

**IMPACT OF CODE CLONING ON FUNCTIONAL  
CORRECTNESS OF OBJECT ORIENTED CLASSES**

BY

**YASSER SAAD AL-GHAMDI**

A Thesis Presented to the  
DEANSHIP OF GRADUATE STUDIES

**KING FAHD UNIVERSITY OF PETROLEUM & MINERALS**

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the  
Requirements for the Degree of

**MASTER OF SCIENCE**

In

**COMPUTER SCIENCE**

**May 2014**

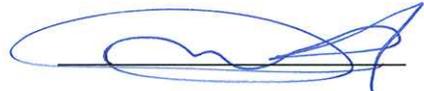
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS  
DHAHRAN- 31261, SAUDI ARABIA  
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **YASSER SAAD ABDULRAHMAN AL-GHAMDI** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

**Thesis Committee:**



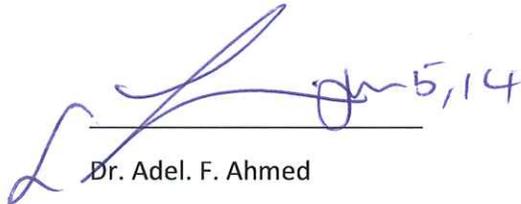
Dr. Mahmoud. Elish (Advisor)



Dr. Mohamed. El-Attar (Member)



Dr. Sajjad Mahmood (Member)



Dr. Adel. F. Ahmed

Department Chairman



Dr. Salam A. Zummo  
Dean of Graduate Studies

12/6/14  
Date



© Yasser Saad Al-Ghamdi

2014

## DEDICATION

*To my dear parents, for their prayers and support*

*To my precious wife, for her patience and support*

## **ACKNOWLEDGMENTS**

In the name of Allah, most gracious, most merciful

I would like to express my deep appreciation for the encouragement extended to me by my employer Saudi Aramco, the world leading oil and Gas Company to pursue a Master's degree in Computer Science. Special thanks to my management Mr. Adel Al-Naji, Mr. Abdallah Al-Eidi, and Mr. Abdulkarim Al-Zahrani for their continuous support and encouragement.

I further extend my acknowledgment to my thesis advisor Dr. Mahmoud Elish for his priceless support, commitment, encouragement and the constructive feedback.

Great thanks to my thesis committee members Dr. Mohamed El-Attar and Dr. Sajjad Mahmood for their attention, comments, help and constructive feedback.

Special thanks go to my beloved family, brothers, sisters, and children for their constant prayers and encouragement. I convey my sincere thanks to all of my friends and colleagues who helped me throughout the research.

# TABLE OF CONTENTS

ACKNOWLEDGMENTS .....	iv
TABLE OF CONTENTS .....	v
LIST OF TABLES .....	vii
LIST OF FIGURES .....	ix
ABSTRACT .....	x
ملخص الرسالة .....	xi
CHAPTER 1 INTRODUCTION.....	1
1.1. Problem Statement.....	3
1.2. Research Objective and Questions .....	4
1.3. Motivation .....	5
1.4. Research Contribution.....	6
1.5. Thesis organization .....	7
CHAPTER 2 LITERATURE REVIEW.....	8
2.1 Impact of Software Clones on Software Quality .....	8
2.2 Software Clones Taxonomies and Types.....	14
2.2.1 Taxonomies Based on Similarity: .....	14
2.2.2 Taxonomies Based on Clone Location.....	15
2.2.3 Taxonomies Based on Refactoring Opportunities.....	15
2.3 Software Clones Detection Techniques and Tools.....	15
CHAPTER 3 TYPES AND USED METRICS SUITES.....	18
3.1 Clone Types.....	18
3.2 Clone Metrics.....	22
3.3 C&K Metrics Definitions .....	25
CHAPTER 4 EMPIRICAL STUDY OF SOFTWARE CLONES AND FAULT- PRONENESS.....	27
4.1 Goal and Research Questions: .....	27
4.2 Variables and Data Collection: .....	31
4.2.1 Variables Used in the Study .....	32
4.2.2 Data Collection Tools:.....	33
4.3 Descriptive Statistics .....	35

4.4	Principal Component Analysis.....	42
4.5	Mann-Whitney Test .....	44
4.6	Univariate Analysis .....	67
4.7	Multivariate Analysis.....	77
4.7.1	Validation of Fault-Proneness Prediction Model .....	84
4.7.2	Comparisons of Code Clones Metrics and C&K Metrics in Predicting Fault-Proneness.....	85
4.7.3	Comparisons of Regression models of Code Clones Metrics and Neural Network Models in Predicting Fault-Proneness.....	92
4.8	Threats to validity .....	93
4.8.1	Construct Validity .....	93
4.8.2	Internal Validity .....	93
4.8.3	Statistical Conclusion Validity .....	94
4.8.4	External Validity .....	94
<b>CHAPTER 5 EMPIRICAL STUDY OF SOFTWARE CLONES AND FAULT-DENSITY.....</b>		<b>96</b>
5.1	Goal and Research Questions: .....	96
5.2	Descriptive Statistics .....	100
5.3	Principal Component Analysis.....	105
5.4	Mann-Whitney Test .....	105
5.5	Univariate Analysis .....	131
5.6	Multivariate Analysis.....	141
5.6.1	Validation of Fault-density Prediction Model .....	148
5.6.2	Comparisons of Code Clones Metrics and C&K Metrics in Predicting Fault-density.....	149
5.6.3	Comparisons of Regression models of Code Clones Metrics and Neural Network Models in Predicting Fault-density.....	154
5.7	Threats to validity .....	155
<b>CHAPTER 6 CONCLUSION AND FUTURE WORK.....</b>		<b>156</b>
6.1	Research Contributions .....	156
6.2	Future Work.....	157
<b>REFERENCES .....</b>		<b>159</b>
<b>Appendix .....</b>		<b>163</b>
<b>Vitae .....</b>		<b>167</b>

## LIST OF TABLES

<b>Table 1: Literature summary of studies on code clones</b> .....	13
<b>Table 2: Summary table of clone detection tools from the literature</b> .....	16
<b>Table 3: Descriptive Statistics of the subject systems</b> .....	32
<b>Table 4: Descriptive Statistics of Velocity</b> .....	37
<b>Table 5: Descriptive Statistics of Synapse</b> .....	38
<b>Table 6: Descriptive Statistics of Ant</b> .....	39
<b>Table 7: Descriptive Statistics of Xalan</b> .....	40
<b>Table 8: Descriptive Statistics of Camel</b> .....	41
<b>Table 9: Descriptive Statistics of Total</b> .....	42
<b>Table 10: Principal Components of Velocity</b> .....	43
<b>Table 11: Principal Components of Synapse</b> .....	43
<b>Table 12: Principal Components of Ant Table</b> .....	44
<b>Table 13: Principal Components of Xalan</b> .....	44
<b>Table 14: Principal Components of Camel</b> .....	44
<b>Table 15: Mann-Whitney test for general clones including all types</b> .....	45
<b>Table 16: Mann-Whitney test for clone locations whether inter clones or intra clones</b> .....	45
<b>Table 17: Mann-Whitney test for clone locations whether inter clones or intra clones</b> .....	46
<b>Table 18: Mann-Whitney test for clone locations whether inter clones or intra clones</b> .....	48
<b>Table 19: Mann-Whitney test for clone locations whether inter clones or intra clones</b> .....	49
<b>Table 20: Mann-Whitney test for clone locations whether inter clones or intra clones</b> .....	50
<b>Table 21: Mann-Whitney test for clone locations whether inter clones or intra clones</b> .....	52
<b>Table 22: Mann-Whitney Test Based on Types</b> .....	54
<b>Table 23: Mann-Whitney Test Based on Types for Synapse</b> .....	56
<b>Table 24: Mann-Whitney Test Based on Types</b> .....	57
<b>Table 25: Mann-Whitney Test Based on Types</b> .....	59
<b>Table 26: Mann-Whitney Test Based on Types</b> .....	60
<b>Table 27: Mann-Whitney Test Based on Types</b> .....	61
<b>Table 28: Summary of Mann-Whitney Test for Clones Based on Location</b> .....	63
<b>Table 29: Summary of Mann-Whitney Test for Clones Based on Types</b> .....	64
<b>Table 30: Univariate Analysis for Velocity system</b> .....	68
<b>Table 31: Univariate Analysis for Synapse system</b> .....	69
<b>Table 32: Univariate Analysis for Ant system</b> .....	71
<b>Table 33: Univariate Analysis for Xalan system</b> .....	72
<b>Table 34: Univariate Analysis for Camel system</b> .....	74
<b>Table 35: Univariate Analysis for Comprehensive system</b> .....	75
<b>Table 36: Multivariate Analysis for Velocity system</b> .....	78
<b>Table 37: Multivariate Analysis for Synapse system</b> .....	78
<b>Table 38: Multivariate Analysis for Ant system</b> .....	79
<b>Table 39: Multivariate Analysis for Xalan system</b> .....	81

<b>Table 40: Multivariate Analysis for Camel system</b> .....	82
<b>Table 41: Multivariate Analysis for Comprehensive system</b> .....	83
<b>Table 42: Correct classification rate for training data and Cross-Validation Data</b> .....	84
<b>Table 43: ROC Area for training data and Cross-Validation Data</b> .....	85
<b>Table 44: Multivariate Analysis - (Goodness-of-fit)</b> .....	86
<b>Table 45: Multivariate Analysis - (Correctly Classified Rate)</b> .....	87
<b>Table 46: Multivariate Analysis - (ROC Area)</b> .....	88
<b>Table 47: Mann-Whitney Test Result-Fault-Density Based on Clones Location Velocity</b> ..	106
<b>Table 48: Mann-Whitney Test Result-Fault-Density Based on Clones Location Synapse</b> ..	108
<b>Table 49: Mann-Whitney Test Result-Fault-Density Based on Clones Location Ant</b> .....	109
<b>Table 50: Mann-Whitney Test Result-Fault-Density Based on Clones Location Xalan</b> ....	111
<b>Table 51: Mann-Whitney Test Result-Fault-Density Based on Clones Location Camel</b> ....	112
<b>Table 52: Mann-Whitney Test Result-Fault-Density Based on Clones Location Comprehensive</b> .....	114
<b>Table 53: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Velocity</b>	116
<b>Table 54: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Synapse</b>	118
<b>Table 55: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Ant</b> .....	120
<b>Table 56: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Xalan</b> ....	122
<b>Table 57: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Camel</b> ...	124
<b>Table 58: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Comprehensive System</b> .....	126
<b>Table 59: Summary of Mann-Whitney Test for Fault-Density Based on Clone Location</b> ..	128
<b>Table 60: Summary of Mann-Whitney Test for Fault-Density Based on Clone Types</b> .....	129
<b>Table 61: Univariate Analysis for Fault-Density of Velocity System</b> .....	132
<b>Table 62: Univariate Analysis for Fault-Density of Synapse System</b> .....	134
<b>Table 63: Univariate Analysis for Fault-Density of Ant System</b> .....	135
<b>Table 64: Univariate Analysis for Fault-Density of Xalan System</b> .....	136
<b>Table 65: Univariate Analysis for Fault-Density of Camel System</b> .....	137
<b>Table 66: Univariate Analysis for Fault-Density of Comprehensive System</b> .....	139
<b>Table 67: Univariate Analysis for Fault-Density of Velocity System</b> .....	141
<b>Table 68: Univariate Analysis for Fault-Density of Synapse System</b> .....	142
<b>Table 69: Univariate Analysis for Fault-Density of Ant System</b> .....	143
<b>Table 70: Univariate Analysis for Fault-Density of Xalan System</b> .....	144
<b>Table 71: Univariate Analysis for Fault-Density of Camel System</b> .....	146
<b>Table 72: Univariate Analysis for Fault-Density of Comprehensive System</b> .....	147
<b>Table 73: Comparison of Multivariate Analysis - Training data Vs. Cross-validation</b> .....	148
<b>Table 74: Comparison of Multivariate Analysis - Goodness-of-fit</b> .....	149
<b>Table 75: Comparison of Multivariate Analysis -Prediction Performance</b> .....	152
<b>Table 76: Comparison of Multivariate Analysis -Prediction Performance versus MLP</b> ....	155
<b>Table 77: Literature summary of studies on code clones</b> .....	166

## LIST OF FIGURES

<b>Figure 1: Histogram of Clones and Faults Distribution .....</b>	<b>36</b>
<b>Figure 2: Velocity-Faults Distribution Based on Location.....</b>	<b>46</b>
<b>Figure 3: Synapse-Faults Distribution Based on Location.....</b>	<b>47</b>
<b>Figure 4: Ant-Faults Distribution Based on Location .....</b>	<b>49</b>
<b>Figure 5: Xalan-Faults Distribution Based on Location .....</b>	<b>50</b>
<b>Figure 6: Camel-Faults Distribution Based on Location .....</b>	<b>52</b>
<b>Figure 7: Comprehensive-Faults Distribution Based on Location .....</b>	<b>53</b>
<b>Figure 8: Velocity-Faults Distribution Based on Types .....</b>	<b>55</b>
<b>Figure 9: Synapse-Faults Distribution Based on Types .....</b>	<b>57</b>
<b>Figure 10: Ant-Faults Distribution Based on Types .....</b>	<b>58</b>
<b>Figure 11: Xalan-Faults Distribution Based on Types .....</b>	<b>59</b>
<b>Figure 12: Camel-Faults Distribution Based on Types .....</b>	<b>61</b>
<b>Figure 13: Comprehensive-Faults Distribution Based on Types.....</b>	<b>62</b>
<b>Figure 14: Histogram of Correctly Classified Rate in Multivariate Analysis .....</b>	<b>87</b>
<b>Figure 15: Multivariate Analysis - ROC Curve for Velocity .....</b>	<b>88</b>
<b>Figure 16: Multivariate Analysis - ROC Curve for Synapse .....</b>	<b>89</b>
<b>Figure 17: Multivariate Analysis - ROC Curve for Ant.....</b>	<b>90</b>
<b>Figure 18: Multivariate Analysis - ROC Curve for Xalan.....</b>	<b>90</b>
<b>Figure 19: Multivariate Analysis - ROC Curve for Camel.....</b>	<b>91</b>
<b>Figure 20: Box-Plot of Fault-Density for Velocity System .....</b>	<b>101</b>
<b>Figure 21: Box-Plot of Fault-Density for Synapse System .....</b>	<b>102</b>
<b>Figure 22: Box-Plot of Fault-Density for Ant System.....</b>	<b>102</b>
<b>Figure 23: Box-Plot of Fault-Density for Xalan System.....</b>	<b>103</b>
<b>Figure 24: Box-Plot of Fault-Density for Camel System .....</b>	<b>103</b>
<b>Figure 25: Box-Plot of Fault-Density for Comprehensive System.....</b>	<b>104</b>
<b>Figure 26: Box-Plot of Fault-Density Based on Location for Velocity System.....</b>	<b>107</b>
<b>Figure 27: Box-Plot of Fault-Density Based on Location for Synapse System.....</b>	<b>109</b>
<b>Figure 28: Box-Plot of Fault-Density Based on Location for Ant System .....</b>	<b>110</b>
<b>Figure 29: Box-Plot of Fault-Density Based on Location for Xalan System.....</b>	<b>112</b>
<b>Figure 30: Box-Plot of Fault-Density Based on Location for Camel System.....</b>	<b>113</b>
<b>Figure 31: Box-Plot of Fault-Density Based on Location for Comprehensive System .....</b>	<b>115</b>
<b>Figure 32: Box-Plot of Fault-Density Based on Type for Velocity System .....</b>	<b>117</b>
<b>Figure 33: Box-Plot of Fault-Density Based on Type for Synapse System .....</b>	<b>119</b>
<b>Figure 34: Box-Plot of Fault-Density Based on Type for Ant System.....</b>	<b>121</b>
<b>Figure 35: Box-Plot of Fault-Density Based on Type for Xalan System.....</b>	<b>123</b>
<b>Figure 36: Box-Plot of Fault-Density Based on Type for Camel System .....</b>	<b>125</b>
<b>Figure 37: Box-Plot of Fault-Density Based on Type for Comprehensive System.....</b>	<b>127</b>
<b>Figure 38: Goodness-of-fit - R-Square .....</b>	<b>150</b>
<b>Figure 39: Goodness-of-fit - Adjusted R-Square .....</b>	<b>151</b>
<b>Figure 40: Prediction Performance - MAE .....</b>	<b>152</b>
<b>Figure 41: Prediction Performance - RMSE .....</b>	<b>153</b>
<b>Figure 42: Prediction Performance - SDAE.....</b>	<b>153</b>

## **ABSTRACT**

Full Name : Yasser Saad Abdulrahman Al-Ghamdi  
Thesis Title : IMPACT OF CODE CLONING ON FUNCTIONAL CORRECTNESS  
OF OBJECT-ORIENTED CLASSES  
Major Field : Computer Science  
Date of Degree : May, 2014

Many software developers are incorporating code clones in source code as a natural habit while software is evolving. Several arguments about the code clones if they are harmful or beneficial, several studies in the literature discussed code clones impact on the software from different perspectives on different levels. However, the important question of the impact of code clones on software quality is in need of further studies to identify positive or negative impacts on quality aspects of the software. An important aspect of software quality is functional correctness which is a sub-characteristic of software functional suitability. Exploring code clone metrics to assess functional correctness might discover the impact of code clones on the product quality.

The main objective of this research is to study empirically the relationships between software clones and functional correctness in object-oriented classes. This research contains two empirical studies, the first one is to study clones and fault-proneness, and the second is to study code clones and fault-density.

The results indicate that there are relationships between code clones and functional correctness. In addition, there are relationships between clone location and fault-proneness. Similarly, we proved the relationships between types of code clones and fault-proneness in addition to fault-density as well. This research revealed 3 clone metrics that are good indicators of fault-proneness, in addition to one clone metric that is a good indicator of fault-density. Our study showed that when combine C&K metrics with clone metrics they perform better in fault-proneness and fault-density prediction. Finally, some recommendations for software developers are presented in this research towards incorporating code clones in the development of the software product.

## ملخص الرسالة

الاسم: ياسر بن سعد بن عبدالرحمن الغامدي

عنوان الرسالة: تأثير نسخ رموز البرمجة على الدقة الوظيفية للبرامج كائنية التوجه

التخصص: علوم الحاسب الآلي

تاريخ الدرجة العلمية: مايو 2014

العديد من مطوري البرمجيات معتادون على نسخ ولصق رموز البرمجة من وقت لآخر اثناء دورة تطوير البرامج. هناك العديد من وجهات النظر المختلفة في كون استخدام النسخ للرموز البرمجية هل هو ضار أم نافع, وكان سبب هذه الاختلافات هي النتائج المغايرة للعديد من الابحاث والدراسات في هذا المجال. من المهم جدا تحديد تأثير استخدام الرموز البرمجية اثناء تطوير البرامج لتلافي الأخطار المحتملة ان وجدت, وايضا للتركيز على الجوانب الايجابية ايضا في حال توажدها.

الهدف الرئيسي من هذه الرسالة البحثية هو دراسة تأثير نسخ الرموز البرمجية على الدقة الوظيفية للبرامج كائنية التوجه. في نهاية هذا البحث واعتمادا على النتائج وجهنا بعض النصائح العملية لمطوري البرامج للاستخدام الامثل عند نسخ رموز البرمجة.

## CHAPTER 1 INTRODUCTION

The demand on software applications is increasing in all of our daily activities. In the same time, the expectations are also increasing which lead to the need for more complex and sophisticated software. With this pressure to accommodate what the software industry needs, software developers tend to reuse some existing code to expedite the process of developing software. The simplest form of reuse is what many software developers do which is “copy and paste”, which can be referred to it by “code cloning”.

Code cloning is a common practice during software development where a software developer reuses similar code fragments to be part of a new development of software systems with or without minor modifications [1-4].

Several studies have been conducted against identifying and measuring the impact of software cloning with respect to quality [5-11]. Some studies found negative impact of software clones [12, 13] and some researchers realized positive impact of software clones on the quality of the software [9]. The software quality is a strong desire to any project manager who cares about the success of the project. Previous studies looked at the impact of code clones on the maintainability of the software [5, 10, 11]. Further studies to the nature of the impact of code cloning are important to measure and steer the right approach of software developer in order to decide about incorporating code clones or not.

The functional suitability of a software system can be measured with multiple internal quality attributes and metrics [14]. Functional correctness used as sub-characteristic towards measuring the functional suitability of the software where fault-density and fault-proneness will be used as metrics for functional correctness. Therefore, there is a need to

conduct a study to identify the impact of code cloning on the functional correctness either positively or negatively.

The ISO/IEC 25010 [14] standard defines eight characteristics of software product quality: functional suitability, maintainability, performance efficiency, compatibility, usability, security, reliability and portability. These characteristics are further divided into sub characteristics. Functional correctness is a sub-characteristic of the functional suitability characteristic. It is defined as “the degree to which a product or system provides the correct results with the needed degree of precision” [14]. Inverted proxy measures for functional correctness include fault proneness and fault density.

By reaching any conclusion out of this study, it will contribute positively in considering quality aspects while incorporating code clones in software development. This study will help software developers in managing their projects more effectively with less risk. During the life cycle of software development and especially in the coding phase, developers tend to use code clones for several reasons. For example, saving development time to implement a newly added feature that has a similar component in another place within the same system or across similar systems, and/or reusing code clones that are well tested for a critical function that cannot handle any risk. All of the stated reasons are very legitimate to justify the use of code clones although some researches claim that the use of code clones has an impact on the software quality [12, 13]. Further studies are needed to clarify the impact of code clones on one aspect of product quality which is functional suitability, focusing particularly on a sub-characteristic which is functional correctness. Two software measures will be used to study functional correctness, namely: fault-proneness, and fault density.

The definition of a code clone can be generally stated as an exact or similar copy of a code fragment. There are many other definitions of a code clone that depend on the type and degree of similarity [2]. Similarity of code fragments can be classified in two main categories, namely: textual similarity, and functional similarity (semantic).

Based on the two main categories, C.Roy and J.Cordy came up with four types of code fragment similarities. The first three types are textual-similar [1, 2, 15] and the fourth type is functional-similar [3, 4].

The functional correctness will be evaluated using two software metrics, which are fault density, and fault proneness. Functional correctness will indicate a certain aspect of software quality which is functional suitability. In order to investigate the code cloning impact, a set of code clone metrics will be used. Two different code clone analysis tools will be used to study a set of open source subject systems to detect and analyze the clones and try to achieve the needed measurements to judge the impact on functional correctness. The focus of our research is to study the code clones on the class level and their impact on the functional correctness of the software system.

## **1.1. Problem Statement**

A very common practice in software development environments is to use code cloning [1-4]. There are different reasons for incorporating code clones in the development cycle, such as cutting the time of development, and reaching earlier delivery date. The quality of software product is a very essential requirement for the product stakeholders [9,16-18]. Functional suitability is a very important product quality attribute that software developers and project managers are looking for [19-22]. Moreover, two sub-

characteristics of functional correctness are fault density and fault-proneness. The current researches have reached the maturity level of detecting code clones effectively [23, 24], but, the next question is what the benefits of these code clones are [13, 25]. Several studies in the literature explored the relationships between code clones and reliability [5], maintainability [10], stability [11, 26], error prone [6, 27]. The studies that discussed the relationships between code clones and error prone were focusing on certain code fragments namely, function level by Selim et al. and free code block by Rahman et al.

To our knowledge only two researches discussed the relationship between clones and fault prediction in abstract level i.e. on component level [28] and on file level [29]. Therefore, to the best of our knowledge there is no study discussed code cloning and its impact on functional correctness of object-oriented classes.

## **1.2. Research Objective and Questions**

The main objective of this research is to empirically study the relationships between software clones and functional correctness. Two empirical studies will cover the relationships between both aspects of functional correctness, i.e. fault-proneness and fault-density, with code clones. In addition, an evaluation study that considers code clones metrics as inputs to an Artificial Neural Network of functional correctness prediction models.

The following questions summarize the main objectives of this study:

1. Are there relationships between code clones and fault proneness of object-oriented classes?
2. Are clone metrics good indicators of fault proneness?
3. Do clone metrics have better fault proneness prediction than C&K metrics?
4. Does the combination of clone metrics and C&K metrics as input in prediction models of fault proneness yield better results than clone metrics?
5. Does the combination of clone metrics and C&K metrics as input in prediction models of fault proneness yield better results than C&K metrics?
6. Are there relationships between code clones and fault density of object-oriented classes?
7. Are clone metrics good indicators of fault density?
8. Do clone metrics have better fault density prediction than C&K metrics?
9. Does the combination of clone metrics and C&K metrics as input in prediction models of fault density yield better results than clone metrics?
10. Does the combination of clone metrics and C&K metrics as input in prediction models of fault density yield better results than C&K metrics?

### **1.3. Motivation**

The importance of software quality has recently rapidly increased, for large software products in particular [23], resulting in identifying various aspects of software quality attributes which can affect the quality, either in a positive way, so to focus on them, or in a negative way, so to avoid them. Since code cloning is a common practice in software

development [24-27], the importance of studying its effects on software quality and on functional suitability in particular rises. Knowing the impact of code cloning might affect any project plan [28]. With the assumption of having a negative impact, certain techniques are needed to overcome this issue, such as increasing the testing resources and efforts, or using a change tracking tool to assure the consistency in changes among the clones. In addition, using code clones metrics to predict functional correctness will help future researchers to build new prediction models using the new metrics.

#### **1.4. Research Contribution**

This research provides the following list of contributions:

- Build and evaluate logistic regression models based on code clone metrics to predict fault-proneness
- Build and evaluate logistic regression models based on code clone location metrics to predict fault-proneness
- Build and evaluate logistic regression models based on code clone type metrics to predict fault-proneness
- Build and evaluate linear regression models based on code clone metrics to predict fault-density
- Build and evaluate linear regression models based on code clone location metrics to predict fault-density
- Build and evaluate linear regression models based on code clone type metrics to predict fault-density

- Build and compare multivariate logistic regression models for fault-proneness using code clones metrics suite, C&K metrics suite, and the combinations of both suites
- Build and compare multivariate linear regression models for fault-density using code clones metrics suite, C&K metrics suite, and the combinations of both suites
- Build and compare Artificial Neural Network prediction models for fault-proneness using code clones metrics suite and logistic regression models
- Build and compare Artificial Neural Network prediction models for fault-density using code clones metrics suite and linear regression models

## **1.5. Thesis organization**

The following organization is for the rest of the thesis. Chapter 2 highlights the literature of the relevant studies, followed by Chapter 3 which defines code clones suite of metrics, and contains definitions, taxonomies, examples, and metrics. Chapter 4 defines the empirical study of fault-proneness prediction using code clones metrics. Similarly, Chapter 5 defines the empirical study of fault-density prediction using code clones metrics. Finally, Chapter 6 concludes the contribution of the study and future work.

