前函数声明后改变函数的位置，简单改变无逻辑关系的语句顺序，改变函数参数顺序，原函数内的变量声明改为函数外。

（3）等价替换克隆的目标文件4个，采用的抄袭手段有：

⑥改变表达式中的关系运算符，如B>=A改为A<=B；等价替换算数运算表达式，如将C+=1，C++和C=C+1互换。

⑦if语句的判断条件改变后，if和else对换。

⑧把条件判断表达式替换成if语句。

⑨while语句改为for语句。

步骤2　将源代码形式的基准文件和目标文件用相同的编译优化选项，生成二进制文件。

步骤3　用二进制文件比对工具检测基准文件和目标文件，得到基础数据。

步骤4　采用统一的文件相似度计算标准（3.3节式（1））对步骤3的基础数据进行处理，得到最终的相似度值。

3.3　相似度计算标准

不同的二进制文件比对工具，结果输出的方式略有不同，文件相似度的计算标准也不一致。为了能合理评测不同的二进制比对工具，需要用统一的评价标准来进行文件相似度计算。目前二进制比对工具都能给出检测文件中的函数是否相似，利用此中间结果作为基础数据，采用统一的文件相似度计算方法，计算出不同的二进制比对工具的文件相似度。

设二进制基准文件 A 中所含的函数集合为 $\{a_1, a_2, \cdots, a_m\}$，目标文件 B 中所含的函数集合为 $\{b_1, b_2, \cdots, b_n\}$。

检测工具 k 检测出基准文件 A 中的 m_k 个函数 $\{a_{l_1}, a_{l_2}, \cdots, a_{l_{m_k}}\}$，检测出目标文件 B 中 n_k 个函数 $\{b_{p_1}, b_{p_2}, \cdots, b_{p_{n_k}}\}$。

测试工具 k 检测出的基准文件 A 与目标文件 B 中相似的函数集合为：

$$C_k=\{(a_i,b_j)|i=l_1,l_2,\cdots,l_{m_k},j=p_1,p_2,\cdots,p_{n_k},\ a_i\text{相似于}b_j\}$$

$S_k(A,B)$ 表示测试工具 k 检测出的二进制基准文件 A 和目标文件 B 的相似度：

$$S_k(A,B)=\frac{|C_k|}{m_k \vee n_k} \quad (1)$$

相似度 S_k 满足 $S_k(A,B)=S_k(B,A)$。

4 实验结果及分析

根据2.2节对二进制文件比对工具的介绍，选择有代表性的IDACompare，Sabre BinDiff 2 Evaluation，PatchDiff，DarunGrim2进行比对测试。在下面的实验中按照第3.2节的评价流程进行测试。在构造测试用例时，使用的基准文件源代码相同，按照评价流程的步骤1设计了9类目标文件源代码。为了消除编译优化的干扰，编译时选择缺省的编译优化选项，生成可执行文件。用选择的四种二进制文件比对工具检测基准文件和目标文件，得到文件中的函数个数，以及对应的函数是否相似。最后采用统一的文件相似度计算标准，计算出四种二进制比对工具的文件相似度。为了能对比同样状态下源代码比对工具的效率，选择基于token比对的源代码检测工具CCFinder[15]对编译之前的源代码测试用例进行对比实验。

4.1 连续克隆检测结果及分析

连续克隆不会改变代码段顺序，对文件进行修改也比较简单。如表1所示，采用抄袭手段①，改变标识符名等方式不会对程序的执行效果产生任何影响，编译系统对源程序中定义的每一个标识符都分配相应的内存，并建立对应的地址表。因此对这种方式，四种二进制检测工具都检测出相似度为100%。使用源代码检测工具CCFinder检测得到的相似度为80%。

抄袭手段②，改变数据类型，将int型改为char型，由于int是无符号的，而char是有符号的，所以编译后的汇编指令不同。如语句while(noteof)，对应的汇编指令如下。

基准文件中对应的汇编指令：

```
cmp  noteof,0
jz  short  loc_401793
```

目标文件中对应的汇编指令：

```
movsx  ecx,noteof
test  ecx,ecx
jz  short  loc_401785
```

两个代码块的语义是相同，完成同样的功能，但用的指令不同，如果比对工具使用汇编指令生成比对因子将不能正确检测出相似度。源代码中只有一条语句不同，所以CCFinder检测得到的相似度为99.9%。

4.2 分割克隆检测结果及分析

分割克隆代码段的语句顺序改变，重新调整或分割，会采用一些改变语句结构的克隆手段，将分割克隆分为三种不同的情况进行比对测试，如表2所示。采用抄袭手段③，会改变函数中的指令数。因为本次实验对文件编译时采用缺省编译优化选项，冗余的变量、语句、参数编译后仍然存在，所以检测结果类似抄袭手段②，只影响用汇编指令生成比对因子的检测工具。冗余对源代码检测影响较大，工具CCFinder检测得到的相似度为86.6%。

