A Study on the Nature of Code Clone Occurrence Predominantly in Feature Oriented Programming and the Prospects of Refactoring

U. Devi
Dept. of CS &IT,
The IIS University
Jaipur, Rajasthan

A. Sharma
Dept. of CS & IT,
The IIS University
Jaipur, Rajasthan

N. Kesswani
Dept. of Comp. Science
The Central University
Rajasthan

ABSTRACT
In this position paper, it is tried to analyze the diverse type of code clones which is present and can easily be perpetuated in feature oriented programming. Along with that, a brief summary of the type of code clones and the use of Refactoring methodologies and tools which is effectively known to remove the problem of code clones is also discussed. The main observation that is made in this paper is the various type of code clones which are present in FOP. Through this discussion, it is intended to draw the attention to the various ways in which code clones could propagate and how important it is to curb it at the initial stages to reduce the complexities.

General Terms
Code Clone Detection techniques and tools, Metrices, Refactoring Methods.

Keywords
Code clones, Refactoring, Code Clone metrics, Code clones in FOP and OOP

1. INTRODUCTION
Replicated code fragments in source code, commonly known as *Code Clone*s [1], is an exhaustive research topic. A substantial effort is also invested in analyzing how and when the code clones negatively influence the software quality. Some of the most common consequences of Code Clones on the quality of software systems are : high maintainability, increased code size, increased cost and significantly increased errors due to inconsistent changes. An established modus operandi to counter the problem of Code Clones is *Refactoring. Refactoring* is the process of changing a software system in such a way that it doesn't alter the external behavior of the code but still improves the internal structure [2]. Typically a *Software Product Line (SPL)*, consists of a set of features. To efficiently implement SPL, novel programming paradigms, such as Feature Oriented Programming has gained momentum over the years, which subsequently help in overcoming certain limitations of Object Oriented Programming (OOP). FOP has higher level of modularity and reusability in comparison to OOP[3]. Although FOP has potential to alleviate OOP related code clones, it is possible that it may introduce some of its own – FOP related code clones.

The paper is structured as follows. In Section 2, we discuss the various types of Code Clones, their presence with respect to FOP and OOP and a discussion on Code Clones detection tools. Afterwards, we point out the various Refactoring techniques used along with the tools for its efficient implementation which is explained in the next section. In Section 4, we have distinguished between FOP related code clones and OOP related code clones with the help of an example. We conclude our paper with a discussion which summarizes our observations (Section 5) and a conclusion (Section 6).

2. CODE CLONES: TYPES AND TOOLS
This section discusses the concept of Code Clones and its various types, rather the forms, in which it shows its presence and also the tools to deal with them.

2.1 Code Clone Types
It is widely accepted that code clones have a negative effect on the software system [7]. Code clones or Code smells can aid the identification of SPL refactorings and it improves evolvability and maintainability of delta-oriented SPLs. Given under are the types of Code clones based on their textual (Type I to III) and functional (Type IV) similarities and their relationship (Fig. 1) :

a) Code Fragments that are (almost) identical are called **Type I clones**. Only minor differences regarding formatting such as comments or whitespaces are allowed.

b) A common pattern of cloning is Copy & Paste-and-Modification, which leads to **Type-II clones**. These clones diverge more than Type-I clones so that even differences in names of identifiers, literals, types, layout, or comments are included in this type of clones.

c) **Type-IIIclones** additionally allow changing, adding, or deleting statements. Since deleting a statement from one code fragment can be also interpreted as adding to the corresponding (cloned) statement , both terms (deleting and adding statements) are treated synonymously. Type-III clones are also referred to as *gapped clones*, where the missing statements are called gaps.

Type-IV clones can be syntactically different: The cloning relation for this clone type is based on the semantic similarity between two or more code fragments and thus they are also called *semantic clones* [1].

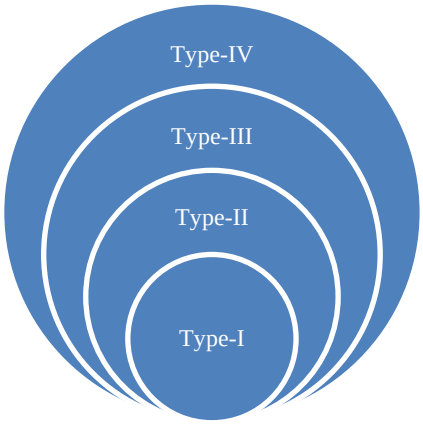

Fig 1 : Relation between different Clone Types

2.2 Code Clones Detection Techniques and Tools

Many clone detection approaches have been proposed in literatures, however, the techniques can essentially be distinguished on the basis of type of information their analysis is based upon and the kind of analysis techniques that have been used. There are mainly five types of clone detection techniques. They are summarized here briefly. Table 1 provides the classification of code clones and its techniques[6].

