THINK INDIA JOURNAL

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Enhancing Code Quality By Clone Management Framework

Geetika Chatley
Department of Computer science and engineering
Lovely Professional University,
geetikachatley@gmail.com

Abstract-The rise of an Open and free Source Software (OFSS) improvement model is highlyaccepted in the product development industry. Recently OFSS has become the interest area forseveral researchers in the direction of the software quality. Duplicate codes are injectedin the product framework by an OFSS engineers. This injected code helps and guide the clone investigation community to watch the impacts of duplicate codes on the program quality. Usually duplicate codes are not always unsafe, however they may get critical to large scale projects such as OFSS projects. This paper uncovers the significance and need for enhancing program quality of an OFSS projects by utilizing clone codes. Also, it representsa clone code advancement in OFSS projects. This frameworkwill shows a standard to begin the activities in the direction of enhancing a program quality by the execution of clone advancement framework for OFSS.

Keywords: clone management, duplicate codes, parsing, plagiarism, clone pairs

1. INTRODUCTION

Practices of reusing code are usually received by designers to improve efficiency. The vigorous implementation of code reuse practices during the improvement procedure leads to thesametype of code sections in a software package. Such same type of code sections are called as duplicate codes. Specialists accept that duplicate codes have double effect (great and terrible) on the support as well as development of programming frameworks [1], [2]. So it implies duplicate codes advantage designers by diminishing the creating exertion and time, however then again, they may tell code errors and lead with extended upkeep exertion in future. Fowler [3] distinguished duplicate codes as a "terrible stink" that might mess up programming primary maintenance. Other than this, code clones are not really unsafe and don't generally should be removed. In any case, they may be viewed as basic to the nature of anOFSS projects because of their advancement among variousoptions of the software projects after some time. Toward the beginning, they may not be unsafe; but with the passage of time and development they may turn into a genuine risk to sustainability and dependability.

Specialists recommend that it is a decent way to deal with recognized duplicate codes and detecttheir development crosswise over different forms of anOFSS schemes or projects. Clone

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

development following may uncover various examples and qualities showed by duplicate codes as they advance with time. In addition, duplicate code view procedures could be utilized to show the acquired duplicate code advancement data.

This paper emphasis that there is a requirement for a sustainable duplicate code advancement in OFSS improvement condition. Such a system guarantees code clone location and following among various forms of OFSS undertakings/projects and perception of the clone data in a way that can be useful for engineers to comprehend clone development. This data is basic to settle on practicality choices to enhance code nature of OFSS projects, making the OFSS program increasingly practical, recyclable, proficient, and logical.

In this paper, a system for successful duplicate code development following and representation in OFSS ventures is recommended which intends to enhance program quality of OFSS ventures/projects.

2. FREE/OPEN SOURCE SOFTWARE (OFSS) ADVANCEMENT MODEL

OFSS improvement model gives a stage to developers having a place with various topographical areas to cooperate in a community oriented advancement condition so as to make a financially savvy, increasingly proficient, and progressively dependable software [4]. In the ongoing scarcely any years, OFSS improvement model has risen as a broadly embraced worldview in the product advancement industry. It has ruled the acclaimed exclusive programming improvement models because of its capacities to deliver increasingly secure and top notch code which experiences continuous peer-survey procedure, and the receptiveness of program to program donors with various ability.

This broad acknowledgment as well as notoriety of OFSS improvement framework has constrained specialists to investigate different significant parts of OFSS ventures, above all the program quality. The program nature of anOFSS ventures has been under discussion in the examination network since quite a while, however with the headway of innovation new difficulties are being confronted that should be tended to. OFSS ventures are typically enormous programming frameworks involving various forms and having millions to a large number of code lines. They continue developing after some time so as to adapt to the changing client necessities.

OFSS advancement process acquires certain customary delicate product improvement

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

rehearses; code reuse is one of them. More often than not OFSS engineers embrace reusing code instrument to expand profitability and to accelerate the advancement procedure. Code reuse instrument is cultivated by duplicating current scraps of program as well as sticking them at some other area in the program in the wake of presenting minor or significant changes to it. This prompts the presentation of comparative or about comparative bits of program in a product. This type of program sections coming about because of code reuse instrument are regularly alluded to as "code clones" by the exploration network. Likewise "code cloning" is utilized as an elective term to program reuse.

