International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 2
ISSN: 2454-4248
392 – 399

The Survey of the Code Clone Detection Techniques and Process with Types (I, II, III and IV)

Gundeep Kaur[1], Er. Sumit Sharma[2]
M.Tech(Scholar), Assistant Professor
Department of Computer Science & Engineering
Chandigarh University, Gharuan, Mohali, Punjab, India-140413
E-mail: kaurgundeep7@gmail.com, cu.sumitsharma@gmail.com

Abstract- In software upgradation code clones are regularly utilized. So, we can contemplate on code location strategies goes past introductory code. In condition of-craftsmanship on clone programming study, we perceived the absence of methodical overview. We clarified the earlier research-in view of deliberate and broad database find and the hole of research for additionally think about. Software support cost is more than outlining cost. Code cloning is useful in several areas like detecting library contents, understanding program, detecting malicious program, etc. and apart from pros several serious impact of code cloning on quality, reusability and continuity of software framework. In this paper, we have discussed the code clone and its evolution and classification of code clone. Code clone is classified into 4 types namely Type I, Type II, III and IV. The exact code as well as copied code is depicted in detail for each type of code clone. Several clone detection techniques such as: Text, token, metric, hybrid based techniques were studied comparatively. Comparison of detection tools such as: clone DR, covet, Duploc, CLAN, etc. based on different techniques used are highlighted and cloning process is also explained. Code clones are identical segment of source code which might be inserted intentionally or unintentionally. Reusing code snippets via copying and pasting with or without minor alterations is general task in software development. But the existence of code clones may reduce the design structure and quality of software like changeability, readability and maintainability and hence increase the continuation charges.

Keywords – *Software Upgrading code clone, Evolution of code clone, classification method and hybrid method.*

*****

I. INTRODUCTION

Software maintenance is last and big-budgeted period of SDLC[1]. The major concern behind product support is the change of current programming framework by including new functionalities, to rectify errors in the product framework or because of the new necessities of the association that are not distinguished amid the prerequisite stage. Yet, the most extreme endeavors arerequired while expanding the current programming by including new functionalities. One of the systems utilized for programming support is Software Re-engineering [2]ois the most utilized .

Software Re-engineering is to examine the current framework and manufacture it again with better functionalities. Software Re-engineering is wide and challenging part approaching recently. Since, while re-designing the current framework the software engineer reuse the code with or without assist changes which can prompt the repeat of code over andover.

In software re-engineering code cloning is finished by reusing code as it is or differently with a few alterations. This procedure is known as Code Cloning. One noteworthy writing study, we portrayed the past procedures and location apparatus in the code clone detection.In writing overview checks different key zones of research on clone programming, depicted the fundamental idea, strategy utilized and recognition instrument.As per Brooks more than ninety level of the product cost relies upon programming support occasions. In the middle of 7 percent up to 23 percent of programming frameworks contains clone code. The principle test of code cloning for programming preserves because it duplicates without cause amplify program-estimate. Since a few upkeep endeavors associated with program-estimate that expand the support exertion. At the point when adjusts to copy source code parts are performed in- reliably, this could decideblunders

1.1 Definition of code clone

Code Cloning characterizes by and large, all through the planning and advancement of programming frameworks. An Ad-hoc structure of re-utilize contains of duplicating and changing a square of past code that plan a bit of required usefulness [3]. Replicated squares are known as clones and duplicating execution, comprising little changes is cloning. Piece duplicating code and after that re-use by sticking without or with little changes/varieties are typical occasions in programming advancement. Sort of re-utilize system of earlier code is known as cloning and the stuck section program is known as a duplicate of thefirst.

Figure illustrates the code clones. The results of several analyses proves that a notable section of 5- 10% of source-code in big software systems is duplicate code. The rationale of code cloning could be intentional or accidental. Escalating the cloning condition is done quickly and irrelevant to surrounding i.e. bug free code becomes incorrect after cloning.