## **CHAPTER 2 LITERATURE REVIEW**

Several studies have been conducted to identify and measure the impact of software cloning with respect to software product quality in general. Some studies discuss the negative impact of software clones [29-31]. The literature is covering three aspects of studying code cloning, namely: impact of software clones on software quality, software clones taxonomies and types, and software clones detection techniques and tools.

### **2.1 Impact of Software Clones on Software Quality**

Monden et al. [5] conducted a study where they looked into the relation between the software clones and the quality of the software by clarifying the relation of code clones on the reliability and maintainability of the software. This study was applied to a legacy system that lived in the industry for more than 20 years. The experiment used a number of faults to measure the reliability, and revision numbers to estimate the maintainability. The estimation of reliability and maintainability were conducted for both cloned and none-cloned modules. For the reliability, the results of the study showed that generally clone-included modules are more reliable than none-clone modules, except from the large portions of clones that decrease the reliability drastically compared to the no-clone modules. For maintainability, the results of the study showed that the clone-included modules are less maintainable comparing to no-clone modules. The authors have recommended a quantitative analysis to reveal the reason of the findings of the work they have done.

Selim et al. [6] studied the factors that lead software systems, which incorporate cloned code, to be defect-prone. The study has focused on identifying the parts of the cloned

code that are high defect-prone by using certain predictors, which help the developers to identify the portions of their code that need intensive testing. Two categories of predictors were used in this study, namely: control predictors, and clone predictors. They were used to build survival models using “cox models”. The researchers have selected two open source systems and two clone detection tools to conduct the study. As a result, they found that the defect-prone level of the clone-code is system-dependent due to the inconsistent results between the studied systems. One clone code was high risk to have defects, and the other is the opposite. Also, modules with higher number of revisions need to be tested more intensively as they have higher risk of defects to occur.

Mondal et al.[7] empirically studied the argument whether code clones are harmful or not, and tried to identify the contradictions sources proposed by earlier studies. They tried to come up with a framework that minimizes the differences in evaluating the experiments as much as possible. Mondal et al. have selected nine of the leading studies that had contradictory results to build the new framework, and used almost all the metrics that are used by the nine methodologies. Their plan was to apply their study against 15 open source systems with variety of languages and different types. Initial results of some systems showed the contradictory findings, but at the end the researchers claimed that this empirical study will judge the debate of the previous studies and would suggest solutions to overcome any harmfulness of cloned code, if any.

Kozlov et al. [8] analyzed the relation between the internal quality attributes of open source projects and code cloning metrics. The study covered three groups of internal quality attributes, namely: complexity metrics, low level attributes, and high level derived

attributes. Apart from the internal quality attributes, seven code cloning metrics were studied as well, which were:-

1. Number of clones in a file
2. Number of files that have crosscutting clones
3. Ratio of cloned code tokens within a file
4. Ratio of tokens between any code clones across system
5. Population of number of code clones
6. Number of files that contain any code clones
7. Range of hierarchy of the source files that contain the code clones

Kozlov et al have come up with 14 hypotheses to study the relation between the code clones and the quality of the software system. Thirteen out of the 14 hypotheses were supported by the results of the study which indicated the strong relationships between the internal quality attributes and code clones metrics.

Kapsner and Godfrey [9] have studied the pros and cons of using code cloning through analyzing three categories of cloning patterns, namely: forking, templating, and customization. They have explored eight cloning patterns of large systems that highlight both the positive and negative motivations of cloning. The study mentioned several conditions where clones are used in a positive way and suggested solutions to manage code clones. The study also showed, in real software systems, that cloning is an acceptable design solution taking into consideration a clone management tool to manage the clones for the long run to maintain the changes accordingly. Lozano and Wermelinger [10] tried to assess the impact of using code clones on the changeability of software

systems. They came up with a hypothesis that claims the increase in changing efforts for a method is because it contains clones. The researchers represented the changeability efforts by computing two other measures, namely: likelihood, and impact of change. For consistency, the measure of change was considered for a period of time, either for the likelihood or the impact of the clones, from both measures they computed the work resulted by the change as a result of multiplying likelihood by the impact values and the lower value of the work will reflect a better changeability. The final result of this study is a method that has clones may increase the work of change.

Krinke [11] has studied the stability of clone code over the non-clone code of 5 large open source software systems. These 5 systems have long life development duration of more than 200 weeks of evolution. The study yielded the following interesting observations: 1) the most dominant factor of stability issue is the massive number of deletions in clone code, 2) in general clone code has less number of additions, deletion, and other changes on average than non-clone code, 3) the clone code is more stable than non-clone code. Krinke has suggested that eliminating the dominant factor which is deletions of clone code will increase the stability of the clone code and it will be more stable than non-clone code. Krinke extended his work to estimate code stability by comparing the age of a cloned code against non-cloned code, he yielded that cloned code is more stable than non-cloned code[39].

Juergens et al. [40] have studied the relationship between inconsistent clones and defect prone, they studied large scales subject systems. They came up with a new algorithm to detect clones inconsistencies and it is an open source tool. Their study yielded

inconsistent changes are significant lead to defects which increase the efforts of maintenance.

Juergens et al. [41] have studied the application of code clone detection to assess the quality of requirements specifications to detect redundant information. The case study covered more than 8000 pages of requirements specifications. Juergens et al. have reached the conclusion that clones in requirements specifications are harmful.

Rahman et al. [27] explored the relationship between code clones and defect proneness on free block code fragments. Their findings were not in agreement with the claim that code cloning is “a bad smell”. Moreover, they said incorporating code clones in development could be safe.

Bettenburg et al. [25] studied the relationship between code clones and software defects at the release level as opposite to other studies which focused on the revision level. They used three subject systems for their case study, they yielded that clones do not have major impact on defect proneness of the software quality at release level. They have observed significant difference between their findings as rate of defects of the software at release level when compared with the revision level, which conclude that developers are able to manage clones evolution while software development.

Gode and Harder [26] have studied the impact of code clones on maintenance efforts by analyzing clone stability and compare it with the stability of non-cloned code. They were in agreement with work results done by Krinke [11] which say code clones are more stable than non-cloned code in general. They conducted their study against two open source subject systems, exactly same as the subject systems that Krinke used in his study.

In this research the impact of code cloning will be studied on a particular aspect of software functional suitability which is functional correctness using two software metrics namely: fault density and fault proneness. To the best of our knowledge, we believe that the impact of code cloning on software quality was not explored thoroughly on the class level using adequate number of subject systems and wider range of clone metrics. Hence, impact of code cloning will be studied on the class level in five object-oriented software systems. The focus on the class level comes from the fact that classes in object-oriented is the basic building block of object-oriented software. In the literature, only one study was conducted to study code clones in fault-prediction on a single system with three different releases, which is not enough to generalize any findings. The following table is summarizing the previous studies on code clones in fault prediction.

**Table 1: Literature summary of studies on code clones**

Research	Granularity	Modeling approach	# of modules	# of systems	Clone metrics	Quality Attribute(s)	Cloning impact results
Baba et al. (2008)	Component	Logistic Regression	40, 32	2	2 metrics	Fault-Prone	Clone metrics improved fault-prediction
Kamei et al. (2011)	File	Logistic Regression	(8,313),(9,663) ,(11,525)	one (3releases)	5 metrics	Fault-Density	For large modules, clone metrics improved fault-prediction
Monden et al. (2002)	File	None	2000	1	2 metrics	Reliability and maintainability	Increase reliability and decrease maintainability
Lozano et al. (2007)	Methods	None	N.A	1	None	Changeability	Clones increase maintenance efforts
Kasper and Godfrey (2008)	Function, Free blocks	None	783, 530	2	None	Maintainability	Positive impact on maintainability of software system
Krinke (2008)	Free blocks	None	N.A	5	None	Stability	Cloned code are more stable than non-cloned code
Juergens et al. (2009)	Free blocks/Tokens	None	N.A	5	None	Program correctness	Inconsistent changes of clone increase faults
Juergens et al. (2010)	SRS Documents	None	8,667 pages	28	None	Requirement redundancy	Lower the quality of SRS which implies increase development and maintenance efforts
Rahman et al. (2010)	Free blocks	None	N.A	4	3 metrics	Fault-proneness	Cloning is not harmful
Selim et al. (2010)	Function	Cox hazard model	N.A	2	9 metrics	Fault-proneness	Risk of clones is system dependent
Bettenburg et al. (2010)	Free blocks	None	N.A	3	2 metrics	Defect proneness/Post-release quality	No significant impact on post-release level
Göde&Harder (2011)	Free blocks/Tokens	None	N.A	2	4 metrics	Stability	Cloned code are more stable than non-cloned code and might require less maintenance efforts
Krinke (2011)	Free block, File	None	2202, 6406, 552	3	None	Stability	Cloned code are more stable than non-cloned code
Our study	Class	Logistic/Linear Regression, ANN	229, 256, 741, 875, 935	5 systems	13 metrics	Functional correctness	Cloning increases fault-proneness, non-cloned code has lower fault-density

## **2.2 Software Clones Taxonomies and Types**

This section discusses different dimensions of software clones taxonomies based on what have been surveyed in the literature.

### **2.2.1 Taxonomies Based on Similarity:**

Mayrand et al. [12] Taxonomy: they have done a systematic taxonomy with 8 levels of clones for functions, where every level has some metrics to be used as comparison criteria.

Balazinska et al. [42] Taxonomy: This taxonomy has designed four levels of classification of clones where the first level has 2 categories of similarity, the second level has 3 categories of classification which talk about token differences and attributes, the third level has 10 categories of single token differences, and the fourth level has 3 categories based on token-sequence differences.

Bellon and Koschke [15] Taxonomy: They came up with a taxonomy that has three categories of to be used in comparison studies for the detection tools.

Davey et al. [43] Taxonomy: They came up with 4 types of clones that are very similar to the taxonomy that Bellon and Koschke have done based on similarity between code portions, except that they consider a new type which considers the coincident clones.

Kontogiannis [44] Taxonomy: Kontogiannis came up with four types of clones that are based on procedural code cloning which are considered to be basic types.

### **2.2.2 Taxonomies Based on Clone Location**

Kapsner and Godfrey [45] Taxonomy: They have provided a different taxonomy among other taxonomies which the classification was based on a hierarchy defined by two characteristics of function and location. The taxonomy is divided into three categories, 1) based on the source file location that contains the clones, 2) based on type of code structure, 3) based on degree of similarity and overlap.

Monden et al. [5] Taxonomy: The motivation to come up with this taxonomy is to find the relation between code cloning and quality of the software, it differentiates between the clones if they are in the same module (source file) or outside or could be a hybrid.

### **2.2.3 Taxonomies Based on Refactoring Opportunities**

Balazinska et al. Taxonomy: This taxonomy is associating each clone category a risk value for potential removal and the removal strategy.

Fanta and Rajlich's [46] Taxonomy: Their taxonomy based on object-oriented code that is probably will be removed, by dividing clones in two types namely: function clones and class clones.

Golomingi's [47] Taxonomy: He proposed a number of scenarios of clone relationships and a set of refactoring methods

## **2.3 Software Clones Detection Techniques and Tools**

During the last two decades many detection techniques of code clones detection have been developed and new detection tools were developed as well. Several comparison studies that investigated software clones detection techniques and tools are listed below:

Roy and Cordy [2], came up with taxonomy of scenarios to build the model of creating the different types of clones (Type-1, Type-2, Type-3, and Type-4) and evaluate the quality in addition to the classified techniques and tools.

The comparison study was conducted over 6 detection tools, where discovering strengths and weaknesses of the tools was the main objective. Each tool was running against 8 software systems to detect the clones with different techniques, and after that results were collected and analyzed [18].

Burd and Bailey [48], have conducted an evaluation study that shares almost the same goals of what Bellon et al. have done but for 5 detection tools.

Roy and Cordy [49], have developed a framework for empirical evaluating code clone detection tools that automates the process of measuring recall and precision of different detection tools.

As a result of surveying the literature, we have spotted some detection tools that might be helpful to our study. A comparison table has been constructed as follows:

**Table 2: Summary table of clone detection tools from the literature**

Tool	Detection Technique	Supported Language(s)	Clone Type	Clone Granularity
SimScan	Tree-based	Java	1, 2, 3	Free Text
CCFinder	Token-based	C, C++, Java, COBOL	1, 2, 3	Free Text
Simian	Text-based	C, C++, C#, Java,	1, 2	Free Text
NiCad	Text-based	C, C#, Java	1, 2, 3, 4	Any structure
CCFinderX	Token-based	C, C++, C#, Java, COBOL	1, 2, 3	Free Text

As a result out of this comparison analysis, we have chosen CCFinderX to detect code clones from the five java subject systems.



## CHAPTER 3 TYPES AND USED METRICS SUITES

In this chapter, code clone types and calculated metrics are discussed in details with some examples and illustrations in addition to C&K metrics. As the practice of simple reuse activity is commonly incorporated by copy and paste with or without modifications in the cycle of software development, we need to analyze and try to identify such advantages or disadvantages out of using code clones. After introducing clone types, clone metrics will be discussed with regards to definitions, scale and range, and what implications are out of these metrics on software.

### 3.1 Clone Types

Many researches discussed clone types and taxonomies are mentioned in (Sec 2.2). For the purpose of this study, two dimensions of clone types were chosen to be the part of code clones study on functional correctness, namely clones that are based on similarity and clones that are based on location.

- **Clones that are based on similarity**

Similarity of code fragments can be classified in two main categories [2], namely: 1) textual similarity, 2) functional similarity (semantic). In this study, textual similarity will only be considered, whereas functional similarity has a limitation that there is no detection tool exist to detect functional clones [2].

Textual clones are categorized into three types, namely Type-1, Type-2, and Type-3. The criterion of dividing the clones into three types is the degree of similarity between the clone fragments. Below are the definitions of the three types with examples.

### Type-1:

In this type, the two code fragments are exact copies except the whitespaces and comments might be different.

#### Example of Type-1:

<pre>Fragment A: if (x &gt;= y) { z = w + y; // Comment1 w = w + 1;} else z = w - x; //Comment2</pre>	<pre>Fragment B: if (x&gt;=y) { // Comment1' z=w+y; w=w+1;} else // Comment2' z=w-x;</pre>
---	--

Fragment B is an exact copy of A except that some comments and whitespaces are different, which tells that it is a type-1 clone.

### Type-2:

This type is very similar to Type1 except fragment A may have different identifiers, variable types, comments and layout than fragment B.

#### Example of Type-2:

<pre>Fragment A: if (x &gt;= y) { z = w + y; // Comment1 w = w + 1;} else z = w - x; //Comment2</pre>	<pre>Fragment B: if (a &gt;= b) { // Comment1' y = x + b; x = x + 5; //Comment3 } else y = x - a; //Comment2'</pre>
---	---

Fragment B is a copy of A except that some of the identifiers, variables and whitespaces are different which tells that it is a type-2 clone.

### Type-3:

This type is very similar to type-2 except that extra modifications like having deleted, added, or updated statements.

Example of Type-2:

<pre>Fragment A: if (x &gt;= y) { z = w + y; // Comment1 w = w + 1;} else z = w - x; //Comment2</pre>	<pre>Fragment B: if (x &gt;= y) { z = w + y; // Comment1 e = 1; // This statement is added w = w + 1; } else z = w - x; //Comment2</pre>
---	--

Fragment B is a copy of A except that a new line was added and some of the identifiers, variables and whitespaces are different which tells that it is a type-3 clone.

### **Clones that are based on location**

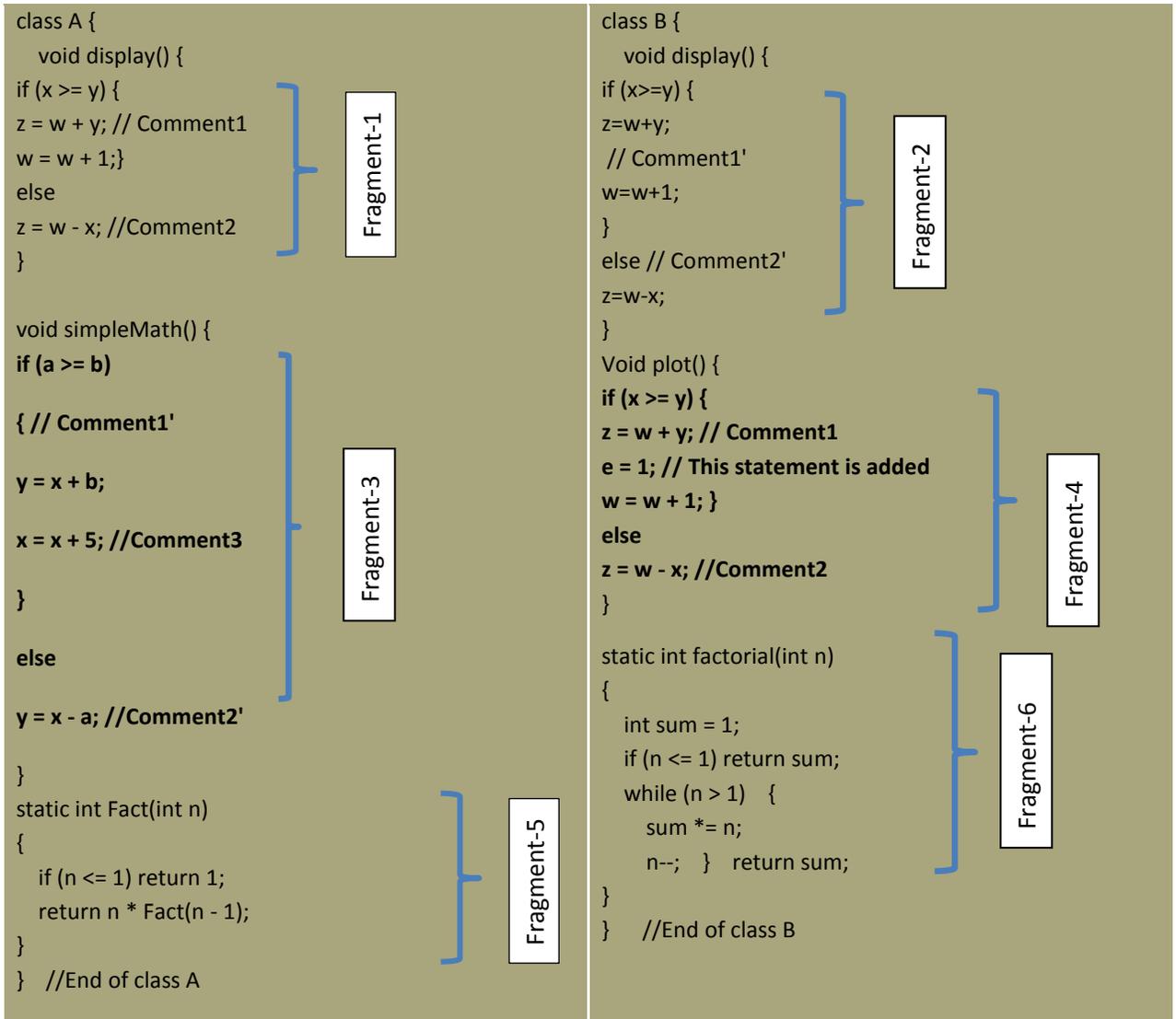
One of the aspects that was investigated while studying code clones is clone location as reported by Monden et al.[5]. In their study, they came up with two classifications for clone-pairs, namely in-module and inter-module clone pairs. In our study, inter-clones, intra-clones and hybrid-clones terms were named to classify the clones that are based on location. The following are the definitions of the clones based on location.

Inter-clone: This type is corresponding to two clone fragments that are located into two different files (in our case two different object-oriented classes).

Intra-clone: This type is corresponding to two clone fragments that are located within the same file (in our case same object-oriented class).

Hybrid-clones: Which refers to the clone fragment that has two corresponding pairs, one within the same file and the other pair is outside the file.

The following is a general example that contains all the types mentioned above, namely type-1, type-2, type-3, inter-clone, intra-clone, and hybrid clones.



In the previous example, Fragment-1 and Fragment-2 make a type-1 clone-pair where both fragments are exact ones except whitespaces and comments are different. Whereas Fragment-1 and Fragment-3 are making type-2 clone-pair since both fragments are

similar except in the identifiers and variable names including whitespaces are different. Fragment-3 and Fragment-4 make a type-3 clone-pair, where they are similar code fragments but there was an added statement in Fragment-4 with compare to Fragment-3. The last example is for type-4, which is represented by the clone pair of Fragment-5 and Fragment-6, where they are similar in the functionality of calculating the factorial of a given number but implemented into different coding methods. One coded in iteration method and the other in recursive method.

### **3.2 Clone Metrics**

The clone metrics used in this study can be grouped into two sets, namely existing metrics and new metrics. The existing metrics are system generated by detection tools, and the new metrics are new to be used to study the impact on functional-correctness and they are manually collected metrics. The set, which was generated by clone detection tools and analysis, consists of: NBR (neighbor): Count of classes containing a cloned fragment from current class, RSA (ratio of similarity between another files): it is the percentage of tokens covered by a clone between a class (in our case a file) and another class, RSI (ratio of similarity within the class): it is the percentage of tokens similar to another within a class, CVR (coverage): represented by  $\max(\text{RSA}, \text{RSI})$ , RNR (ratio of non-repeated tokens): Ratio (percentage) of tokens that are not included in repeated part of a code fragment of the code clone [40], Total Clone fragments (TCF), Clone-Density (CD) (number of clones per class normalized by size of the class based on lines of code), CRFL: the percentage of cloned LOC in class. The second set is clone metrics, which

was produced by our study based on location and similarity types. Further details of the 13 clone metrics such as definitions and examples are listed below.

***M1: Neighbor (NBR)***

Neighbor is a metric that refers to the number of classes that share a particular code fragment. It ranges from 0 to (n-1) where n is the total number of classes.

***M2: Ratio of similarity between files (RSA)***

It is the percentage of code covered by a clone between a file (in our case a class) and another class. It ranges between 0.0 to 1.0 where 0 value means there is no similarity between the two classes and 1 means the whole class was copied.

***M3: Ratio of similarity within the class (RSI)***

It is the percentage of code covered by a clone between a file (in our case a class) and another class. It ranges between 0.0 to 1.0 where 0 value means there is no similarity within the class and 1 means half of the class is a clone of the other half.

***M4: Coverage (CVR)***

It is the percentage of code covered by a clone within the class or outside the class. So, it ranges from 0.0 to 1.0 similarly to RSA and RSI.

***M5: Ratio of non-repeated tokens (RNR)***

It is the ratio (percentage) of tokens that are not included in the repeated part of a code fragment of the code clone. It ranges from 0.0 to 1.0.

***M6: Total Clone fragments (TCF)***

This metric represents number of code fragments that have clones whether internal clones or external clones.

***M7: Percentage of cloned LOC in class (CRFL)***

This metric represent the ratio of cloned code in number of lines of code in the class. It ranges from 0.0 to 1.0 where 0 means there is no clone in the class and 1 means the whole class is cloned somewhere else.

***M8: Clone density (CD)***

This metric measures the clone density of a class by dividing number of clones by LOC of the class and multiplied by 1000 to get more readable figures.

***M9: number of type-1 clones (T-1)***

This metric represents number of code fragments that are of type-1 clones.

***M10: number of type-2 clones (T-2)***

This metric represents number of code fragments that are of type-2 clones.

***M11: number of type-3 clones (T-3)***

This metric represents number of code fragments that are of type-3 clones.

***M12: number of inter-clone fragments (inter-clone)***

This metric represents number of code fragments that are of type inter-clone.

***M13: number of intra-clone fragments (intra-clone)***

This metric represents number of code clones that are of type intra-clone.

There are some relations between clone metrics used in this study such as:

$$M1 = (M9: T-1) + (M10: T-2) + (M11: T-3) = (M12: \text{inter-clone}) + (M12: \text{inter-clone})$$

$$M4 = \max (M2: \text{RSA}, M3: \text{RSI}).$$

Below is a subset of generated metrics out of class A and class B that were used earlier to explain types of clones based on similarity and location:

<b>Class</b>	<b>M9: T-1</b>	<b>M10: T-2</b>	<b>M11: T-3</b>	<b>M12: <i>inter-clone</i></b>	<b>M12: <i>inter-clone</i></b>
<b>A</b>	1	1	0	1	1
<b>B</b>	1	0	1	1	1

### 3.3 C&K Metrics Definitions

This suite of metrics is a well-known object-oriented suite of class structural metrics. These metrics are used in the comparison study against the code clone metrics in predicting fault-proneness and fault-density. This suite of metrics consists of six metrics, namely weighted methods per class (WMC), depth of inheritance tree of a class (DIT), number of children (NOC), coupling between objects (CBO), response for a class (RFC), and lack of cohesion in methods (LCOM). Further details are listed below about C&K metrics.

WMC: It measures number of methods defined locally within a class given that all methods have same complexity.

DIT: It measures number of parent classes of a class, by calculating the inheritance hierarchy of a class's ancestors.

NOC: It measures the number child classes that are directly related to a class.

CBO: When two classes are coupled that means there is shared member functions and/or instance variables, thus CBO is the number of classes that are coupled with a class of interest.

RFC: It is the number of methods that can be called directly by member methods including the internal methods of a class.

LCOM: A class is called cohesive when it is operating a single, so, lack of cohesion is a value between 0 and 1 that represent a state of class when its methods' pairs are not sharing instance variables. The formula used to calculate this metric is: non-cohesive methods – cohesive methods, where negative values are set to zero and considered a cohesive class.

## **CHAPTER 4 EMPIRICAL STUDY OF SOFTWARE CLONES AND FAULT-PRONENESS**

In this chapter the fault-proneness prediction using clone metrics will be explored empirically to find useful results regarding clone metrics. Additionally, it contains a comparison study between clone metrics suite and C&K metrics suite performance in predicting fault-proneness.

### **4.1 Goal and Research Questions:**

The goal of this empirical experiment is to analyze the code clone metrics for the purpose of exploring the relationships with respect to class fault-proneness from the point of view of the software developer in the context of object-oriented open source systems.

In this part of the study, the following research questions are the motivation of the research:

- A. Are there relationships between code clones and fault proneness of object-oriented classes?

