Investigating Developers Prevalent Copy and Paste Activities and validating associated Cloning Patterns

Sarveshwar Bharti
and
Hardeep Singh
Guru Nanak Dev University, Amritsar, India

To reuse available functionality, software developers usually accomplish copy and pasting of the source code. Empirical evaluations have established this activity as a prevalent cloning activity that acts as the main cause of the presence of clones into the software system. Analysis of these software systems has revealed various frequent cloning patterns, mainly classified into four categories viz. templating, forking, customization, and exact match. There are various tools that deal with clones proactively within the Integrated Development Environments. For a tool to be effective, requires the capturing of developers intentions. In this paper, we attempt to identify the hidden intentions of the software developers behind cloning practices via an online industrial survey involving professional software developers. This work will shed a light on what computer programmers are doing while reusing code and why. This study uncovers various intentions, extent, source etc. of the copy and paste activity done by the software developer. The observational determinations from this survey were then employed to validate different cloning patterns based on the developers perspective. Finally, based on the survey results, the solution to these cloning practices is discussed. Results depicted in this paper suggest an incorporation of these developers behavioural inference into the existing IDE based clone management systems.

Keywords: Code Clones, Copy and Paste Activity, Developer Intention, Cloning Patterns.

1. INTRODUCTION

The aim of Software Engineering discipline is to enhance the software quality and minimize the development and maintenance cost of the software system. The literature mentions that code clones have a direct impact on the quality and maintenance cost of the software (Roy and Cordy [2007] , Baker [1995], Kamiya et al. [2002], Rattan et al. [2013], Johnson [1994]). Thus, this field of study on software clones has now become one of the mature areas of research under the software engineering domain. There are different ways to define a clone based on different perspective, but, the definition accepted by the research community was given by Baxter et al. [1998], stating a clone is a program fragment that [is] identical to another fragment. Four different types of clones viz. Type 1, Type 2, Type 3 and Type4 have been classified based on syntactic and semantic similarity by Bellon et al. [2007]. Over the decades of research on software clones, a number of different reasons have been identified by the research community that induces clones into the software system. There are various unintentional as well as intentional reasons behind the induction of code clones, as listed by Roy and Cordy [2007]. Copy and paste reuse approach is one of the main ways that make a duplicated code fragment to be present in the software system. Software reuse involves utilizing the predefined software components in the development of software systems. These software systems can be reused at different levels as per the requirement, viz. code block, function, libraries or at an entire subsystem level. Reusing software components decreases software development time and thus software development cost,

Authors' addresses: Sarveshwar Bharti, Hardeep Singh, Department of Computer Science, Guru Nanak Dev University, Amritsar, INDIA. Acknowledgements: Authors express their sincere gratitude to the UGC for Senior Research Fellowship to the first author and also would like to thank the Department of Computer Science, Guru Nanak Dev University, Amritsar for the academic and infrastructural support to the ongoing research.

as well as the maintenance cost, is reduced. Software reuse being beneficial may not always prove to be satisfactory, as in the case of software cloning, software developers under pressure of time, productivity etc. usually perform copy-and-paste reuse of the software mostly at the level of code blocks encapsulated within the opening and closing braces and then attempt to adopt the pasted code to meet the desired functionality, thus has the maximum possibility of bug insertion and propagation. These modifications are mostly local and unknown to other developers and maintenance engineers. The increase in the complexity of the software system due to the desired modifications applied by going beyond the initial software design makes it difficult to be understood by the maintenance engineers and thus increases the maintenance effort. Thus, this reuse approach seems to be desirable during software development but has negative implications in the long-term. Code duplication creates several problems viz. an increase in the size of the software system, difficulty in maintaining the already developed code, bug propagation, bug fixing etc. Thus, this programming practice of the developer necessitates being analyzed.

There are various studies (Juergens et al. [2009] etc.) that bring out the importance of studying this issue of software cloning. The after effects of the cloning pose challenges in the management of the software cloning and the code clones. Various researchers reveal the consequences of the cloning and thus gave the indication of the harmfulness of clones towards maintainability of the software system. Formally we can say that code clones co-evolve, i.e. a change to one cloned code fragment may or may not be needed to be propagated to the other copies of the same code fragment. This co-evolution incurs unnecessary maintenance effort and cost (Geiger et al. [2006], Thummalapenta et al. [2010]). 80% of the overall cost and effort is spent on maintenance during the software development life cycle (Thummalapenta et al. [2010]), thus software cloning should be considered as an important aspect that has a direct impact on one of the very important software quality attributes i.e. maintainability.

As per Smith [2012], in 90s out of total 100 billion Dollars spent on software, 70 billion Dollars were spent on software maintenance i.e. cost of maintenance was 70% with just 30% of new development cost. Out of the total 70% maintenance cost 60% cost is spent on locating bugs and rest 40% for everything else like fixing, testing, reviews, integration, deployment etc. As per Juergens et al. [2009], there is an overall 20% effort increase in software maintenance due to code clones. So these cost escalation statistics clearly point to the importance of research in this area of software cloning to minimize the maintenance effort and cost.

With decades of research in this field of software cloning various code clone detection, analysis and removal techniques are proposed. There are also different tools available that help in tracking developer activities viz. CloneBoard, CPC, CnP etc. These tools help in proactively managing code clones. Developer activities can be better tracked by knowing the possible intentions of the developer.

Thus, this study motivates from the fact that, to have a better copy and paste management and thus efficient proactive clone management, the programmer intentions must be taken into consideration while developing a clone management tool. This stimulus drives authors to conduct a study involving professional developers and gather the results for further dissemination. The work presented in this paper will put a light on what developers do and why, during coding.

Literature also mentions categorizations of clones based on the intent of the software programmer. Kapser and Godfrey [2008], presented four cloning patterns based on the analysis and empirical evaluation of the large software systems. These patterns include templating, forking, customization, and exact match. To the best of our knowledge, no study has performed the evaluation and thus validation of these cloning patterns based on the developers perspective. Thus it necessitates being validated based on the empirical evidence gathered from the survey involving software developers, which is the other main goal of this paper.

Thus, the main goal of this paper is to first identify the developers intentions behind the prevalent copy and paste code cloning activity and then utilizing the developers responses from the industrial survey, validation of the cloning patterns identified by Kapser and Godfrey [2008] is

conducted and finally seven key solutions for improving the software cloning practices are portrayed based on the survey results.
The main contributions of this paper are listed below:
Reasons for copy and paste activity
Source of code fragments copied
The extent of copy and pasting
Type of cloning
The solution for improving Software Cloning Practices
Validation of Categorization of Clones based on intent
Organization of the rest of the paper is as follows: Section 2 discusses the literature related to the present study along with the possible extension to it. Section 3 discusses the motivation behind this study and identifies various research questions that necessitate being answered. Section 4 presents the survey design and the adopted methodology. Section 5 presents survey results and then, findings are analyzed. Section 6 discusses the validation of various cloning patterns along with the analysis of results in Section 7, and Section 8 lists various solutions to improve software cloning practices. Section 9 points to the various threats to the validity of this study. Section 10 concludes this paper and then, finally, acknowledgments and then references in support of this paper are listed. The work presented in this paper is an extended version of the work Bharti and Singh [2017] reported in the Proceedings of the International Conference on Next Generation Computing and Information Systems.