 ISSN: 2454-4248
392 – 399

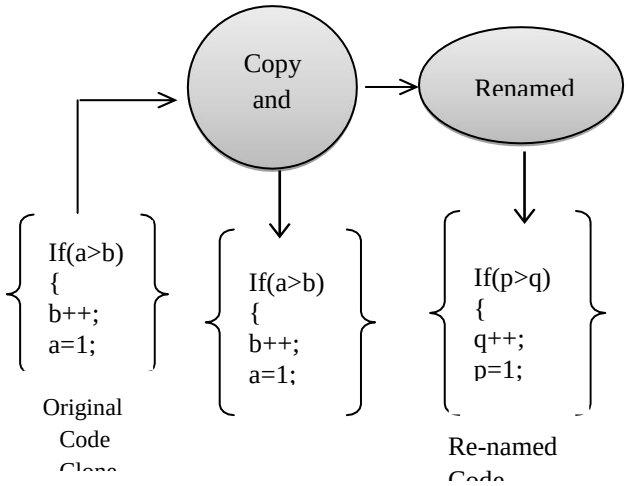

Fig 1. Codes with Clones

II. EVOLUTION AND TYPES OF CODE CLONE

Clones are differently classifiedas per their similarity level. In TYPE-1 clone is confirmed clone duplicate dis-regard to blank areas and remarks. These are called as homogeneous clones. In TYPE-2 clones a few changes in factor changes, literals and sorts, white area, outline design and remarks. The third type of clone includesall changes of TYPE-1 and 2 clones e.g. expansion or disposal of articulations. The non-likeness most extreme edge of a clone finder chooses how much non-comparable parts can be clone class is same[4].

Utilize string construct energetic example coordinating with respect to speck plots to liken entire lines that have been managed to overlook void area and remarks. Diagonals with holes demonstrate conceivable clones of Type-3, and an outline tracker is keep running on the framework to discover diagonals with gaps up to a positive area.

Some paper contemplated, TYPE-2 and 3 clones similarly viewed together as close duplications. It has been inquired about on dissecting the development of code clones. This examination dissected how code clones are modified through Versions for different levels of granularity, for assorted subject frameworks recorded in disparate dialects, utilizing different clone recognition apparatuses and from various perspectives.

The majority of the concentrated on TYPE 2 and 1look alike and existing data according to development of TYPE-1 and 2 clones are for all intents and purposes rich. Sort 1 implies indistinguishable source code replicated; TYPE-2 duplicates character re-names of symbols; TYPE-3 clone's night character increasingly broad changes. The current hash based systems of clone recognition oversee just TYPE-1 and TYPE-2. We portrayed hash-capacities for TYPE-1, 2 and 3 clones which display sensible presentation precision. In TYPE-4 in light of capacity same yet they are different in sentence structure. This clone is named as TYPE-4 semantic clones, in these sorts of codes; cloned part isn't basically copied from the first code. Codes are identical and termed as code clone. The clone could be determined into two types:

Similar type of one the texture similarity and other considers the semantic similarity in which the code clone must have similar activities, means functionality similarity. The type of clones are generally consequence of duplicate a code segment and the copying to differentposition.

Here, we describe duplicate kinds relied-on the type of identical binary code segments could be:

Texture Similarity: A binary programs segments could be same depending upon the resemblance of their code-text. The clonesclasses are described to search textureresemblance.

Type I: is called perfect clones where a duplicate code portion is similar to unique code segment except for some feasible variations in blanks and comments.For e.g.

Exact Code	Copied code

int a, b, big;
//Comment 1
if(a>b)
{
big = a;//a assigned as biggest
}
else
{
big = b;//b assigned as biggest
}
System.out.print("Largest of Two Number is " +big);// Comment 2
	int a, b, big;
	if(a>b) //Comment 1
	{
	big = a;//a assigned as biggest
	}
	else// Comment 2
	{
	big = b;//b assigned as biggest
	}
	System.out.println("Largest of Two Number is " +big);

In the above code there is a text similarity if we remove comments and whitespaces.

Type II: is called renamed clones where duplicate code segment is sameas native code fragment excluding few probable dissimilarities of user- defined variables (methods,constants, classes etc.), types, design or comments.

