IJDCST @Oct-Nov-2015, Issue- V-3, I-7, SW-12
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Dupian: Duplicate Code Analyzer using Ontology Editor

Rayavarapu Veeranjaneyulu[1], Singaraju Srinivasulu[2]
[1] **Professor, Department of IT, Pace Institute of Technology and Sciences, Ongole.**
[2]**Assoc. prof, Department of IT, Pace Institute of Technology and Sciences, Ongole.**

Abstract: Plagiarism, copying, rewriting is basic things to modify the text or content according to the detected content. Likewise, now a day's Duplicate code becomes the major issue to find out and remove the code from the copied code. Existing, there are so many duplicate or cloning code detectors are available in the real world but the accuracy of detecting code is very low. In this paper, we proposed dupian which identifies the duplicate code.

We implemented the code processing algorithm and code matching algorithm approach by using Ontology Editor. Proposed system will detect clone from c, c++, JavaScript & HTML languages and results will show the accuracy of finding the duplicate code up to 70%.

Keywords: Dupian, Clone, Ontology.

1. INTRODUCTION:

Code cloning is found to be more serious problem in industrial software. It is observed to have negative impact on software evolution. According to studies on open source and commercial code, 66% of cloned code is modified, i.e. it's not an identical clone [1]. Also, detecting the maximal clone pattern is very challenging as the input is not known in advance. It is similar to looking for all the pairs or tuples of people that match each other in a very large population. Several search and hashing-based solutions have been suggested in the past, but they all lack accuracy and coverage. Several studies show that software with code cloning is more difficult to maintain, then the software without code [3, 4, 5], because the code clone increases maintenance costs [2].

It may adversely affect the software system quality, maintainability and comprehensibility. This paper provides an improved analysis, identification and removal Technique for these code clones and also to develop ontology editor use some components like

- Classes: sets, collections, concepts, classes in programming, types of objects, or kinds of things
- Attributes: aspects, properties, features, characteristics, or parameters that objects (and classes) can have
- Relations: ways in which classes and individuals can be related to one another clones are segments of code that are similar according to some definition of similarity. —Ira Baxter, 2002.

Dupian will scan your source code and attempt to identify methods that are functional duplicates of another method. Once a clone is identified dupian will prompt the user to replace the body of a method with a call to its clone.

In this paper, our proposed implemented the code processing algorithm and code matching algorithm approach is implemented as tool developed in JAVA. This tool efficiently and effectively detects the duplicate code from the various resources and identifies the similarity code. A better and effective code editor is developed for users to use the tool. This paper contains four major sections. Section II describes about the related research work are currently implemented in various domain. Section III describes the implementation of the proposed method. Finally, Section III concludes the paper.

II. RELATED WORK:

While there is an on-going debate as to whether remove clones, there is a consensus about the importance to at least detect them. Clone avoidance during normal development, as described in the previous section, as well as making sure that a change can be made consistently in the presence of clones requires knowing where the clones are. Manual clone detection is infeasible for large systems; hence, automatic support is necessary.

Automated software clone detection is an active field of research [2]. This section summarizes the research in this area. The techniques can be distinguished at the first level in the type of information their analysis is based on and at the second level in the used algorithm. Some of the clone detection tools describe below.

A.1PMD (http://www.PMD.sourceforge.net/)

It scans Java source code and looks for potential problems like duplicate code, possible bugs, and dead code. PMD allows the user to set the metrics thresholds for clone detection and allows setting the number of tokens of duplicated code; we chose to keep the default configuration (25 tokens).

B. Bauhaus (http://www.bauhaus-stuttgart.de/)

It provides support to analyze and recover a system's software architecture; several maintenance tasks are supported as the derivation of different views on the architecture of legacy systems, identification of reusable components, and estimation of change impact.

The Bauhaus module for finding duplicated code looks for three types of clones: portions of identical code, their variation with different variable names and identifiers, and portions of identical code with added or removed statements.

C. Google CodePro Analytix (https://developers.google.com/java-dev tools/codepro/doc/*)*

It is a Java testing tool for Eclipse developers who are concerned about improving software quality. The main features are related to code analysis, metrics computation, JUnit test generation, dependency analysis, and similar code analysis. For clone detection, the tool offers three types of search: (1) code that can possibly be refectory, (2) code that contains possible renaming errors, and (3) just looks similar. We chose the last option to find as many duplicated code occurrences as possible.