3. DUPLICATE CODES

There has been over too many years of analysis on duplicate code, still the meaning of duplicate code has stayed hazy, as can be viewed through the definition gave by Ira Baxter that duplicate codes are comparable as per a few benchmark of likeness [5]. There are a few duplicate code definitions given by scientists in the writing. As per [6], duplicate codes are code pieces which are comparative lexically, grammatically, or semantically. Roy et al. [7] propose that duplicate codes are comparable or even copied bits of code. Though Zibran [8] concedes that almost comparable parts of a program are additionally duplicate codes. In spite of the fact that there has been a great deal of discussion on code clones in the examination network, still the scientists can't think of an exact definition for duplicate code yet. It is on the grounds that the definitions gave by scientists are increasingly limited by the idea of devices and methods utilized or by the objectives and data requirements of the specialists .for instance re-figuring, logic understanding, virus restriction, and literary theft identification. That’s the reason, there is a requirement to give objective arranged duplicate code scientific classifications which will make further investigation all the more clear in the unique circumstance as well as motivation behind the duplicate code explore.

4. Varieties of duplicate codes or clones

There are diverse duplicate code scientific classifications presented in the writing. The scientific categorization presented by Bellon et al. [9] is generally acknowledged by the examination network because of its effortlessness and clearness. This scientific categorization orders clones into four sorts, depicted beneath:

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

1) Type-one Clones:A piece of program parts that are comparative aside from certain adjustments in space and remarks.

2) Type-two Clones: A piece of program parts that are comparative in structure and language structure aside from some slight varieties in recognizers as well as in their sorts, and general code design.

3) Type-three Clones: A piece of program pieces that are near-miss duplicate codes alongside included, erased or changed explanations.

4) Type-four Clones: Semantic clones are increasingly refined all things considered code parts are distinctive in structure and linguistic structure however present a similar usefulness.

5. Clone Evolution

Duplicate codes are utilized and developed with included usefulness and changes for the duration of the existence of a product framework. An investigation of this type of life cycle wonders of duplicate codes is called as duplicate code advancement [6]. Scientists dissect how program in duplicate codes advance or is evolved from one rendition to the next. So as to comprehend duplicate code transforms, they need to identify and followduplicate codes among different variants of programming framework. The development and investigation of duplicate codes includes the utilization of different apparatuses as well as strategies combine with duplicate code location, mapping of clones, duplicate code designs distinguishing proof, duplicate code heredity arrangement, duplicate codes ancestry extrication, and duplicate code perception. The writing contains numerous investigations on clone advancement [6],[10], [11], [12], [13], [14], [15], [16], involving precise audits, studies, structures, devices and systems assessment. In any case, with the latest patterns in programming improvement industry, the writing requires more examinations that catch latest patterns in duplicate code advancement.

6. MANAGING CLONES

Clone administrators is worried about dealing with all duplicate code related exercises, for example, duplicate code recognition,removal of clones [17]. It contains an immense scope of exercises including location, following, investigation, perception, and restructuring of duplicate codes. Its principle center is to adequately as well as proficiently oversee duplicate codesso as to evade their unfriendly impacts on the feasibility and nature of programming frameworks.

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

6.1.Activities for managing clones

Clone the board envelops a scope of exercises that mean to oversee duplicate codes all the more viably and effectively. There are diverse duplicate codeexecutives' exercises distinguished by scientists in the writing. Roy et al[18] distinguished eleven clone the executives' exercises, for example, duplicate codePrevention, duplicate code Detection, duplicate code Documentation, duplicate code Tracking, duplicate code Visualization, duplicate codeAnalysis, duplicate code Annotation, duplicate code Recommendation, duplicate code Restructuring, duplicate codeRestructuring Scheduling, and Clone Restructuring Validation. Basit et al. [19] talked about 4 clone the executives' exercises, for example, duplicate code Assessment and Ranking (AR), Clone Awareness (CA), Clone Compensation (CC), and Clone Prevention (CP). These exercises can be consolidated to define a work process for a duplicate code the board framework as delineated by Roy et al. [18].