Exact Code	Copied code

intx,y,grtr;
if(x>y)
{
grtr =x; }
else
{
grtr=y; }
System.out.println("Largestof Two Number is" +grtr);
	int x1,y1,grtr1;
	if(x1>y1)
	{
	grtr1 =x1;
	}
	else
	{
	grtr1 =y1; }
	System.out.println("Large stofTwo Number is" +grtr1);

 ISSN: 2454-4248
392 – 399

In the above code there is a change in the names of variables.

Type III: a Duplicate code segment is altered by shifting the design of original code segment e.g. adding or discardingfew statements.

Exact Code (Fragment1 developed by developer A to calculate the factorial.)	Copied code ((Fragment1developed by developer A to calculate the factorial.)

Inta, b=1;
for (a=1; a<=VALUE; a++)
b=b*i;
	int fact(int x) {
	if (x == 0) return 1 ;
	else return x * fact(x-1) ;
	}

Functionality Similarity: Fragment code could be same on the similarity of theirfunctionalities

without being text same. When the usefulness of binary program segments are same or identical, like post and pre-situations referred as Type –IV clones:

*Type IV:*Such clones have semantic look alike among code segments. Clones of such kind are not essentially copied from native code because there exists similar meaning and are alike usefulness but already developed by another programmer..

Exact Code	Copied code

int a, b, big;
//condition to check biggest number
if(a>b)
{
big = a;
}
else
{
big = b;
}
System.out.println("Largest of Two Number is " +big);
	int x , y,result
	int result = (x>y)?i:y;
	System.out.println(("Largest of Two Number is " +result);
	}
	}

Other Clone Detection Terms are which also comes under above mentioned types:

i. Exact Clones: Exact clones are essentially Type I.

ii. Renamed Clones: Renamed Clones are Type 2 clones.

iii. Parameterized Clones: A parameterized look alike are renamed with organized renaming. These clones are subset of TypeII.

iv. Near-Miss Clones: All clones of Type II are near-miss clones. However, modification within a statement(s) are considered as near – miss look alike. So, Type III look alike can be named as near-missclones.

v. Structural Clones: Itconcentrates to discover identical architecturestructures.

vi. Reordered Clones: Based upon semantic resemblance, arranged clones are of TypeIV.

vii. Intertwined Clones: segments are identicalas per their usefulness. Thus interlaced lookalike are taken as Type IVduplicate.

Table 1. Types of Code Clone Terminology

Types of Clone	Description
TYPE-1 Clone[5]	Same program segment excluding to deviation of comments, layout design or whitespaces.
TYPE -2 Clone [5]	Identifiers, variables and literals, comments and whitespaces
TYPE-3 Clone	Increase the code line in the original code
TYPE-4 Clone	Same computation but implements the different logic.
Same Function Code[6]	Gives a same functionality with respect to explanation of resemblance, but could be designed uniquely.
Solution Records[6]	Individual codeof single record developing ananswer to coding issue.
Solution Set[6]	Solutions records re-solve the similar coding issue.
Clone set[7]	Binary solution records out of similarresult-set that supposed to function equally.

III. LITERATURESURVEY

E. Kodhai, et al (2016) [17] presented an incremental clone detection along with hybrid approach to locate clones in multiple alterations of program. This hybrid technique is a merger of metrics computation and textual analysis. In period of last ten years, considerable research effort was made for detection and expulsion of clones from software framework. However, some practical tools are available for programming languages. Majority of techniques used for clone detection are limited one alteration of program. Both techniques of clone detection and modification functionalities are united with Clone Manager, is a tool for Java and C programs. This incremental technique is an improved feature to Clone Manager tool. They examined the improved Clone Manager tool with parameters recall ratio and precision for 6 open source projects. **Dongjin Yu, Jie Wang et al., (2017) [18]** proposed a novel technique of code clone detection based on Java bytecode. Code clones are commonly believed as unwanted for many reasons, despite of ease provided to developers. Identification of code clones improvise the quality of source code via software re- engineering. Several methods were proposed in Java source code while just few concerned to its bytecode. The Java bytecode displays semantic nature of code. Using the block-level code fragments extracted from bytecode, and simultaneously identify code clones at both method level and block level. During code clone detection process the similarities of instruction sequences and call sequences are calculated to enhance accuracy and performance. The results proves that proposed method is