a) **Textual Approach** : They are the text based approaches which uses little or no transformation on the actual source code before any comparison and in fact taken as it is for clone detection process.

b) **Lexical Approach** : Also called Token-based approach wherein source code is transformed into a sequence of token and scanned for duplicated sub sequence and corresponding original clones are returned as clones.

c) **Syntactic Approach** : It uses a parser to convert source program into parse trees or abstract syntax trees (AST), which is later processed using tree matching or structural metrics to find clones.

d) **Semantic Approach** : It uses static program approaches than simply using syntactic similarity for finding clones.

e) **Metric Based Approach** : It collects a number of metrics for code fragments and then compares the vectors of those metrices rather than comparing the source code or ASTs directly.

Table 1.Classification of Code Clones AndTechniques

Category	Clone Types
Textual Approach	Type – I
Lexical Approach	Type - I,II
Syntactic Approach	Type – I, II, III
Semantic Approach	Type – I, II, III

When comparing code clone detection techniques, precision and recall are often referenced as measures of accuracy and completeness of candidate code clones [4].

Precision (P) = Number of clones correctly found / Total number of clones found

Recall (R) = Number of clones found correct / Total number of clones found

A list of code clone detection tools on the basis of their techniques has been listed below in Table 2 [6].

Table 2. Taxonomy of Clone Detection Techniques And Tools

	Tools
Text Based	Johnson, Duploc, sif, DuDe, SDD, Marcus, Basic NICAD, Full NICAD, Nasehi, Simian
Token Based	Dup, CC Finder(X), D-CCFinder, GemX/Gemini, RTF, CP-Miner, SHINOBI, CPD, CloneDetective, clone, iClones
Tree Based	CloneDr, Asta, cdiff, cpdetector, Tairas, Deckard, CloneDetection, CloneDigger, C2D2, Juiellerat, SimScan,clast, Coogle
Metric Based	Davey
Graph Based	Duplix, Gabel, Komondoor

CodeClone Metrics: There are some basic set of clone metrics on the basis of which the tools are compared which are regarded for consideration in this research [8]. They are given in Table 3.

Table 3. Metrics Used Based Onthe Clone Form

Code Clone Category	Metrics Used
File Metrics	NBR, RSA, RSI, CVR,RNR
Clone Set Metrics	LEN, POP, NIF, RAD, RNR, TKS, LOOP, COND
Line Based Metrics	LOC, SLOC, CLOC, CVRL
FOP Based Metrics	SLOC, CR

3. REFACTORING: AN ANTIDOTE FOR CODE CLONES

Refactoring is one popular and promising technique to eliminate the problem of Code Clones. Refactoring as described earlier, is the sequence of code changes which improves the quality of design (internal structure) without changing the behavior of software (external structure) [2].It is a disciplined way to clean up code that minimizes the chances of introducing bugs. As such, a refactoring is usually a small change to the software, although one refactoringcan involve others. And by applying appropriate methods code clones can be improved, thereby, improving the efficiency, performance, reuse and maintainability of the software programs [9],[10],[11],[12],13]. For FOP, features are considered as the source and targets of refactoring rather than classes. Benefits of Refactoring includes[14],[15],[16],[17] :

- Improves maintainability
- Helps in better understandability and easier modification
- Easier to add new features

Improves code structure and design thereby helps in better and faster code development.Though refactoring is a feasible approach to deal with code clones, it involves a series of steps, the guidelines for which are given below :

- Identifying the part of software that should be refactored
- Deciding which refactoring method should be applied
- Applying refactoring

Assessing the effects of applied refactoring methods on code quality attributes.

3.1 Refactoring Techniques

M.Fowler has presented refactorings in a catalogue like manner [2],[5]. While these refactorings are mainly done for object oriented programming, their applicability is limited for software product lines [2]. Given below, in Fig. 2, are the different types of refactorings which can be used either individually or with the combination of other refactoring techniques.