6.2.Detecting Clones

Duplicate code location is the most essential duplicate code the board movement that includes the recognition of duplicate code in the source program. The writing identified with duplicate code is for the most part contained the devices and methods formulated for duplicate code recognition. There are data rich reviews, for example, [7], [20], [9], [21] that include itemized talks about clone recognition and the comparing apparatuses as well as systems. Rattan et al. [20] arranged the duplicate code discovery systems into 7 kinds that incorporate on the basis of token, on the basis of content, diagram and on the basis of tree, measurements and on the basis of model, and furthermore crossover procedures for duplicate code location. OFSS venture includes exceptionally visit modifications in the source program, which might prompt the presentation of latest duplicate codes or adjust the current ones. Duplicate code identification in the developing programming frameworks can be done utilizing 2 methodologies named as duplicate code re-recognition and duplicate code gradual discovery [22].

1) **Rediscovery of Duplicate Codes:** It includes identifying duplicate codes in all variants of a product framework each time the latest form is again rented. It bolsters responsive duplicate code the executives. This methodology might have an excess of burden and can be computably expensive for enormous programming frameworks.

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

2) Incremental Detection of Duplicate Codes: This includes the detection of duplicate codesuch that lone the adjusted program in the latest form is reviewed for any new duplicate codes or duplicate code modifications and the outcome is amassed with the past duplicate codediscovery outcomes. It underpins actively clone the board. This methodology is increasingly helpful when contrasted with re-location as far as execution and effectiveness. There are a not many examinations [15], [23], [24], [25] in the writing concentrated on steady clone recognition.

3) Tracking Of Clone Evolution: Duplicate code development following is a duplicate code the executives' movement that guarantees the following of code duplicate codemodifications among different versions of a product framework by mapping duplicate codes, duplicate code advancement designs recognizable proof and duplicate code lineage extraction. Duplicate codefamily history is the investigation of how duplicate code sections in a duplicate code bunch develop through various renditions of the sourceprogram during the advancement of the product framework. The current writing incorporates ponders [2], [11], [13] covering different parts of clone advancement following. Be that as it may, the writing is still a long way from concentrating on significant viewpoints, for example, incorporation of clone advancement following inside the product improvement life cycle, for example, OFSS advancement.

4) Visualizing Clones: The excellent and complicated information created by duplicate code identification and duplicate code following in huge advancing programming frameworks is difficult to comprehend by programming experts and engineers. The utilization of duplicate code information in different perspectives in programming advancement and support has expanded to the degree that requests approaches to make it progressively sorted out, justifiable and far reaching. Clone perception is a clone the board action that guarantees the visual portrayal of intricate and voluminous clone data in a manner that is increasingly justifiable by the clone investigators. It permits clone experts to utilize those visualizations in settling on significant choices. There are different kinds of perceptions proposed in the writing that center different parts of clones, for example, clone type, level of likeness, clone area, clone scattering, clone size, clone effect, and clone development. Basit et al. [26] displayed a study on objective arranged clone representation of clone information. Along these lines review relates client objectives, data needs, and various kinds of perceptions tending to those necessities. In addition there are additionally different investigations in the writing dependent on clone

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

perception centering its different viewpoints [27], [22], [28].

7. RELATED WORK

Programming duplicate codes have been an intrigued region of analysis for over 2 decades. The writing on programming duplicate code research can be classified into 4 significant sub-regions of programming duplicate codes explore including recognition,studying, the board and device assessment. Over the whole length from 1994 to 2016, a large portion of the writing includes look into take a shot at discovery and investigation of clones, deserting clone the board and instrument support for assessment with not many commitments. In any case, clone the executives has progressed and turned into an unmistakable research zone in the earlier years. The perceived significance of study in the zone of duplicate codethe executives, prompts the requirement of future study here. The writing research stick point 3 significant parts of duplicate code the executives that require consideration by duplicate code explore network at display and can get upset the zone of duplicate codelook into in future. These are coordinated clone the board [1], [18], objective arranged duplicate codethe executives, and programming improvement on the basis of model duplicate code the board. This segment quickly depicts the commitments of code clone inquire about network in the territory of duplicate code advancement following and duplicate code perception. Wang et al. [28] displayed a methodology for following duplicate code modifications thatperforms ID of duplicate codeadvancement examples and duplicate code parentage extrication. They presented a structure for duplicate codefamily history extrication that performs duplicate code location utilizing FCD, at that point utilizes the subsequent duplicate codebunches for mapping of clones and distinguishing proof of duplicate codeadvancement designs so as to separate clone parentage. They actualized two apparatuses dependent on the presented structure named clone lineage extricates cGen and clone ancestry visualizer cGenShow. These devices are probed eighteen stable renditions of a product framework known as Bluefish. The devices are competent to get duplicate code modification data that can be utilized to follow duplicate codemodification designs, modification recurrence and change characteristics. At present, these instrument manage just type-one and type-twoduplicate codesand bolster programming frameworks created in C programming language.