采用抄袭手段④，增加5个冗余函数，函数不执行任何操作，并被调用。这种方式不仅会影响到指令，而且会影响文件中函数的数量和函数内的调用数，四种二进制检测工具都受到影响。CCFinder检测得到的相似度为88.1%。

采用抄袭手段⑤改变函数的位置，改变函数参数顺序，不会影响二进制相似度检测，但改变无逻辑关系的语句顺序，会影响基本块内的指令顺序。如

表1 连续克隆检测结果

工具名称	IDACompare		BinDiff		PatchDiff		DarunGrim2	
抄袭手段	①	②	①	②	①	②	①	②
基准文件函数个数	140	140	232	232	183	183	229	229
目标文件函数个数	140	140	232	232	183	183	229	229
相同函数个数	140	140	232	232	183	183	229	227
相似度/(%)	100	100	100	100	100	100	100	99.1

表2 分割克隆检测结果

工具名称	IDACompare			BinDiff			PatchDiff			DarunGrim2		
抄袭手段	③	④	⑤	③	④	⑤	③	④	⑤	③	④	⑤
基准文件函数个数	140	140	140	232	232	232	183	183	183	229	229	229
目标文件函数个数	140	145	140	232	242	232	183	193	183	229	229	229
相同函数个数	140	140	140	232	222	232	183	183	183	225	224	226
相似度/(%)	100	96.6	100	100	91.7	100	100	94.8	100	98.3	97.8	98.7

表3　等价替换克隆检测结果1

工具名称	IDACompare		BinDiff		PatchDiff		DarunGrim2	
抄袭手段	⑥	⑦	⑥	⑦	⑥	⑦	⑥	⑦
基准文件函数个数	140	140	232	232	183	183	229	229
目标文件函数个数	140	140	232	232	183	183	229	229
相同函数个数	140	139	232	230	183	181	229	228
相似度/(%)	100	99.3	100	99.1	100	98.9	100	99.6

表4　等价替换克隆检测结果2

工具名称	IDACompare		BinDiff		PatchDiff		DarunGrim2	
抄袭手段	⑧	⑨	⑧	⑨	⑧	⑨	⑧	⑨
基准文件函数个数	140	140	232	232	183	183	229	229
目标文件函数个数	140	140	232	232	183	183	229	229
相同函数个数	140	140	228	232	179	181	227	229
相似度/(%)	100	100	98.3	100	97.8	98.9	99.1	100

表5　二进制文件比对检测工具

工具名称	IDACompare	BinDiff	PatchDiff	DarunGrim2
检测方法	签名扫描	图像和指纹	CG同构比较	指纹哈希
比较级别	函数	基本块，函数	基本块，函数	基本块，函数
比对因子	CRC　Call/Push API　Profile　ESP	NEC(node/edge/call)	CRC Address	Fingerprint tree　name

果根据基本块内的指令生成比对因子，则相似度检测会受影响。改变函数参数顺序，改变语句顺序都会干扰源代码检测，工具CCFinder检测得到的相似度为72.6%。

4.3　等价替换克隆检测结果及分析

等价替换克隆，克隆后的文件功能相同或相似，从源码的角度看变化比较复杂。将等价替换克隆分为四种情况进行测试。首先采用克隆手段⑥，编译后的二进制文件完全相同，从表3中可以看出四种检测工具都能很好检测出相似度为100%。CCFinder检测得到的相似度为85.3%。

采用克隆手段⑦if和else对换，实验中采用两种变换方式。一种是简单改变if语句的表达式后，if和else所包含的语句完整对换，四种检测工具仍能检出这部分的相似度为100%。当采用复杂的变换，改变if语句的表达式后，合并某些语句，并减少返回指令，四种工具都检出有函数改变。源代码检测工具CCFinder检测得到的相似度为67.1%。

如表4所示，采用克隆手段⑧把条件判断表达式替换成if语句，所涉及的函数中的基本块数量、基本块之间的关系，以及基本块内的指令都会改变。检测基本块的工具都会受到干扰，判断函数发生改变。使用源代码检测工具CCFinder检测得到的相似度为86.9%。

采用克隆手段⑨while语句改为for语句，所涉及的函数中基本块的个数和基本块之间的关系不变，但基本块内的指令数量不同，块内跳转指令不同。不能对跳转指令做噪声处理的工具就会受到干扰。使用源代码检测工具CCFinder检测得到的相似度为88.9%。

4.4　二进制文件相似性检测工具对比

由于实验用例中标准库头文件的函数占一定比例，这些函数编译后在基准文件和目标文件中都相同，使得二进制比对的相似度比值偏高。因此在评价检测效果时，侧重考核相似度值为100%的完全检出的情况。在表1到表4中，对于抄袭手段①和⑥，四种二进制比对工具检测到的相似度值均为100%。对于抄袭手段②、③、⑤、⑨，有三种二进制比对工具检测到的相似度比值为100%。同样状态下，使用基于token的源代码比对工具CCFinder对编译之前的测试用例进行检测，得到的相似度值低于二进制文件比对检测的相似度值。