For this question three hypotheses were constructed:

HYP-A1:

H0: There are no relationships between code clones and fault proneness of object-oriented classes

H1: There are relationships between code clones and fault proneness of object-oriented classes

HYP-A2:

H0: There is no relationship between clone types and fault proneness

H1: There is a relationship between clone types and fault proneness

HYP-A3:

H0: There is no relationship between clone location and fault proneness

H1: There is a relationship between clone location and fault proneness

B. Are clone metrics good indicators of fault proneness?

For this question we have 13 hypotheses:

HYP-B1:

H0: number of clone fragments is not a good indicator of fault proneness

H1: number of clone fragments is a good indicator of fault proneness

HYP-B2:

H0: clone density is not a good indicator of fault proneness

H1: clone density is a good indicator of fault proneness

HYP-B3:

H0: CRFL is not a good indicator of fault proneness

H1: CRFL is a good indicator of fault proneness

HYP-B4:

H0: NBR is not a good indicator of fault proneness

H1: NBR is a good indicator of fault proneness

HYP-B5:

H0: RSA is not a good indicator of fault proneness

H1: RSA is a good indicator of fault proneness

HYP-B6:

H0: RSI is not a good indicator of fault proneness

H1: RSI is a good indicator of fault proneness

HYP-B7:

H0: CVR is not a good indicator of fault proneness

H1: CVR is a good indicator of fault proneness

HYP-B8:

H0: RNR is not a good indicator of fault proneness

H1: RNR is a good indicator of fault proneness

HYP-B9:

H0: Inter-class clone is not a good indicator of fault proneness

H1: Inter-class clone is a good indicator of fault proneness

HYP-B10:

H0: Intra-class clone is not a good indicator of fault proneness

H1: Intra-class clone is a good indicator of fault proneness

HYP-B11:

H0: Type-1 clone is not a good indicator of fault proneness

H1: Type-1 clone is a good indicator of fault proneness

HYP-B12:

H0: Type-2 clone is not a good indicator of fault proneness

H1: Type-2 clone is a good indicator of fault proneness

HYP-B13:

H0: Type-3 clone is not a good indicator of fault proneness

H1: Type-3 clone is a good indicator of fault proneness

C. Do clone metrics have better fault proneness prediction than C&K metrics?

HYP-C:

H0: Clone metrics are not better predictors of fault proneness than C&K metrics

H1: Clone metrics are better predictors of fault proneness than C&K metrics

D. Does the combination of clone metrics and C&K metrics as input in prediction models of fault proneness yield better results than clone metrics?

HYP-D:

H0: The combination of clone metrics and C&K metrics as input in prediction models of fault proneness does not yield better results than clone metrics only

H1: The combination of clone metrics and C&K metrics as input in prediction models of fault proneness yields better results than clone metrics only

E. Does the combination of clone metrics and C&K metrics as input in prediction models of fault proneness yield better results than C&K metrics?

HYP-E:

H0: The combination of clone metrics and C&K metrics as input in prediction models of fault proneness does not yield better results than C&K metrics only

H1: The combination of clone metrics and C&K metrics as input in prediction models of fault proneness yields better results than C&K metrics only

## **4.2 Variables and Data Collection:**

The study of the relationships between software clones and fault proneness used thirteen metrics as independent variables. Fault proneness is represented by a dependent variable which has only two values (dichotomous), either a class has a fault or not. The fault data were imported from a published study [50]. To measure the feasibility of using the clone metrics to predict fault proneness, we used a probabilistic statistical

classification model called logistic regression. Logistic regression was selected because of the fact that a simple classifier such as logistic regression gives similar results to sophisticated classifiers [51].

Clone metrics have been collected from five subject systems, Velocity, Synapse, Camel, Ant, and Xalan. There are common characteristics between subject systems such as they all are written in Java. In addition, they are open source systems and were obtained from Apache repositories. The subject systems have variant combinations of different sizes of classes. The variance in sizes of the subject systems will minimize the possibility of the data to be biased to certain type or category of systems.

**Table 3: Descriptive Statistics of the subject systems**

<b>System:</b>	<b>Total # Classes</b>	<b>No. of classes with faults</b>	<b>No. of classes with clones</b>
<b>Velocity</b>	229	78 (34%)	56 (24%)
<b>Synapse</b>	256	86 (34%)	128 (50%)
<b>Ant</b>	741	166 (22%)	288 (39%)
<b>Xalan</b>	875	411 (47%)	381 (47%)
<b>Camel</b>	935	188 (20%)	174 (20%)
<b>Total</b>	3036	929 (31%)	1027 (34%)

The above table shows the variance in the selected systems for this study in terms of size (classes), faults and clones. Comprehensive system is an auxiliary system that combines all the five subject systems into one system.

#### **4.2.1 Variables Used in the Study**

Data that were used in this study can be grouped into three sets: first, ready data which were collected from previous studies, this set has the following data: faults and

line of code. The second set is data that were generated by clone detection tools and analysis which are: NBR (neighbor): Count of classes containing a cloned fragment from current class, RSA (ratio of similarity between another files): it is the percentage of tokens covered by a clone between a class (in our case a file) and another class, RSI (ratio of similarity within the class): it the percentage of tokens similar to another within a class, CVR (coverage): represented by  $\max(\text{RSA}, \text{RSI})$ , RNR (ratio of non-repeated tokens): Ratio (percentage) of tokens that are not included in repeated part of a code fragment of the code clone [49], Total Clone fragments (TCF), CRFL: the percentage of cloned LOC in class. The third set is data were produced by our study based on some taxonomy, like whether a clone is exist within the same class or with a different class that resides in different file, and also, based on the clone fragment types which depend on degree of similarities between clones.

For comparison purposes Chidamber & Kemerer (C&K) suite of metrics were included for the same subject systems, data were imported from [50].

#### **4.2.2 Data Collection Tools:**

Clone metrics were collected from two tools, namely: CCFinderX and VisCad. CCFinderX [13] is a token-based clone detection and visualization tool that detects clones and produces metrics as well. It is an evolutionary version of the command line CCFinder, it provides extra features and has a GUI that provides some metrics and enables the user to preview the clone code fragments and compare them visually side by side. VisCad [52] is a clone visualization tool that analyzes clones and produces clone metrics as well. Data collection went through two phases which are 1) Clone detection,

and 2) metrics collection. The below section is explaining the two phases and the tools used in each:

First, in order to detect the clones CCFinderX is used to generate the clone fragments for the five subject systems. CCFinderX from the literature has higher recall and comparable precision [47]. The default settings of the tool were used as is, 30 tokens as minimum length of tokens. This setup will ensure only non-trivial clones to be detected, this setup was used by many research projects [53]. CCFinderX provided visual distribution of the clones over different packages of the system.

Second, in order to get the clone metrics on the class level, both CCFinderX and VisCad were used to extract clone metrics on the class level. After sufficient search and comparisons among many clone detection tools, CCFinderX was found to be widely used in many studies [48]. CCFinderX provides seven types of metrics on class level but 5 clone metrics were selected to be part of this study and the remaining two metrics namely: 1) length of clone fragments in tokens and 2) number of clone fragments. The two metrics were eliminated because the length in tokens is not a standard measurement and the second metric which is number of clone fragments was dropped because it contained quite big number of false positives, instead manual corrected numbers of clones were included. VisCad is also providing four metrics on the class level which are 1) number of clone fragments 2) number of lines of code 3) number of lines of clone fragments 4) the percentage of cloned LOC in class. Metrics 1 and 4 are selected to conduct our study, and 2 and 3 were omitted for the reason that they are covered and normalized in metric 4. In addition, we have identified 5 clone metrics based on some categorization taxonomies.

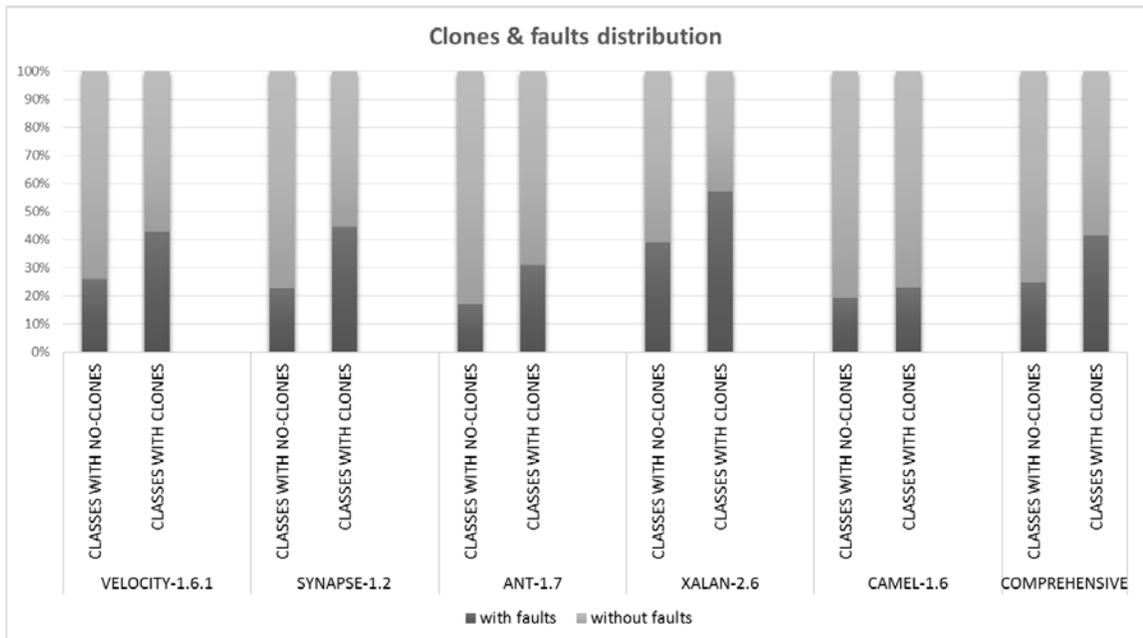
Since dealing with raw clone data is very challenging [52], a need for a tool to analyze and visualize is a desire. VisCad is a visualization and analysis tool that helps to get useful information from raw clone data that was produced by detection tools.

### **4.3 Descriptive Statistics**

After generating some statistics of the subject systems such as Min, Max, Mean, and Standard Deviation, we found some general observations of these subject systems. All of the systems as showed in the histogram [*Figure 1*] have a common feature which is the classes that contain clones are more faulty than those clone-free classes, this consistent reading looks to be an interested observation. The opted subject systems represent sufficient variations in size as number of classes, in addition to the percentage of classes that have clone/s. Fair different distributions are existing as well with respect to faults among the five systems.

Camel and velocity have the smaller average of number of clone fragments per class, whereas synapse and xalan have the highest average values. When the clone fragments are normalized with class KLOC, we observed that synapse has the highest value, while velocity and camel have the least values. The percentage of cloned LOC in a class was extremely high in xalan system camel was on the other extreme very low, whereas velocity, synapse and ant were almost close to each other.

**Figure 1: Histogram of Clones and Faults Distribution**



Further information can be found in the below tables:

**Table 4: Descriptive Statistics of Velocity**

Descriptive Statistics (velocity)				
	Mean	Minimum	Maximum	Std.Dev.
TCF	0.51092	0	13	1.3133
CD	2.55563	0	35.714	6.0914
CRFL	0.37005	0	16.046	1.6215
NBR	0.25328	0	4	0.6987
RSA	0.07958	0	0.997	0.2209
RSI	0.04483	0	0.994	0.1497
CVR	0.11947	0	0.997	0.2536
RNR	0.92103	0.043	1	0.1089
Inter-clones	0.23144	0	4	0.5876
Intra-clones	0.27948	0	13	1.1996
Type-I	0.20961	0	4	0.5847
Type-II	0.14847	0	4	0.5885
Type-III	0.15284	0	7	0.6677
wmc	9.02183	0	153	14.1351
Dit	1.67686	1	5	0.8887
Noc	0.43668	0	39	2.7548
Cbo	10.8079	0	80	12.7227
Rfc	22.9782	0	250	27.3691
Lcom	80.3406	0	8092	554.868

From the above table, we noticed velocity system has a low average of TCF when compared with other subject systems, similarly applies to clone density. The locations of the clones in velocity system were almost equal on average between inter-clones and intra-clones. The type-1 clones are higher than type-2 and type-3. From the C&K metrics values, it reflects that the degree of inheritance and number of children classes are relatively low when compared with other subject systems. This could be due to the small size of the system in number of classes.

**Table 5: Descriptive Statistics of Synapse**

Descriptive Statistics (synapse)				
	Mean	Minimum	Maximum	Std.Dev.
TCF	1.47656	0	12	2.1979
CD	7.29426	0	63.83	11.8548
CRFL	0.70372	0	5.083	1.0893
NBR	4.625	0	28	8.0206
RSA	0.13759	0	0.992	0.2205
RSI	0.08355	0	0.976	0.1684
CVR	0.20399	0	0.992	0.2549
RNR	0.91536	0.234	1	0.0827
Inter-clones	0.98047	0	8	1.5093
Intra-clones	0.49609	0	12	1.4686
Type-I	0.33984	0	6	0.9361
Type-II	0.50781	0	8	1.243
Type-III	0.62891	0	6	1.2299
wmc	7.62109	0	67	9.1098
dit	1.64063	1	5	0.7845
noc	0.41016	0	19	2.1872
cbo	12.7539	0	83	11.7426
rfc	29.5273	0	172	25.8331
lcom	41.0859	0	1931	175.819

For synapse system, we noticed it has the highest average of TCF and CD when compared with other subject systems. The location of the clones in synapse system, where the inter-clones have almost two times the intra-clones in the system. The type-1 clones are less than type-2 and type-3 whereas type-3 is the highest value of clone types. From the C&K metrics values, the scores of synapse are much similar to velocity system, it reflects that the degree of inheritance and number of children classes are relatively low compared with other subject systems. Again, this could be due to the relatively small size of the system in number of classes.

**Table 6: Descriptive Statistics of Ant**

Descriptive Statistics (ant)				
	Mean	Minimum	Maximum	Std.Dev.
TCF	1.22672	0	23	2.5223
CD	4.2636	0	125	8.6587
CRFL	0.38031	0	12.662	1.2115
NBR	1.06343	0	17	2.7054
RSA	0.10364	0	0.995	0.2207
RSI	0.04078	0	0.719	0.1015
CVR	0.13476	0	0.995	0.2306
RNR	0.92568	0.401	1	0.0714
Inter-clones	0.64238	0	13	1.3112
Intra-clones	0.58435	0	15	1.8795
Type-I	0.4143	0	12	1.1797
Type-II	0.39676	0	12	1.1386
Type-III	0.41565	0	14	1.1866
wmc	11.11471	0	120	11.9931
dit	2.52632	1	7	1.3982
noc	0.73549	0	102	4.813
cbo	11.10121	0	499	26.4039
rfc	34.50202	0	288	36.0712
lcom	89.61538	0	6692	350.8238

For ant system, we noticed it has type-1 and type-3 are more popular than type-2. The location of the clones in ant system, the inter-clones is a bit higher than intra-clones in the system. From the C&K metrics values, the scores of ant system, it shows that the degree of inheritance is the highest compared with other subject systems.

**Table 7: Descriptive Statistics of Xalan**

Descriptive Statistics (xalan)				
	Mean	Minimum	Maximum	Std.Dev.
TCF	1.5543	0	46	3.4885
CD	3.9639	0	90.164	7.8479
CRFL	5.3335	0	142.558	21.2005
NBR	4.5051	0	53	12.2377
RSA	0.1885	0	0.999	0.3397
RSI	0.0405	0	0.99	0.1179
CVR	0.2203	0	0.999	0.3415
RNR	0.9264	0.176	1	0.0789
Inter-clones	0.784	0	18	1.6166
Intra-clones	0.7703	0	44	2.6829
Type-I	0.6789	0	24	1.6949
Type-II	0.5623	0	18	1.6409
Type-III	0.3131	0	42	1.7629
wmc	11.1166	0	133	16.2814
Dit	2.5223	1	8	1.444
Noc	0.5166	0	29	2.3329
Cbo	12.1714	0	168	17.9159
Rfc	29.5394	0	409	37.2219
Lcom	125.2491	0	7774	584.212

For xalan system, we noticed it has type-1 more popular than type-2 and type-3 are. The location of the clones in xalan system where almost equal on average between inter-clones and intra-clones. The NBR score is relatively high which tells that a sufficient number of inter-clones are existing in xalan system. From the C&K metrics values, the scores of xalan system, it shows that the degree of inheritance is relatively high compared with other subject systems.

**Table 8: Descriptive Statistics of Camel**

Descriptive Statistics (camel)				
	Mean	Minimum	Maximum	Std.Dev.
TCF	0.41649	0	13	1.2069
CD	2.57428	0	96.15	8.0377
CRFL	0.15795	0	6.7	0.5458
NBR	0.24411	0	6	0.741
RSA	0.05939	0	1	0.1892
RSI	0.02318	0	0.79	0.0962
CVR	0.08051	0	1	0.2066
RNR	0.9185	0.289	1	0.0829
Inter-clones	0.25161	0	13	0.9025
Intra-clones	0.16488	0	7	0.7779
Type-I	0.10064	0	6	0.4275
Type-II	0.07281	0	7	0.4619
Type-III	0.24304	0	9	0.8536
wmc	8.58458	0	166	11.2927
Dit	1.97216	0	6	1.2878
Noc	0.52891	0	39	2.6425
Cbo	11.14775	0	448	22.836
Rfc	21.70664	0	322	25.1681
Lcom	79.99893	0	13617	531.7098

For camel system, we noticed that type-2 is more popular than type-1 and type-3. The location of the clones in camel system, the inter-clones is higher than intra-clones in the system. From the C&K metrics values, the scores of camel system, it shows that the degree of inheritance is relatively on average compared with other subject systems.

**Table 9: Descriptive Statistics of Total**

Descriptive Statistics (total)				
	Mean	Minimum	Maximum	Std.Dev.
TCF	1.03888	0	46	2.5065
CD	3.78409	0	125	8.495
CRFL	1.7664	0	142.56	11.6362
NBR	2.04283	0	53	7.3703
RSA	0.11553	0	1	0.258
RSI	0.0392	0	0.99	0.117
CVR	0.14742	0	1	0.2707
RNR	0.92246	0.043	1	0.0815
Inter-clones	0.56046	0	18	1.3071
Intra-clones	0.47842	0	44	1.8631
Type-I	0.37232	0	24	1.1725
Type-II	0.33542	0	18	1.1643
Type-III	0.33114	0	42	1.2797
Wmc	9.88402	0	166	13.2085
Dit	2.21582	0	8	1.3481
Noc	0.55881	0	102	3.217
Cbo	11.54135	0	499	21.1423
Rfc	27.84448	0	409	32.3868
Lcom	92.13608	0	13617	491.5333

For total (aggregated) system, we noticed that all types are almost equal in average. The location of the clones in the aggregated system, the inter-clones is higher than intra-clones in the system. From the C&K metrics values, the scores of the aggregated system, they show that the degree of inheritance is relatively on average compared with other subject systems.

#### **4.4 Principal Component Analysis**

In order to study the correlations of a suite of metrics and find out whether the correlation between the metrics is weak or strong, Principal Component Analysis (PCA) is conducted to measure common dimensions between the metrics if any [54-58].

All the systems have three Principal Components (PC), except xalan which has additional Principal Component to be four in total. In general, there is no single metric that belongs to a specific PC across all the systems. This reflects that these metrics are not strongly correlated to each other. It was observed that some metrics coexist together over different systems, for example, (RSA, CVR) are found together in the same component across all the systems. (NBR, RSA, CVR), for velocity, synapse, and camel, this group belongs to the first component (PC-1) for three systems and in PC-2 in one system. (CD, RSA, CVR), for velocity, synapse, and camel, this group belongs to PC-1 for three systems and in PC-3 in one system.

In summary of PCA analysis, these clone metrics don't have strong correlations between them. In addition, there is no specific dimension that can be generalized for all of the subject systems.

The results of PCA analysis are summarized in the following tables and PCA detailed tables are in the Appendix A:

**Table 10: Principal Components of Velocity**

Velocity		
PC-1	PC-2	PC-3
CD	TCF	CRFL
NBR	Intra-clones	RSI
RSA	TypeII	RNR
CVR	TypeIII	
Inter-clones		
TypeI		

**Table 11: Principal Components of Synapse**

Synapse		
PC-1	PC-2	PC-3
CD	TCF	RSI
CRFL	Intra-clones	RNR
NBR	TypeI	TypeIII
RSA	TypeII	
CVR		
Inter-clones		

**Table 12: Principal Components of Ant Table**

Ant		
PC-1	PC-2	PC-3
TCF	CRFL	CD
RSI	NBR	RSA
Intra-clones	Inter-clones	CVR
TypeII	TypeI	RNR
TypeIII		

**13: Principal Components of Xalan**

Xalan			
PC-1	PC-2	PC-3	PC-4
TCF	CRFL	C	RNR
RSI	NBR	Inter-clones	
Intra-clones	RSA	TypeI	
TypeII	CVR		
TypeIII			

**Table 14: Principal Components of Camel**

Camel		
PC-1	PC-2	PC-3
CD	RSI	TCF
CRFL	RNR	Inter-clones
NBR	Intra-clones	TypeI
RSA	TypeIII	TypeII
CVR		

## 4.5 Mann-Whitney Test

In this section, we are trying to answer question A and its sub-questions (sec.4.1). In this empirical study, Mann-Whitney test is used to help in answering the question of having a relationship (if any) between code clones and fault-proneness. Mann-Whitney test is a non-parametric test that gives cumulative probabilities of certain null hypothesis by measuring the sum of the ranked two-sample data [59]. The software used to conduct Mann-Whitney tests is STATISTICA software package. The readings of Mann-Whitney tests are summarized in the below tables:

**Table 15: Mann-Whitney test for general clones including all types**

Fault-Proneness	p-value
velocity	0.11083
synapse	<b>0.00022</b>
ant	<b>0.00001</b>
xalan	<b>0</b>
camel	0.29342
comprehensive	<b>0</b>

As we can see in the above table, the numbers in bold show the significance figures. Although velocity and camel systems' p-values are  $> 0.05$ , the general trend, as it is clear in [Figure 1], shows that there is a relationship between code clones and fault-proneness. So, we reject the null hypothesis of HYP-A1.

To answer question Q.A-2, Mann-Whitney test for all systems was conducted and presented below based on clone location, namely inter-clone, intra-clone, and hybrid-clone.

**Table 16: Mann-Whitney test for clone locations whether inter clones or intra clones**

Glossary B: Velocity	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	0.837674			
C	0.588365	0.54998		
D	0.370406	0.381106	0.741486	

Mann-Whitney test shows that there is no significant difference among the individual groups of data. This cannot lead us to any conclusion for clone location relationship with fault-proneness. Another visit to the histogram for fault distribution over the clone location groups will give more information that will help us in reaching a conclusion.

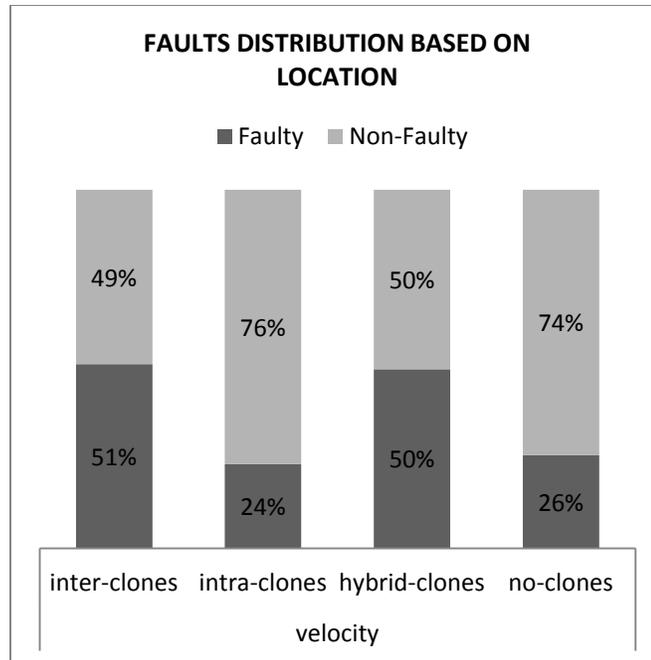


Figure 2: Velocity-Faults Distribution Based on Location

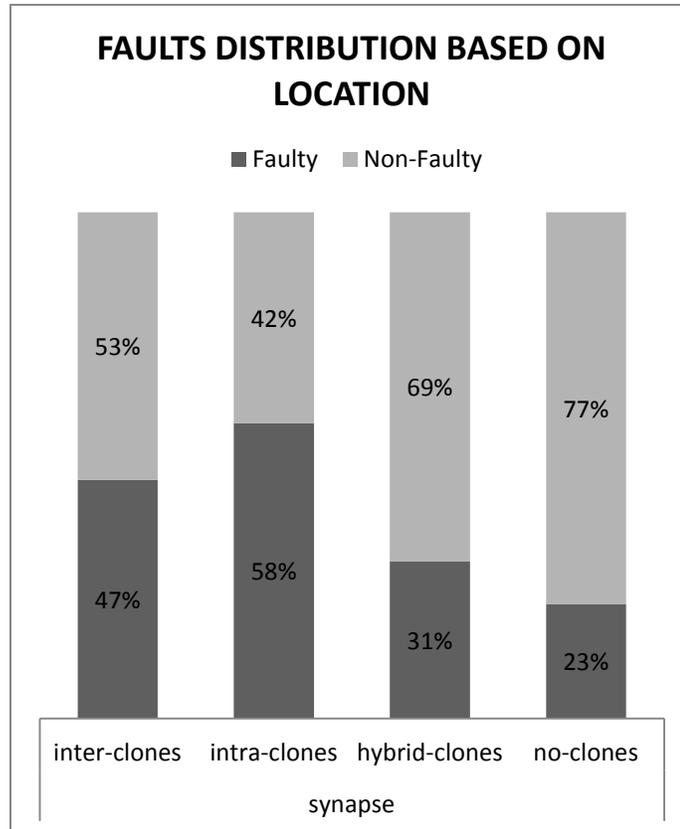
From the p-values above, there is no significant difference between clones based on location with respect to fault-proneness. However, the histogram of the system shows that classes having inter-clones and hybrid-clones are more fault-proneness than classes without clones. The summary in [Table 28] has a generic view of the test results with the context of all subject systems in this study.

Table 17: Mann-Whitney test for clone locations whether inter clones or intra clones

Glossary B: Synapse	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	0.530538			
C	0.195563	0.173503		
D	<b>0.002341</b>	<b>0.041395</b>	0.481821	

Mann-Whitney test shows that there are significant differences between inter-clones and the group of no-clones, similarly between intra-clones and no-clones group. This

observation alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-proneness. Another visit to the histogram for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.



**Figure 3: Synapse-Faults Distribution Based on Location**

From the p-values above, there are some significant differences between clones based on location groups of data with respect to fault-proneness. However, the histogram of the system shows the classes that have inter-clones, intra-clones and hybrid-clones are more fault-proneness than classes without clones. The summary in [Table 28] has a generic view of the test results with the context of all subject systems in this study.

**Table 18: Mann-Whitney test for clone locations whether inter clones or intra clones**

Glossary B: Ant	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	<b>0.012847</b>			
C	<b>0.024002</b>	0.790891		
D	0.264261	<b>0.000748</b>	<b>0.001512</b>	

Mann-Whitney test shows that there are significant differences between inter-clones and the groups of no-clones, intra-clones, and hybrid-clones. Similarly, between intra-clones, hybrid-clones and no-clones group. These observations alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-proneness. Another visit to the histogram for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.