Gode and Koschke [15] exhibited a gradual duplicate code identification calculation. This

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

calculation performs duplicate code recognition dependent on the aftereffects of the past variants examination. A device named iClones, a steady duplicate codefinder, was created dependent on this calculation. This device performs duplicate codelocation dependent on duplicate code change data got from mapping of clones in back to back forms of the product framework. They likewise assessed iClones and approved it to be proficient and valuable for duplicate codeadvancement following an investigation in oftentimes modifying programming frameworks when contrasted with different systems.

Saha et al. [29] presented a structure for programmed extraction of duplicate code family histories of both Exact and Near-Miss clones crosswise over various renditions of a product framework via robotized duplicate code change design distinguishing proof utilizing key similitude factors. They conceived a close miss clone genealogy extractor dependent on the presented system named gCad [2]. It is an apparatus that permits the extrication of correct and approach miss clone ancestries, at that point performs programmed distinguishing proofwhat's more, order of the change designs lastly gives significant clone advancement data utilizing measurements calculation. It underpins the investigation of clone advancement and encourages engineers to comprehend the development of clones so as to oversee them all the more productively. To approve the created model, they tested it on free/open source tasks, for example, Linux Kernel and a transformation/infusion based structure. This is versatile to different duplicate code identification apparatuses, with more accuracy and review.

Duala-Ekoko et al. [11] presented conceptual clone district descriptors (CDD) based method for duplicate code following in developing programming frameworks. They executed an overshadowing module dependent on this system, called Clone Tracker [13]. The apparatus consequently produces dynamic portrayals for clone areas, recognize changes to followed locales, and supports altering of those duplicate codedistricts. Their examinations uncovered that the utilization of Clone Tracker can permit following of a greater part of three thousand two hundred and seventy five duplicate code areas.

Bakota et al. [12] introduced an AI approach dependent on AST to follow the advancement of individual clone parts in enormous programming frameworks. They presented duplicate codedevelopment mapping so as to delineate parts of 2 unique adaptations of programming, utilizing a comparability extent. They additionally approved the handiness of strategy by utilizing it on the Mozilla Firefox program, where the method was effective to discover explicit

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

bugs because of disregarding prior reorder exercises.

Nguyen et al. [31] presented an innovative methodology for comprehensive duplicate code bunch the board in developing programming frameworks. This methodology depends on Cleman, an algorithmic structure. Cleman permits to create effective and precise clone bunch the board devices in a methodical way. They assemble a clone the board device dependent on the proposed edge work and afterward tested it on certifiable frameworks to test the adequacy of the methodology in steady identification and following of more excellent clone gatherings.

Kim et al. [16] exhibited a duplicate code family history model. In light of that model, they manufactured a duplicate codefamily history device that consequently removes the duplicate code history from a source program storehouse. They introduced duplicate code upkeep instruments that can utilize the separated duplicate code parentage data that can help with keeping up code clones. They additionally played out an observational examination [10] of duplicate codelineages. They utilized the created device to extricate cduplicate code family history data for 2 Java open source activities, ditty as well as dns java and broke down their development.

Asaduzzaman et al. [27] created VisCad, based on Java far reaching code duplicate codeinvestigation and representation device that provides the close miss mixture duplicate code identification device, NiCad. Utilizing deliberately chosen measurements and perception strategies, VisCad can manage clients about cloning in a product framework from alternate points of view. The instrument has been probed the Linux Kernel framework and the expertise recommends that this is powerful for breaking down and picturing duplicate codesin any event, for enormous frameworks.

