International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT217290

Analysis of Software Clones

Chavi Ralhan, Rakesh Bishnoi, Ankit, Anjali, Hitesh Kumar
Lovely Professional University Jalandhar Punjab, India

Article Info
Volume 7, Issue 2
Page Number : 439-450

Publication Issue :
March-April-2021

Article History
Accepted : 20 April 2021
Published : 25 April 2021

ABSTRACT

Copied code or code clones are a sort of code that contrarily affect the improvement and support of software frameworks. Software clone research in the past generally cantered around the discovery. what's more, examination of code clones, while research lately reaches out to the entire range of clone the board. In the last decade, three reviews showed up in the writing, which cover the recognition, examination and transformative attributes of code clones. This paper presents a complete overview on the state of the workmanship in clone the board, with top to bottom examination of clone the executives exercise (e.g., following, refactoring, cost benefit investigation) past the recognition and examination. This is the main overview on clone the board, where we highlight the accomplishments up until now, and uncover roads for additional exploration essential towards an incorporated clone the board framework. We accept that we have worked really hard in studying the territory of clone the board and that this work may fill in as a guide for future research in the area.

Keywords : Software Cloning, Clone Analysis, Clone Management, Clone Detections

I. INTRODUCTION

Replicating existing code and gluing it in elsewhere followed by minor or major alters is a typical practice that engineers embrace to expand profitability. Such a reuse instrument commonly brings about copy or fundamentally the same as code parts dwelling in the code base. Those copy or near duplicate code fragments are generally known as code clones. There are numerous reasons why engineers purposefully perform such code cloning. Clear reasons incorporate reuse of existing executions without "re-designing the wheel. Code clones may likewise show up in the code base without the consciousness of the designers. Such unexpected coincidental clones might be presented, for model, because of the utilization of certain plan designs, utilization of certain APIs to achieve comparable programming assignments, or coding shows forced by the association. The reuse instrument by code cloning offers a few advantages. For example, cloning of existing code that is now known to be faultless, might save the engineers from plausible mix-ups they may have made in the event that they needed to execute something very similar from scratch. It additionally saves time and exertion in concocting the rationale furthermore, composing the relating printed code. Code cloning may likewise help in decoupling classes or segments and encourage.

On the opposite finish of the range, code clones may likewise be adverse much of the time. Clearly, excess code may swell the code base and may expand asset necessities. This might be critical for inserted frameworks and frameworks such as hand-held gadgets, telecom switches, and little sensor frameworks. In addition, cloning a code piece that contains any obscure flaw may bring about proliferation of that deficiency to all duplicates of the flawed part. From the support point of view, an adjustment in one code fragment may require reliable changes altogether clones of that piece. Any irregularity may present bugs or weaknesses in the framework. During the product improvement measure, duplication can't be evaded now and again. For instance, duplication may be upheld by the impediment of the writing computer programs language's fundamental system to carry out a proficient conventional arrangement of an issue nearby. Code generators may likewise produce copied code that the designers may need to adjust. Albeit disputable, past research reports exact confirmations that a critical bit (by and large 9%-17%) of a common programming framework comprises of cloned code, and the extent of code clones in the code base might be as low as 5% and as high as even half. Surely, due to the negative effect of code clones in the support exertion, one should eliminate code clones by dynamic refactoring, any place achievable regardless, in reality, powerful refactoring of code clones appears to be not to be a marvelous idea and not all clones are genuinely removable through refactoring. Because of the double part of code clones in the turn of events and upkeep of programming frameworks, just as the sober minded trouble in maintaining a strategic distance from or eliminating those, analysts and professionals have concurred that code clones ought to be distinguished and overseen proficiently. Since the development of programming clones as an examination territory in mid 1990s, critical commitments over years made the field develop and turn out to be a significant experienced territory of examination. Absurd whole course of programming clone research there have been as it were two eminent general overviews on clones. In 2007, introduced a short outline of the significant discoveries about various parts of programming clones including cause-impact of cloning, clone shirking, location, and development alongside a set of open inquiries. Around the same time, Roy and Cordy [40] likewise distributed another study containing an intensive survey on those equivalent zones with explicit spotlight on clone identification devices and procedures. A couple of ongoing reviews either center around location [02, 36] or development of clones [27]. In this vision paper, we give a broad overview on code clone research with solid accentuation on clone the executives and point readers to future examination headings. This paper is coordinated as follows. In Section II, we present a precise audit on an archive of 353 publications showed up more than 20 years. The audit draws a "higher" see on the general commitments and development along various measurements of programming clone research. This study is the result of cautious examination of writing past the said vault (depicted in Section II), and through investigation in the light of our experience. Segment III presents various perspectives of clone the executives exercises beginning with the definition and kinds of clones. While in Section IV, we list distinctive independent clone identification strategies, we examine the IDE-based clone indicators. We at that point talk about clone documentation in Section V. In Section XI, we portray the investigations for the ID of possible clones as possibility for refactoring/reengineering counting the representation of clones, money saving advantage examination also, planning of clones for refactoring. Our view on the difficulties for modern selection of clone the board is introduced in Section IX. At long last, Area X finishes up the paper with a

harsh rundown of the cutting edge alongside future research directions.