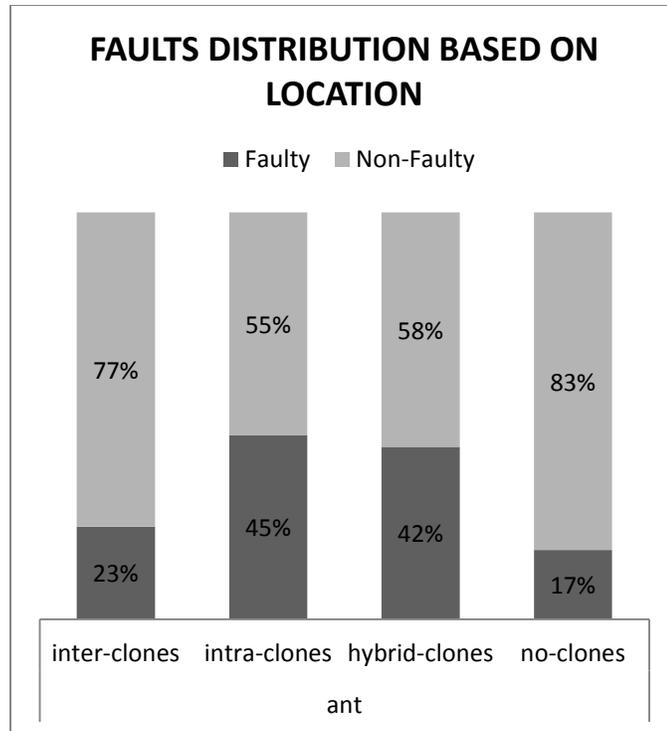


Figure 4: Ant-Faults Distribution Based on Location

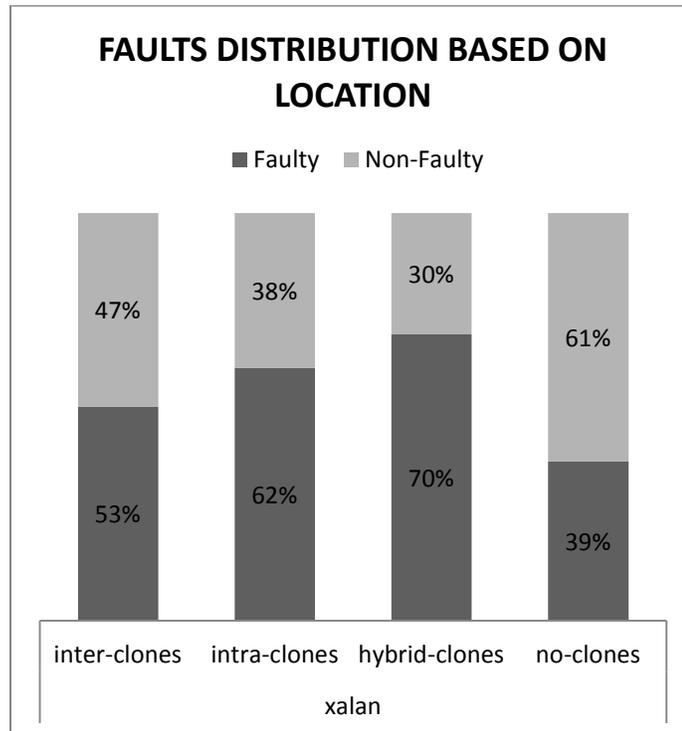
From the p-values above, there are some significant differences between clones based on location groups of data with respect to fault-proneness. However, the histogram of the system shows that classes having inter-clones, intra-clones and hybrid-clones are more fault-proneness than classes without clones. The summary in [Table 28] has a generic view of the test results with the context of all subject systems in this study.

Table 19: Mann-Whitney test for clone locations whether inter clones or intra clones

Glossary B: Xalan	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	0.289347			
C	<b>0.020035</b>	0.435781		
D	<b>0.002452</b>	<b>0.007499</b>	<b>0.000015</b>	

Mann-Whitney test shows that there are significant differences between inter-clones and the groups of no-clones, and hybrid-clones. Similarly, between intra-clones and no-

clones groups. These observations alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-proneness. Another visit to the histogram for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.



**Figure 5: Xalan-Faults Distribution Based on Location**

From the p-values above of xalan system, there are some significant differences between clones based on location groups of data with respect to fault-proneness. However, the histogram of the system shows that classes having inter-clones, intra-clones and hybrid-clones are more fault-proneness than classes without clones. The summary in [Table 28] has a generic view of the test results with the context of all subject systems in this study.

**Table 20: Mann-Whitney test for clone locations whether inter clones or intra clones**

Glossary B: Camel	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	<b>0.015312</b>			
C	0.716188	0.28341		
D	0.653338	<b>0.01442</b>	0.822319	

Mann-Whitney test for camel system shows that there are significant differences between inter-clones and the group of inter-clones. Similarly, between intra-clones and no-clones groups. These observations alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-proneness. Another visit to the histogram for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.

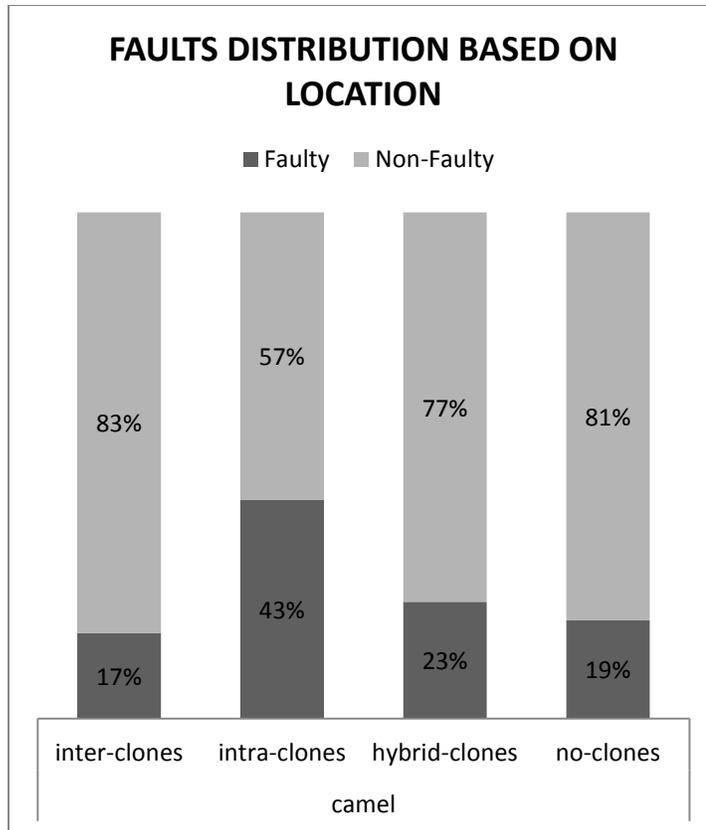


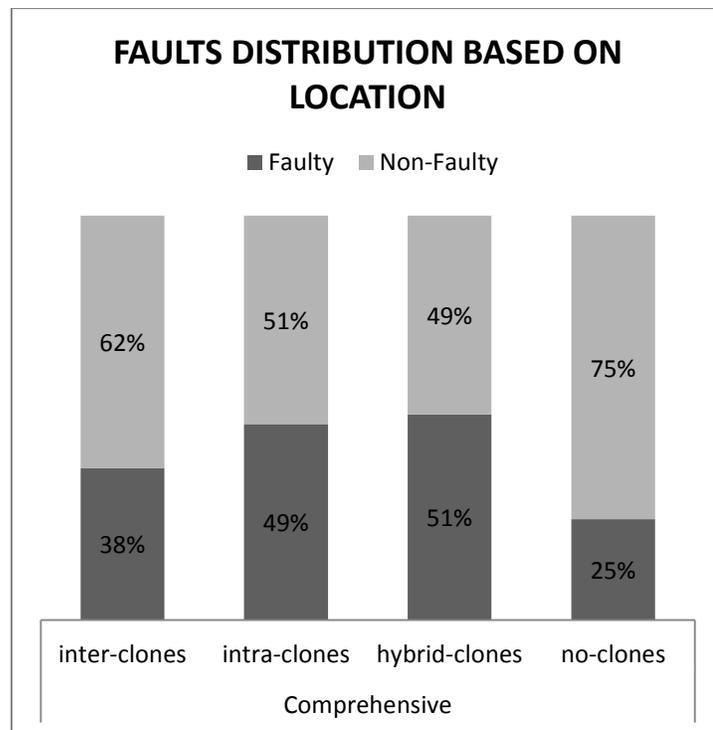
Figure 6: Camel-Faults Distribution Based on Location

From the p-values above of camel system, there are some significant differences between clones based on location groups of data with respect to fault-proneness. However, the histogram of the system shows the classes that have inter-clones, intra-clones and hybrid-clones are more fault-proneness than classes without clones. The summary in [Table 28] has a generic view of the test results with the context of all subject systems in this study.

Table 21: Mann-Whitney test for clone locations whether inter clones or intra clones

Glossary B: total	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	<b>0.027711</b>			
C	<b>0.006534</b>	0.708274		
D	<b>0.000001</b>	<b>0</b>	<b>0</b>	

Mann-Whitney test for the aggregated system shows us a glance for the overall systems behavior with respect to fault-proneness. The results show that there are significant differences between all different groups except between intra-clones and the group of hybrid-clones. These observations alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-proneness. Another visit to the histogram for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.



**Figure 7: Comprehensive-Faults Distribution Based on Location**

From the p-values above of the comprehensive system, there are many significant differences between clones based on location groups of data with respect to fault-proneness. However, the histogram of the system shows the classes that have inter-clones, intra-clones and hybrid-clones are more fault-proneness than classes without

clones. The summary in [Table 28] has a generic view of the test results with the context of all subject systems in this study.

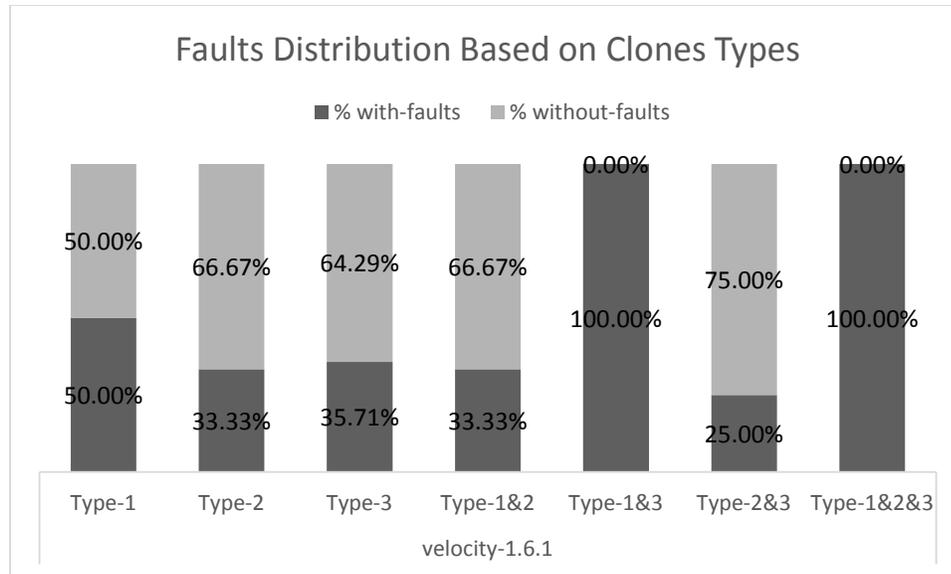
**Mann-Whitney test for clones based on types:**

Velocity:

**Table 22: Mann-Whitney Test Based on Types**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.471758							
C	0.399217	0.920573						
D	0.471758	1	0.920573					
E	1	1	1	1				
F	0.362051	0.789268	0.696835	0.789268	1			
G	1	1	1	1	1	1		
H	0.068368	0.912591	0.72815	0.912591	1	0.791214	1	

Similar to location based clones, Mann-Whitney test shows that there is no significant difference among the individual groups of data. This cannot lead us to any conclusion for clone types relationships with fault-proneness. Another visit to the histogram for fault distribution over the clone types groups will give more information that will help us in reaching a conclusion.



**Figure 8: Velocity-Faults Distribution Based on Types**

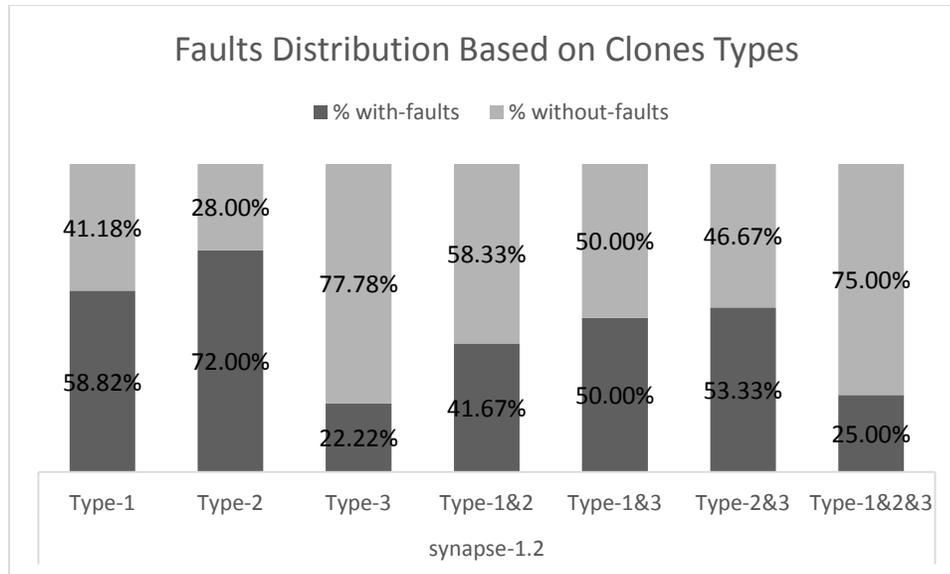
From the p-values above, there is no significant difference between clones based on types with respect to fault-proneness. However, the histogram of the system shows that all clone types in the classes except the ones that contain Type-2 and Type-3 are more fault-proneness than classes without clones. The summary in [Table 29] has a generic view of the test results with the context of all subject systems in this study.

Synapse:

**Table 23: Mann-Whitney Test Based on Types for Synapse**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.379672							
C	<b>0.006371</b>	<b>0.000053</b>						
D	0.370899	0.078965	0.177933					
E	0.66194	0.222127	0.077101	0.70255				
F	0.758467	0.236724	<b>0.023944</b>	0.554114	0.87278			
G	0.234322	0.071191	0.899503	0.563703	0.410594	0.326349		
H	<b>0.00164</b>	<b>0.000001</b>	0.952347	0.143415	0.054155	<b>0.010536</b>	0.912624	

Mann-Whitney test shows that there are significant differences among some of the individual groups of data such as Type-1 and Type-3, similarly Type2 and Type-3. Additionally, four groups have significant differences between them. This might not lead us to any conclusion for clones based on type relationships with fault-proneness. Another visit to the histogram for fault distribution over the clone types groups will give more information that will help us in reaching a conclusion.



**Figure 9: Synapse-Faults Distribution Based on Types**

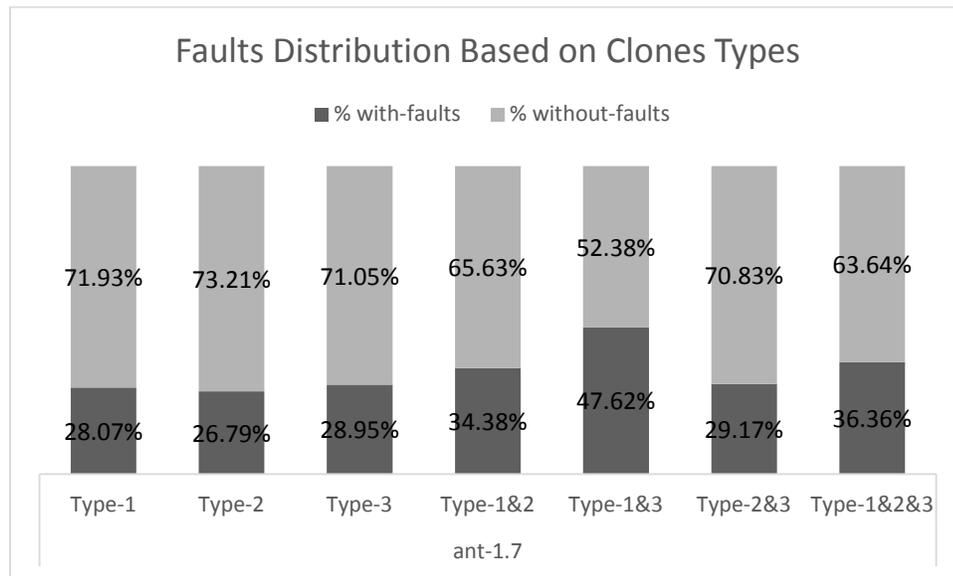
From the p-values above, there are significant differences between clones based on types with respect to fault-proneness. However, the histogram of the system shows that all clone types in the classes except the ones that contain Type-3 are more fault-proneness than classes without clones. This could be due to the fact that usually the developer gets something to clone and modify, will reassess the new code and probably resolve any faults. The summary in [Table 29] has a generic view of the test results with the context of all subject systems in this study.

Ant:

**Table 24: Mann-Whitney Test Based on Types**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.87893							
C	0.912091	0.785434						
D	0.536988	0.455452	0.577842					
E	0.10651	0.084094	0.109041	0.339557				
F	0.920888	0.828061	0.983615	0.682326	0.207859			
G	0.475282	0.406865	0.508444	0.881586	0.459849	0.606935		
H	<b>0.041517</b>	0.072827	<b>0.013538</b>	<b>0.013795</b>	<b>0.0004</b>	0.127608	<b>0.020791</b>	

Mann-Whitney test shows that there are significant differences among some of the individual groups of data such as Type-1 and Type-3, Type1&2, Type-1&3 and Type-1&2&3 against no-clones group. This might not lead us to any conclusion for clone types relationships with fault-proneness. Another visit to the histogram for fault distribution over the clone types groups will give more information that will help us in reaching a conclusion.



**Figure 10: Ant-Faults Distribution Based on Types**

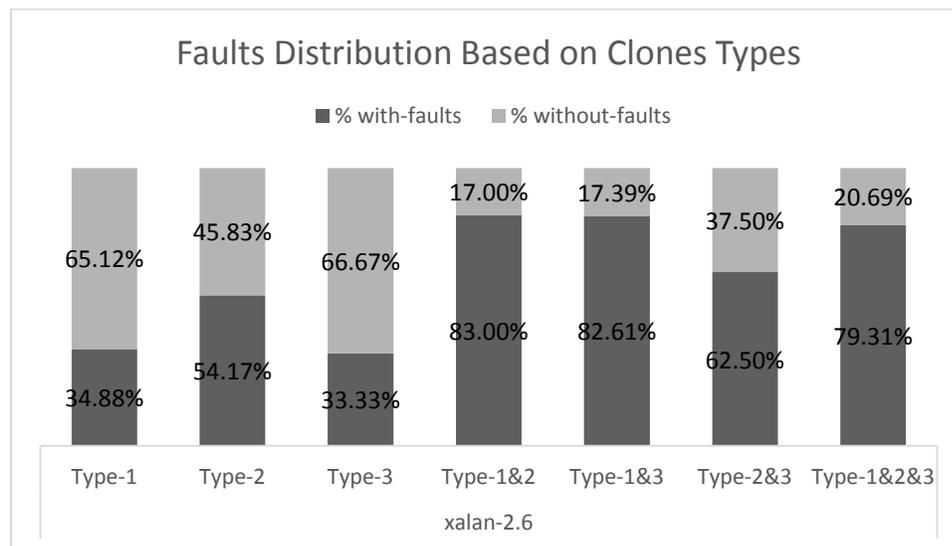
From the p-values above, there are significant differences between clones based on types with respect to fault-proneness. However, the histogram of the system shows that all clone types in the classes are more fault-proneness than classes without clones. The summary in [Table 29] has a generic view of the test results with the context of all subject systems in this study.

Xalan:

**Table 25: Mann-Whitney Test Based on Types**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	<b>0.02032</b>							
C	0.863085	0.059145						
D	<b>0</b>	<b>0.000203</b>	<b>0</b>					
E	<b>0.000021</b>	<b>0.020804</b>	<b>0.000247</b>	0.964277				
F	<b>0.032362</b>	0.563703	0.051673	0.05726	0.162604			
G	<b>0.000013</b>	<b>0.027252</b>	<b>0.000245</b>	0.648912	0.76659	0.227425		
H	0.384058	<b>0.042036</b>	0.495527	<b>0</b>	<b>0.000034</b>	0.059755	<b>0.000019</b>	

Mann-Whitney test shows that there are significant differences among some of the individual groups of data such as Type-1, Type-2, Type-3, Type-1&2 against Type-1&3. Type-2, Type1&2, Type-1&3 and Type-1&2&3 against no-clones group. This might not lead us to any conclusion for clone types relationships with fault-proneness. Another visit to the histogram for fault distribution over the clone types groups will give more information that will help us in reaching a conclusion.



**Figure 11: Xalan-Faults Distribution Based on Types**

From the p-values above, there are significant differences between clones based on types with respect to fault-proneness. However, the histogram of the system shows that all

clone types in the classes except the ones that contain Type-1 & Type-3 are more fault-proneness than classes without clones. The yielded result for Type-3 could be due to the fact that usually the developer gets something to clone and modify, will reassess the new code and probably resolve any faults. The summary in [Table 29] has a generic view of the test results with the context of all subject systems in this study.

Camel:

**Table 26: Mann-Whitney Test Based on Types**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.279758							
C	0.486572	0.08141						
D	0.447336	0.181833	0.628601					
E	0.794861	0.524923	0.441906	0.397887				
F	0.289964	0.726095	0.144992	0.175735	0.435866			
G	0.170904	0.073639	0.23353	0.386477	0.153643	0.066193		
H	0.390033	<b>0.04826</b>	0.983603	0.621091	0.394569	0.125032	0.229494	

Mann-Whitney test shows that there is a significant difference with the individual group of data as Type-2 against no-clones group. This might not lead us to any conclusion for clone types relationships with fault-proneness. Another visit to the histogram for fault distribution over the clone types groups will give more information that will help us in reaching a conclusion.

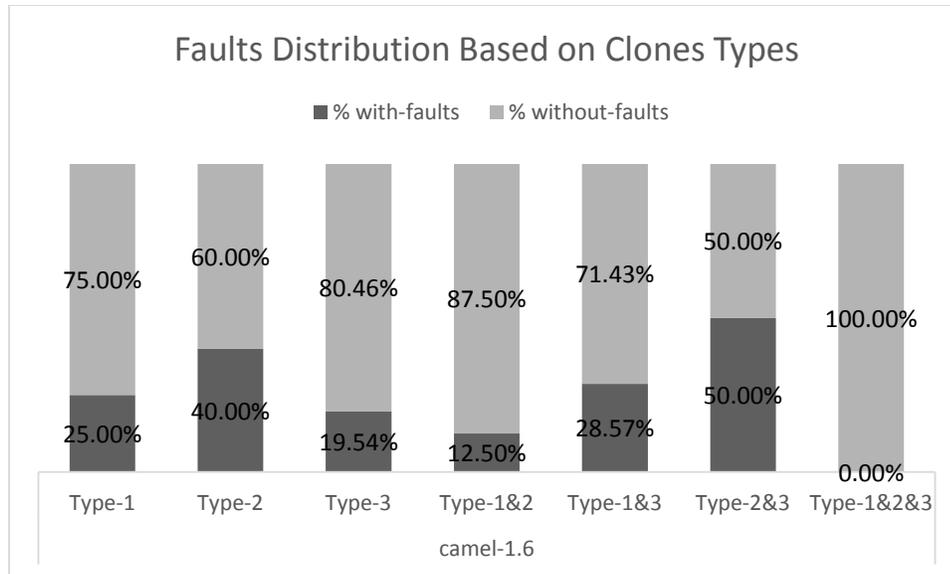


Figure 12: Camel-Faults Distribution Based on Types

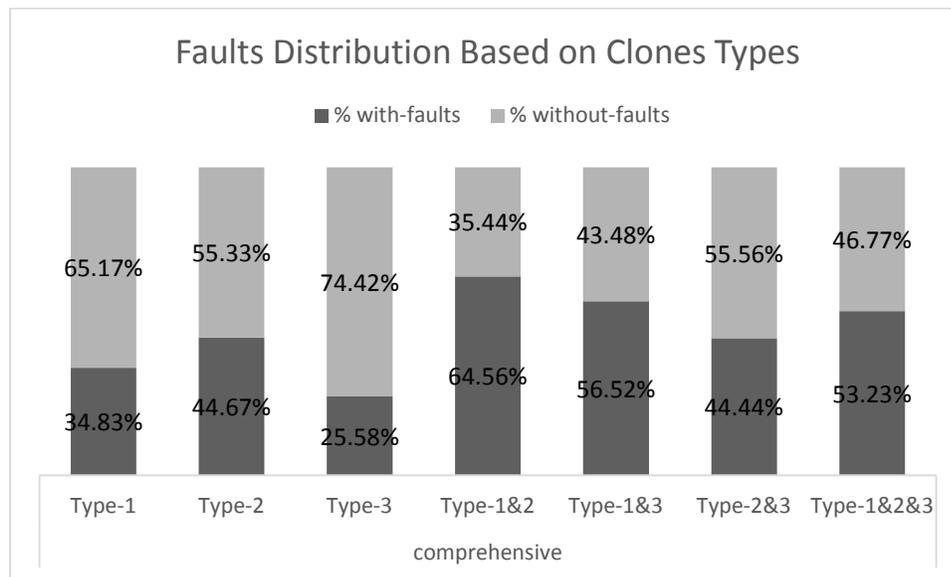
From the p-values above, there are significant differences between clones based on types with respect to fault-proneness. However, the histogram of the system shows that all clone types in the classes except the ones that contain Type-1&2 are more fault-proneness than classes without clones. Exceptional case is the group of classes that have Type-1&2&3 has only 6 classes and none of them is fault-proneness. The summary in [Table 29] has a generic view of the test results with the context of all subject systems in this study.

Total (aggregated) system:

Table 27: Mann-Whitney Test Based on Types

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	<b>0.047735</b>							
C	<b>0.021237</b>	<b>0.000075</b>						
D	<b>0</b>	<b>0.000464</b>	<b>0</b>					
E	<b>0.001025</b>	0.10372	<b>0.000001</b>	0.252056				
F	0.155011	0.9763	<b>0.003229</b>	<b>0.006211</b>	0.167268			
G	<b>0.007362</b>	0.257244	<b>0.000024</b>	0.121303	0.706086	0.328025		
H	<b>0.000545</b>	<b>0</b>	0.822238	<b>0</b>	<b>0</b>	<b>0.000473</b>	<b>0.000001</b>	

Mann-Whitney test shows that there are significant differences among some of the individual groups of data such as Type-1, Type-2, Type-3, Type-1&3 against Type-1&2. Type-1, Type-2, Type-1&2, Type-1&3, Type-2&3 and Type-1&2&3 against no-clones group. This might not lead us to any conclusion for clone types relationships with fault-proneness.