8. PROPOSED FRAMEWORK

The proposed system for clone administrators in OFSS projects is conceptualized in Figure 1. It gives an incorporated clone the executives approach concentrating on fundamental clone management exercises for the duration of the existence cycle of OFSS venture advancement. The current systems and applications (discussed in writing survey) doesn't give a real existence cycle clone the board arrangement yet are intended to offer individual clone the board exercises, for example, clone identification or clone examination or clone perception. Combination with a remote code base utilized for OFSS ventures (for instance, GitHub) is

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

another component of this system. The proposed system is likewise expected to potentially deal with all clone types. The greater part of the current applications give answers for just a couple of explicit clone types. For instance [29] works just for type-1 and type-2 clones [2], dissects type-2 and type-3 clones in particular. Additionally there is no clone the board application or structure intended to suit the requirements of OFSS improvement model. The significant segments characterizing the proposed structure are clarified in the accompanying sub-areas.

1. Version Control System (VCS)

VCS is a normal part of anOFSS improvement model. Each new amendment of source code focused on a repository by OFSS code contributor. VCS in furthermore designed to duplicate every single ongoing version to the OFSS program.

Integrated Development Environment (IDE)

Integrated Development Environment is the principle part of anOFSS improvement model. It permits code supporters of compose and oversee piece of program. It is associated with the VCS which permit OFSS piece of program supporters of offer program in a circulated improvement condition. At the point when another rendition is refreshed by any program donor, other program supporters can duplicate the source program of the new form from ace task storehouse in their neighborhood venture archive to utilize it in their IDEs. Besides, when a program supporter changes program, at that point he refreshes this amendment of program as another rendition to the ace undertaking storehouse, from where other program patrons can duplicate this new form to their nearby vault to utilize it in advance in their IDEs.

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

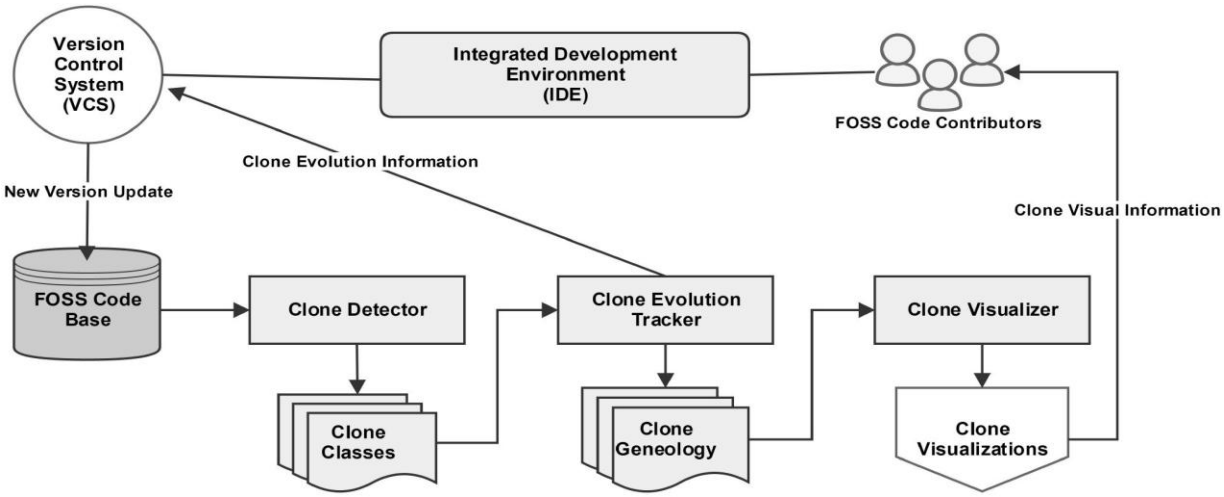

Fig. 1. Clone Management Framework for OFSS Project

2. OFSSProgram Base

OFSSprogram base is a repository of various versions of OFSS venture/project. It is designed to refresh consequently on arrival of another version from OFSSprogram contributors. It is maintained so as to give stable OFSSproject versions to perform duplicate code identification and duplicate code development tracking.

3. Clone Detector

This segment permits identification of code clones among various versions of OFSS projects. It recognizes duplicate codes in OFSS program base and creates duplicate code classes as an outcome.

4. Clone Evolution Tracker

This segment permits following duplicate code development by separating duplicate code parentages utilizing duplicate code mapping and duplicate code modify designs ID in various adaptations of OFSS projects. It takes duplicate code classes as an information and creates duplicate code genealogy.

5. Clone Visualizer

This section permits picturing the duplicate code development information in a manner progressively reasonable by program contributors. So the program contributors can settle on

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

significant choices dependent on this data. It takes duplicate code genealogy as an information and creates duplicate code representations.