 ISSN: 2454-4248
392 – 399

more effective than existing methods.**Yingnong Dang, et al., (2017)[19]** described the encounter of shifting XIAO, a code-clone detection and analysis approach and supporting tool, to wide industrial practices i.e., (1) shipped in Visual Studio 2012, a broadly used industrial IDE; (2) deployed and intensively used at the Microsoft Security Response Centre. Amid programming improvement, code clones are normally delivered, as some of the same or comparative code pieces spreading inside one or numerous expansive code bases. Various research ventures have been done on experimental investigations or apparatus bolster for distinguishing or dissecting code clones. Nonetheless, practically speaking, couple of such research ventures have brought about generous industry adoption. According to our encounters, innovation exchange is a fairly confounded excursion that requirements huge endeavors from both the specialized viewpoint and social perspective. From the specialized perspective, huge endeavors are expected to adjust an examination model to an item quality device that tends to the requirements of genuine situations, to be coordinated into a standard item or advancement process. From the social viewpoint, there are solid needs to cooperate with professionals to recognize executioner situations in mechanical settings, make sense of the hole between an examination model and an apparatus fitting the necessities of genuine situations, to comprehend the prerequisites of discharging with a standard item, being coordinated into an improvement procedure, understanding their discharge rhythm, and so forth. **ShrutiJadon, (2016) [20]** proposed to create a feature set by analyzing C program for fragments of code and matching similarities. Code clones characterized as succession of source code that happen more than once in a similar program or crosswise over various projects are unfortunate as they increment the span of program and makes the issues of excess. Settling of bugs recognized in one clone require discovery of all clones. Henceforth, it is basic to recognize and evacuate all code clones in a program. The concentrate of past research chip away at the code clone location was to discover indistinguishable clones, or clones that are indistinguishable up to identifiers and strict esteems. Be that as it may, identification of comparable clones is regularly essential. Based on highlight sets the grouping of calculation is being performed by utilizing the Support Vector Machine (SVM) as a machine learning apparatus. The yield of the machine device would be the closeness proportion with which the two C programs are identified with each other and furthermore the class in which they would happen. It was watched that the test consequences of the instrument execution indicate identification of code clones in the program and its exactness increments with the expansion in number of occurrences. **Abdullah Sheneamer et al., (2015) [21]**presented a hybrid technique which utilized a coarse grain method to break down the clones efficiently to enhance precision. In the event that two parts of program code is indistinguishable to all of them, are known as code clones. Program look alike present challenges of programming upkeep, virus engendering. Coarse-grained duplicates indicators have maximum accuracy than finely grained, yet such identifiers have better review in comparison coarse-grained. Such manner, utilizes fine-grained identifier to get extra data related to clones and enhance review. This technique distinguishes Type 1, 2 look alike utilizing hash esteems to pieces, and missingprograms clones utilizing square discovery or resulting examination among them utilizing Leven-shtein separation and Cosine methodsof changing limits. **Chanchal K. Roy, et al., (2014)**

[22] presented a complete survey on recent clone management with deep study of its activities such as tracing, cost benefit analysis, etc. which are past detection analysis. Copied code or code clones are a sort of code notice that have both positive and negative effects on the advancement and upkeep of programming frameworks. Programming clone inquire about in the past for the most part centeredon the recognition and examination of code clones, while look into as of late reaches out to the entire range of clone administration. In recent decade, three studies showed up in the writing, which cover the identification, investigation, and developmental attributes of codeclones.

Table 2. Literature Survey Performance Parameters Analysis

Author Name	Technique s Used	Drawback	Parameters
.Kodhai et al.,2016	Incremental clone detection	duplicate d code fragment s or code clones.	Recall, Precision
Dongjin Yu et al., 2017	block-level code fragments extracted	maintain ing of software	Precision, Recall and F-measure
Yingnong Dang et al., 2017	XIAO classifies	inconsistencies,	-
ShrutiJadon et al., 2016	Support Vector Machine	Problem sof redundan cy.	Accuracy
Abdullah Sheneamer et al.,2015	Grouping and Hashing Normalized Blocks	software maintenance and cause bug propagat ion	Precision, Recall and F-measure
Chanchal K. Roy et al.,2014	suffix-tree- based	strain tsatisfacti on optimization	-

This is the main overview on clone administration, where we point to the accomplishments up until now, and uncover roads for additionally look into vital towards a coordinated clone administration framework. We trust that we have completed a great job in looking over the territory of clone administration and that this work may fill in as a guide for future research in theregion.