2. RELATED WORK AND POSSIBLE EXTENSION

There are various studies found in the literature that surveyed programming practices for identifying intentions and thus reasons behind cloning, as listed in Table 1. In the year 2004, Kim et al. [2004] came up with an ethnographic study of copy and paste programming practices in Object-Oriented Programming Languages (OOPLs). They observed the developers and presented a taxonomy of copy and paste programming practice. It directly relates to our study, but we presented more detailed results and thus better inference can be drawn from our study. Cory [2006] in the year 2006 presented a survey depicting the harmfulness of clones. They observed developers and investigated several different patterns of cloning, but not copy and paste intentions of programmers. Roy and Cordy [2007] presented a detailed survey of research on software clone detection. They have touched almost every aspect of clone research using various studies, but not as of the study presented in this paper. Wani and Dang [2015], in the year 2015, surveyed effects of code cloning on software quality along with various reasons of cloning. They presented a broad perspective and not the one presented in the current study.

Reference	**Description**	**Possible Extension**
Kim et al. [2004]	An ethnographic study of copy and paste programming practices in Object-Oriented Programming Languages (OOPLs) was conducted	More detailed results from an industrial survey can be gathered to draw the better inference
Cory [2006]	Observed developers and investigated several different patterns of cloning	Investigation of developers intensions towards copy and paste programming activities can be done
Roy and Cordy [2007]	Presented a detailed survey of research on software clone detection	A dedicated study on software developers programming practices can be presented
Wani and Dang [2015]	Surveyed effects of code cloning on software quality along with various reasons of cloning	Instead of broad perspective a developer centric study can be performed

Table I: Description of Related Literature along with Possible Extension

As referenced in the previous section, some portion of this study has already been published in a conference proceedings Bharti and Singh [2017] and in another related work Bharti and Singh [2018] we have explored sentiments of the developers related with software cloning practices utilizing developers responses as a dataset for sentiment analysis.
Complementing all the above-mentioned literature, this paper presents a study in a different perspective involving the copy and paste programming practices by the developer using an industrial study and in a much-elucidated way.

3. MOTIVATION AND RESEARCH QUESTIONS

For developing a better clone management tool (more specifically copy and paste management tool), the associated intensions of programmers towards copy and paste activities should be taken into consideration. With the motivation from this fact, literature study discussed in the previous section revealed that there is a much scope for the possible extension of the related work as listed in Table 1. Thus, after going through the literature, various research questions were identified as listed in Table 2 that necessitates being answered to elucidate the intentions behind copy and paste practice and the type of activity itself.

Research Question No.	**Research Questions**
RQ1	Distribution of developers based on gender
RQ2	Distribution of developers based on working experience
RQ3	Do Software Developers Copy and Paste?
RQ4	What is the source of the copied fragments?
RQ5	What is the extent of cloning?
RQ6	What type of copy and pasting is done?
RQ7	What are the reasons for copy and pasting?
RQ8	Validation of Categorization of Clones based on intent
RQ9	How can we improve Software Cloning Practices?

Table II: Research Questions

4. SURVEY DESIGN AND METHODOLOGY

After identifying the Research Questions (RQs) as discussed in the previous section, a questionnaire was prepared and used to identify the developers intensions. To analyze the developer behavior, the authors conducted an online industrial survey. Developers from various development organizations were contacted via emails containing a link to the questionnaire created using Google Form. This paper will present the part of the online industrial survey, thus only those questions of the communicated questionnaire that are related to this paper are listed in Table 3. Out of the mentioned 5 questions in Table 3, 4 questions are close-ended, while remaining 1 is an open-ended question. Developers were asked to respond to the questions by filling the Google Form and the results were collected in Google Drive for further analysis. Figure 1 depicts the overview of the adopted methodology for this current study.

Question No.	**Question in the Questionnaire**
Q1.	When you are coding and you find that a particular code fragment you needed is already present somewhere in the code base, then you copy and paste that code fragment
Q2.	In your view, what is the reason for Copy and Pasting i.e. Code Cloning?
Q3.	From where you copy the code fragment?
Q4.	What is the extent of Copy and Pasting?
Q5.	What type of Copy and Paste i.e. Code Cloning you do?

Table III: Summary of Survey Questions

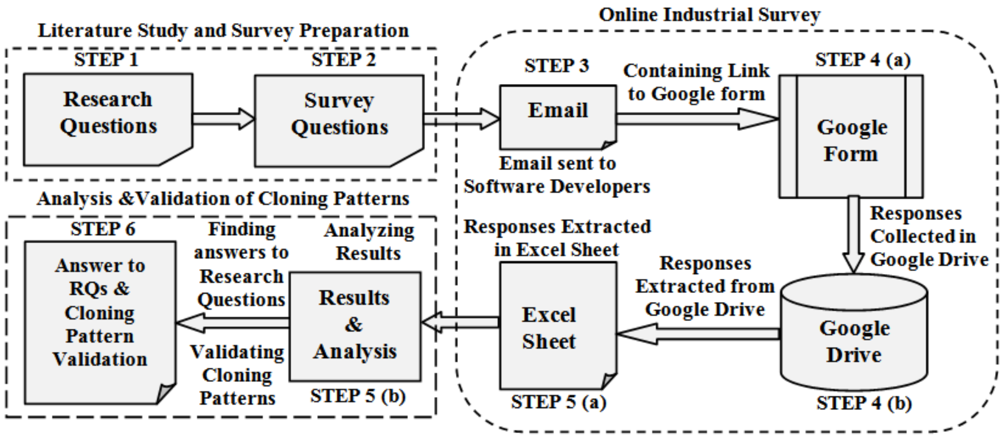
Literature Study and Survey Preparation
STEP 1
Research Questions
STEP 2
Survey Questions
Online Industrial Survey
STEP 3
Email
Containing Link to Google form
Email sent to Software Developers
STEP 4 (a)
Google Form
Responses Collected in Google Drive
Google Drive
STEP 4 (b)
Responses Extracted from Google Drive
Excel Sheet
STEP 5 (a)
Responses Extracted in Excel Sheet
Analysis &Validation of Cloning Patterns
Results & Analysis
STEP 5 (b)
Analyzing Results
Finding answers to Research Questions
Validating Cloning Patterns
STEP 6
Answer to RQs & Cloning Pattern Validation

Figure 1. Overview of the adopted methodology

5. SURVEY RESULTS AND ANALYSIS

Responses from the developers were captured via Google Form and collected in the Google Drive. This section will present the results from an online survey involving professional developers from different organizations and the analysis of findings towards answering research questions raised in Section 3. Survey results along with the answers to the research questions are discussed below:

5.1 Distribution of Developers based on Gender

We conducted this online industrial survey that involved 35% female software developers and 65% male developers as shown in Figure 2.