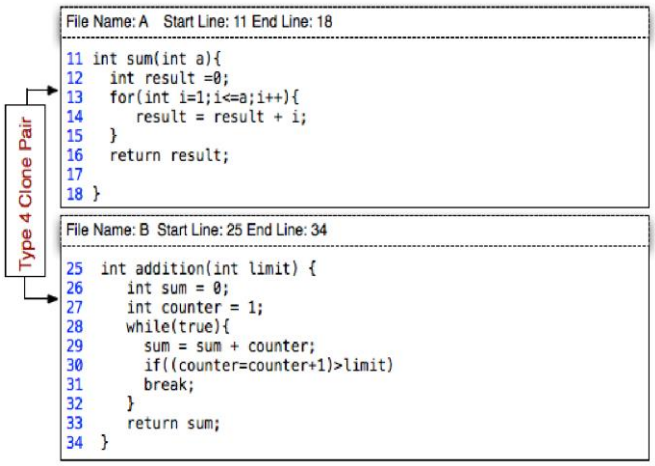


Fig1. Example of software clone

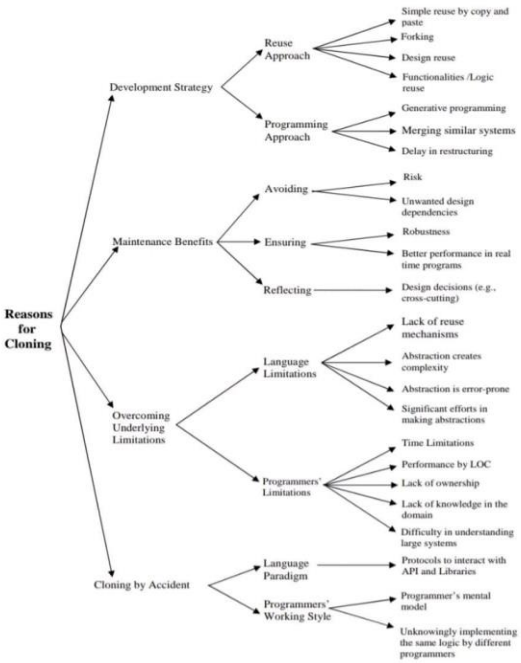
II. A PRECISE REVIEW OF CLONE WRITING

There has been over a time of exploration in the field of programming clones. To comprehend the development and patterns in the various elements of clone research, we did a quantitative survey on related publications. The vault arranges the publication by categorizing them dependent on their commitments in four significant sub-zones of clone research. The classes are as per the following.

Detection Publications in this class address procedures and apparatuses for the detection of software clones.

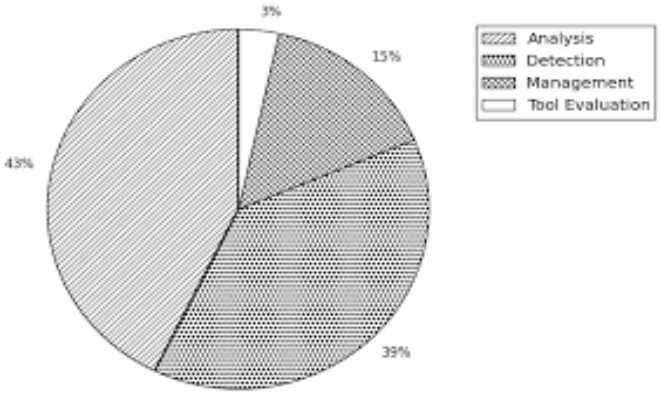
Analysis This classification contains publication that perform analysis on the different qualities of software clones, their etiology, presence, impacts in programming frameworks, just as examination of clone reengineering openings and suggestions. A greater part of such distributions report findings from subjective or quantitative exact investigations.

Management Publication in this classification address the issues, procedures and instruments for the administration of code clones past discovery.



