A Mutation Operator-Based Scenario for Evaluating Software Clone Detection Tools and Techniques

Pratiksha Gautam, Jaypee University of Information Technology, Waknaghat, India
Hemraj Saini, Jaypee University of Information Technology, Waknaghat, India

ABSTRACT
Over the past few years, several software clone detection tools and techniques have been introduced by numerous researchers. The software clone detection techniques and tools are based on their numerous attributes and sub-attributes which make them difficult to complete a comparative study. Therefore, the authors propose a mutation operator-based editing taxonomy for generating different software clone types. In addition, a hypothetical scenario is developed using mutation operator-based editing taxonomy and this hypothetical scenario is used to evaluate various software clone detection techniques and tools. Further, the existing evaluation criterion is extended by the hypothetical scenario which is clearly represented by the analysis of results.

KEYWORDS
Clone Attributes, Clone Code Detection Techniques, Clone Detection, Hypothetical Scenario, Mutation Operator

INTRODUCTION
Copying a source code segment possibly with or without minor alterations is a typical activity when it comes to software development. As a result, the copied code is called software clone and the process is known as cloning. Software clones introduce bugs in software system and increase maintenance efforts. Several scientists (Gautam and Saini, 2016; Baker, 1995; Ducasse et al., 1999; Mayrand et al., 1996; Kontogiannis et al., 1996; Lague et al., 1997) have reported more than 20-59% code is copied code due to replication. Further, if a duplicated fragment is detected in one section of the code then it should be checked for the same bug throughout the software. Auspiciously, various clone detection techniques and tools have been developed and a number of comparisons and evaluation studies have been done which are related to them. Presently, Rattan et al. (Rattan et al., 2013) presented an extensive survey on software clones while (Roy et al., 2009) provided a correlation and assessment of software clone detection tools and techniques based on their qualities. A hypothetical editing scenario-based evaluation of clone detection techniques is provided by (Roy and Cordy, 2009). (Bellon et al., 2007) has provided six clone detector tools evaluation for clone detection which are mostly based on java and C. In addition, a number of potential studies have evaluated clone detection tools and techniques using precision, recall, portability, computational complexity (Sheneamer and Kalita, 2016).

DOI: 10.4018/IJISP.2019010103

Copyright © 2019, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
This paper provides a comprehensive illustration of software clone detection techniques and tools which are available presently using attribute and mutation operator-based editing hypothetical scenario for distinct software clone types. The existing studies focuses on empirically evaluated tools, we intend to point out the critical potency and limitations of individual tools and techniques on the basis of their attributes and proposed mutation operator-based editing hypothetical scenarios. Our aim is to present a complete catalogue of available techniques and it’s prospective to identify “factual software clone.” Additionally, we generate a mutation operator based editing taxonomy which is used to create different software clone types. Further, these scenarios are used to compare various software clone detection approaches and tools.

While considering the editing scenarios as the basis, we can clearly analyze that our work has a large variety of different aspects when compared with previous surveys. Thus, our main aim is to provide a successful evaluation which is sufficient, practical and future proof as we want an evaluation which has a great amount of future potential and not just being an implementation tool for the present. The rest of the paper is structured as follows. We begin with the introduction of software clones and classifying the clone detection methods in Section 2. The section 3 presents, attribute-based comparisons of software clone detection techniques and tools. A mutation operator-based hypothetical scenario is provided in Section 4. At last, Section 5 summarizes the paper.

AN OUTLINE OF SOFTWARE CLONE DETECTION TECHNIQUES

Software clone detection techniques can be classified into four categories on the basis of analysis. Different types of software clone detection techniques are as follows-

Text-based Techniques

In this approach, the little normalization/transformation uses on source code before the concrete comparisons and source segments are analyzed as strings of source text. Usually, the raw source text is used directly in software clone detection scheme (Gautam and Saini, 2016; Roy and Cordy, 2007). One of the foremost textual-based approach based on dynamic pattern matching [DPM] was proposed by (Ducasse et al., 1999). Further, (Wettle and Marinesu, 2005), also used similar approach (DPM) which compares all the lines in textually that has been normalized to ignore comments and white spaces. (Johnson, 1994), presented a redundancy finder which was established on “fingerprints” on substrings of the source text. The fingerprints- based approach was also used by (Manber, 1994) which is based on subsequences and it is identified by keywords. (Marcus and Maletic, 2007) a proposed latent semantic indexing (LSI) approach which is used for identifying similar source code fragments in the program text. (Lee and Jeong, 2005) proposed another text-based approach as SDD and claimed that it was suitable for larger system. Although, there are some textual approach-based tools which have been developed for envisioning software clone similarity. (Church and Helman, 1993) developed a tool named as Dotplot which visualizes the identical code fragments by tokenizing the code into lines.

Lexical-based Techniques

In this method, the whole software code is divided into a series of token using lexical analysis(compiler method) and then all the subsequence is scanned for duplicate code (Gautam and Saini, 2016; Roy and Cordy, 2007). One of the primary lexical approach-based tools was proposed by Kamiya et al. (2002) named as CCFinder. D-CCFinder was based on a distributed implementation of CCFinder which permits the method to scale the large software system proposed by (Livieri et al., 2007). Ueda et al., 2002) presented a tool named as Gemini which was based on CCFinder to envision near-miss code clones using scatter-plots. A similar approach used in Dup (Baker, 1995) uses related approach for searching parameterized and exact clones using suffix tree. (Basit et al., 2007) used memory-efficient suffix array instead of suffix tree in their proposed tool named as RTF and assigned the user to adapt tokenization for improved clone detection. (Li et al., 2006) proposed another tool named as...
Related Content

Network and Data Transfer Security Management in Higher Educational Institutions
www.igi-global.com/chapter/network-and-data-transfer-security-management-in-higher-educational-institutions/224861?camid=4v1a
Multimedia Copyright Protection Scheme Based on the Direct Feature-Based Method
www.igi-global.com/chapter/multimedia-copyright-protection-scheme-based/70299?camid=4v1a

PCI Compliance: Overcoming the Challenges
www.igi-global.com/article/pci-compliance-overcoming-challenges/34058?camid=4v1a

Linux Essentials Before We Start
(2019). Constructing an Ethical Hacking Knowledge Base for Threat Awareness and Prevention (pp. 44-71).
www.igi-global.com/chapter/linux-essentials-before-we-start/218414?camid=4v1a