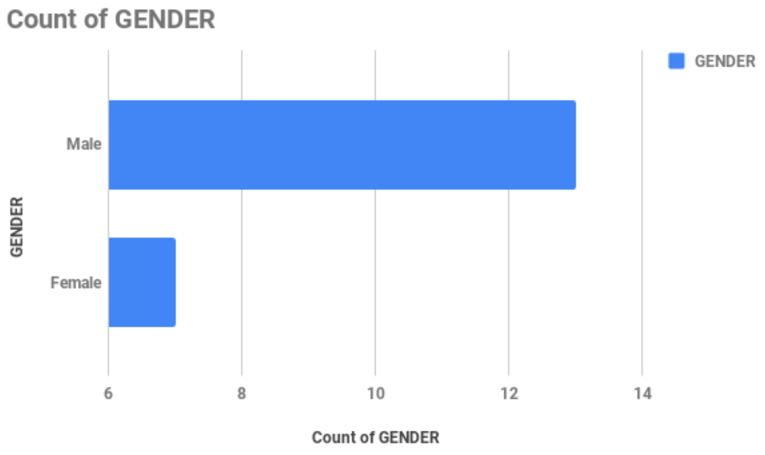

Figure 2. Distribution of developers based on Gender

5.2 Distribution of Developers based on Working Experience

This survey involved professional developers from various national and international firms with varying working experience as depicted in Figure 3. There were developers with just one year of developer experience up to five years of working experience as a code developer.

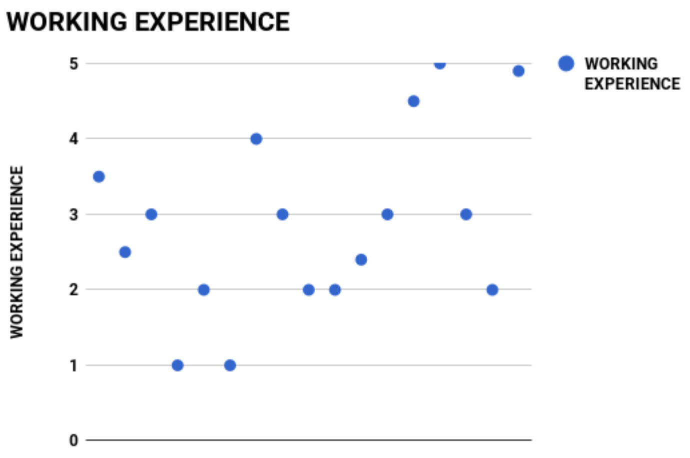

Figure 3. Distribution of developers based on working experience

5.3 Do Software Developers Copy and Paste?

To answer this question, a close ended statement was put in questionnaire mentioned in Table 3 at serial number 1 as; When you are coding and you find that a particular code fragment you needed is already present somewhere in the code base, then you copy and paste that code fragment. 85.0% of developers responded yes whereas 15% say no to this statement as shown in Figure 4. Inference can be drawn from this that programmers usually get involved in copy and paste activity.

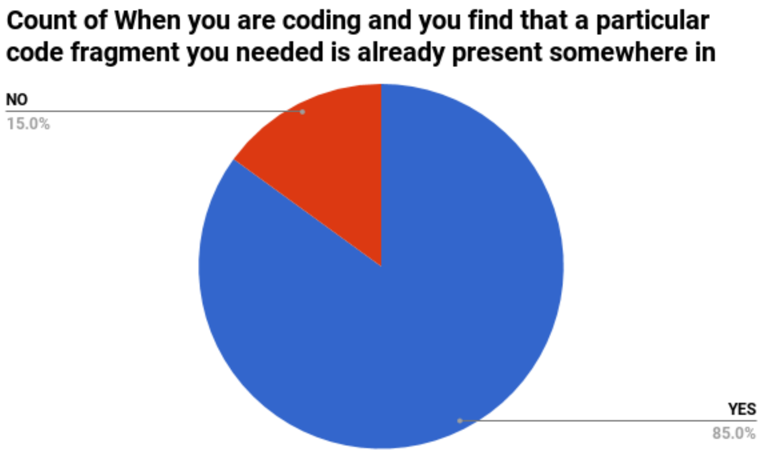

Figure 4. Percentage of developers that copy and paste

5.4 What is the source of the Copied Fragments?

Question number 3 of Table 3 was used in the questionnaire to investigate the source program, module, software etc. used as a parent file from which code fragment is copied. This was the close-ended question with five options and each option having a preference related to it. Table 4 presents the summary of the possible source of the copied code fragments. It is observed from the table that 68.8% developers mainly copy a fragment from the same program at first preference and 80.0% copy from other modules but at second preference. The inference drawn from this table can more clearly be visualized as depicted in Figure 5 and Figure 6.

Source/Origin	**1st Preference**		**2nd Preference**		**3rd Preference**		**4th Preference**		**5th Preference**	
	F	%age	F	%age	F	%age	F	%age	F	%age
From same program you are coding	11	68.8%	1	6.3%	1	6.3%	1	6.3%	2	12.5%
Other modules of same system	2	13.3%	12	80.0%	0	0.0%	1	6.7%	0	0.0%
Other software systems within organization	1	5.6%	4	22.2%	8	44.4%	4	22.2%	1	5.6%
From internet	4	22.2%	1	5.6%	4	22.2%	7	38.9%	2	11.1%
Other source	1	6.7%	1	6.7%	2	13.3%	2	13.3%	7	60.0%

Table IV: Summary of Source of Copied Fragments

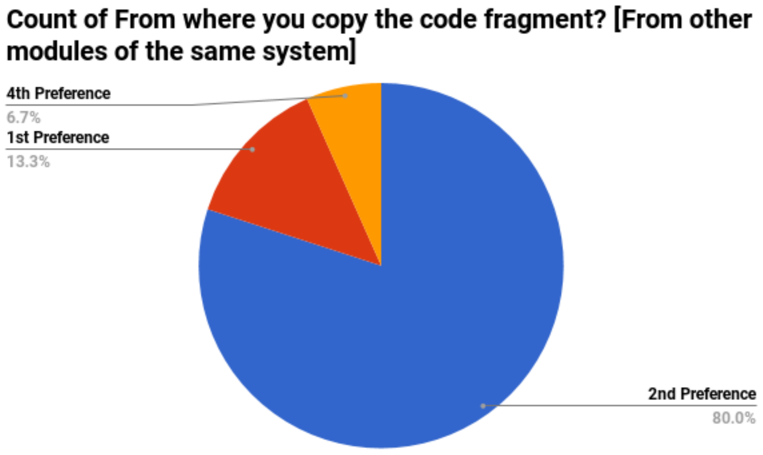

Figure 5. Percentage of developers that copy and paste from other modules of the same system

5.5 What is the extent of Cloning?

Copy and paste programming i.e. cloning can be done by copying just a single line, more than one line etc. To explore the extent of cloning, question number 4 of Table 3 was used in a questionnaire. Results collected from the developers are depicted in Table 5. This table elucidates at first preference 41.2% developers copy just a single line of the source code, 55.6% copy more than one line but at second preference and 64.7 copy whole module but at fourth preference. Observations from this table can be clearly visualized as presented in Figure 7 and Figure 8.