D. SIMIAN - SIMILARITY ANALYSER

Simian (Similarity Analyser) acknowledges duplication in Java, C#, C, C++, COBOL, Ruby, JSP, ASP, HTML, XML, Visual Basic, Groovy ASCII text file and even plain content documents. Indeed, simian are often used on any accessible records, for instance, ini documents, arrangement descriptors, you name it.

Particularly on Projects like enterprise resource management, it are often difficult for anyone engineer to keep on regarding all the highlights (classes, methods, properties, and so on.) of the framework.

Simian runs regionally in any .NET 1.1 or higher upheld atmosphere and on any Java five or higher virtual machine, significance Simian is often run on just about any instrumentality and any operating framework you'll look for once. Each the Java and .NET runtimes are incorporated as a element of the dispersion.

Simian are often used as an element of the manufacture methodology amid improvement or as AN aide once re-calculating. Think about Simian a free mix of eyes which will facilitate in raising the character of your product.

Inside minutes, Simian will spare you truly an outsized variety of bucks in time spent acting maintenance, troubleshooting and re-calculating.

Running against a huge supply base, for instance, the complete 390,309 LOC* (1.2 million lines of crude source) in four, 242 documents of the JDK one.5.0_13 supply, distinguished sixty six, 375 copy LOC* in one, 260 records in underneath ten seconds utilizing as meager as 48M of heap**!

* A line of code is any line thought to be crucial. Clear lines, remarks, soon do not tally towards this figure.

** Results could amendment relying upon variables, for instance, equipment, operating framework, making ready selections, and so on.

Envision as an example that a bug is found in a very strategy some place in a very task. The engineer properly composes AN experiment, rolls out the essential code enhancements, guarantees the check passes, weighs the code in and considers the utilization wrapped up!

Correct?

Off-base!

Obscure to the engineer, many weeks previous, a kindred partner found identical little bit of code and understood that it did almost all that they expected to tackle a difficulty they were taking an effort at the time. So that they duplicated the fifteen lines of code into their new system, side some a lot of code to try and do the extra utility duty-bound and weighed within the progressions.

Obviously what they did not understand at the time was that the code they were duplicating had a bug in it! Truly at the time no one knew this. therefore currently the primary bug has been altered but shockingly none of the duplicates were settled in light-weight of the actual fact that no-one knew they even existed.

Duplicating and gluing is not the main route for this to happen. Copy code will likewise crawl into through designers freely actualizing comparable highlights.

Simian gets these and completely different occasions of duplication and might be organized to either signal them as notices or maybe "break the build", guaranteeing that duplicate and projected nevermore causes you or your endeavor problems.

III. IMPLEMENTATION OF PROPOSED SYSTEM:

The proposed system is implemented in java which is more user-friendly and gives good result.

A. *Ontology Editor Tool- An Advanced Code Editor:*

According to our proposed system, we designed the ontology editor tool we have to paste the code in the editor for the identifying of clone.

In this paper, we have implemented the code processing algorithm. It will check the code and identify the programming language and process the line by line code.

Algorithm 1:

```
Step 1: Start
Step 2: Declare the variable with number of lines of code.
Step 3: Analyze the Programming code.
Step 4: Interpret all the lines of code.
Step 5: Identify the programming language.
Step 6: Ready for duplicate code matching.
Step 7: Stop
```

Also we have implemented the code matching algorithm. It will check the duplicate code from the various resources and show the accurate result.

Algorithm 2:

```
Step 1: Start
Step 2: Match the no of lines
Step 3: Search the similar code from various resources using word search
Step 4: Detect the matched code
Step 5: Calculate the % of the duplicate code
Step 6: Total no of lines of code-Total no of lines of matched code/ 100
Step 7: Show result
Step 8: Stop
```

B. Words or String Search

Searching process is very important process in similar word search. Here, word search means method search and parameters used in the program. This tool searches the multiple string matching given a program K=k1,k2,k3…..kn and want to search similar set of strings.

$L= l_1,l_2,l_3$…..ln be the total no. of lines of code.