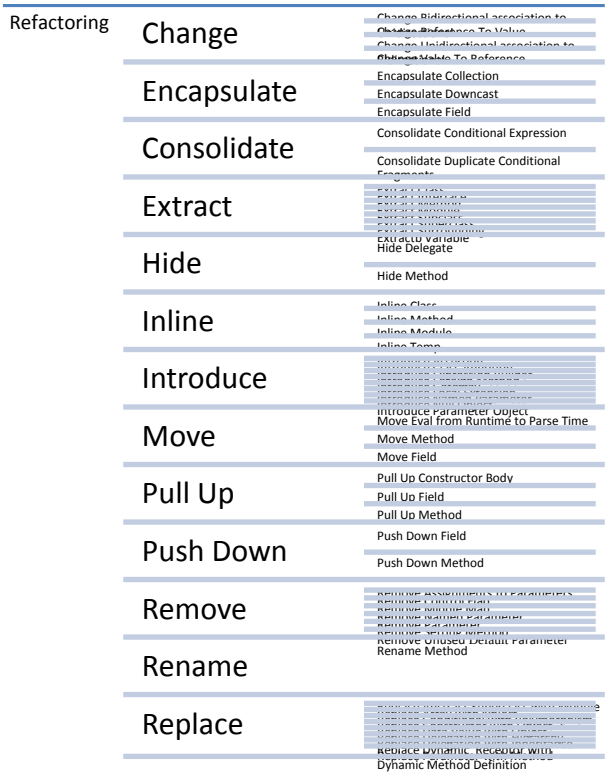

Fig 2. Types of Refactoring

3.2 Refactoring Tools

Many software editors and IDEs have automated refactoring support. There is no standard refactoring browser for Java as there is for SmallTalk, however given under are few of these editors or *refactoring tools*[18],[19],[20],[21],[22],[23],[24],[25]:

(a) Based on Java –

Table 4 (a). Refactoring Tools Based On JAVA

Refactoring Tools	Supporting Language
IntelliJ IDEA	Java, JSP, XML, CSS,HTML and JavaScript
WebStorm	JavaScript
Eclipse	Java, to a lesser extent c++,PHP,Ruby,Javascript, R, Ada, C, COBOL
NetBeans	Java
JDeveloper	Java
DesignPatternTransformer	Java, C, C++
JRefactory	Java
Condenser	No support beyond JDK 1.4
RefactorIT	Java
XRefactory	C, Java (doesn't support newer JDK)

(b) Based on other languages –

Table 4 (b). Refactoring Tools Based On OtherLanguages

Refactoring Tools	Supporting Language
CloneDR	C# versions 2.0,3.0,4.0 and 5
TransMogrify	CSS, HTTPS server support
Visual Studio	.NET, C++
ReSharper	Addon for Visual Studio
CodeRush	Addon for Visual Studio
XCode	Objective C
AppCode	Objective C, C++
SmallTalk Refactoring Browser	Smalltalk

(c) Based on Product Lines -

VAmPiRE (Variant-Preserving Refactoring for feature oriented software product lines), Aries, RefactoringCrawler, CeDAR, FLiPeX.

4. CODE CLONES IN FOP AND OOP HOW IT STILL PROPAGATES

Clones mainly fall into three categories : IfStatement, MethodDeclaration and TypeDeclaration filtered out by syntactical classification [3].FOP specific clones are mostly distributed over alternative features with a common parent feature. Clones detected are inside Syntactical blocks such as Conditionals or Methods because FOP has coarse grained nature due to decomposition of modules (most of the detected codes are at block level). FOP related clones :

- Occur between alternative features, such features are often semantically related because of similar concepts they implement.
- Amount of clones in SPLs developed from scratch is higher than in SPLs decomposed from legacy applications.

Given below, fig.3, is a diagrammatic representation of FOP related and OOP related code clones.

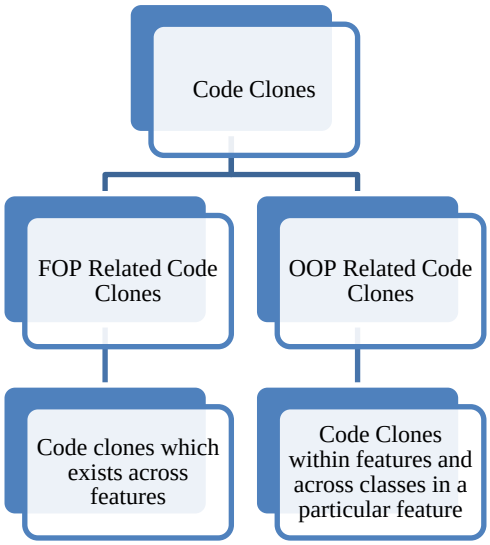

Fig 3. FOP and OOP related Code Clones

Let us take an example for further clarification in this context. Here, F1, F2, F3,….,F10 represent features of a software product line.

And

C1, C2,C3,C4,C5,C6 are the various classes present in the features. (Fig.4)

Next, is the explanation of the types of Code Clones which can be diagnosed in this product lines and if not removed could have a detrimental effect on product line.