III. CLONE MANAGEMENT

Clone the heads sums up all collaboration rehearses which are locked in at perceiving, sidestepping or shedding clones" [37]. As needs be, clone the executives encompasses a wide extent of classes of activities including clone area, following of clone advancement, and refactoring of code clones. As help for these activities, the documentation and investigation of code clones can be viewed as parts of clone the board. In addition, clone perception may likewise be a successful guide to clone investigation, and in this way to clone management.



a. Definition of Clone Code

In spite of the fact that copy or comparable code sections are generally known to be code clones, the meaning of clone has stayed pretty much obscure in the course of the most recent decade. The dubiousness is reflected in the definition given by Ira Baxter, "Clones are sections of code that are comparable as per some meaning of closeness" [12]. Regardless of continuous discussions in the exploration local area, there is no agreement on an exact definition yet. Presently, a scientist's meaning of comparability is regularly compelled by the program portrayal and discovery component of their specific clone identifier and, henceforth, fluctuates from one apparatus to another and furthermore from boundary settings controlling a device. The lowest shared factor broadly acknowledged today is the accompanying scientific categorization, which was made with regards to an investigation on contrasting clone indicators [15].

b. Types of clones

There is a need to learn about various types of clones to distinguish them effectively [16]. Principally the clones are separated in to Exact, Renamed or Parameterized, Near-miss and Semantic clones. The other name given to these clones are Type 1, 2, 3 and 4 individually.

Type-1 Clone These sorts of clones look like unique code with variety of remarks furthermore, clear spaces. They can be distinguished by utilization of text based, token based or then again, any straightforward clone recognition procedure.

Type-2 Clone Varieties in name of literals, factors, catchphrases make renamed or type 2 clones. The utilization of token or metric based methods will actually want to distinguish these sorts of clones in a code. Family, Columbus, MCD-locater are a few of the apparatuses utilized by designers to recognize renamed clones.

Type-3 Clone They are made from base code by articulation adjustment, expansion or cancellation [17]. Clone Digger dependent on tree procedure can be utilized to distinguish close to miss clones. The AST sub braid is contrasted and each other in tree-based methods.

Type-4 Clone The program coding or punctuation is diverse yet the conduct or capacity of the clone stays same in semantic clones. Devices like Duplex, Scorpio in view of chart strategy is end up being productive in recognizing these sorts of clones [18].

c. Advantage of Cloning

Reusing a code part for comparable necessities by cloning is an effective strategy in programming improvement that helps in decreasing expense and time. The utilization of layouts has been empowered by a portion of the ideal models of programming [9]. Now and then, an engineer composes a piece of code which coincidentally coordinates with some current code. This prompts incidental cloning. Designers dread to compose the code without any preparation because of its huge size. Now and again, individuals duplicate a segment of code because of trouble in getting it. There are various benefits of cloning as it ends up being useful when there is absence of information about language on which developer is working. It additionally helps in complying with constantly constraints of errand doled out to each designer that are associated with some specific work. The examination here has demonstrated that it will save time and cash as it makes programming framework improvement quicker [12]. The product development and investigation gets simpler utilizing it. It additionally helps in recognizing various bugs existing in the code piece so those bugs won't influence the other program where that part will be utilized. The size of program gets decreased and it improves the comprehend capacity of a program. To shield copyright content from being replicated a

literary theft recognition device plays a significant job which utilizes the idea of cloning.

d. Disadvantage of Cloning

Expansion in support exertion is an unfavorable impact of cloning as we need to monitor all the replicated segments. On the off chance that any change is made in one duplicated area, it should be reflected on the whole the replicated areas to eliminate irregularity [13]. The bugs get proliferated from unique code to the duplicated one which is known as bug engendering.

IV. CLONE DECTECTION

To identify code clones, various strategies and instruments have been created after some time by taking upkeep exertion of clones in to account. The strategies are classified by info, portrayal and calculations utilized by them. Extensively these strategies are delegated text based, token based, metric based, Abstract Syntax Tree (AST) based, Program Dependency Graph (PDG) based and half and half as demonstrated in the itemized depiction of various procedures based on different boundaries alongside their correlation is Related research nearby clone identification and the executives.

Matrix Based-: Metrics based methods [38, 28] are normally used to recognize work clones. The procedures depend with the understanding that comparative code sections should yield fundamentally the same as qualities for various programming measurements (e.g., cyclometric intricacy, fan-in, fan-out). Regularly, for the code portions a bunch of measurements are accumulated into vectors. The distinctions in the vectors are determined, where close vectors (e.g., estimated by Euclidean distance) show that their comparing code pieces are clones.

Token Based-: In token-based techniques [5, 33], the whole program is changed into a surge of tokens (i.e., singular units/expressions of importance) through lexical investigation. At that point the symbolic stream is examined to discover comparable symbolic aftereffects, and the first code divides relating to those aftereffects are accounted for as clones.

Abstract Syntax Tree based-: Abstract syntax tree-based techniques [12, 44, 41] are created on the way that comparative code portions ought to likewise have comparable syntactic design. In this manner, the program is parsed to create a grammar tree, where comparative subtrees demonstrate that their relating code fragments are clones.