**Figure 13: Comprehensive-Faults Distribution Based on Types**

From the p-values above, there are significant differences between clones based on types with respect to fault-proneness. However, the histogram of the system shows that all clone types in the classes are more fault-proneness than classes without clones. The summary in [Table 29] has a generic view of the test results with the context of all subject systems in this study.

**Table 28: Summary of Mann-Whitney Test for Clones Based on Location**

	Velocity	Synapse	Ant	Xalan	Camel	Total	
A: InterOnly	>	<	<*	<	<*	<*	B: IntraOnly
A: InterOnly	>	>	<*	<*	<	<*	C: Inter&Intra
A: InterOnly	>	>*	>	>*	<	>*	D: Has No Clones
B: IntraOnly	<	>	>	<	>	<	C: Inter&Intra
B: IntraOnly	<	>*	>*	>*	>*	>*	D: Has No Clones
C: Inter&Intra	>	>	>*	>*	>	>*	D: Has No Clones

In the summary table [Table. 28], the data groups and their test results that participated in the Mann-Whitney test for clones based on location, where compared by fault-proneness. So, InterOnly > Has No Clones, means the classes that contain inter-clones are more fault-proneness than the classes with no-clones. If \* is associated with it, such as InterOnly >\* Has No Clones, it means the classes that contain inter-clones are more fault-proneness than the classes with no-clones and this difference is significant.

From the summary table we observed that the classes that have both inter-clones and intra-clones together are more fault-proneness than the classes with no-clones across all the system. In addition, two out of 5 systems have significant differences and consolidated with the results of the aggregated system that contains all the inputs of the subject systems in and treated as a single system. Also, the classes that have intra-clones only, are more likely to be more fault-proneness than the classes with inter-clones only. An exception to this observation was found in velocity system, which could be due to its small size as number of classes. Additionally, we have noticed for the medium-big systems (number of classes: 700+), they are more fault-proneness if the classes contain both inter-clones and intra-clones than classes with inter-clones only. The classes that have intra-clones only are more fault-proneness than classes with no-clones, and this difference in fault-proneness is significant. An exception from the last observation, again the velocity system which is the smallest subject system has opposite result. We believe

this result for velocity system is negligible according to the general trend in addition to the result of the aggregated system which supports our observation.

This concludes that there is a relationship between clone location and fault-proneness. So, we reject the null hypothesis HYP-A3.

**Table 29: Summary of Mann-Whitney Test for Clones Based on Types**

	Velocity	Synapse	Ant	Xalan	Camel	Total	
A:Type-I	>	<	>	<*	<	<*	B:Type-II
A:Type-I	>	>*	<	>	>	>*	C:Type-III
A:Type-I	>	>	<	<	>	<*	D:Type-I&II
A:Type-I	<	>	<	<*	<	<*	E:Type-I&III
A:Type-I	>	>	<	<*	<	<	F:Type-II&III
A:Type-I	<	>	<	<*	>	<*	G:Type-I&II&III
A:Type-I	>	>*	>*	<	>	>*	H:Has No Clones
B:Type-II	<	>*	<	>	>	>*	C:Type-III
B:Type-II	=	>	<	<*	>	<*	D:Type-I&II
B:Type-II	<	>	<	<	>	<	E:Type-I&III
B:Type-II	>	>	<	<	<	>	F:Type-II&III
B:Type-II	<	>	<	<*	>	<	G:Type-I&II&III
B:Type-II	>	>*	>	>*	>*	>*	H:Has No Clones
C:Type-III	>	<	<	<*	>	<*	D:Type-I&II
C:Type-III	<	<	<	<*	<	<*	E:Type-I&III
C:Type-III	>	<*	<	<	<	<*	F:Type-II&III
C:Type-III	<	<	<	<*	>	<*	G:Type-I&II&III
C:Type-III	>	<	>*	<	>	>	H:Has No Clones
D:Type-I&II	<	<	<	>	<	>	E:Type-I&III
D:Type-I&II	>	<	>	>	<	>*	F:Type-II&III
D:Type-I&II	<	>	<	>	>	>	G:Type-I&II&III
D:Type-I&II	>	>	>*	>*	<	>*	H:Has No Clones
E:Type-I&III	>	<	>	>	<	>	F:Type-II&III
E:Type-I&III	=	>	>	>	>	>	G:Type-I&II&III
E:Type-I&III	>	>	>*	>*	>	>*	H:Has No Clones
F:Type-II&III	<	>	<	<	>	<	G:Type-I&II&III
F:Type-II&III	<	>*	>	>	>	>*	H:Has No Clones
G:Type-I&II&III	>	>	>*	>*	<	>*	H:Has No Clones

In the summary table [Table. 29], the data groups and their test results that participated in the Mann-Whitney test for clones based on types, were compared by fault-proneness. So,

Type-1 > Type-2, means the classes that contain Type-1 clones are more fault-proneness than the classes with Type-2 clones. If \* is associated with it, such as Type-1 >\* Type-2 Clones, it means the classes that contain Type-1 clones are more fault-proneness than the classes with Type-2 clones and this difference is significant.

From the summary table we observed that classes that contain Type-2 clones are more fault-proneness than the classes that have no-clones. This observation is true for all the participated subject systems. Additionally, three out of five systems have significant differences, and even the aggregated system is supporting this finding with significant differences. Another observation is, when Type-1&3 represent the clone types in a class, most probably it will be more fault-proneness than if it has all the types Type-1&2&3. Third finding is the classes that have Type-1&3 is more fault-proneness than classes that have no-clones, and two out of the five subject systems have significant differences with respect to fault-proneness, yet the aggregated system consolidates the fact that the difference is significant.

In general, the classes that contain only Type-3 have a tendency to be less fault-proneness. We got this observation from the Mann-Whitney test for the following groups in *Table 29*: (group A: group C), (group C: group E), (group C: group F), and (group C: group G). For all of these groups either all the systems have consistent results or at least four out five in addition to the aggregated system with some differences are being significant.

We observed as well the classes that contain Type-1&3 are more fault-proneness than if the classes have only Type-1 clones. This observation was valid for all subject systems

except for synapse system, which could be due to its small size. We believe that we could overlook this difference since the aggregated system supports this result.

The classes that contain clones of Type-1&2 are more likely to be fault-proneness than the classes that have no-clones, an exception is observed with camel subject system. However, the general trend in addition to the aggregated collected system supports that classes with type-1&2 are more fault-proneness. The result with camel system was not significant difference. We can notice the same behavior of camel system for classes that contain Type-1&2&3 when compared with classes with no-clones of fault-proneness, whereas all the other subject systems show they are more fault-proneness than classes with no-clones.

Similarly, for the classes that have Type-2&3 clones, they are more fault-proneness than classes with no-clones, we observe all systems except velocity support this finding including the aggregated system, which suggests we overlook this exception.

These findings conclude that there are relationships between clone types and fault-proneness. So, we reject the null hypothesis HYP-A2.

Summary of Mann-Whitney Results:

Although velocity system has all null hypothesis accepted, the histogram in [*Figure 1*] shows the general trend across all the systems having the classes that contain clones are more fault-proneness. So, this concludes that there is a relationship between code clones and fault-proneness.

Similarly, from the summary in [*Table 28*] a relationship between clone location and fault-proneness has been proven.

Additionally, from the summary in [Table 29] relationships between clone types and fault-proneness have been proven. Hence, we reach to a conclusion that we reject the null hypotheses of HYP-A1, HYP-A2, and HYP-A3.

## 4.6 Univariate Analysis

In this section, we are trying to answer the question B and its sub-questions (sec.4.1). For the five subject systems, univariate regression models for the fault-proneness variables were created. The goal of univariate analysis is to find whether code clones metrics are good indicators of fault-proneness or not [60]. A summary of the results is shown in tabular format for each subject system. The following approach will be followed during the univariate analysis for all the systems:

- The univariate models will be assessed if they are **statistically significant** for all variables.
- The **Degree and direction of impact** of the independent variables will be measured against fault-proneness.
- The univariate models will be checked for **goodness-of-fit** using three measures, namely -2 Log Likelihood (-2LL), Cox & Snell R Square (CSRS), and Nagelkerke R Square (NR2).
- The **classification performance** of these univariate models will be evaluated using Correctly Classified Rate and Receiver Operating Characteristic (ROC) Area, according to [61], we consider the following values for evaluation: ROC = 0.5 shows the classification is “not good”, “poor” for the values  $0.5 < \text{ROC} < 0.6$ , “fair” for the values  $0.6 \leq \text{ROC} < 0.7$ , “acceptable” if  $0.7 \leq \text{ROC} < 0.8$ .

For velocity system, as shown below:

**Table 30: Univariate Analysis for Velocity system**

velocity	C <sub>0</sub> (intercept)	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate %	ROC Area
TCF	-0.737	0.143	291.909	0.008	0.011	0.029	65.939	0.519
CD	-0.652	-0.003	293.760	0.000	0.000	0.079	65.939	0.427
CRFL	-0.762	0.308	286.641	0.031	0.042	0.004	67.249	0.541
NBR	-0.787	0.473	288.079	0.025	0.034	0.000	65.502	0.551
RSA	-0.735	0.873	291.729	0.009	0.012	0.001	64.192	0.548
RSI	-0.667	0.143	293.756	0.000	0.000	0.971	65.939	0.452
CVR	-0.741	0.639	292.357	0.006	0.009	0.019	66.376	0.524
RNR	0.430	-1.186	292.865	0.004	0.006	0.924	65.939	0.491
Inter-clones	-0.823	0.659	286.054	0.033	0.046	0.000	66.812	0.545
Intra-clones	-0.665	0.016	293.760	0.000	0.000	0.284	65.939	0.459
Type-1	-0.779	0.524	288.724	0.022	0.030	0.002	66.376	0.526
Type-2	-0.664	0.023	293.770	0.000	0.000	0.859	65.502	0.438
Type-3	-0.681	0.127	293.383	0.002	0.002	0.410	65.502	0.473

**Independent variables assessment for statistically significance:**

Based on the p-values, TCF, CRFL, NBR, RSA, CVR, Inter-class, and Type-1 showed significant indicators of fault-proneness. Whereas CD, RSI, RNR, Intra, Type-2 and Type-3 are none significant fault-proneness indicators. NBR, RSA, CVR inter-class, and Type-1 are together as they are from the same leading principal component PC-1. Based on these results, the following can be observed:

**Degree and direction of impact of the independent variables to fault-proneness:**

From the table the coefficients C1 shows the direction of impact, the independent variables that are based on clone types and location have positive impact, but for the remaining clone metrics have different impact. All variables have positive impact except CD and RNR have negative impact, which means the class that has higher CD or RNR will be less fault-prone.

**Goodness-of-Fit:**

The following measures are representing the goodness-of-fit of the regression models:

-2 Log Likelihood (-2LL): it measure how bad is the prediction of dependent variables out of independent variables where a high value means the poorer is the model.

Cox & Snell R Square (CSRS): it is one type of "pseudo-R" statistic which measures the goodness-of-fit of the model, a higher value means better goodness-of-fit of the model.

Nagelkerke R Square (NR2): it is also a "pseudo-R" value where a high value means that the model fits well.

For all the independent variables, they have relatively high (-2LL) values and low values of CSRS and NR2 which represent weak goodness-of-fits

**Classification Performance:**

The univariate analysis results for velocity system show that six out of thirteen metrics scored “poor” as a classification performance, and the remaining seven metrics scored “not good” for the classification. As a result, all of these univariate models are not concrete classifiers for fault-proneness.

Synapse system:

**Table 31: Univariate Analysis for Synapse system**

synapse	C <sub>0</sub>	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate %	ROC Area
TCF	-0.783	0.067	325.531	0.005	0.007	0.000	65.625	0.583
CD	-0.639	-0.006	326.539	0.001	0.001	0.043	66.406	0.449
CRFL	-0.738	0.079	326.376	0.002	0.002	0.001	66.406	0.555
NBR	-0.986	0.060	312.757	0.053	0.074	0.000	67.969	0.605
RSA	-0.692	0.078	326.795	0.000	0.000	0.016	66.406	0.434
RSI	-0.724	0.499	326.396	0.002	0.002	0.040	66.406	0.494
CVR	-0.780	0.471	325.971	0.003	0.005	0.002	66.406	0.553
RNR	-0.245	-0.477	326.721	0.000	0.000	0.161	66.016	0.449
Inter-clones	-0.747	0.065	326.241	0.002	0.003	0.003	66.406	0.540
Intra-clones	-0.723	0.080	325.976	0.003	0.005	0.172	64.844	0.481
Type-1	-0.707	0.073	326.533	0.001	0.002	0.002	66.406	0.541
Type-2	-0.818	0.252	320.946	0.023	0.031	0.000	66.406	0.576
Type-3	-0.622	-0.099	326.025	0.003	0.004	0.555	66.406	0.456

**Independent variables assessment for statistically significance:**

Based on the p-values, TCF, CD, RSI, CRFL, NBR, RSA, CVR, Inter-class, Type-1, Type-2 and Type-3 showed significant indicators of fault-proneness. Whereas RNR and Intra are none significant fault-proneness indicators.

**Degree and direction of impact of the independent variables to fault-proneness:**

From the table the coefficients C1 shows the direction of impact, the independent variables that are based on clone types and location except Type-3 have positive impact, but for the remaining clone metrics have different impact. All variables have positive impact except CD, RNR and Type-3 have negative impact, which means the class that has higher CD, RNR, and/or Type-3 will be less fault-prone.

**Goodness-of-Fit:**

In general (NR2) are relatively low like the case in velocity, but different behavior for the same variable between the two systems, and the same thing applies to (-2 LL) and CSRS values. For all the independent variables, they represent weak goodness-of-fit.

**Classification Performance:**

The univariate analysis results for synapse system show that only one independent variable which is NBR has scored “fair” as classifier and six out of thirteen metrics scored “poor” as a classification performance, and the remaining six metrics scored “not good” for the classification. As a result, all of these univariate models are not concrete classifiers for fault-proneness.

Ant system:

**Table 32: Univariate Analysis for Ant system**

ant	C <sub>0</sub>	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate	ROC Area
TCF	-1.517	0.191	753.376	0.046	0.070	0.000	0.783	0.612
CD	-1.168	-0.019	785.786	0.003	0.005	0.002	77.598	0.442
CRFL	-1.215	-0.078	787.431	0.001	0.002	0.000	77.598	0.442
NBR	-1.304	0.053	785.260	0.004	0.006	0.001	77.598	0.536
RSA	-1.134	-1.236	781.313	0.009	0.014	0.266	77.598	0.465
RSI	-1.325	1.817	783.233	0.007	0.011	0.000	77.598	0.584
CVR	-1.184	-0.453	787.067	0.002	0.003	0.001	77.598	0.435
RNR	-1.448	0.222	788.320	0.000	0.000	0.707	77.598	0.465
Inter-clones	-1.398	0.214	776.134	0.016	0.025	0.000	77.463	0.535
Intra-clones	-1.408	0.234	758.556	0.039	0.060	0.000	78.273	0.581
Type-1	-1.432	0.379	758.608	0.039	0.060	0.000	78.138	0.546
Type-2	-1.344	0.220	778.286	0.013	0.021	0.000	77.328	0.522
Type-3	-1.381	0.285	771.357	0.023	0.035	0.000	77.733	0.545

**Independent variables assessment for statistically significance:**

Based on the p-values, all variables except RSA and RNR, showed significant indicators of fault-proneness, whereas RSA and RNR showed none significant fault-proneness indicators. These results are very similar to synapse where RNR is common between the two systems as non-significant fault-proneness indicator.

**Degree and direction of impact of the independent variables to fault-proneness:**

From the table the coefficients C1 shows the direction of impact, the independent variables that are based on clone types and location have positive impact, but for the remaining clone metrics have different impact. All independent variables have positive impact except four namely: CD, CRFL, RSA and CVR have negative impact.

**Goodness-of-Fit:**

In general (NR2) are relatively low like the case in velocity and synapse, but different behavior for the same variable between the two systems, and the same thing applies to -2 LL and CSRS values. In this system, clone types and location metrics scored a little bit higher scores in general when comparing them with same metrics from velocity and

synapse systems. In general, all the independent variables represent weak goodness-of-fits.

**Classification Performance:**

The univariate analysis results for ant system show that only one independent variable which is TCF has scored “fair” as classifier and seven out of thirteen metrics scored “poor” as a classification performance, and the remaining five metrics scored “not good” for the classification. As a result, all of these univariate models are not concrete classifiers for fault-proneness.

Xalan system:

**Table 33: Univariate Analysis for Xalan system**

xalan	C <sub>0</sub>	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate	ROC Area
TCF	0.328	-0.145	1175.335	0.039	0.052	0.000	60.457	0.617
CD	0.101	0.005	1209.442	0.000	0.001	0.000	0.530	0.465
CRFL	0.253	-0.037	1166.600	0.048	0.064	0.000	53.600	0.530
NBR	0.300	-0.046	1161.495	0.054	0.072	0.000	53.943	0.573
RSA	0.143	-0.112	1209.477	0.000	0.000	0.000	51.657	0.474
RSI	0.161	-0.969	1207.001	0.003	0.004	0.000	53.714	0.540
CVR	0.171	-0.223	1208.532	0.001	0.002	0.000	50.971	0.513
RNR	1.974	-1.998	1204.968	0.006	0.007	0.000	51.200	0.560
Inter-clones	0.361	-0.333	1172.167	0.042	0.056	0.000	62.743	0.593
Intra-clones	0.206	-0.119	1194.843	0.017	0.023	0.000	57.371	0.546
Type-1	0.329	-0.350	1168.788	0.046	0.061	0.000	59.771	0.584
Type-2	0.267	-0.295	1178.292	0.035	0.047	0.000	61.943	0.598
Type-3	0.125	-0.011	1209.717	0.000	0.000	0.000	52.800	0.485

**Independent variables assessment for statistically significance:**

All independent variables showed significant indicators of fault-proneness. Xalan has 57% of its classes that contain clone fragments are fault classes which was the highest value among other systems, that could be related to the result of having all variables are significant fault-proneness indicators.

**Degree and direction of impact of the independent variables to fault-proneness:**

From the table the coefficients C1 shows the direction of impact, all independent variables have negative impact, except for CD has positive impact. This is surprising results as xalan behavior is the opposite of other systems with respect to direction of impact.

### **Goodness-of-Fit:**

In general (NR2) are relatively low like the case in velocity, synapse and ant, but different behavior for the same variable between the two systems, and the same thing applies to CSRS values with higher scores for xalan. On the other hand, (-2 LL) scored relatively higher values and this could be due to the concentration of faults in classes that have clone fragments. In this system, clone types and location metrics scored a little bit higher scores in general in addition to CRFL and NBR when comparing them with same metrics from velocity, synapse and ant systems. In general, all the independent variables have relatively low values which represent low goodness-of-fits.

### **Classification Performance:**

The univariate analysis results for xalan system show that only one independent variable which is TCF has scored “fair” as classifier and nine out of thirteen metrics scored “poor” as a classification performance, and the remaining three metrics scored “not good” for the classification. As a result, all of these univariate models are not concrete classifiers for fault-proneness.

Camel system:

**Table 34: Univariate Analysis for Camel system**

camel	C <sub>0</sub>	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate %	ROC Area
TCF	1.458	-0.164	930.995	0.008	0.013	0.046	79.893	0.507
CD	1.425	-0.016	935.512	0.003	0.005	0.129	80.000	0.501
CRFL	1.415	-0.203	936.218	0.002	0.004	0.046	79.786	0.512
NBR	1.346	0.152	936.951	0.002	0.003	0.085	79.893	0.503
RSA	1.338	0.816	935.744	0.003	0.005	0.074	79.893	0.504
RSI	1.478	-3.237	918.381	0.021	0.034	0.000	80.214	0.539
CVR	1.400	-0.249	938.097	0.000	0.001	0.022	79.893	0.507
RNR	1.050	0.359	938.381	0.000	0.000	0.878	79.893	0.481
Inter-clones	1.400	-0.077	937.661	0.001	0.001	0.290	79.893	0.469
Intra-clones	1.439	-0.282	928.644	0.011	0.017	0.000	79.893	0.514
Type-1	1.380	-0.004	938.518	0.000	0.000	0.535	79.893	0.462
Type-2	1.409	-0.323	933.973	0.005	0.008	0.105	79.893	0.500
Type-3	1.444	-0.224	931.367	0.008	0.012	0.574	80.107	0.493

**Independent variables assessment for statistically significance:**

Based on the p-values, TCF, CRFL, RSI, CVR, and Intra showed significant indicators of fault-proneness. Whereas, CD, NBR, Inter-class, RSA, RNR, Type-1, Type-2 and Type-3, are none significant indicators for fault-proneness.

**Degree and direction of impact of the independent variables to fault-proneness:**

From the table the coefficients C1 shows the direction of impact, the independent variables that are based on clone types and location have positive impact, but for the remaining clone metrics have different impact. All variables have negative impact except NBR, RSA and RNR have positive impact. Again, there is no consistent with other subject systems of this study.

**Goodness-of-Fit:**

The following measures are representing the goodness-of-fit of the regression models: -2 Log likelihood (-2 LL), Cox & Snell R Square (CSRS), and Nagelkerke R Square (NR2). For all the independent variables, they represent weak goodness-of-fit.

**Classification Performance:**

The univariate analysis results for camel system show that eight out of thirteen metrics scored “poor” as a classification performance, and the remaining five metrics scored “not good” for the classification. As a result, all of these univariate models are not concrete classifiers for fault-proneness.

Comprehensive system (aggregated subject systems):

**Table 35: Univariate Analysis for Comprehensive system**

total	C <sub>0</sub>	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate %	ROC Area
TCF	-1.009	0.172	3633.675	0.034	0.048	0.000	70.356	0.605
CD	-0.823	0.001	3739.443	0.000	0.000	0.000	69.401	0.501
CRFL	-0.897	0.059	3651.408	0.029	0.040	0.000	70.982	0.580
NBR	-0.948	0.062	3623.006	0.038	0.053	0.000	70.652	0.565
RSA	-0.851	0.273	3736.158	0.001	0.002	0.000	69.401	0.541
RSI	-0.883	1.521	3716.898	0.007	0.010	0.000	69.236	0.550
CVR	-0.901	0.534	3725.207	0.005	0.007	0.000	69.401	0.575
RNR	-1.267	0.485	3738.529	0.000	0.000	0.004	69.401	0.527
Inter-clones	-0.967	0.248	3672.732	0.022	0.031	0.000	69.532	0.567
Intra-clones	-0.911	0.180	3675.584	0.021	0.029	0.000	70.356	0.548
Type-1	-0.967	0.378	3635.863	0.034	0.047	0.000	70.191	0.572
Type-2	-0.927	0.302	3664.993	0.024	0.034	0.000	69.961	0.572
Type-3	-0.850	0.092	3731.284	0.003	0.004	0.000	69.401	0.516

**Independent variables assessment for statistically significance:**

All independent variables showed significant indicators of fault-proneness.

**Degree and direction of impact of the independent variables to fault-proneness:**

From the table the coefficients C1 shows the direction of impact, all variables have positive impact, which means that any class that has high value of any of clone metrics has also more fault-proneness.

**Goodness-of-Fit:**

The following measures are representing the goodness-of-fit of the regression models: for (-2 Log likelihood), comprehensive scored highly when it is compared to other systems

(3x results of camel, 10x results of velocity), but for Cox & Snell R Square which tells that how bad is the model fits, and Nagelkerke R Square (NR2) it didn't perform much differently comparing to individual systems. For all the independent variables, they have relatively low values which represent weak goodness-of-fit.

### **Classification Performance:**

The univariate analysis results for comprehensive system show that 12 out of 13 metrics scored "poor" as a classification performance, and the excluded metric scored "fair" for the classification. As a result, all of these univariate models are not concrete classifiers for fault-proneness.

### **Summary of Univariate Analysis:**

From the results of the analysis the following findings are concluded to answer the thirteen hypotheses (HYP-B1, HYP-B2, HYP-B3, HYP-B4, HYP-B5, HYP-B6, HYP-B7, HYP-B8, HYP-B9, HYP-B10, HYP-B11, HYP-B12, and HYP-B13):

Decision criteria:

- 1) If a metric scored  $p\text{-value} < 0.05$  in across all the systems, then it is considered a significant indicator of fault-proneness. Hence, there is a clear evidence to reject the null hypothesis.
  - 2) If a metric scored  $p\text{-value} \geq 0.05$  in across all the systems, then it is considered non-significant indicator of fault-proneness. Hence, there is a clear evidence to accept the null hypothesis.
  - 3) If a metric across systems has inconsistent results, we don't have clear evidence to accept or to reject a hypothesis.
- TCF metric is a fault-proneness significant indicator, so HYP-B1 is **rejected**.

- CD, there is no clear evidence to accept the hypothesis HYP-B2.
- CRFL is a fault-proneness significant indicator, so HYP-B3 is **rejected**.
- NBR, there is no clear evidence to accept the hypothesis HYP-B4.
- RSA, there is no clear evidence to accept the hypothesis HYP-B5.
- RSI, there is no clear evidence to accept the hypothesis HYP-B6.
- CVR is a fault-proneness significant indicator, so HYP-B7 is **rejected**.
- RNR, there is no clear evidence to accept the hypothesis HYP-B8.
- Interclass, there is no clear evidence to accept the hypothesis HYP-B9.
- Intra, there is no clear evidence to accept the hypothesis HYP-B10.
- Type-1, there is no clear evidence to accept the hypothesis HYP-B11.
- Type-2, there is no clear evidence to accept the hypothesis HYP-B12.
- Type-3, there is no clear evidence to accept the hypothesis HYP-B13.

#### **4.7 Multivariate Analysis**

In this section, we are trying to answer the questions C, D, and E in 4.1. For the five subject systems, univariate regression models for the fault-proneness variables were created [62]. This section shows the study of the subject systems with multivariate perspective, where WEKA software is used to generate features selection. The attribute evaluator used is CfsSubsetEval and the search method used is BestFirst with bi-directional setting. After getting these attributes that mostly representative of the data in a subject system, they are incorporated in multivariate analysis to study their features using weka software with default values.

For velocity system:

**Table 36: Multivariate Analysis for Velocity system**

	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate	ROC Area
TCF	0.143	291.909	0.008	0.011	0.029	66%	0.519
Intercept	-0.737						

**Independent variables assessment for statistically significance:**

The features selection yielded only one attribute to be selected which is TCF which is a member of PC-2, this means that the multivariate model will not reach better results than univariate model for TCF. The selected attribute TCF was not even the best independent variable as fault-proneness indicator with regards to the correctly classified rate.