Structure of Clone Detection: The overall structure involved in Clone Detection include methods such as determining type of clones, Classification of code clone, and Transformation/Normalization.

 ISSN: 2454-4248
392 – 399

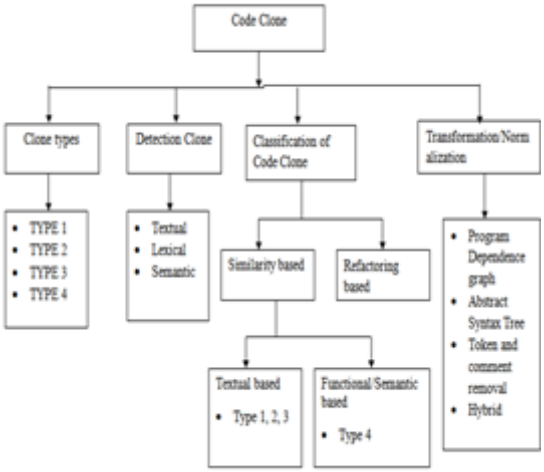

Fig 2. Structure of Clone Detection

IV. CLONE DETECTIONTECHNIQUES

Clone detection deals with finding similar code in two programs or more than two. Detection can be based on Textual analysis, Lexical analysis, Syntax analysis, Semantic analysis, Hybrid analysis and Metric analysis.

Text Base Method: This technique is used to find similar text in large software systems, web pages or in text files. In this no transformation is required. This approach is used to detect Type1 clones. Text based approach doesn't perform any semantic and syntactic study of source code it is fastest clone detection methods. It is the simple way to detect clone, which precedes each LOC (Line of code) representation. The objective source program is characterized to be arrangement of lines or strings. At that point the paired code pieces are assessed with each other to look throughthe

coordinated arrangement of content. On the off chance that match is sought, that is more code portions are observed to be similar, at that point they are continued as clone class/combine by the discovery technique.[8].

Token-Based Method: Token-based approach is same as text-based approach however instead of taking a LOC as representation directly, a lexical study converts each lines of code into a series of tokens. After information values, identifier are replaced by several tokens. The token series of these segments are evaluated effectively through a suffix tree algorithm. The result is also presented in dot deployment graph. This method is slower than text-based technique since of the tokenization phases. Applying the suffix tree matching method, the time-complexity is same as text based method. In this, source code is transformed in to lexical, also known as tokens. Type-1 and Type-2 clones can be detected using this approach[9].

Abstract syntax tree (AST): In Tree based method a program is generated into aparsed tree/ syntax tree with the block of the code of interest. Same sub-trees are then found in the tree with some tree coordinating strategies and the comparing source code of a similar sub-trees are continued as clones classes/sets. The AST comprise the entire data about the source code. Variable names of the source are rejected in the tree represents; refined techniques for the detection of clones still could be applied. It converts token in to syntax tree. A clone detected by using such kind of approach is known as Type 3 clones[10].

4.4. Program Dependence Graph Based approach: Data dependencies of a program could be represented as a execute program dependency graph. Since it files the relationship between the structure and data, it could be used to trace the change after developers copy and paste events. The PDG method takes one-step further than abstract syntax tree method i.e, to find the PDG of the system. This technique is used to detect syntactic as well as semantic behavior of source code. It shows the control flow and data dependence. It is used to detect Type 4 clones[11].

Metric Based Method: Despite of evaluating the program flatly dissimilar metric of program are collected and compared to discover clones [12]. A few clone location strategies today utilize measurements for identifying same codes. In the beginning capacities which are obscure however an arrangement of programming measurements are assessed for syntactic units, for example, a capacity or class, strategy or even an announcement then these metric-values are computed to seek clones. Then the metric were compared from lay-out, expression, names and individual control-flowof the function.A clone is detected only when class of whole function bodies that have same metric values are verified.