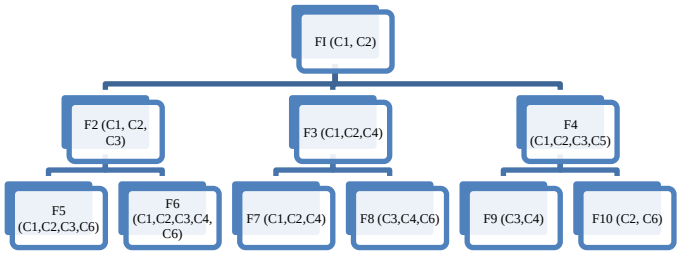

Fig 4. Features and Classes

1. Code Clones lying in Same Feature but Different Classes :

   F2 -> C1 and F2 -> C2.

2. Code Clones lying in Different features : Same classes appearing in different features :

   F2 -> C1, C2 and F3 -> C1, C2.

3. Code Clones lying in different features and different classes :

   F2 -> C3, F4 -> C5

4. Hybrid Clones : Code Clones lying in multiple features and multiple classes :

   F5 -> C1, F6 -> C2 and F8 -> C3

Next, is the elaboration upon the above explained code clone classification with the help of an example, BankAccount where Overdraft, Interest (sub feature is InterestEstimation), CreditWorthiness, DailyLimit are optional features and Lock is a mandatory feature with sub feature as Transaction. The above features and sub features have java classes.

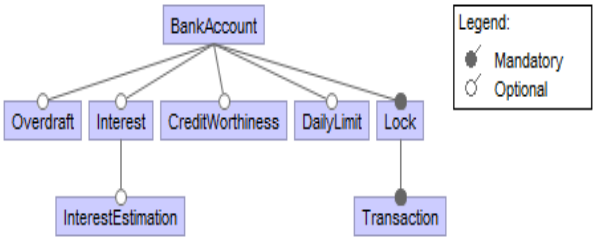

Fig.5. BankAccount Example

Thus, the code clones based on the above discussion could be categorized as :

1. Code Clones lying in Same Feature but Different Classes :Interest -> Account.java , Application.java
2. Code Clones lying in Different features : Same classes appearing in different features :
   Interest -> Account.java, Application.java
   DailyLimit -> Account.java, Application.java
3. Code Clones lying in different features and different classes :
   Overdraft -> Account.java
   Transaction -> Transaction.java
4. Hybrid Clones : Code Clones lying in multiple features and multiple classes :
   Interest -> Application.java
   Transaction -> Transaction.java

5. DISCUSSION

In the following, we shortly discuss some of the observations made from above literature.

- Firstly, the different types of code clones have been classified and the code clone detection tool have also been mentioned based on the clone detection techniques. But there's still a lack of proper understanding of which clone type are FOP related or OOP related. Of course, some of the code clones in OOP have been successfully avoided in FOP but it is still not clear if they are the similar code clones originating as FOP related code clones. The presence of code clones in FOP directly affects the quality, cost and increases complexity.

- Secondly, it is attempted to classify FOP related code clones in four different types with increased complexity in each subsequent type. This kind of observation clearly places the importance towards the study of code clones in FOP with respect to complexity and how further it could be propagated if not controlled.

- Thirdly, there is a lack of refactoring approach for proactive and reactive development of software product lines. Also most of the refactorings proposed are semi-automated. Though there are works on theory of stepwise evolution of SPL with usability as main focus, no implementation or case study exists []. Also, more study is required to quantify which causes are important for FOP related clones and which are not.

- Another important point which was observed in the light of this study was that in the existing methodologies, FOP is able to alleviate OOP related code clones but instead introduces FOP related code clones. Aspect Oriented Programming (AOP) is also known to have AOP related code clones. But in the recent times, there have been no such literature or work which could identify the presence of code clones in Delta Oriented Programming (DOP), another programming paradigm for implementation of SPL. The research is still ongoing.

- An additional dimension in the code cloning area could be upon the Product Configuration, as in, there could be a possibility of a product which consists of clone free features. Then the product shall have low maintainability with respect to code clones. But the reverse may also be true, i.e. a product in which all its features suffer from the problem of code clones, shall have high maintainability. A study in this direction could prove helpful in many areas.

- It is also a potential area of study whether and if cross cutting concerns are related to the propagation of code clones. If the problem of code clones are present, then how can it be handled effectively? Another facet is the concept of separation of concerns which promotes modularity. The notion of cross cutting concerns has gained wide popularity in AOP.

- Lastly, one of the most essential requirements of today is of a tool which could easily perform all the activities starting with detecting code clones, classifying them according to their types and refactor them.