Program Dependency Graph (PDG)-: For a given program, a bunch of PDGs (Program Dependency Graphs) are created dependent on the information and control conditions among the assertions of the program. The code fragments relating to the isomorphic subgraphs are recognized and revealed as clones [32, 31, 26].

Other Techniques-: Other than the previously mentioned prominent procedures for clone discovery, different strategies, for example, formal techniques [25], and mix of unmistakable techniques [17] were likewise drawn closer. Following of conceptual memory states during the execution of the program was additionally endeavored to recognize semantic clones [21]. As recorded above there have been a large number cutting edge clone finders accessible. Notwithstanding, still little is thought about the convenience of the clones recognized by various clone locators. Moreover, assessment of the clone locators is as yet an open test [40, 36] as we don't have solid benchmarks with the exception of the apparatus examination trial of Bellon et al. [15] and the change-based structure of Roy, Cordy and Svajlenko [18, 23]. The boundary settings of the clone

indicators are another danger as demonstrated by Wang et al. [19] as jumbling design decision issue and directed a broad examination considering six clone locators to improve the impacts of the issue. Also, the issue of enormous information clone location is a developing test for clone the board and for some other related applications [20].

V. STEP INVOLVED IN CLONE DETECTION

In this part, the cycle of clone location has been given to sum things up. There are such countless existing instruments and strategies to identify clones and to do so various advances should be followed which are discussed below-:

Step 1: Pre-processing This the first step of clone detection which is future divided into different step as given below:

Removal of unessential code: All insignificant source code like whitespace and remarks will be disposed of.
Generation of source units: The excess code will be isolated into various arrangement of disjoint sections known as source units. These source units are additionally separated into much more modest units dependent on various correlation strategies utilized by the device.

Step 2: Transform: In this progression, source units got in the past advance are changed over in to some transitional structure which can be provided as a contribution to correlation algorithm. There is a need of this progression altogether the procedures but text-based procedure. It very well may be accomplished either utilizing standardization or extraction. The extraction is further partitioned into three subcategories given below:
Tokenization: Gathered source units from past advance are changed over into tokens utilizing a few strategies or then again lexical conventions subsequent to eliminating remarks, deletes and so on Tokens are additionally orchestrated in to groupings.
Parsing: Abstract Syntax Tree (AST) is created by examining the whole source code. AST is further separated in to subtrees [15]. Those subtrees are thought about for clone recognition.

Control and information stream analysis: Program Dependency Graph (PDG) diagram is made through certain apparatuses in which control and information reliance is addressed by edges and articulations by hubs. The PDG sub diagrams are analyzed for clone location.

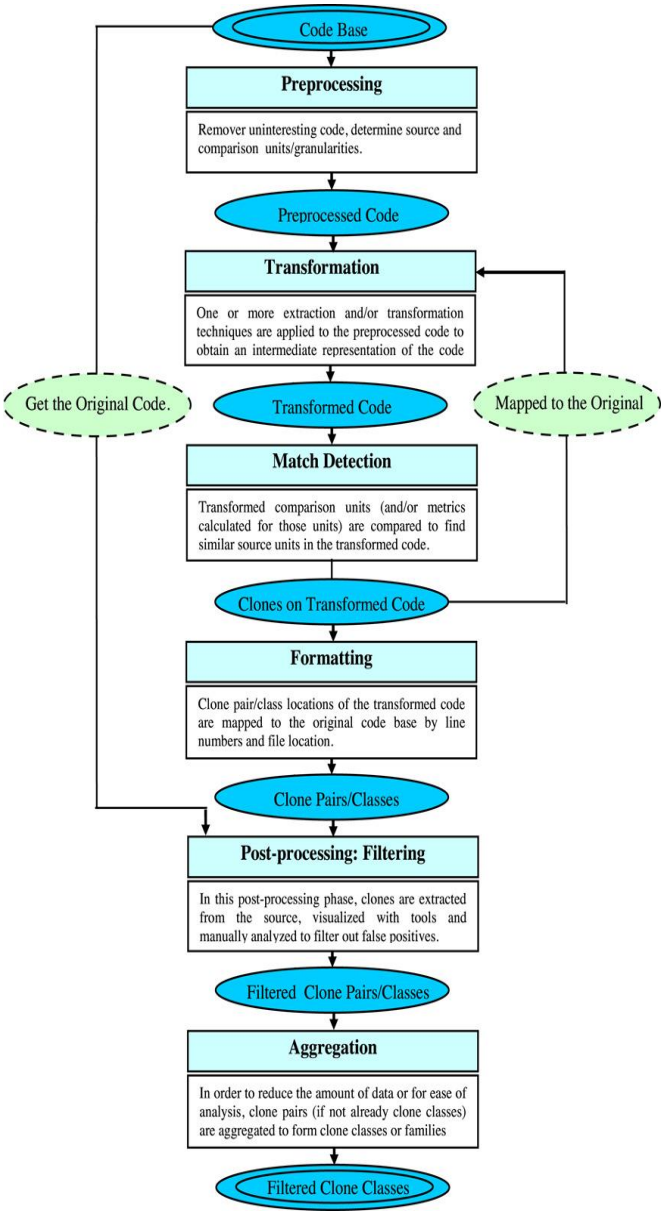
Step 3: Match Detection: In this progression, an appropriate match is found by contrasting the yield of change stage for example changed units utilizing examination calculation. The clones are addressed as clone sets, family and classes. The addition trees and hashing are a portion of the correlation moves toward that can be utilized.