Extent	**1st Preference**		**2nd Preference**		**3rd Preference**		**4th Preference**		**5th Preference**	
	F	%age	F	%age	F	%age	F	%age	F	%age
One line	7	41.2%	1	5.9%	7	41.2%	2	11.8%	0	0.0%
More than one line	3	16.7%	10	55.6%	3	16.7%	2	11.1%	0	0.0%
Function as a whole	6	31.6%	7	36.8%	5	26.3%	1	5.3%	0	0.0%
Module as a whole	3	17.6%	1	5.9%	2	11.8%	11	64.7%	0	0.0%

Table V: Summary of Extent of Copy and Pasting of Code Fragments

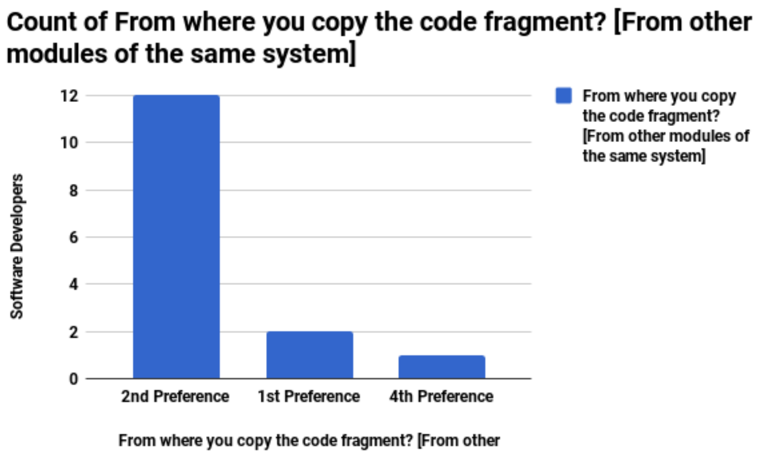

Figure 6. Count of developers that copy and paste from other modules of the same system (with preference)

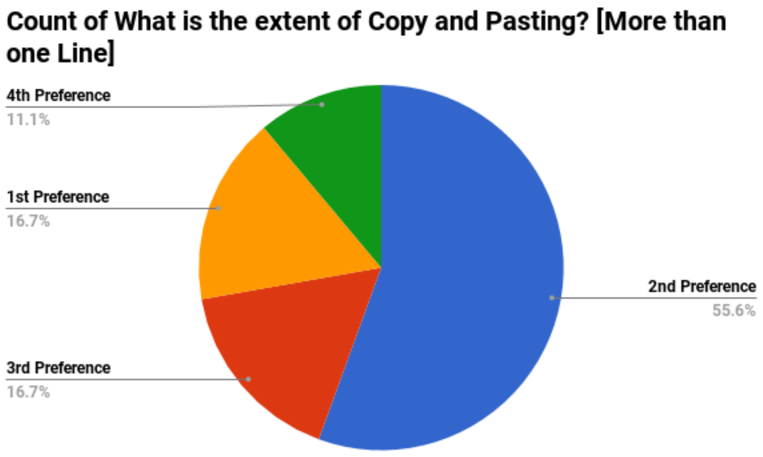

Figure 7. Percentage of developers that copy more than one line

5.6 What type of Copy and Pasting is done?

To answer this question, from Table 3, question number 5 was utilized to collect the responses from the developers. The code can be copied and then pasted with or without modifications, or used as a template or just the logic. Survey results are presented in Table 6 from which it is clearly observed that 40% of the developers strongly agree of using the logic of the code fragment whereas 80% agree of using code fragments with some modifications. Figure 9, 10 and 11 visualizes the observation drawn from this table.

5.7 What are the reasons for Copy and Pasting?

In a questionnaire, there was one open-ended question listed at serial number 2 of Table 3, which was utilized to gather the developer thinking about the reason behind the copy and paste activity. Developers listed reason according to their knowledge domain but most of the developers emphasized three main issues as listed below: Lack of knowledge Need to combine two modules The requirement of the system

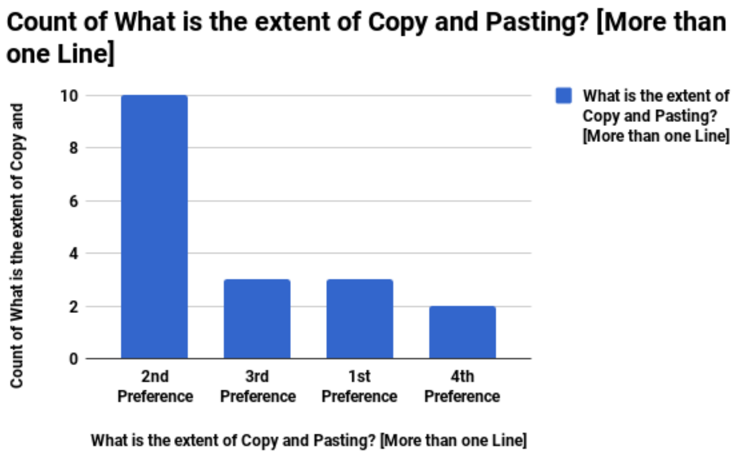

Figure 8. Count of developers that copy more than one line

Type of Code Cloning	**Strongly Agree**		**Agree**		**Cant Say**		**Disagree**		**Strongly Disagree**	
	F	%age	F	%age	F	%age	F	%age	F	%age
Copy code fragment and paste it without any modifications	2	10%	5	25%	3	15%	8	40%	3	10%
Copy code fragment and paste it with some modifications	4	20%	16	80%	0	0%	0	0%	0	0%
Copy code fragment and use it only as a template	4	20%	12	60%	3	15%	1	5%	0	0%
Just use the logic of the code fragment	8	40%	8	40%	1	5%	2	10%	1	5%

Table VI: Summary of type of Code Cloning done by the Developer

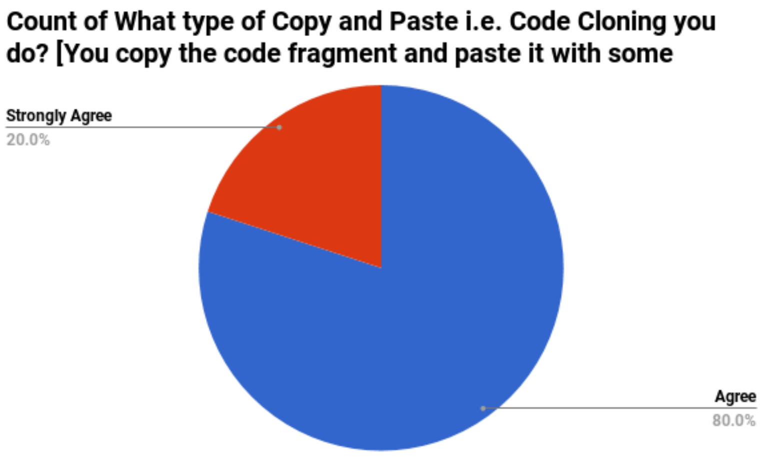

Figure 9. Percentage of agreement of developers that copy and paste code fragment with some modifications