6. PERCEIVED BENEFITS FOR THE DEVELOPER COMMUNITY

This structure significantly highlights on the requirement of an undeniable clone administrative framework for an OFSS assignment advancements. At present, the engineers use different methods and techniques to perform clone administration activities. The OFSS engineer community requests a clone maintenance framework that offers sustainable programming support and development. OFSS system is explicitly designed keeping in mind the difficulties of OFSS improvement environment and will fulfill the requirements of OFSS code supporter in every single possible way.

7. CHALLENGES

The difficulties looked during usage of a system are obvious. The huge difficulties distinguished in understanding the proposed system are featured underneath:

A. Available Tools and Techniques

The appropriation of accessible methods, for example, with the end goal of clone recognition and perception would require an intensive information on their abilities and complications. For instance, there are numerous clone discovery techniques and methods proposed by duplicate code analysts. It will challenge to pick one of them or make the structure to adjust to various discovery techniques and procedures.

B. Integration

The mix of various segments determined in the system can be a testing project.Besides, the adjustment in separate methods and strategies utilized as segments of the proposed structure can be of risk to the strength of the whole system.

C. Scalability and Performance

The proposed structure may confront execution and scalability issues in the selection of accessible devices and techniques for huge and developing frameworks, for example, OFSS ventures/projects.

D. Compatibility

There may be similarity issues associating the IDE, VCS, and programming language. It is beyond the realm of imagination to expect to investigate an execution that supports every

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

single part of the integral segments of the proposed structure.

E. Adoption

The program developers will need sufficient information and mastery in the utilization of fused instrumentsand methods for the selection of the proposed structure.

8. CONCLUSION

This paper has represented a clone management frameworkfor effective clone evolution tracking. This framework helps in visualization of OFSS projects based on the idea of duplicated code management. The proposed structure guarantees the enhanced program quality of OFSS projects. Also, it encompassesclonemanagementthroughoutthelife cycle of an OFSS project. The paper also outlines potential challenges that may be faced during implementation of the framework. In addition, this research greatly emphasizes on the need of integrated tool support for effective clonemanagement in OFSS projects and invites software clone community for collective efforts in thisregard.

REFERENCES

[1] M. F. Zibran, "Towards implementation of an integrated clone management infrastructure", in *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 3. IEEE, pp.60–61, *2016.*

[2] R. K. Saha, C. K. Roy, and K. A. Schneider, "gcad: A near-miss clone genealogy extractor to support clone evolution analysis", in *IEEE International Conference on Software Maintenance*. IEEE, pp. 488–491, 2013.

[3] F. Martin *et al.*, "*Refactoring: improving the design of existing code"*. Pearson Education India,1999.

[4] I. Ahmed, S. Ghorashi, and C. Jensen, "An exploration of code quality in foss projects", in *Open Source Software: Mobile Open Source Technologies*. Springer, pp.181–190, 2014.

[5] I. D. Baxter, A. Yahin, L. Moura, M. S. Anna, and L. Bier, "Clone detection using abstract syntax trees," in *Software Maintenance Proceedings, International Conference on*. IEEE, pp.368–377, 1998.

[6] J. R. Pate, R. Tairas, and N. A. Kraft, "Clone evolution: a systematic review",*Journal of software: Evolution and Process*, vol. 25, no. 3, pp. 261–283,2013.

[7] C. K. Roy and J. R. Cordy, "A survey on software clone detection research", Technical Report 541, Queens University at Kingston, Tech. Rep.,2007.

[8] M. Zibran and C. Roy, "The road to software clone management: A survey",*Dept. Computer. Sci., Univ. of Saskatchewan, Saskatoon, SK, Tech. Rep*, vol. 3,2012.

[9] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo, "Comparison and evaluation of clone detection tools",*Software Engineering, IEEETransactionson*,vol.33,no.9,pp.577–591,2007.

[10] M. Kim, V. Sazawal, D. Notkin, and G. Murphy, "An empirical study of code clone

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

genealogies", in *ACM SIGSOFT Software Engineering Notes*,vol.30,no.5.ACM,pp.187–196, 2005.

[11] E. Duala-Ekoko and M. P. Robillard, "Tracking code clones in evolving software",in*SoftwareEngineering.ICSE2007.29thInternational Conference on*. IEEE, pp.158–167, 2007.