Where $l^i = l^i_1, l^i_2, l^i_{3,\ldots\ldots} l^i_{ni}$, is the length of strings n_i, for i=1,…..n.

Define $|L| = \Sigma^r_{\ i=1} |L^i| = \Sigma^r_{i-1}$ mi and let lmin and lmax denote the minimum and maximum length of any pattern in L, respectively

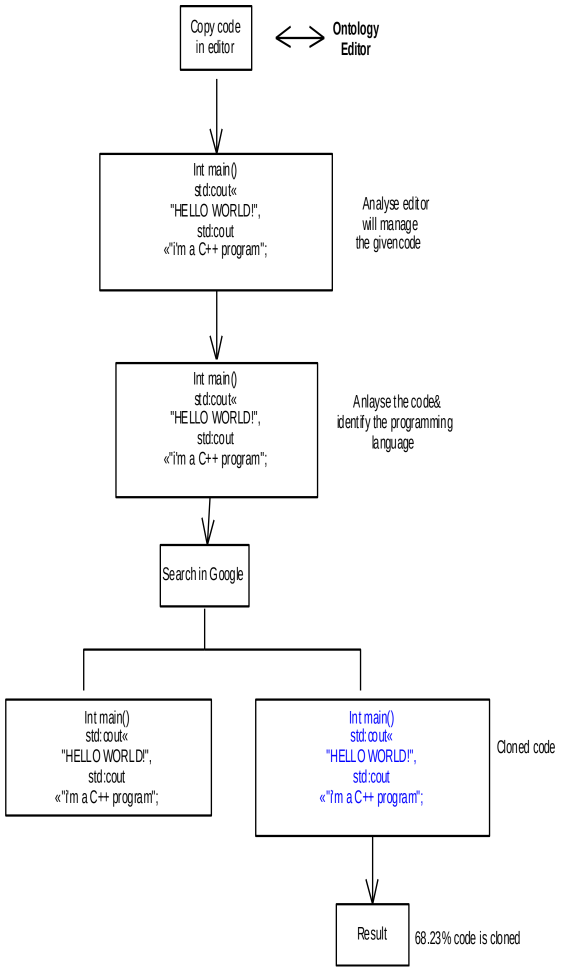

Fig 1: Matching the cloning or duplicate code for C++ using ontology editor

IV. EXPERIMENTAL RESULTS:

Dupien is developed with Netbeans IDE with java code and integrated with ontology editor tool. Installation process as follows: Microsoft Windows (SolidSDD has been tested under Windows XP, Windows Vista, and Windows 7)

Memory: 1GB minimum, 4 GB advised;

Graphics card: OpenGL 1.0 compliant in full-color (RGBA) mode, resolution of 1024 x 768 minimum, 1280 x 1024 or higher advised;

Hard disk space: 100 MB free minimum. The actual amount of free space required is dependent on the size of the analyzed repository and the type of analysis being performed.

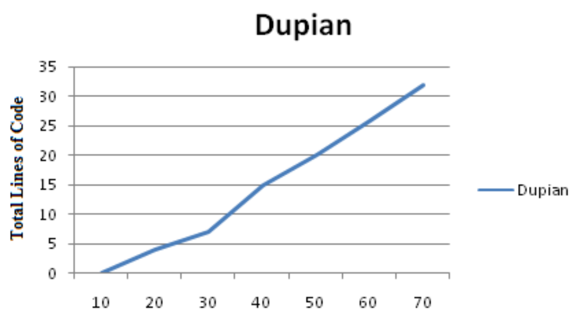

Fig: 2 Performance of Dupian

V. CONCLUSION

The proposed work research on the detection of clone code or duplicate code from various resources using two algorithms that are code processing algorithm and code matching algorithm. After research on various tools of cloning detection codes our approach shows the better detection process compare with other cloning tools.

In this paper, we develop dupian for code editing and checking of the code. It checks the various string matching, method matching and parameters matching using ontology editor. Our tool will show the matching content in terms of percentage (%). Approximately we got 70% of cloning code is analyze and detected by using ontology editor that increase rate of detection is up to 10 % overall accuracy is 70%. In future work, increase the program compatibility of the detection of duplicate code for number of programming languages and develop the advanced compiler for checking all the compatible programming languages.