**Degree and direction of impact of the independent variables to fault-proneness:**

TCF is still affecting fault-proneness positively as it was in univariate analysis.

**Goodness-of-Fit:**

Similar behavior to (-2 LL), CSRS and NR2 where other variables have higher scores which reflect even weaker representation of goodness-of-fit.

**Classification Performance:**

This multivariate model has 0.519 as ROC area value which is less than 6 other variables in univariate models which tells how poor is the classification performance of the multivariate model.

For synapse system:

**Table 37: Multivariate Analysis for Synapse system**

	C1	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate %	ROC Area
NBR	0.060	312.757	0.053	0.074	0.000	68%	0.605
Intercept	-0.986						

**Independent variables assessment for statistically significance:**

The features selection yielded only one attribute to be selected which is NBR that belongs to PC-1, this means that the multivariate model will not reach better results than univariate scores for NBR. The selected attribute NBR was the best independent variable as fault-proneness indicator with regards to the correctly classified rate.

**Degree and direction of impact of the independent variables to fault-proneness:**

NBR is still affecting fault-proneness positively as it was in univariate model.

**Goodness-of-Fit:**

Except (-2 LL) score for NBR which was not the highest score among others, the multivariate model has similar behavior to univariate CSRS and NR2 where other variables have smaller scores which reflects even weaker representation of goodness-of-fit.

**Classification Performance:**

This multivariate model has 0.605 as ROC area value which is higher than all other variables in univariate models which tells that same classification performance is maintained.

For ant system:

**Table 38: Multivariate Analysis for Ant system**

	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate	ROC Area
CD	-0.094	734.670	0.070	0.107	0.002	79%	0.577
RSI	2.535				0.000		
RNR	-0.451				0.707		
Type-I	0.574				0.000		
Intercept	-0.873						

**Independent variables assessment for statistically significance:**

The features selection yielded four attributes to be selected which are: CD, RSI, RNR, and Type-1, they represent all PC groups in PCA. The multivariate model did not reach better results than univariate scores, because there were three variables yielded better results for NBR. The selected attribute NBR was the best independent variable as fault-proneness indicator with regards to the correctly classified rate.

**Degree and direction of impact of the independent variables to fault-proneness:**

RNR still positively affects fault-proneness, whereas RSA still negatively affects fault-proneness.

**Goodness-of-Fit:**

Except (-2 LL) score for NBR which was not the highest score among others, the multivariate model has similar behavior to univariate CSRS and NR2 where other variables have smaller scores which reflects even weaker representation of goodness-of-fit.

**Classification Performance:**

This multivariate model has 0.605 as ROC area value which is higher than all other variables in univariate models which tells that same classification performance is maintained.

For xalan system:

**Table 39: Multivariate Analysis for Xalan system**

	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate %	ROC Area
NBR	-0.117	1057.561 <sup>a</sup>	0.160	0.213	0.000	67%	0.726
RSA	3.470				0.000		
CVR	-0.311				0.000		
RNR	-2.263				0.000		
Inter-class	-0.328				0.000		
Type-II	-0.126				0.000		
Intercept	2.408						

**Independent variables assessment for statistically significance:**

The features selection yielded six attributes to be selected which are: NBR, RSA, CVR, RNR, Inter-class, and Type-2, again these metrics belong to all types of PC groups in PCA. The multivariate model produced better results for correctly classified rate than all univariate variables scores. The selected attributes are also statistically significant fault-proneness indicators.

**Degree and direction of impact of the independent variables to fault-proneness:**

RSA still positively affects fault-proneness, whereas NBR, CVR, RNR, Inter-class and Type-2 still negatively affect fault-proneness.

**Goodness-of-Fit:**

The multivariate model has relatively better behavior when compare to univariate variables, especially Cox & Snell R Square and NR2 where the multivariate model has significant improvement over univariate models, NR2 value (0.213) which indicates much better representation of goodness-of-fit.

**Classification Performance:**

This multivariate model has 0.726 as ROC area value which is acceptable and higher than any ROC area values in univariate models, which tells that classification performance is much better for the multivariate model.

For camel system:

**Table 40: Multivariate Analysis for Camel system**

	C <sub>1</sub>	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate %	ROC Area
RSI	-3.062	918.222	0.021	0.034	0.000	80%	0.522
Type-III	-0.040				0.574		
Intercept	1.484						

**Independent variables assessment for statistically significance:**

The features selection yielded two attributes to be selected which are: RSI, and Type-3, both of the metrics belong to PC-2. The multivariate model produced no better than the highest correctly classified rate for any univariate models. Only RSI from the selected features is statistically significant fault-proneness indicator.

**Degree and direction of impact of the independent variables to fault-proneness:**

RSI and Type-3 still negatively affect fault-proneness as they were in univariate models.

**Goodness-of-Fit:**

The multivariate model has maintained almost the same behavior as the best of univariate models, this applies to Cox & Snell R Square and NR2 where the multivariate model has slight improvement over univariate models, NR2 value (0.034) which indicates weak goodness-of-fit.

**Classification Performance:**

This multivariate model has (0.522) as ROC area value which indicates poor classification performance. This value was slightly less than the best ROC value of the univariate models.

For comprehensive collection:

**Table 41: Multivariate Analysis for Comprehensive system**

	$C_1$	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square	p-value	Correctly Classified Rate %	ROC Area
TCF	0.169	3633.108	0.034	0.049	0.000	70%	0.609
RSA	-0.293				0.000		
CVR	0.236				0.000		
Intercept	-1.006						

**Independent variables assessment for statistically significance:**

The features selection yielded three attributes to be selected which are: TCF, RSA, and CVR, the multivariate model produced almost the same results for correctly classified rate as the best value of univariate models. The selected attributes are statistically significant fault-proneness indicators.

**Degree and direction of impact of the independent variables to fault-proneness:**

RSA still negatively affects fault-proneness, whereas TCF, and CVR still positively affect fault-proneness.

**Goodness-of-Fit:**

The multivariate model has relatively better behavior when compare to univariate variables, especially Cox & Snell R Square and NR2 where the multivariate model has significant improvement over univariate models, NR2 value (0.049) which indicates slightly better representation of goodness-of-fit.

**Classification Performance:**

This multivariate model has 0.609 as ROC area value which is fair and higher than any ROC area values in univariate models, which tells that classification performance is better for the multivariate model.

#### 4.7.1 Validation of Fault-Proneness Prediction Model

A cross-validation has been used to minimize the chance of getting a very good-fit of the explanatory model with high accuracy, as this is considered to be a threat to validity of the model. Hence, all the five subject systems are combined in one data set, and each time one system will be left out to build the model and tested against the system that was left out. The approach of using leave-one-out in building the model will help in generalizing the results and moving away from being biased to the small data set.

**Table 42: Correct classification rate for training data and Cross-Validation Data**

	Correctly Classified Rate	
	Training Data	Cross-V
velocity	65.94%	<b>73.03%</b>
synapse	67.97%	<b>72.88%</b>
ant	<b>79.08%</b>	70.76%
xalan	66.63%	<b>76.08%</b>
camel	<b>80.00%</b>	70%
Avg.	71.92%	<b>72.52%</b>

Keys: Numbers in **Bold** indicate the best value among others.

**Training Data:** model is built over data training being part of the same data set.

**Cross-V:** model is built over data set based on cross-validation leave-one-out approach

**Table 43: ROC Area for training data and Cross-Validation Data**

	ROC Area	
	Training Data	Cross-V
velocity	0.519	<b>0.644</b>
synapse	0.605	<b>0.614</b>
ant	0.577	<b>0.621</b>
xalan	<b>0.726</b>	0.584
camel	0.522	<b>0.649</b>
Avg.	0.5898	<b>0.6224</b>

*Keys:* Numbers in **Bold** indicate the best value among others.

**Training Data:** model is built over data training being part of the same data set.

**Cross-V:** model is built over data set based on cross-validation leave-one-out approach

The above two tables shows the comparisons between the results of the multivariate models when the training data used within the same system and when cross validation leave-one-out is used. In both accuracy measures, namely correctly classified rate and ROC area the all average were for the advantage of the models that were built using the cross-validation. These results give higher confident in the regression model evaluation in being more realistic by using cross-validation.

#### **4.7.2 Comparisons of Code Clones Metrics and C&K Metrics in Predicting Fault-Proneness**

The comparisons between code clone metrics suite, C&K suite, and both combined in a single suite. All suites are analyzed from different perspectives, namely goodness-of-fit, correct classified rate, and ROC area. In general, the results are showing that the combined suite of clone metrics and C&K has better scores in all three perspectives than either clone metrics or the C&K metrics individually. Starting with goodness-of-fit, the following table shows the scores for all the three suites against the five subject systems:

**Table 44: Multivariate Analysis - (Goodness-of-fit)**

	-2 Log likelihood			Cox & Snell R Square			Nagelkerke R Square		
	Both	CLN	C&K	Both	CLN	C&K	Both	CLN	C&K
velocity	<b>3051.544</b>	3179.741	3167.081	<b>0.130</b>	0.090	0.093	<b>0.184</b>	0.127	0.132
synapse	<b>3047.683</b>	3174.547	3154.646	<b>0.122</b>	0.082	0.088	<b>0.173</b>	0.116	0.124
Ant	<b>2608.390</b>	2719.549	2727.127	<b>0.126</b>	0.083	0.079	<b>0.175</b>	0.116	0.110
Xalan	<b>2120.714</b>	2325.392	2122.386	<b>0.112</b>	0.025	0.111	<b>0.168</b>	0.038	0.167
camel	<b>2392.909</b>	2512.682	2487.452	<b>0.147</b>	0.097	0.108	<b>0.203</b>	0.134	0.149
Avg.	<b>2644.248</b>	2782.382	2731.738	<b>0.127</b>	0.075	0.096	<b>0.180</b>	0.106	0.136

*Keys:* Numbers in **Bold** indicate the best value among others.

**Both:** model is built over some selected attributes of both the clone and C&K metrics

**CLN:** model is built over some selected attributes of the clone metrics only.

**C&K:** model is built over some selected attributes of the C&K metrics only.

It is clear that the combined suite is giving better prediction models than the individual suites. This applies to -2 Log Likelihood, Cox & Snell R Square and Nagelkerke R Square values. Similarly, C&K models are performing better than clone models.

For the corrected classified rate which is used as accuracy measure, the generated models had some variations in the behavior, but in general, the overall average was for the sake of clone metrics. The clone metrics had better correctly rate than other models in 3 subject systems in addition to the average of all systems. It was found that for one system out of five, the combined suite of metrics was performing better than other models. On the other hand, the clone suite is performing better than the rest for three systems, whereas two systems the C&K suite was performing better than the remaining suites. More details are in the following table in addition to the histogram:

**Table 45: Multivariate Analysis - (Correctly Classified Rate)**

	Correctly Classified Rate		
	Both	CLN	C&K
velocity	66.81%	<b>73.03%</b>	66.81%
synapse	71.09%	<b>72.88%</b>	70.70%
ant	78.95%	70.76%	<b>81.11%</b>
xalan	59.31%	<b>76.08%</b>	59.66%
camel	<b>78%</b>	70%	<b>78%</b>
Avg.	70.87%	<b>72.52%</b>	71.33%

Keys: Numbers in **Bold** indicate the best value among others.

**Both:** model is built over some selected attributes of both the clone and C&K metrics

**CLN:** model is built over some selected attributes of the clone metrics only.

**C&K:** model is built over some selected attributes of the C&K metrics only.



**Figure 14: Histogram of Correctly Classified Rate in Multivariate Analysis**

The other perspective is the ROC area, the observation for ROC area regarding the inconsistent behavior. This is the case with the combined suite, it gives the best model on average among the rest of suites. However, the clone metrics suite was the best for the

camel system, whereas C&K metrics suite was the best for two subject systems. Further details are in the following table:

**Table 46: Multivariate Analysis - (ROC Area)**

	ROC Area		
	Both	CLN	C&K
velocity	0.684	0.644	<b>0.708</b>
synapse	0.751	0.614	<b>0.762</b>
Ant	0.787	0.621	<b>0.798</b>
Xalan	<b>0.690</b>	0.584	0.605
camel	0.631	<b>0.649</b>	0.636
Avg.	<b>0.709</b>	0.622	0.702

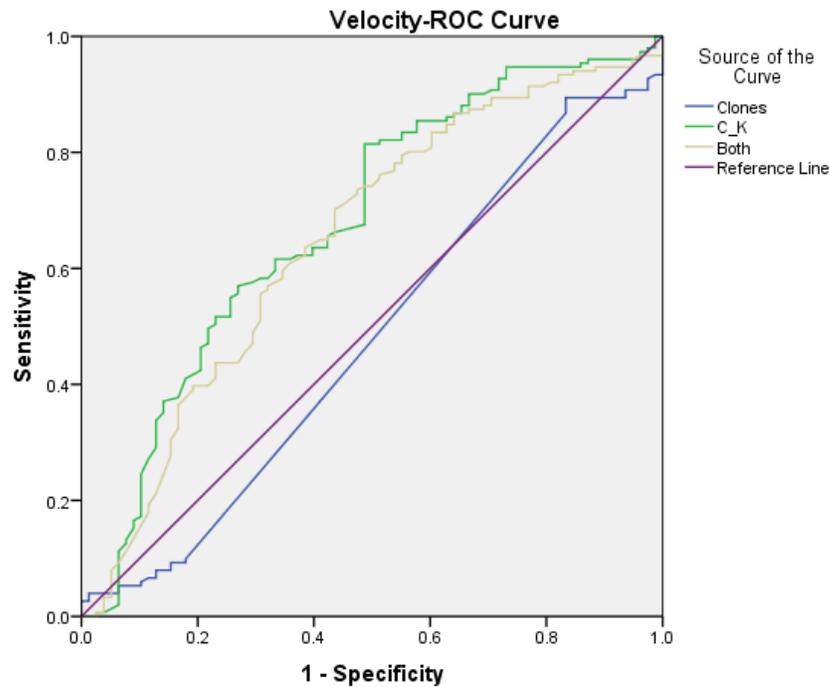
*Keys:* Numbers in **Bold** indicate the best value among others.

**Both:** model is built over some selected attributes of both the clone and C&K metrics

**CLN:** model is built over some selected attributes of the clone metrics only.

**C&K:** model is built over some selected attributes of the C&K metrics only.

The ROC curves are showing the performance of each suite in comparison with respect to others as listed below:



Diagonal segments are produced by ties.

**Figure 15: Multivariate Analysis - ROC Curve for Velocity**

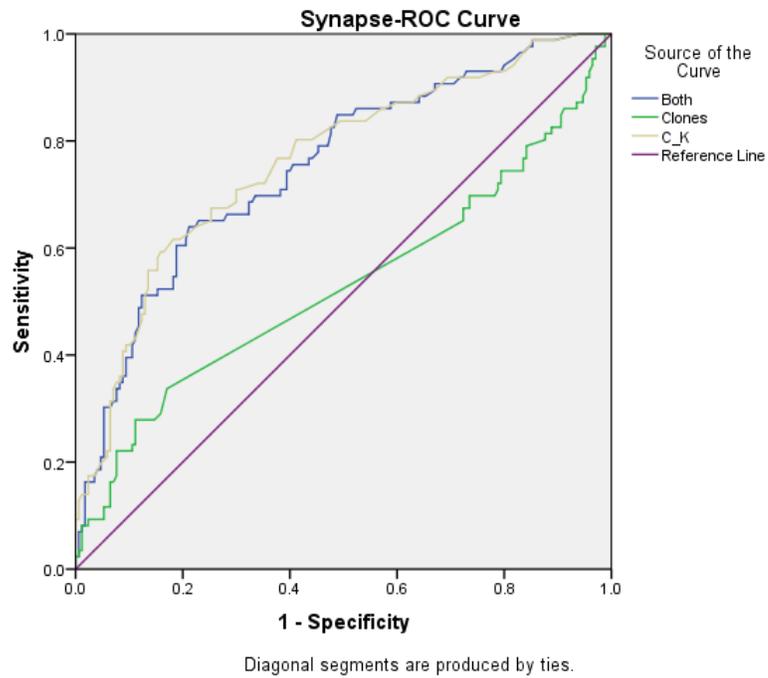
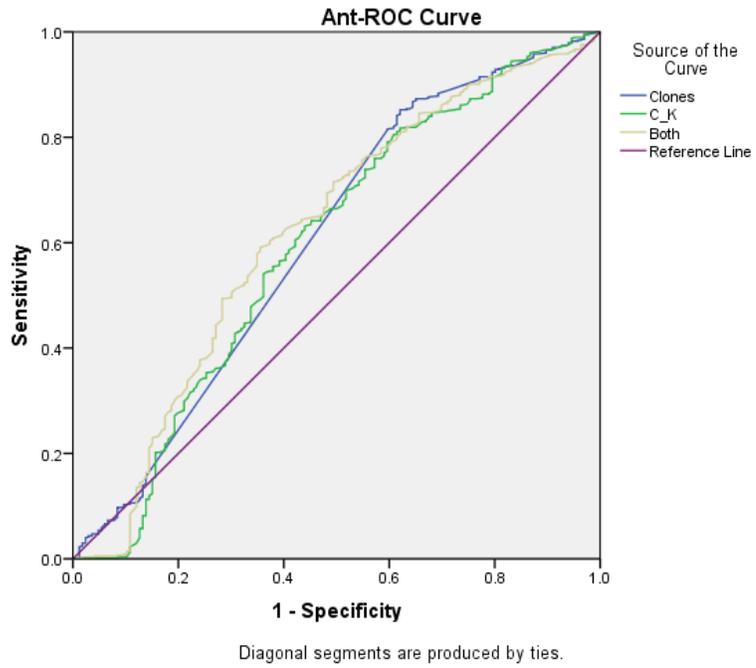
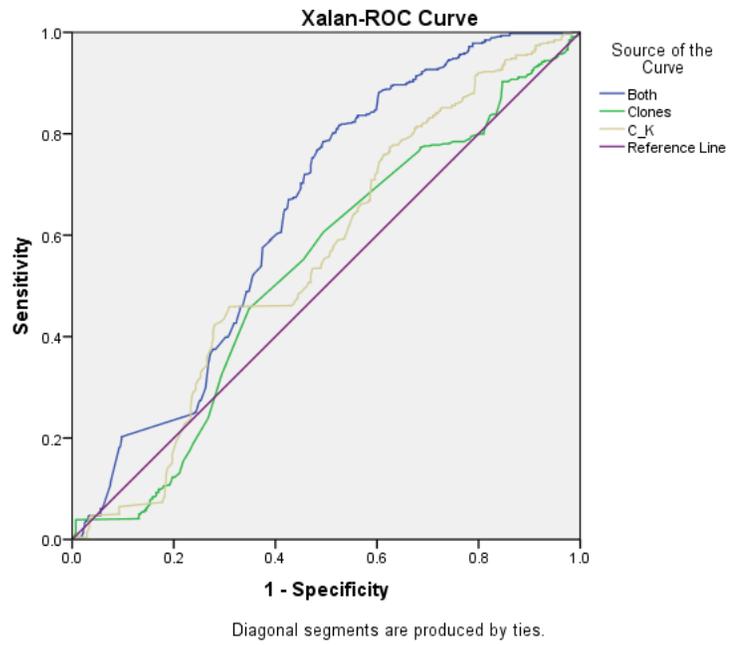


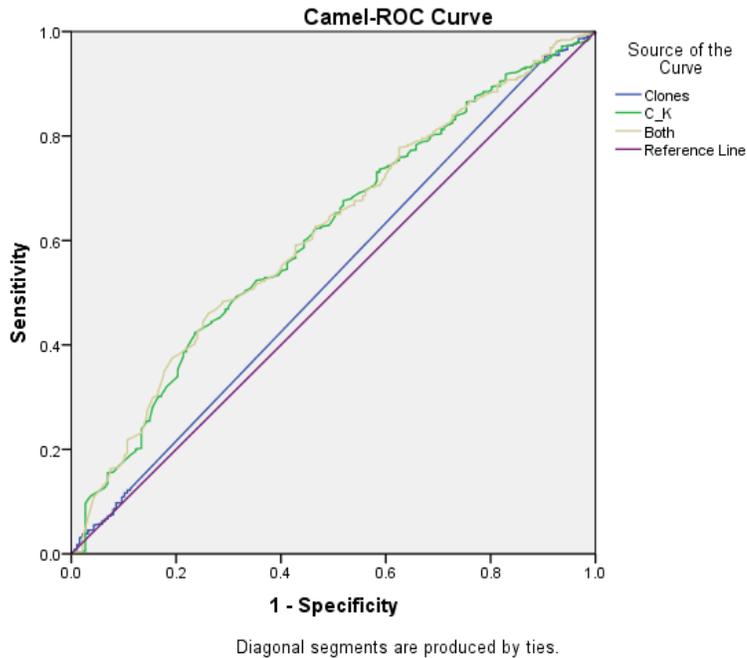
Figure 16: Multivariate Analysis - ROC Curve for Synapse



**Figure 17: Multivariate Analysis - ROC Curve for Ant**



**Figure 18: Multivariate Analysis - ROC Curve for Xalan**



**Figure 19: Multivariate Analysis - ROC Curve for Camel**

As the observed results show that the ROC curve for multivariate models based on C&K metrics are performing better than multivariate models of clone metrics in average and in four out five subject systems as well. Similarly, we can see also the behavior of the multivariate models of combined metrics suite, it is better than multivariate models of clone metrics in average and in four out five subject systems as well. For the comparison of ROC behavior between multivariate models of C&K and the combined suite of metrics, there were no consistent results to judge which is better in general, because the average of the results was for the sake of the combined suite. However, four out the five subject systems were C&K models performing better than the combined suite models. We conclude the following with respect to the hypotheses, we accept the null hypothesis of HYP-C, and we reject the null hypothesis HYP-D. For the HYP-E, we don't have a clear evidence to accept or to reject the null hypothesis.

### 4.7.3 Comparisons of Regression models of Code Clones Metrics and Neural Network Models in Predicting Fault-Proneness

Although, basic classifiers such as logistic regression gives similar results to more sophisticated classifiers [51], it could be different for code clones when incorporated with ANN prediction models. The Neural Network (NN) models that are used in this study is Multi-Layer Perceptron (MLP) model. MLP is an artificial neural network of type feed-forward that use back-propagation as supervised learning technique [63]. The results of regressions are compared with MLP models results. After generating the MLP models with code clones metrics as inputs, slight differences are exist for the advantage of MLP models regarding the correctly classified rate and also for ROC area as can be seen in the following two tables:

	Correctly Classified Rate			ROC Area	
	Regression	NN:MLP		Regression	NN:MLP
velocity	73.03%	<b>74.28%</b>	velocity	0.644	<b>0.645</b>
synapse	72.88%	<b>73.38%</b>	synapse	0.614	<b>0.632</b>
Ant	<b>70.76%</b>	70.59%	Ant	0.621	<b>0.627</b>
Xalan	76.08%	<b>76.40%</b>	Xalan	<b>0.584</b>	0.580
camel	70%	<b>71.63%</b>	camel	0.649	<b>0.667</b>
Avg.	72.52%	<b>73.26%</b>	Avg.	0.622	<b>0.630</b>

The ANN models are better in four subject systems with regard to correctly classified rate, yet the fifth subject system that the regression model was performing better, the difference is very marginal. Moreover, for the ROC area ANN models scored better in four subject systems and again, the fifth subject system that regression models performed better, the difference in ROC value is very small. Hence, using code clones metrics as inputs to artificial neural networks will probably enhance fault-proneness prediction.

## **4.8 Threats to validity**

In this section, threats to validity will be discussed from four perspectives, namely threats to construct validity, threats to internal validity, threats to statistical conclusion validity, and threats to external validity.

### **4.8.1 Construct Validity**

We have used fault-proneness and fault-density as attributes lead to functional correctness and leaving away the third dimension which is fault severity. But our main focus of this study is to explore the implication of a class to be faulty when it contains a clone not to measure its severity. Another threat to construct validity is when we have measured types of clones based on location. We have looked at the clone whether it is inside the class or outside. We didn't consider where it is outside, it could be in the same component and it could be in another component. The focus of this study is on class level so this has minimized the effects of ignoring specific location of the clone outside the studied class.

### **4.8.2 Internal Validity**

In this empirical study, two threats to internal validity have been identified. First, all of the faults information is gathered from another study where we relied on the accuracy they yielded [50]. Second, CCFinder was used exclusively to detect clones out of the five subject systems which might affect the results. This threat was eliminated by having a manual inspection of the reported clones and then number of clones was corrected by removing false positive results.

### **4.8.3 Statistical Conclusion Validity**

In this research, we have done several statistical tests as part of our empirical studies of impact of code cloning on fault-proneness and fault-density of object-oriented classes. Thus, we have the following elements as threats to conclusion validity [64]. First, we might have an error in our answers to the null hypotheses, such as rejecting the null hypothesis where accepting the null hypothesis is the correct answer. Second, we might accept a null hypothesis that should be rejected which is the correct answer. In order to minimize the threats of statistical conclusion validity, the statistical power has to be increased [65-67]. As we tried to increase the mentioned statistical power by incorporating the following ways: First, we have increased the sample size of the experiment which increases the precision of estimate to dependent variables [68]. Second, we have included only the factors that are directly related to our questions in order to answer the null hypotheses [69].

### **4.8.4 External Validity**

This study has five external threats to validity. First, all the subject systems are coming from a sole source which is Apache organization, other organizations could have different standard of coding and faults reporting which might affect the results. Second, all of the subject systems are open source systems, which don't represent all other spectrums of software industrials. Third, all of the subject systems are written in Java, which other languages could have different behavior as LOC and inheritance play an important role in supplying metrics with different values from what Java systems provides like python and C++. Fourth, the selected subject systems are ranged from quite small to relatively medium size, but we believe that the diversity of systems sizes is

giving us higher confidence that the subject systems are biased to a particular tier of systems size. Fifth, the level of experience of the developers and domain knowledge of the software are important aspects that might influence number of faults or clones in the subject systems, but having five subject systems from different domains and sizes will minimize the effects of the mentioned factors.

## **CHAPTER 5 EMPIRICAL STUDY OF SOFTWARE CLONES AND FAULT-DENSITY**

Fault density is represented by a derived metric from normalizing the number of bugs per 1K line of code (LOC). Both faults and LOC were collected from published study results [41]. We used linear regression to model the usefulness of using clone metrics as predictors of fault density.

### **5.1 Goal and Research Questions:**

The goal of this empirical experiment is to analyze code clone metrics for the purpose of exploring relationships with respect to class fault-density from software developer's perspective in the context of object-oriented open source systems.

