CODE CLONE DETECTION USING STRING BASED TREE MATCHING TECHNIQUE

NORFARADILLA BINTI WAHID

UNIVERSITI TEKNOLOGI MALAYSIA

ABSTRAK

Pengklonan kod telah menjadi suatu isu sejak beberapa tahun kebelakangan ini selari dengan pertambahan jumlah aplikasi web dan perisian berdiri sendiri pada hari ini. Pengklonan memberi kesan yang sangat besar kepada fasa penyelenggaran sistem kerana secara tidak langsung peningkatan bilangan pengulangan kod yang sama di dalam sesebuah sistem akan menyebabkan kompleksiti sistem turut meningkat. Terdapat banyak teknik pengesanan klon telah dihasilkan pada hari ini dan secara umumnya ianya boleh dikategorikan kepada pengesanan berasaskan jujukan perkataan, token, pepohon dan semantik. Tujuan projek ini adalah untuk mengetahui kemungkinan untuk menggunakan suatu teknik dari pemetaan ontologi untuk menyelesaikan masalah ini, tetapi kami tidak menggunakan ontologi di dalam pengesanan klon. Telah dibuktikan di dalam eksperimen awalan bahawa ia mampu untuk mengesan klon. Di dalam tesis ini kami menggunakan dua aras pengesanan. Aras pertama menggunakan ’pelombong sub-pepohon terkerap’ di mana ia mampu mengesan sub-pepohon yang sama antara fail yang berbeza. Kemudian sub-pepohon yang sama dinyatakan dalam bentuk ayat dan persamaan antara kedua-duanya dikira menggunakan ‘metrik ayat’. Daripada eksperimen, kami mendapati bahawa sistem kami adalah tidak berganting kepada sebarang bahasa dah menghasilkan keputusan yang bagus dari segi *precision* tetapi tidak dari segi *recall*. Ia mampu mengesan klon serupa dan yang hamper sama.

ABSTRACT

Code cloning have been an issue in these few years as the number of available web application and stand alone software increase nowadays. The major consequences of cloning is that it would risk the maintenance process as there are many duplicated codes in the systems that practically increase the complexity of the system. There are many code clone detection techniques that can be found nowadays which generally can be group into string based, token based, tree based and semantic based. The aim of this project is to find out the possibility of using a technique of ontology mapping technique to solve the problem, but we are not using the real ontology for the clone detection. It has been prove that there is the possibility as it manages to detect clone code. In this thesis the clone detection is using two layers of detection; i.e. structural similarity and string based similarity. The structural similarity is by using subgraph miner where it capable to get the similar subtree between different files. And then we extract all elements of that particular subtree and treat the elements as a string. Two strings from different files then applied with similarity metric to know whether it is a clone pair. From the experimental result, we found that the system is language independent but the result is good in precision but not so good recall. It is also capable to detect two main types of clone, i.e identical clones and similar clones.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Overview

As the world of computers is rapidly developing, there are tremendous needs of software development for different purposes. And as we can see today, the complexity of the software been developed are different between one and another. Sometimes, developers take easier way of implementation by copying some fragments of the existing programs and use the code in their work. This kind of work can be called as code cloning. Somehow the attitude of cloning can lead to the other issues of software development, for example the plagiarism and software copyright infringement (Roy and Cordy, 2007).

In most of the cases, in order to figure out the issues and to help better software maintenance, we need to detect the codes that have been cloned (Baker, 1995). In the web applications development, the chances of doing clones are bigger since there are too many open source software available in the Internet (Bailey and Burd, 2005). The applications are sometimes just a ‘cosmetic’ of another existing system. There are quite a number of researches in software code cloning detection, but not so particularly in the area of web based applications.