However, it is established that a fraction of FOP related code clone could be removed and a feasible approach to do this is through refactoring. But more study and research is to be done to find out more about the classification of code clones in different programming paradigm, their characteristics, the causes and their removal in feature oriented SPLs.

6. CONCLUSION

In this paper, we have tried to analyze how negatively code clones affect the source code. Code clones are categorized and detected based on certain established approaches. The tools which have been given are able to categorize code clones based on the approaches. Refactoring is one remedial measure to tackle with the problem of code clones. The methods generally lie in four category which has been exhaustively mentioned above along with the refactoring tools. We have also tried to explain with an example How code cloning is interrelated and overlapping among classes and features and how there is but a small difference between FOP related and OOP related code clones, an area which needs to be explored further to help throw light for its cause and effect, particularly in regard to feature oriented SPLs. And finally, we have put forward some research questions which are open ended and could be further initiated with proper study.

7. REFERENCES

[1] S. Schulze."Analysis and Removal of Code Clones in Software Product Lines". Dissertation. University of Magdeburg, 2012.

[2] S. Schulze, O. Richers, and I. Schaefer. "Refactoring delta-oriented software product lines". In *AOSD*. pp. 73-84. ACM, 2013.

[3] S. Schulze, S. Apel and C. Kästner. "Code Clones in Feature-Oriented Software Product Lines". In *Proceedings of the 9th International Conference on Generative Programming and Component Engineering (GPCE)* (Eindhoven, The Netherlands), New York, NY, USA. pp. 103-112. ACM Press, October 2010.

[4] Balwinder Kumar, Dr. Satwinder Singh, "Code clone detection and Analysis using Software Metrics and Neural Network- A Literature Review", In *IJCST*, Volume 3, issue 2, March-April 2015.

[5] M. Fowler. "Refactoring: Improving the Design of Existing Code". Addison-Wesley, 1st Edition. USA, 2000.

[6] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: a qualitative approach," In *Science of Computer Programming*. Volume. 74, no. 7. pp. 470–495. 2009. Special Issue on Program Comprehension (ICPC 2008).

[7] C.K. Roy and J.R. Cordy, "A survey on software clone detection research," *Queen's Technical Report:* 541. , page 115. 2007.

[8] http://www.ccfinder.net/ccfinderx.html

[9] E. Murphy-Hill and A. P. Black, "Refactoring Tools: Fitness for Purpose," In *IEEE Softw.*. Volume. 25, no. 5. pp. 38–44. 2008

[10] S. Schulze, T. Thum, M. Kuhlemann, and G. Saake. "Variant-preserving refactoring in feature-oriented software product lines". In *VaMoS*. pp 73-81. ACM, 2012.

[11] William F. Opdyke, "Refactoring Object-Oriented Frameworks," PhD thesis, University of Illinois, Urbana-Champaign. 1992. *Tech. Report UIUCDCS-R-92-1759*

[12] N. Kumari and A. Saha, "Effect of Refactoring on Software Quality," In*Fourth International Conference on Advances in Computing and Information Technology (ACITY 2014)*. Delhi ,India. May 2014.

[13] Bart Du Bois, P. V. Gorp, A. Amsel, N. V. Eetvelde, H. Stenten, S. Demeyer, and T. Mens, " A discussion of refactoring in research and practice," *Technical report.* University of Antwerp. January 2004.

[14] Y. Higo, T. Kamiya, S. Kusumoto, and K. Inoue, "Refactoring support based on code clone analysis," In *Proc. of the Product Focused Software Process Improvement Int'l Conference*. pp. 220–233. 2004.

[15] E. Kodhai and S. Kanmani, "Method-Level Code Clone Modification using Refactoring Techniques for Clone Maintenance,"In *Advanced Computing: An International Journal (ACIJ)*. Vol.4, no.2. March 2013.

[16] D. G. Devi and Dr.M.Punithavalli, "Comparison and evaluation on metrices based approach for detecting code clone," In *Indian Journal of Computer Science and Engineering (IJCSE).*

[17] G Singh and N. Kohli ,"Evaluating the effect of code clones on software maintenance cost," In *Apeejay Journal of Computer Science and Applications*. 2014.

[18] http://dpt.kupin.de/

[19] http://jrefactory.sourceforge.net/

[20] http://www.eclipse.org/

[21] http://www.semdesigns.com/Products/Clone/.

[22] http://www.intellij.com/idea/

[23] http://www.instantiations.com/jfactor/

[24] http://www.refactorit.com/

[25] http://www.xref-tech.com/xrefactory/