In this part of the study, the following research questions are the motivation of the research:

F. Are there relationships between code clones and fault density of object-oriented classes?

For this question three hypotheses were constructed:

#### **HYP-F1:**

H0: There is no relationship between code clones and fault density of object-oriented classes

H1: There is a relationship between code clones and fault density of object-oriented classes

#### **HYP-F2:**

H0: There is no relationship between clone types and fault density

H1: There is a relationship between clone types and fault density

**HYP-F3:**

H0: There is no relationship between clone location and fault density

H1: There is a relationship between clone location and fault density

G. Are clone metrics good indicators of fault density?

For this question we have 13 hypothesis:

**HYP-G1:**

H0: number of clone fragments is not a good indicator of fault density

H1: number of clone fragments is a good indicator of fault density

**HYP-G2:**

H0: clone density is not a good indicator of fault density

H1: clone density is a good indicator of fault density

**HYP-G3:**

H0: CRFL is not a good indicator of fault density

H1: CRFL is a good indicator of fault density

**HYP-G4:**

H0: NBR is not a good indicator of fault density

H1: NBR is a good indicator of fault density

**HYP-G5:**

H0: RSA is not a good indicator of fault density

H1: RSA is a good indicator of fault density

**HYP-G6:**

H0: RSI is not a good indicator of fault density

H1: RSI is a good indicator of fault density

**HYP-G7:**

H0: CVR is not a good indicator of fault density

H1: CVR is a good indicator of fault density

**HYP-G8:**

H0: RNR is not a good indicator of fault density

H1: RNR is a good indicator of fault density

**HYP-G9:**

H0: Inter-class clone is not a good indicator of fault density

H1: Inter-class clone is a good indicator of fault density

**HYP-G10:**

H0: Intra-class clone is not a good indicator of fault density

H1: Intra-class clone is a good indicator of fault density

**HYP-G11:**

H0: Type-1 clone is not a good indicator of fault density

H1: Type-1 clone is a good indicator of fault density

**HYP-G12:**

H0: Type-2 clone is not a good indicator of fault density

H1: Type-2 clone is a good indicator of fault density

**HYP-G13:**

H0: Type-3 clone is not a good indicator of fault density

H1: Type-3 clone is a good indicator of fault density

H. Do clone metrics have better fault density prediction than C&K metrics?

**HYP-H:**

H0: Clone metrics are not better predictors of fault density than C&K metrics

H1: Clone metrics are better predictors of fault density than C&K metrics

I. Does the combination of clone metrics and C&K metrics as input in prediction models of fault density yield better results than clone metrics?

**HYP-I:**

H0: The combination of clone metrics and C&K metrics as input in prediction models of fault density does not yield better results than clone metrics only

H1: The combination of clone metrics and C&K metrics as input in prediction models of fault density yields better results than clone metrics only

- J. Does the combination of clone metrics and C&K metrics as input in prediction models of fault density yield better results than C&K metrics?

**HYP-J:**

H0: The combination of clone metrics and C&K metrics as input in prediction models of fault density does not yield better results than C&K metrics only

H1: The combination of clone metrics and C&K metrics as input in prediction models of fault density yields better results than C&K metrics only

## 5.2 Descriptive Statistics

After generating some statistics of subject systems such as Min, Max, Mean, and Standard Deviation, we found some general observations of these subject systems. As discussed in sec.4.3, for distributed faults over classes, similarly, in this section we shed some lights on fault-density statistics. As fault-density has non-discrete scale, some box-plots have been generated to describe the distribution of fault-density in subject systems classes.

The following graphs (box-plots) show statistics of subject systems in addition to the auxiliary system (combining all subject systems into one):

Fault-Density for Velocity System:

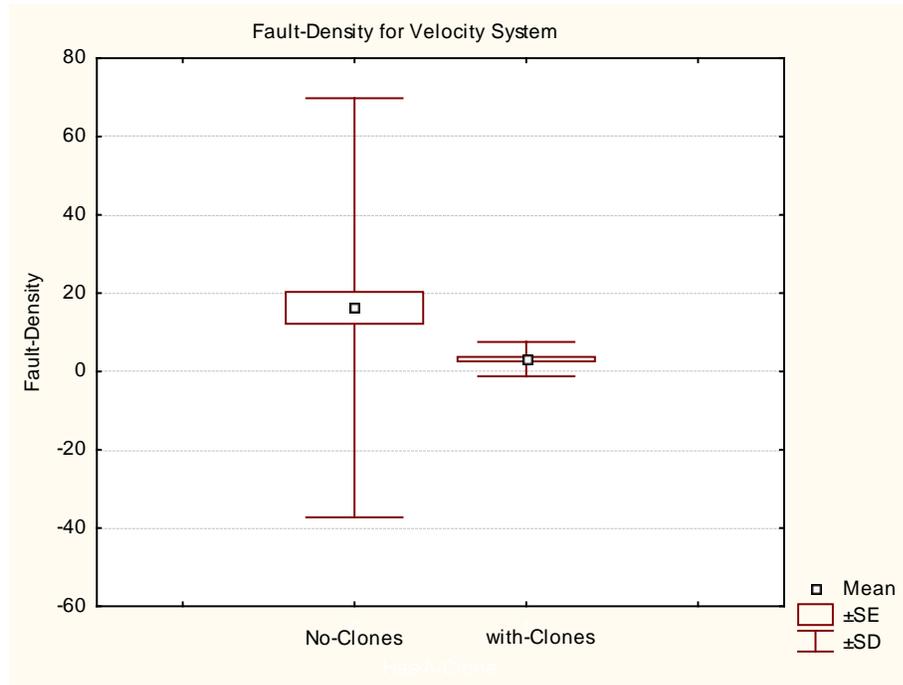
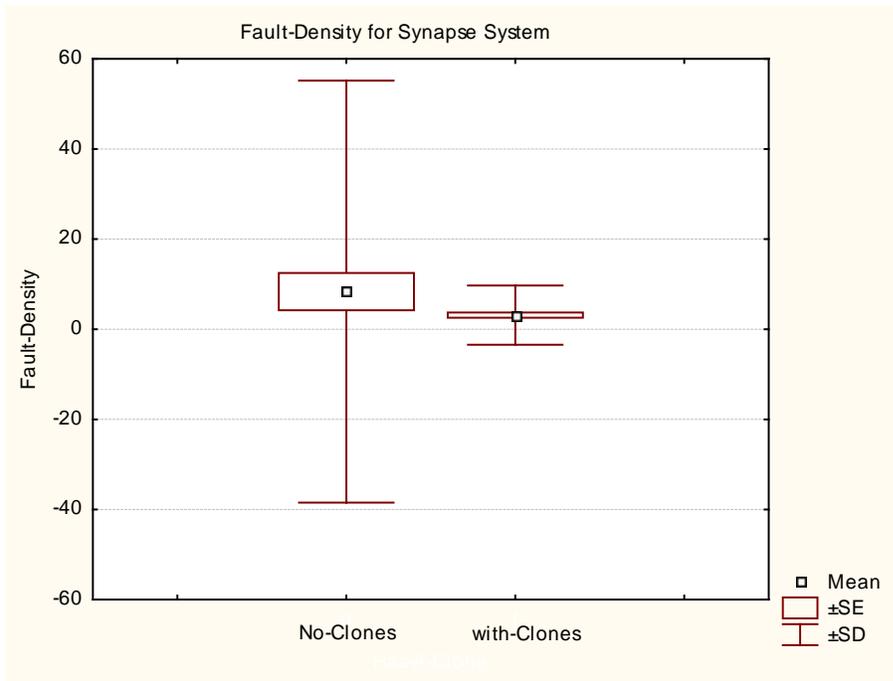


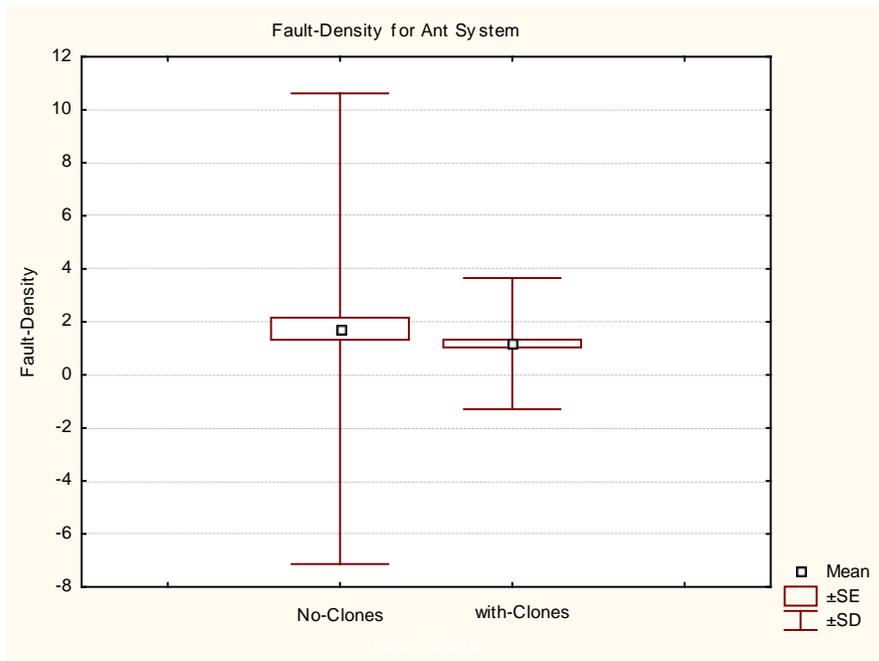
Figure 20: Box-Plot of Fault-Density for Velocity System

Fault-Density for Synapse system:



**Figure 21: Box-Plot of Fault-Density for Synapse System**

Fault-Density for Ant System:



**Figure 22: Box-Plot of Fault-Density for Ant System**

Fault-Density for Xalan System:

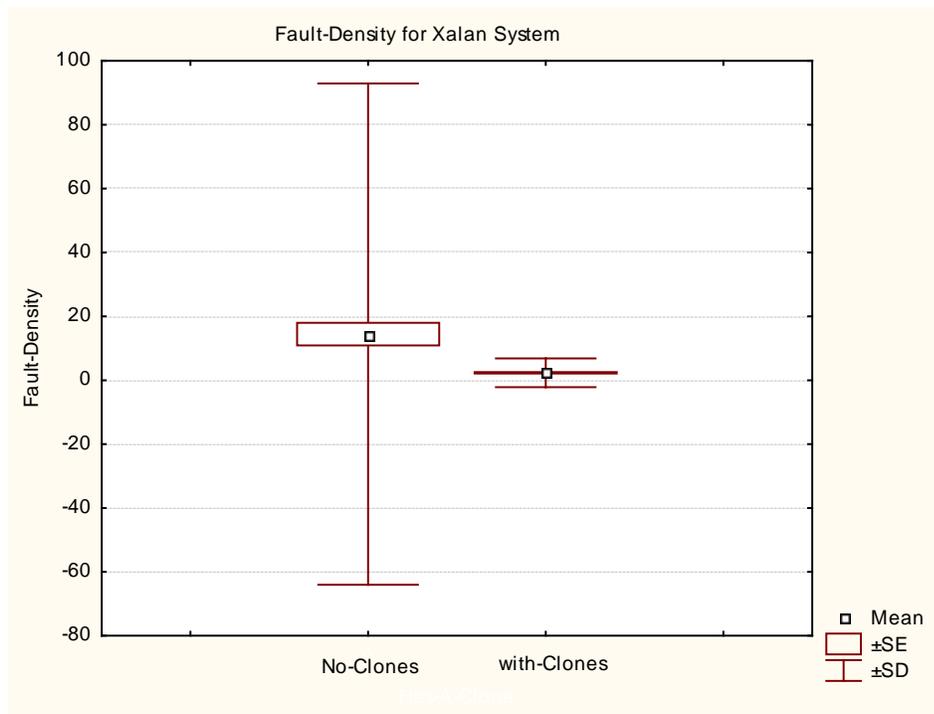


Figure 23: Box-Plot of Fault-Density for Xalan System

Fault-Density for Camel System:

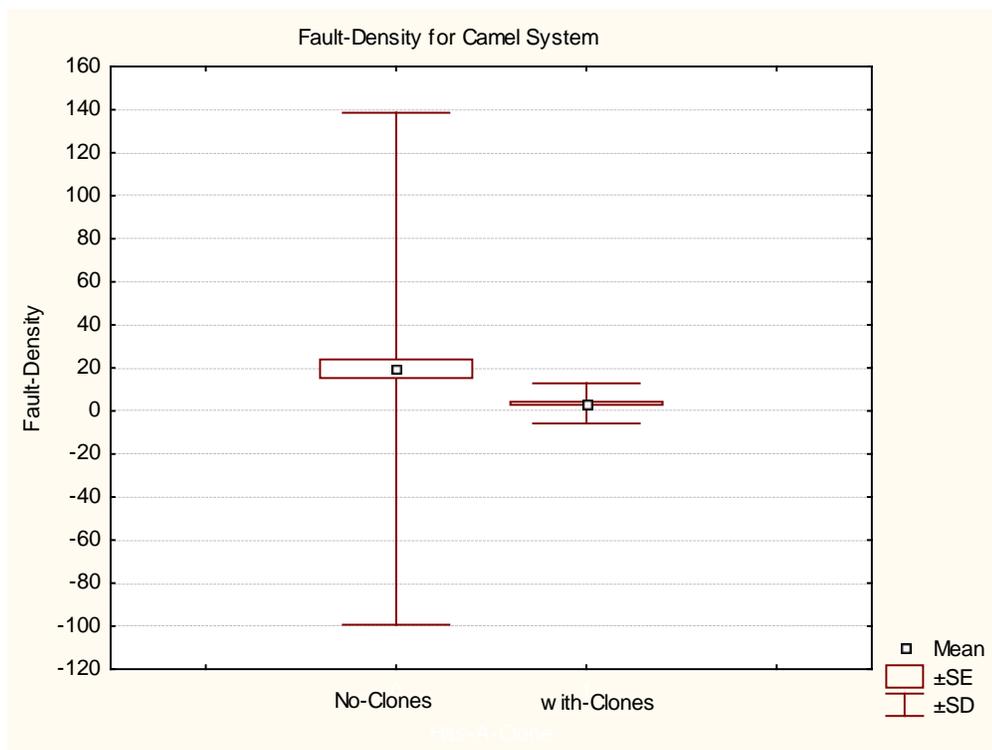
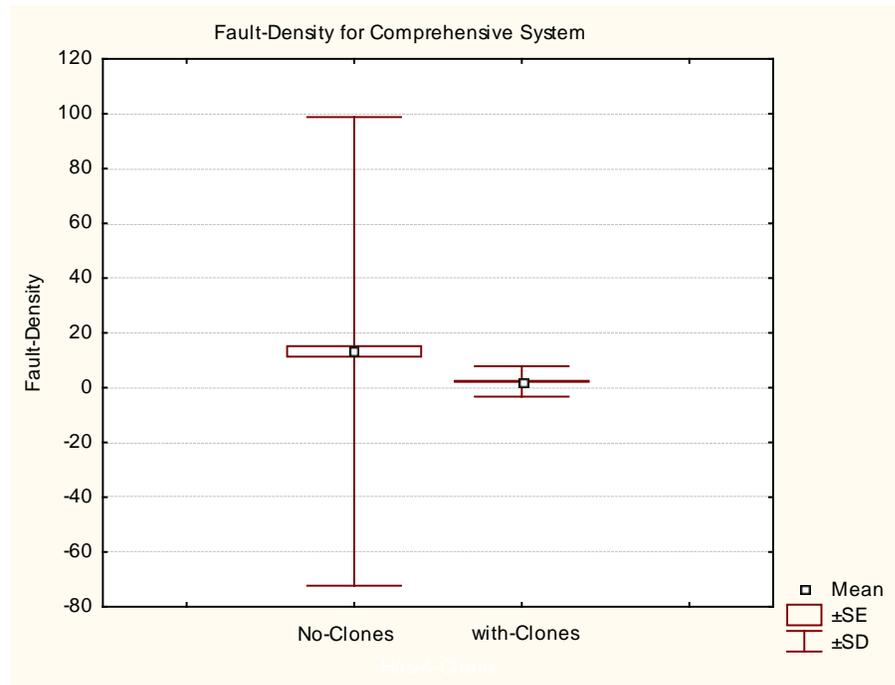


Figure 24: Box-Plot of Fault-Density for Camel System

## Fault-Density for Comprehensive System:



**Figure 25: Box-Plot of Fault-Density for Comprehensive System**

### Observations:

As opposite to fault-proneness, classes without clones have higher fault-density than classes with clones.

After investigating the input data, the average size of classes that have clones are almost double the size of classes that don't have clones.

Some of the classes that are faulty are very small in size as line of code, which will increase the fault-density in that class.

The class size when it contains clone/s usually is not tiny class as per lines of code; it makes sense as clones needed when a class requires multiple features that are similar.

The following table shows the average LOC for the subject systems in both cases:

	velocity		synapse		ant		xalan		camel		comprehensive	
	no-clones	with-clones	no-clones	with-clones								
Avg. LOC	230.92	304.70	104.02	313.95	160.37	471.20	189.27	833.67	86.76	261.08	142.08	541.39

The collected clone metrics that are based on types and location have been analyzed from fault-density perspective. Based on each type, the ratios of faulty classes were generated.

To Sum up, fault-density behavior didn't match with fault-proneness behavior in the case when classes that contain clones were more fault-proneness, but apparently less fault-density. Therefore, we cannot judge that there is a direct relationship between fault-density and fault-proneness.

### 5.3 Principal Component Analysis

Principal component analysis for the subject systems is discussed in details in (sec.4.4).

### 5.4 Mann-Whitney Test

In this section, Mann-Whitney test is used to help in answering the question of having a relationship (if any) between code clones and fault-density. Mann-Whitney test is a non-parametric test that gives cumulative probabilities of certain null hypothesis by measuring the sum of the ranked two-sample data [50]. The software used to conduct Mann-Whitney tests is STATISTICA software package. The readings of Mann-Whitney tests are summarized in below tables:

Mann-Whitney test for general clones including all types:

Fault-Density	p-level
velocity	0.54096
synapse	<b>0.00478</b>
ant	<b>0.0002</b>
xalan	<b>0.04647</b>
camel	0.64963
comprehensive	<b>0</b>

As we can see in the above table, numbers in bold highlight significant figures whereby velocity and camel systems' p-values are more than 0.05, which is similar to the results of fault-proneness in sec.4.5 . However, the general trend as it is clear in the following box-plots: *Figure 20, Figure 21, Figure 22, Figure 23, Figure 24, and Figure 25*, show that there are inconsistent results of code clones and fault-density. This means that where in some systems, the difference is significant, the box-plot shows the fault-density of classes with no-clones have higher fault-density. So, we don't have clear evidence to accept or reject the null hypothesis of HYP-F1.

To answer question Q.F-2, Mann-Whitney test for all systems were conducted and presented below based on clone location, namely inter-clone, intra-clone, and hybrid-clone.

Mann-Whitney test for clone locations whether inter clones or intra clones:

**Table 47: Mann-Whitney Test Result-Fault-Density Based on Clones Location for Velocity**

Glossary B: Velocity	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	0.884679			
C	0.778884	0.668336		
D	0.306627	0.393789	0.980535	

Mann-Whitney test shows that there is no significant difference among individual groups of data. This cannot lead us to any conclusion for clone location relationship with fault-density. Another visit to the box-plot for fault distribution over the clone location groups will give more information that will help us in reaching a conclusion.



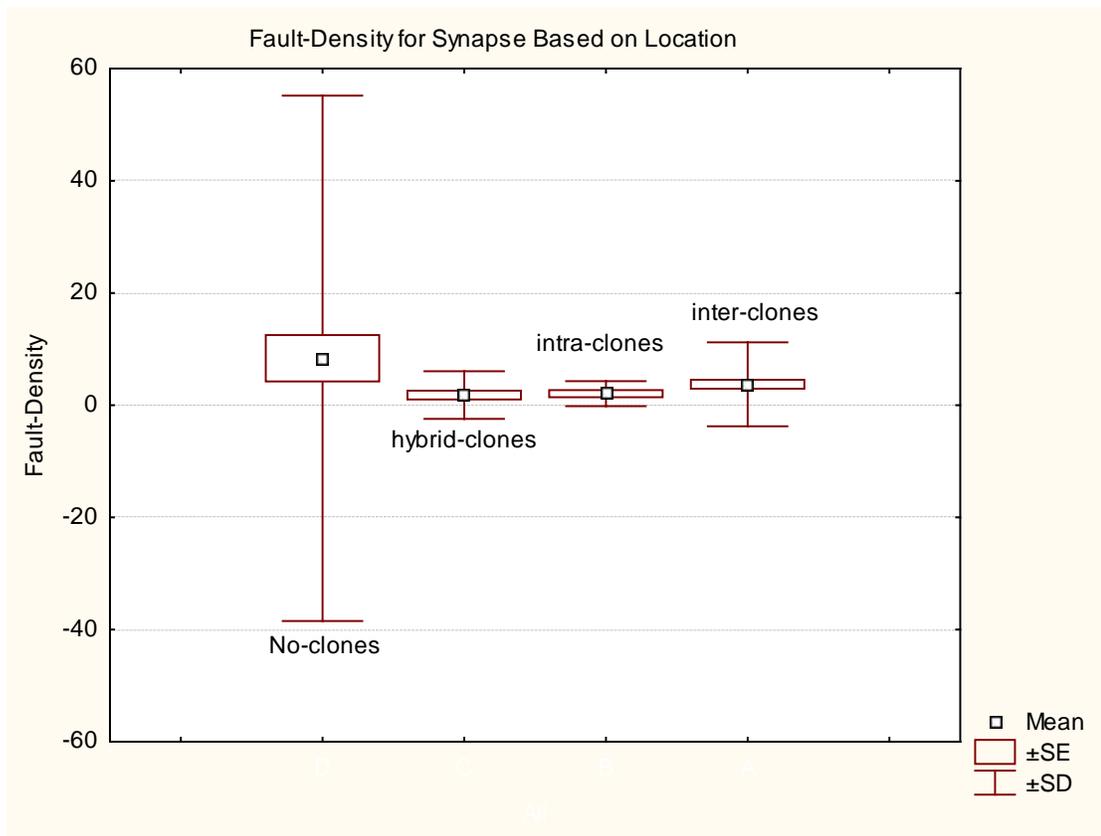
**Figure 26: Box-Plot of Fault-Density Based on Location for Velocity System**

From the above box-plot, it shows that not only there is no significant difference from the Mann-Whitney test, but also the fault-density in the classes with no-clones are higher than classes with clones regardless of location. The summary in [Table 59] has a generic view of the test results with the context of all subject systems in this study.

**Table 48: Mann-Whitney Test Result-Fault-Density Based on Clones Location for Synapse**

Glossary B: Synapse	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	0.940214			
C	0.078206	0.121719		
D	<b>0.00266</b>	<b>0.049733</b>	0.617308	

Mann-Whitney test shows that there are significant differences between inter-clones and the group of no-clones, similarly between intra-clones and no-clones group. This observation alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-density. Another visit to the box-plot for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.



**Figure 27: Box-Plot of Fault-Density Based on Location for Synapse System**

From the p-values above, there are some significant differences between clones based on location groups of data with respect to fault-density. However, the box-plot of the system shows the classes that have inter-clones, intra-clones and hybrid-clones are less fault-density than classes without clones. The summary in [Table 59] has a generic view of the test results with the context of all subject systems in this study.

**Table 49: Mann-Whitney Test Result-Fault-Density Based on Clones Location for Ant**

Glossary B: Ant	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	<b>0.003809</b>			
C	<b>0.017493</b>	0.59593		
D	0.173831	<b>0.000018</b>	<b>0.000159</b>	

Mann-Whitney test shows that there are significant differences between inter-clones and the group of intra-clones, similarly between inter-clones and hybrid-clones. The same applies to intra-clones and no-clones group whereby the difference is significant. Data groups of hybrid-clones and no-clones group have also a significant difference such that the no-clones data group has higher fault-density than classes with hybrid-clones. These observations alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-density. Another visit to the box-plot for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.

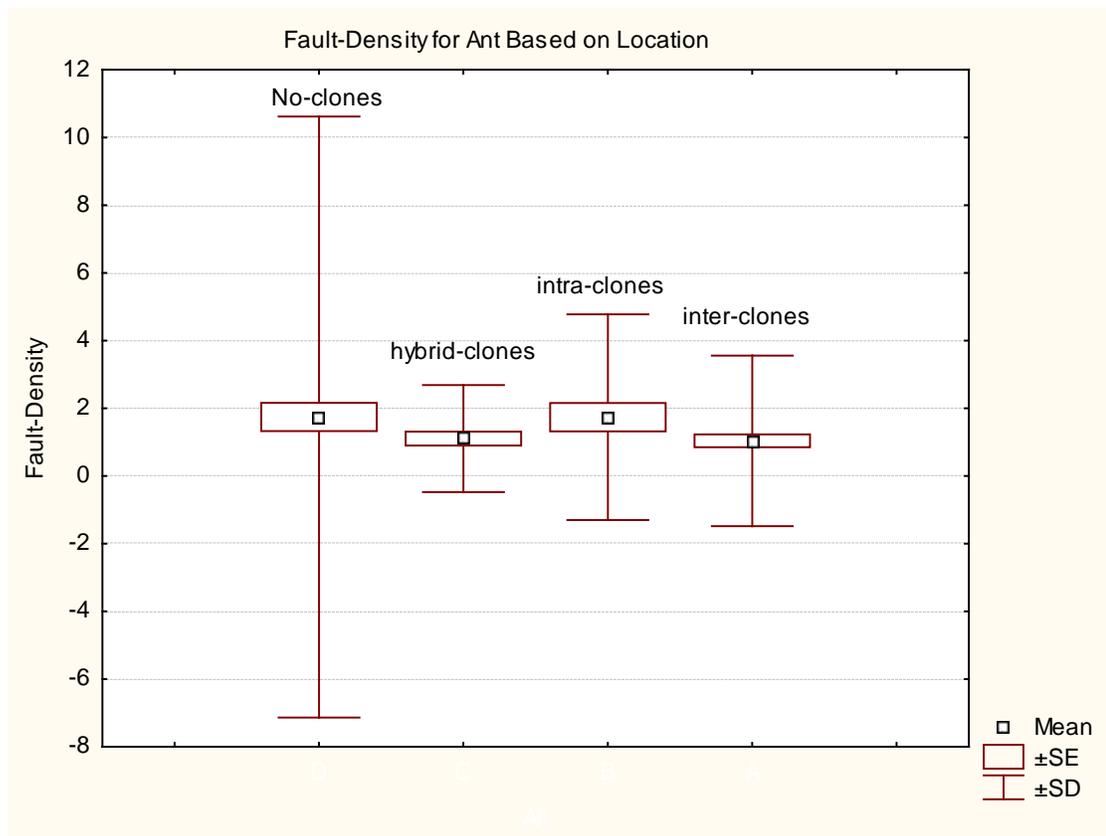


Figure 28: Box-Plot of Fault-Density Based on Location for Ant System

From the p-values above, there are some significant differences between clones based on location groups of data with respect to fault-density. However, the box-plot of the system shows the classes that have inter-clones, intra-clones and hybrid-clones are always less fault-density than classes without clones. The summary in [Table 59] has a generic view of the test results with the context of all subject systems in this study.