*Hybrid-based Method: V*arious different detecting methods utilize hybrid method in detecting clones. Such approach is a combination of the other detection methods. This methodintegrates syntactic method relied upon abstract syntax tree semantic and metrics methods in combined with particular comparisonfunctions.

Table 3. Semantic Clone detection and Comparative Study

Transformation /Normalization s	Source Code Representation	Clone MatchingTechnique	Merits	Demerits	Tools
Program Dependence [12]Graph	Find Grained	n- length patch matching	High Precision and Recall	Needs a PDG generation for dissimilar language, works for C language	Duplex

 ISSN: 2454-4248
392 – 399

Suffer code to PDG	PDG	Dependence Graph sub-graph comparing using program slicing	Feature Extraction can process of refactoring	Slow and huge code bases	PDG-Dup
Token and CommentRemoval	Text	Vector defined in LSI	Search high level structure clones	High dependent on comment, less precision	-
Abstract Syntax Tree	Programming	detecting exact and near miss **clones** overarbitrary program fragments	block level clone detection	Time consuming	-
Hybrid	syntactic and semanticc haracteristics	Texture and Metric based	LightweightPrecision value comes high	-	MCD Finder (Java)

V. CODE CLONE DETECTIONTOOL

Anconcept of such result was that the study recognized answers to execute the coding languages since we distinguish for an often issue, the main results must executionaly same. Consequently, the choosinganalyzed things required to add clone detectingmethods. The program can be accessed and results while executing coding platforms supports by almost allsimulation methods fordetection.

Principally, they required duplicate detection simulation methodto detect Type 1,2 and3 clones to study the syntactic resemblanceof FSCs. They generated a survey and look for existing simulation methods.Severalwork pro-types weren't available or cannot be brought to execute acceptably. Numerous simulation tools weren't added in the analyses because of scalability and under performance / decreases of help for few clone-types [13].

Clone-DR and CP-Miner have less execution and efficiency evaluated to Deckard. CC- Finder has less executionas compared to Deckard and doesn't provision type3 clones. In last, they select binary clone detection simulation tools that together could study JAVA and C programs: ConQAT and Deckard. ConQAT is discussed in as newest, useful or speedily easily available detector of duplicate structure or system. While analyses, Deckard has defined withbetter execution and scalable asthey are well explained and have been used in prior study. Particularly, at time of analyses, those were binary methodsareeasily available and easy to create them work forourselves.

ConQAT is a steady freeware dash-board tool-kit used in trade. It is normal aim simulation approach for several types of code size and explanation theory. ConQAT , gives various particular code duplication detection for several coding platforms adding C/C++, cobol, and JAVA. This divide detecting methods for Type-1 or 2 or 3 clones. They used the previous method. ConQAThasbeen described in several analyses in look alike detection adding the study they construct on. Deckard uses an effective method for verifying same sub-trees and registered it to tree re-presentations of native code. This commonly create a parsetree constructor to construct parse-trees necessary due to its method. By a same parameter it is possible to control whether only Type-1, Type-2 clones and Type-3 clones are detected. Deckard is stable tool described in-other analyses in adding the study, we construct on.CCCD is a new clone detection simulation tool that describes concolic study as its main method to detect code-clonesConcolic examine affiliation's representative program execute and testing. CCCD identifies just copies in execute programs planned in C. The concolic contemplate offers CCCD to objective on the usefulness of execute-program as opposed to the syntactic properties. It has the strict that it just identifies work level clones[14].

Table no. 4 Comparison of code clone detection tool

Tools Compared	Techniques
CCFinder [14]	Suffix Tree and Token
Clone DR	Abstract Syntax Tree, Hashing
Covet	Abstract Syntax Tree and Metrics
Duploc	Texture, Sub-string Matching
Dup[14]	Text, Token and suffix Tree
CLAN	Abstract Tree and Metric
CCDIMI	AST, Tree Matching
PDG-DUP	PDG, Program Slicing
Clones	Token, Suffix Tree

VI. CODE CLONE DETECTIONPROCESS

Code detection must need to search blocks of programs with maximum resemblance in system source texture. Major issue is that it isn't identified before-hand which code fragments might be rewritten. The detection actually should evaluate each possible piece with each other possible portion.