二进制文件检测工具采用的检测方法及比对因子不同如表5所示，其检测效果也各不相同，如表6针对不同的克隆分类，为消除标准库头文件的影响，只考虑比对结果相似度值为100%的完全检出的情况。工具IDACompare统计每个函数的长度Length，调用数Calls，跳转数Jmps，栈帧大小ESP，校验值CRC，常数Consts，API调用。匹配时有首先尝试精确CRC匹配，不成功再用相同API调用匹配，或ESP匹配，或Call/Push匹配。IDACompare用的比对因子最多，采用多种匹配方式，检测效果较好，但由于采

用mdb文件存储,不适于检测大文件。BinDiff的函数匹配基于NEC值,即基本块数(Node)/边数(Edge)/调用数(Call),个别指令的细微变化不会影响NEC值。Patchdiff首先使用CRC值匹配,若不成功再用函数的起始地址比较,个别指令的变化会影响CRC值。DarunGrim2使用指令和操作数生成指纹,指令变化对其影响较大,比较敏感,相似性比对方面表现一般。

表6 二进制文件比对检测效果比较

工具名称	IDACompare	BinDiff	PatchDiff	DarunGrim2
连续克隆	2/2	2/2	2/2	1/2
分割克隆	2/3	2/3	2/3	0/3
等价替换克隆	3/4	2/4	1/4	2/4

注:表中比值是对应三种克隆分类,相似度100%的实验用例/总实验用例。

5 结论

二进制文件相似性检测是近年来软件相似性度量的一个重要方向。为了能有效评价二进制文件相似性检测比对技术的适用范围及检测效果,本文提出了二进制文件克隆相似性检测的评价方法,该方法包括适用于二进制文件的克隆分类方式、二进制文件相似性检测实验流程和相似度计算标准三部分。采用此方法评价了四种二进制文件相似性检测比对工具,结果表明对于连续克隆,不影响调用关系的分割克隆,不影响基本块数量和调用关系的等价替换克隆,采用二进制文件相似性检测较采用基于token的源代码文件相似性检测能得到更准确的检测结果。因此二进制文件相似性检测可以作为源代码检测的很好的补充。如何选择有代表性的比对因子,提高改变函数调用关系的分割克隆,改变基本块数量和基本块间关系的等价替换克隆的检测准确率,降低误检率和漏检率是未来值得关注的研究方向。

参考文献:

[1] Cui Baojiang,Li Jiansong,Guo Tao,et al.Code comparison system based on abstract syntax tree[C]//Proceedings-2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology,Beijing,China,2010:668-673.

[2] Schulman A.Finding binary clones with opstrings and function digests[J].Dr.Dobb's Journal,2005,30(9):64-70.

[3] Sabin T.Comparing binaries with graph isomorphism[EB/OL].(2004)[2011-05].http://razor.bindview.com/publish/papers/comparing-binaries.html.

[4] Flake H.Structural comparison of executable objects[C]//Detection of Intrusions and Malware & Vulnerability Assessment,Dortmund,Germany,2004:161-173.

[5] Dullien T,Rolles R.Graph-based comparison of executable objects[EB/OL].(2005)[2011-05].http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.5076&rep=rep1&type=pdf.

[6] Zimmer D.IDACompare[EB/OL].(2006)[2011-05].http://labs.idefense.com/software/download/?downloadID=17.

[7] DataRescue.IDA pro disassembler[EB/OL].[2011-05].http://www.hex-rays.com/idapro/.

[8] Zynamics.Sabre BinDiff[EB/OL].[2011-05].http://www.zynamics.com/bindiff.html.

[9] Pouvesle N.Patchdiff2[EB/OL].(2010)[2011-05].http://code.google.com/p/patchdiff2/.

[10] eEye binary diffing suite[EB/OL].(2009)[2011-05].http://www.eeye.com/Resources/Security-Center/Research/Tools/eEye-Binary-Diffing-Suite-EBDS.

[11] Oh J.DarunGrim2[EB/OL].(2010)[2011-05].http://www.softpedia.com/progDownload/DarunGrim2-Download-151207.html.

[12] Oh J.Fight against 1-day exploits:diffing binaries vs anti-diffing binaries[EB/OL].(2009)[2011-05].http://nchovy.kr/uploads/3/476/BHUSA09-Oh-DiffingBinaries-SLIDES.pdf.

[13] 宋杨,张玉清.结构化比对算法研究及软件实现[J].中国科学院研究生院学报,2009,26(4):549-554.

[14] Roy C,Cordy J,Koschkeb R.Comparison and evaluation of code clone detection techniques and tools:a qualitative approach[J].Science of Computer Programming,2009,74(7):470-495.

[15] Kamiya T,Kusumoto S,Inoue K.CCFinder:a multilinguistic token-based code clone detection system for large scale source code[J].IEEE Transactions on Software Engineering,2002,28(7):654-670.