1.2 Background of the Problem

Software maintenance has been widely accepted as the most costly phase of a software lifecycle, with figures as high as 80% of the total development cost being reported (Baker, 1995). As cloning is one of the contributors towards this cost, the software clone detection and resolution has got considerable attention from the software engineering research community and many clone detection tools and techniques have been developed (Baker, 1995). However, when it comes to commercialization of the software codes, most of the software house developers tend to claim that their works are 100% done in house without using other codes copies form various sources. This has made a difficulty for the intellectual property copyright entities such as SIRIM and patent searching offices in finding the genuineity of the software source codes developed by the in house company. There is a need to identify the software source submitted for patent copyright application to be a genuine source code without having any copyright infringements. Besides that, the cloning is somehow raising the issue of plagiarism. The simplest example is in the academic area where students tend to copy their friends' works and submit the assignments with only slight modifications.

Usually, in software development process, there is a need for components reusability either in designing and coding. Reuse in object-oriented systems is made possible through different mechanisms such as inheritance, shared libraries, object composition, and so on. Still, programmers often need to reuse components which have not been designed for reuse. This may happen during the initial of systems development and also when the software systems go through the expansion phase and new requirements have to be satisfied. In these situations, the programmers usually follow the low cost copy-paste technique, instead of costly redesigning-the-system approach, hence causing clones. This type of code cloning is the most basic and widely used approach towards software reuse. Several studies suggest that as much as 20-30% of large software systems consist of cloned code (Krinke, 2001). The problem with code cloning is that errors in the original must be fixed in every copy. Other kinds of maintenance changes, for instance, extensions or

adaptations, must be applied multiple times, too. Yet, it is usually not documented where code was copied. In such cases, one needs to detect them. For large systems, detection is feasible only by automatic techniques. Consequently, several techniques have been proposed to detect clones automatically (Bellon et al., 2007).

There are quite a number of works that detect the similarity by representing the code in tree or graph representation and also some using string-based detection, and semantic-based detection. Almost all the clone detection technique had the tendency of detecting syntactic similarity and only some detect the semantic part of the clones. Baxter in his work (Baxter et al., 1998) proposes a technique to extract clone pairs of statements, declarations, or sequences of them from C source files. The tool parses source code to build an abstract syntax tree (AST) and compares its subtrees by characterization metrics (hash functions). The parser needs a “full-fledged” syntax analysis for C to build AST. Baxter's tool expands C macros (define, include, etc) to compare code portions written with macros. Its computation complexity is *O(n)*, where n is the number of the subtree of the source files. The hash function enables one to do parameterized matching, to detect gapped clones, and to identify clones of code portions in which some statements are reordered. In AST approaches, it is able to transform the source tree to a regular form as we do in the transformation rules. However, the AST based transformation is generally expensive since it requires full syntax analysis and transformation.

In other work (Jiang et al, 2007) present an efficient algorithm for identifying similar subtrees and apply it to tree representations of source code. Their algorithm is based on a novel characterization of subtrees with numerical vectors in the Euclidean space R^n and an efficient algorithm to cluster these vectors with respected to the Euclidean distance metric. Subtrees with vectors in one cluster are considered similar. They have implemented the tree similarity algorithm as a clone detection tool called DECKARD and evaluated it on large code bases written in C and Java including the Linux kernel and JDK. The experiments show that DECKARD is both scalable and accurate. It is also language independent, applicable to any language with a formally specified grammar.