Here, we describe an overview of fundamental phases ofduplicatecode detection method. Clone Detection Process as described as follows:

*Pre-processing:*In this phase all the uninterested part in the source code is removed and then source units are determined (functions or methods,files, begin-end blocks, classes,statements or seriesof source lines). Then native units can be sub-divided into tokens or lines for assessment. Contrast entities can be resulting out of the syntactic design of source unit[15].

 ISSN: 2454-4248
392 – 399

In pre-processing are three main steps:

- *Un-interesting Part Remove:* Every source-code un-interesting to the comparison step is clarified out in thisstep.
- *Conclude Source Units:* Later reducing the un- interesting program, the left source code is divided into a set of dis-joint fragments known as native entity. This unit is the main source- parts that might be complicated in flat clone connections with everyone. Source-units could be at any level of granularity i.e, Records, clusters or classes, methods or functions, sequence of source lines and start-endblocks.
- *Conclude Evaluation Units:* In source-units might require to be further divided into small units provisional on the evaluation method used be the tool.

Transformation: Evaluation is concluded, when the evaluation method is different from text, the basic code of the evaluation units is converted to a suitable intermediate form for evaluation. Conversion of the source-code into centerformation is frequently known as withdrawn in the converse disciplinecommunal.

Extraction: It converts source-code to appropriate form as input to the original evaluation methods. It is generally includes various following phases.

- *Tokenization:* Every line of source is portioned into tokens giving to lexical code of any coding platform.
- *Parsing:* A syntactic method, the complete source-code base is decoded to construct anAST orparsetree.
- *Data Flow and Control Analysis:* Semantics methods create PDG from the source-code. The PDG describe the statements and situations of a code, while regions shows data dependencies or control.

Normalization: It is optional phase considered to remove superficial diverse like dissimilar in white-spaces, identifiers, formatting and commenting.

*Match Detection:*The changed code is contribution to an appropriate correlation calculation where changed examination units are contrasted with each other to discover a match. Some mainstream coordinating calculations are the addition tree calculation, dynamic example coordinating and hash-esteemcorrelation.

Formatting: In this stage, the clone combine list acquired as for the changed code is changed over to a clone match list as for the first code base. For the most part, every area of the clone combine gotten from the former stage is changed over into line numbers on the first source records[16].

Filtering: In this stage, false positive clones are sifted through with manual examination as well as a perception apparatus.

Aggregation: keeping in mind the end goal to diminish the measure of information or to play out certain investigation, the clone sets are collected to bunches, classes, inner circles of clones, or clone bunches and soforth.

V. CONCLUSION

Code cloning is aprocess of reusing the code as it is or with several modifications. Code clone recognition is a specialty of recognizing the substance comparability between the projects or WebPages. An endeavor is made to plan a strategy called "SD Code Clone Detection" for both static and dynamic WebPages. It depends on levenshtein's approach. This strategy contains a few stages like, parsing and investigation, tree development, code similitude measure and clone identification.Investigations of clone serve a key part in appreciation and tending to cloning issues in software, In this paper we depicted a systematic written work review that we coordinated to inspect the current situation with data about clone headway We ‘ve examined several papers to explore various tools and techniques used for clone detection.Code clones are indistinguishable fragment of source code which may be embedded deliberately or inadvertently. Reusing code pieces through reordering with or without minor adjustments is general undertaking in programming advancement. In any case, the presence of code clones may decrease the outline structure and nature of programming like variability, meaningfulness and viability and consequently increment the continuation charges. It can implement in proposed work with create a new tool that is code optimized manager. In this research work enhance the performance parameters like time complexity and accuracy with optimized SVMalgorithm.

REFERENCES

[1] R. Koschke, Survey of research on software clones, in: Duplication, Redundancy, and Similarity in Software, Dagstuhl Seminar Proceedings, 2007, p.24.

[2] C.K. Roy, J.R. Cordy, A Survey on Software Clone Detection Research, TechnicalReport 2007-541, Queen’s University at Kingston Ontario, Canada, 2007, p. 115.

[3] B. A. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE- 2007-01, School of Computer Science and Mathematics, KeeleUniversity, Keele and Department of Computer Science, University of Durham, Durham, UK, 2007, p.65.