Step 4: Formatting: This stage is very unique in relation to past stage as in this stage, clone sets of changed code are planned to unique source code utilizing document area.

Step 5: Post handling: In this progression, programmed heuristic or manual investigation is utilized to rank and sift through the clones. Human specialists are utilized to sift through bogus positives by doing manual investigation. Heuristics dependent on variety, length, recurrence and different attributes of clones are utilized to consequently sift through or rank clone applicants.

Step 6: Accumulation: It is the last advance of clone identification measure which incorporates appropriate information constriction and investigation. The clone family and classes are shaped by joining recognized clone sets.

According to C.K. Roy's paper [1], clone detection usually takes 6 steps



VI. CLONE DOCUMENTATION

Different clone detectors report the aftereffects of clone identifiers in various configurations like XML, HTML, and plain content. There are varieties in the detailed data too. Some clone indicators report clone combines just, while some different apparatuses report clones regarding clone gatherings. Such varieties make it hard for information trade between clone locators, which likewise adds to the difficulties in no holds barred experimental examination of clone finders. To limit the distinctions in the presentation of clone data, Harder and Göde [16] as of late proposed the Rich Clone Format (RCF), an extensible blueprint-based information design for capacity, tirelessness, and trade of clone information. Duala-Ekoko and Robillard [30] proposed clone locale descriptor (CRD) to portray clone areas inside strategies in a manner that is autonomous of the specific content of the clone district or its supreme area in a record. Nonetheless, such a plan has various impediments. To begin with, little changes in the code corresponding to the <anchor> (e.g., end state of circle, spreading predicate of contingent proclamations) will refute the CRD. Second, the plan is powerless against settling levels, and accordingly a basic expansion or expulsion of settling level will refute the CRD. Third, the relationship of 'else' blocks with the nearest 'if' block forestalls the CRD conspire separating between the two sorts of squares. In particular, the utilization of the CRD scheme didn't save Clone Tracker [30] from re-conjuring the hidden clone identifier to distinguish potential changes in the clones, however the computational cost of re-identification was demonstrated as one of the inspirations driving the plan of CRD. The above conversation demonstrates that the line and section data, or the theoretical level CRD based documentation of clone locales are pretty much powerless against changes in the advancing code. To defeat such affectability to code change, marker based labelling support in IDEs like Eclipse can be utilized for clone documentation. Such labelling of clones can offer implicit help for obliging changes in the source documents [21]. Further examination might be needed to check this chance. Catching the area of clones dependably is essential for device correlations and furthermore for following clones over ensuing renditions. On the off chance that devices are to be coordinated from various merchants, a concurred approach to report clones is required.

RCF is a stage towards a typical organization [16], however it doesn't address all necessities [8]. A typical theoretical model for clone data is a significant test on account of contending necessities (e.g., it ought to be both conventional and proficient) [7]. There has been some advancement towards a brought together model [8]. We anticipate that real practical progress should occur, in any case, just if diverse exploration groups really begin to trade information – and between two groups as well as among numerous groups. We don't see this occurrence right now aside from trading benchmark information for clone finders and for that utilization case, RCF is by all accounts adequate.

VII. CLONE ANNOTATION

The engineers frequently intentionally make clones, for test plea, to empower free advancement of comparable implementations. During the clone the board cycle, the engineer may not have any desire to refactor/eliminate those clones, and might need to check those to demonstrate such choices so they won't need to experience those equivalent arrangements of clones again and again. In addition, the choice should be recorded and divided between various software engineers, and there ought to be offices for the designers to survey those clones sometime in the not-too-distant future, in the event that they need to re-examine their administration choice. To the most awesome aspect our insight, such an element is discovered uniquely in JSync [41], which permits the designer to clarify sets of clones for dodging future experiences. Despite the fact that there are a few thoughts and executions of clone-development perception, there isn't sufficient observational evaluation of these. We additionally accept that further advancement can be accomplished by considering existing work in data representation.