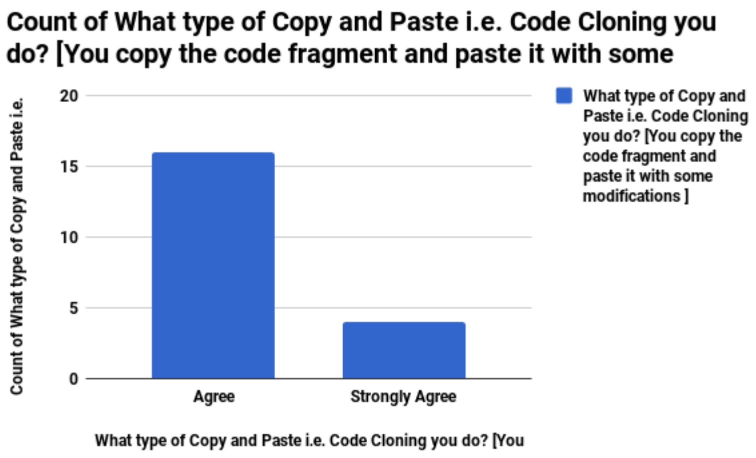

Figure 10. Count of developers that copy and paste code fragment with some modifications

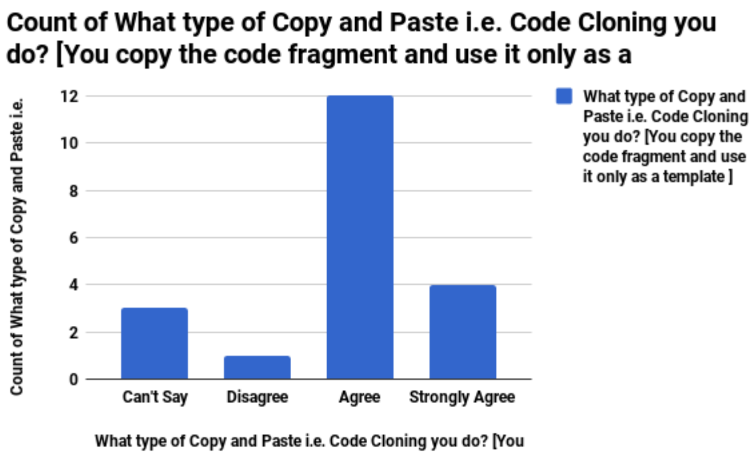

Figure 11. Count of developers that use code fragment as a template

6. CATEGORIZATION OF CLONES BASED ON INTENT

Based on the post hoc source code analysis of large software system, Kapser and Godfrey [2008] presented a categorization of clones, categorized into four main categories viz. Exact matches, Customization, Forking, and Templating. These categories basically depict the four divisions of high-level code cloning patterns and are the results of the quantitative research on software clones. Most of the research on software clones is a quantitative i.e. analysis of results based on clone detection technique applied. Literature reveals that very little emphasis has been given on qualitative analysis of code cloning. Thus to evaluate the applicability and relevance of the code cloning research to the software developers and maintenance engineers, in-depth and focused qualitative analysis of code cloning is necessary. The post hoc code analysis cannot depict the real intentions of the software developers and thus necessitate to be validated. Chatterji et al. [2013] conducted interviews of the software developers to understand their intentions to categorize code clones, but an effective empirical study has better possibility towards acceptance by the clone research community.

Thus based on the study discoursed in this paper, we present the validation towards confirm-

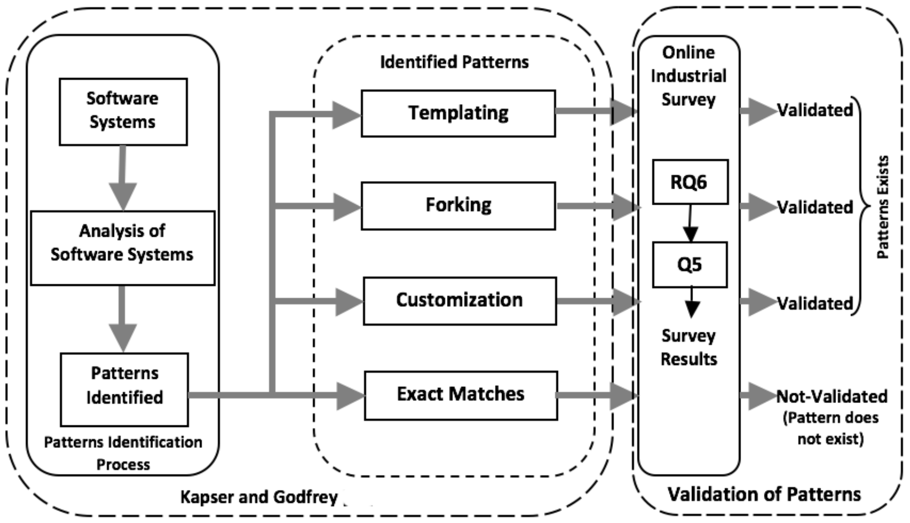

Figure 12. Validation framework

ing the presence of these cloning patterns in the software system from the software developers perspective, thus answering RQ8, listed in Table 2. The detailed validation framework adopted in this paper is presented in the Figure 12. Based on the empirical evidence collected from the survey on software developers, the detailed description of these categories is presented below:

6.1 Exact Matches

Programmers usually copy the code fragment and paste it with or without modifications where ever needed. When used without modifications, pasted code fragment may be categorized under exact match patterns. Kapser and Godfrey [2008] reported that exact matches often arises when a particular problem is repeated within the software system but is either too small to make the creation of an abstraction worthwhile or is incomplete when taken out of the context of its neighboring source code. To check the validation of this statement i.e. whether exact match code patterns are present in the software system or not and whether these patterns are introduced into the software system by the software developer or not, we acquire the results discussed for RQ5. For discussing results for RQ5, Q5 (as listed in Table 3) was put in the questionnaire communicated to the software developers. Summary of responses collected from the programmers is presented in Table 7.

Now, as per the source code analysis done by Kapser and Godfrey [2008], they have observed the

Cloning Pattern	**Agreement Level**	**Responses**	
		Frequency	Percentage
Exact Match (Copying of code fragment and pasting it without any modifications i.e. reusing the exact match code)	Strongly Agree	2	10.0%
	Agree	5	25.0%
	Cant Say	3	15.0%
	Disagree	8	40.0%
	Strongly Disagree	3	10.0%

Table VII: Summary of responses for Exact Matches Patterns

presence of exact matches in the code base and thus identified a cloning pattern Exact Match, but the analysis of the survey responses conducted by us depicts that there is total of 50% disagreement (with 10% strong disagreement and 40% disagreement) of software developers over the copying of code fragments and pasting them without any modifications as presented in Table

7. Thus we conclude that as per the software developers, mostly exact match patterns are not inducted into the software systems and if present, then may be introduced unintentionally.