[4] G. Antoniol, G. Cassaza, M. Di Penta, E. Merlo, Modeling clones evolution through time series, in: Proceedings of the 17th International Conference on Software Maintenance (ICSM ’01), 2001, pp.273–280.

[5] BellonS et.al. Comparison and evaluation of clone detection tools. IEEE Transactions on Software Engineering vol33(9)(2007),pp.577- 591.

[6] AndrianMarcuset.al. Identification of high- level concept clones in source code. In: Proceedings of the 16th annual international conference on automated software engineering (ASE 2001). Piscataway: IEEE(2001), pp.107- 114.

[7] Jiang L, Misherghi G, Su Z, Glondu S. 2007. Deckard: scalable and accurate tree-based detection of code clones. In: Proceedings of the 29th international conference on software engineering (ICSE). Piscataway: IEEE(2007),pp. 96-105.

[8] S. Ducasse, M. Rieger, S. Demeyer, A language independent approach for detecting duplicated code, in: Proceedings of the 15th International Conference on Software Maintenance (ICSM’99), Oxford, England, UK, 1999, pp. 109–119.

[9] Tool Simian<http://www.harukizaemon.com/simian/index.html>(accessed April2012).

[10] S. Lee, I. Jeong, SDD: High performance code clone detection system for large scale source code, in: Proceedings of the

Object Oriented Programming Systems Languages and Applications Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA Companion '05), San Diego, CA, USA, 2005, pp.140–141.

[11] T. Kamiya, The Official CCFinderX website<http://www.ccfinder.net>(accessed April2012).

[12] K. Kontogiannis, R. Demori, E. Merlo, M. Galler, M. Bernstein, Pattern matching for clone and concept detection, Automated Software Engineering 3 (1–2) (1996).pp.77–108.

[13] F Deissenboeck et.al. Challenges of the dynamic detection of functionally similar code fragments. In: Proceedings of the 16th European conference on software maintenance and reengineering (CSMR). Piscataway: IEEE(2012),pp.299-308.

[14] M. Bruntink, A. van Deursen, R. vanEngelen, T. Tourwe, On the use of clone detection for identifying crosscutting concern code, IEEE Transactions on Software Engineering 31 (10) (2005) ,pp.804–818.

[15] J. Mayrand, C. Leblanc and E. Merlo. Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics, in: Proceedings of the 12th International Conference on Software Maintenance, ICSM 1996, pp. 244-253.

[16] M. Kim, L. Bergman, T. Lau, D. Notkin, An Ethnographic study of copy and paste programming practices in OOPL, in: Proceedings of 3rd International ACM-IEEE Symposium on Empirical Software Engineering (ISESE'04), Redondo Beach, CA, USA, 2004, pp. 83–92.

[17] Kodhai, Egambaram, and SelvaduraiKanmani. "Method-level incremental code clone detection using hybrid approach." International Journal of Computer Applications in Technology 54, no. 4 (2016):279-289.

[18] Yu, Dongjin, Jie Wang, Qing Wu, Jiazha Yang, Jiaojiao Wang, Wei Yang, and Wei Yan. "Detecting Java Code Clones with Multi- granularities Based on Bytecode." In Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual, vol. 1, pp. 317-326. IEEE,2017.

[19] Dang, Yingnong, Dongmei Zhang, Song Ge, Ray Huang, ChengyunChu, and Tao Xie. "Transferring code-clone detection and analysis to practice." In Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track, pp. 53-62. IEEE Press, 2017.

[20] Jadon, Shruti. "Code clones detection using machine learning technique: Support vector machine." In Computing, Communication and Automation (ICCCA), 2016 International Conference on, pp. 399-303. IEEE,2016.

[21] Sheneamer, Abdullah, and JugalKalita. "Code clone detection using coarse and fine-grained hybrid approaches." In Intelligent Computing and Information Systems (ICICIS), 2015 IEEE SeventhInternationalConferenceon,pp.472- 480. IEEE, 2015.

[22] Roy, Chanchal K., MinhazF. Zibran, and Rainer Koschke. "The vision of software clone management: Past, present, and future (keynote paper)." In Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on, pp. 18-33. IEEE, 2014.