Cost Benefits Analysis and Scheduling of Refactoring: Not much research has been done towards money saving advantage butt-centric ysis of code clone refactoring and their booking. Bouktif et al. [6] first proposed a straightforward exertion model for the refactoring of clones in procedural code. Zibran and Roy [11, 12, 14] proposed a more complete exertion model for assessing clone refactoring endeavours. They defined booking of code clone refactoring as a limitation fulfilment advancement issue and applied imperative programming (CP) procedure to process an ideal arrangement of the issue. Lee et al. [34] applied requesting muddled GA (OmeGA) to plan refactoring of code clones. Mondal et al. [23] proposed a programmed method of positioning clones for refactor-ing through mining affiliation rules of the advancing clones. Juergens and Deissenboeck [03] portrayed a definite insightful expense model dependent on likely impacts of clones on various upkeep exercises. The current models make a few understood and express suppositions and don't give solid qualities for loads remembered for the formulae. Generally speaking, we realize too minimal about the genuine expenses caused by clones and the dangers and advantages of refactoring and different measures to repay the negative impacts of clones for a reasonable expense model. We scarcely realize the components impacting the expenses. Just through a progression of observational field studies and investigations will we at any point draw nearer to a particularly cost model. We stay distrustful with regards to whether we will at any point draw near sufficient given the numerous factors affecting the expenses and gains of clones.

VIII. Scope of Clone Management Activity

An occurrence of clone the executive's movement might be clone centered or framework cantered. A clone center action manages a limited arrangement of clones of a specific code fragment of interest. In

actuality, a framework centered clone the board action intends to manage an expansive assortment of clones in the whole code base, or specific segments of the framework. We need to additionally research for which sort of occasions clone the executives ought to be set off by whom. For changes in clones – specifically conflicting ones – likely a framework ought to advise an engineer. The test for all activities triggered by a framework should be to keep away from bogus cautions. In any case designers will before long quit any preteens of utilizing a clone the executive's instrument. For general quality affirmation, a quality administrator may notice patterns in clones and take activities when she sees an expansion of excess. The test for human set off activities is to give exact information on request and to discover huge markers of issues.

IX. MODERN ADOPTION OF CLONE MANAGEMENT

In spite of the dynamic examination on programming clones and their sway on the turn of events and upkeep of programming framework, the board of code clones is still a long way from wide mechanical reception. Motivation to this could be the inaccessibility of incorporated device support for flexible clone the board. Or then again perhaps industry is simply not mindful of the issue. Possibly clones are even not a genuine issue in any case in light of the fact that the benefits exceed the burdens. What we as specialists need to show initially is adequate experimental proof of genuine issues brought about by clones. We have gained great headway as of late here. At that point we need to favorable to vide usable working arrangements. We need to show their advantages in genuine contextual investigations. Since benefits are relied upon to appear just over the long haul, we need long haul concentrates in practical mechanical settings. Such long-haul mechanical investigations are hard to direct, notwithstanding. In spite of these challenges, we see signs that clone oversee meant is building up speed in industry. There are a few clone identifiers accessible as Eclipse modules and as of late Microsoft presented a clone the executives highlight in Microsoft Visual Studio [10, 44]. There are a few other modern endeavors also [128, 135] including a new Dagstuhl workshop on the theme [01].

X. CONCLUSION

We summarize the state of the art along the various components of code clone the executives and degrees for additional upgrades. In spite of the fact that product clone research developed in the course of the most recent decade, most of the work zeroed in on the identification and examination of code clones. Contrasted with those, clone the board has acquired late interest because of its reasonable significance. Remarkably a few reviews [43, 27, 02, 40, 36] showed up in the writing, none of which zeroed in on clone the board, and in this manner a study on clone the executives were an ideal need. This paper presents a far-reaching review on clone the board and pin-focuses research accomplishments and extensions for additional work towards an adaptable clone the executives framework. At the key level, the dubiousness in the meaning of clones now and again causes troubles in formalization, generalization, making of benchmark information, just as correlation of procedures and apparatuses. A bunch of errands arranged definitions or scientific classifications can address these issues. The majority of the incorporated instruments have constraints in recognizing Type-3 clones, and the identification of Type-4 clones has still stayed an open issue. Also, the greater part of the examination on programming clones so far stressed clone investigation at various degrees of granularity. An assortment of strategies for the perception of clones and the development has been proposed. Shockingly, while clone investigation focuses to the

significance of considering legacy progressive system for extricating clone reengineering applicants, there is as yet insufficient perception backing to break down clones as for their reality in the legacy pecking order. Exploration on clone the board past discovery has generally been restricted to concocting procedures to recognize clones. While discovery is a need for clone the executives and numerous upgrades have been accomplished here, sifting and positioning important discoveries is as yet a significant test. It isn't yet clear what establishes a terrible clone that requires treatment. Nor is it adequately understood what sort of treatment (refactoring or different kinds of pay) works best under which circum-positions. For the awful clones, we need to direct underlying driver examination to all the more likely comprehend why they appeared and how they could be kept away from. The cutting-edge requests more exploration in semi-robotized apparatus support for clone refactoring and money saving advantage investigation of clone evacuation/refactoring. For coordinated clone the executives, JSync [41] offers a moderately wide arrangement of highlights contrasted with others. In any case, we see that the best in class is still a long way from incorporated device backing, and more is to be done towards an adaptable clone the board framework. Maybe, because of the inaccessibility of such devices, there isn't a lot of engineer driven ethnographic examinations on the examples of clone the executives practically speaking, just as on the convenience and adequacy of hardware support. This study uncovered such possible roads for further research to make a superior effect in the community.