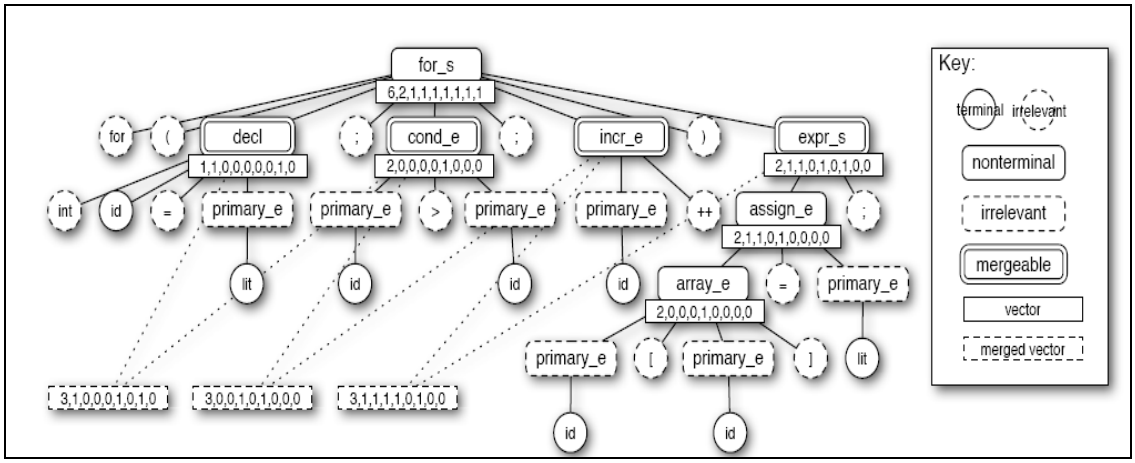

Figure 1.1: A sample parse tree with generated characteristic vectors[14].

In (Krinke, 2001), Krinke presents an approach to identify similar code in programs based on finding similar subgraphs in attributed directed graphs. This approach is used on program dependence graphs and therefore considers not only the syntactic structure of programs but also the data flow within (as an abstraction of the semantics). As a result, it is said that no tradeoff between precision and recall- the approach is very good in both.

Kamiya in one of his work in (Kamiya et al., 2002) suggest the use of suffix tree. In the paper they have used a suffix-tree matching algorithm to compute token-by token matching, in which the clone location information is represented as a tree with sharing nodes for leading identical subsequences and the clone detection is performed by searching the leading nodes on the tree. Their token-by token matching is more expensive than line-by-line matching in terms of computing complexity since a single line is usually composed of several tokens. They proposed several optimization techniques especially designed for the token-by-token matching algorithm, which enable the algorithm to be practically useful for large software.

Appendix B of this thesis, describe briefly some existing techniques of code clone detection and plagiarism. It also discusses the strength and weaknesses of each technique.

1.3 Problem Statement

As we can see from the previous works, some of the works are scalable, can detect more than one type of clone. But some of them face the trade off of the computational complexity. It may be happen because most of the techniques apply expensive syntax analysis for transformation. From the literature that have been done, more than half of existing techniques used tree- based detection as it were more scalable. But, most of the techniques do a single layer detection which means after the transformation into normalized data e.g. tree, graph, and etc, the process of finding the similarity of code, i.e. code clone, were done directly by processing each nodes in the data. All possible clones need to be search directly without some kind of filtering, which it can cause higher cost of computational process.

As ontology has been widely used nowadays, we cannot deny the importance of ontology in current web technology. The major similarity of ontology and clone detection works is that it both can be represented as tree. Beside that, there are many works have been done to do mapping of different ontologies between each other, which is actually to find out which concepts of the first ontology are the same with the second one. This activity is actually almost the same with what need to be done in detecting clone codes.

Since there are some kinds of similarity between both problems, so detecting clone in source code may be able to be done using the same way as mapping the ontologies. The research question of this thesis is *to identify the possibility of using a technique of ontology mapping to detect clones in a web- based application*. Obviously there will be no ontologies that going to be used in the experiments since we are dealing with source code and not ontology. But we will use the technique of mapping to detect clones.

In order to achieve the aim, there are a few questions that need to be solved. What are the attributes or criteria that might be possible to be cloned in web documents? What are the approaches that had been proposed in the previous research in the ontology mapping area than had been used in clone detection tool? What are the issues of the recovered approach and how to solve it?

1.4 Objectives of the Project

The aim of this research is to develop a clone detection framework by manipulating an existing work of mapping ontology. In order to achieve this aim, the following objectives must be fulfilled.

1. To analyze various techniques related to code clone detection that has been proposed by previous researches.

2. To develop a clone detection program by using the ontology mapping technique that will be proposed in the project.

3. To test the program using recall and precision measurements as the main metrics.