REFERENCES:

[1] Using redundancies to find errors. In: Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering, ACM Press (2002) 51–60, Xie, Y., Engler, D

[2] Using redundancies to find errors. IEEE Computer Society Transactions on Software Engineering 29(10) (2003) 915–928, Xie, Y., Engler, D.:

[3] Baker, B.S.: On finding duplication and near-duplication in large software systems. InWills, L., Newcomb, P., Chikofsky, E., eds.: SecondWorking Conference on Reverse Engineering, Los Alamitos, California, IEEE Computer Society Press (1995) 86–95

[4] Kontogiannis, K., Mori, R.D., Merlo, E., Galler, M., Bernstein, M.: Pattern matching for clone and concept detection. Automated Software Engineering 3(1/2) (1996) 79–108

[5] Lague, B., Proulx, D., Mayrand, J., Merlo, E., Hudepohl, J.:Assessing the benefits of incorporating function clone detection in a development process. In: International Conference on Software Maintenance. (1997) 314–321

[6] Ducasse, S., Rieger, M., Demeyer, S.: A Language Independent Approach for Detecting Duplicated Code. In: International Conference on Software Maintenance.(1999) 109–118

[7] Walenstein, A., Jyoti, N., Li, J., Yang, Y., Lakhotia, A.: Problems creating task-relevant clone detection reference data. In: Working Conference on Reverse Engineering, IEEE Computer Society Press (2003)

[8] Baker, B.S.: A program for identifying duplicated code. In: Computer Science and Statistics 24: Proceedings of the 24th Symposium on the Interface. (1992) 49–57

[10] Balazinska, M., Merlo, E., Dagenais, M., Lague, B., Kontogiannis, K.: Measuring clone based reengineering opportunities. In: IEEE Symposium on Software Metrics, IEEE Computer Society Press (1999) 292–303

[11] Balazinska, M., Merlo, E., Dagenais, M., Lague, B., Kontogiannis, K.: Partial redesign of java software systems based on clone analysis. In: Working Conference on Reverse Engineering, IEEE Computer Society Press (1999) 326–336

[12] Balazinska, M., Merlo, E., Dagenais, M., Lague, B., Kontogiannis, K.: Advanced clone-analysis to support object-oriented system refactoring. In: Working Conference on Reverse Engineering, IEEE Computer Society Press (2000) 98–107

[13] Kapser, C., Godfrey, M.W.: Toward a taxonomy of clones in source code: A case study. In: Evolution of Large Scale Industrial Software Architectures. (2003)

[14] Kapser, C., Godfrey, M.: A taxonomy of clones in source code: The reengineers most wanted list. In: Working Conference on Reverse Engineering, IEEE Computer Society Press (2003)

[15] Kim, M., Bergman, L., Lau, T., Notkin, D.: An ethnographic study of copy and paste programming practices in OOPL. In: International Symposium on Empirical Software Engineering, IEEE Computer Society Press (2004) 83–92

[16] Bruntink, M., van Deursen, A., Tourwe, T., van Engelen, R.: An evaluation of clone detection techniques for crosscutting concerns. In: International Conference on Software Maintenance. (2004) 200–209

[17] Kapser, C., Godfrey, M.W.:"clones considered harmful" considered harmful. In: Working Conference on Reverse Engineering. (2006)

[18] Nickell, E., Smith, I.: Extreme programming and software clones. In: Working Conference on Reverse Engineering, IEEE Computer Society Press (2003)

[19] Fowler, M.: Refactoring: improving the design of existing code. Addison Wesley (1999)

[20] Monden, A., Nakae, D., Kamiya, T., Sato, S., Matsumoto, K.: Software quality analysis by code clones in industrial legacy software. In: IEEE Symposium on Software Metrics. (2002) 87–94

[21] Li, Z., Lu, S., Myagmar, S., Zhou, Y.: Copy-paste and related bugs in large-scale software code. IEEE Computer Society Transactions on Software Engineering 32(3) (2006) 176–192

[22] Antoniol, G., Casazza, G., Penta, M.D., Merlo, E.: Modeling clones evolution through time series. In: International Conference on Software Maintenance, IEEE Computer Society Press (2001) 273–280