XI. ACKNOWLEDGMENT

We heartedly express our sincere gratitude to Ms. Chavi Ralhan who guided us for explaining on critical aspect of topics related to the research and useful discussion. We would like to thank all of the faculty member of all other respective departments for their intimate cooperation throughout the period of project completion.

I. REFERENCES

[1]. D. Baxter, M. Conradt, J. R. Cordy, and R. Koschke. Software clone management towards industrial application (dagstuhl seminar 12071). Dagstuhl Reports, 2(2):21–57, 2012.

[2]. D. Rattan, R. Bhatia, and M. Singh. Software clone detection: A systematic review. Infor. and Soft. Tech., 55(7):1165 – 1199, 2013.

[3]. E. Juergens and F. Deissenboeck. How much is a clone? In SQM, 2010

[4]. R. D. Venkatasubramanyam, S. Gupta, and H. K. Singh. Prioritizing code clone detection results for clone management. In IWSC, pages 30–36, 2013

[5]. B. Baker. On finding duplication and near-duplication in large software systems. In WCRE, pages 86 –95, 1995

[6]. S. Bouktif, G. Antoniol, M. Neteler, and E. Merlo. A novel approach to optimize clone refactoring activity. In GECCO, pages 1885–1892, 2006.

[7]. J. Harder. The limits of clone model standardization. In IWSC, pages 10–11, 2013.

[8]. C. Kapser, J. Harder, and I. Baxter. A common conceptual model for clone detection results. In IWSC, pages 72–73, 2012

[9]. Y. Yamanaka, E. Choi, N. Yoshida, K. Inoue, and T.i Sano. Apply- ing clone change notification system into an industrial development process. In ICPC, pages 199–206, 2013.

[10]. Y. Dang, D. Zhang, S. Ge, C. Chu, Y. Qiu, and T. Xie. XIAO: tuning code clones at hands of engineers in practice. In ACSAC, pages 369–378, 2012.

[11]. M. F. Zibran and C. K. Roy. A constraint programming approach to conflict-aware

optimal scheduling of prioritized code clone refactoring. In SCAM, pages 105–114, 2011.

[12]. I. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In ICSM, pages 368–377, 1998.

[13]. M. F. Zibran and C. K. Roy. Conflict-aware optimal scheduling of code clone refactoring: A constraint programming approach. In ICPC, pages 266 – 269, 2011

[14]. M. F. Zibran and C. K. Roy. Conflict-aware optimal scheduling of prioritized code clone refactoring. IET Software, 7(3), 2013.

[15]. S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo. Com- parison and evaluation of clone detection tools. IEEE Trans. on Softw. Engg., 33(9):577–591, 2007

[16]. J. Harder and N. Göde. Efficiently handling clone data: RCF and cyclone. In IWSC, pages 81–82. ACM, 2011

[17]. C. K. Roy and J. R. Cordy. NICAD: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization. In ICPC, pages 172–181, 2008

[18]. C. K. Roy and J. R. Cordy. A mutation/injection-based automatic framework for evaluating code clone detection tools. In ICSTW, pages 157–166, 2009.

[19]. T. Wang, M. Harman, Y. Jia, and J. Krinke. Searching for better config- urations: a rigorous approach to clone evaluation. In ESEC/SIGSOFT FSE, pages 455–465, 2013.

[20]. J. Svajlenko, I. Keivanloo, and C. K. Roy. Scaling classical clone detection tools for ultra-large datasets: An exploratory study. In IWSC, pages 16–22, 2013

[21]. R. Tairas and J. Gray. Phoenix-based clone detection using suffix trees. In ACM-SE, pages 679–684, 2006.

[22]. J. R. Cordy. Comprehending reality: Practical barriers to industrial adoption of software maintenance automation. In IWPC, pages 196–206, 2003.

[23]. R. K. Saha, C. K. Roy, and K. A. Schneider. An automatic framework for extracting and classifying near-miss clone genealogies. In ICSM, pages 293 –302, 2011.

[24]. M. F. Zibran and C. K. Roy. IDE-based real-time focused search for near-miss clones. In ACM-SAC, pages 1235–1242, 2012.