6.2 Customization

It is a prevalent programming practice of the software developers to copy the desired code fragment and modify it according to the current need of the system under consideration. Kapser and Godfrey [2008] classified this type of patterns as customization patterns. Customization involves cloning of existing code and then performing tailoring according to the solution for the new problem. To check for the presence of this cloning pattern in the programming practice we utilized results gathered for RQ5 (as listed in Table 2) from the developers by utilizing the responses for Q5 listed in Table 3. Table 8 presents the summary of responses towards customization pattern.

In a study by Kapser and Godfrey [2008], it has been empirically established from the source

Cloning Pattern	**Agreement Level**	**Responses**	
		Frequency	Percentage
Customization (Copying of code fragment and pasting it with some modifications i.e. adopting the code according to the current need)	Strongly Agree	4	20%
	Agree	16	80%
	Cant Say	0	0%
	Disagree	0	0%
	Strongly Disagree	0	0%

Table VIII: Summary of responses for Customization Patterns

code analysis that these types of patterns are present in the source code. Based on the analysis of the developer responses, we also confirm and validate that these cloning patterns are frequently adopted during cloning practices, as there is total 100% agreement with 20% strong agreement and 80% agreement as presented in Table 8.

6.3 Forking

According to Kapser and Godfrey [2008], Forking is cloning used to bootstrap development of similar solutions. Forking patterns involve duplicating the source code then pasting it that act as the starting point of the further software development. Forking can be seen similar to the concept of using code fragment as a template for code development, where code template is copied and pasted with the intention that the pasted code will perform the task that is already tested and results in the desired solution. Close-ended question Q5 listed in Table 3 was utilized for presenting the developers intention associated with this type of cloning pattern. Summary of responses from the software developers in response to Q5 is presented in Table 9.

From Table 9, it is clear that 60% of software developers agree and 20% strongly agree over

Cloning Pattern	**Agreement Level**	**Responses**	
		Frequency	Percentage
Forking (Copying the code fragment and using it only as a template i.e. bootstrapping of the similar solution)	Strongly Agree	4	20%
	Agree	12	60%
	Cant Say	3	15%
	Disagree	1	5%
	Strongly Disagree	0	0%

Table IX: Summary of responses for Forking Cloning Patterns

using code fragments as templates. Thus confirming the findings of Kapser and Godfrey [2008] that software systems contain forking cloning patterns. From the results presented in this paper, we conclude that the forking patterns are inducted by the software developers during copy and paste cloning practices.

6.4 Templating

As per Kapser and Godfrey [2008], Templating is used as a method to directly copy behavior of existing code. Form the developers perspective templating can be seen as a practice of utilizing the known logic (i.e. behavior) of the source code. The answer to RQ5 via utilizing responses collected in Google Drive against Q5(listed in Table 3) was taken into consideration for validating this cloning pattern. The frequency and percentage of the responses from the developers in response to the templating cloning pattern are listed in Table 10.

Source code analysis performed by Kapser and Godfrey [2008] depicted the presence of templating patterns in the source code. In continuation of their findings, we also validate that templating patterns are surely present in the code base. As presented in Table 10, there is 80% total agreement of adopting this cloning practice with 40% strong agreement and 40% of the developer just agree over the use of the cloning pattern.

Cloning Pattern	**Agreement Level**	**Responses**	
		Frequency	Percentage
Templating (Copying of code fragment and pasting it without any modifications i.e. reusing the exact match code)	Strongly Agree	8	40%
	Agree	8	40%
	Cant Say	1	5%
	Disagree	2	10%
	Strongly Disagree	1	5%

Table X: Summary of responses for Templating Patterns

7. ANALYSIS OF RESULTS

Figure 13 presents the overall results for four cloning patterns viz. Exact matches, Customization, Forking and Templating (as discussed in Table 7, Table 8, Table 9 and Table 10 respectively). Comparative Analysis of results depicted in Figure 13 is presented in Figure 14 clearly depicting the strong agreement towards Customization, Forking and Templating patterns. Percentage of developers that agree over Customization, Forking and Templating is also higher than that of Exact matches. Thus the result presented in Section 6.1 depicting total of 50% disagreement towards Exact matches patterns is justified with this comparative analysis limned in Figure 14.