[12] T. Bakota, R. Ferenc, and T. Gyimothy, "Clone smells in software evolution", in *Software Maintenance. ICSM 2007. IEEEInternational Conference on*. IEEE, pp.24–33, 2007.

[13] E. Duala-Ekoko and M. P. Robillard, "Clonetracker: tool support for code clone management", in *Proceedings of the 30th international conferenceonSoftwareengineering*.ACM,pp.843–846, 2008.

[14] H. A. Nguyen, T. T. Nguyen, N. H. Pham, J. Al-Kofahi, and T. N. Nguyen, "Clone management for evolving software", *Software Engineering,IEEETransactionson*,vol.38,no.5,pp.1008–1026,2012.

[15] N.Gödeand R.Koschke,"Studyingcloneevolutionusingincremental clone detection",*Journal of Software: Evolution and Process*, vol. 25, no. 2, pp. 165–192,2013.

[16] M. Kim and D. Notkin, "Using a clone genealogy extractor for under- standing and supporting evolution of code clones", in *ACM SIGSOFT SoftwareEngineeringNotes*,vol.30,no.4.ACM,pp.1–5, 2005.

[17] S. Giesecke, "Generic modelling of code clones", in *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum fr Informatik,2007.

[18] C. K. Roy, M. F. Zibran, and R. Koschke, "The vision of software clone management: Past, present, and future (keynote paper)", in *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), Software Evolution Week-IEEE Conference on*. IEEE, pp. 18–33, 2014.

[19] H. A. Basit, M. Hammad, S. Jarzabek, and R. Koschke, "What do we needtoknowaboutclones?derivinginformationneedsfromusergoals", in *Software Clones (IWSC), IEEE 9th International Workshop on*. IEEE, pp.51–57, 2015.

[20] D. Rattan, R. Bhatia, and M. Singh, "Software clone detection: A systematic review," *Information and Software Technology*, vol. 55, no.7, pp. 1165–1199,2013.

[21] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach",*ScienceofComputerProgramming*,vol.74,no.7,pp.470–495,2009.

[22] M. F.Zibran, "Analysis and visualization for clone refactoring", in *Software Clones (IWSC), IEEE 9th International Workshop on*. IEEE, pp.47–48, 2015.

[23] B. Hummel, E. Juergens, L. Heinemann, and M. Conradt, "Index-based code clone detection: incremental, distributed, scalable", in *Software Maintenance (ICSM), IEEE International Conference on*. IEEE, pp.1–9, 2010.

[24] Y. Higo, U. Yasushi, M. Nishino, andS.Kusumoto,"Incrementalcode clone detection: A pdg-based approach", in *Reverse Engineering (WCRE),18thWorkingConferenceon*.IEEE,pp.3–12, 2011.

[25] T. T. Nguyen, H. A. Nguyen, J. M. Al-Kofahi, N. H. Pham, and T. N. Nguyen, "Scalable and incremental clone detection for evolving soft- ware",in *Software Maintenance. IEEE International Conference on*. IEEE, pp.491–494, 2009.

[26] H. A. Basit, M. Hammad, and R. Koschke, "A survey on goal-oriented visualization of clone data," in *Software Visualization (VISSOFT), IEEE 3rd Working Conference on*. IEEE, pp.46–55, 2015.

[27] M.Asaduzzaman,C.K.Roy,andK.A.Schneider,"Viscad:flexiblecode clone analysis

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

support for nicad", in *Proceedings of the 5thInternational WorkshoponSoftwareClones*.ACM,pp.77–78, 2011.

[28] C.-H. Wang, Y. Tu, L.-P. Zhang, and D.-S. Liu, "Extracting clone genealogies for tracking code clone changes",*International Journal of SecurityandItsApplications*,vol.10,no.3,pp.21–28,2016.

[29] R. K. Saha, C. K. Roy, and K. A. Schneider, "An automatic framework for extracting and classifying near-miss clone genealogies", in *Software Maintenance (ICSM),27th IEEE International Conference on*. IEEE, pp.293–302, 2011.

[30] T. T. Nguyen, H. A. Nguyen, N. H. Pham, J. M. Al-Kofahi, and T. N. Nguyen, "Cleman: Comprehensive clone group evolution management", in *Automated Software Engineering, 23rd IEEE/ACM International Conference on*. IEEE, 2008, pp.451–454, 2008.