[25]. A. Santone. Clone detection through process algebras and Java bytecode. In IWSC, pages 73–74. ACM, 2011.

[26]. R. Komondoor and S. Horwitz. Using slicing to identify duplication in source code. In SAS, pages 40–56, 2001.

[27]. J. Pate, R. Tairas, and N. Kraft. Clone evolution: a systematic review. Journal of Soft.: Evol. and Proc., pages 1–23, 2011

[28]. J. Mayrand, C. Leblanc, and E. Merlo. Experiment on the automatic detection of function clones in a software system using metrics. In ICSM, pages 244 –253, 1996.

[29]. M. Rieger, S. Ducasse, and M. Lanza. Insights into system-wide code duplication. In WCRE, pages 100–109, 2004.

[30]. E. Duala-Ekoko and M. Robillard. Clone region descriptors: Repre- senting and tracking duplication in source code. ACM Trans. Softw. Eng. Methodol., 20:3:1–3:31, 2010.

[31]. Y. Higo, U. Yasushi, M. Nishino, and S. Kusumoto. Incremental code clone detection: A PDG-based approach. In WCRE, pages 3 –12, 2011.

[32]. Y. Higo and S. Kusumoto. Enhancing quality of code clone detection with program dependency graph. In WCRE, pages 315 –316, 2009.

[33]. R. Falke, P. Frenzel, and R. Koschke. Empirical evaluation of clone detection using syntax suffix trees. Empirical Software Engineering, 13:601–643, 2008.

[34]. S. Lee, G. Bae, H. Chae, D. Bae, and Y. Kwon. Automated scheduling for clone-based refactoring using a competent ga. Softw. Pract. Exper., 41(5):521–550, 2010

[35]. M. Fowler, K. Beck, J.Brant, W. Opdyke, and D. Roberts. Refactoring: Improving the Design of Existing Code. Addison Wesley, 1999.

[36]. C. K. Roy, J. R. Cordy, and R. Koschke. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. Sci. Comput. Program., 74:470–495, 2009.

[37]. S. Giesecke. Generic modelling of code clones. In DRSS, pages 1–23, 2007.

[38]. B. Lague, D. Proulx, J. Mayrand, E. Merlo, and J. Hudepohl. Assessing the benefits of incorporating function clone detection in a development process. In ICSM, pages 314–321, 1997.

[39]. X. Wang, Y. Dang, L. Zhang, D. Zhang, E. Lan, and H. Mei. Can I clone this piece of code here? In ASE, pages 170–179, 2012.

[40]. C. K. Roy and J. R. Cordy. A survey on software clone detection research. Tech Report TR 2007-541, Queens University, 2007.

[41]. H. Nguyen, T. Nguyen, N. Pham, J. Al-Kofahi, and T. Nguyen. Clone management for evolving software. IEEE Trans. on Softw. Engg., 1(1): 1–19, 2011.

[42]. M. F. Zibran, R. K. Saha, M. Asaduzzaman, and C. K. Roy. Analyzing and forecasting near-miss clones in evolving software: An empirical study. In ICECCS, pages 295–304, 2011.

[43]. R. Koschke. Survey of research on software clones. In DRSS, pages 1–24, 2006.

[44]. L. Jiang, G. Misherghi, Z. Su, and S. Glondu. DECKARD: Scalable and accurate tree-based detection of code clones. In ICSE, pages 96–105, 2007

[45]. Bellon, Stefan, Rainer Koschke, Giulio Antoniol, Jens Krinke, and Ettore Merlo (2007), "Comparison and evaluation of clone detection tools." IEEE Transactions on software engineering 33 (9).

[46]. Wagner, Stefan (2013), ”Software product quality control. Berlin ”Springer.

[47]. M. Mondal, C. K. Roy, and K. A. Schneider (2015), “A comparative study on the bug-proneness of different types of code clones,” in Proc. International Conference on Software Maintenance and Evolution (ICSME)IEEE: 91–100.

[48]. Monden, Akito, Daikai Nakae, Toshihiro Kamiya, Shin-ichi Sato, and Ken-ichi Matsumoto (2002), "Software quality analysis by code clones in industrial legacy software." In Software Metrics, 2002. Proceedings. IEEE Symposium (8):87-94.

[49]. Komondoor, Raghavan, and Susan Horwitz(2001), "Using slicing to identify duplication in source code." International Static Analysis Symposium. Springer, Berlin, Heidelberg: 40-56.

Cite this article as :

Chavi Ralhan, Rakesh Bishnoi, Ankit, Anjali, Hitesh Kumar, "Analysis of Software Clones", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 2, pp. 439-450, March-April 2021. Available at doi : https://doi.org/10.32628/CSEIT217290
Journal URL : https://ijsrcseit.com/CSEIT217290