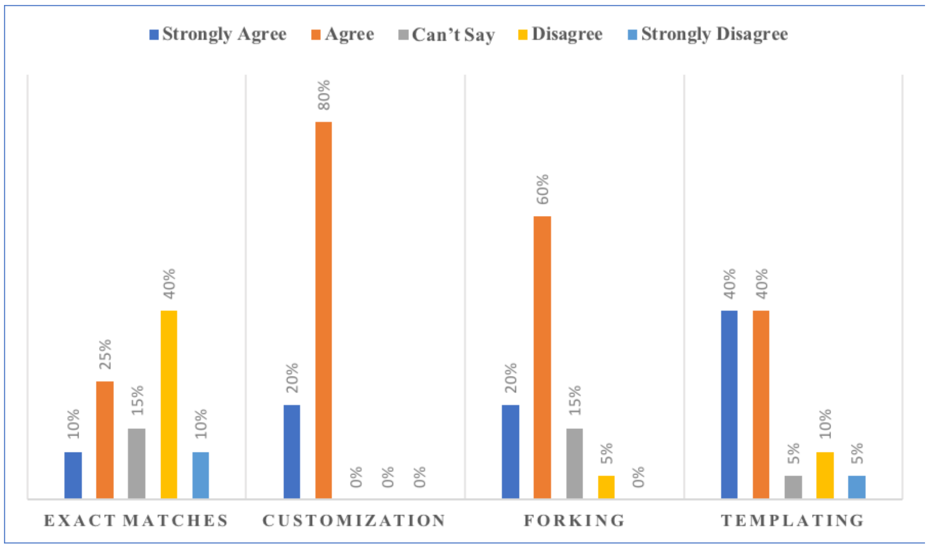

Figure 13. Overall results for four Cloning Patterns

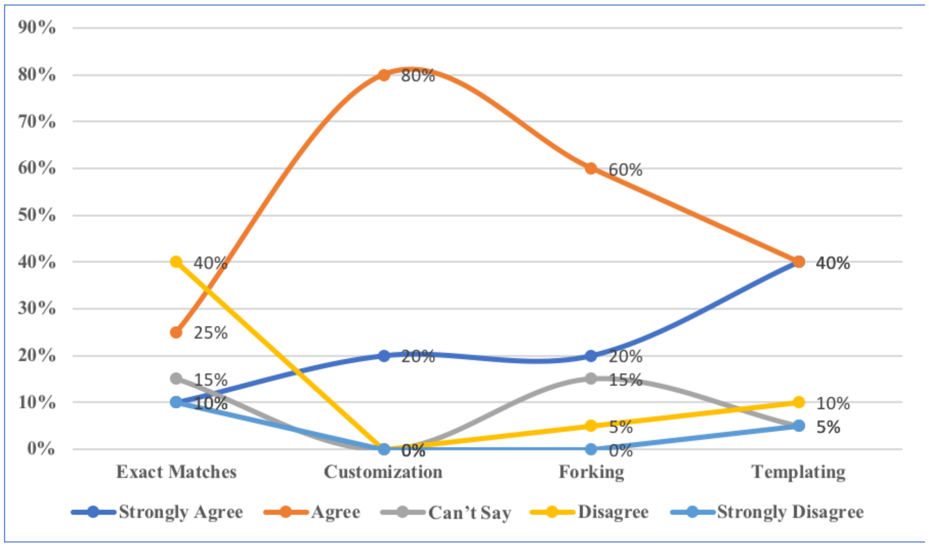

Figure 14. Comparative analysis of results for Cloning Patterns

8. IMPROVING CLONING PRACTICES DURING THE DEVELOPMENT PROCESS

Some interesting facts about software developers intentions and prevalent copy and paste activities were captured after conducting this study. Based on these observations we present the solution for the proper management of the cloning practices thus answering RQ9 of Table 2. Figure 15 envisions the seven ways to improve the software cloning practices identified from the analysis of results acquired from the industrial survey involving software developers. After discussing the survey results, analysis and validations for cloning patterns in the previous sections related to the software cloning practices (more specifically copy and paste practices) it is now desirable to answer one of the main research questions i.e. How can we improve the software cloning practices? Based on the understanding drawn from the survey, we present various ways in which we can improve the software cloning practices during software development processes as discussed below in detail:

8.1 Automatic Tool Support

Various tools have been proposed in the literature (Duala-Ekoko and Robillard [2008], Kawaguchi et al. [2009], Uddin et al. [2015], Rattan et al. [2013], Roy et al. [2009] etc.) to manage clones during software development process ranging from tracking copy and paste activities to the simultaneous editing support. Effective management of clones depends upon the capability of the tool, thus clone management tools must be intelligent enough to sense the coding activity that may lead to clones into the software system. It should automatically trace the duplicated code to further help in refactoring activities. From the survey results and analysis discussed in this paper, we argue that these clone management tools should incorporate a way to handle developers intentions. This is only possible after identifying various reasons behind cloning practices and the associated developers intentions behind it.

8.2 Non-obtrusive Clone Management

It is obviously not acceptable if clone management needs to be started by the developer every time when code change occurs. Clone management should be supported by the automatic tool capable of listening to developers cloning practices and providing the desired clone avoidance or clone removal facilities. The survey results discussed in this paper indicate that the inductions of clones are mainly intentional but it is also identified that reason behind this activity might be that developer indulge in these cloning practices due to lack of knowledge or awareness of

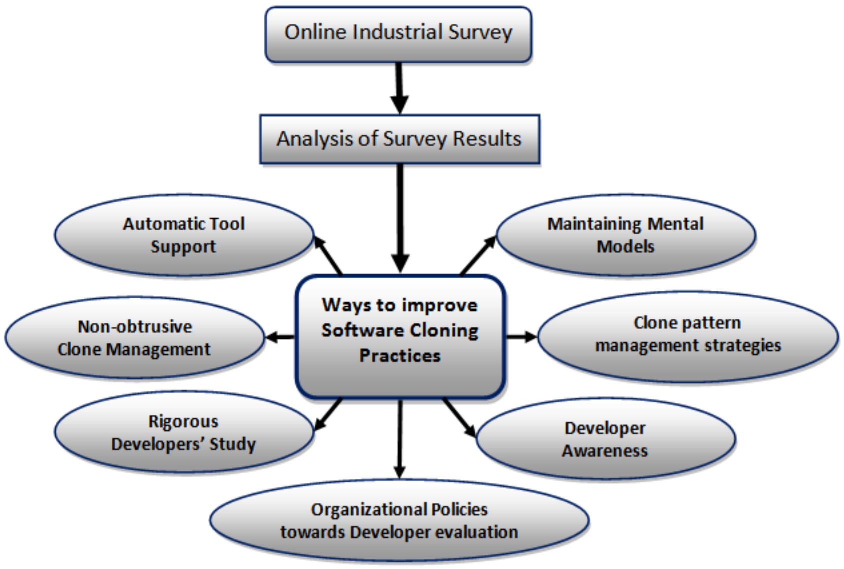
Online Industrial Survey
Analysis of Survey Results
Ways to improve Software Cloning Practices
Automatic Tool Support
Maintaining Mental Models
Non-obtrusive Clone Management
Clone pattern management strategies
Rigorous Developers' Study
Developer Awareness
Organizational Policies towards Developer evaluation

Figure 15. Identified ways to improve software cloning practices

clones and also due to an extra effort needed to learn the available tool. These tools require the knowledge of clones to configure them for the detection and management of clones in the project under consideration. Thus we argue that this configuration choice in these tools should be made automatic without the need of the user to specify them i.e. non-obtrusive clone detection should be adopted into the development process.

8.3 Rigorous Developers Study

As discussed above, we need automatic tool support with non-obtrusive clone management capability. To achieve this goal it necessitates having much more rigorous developers study about their intentions behind cloning practices. Based on these studies we can identify the various reasons behind cloning. As this current study involved just 20 programmers, so we may increase our developer coverage to the maximum number with vast geographical distribution and developer expertise. These studies should involve developers sentiment analysis towards cloning practices, investigation of various prevalent activities that lead to clones into the software system, identification of various reasons for cloning based on programmers perspective as well as supported by rigorous source code analysis.

8.4 Organizational Policies towards Developer Evaluation

Developers do coding to achieve a time-bound goal of the given task. They usually try to reuse the available solution for the given problem under consideration to finish the task in a predetermined time while maintaining the desired quality. This rush or pressure to achieve a goal in limited time may act as a major reason behind cloning, as organizations usually evaluate developers based on their time-bound completion of the task with maintained organizational standards. Thus based on the survey we conducted we argue that organizational policies should be reconsidered according to the current cloning practices to achieve a clone free software systems.

8.5 Developer Awareness

The results presented in this paper clearly indicate that most of the developers are not aware of the clones and their harmfulness. Developers also found it difficult to configure tools for their project as it needed to have prior knowledge of clones and the related parameters to be set for the process of clone detection. Thus we argue that there should be various orientation programmers in the development organizations to notify developers about various cloning practices that may lead to duplicated code into the software system.

8.6 Clone Pattern Management Strategies

There are mainly four clone pattern categories viz. forking, customization, exact match and templating identified by Kapser and Godfrey [2008] and validated by us in this paper. We must implement various strategies for management of these cloning patterns. To manage templating patterns, it is necessary to have synchronous editing support. For forking patterns, we need to take into consideration the historical and architectural dependencies of cloning. Considering both templating and forking pattern management may serve to manage customization patterns in the automatic clone management tools.

8.7 Maintaining Mental Models

Clones are introduced into the software code based by the programmers, so their intentions while coding plays an important role in the presence of duplicated code in the software system. What programmers think while coding makes a big difference, for example, if a programmer is having a pressure to complete a task in defined time, this may impact his logical ability and thus he would try to reuse the available solution to the problem modified according to the current need. These type of mental models need to be taken into consideration and proper counseling should be provided to the software developers because sometimes this approach is useful but mostly have an adverse effect on the maintenance of the software.

9. THREATS TO THE VALIDITY

Discussion of results presented in this paper was acquired from an online industrial survey that involved just 20 software developers. Truthfulness in survey responses may also act as a major setback. Survey questions may have been phrased ambiguously and the responses may also have been misinterpreted. But, despite all these threats, the analysis of responses and the validation of cloning patterns is the remarkable contribution of this paper that may act as a benchmark for further research. Thus the validity of the depicted observational determinations still affirms to be accepted.

10. CONCLUSION AND FUTURE WORK

This paper presented an exploratory study on developers intentions towards copy and paste programming activity. Results were drawn from the online industrial survey conducted by the authors involving professional developers from various national and international software development organizations. This paper emphasized four main key issues related with cloning viz. a source of copied code fragments, reasons behind copying and pasting, type of cloning and extent of cloning. Thus, based on the investigations done, this paper presents exploratory information about the developers intentions while copy and pasting practice. Determinations drawn from this survey were then utilized to check for the validity of four main categories of the cloning patterns viz. forking, templating, customization and exact match. It has been observed that except exact match patterns, the presence of all other three cloning patterns was confirmed grounded on developers perspective.

To manage clones effectively, we present seven solutions to improve cloning practices during the software development process. Based on the inference drawn from the survey, authors suggest incorporating these findings in a copy and paste management tool so that induction of clones into the software system can be better proactively managed.

As a future work, authors are planning to extend this survey to include other key issues relating to various reasons of code cloning, developer awareness etc. so that much more detailed and elucidated inference can be drawn from it.

11. ACKNOWLEDGMENT

Authors express their sincere gratitude to the UGC for Senior Research Fellowship to the first author and also would like to thank the Department of Computer Science, Guru Nanak Dev University, Amritsar for the academic and infrastructural support to the ongoing research

References

Baker, B. S. 1995. On finding duplication and near-duplication in large software systems. In *Proceedings of 2nd Working Conference on Reverse Engineering*. IEEE, 86–95.

Baxter, I. D., Yahin, A., Moura, L., Sant'Anna, M., and Bier, L. 1998. Clone detection using abstract syntax trees. In *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*. IEEE, 368–377.

Bellon, S., Koschke, R., Antoniol, G., Krinke, J., and Merlo, E. 2007. Comparison and evaluation of clone detection tools. *IEEE Transactions on software engineering 33,* 9, 577–591.

Bharti, S. and Singh, H. 2017. An industrial study on developers' prevalent copy and paste activities. In *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*. IEEE, 137–141.

Bharti, S. and Singh, H. 2018. Investigating developers sentiments associated with software cloning practices. In *International Conference on Advanced Informatics for Computing Research*. Springer, 397–406.

Chatterji, D., Carver, J. C., and Kraft, N. A. 2013. Cloning: The need to understand developer intent. In *Proceedings of the 7th International Workshop on Software Clones*. IEEE Press, 14–15.

Cory, K. 2006. " cloning considered harmful" considered harmful. In *Proc. 13th Working Conference on Reverse Engineering, 2006*. IEEE Computer Society.

Duala-Ekoko, E. and Robillard, M. P. 2008. Clonetracker: tool support for code clone management. In *Proceedings of the 30th international conference on Software engineering*. ACM, 843–846.

Geiger, R., Fluri, B., Gall, H. C., and Pinzger, M. 2006. Relation of code clones and change couplings. In *International Conference on Fundamental Approaches to Software Engineering*. Springer, 411–425.

Johnson, J. H. 1994. Substring matching for clone detection and change tracking. In *ICSM*. Vol. 94. 120–126.

Juergens, E., Deissenboeck, F., Hummel, B., and Wagner, S. 2009. Do code clones matter? In *2009 IEEE 31st International Conference on Software Engineering*. IEEE, 485–495.

Kamiya, T., Kusumoto, S., and Inoue, K. 2002. Ccfinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering 28,* 7, 654–670.

Kapser, C. J. and Godfrey, M. W. 2008. cloning considered harmful considered harmful: patterns of cloning in software. *Empirical Software Engineering 13,* 6, 645.

Kawaguchi, S., Yamashina, T., Uwano, H., Fushida, K., Kamei, Y., Nagura, M., and Iida, H. 2009. Shinobi: A tool for automatic code clone detection in the ide. In *2009 16th Working Conference on Reverse Engineering*. IEEE, 313–314.

Kim, M., Bergman, L., Lau, T., and Notkin, D. 2004. An ethnographic study of copy and paste programming practices in oopl. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE'04*. IEEE, 83–92.

Rattan, D., Bhatia, R., and Singh, M. 2013. Software clone detection: A systematic review. *Information and Software Technology 55,* 7, 1165–1199.

Roy, C. K. and Cordy, J. R. 2007. A survey on software clone detection research. *Queens School of Computing TR 541,* 115, 64–68.

Roy, C. K., Cordy, J. R., and Koschke, R. 2009. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of computer programming 74,* 7, 470–495.

Smith, D. D. 2012. *Designing maintainable software.* Springer Science & Business Media.

Thummalapenta, S., Cerulo, L., Aversano, L., and Di Penta, M. 2010. An empirical study on the maintenance of source code clones. *Empirical Software Engineering 15,* 1, 1–34.

Uddin, M. S., Roy, C. K., and Schneider, K. A. 2015. Towards convenient management of software clone codes in practice: An integrated approach. In *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering.* IBM Corp., 211–220.

Wani, S. A. and Dang, S. 2015. Survey based analysis of effect of code clones on software quality. *International Journal of Engineering Research & Technology 4,* 3, 371–379.

Mr. Sarveshwar Bharti is presently working at the Department of Computer Science, Guru Nanak Dev University, Amritsar, India, as a Ph.D. Research Fellow. He has received his Master of Computer Applications (MCA) degree from University of Jammu, Jammu, India. He is a Software Engineering Researcher with research interests including Software Clones, Integrated Clone Management, and Clone Management Plug-in.

Dr. Hardeep Singh is a Professor and Head at the Department of Computer Science and Dean Students' Welfare, Guru Nanak Dev University, Amritsar, India. His research interests lie within Software Engineering and Information Systems. He has been awarded with various prestigious awards including Dewang Mehta Award for best Professor in Computer Engineering, ISTE Award for Best Teacher in Computer Science and Rotract International Award for best Teacher.