**Table 50: Mann-Whitney Test Result-Fault-Density Based on Clones Location for Xalan**

Glossary B: Xalan	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	0.847402			
C	0.074757	0.161262		
D	0.195961	0.395867	<b>0.023198</b>	

Mann-Whitney test shows that there are significant differences between hybrid-clones and the group of no-clones. This observation alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-density. Another visit to the box-plot for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.

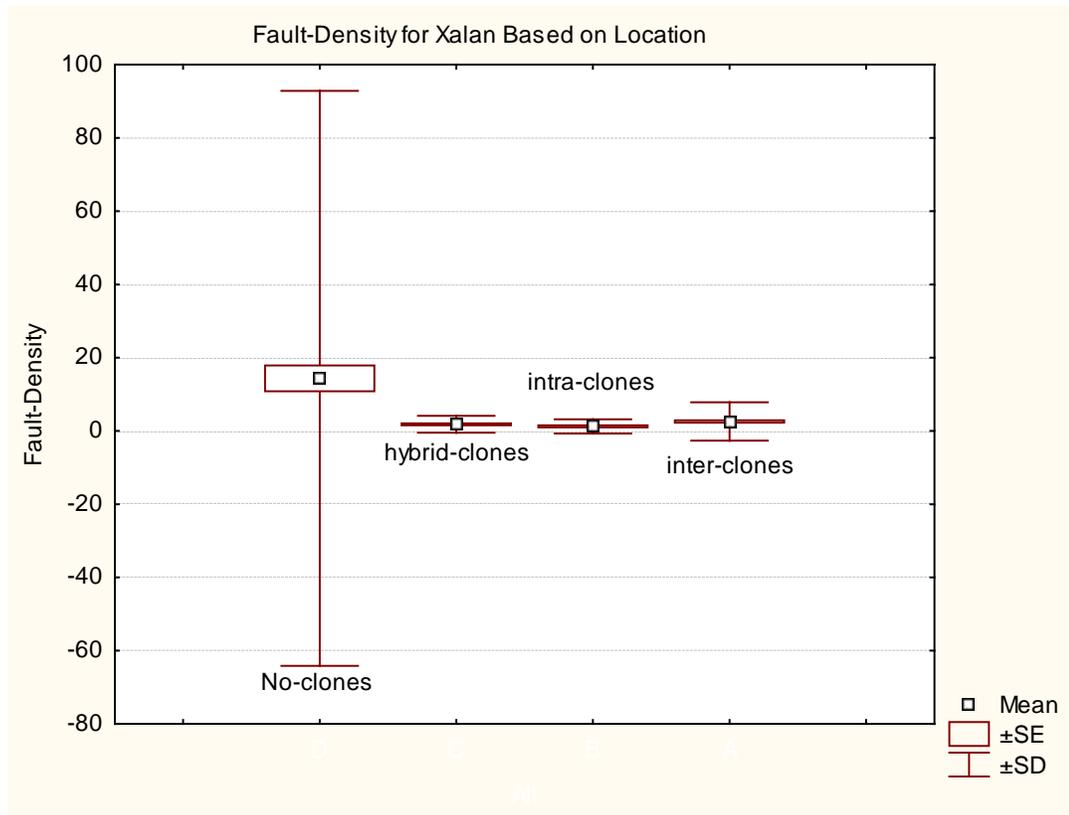


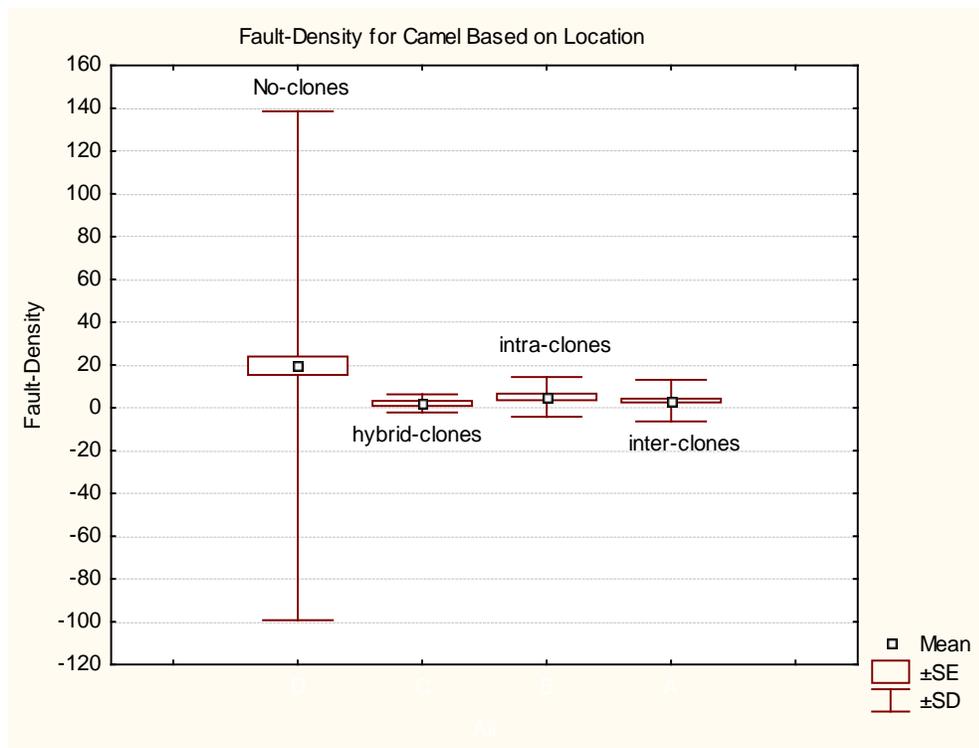
Figure 29: Box-Plot of Fault-Density Based on Location for Xalan System

From the p-values above, there is a significant difference between hybrid-clones and the group of no-clones with respect to fault-density. However, the box-plot of the system shows the classes that have inter-clones, intra-clones and hybrid-clones are less fault-density than classes without clones. The summary in [Table 59] has a generic view of the test results with context of all subject systems in this study.

Table 51: Mann-Whitney Test Result-Fault-Density Based on Clones Location for Camel

Glossary B: Camel	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	<b>0.002939</b>			
C	0.689805	0.209724		
D	0.353864	<b>0.005785</b>	0.953129	

Mann-Whitney test shows that there are significant differences between inter-clones and the group of intra-clones. Similarly between intra-clones and the group of no-clones. This observation alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-density. Another visit to the box-plot for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.



**Figure 30: Box-Plot of Fault-Density Based on Location for Camel System**

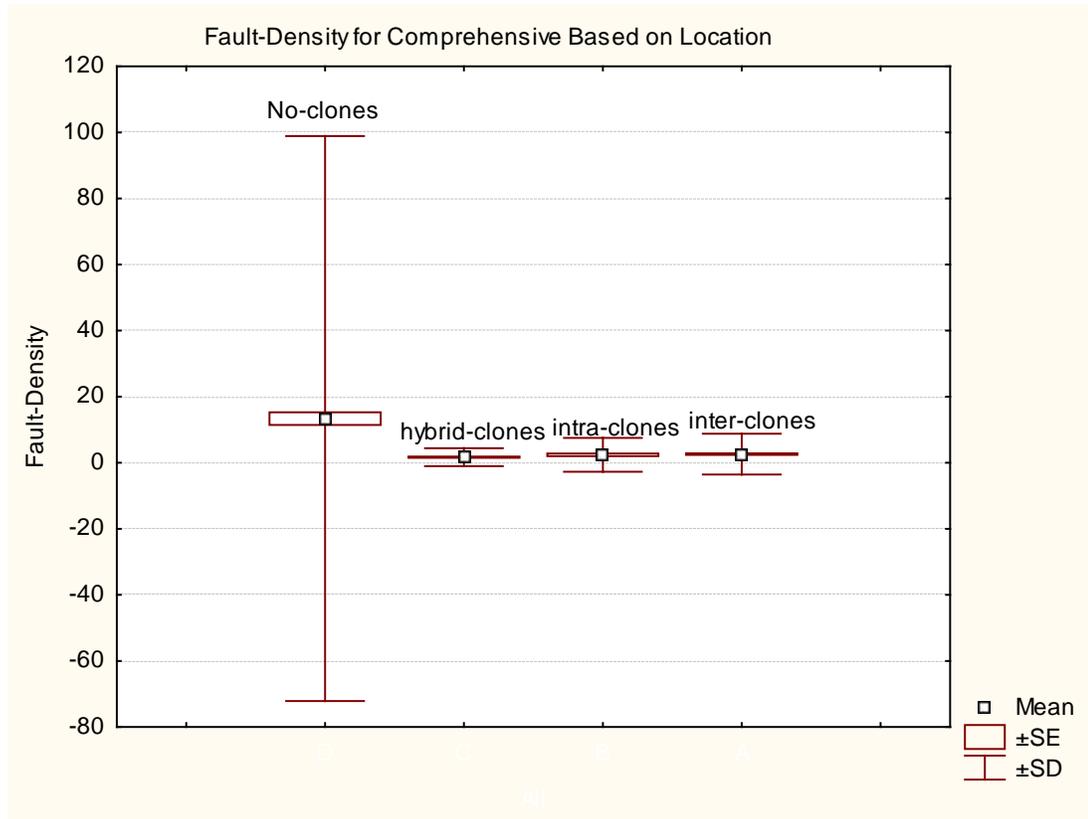
From the p-values above, there is a significant difference between inter-clones and the group of intra-clones with respect to fault-density, in addition to the pair between intra-clones and the group of no-clones. However, the box-plot of the system shows that classes that have inter-clones, intra-clones and hybrid-clones are less fault-density than

classes without clones. The summary in [Table 59] has a generic view of the test results with context of all subject systems in this study.

**Table 52: Mann-Whitney Test Result-Fault-Density Based on Clones Location for Comprehensive**

Glossary B: total	A: InterOnly	B: IntraOnly	C: Inter&Intra	D: Has No Clones
inter-VS-intra	A	B	C	D
A				
B	0.077424			
C	0.073267	0.937621		
D	<b>0.000033</b>	<b>0.000005</b>	<b>0.000001</b>	

Mann-Whitney test shows that there are significant differences between inter-clones, intra-clones, and hybrid-clones each against the group of no-clones. These observations alone cannot lead us to any conclusion to judge if there is a relationship between clone locations with fault-density. Another visit to the box-plot for fault distribution over the clone location groups will provide more information that will help us in reaching a conclusion.



**Figure 31: Box-Plot of Fault-Density Based on Location for Comprehensive System**

From the p-values above, there is a significant difference between inter-clones, intra-clones, and hybrid-clones each against the group of no-clones. Similar to other subject systems, the box-plot of the comprehensive system shows classes that have inter-clones, intra-clones and hybrid-clones are less fault-density than classes without clones. The summary in [Table 59] has a generic view of the test results with context of all subject systems in this study.

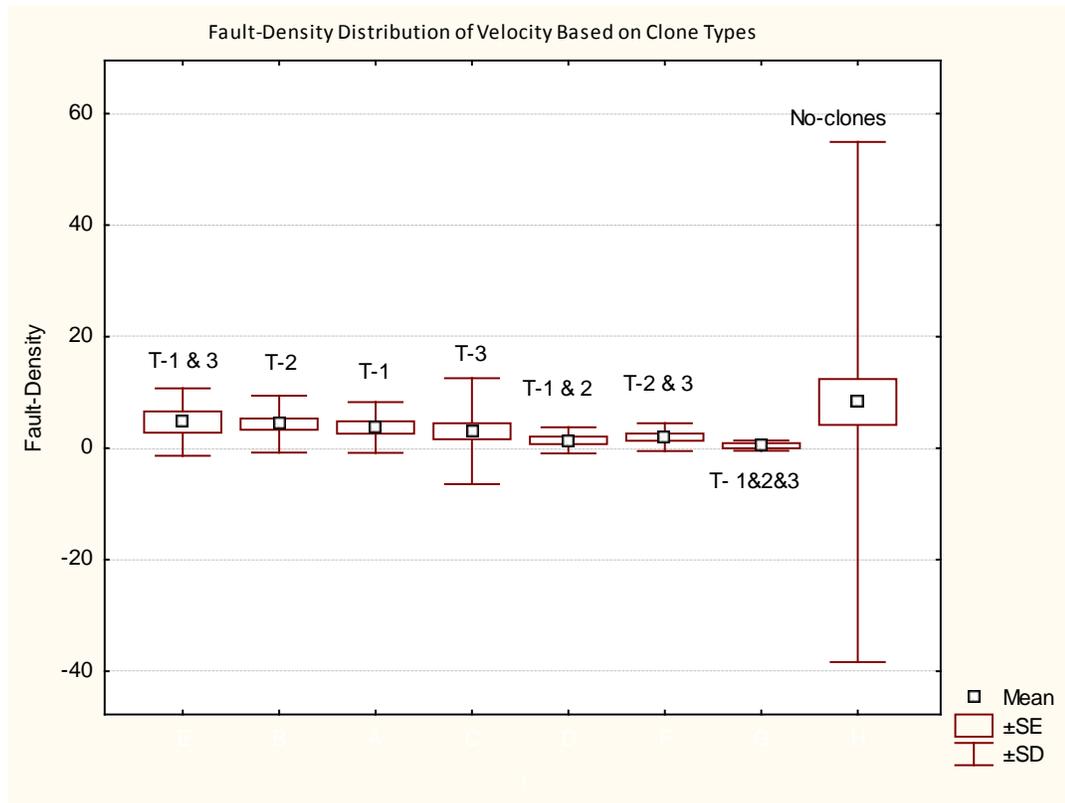
**Mann-Whitney test for clones based on types:**

Velocity:

**Table 53: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Velocity**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.338635							
C	0.572615	0.77156						
D	0.338635	0.84889	0.846523					
E	1	1	1	1				
F	0.25356	0.599846	0.569204	0.599846	1			
G	1	1	1	1	1	1		
H	0.311663	0.845386	0.965223	0.86068	1	0.617445	1	

Similar to location based clones, Mann-Whitney test shows that there is no significant difference among the individual groups of data. This cannot lead us to any conclusion for clone based on type relationships with fault-density. Another visit to the histogram for fault distribution over the clone type groups will give more information that will help us in reaching a conclusion.



**Figure 32: Box-Plot of Fault-Density Based on Type for Velocity System**

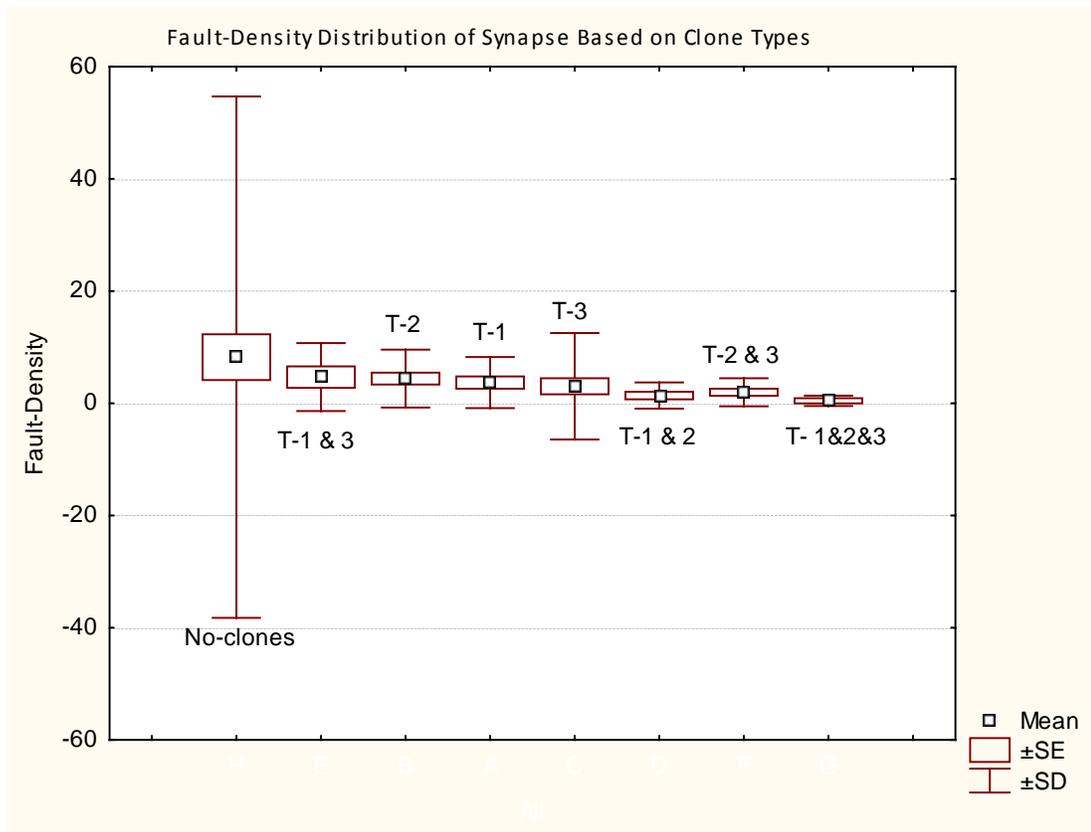
From the p-values above, there is no significant difference between clones based on types with respect to fault-density. However, the box-plot of the system shows that all clone types in the classes are less fault-density than classes without clones. The summary in **Table 60** has a generic view of the test results with the context of all subject systems in this study.

Synapse:

**Table 54: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Synapse**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.695297							
C	<b>0.007791</b>	<b>0.000181</b>						
D	0.230737	<b>0.037091</b>	0.306123					
E	0.813012	0.881486	0.064347	0.264368				
F	0.343776	0.123105	0.060884	0.510996	0.462354			
G	0.117701	<b>0.041953</b>	0.940237	0.530621	0.24015	0.304306		
H	<b>0.0097</b>	<b>0.000058</b>	0.814904	0.370551	0.078556	0.063934	0.877808	

Mann-Whitney test shows that there are significant differences among some of the individual groups of data such as Type-1 and Type-3, similarly Type2 and Type-3. Additionally, three groups have significant differences between them. These observations might not lead us to any conclusion for clones based on type relationships with fault-density. Another visit to the box-plot for fault distribution over the clone type groups will give more information that will help us in reaching a conclusion.



**Figure 33: Box-Plot of Fault-Density Based on Type for Synapse System**

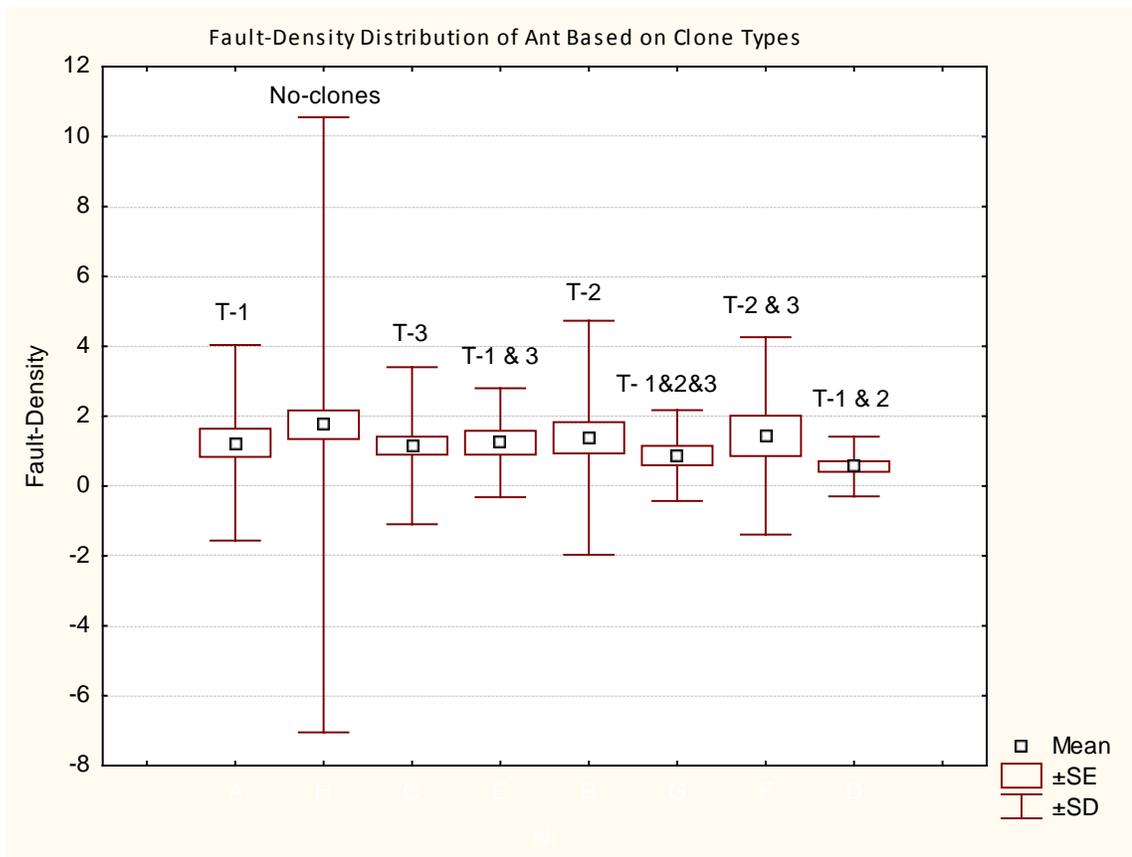
From the p-values above, there are significant differences between clones based on types with respect to fault-proneness. This system in addition to other subject systems shows that all the data groups are less fault-density when compared with the data group of no-clones. The summary in [Table 60] has a generic view of the test results with context of all subject systems in this study.

Ant:

**Table 55: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Ant**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.863691							
C	0.993175	0.816115						
D	0.979041	0.867588	0.960522					
E	0.237335	0.180082	0.220332	0.146619				
F	0.912008	0.774179	0.887866	0.756965	0.539503			
G	0.696801	0.591025	0.730616	0.620432	0.44794	0.936887		
H	0.064249	0.11566	<b>0.032302</b>	0.083191	<b>0.002771</b>	0.155191	0.067196	

Mann-Whitney test shows that there are significant differences among some of the individual groups of data such as Type-3 and no-clones, similarly Type-1&3 and no-clones. These observations might not lead us to any conclusion for clones based on type relationships with fault-density. Another visit to the box-plot for fault distribution over the clone types groups will give more information that will help us in reaching a conclusion.



**Figure 34: Box-Plot of Fault-Density Based on Type for Ant System**

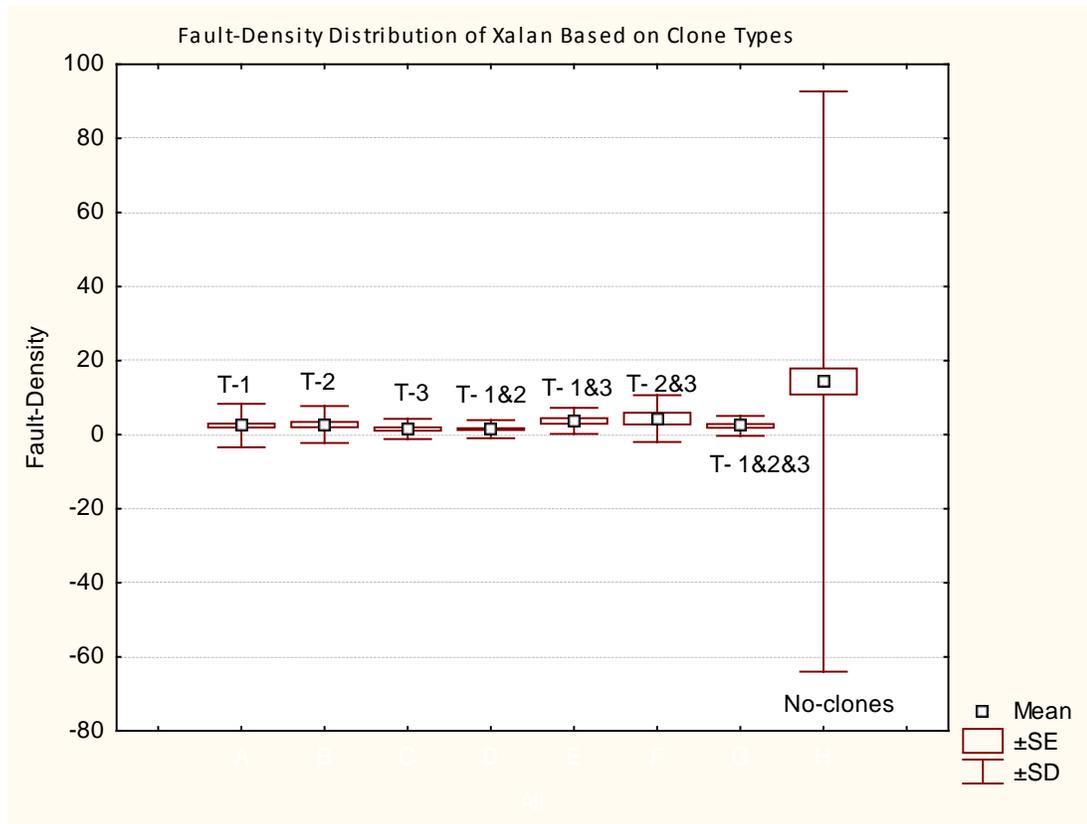
From the p-values above, there are significant differences between clones based on types with respect to fault-proneness. This system in addition to other subject systems continues the same behavior with respect to fault-density distribution. All the data groups are less fault-density when compared with the data group of no-clones. The summary in [Table 60] has a generic view of the test results with context of all subject systems in this study.

Xalan:

**Table 56: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Xalan**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.105281							
C	0.75533	0.183597						
D	<b>0.000317</b>	0.553636	<b>0.009959</b>					
E	<b>0.000508</b>	<b>0.040001</b>	<b>0.000247</b>	<b>0.00889</b>				
F	<b>0.029442</b>	0.19507	<b>0.032411</b>	0.232356	0.645018			
G	<b>0.003527</b>	0.174581	<b>0.013374</b>	0.072066	0.244076	0.649241		
H	0.155765	0.405826	0.260517	<b>0.000475</b>	<b>0.005463</b>	0.124743	<b>0.021507</b>	

Mann-Whitney test shows that there are significant differences among some of the individual groups of data such as Type-1, Type-2, Type-3, Type-1&2 against Type-1&3. Type1&2, Type-1&3 and Type-1&2&3 against no-clones group. This might not lead us to any conclusion about relationships between clone types and fault-density. Another visit to the box-plot of the system for fault distribution over the clone types groups will give more information that will help us in reaching a conclusion.



**Figure 35: Box-Plot of Fault-Density Based on Type for Xalan System**

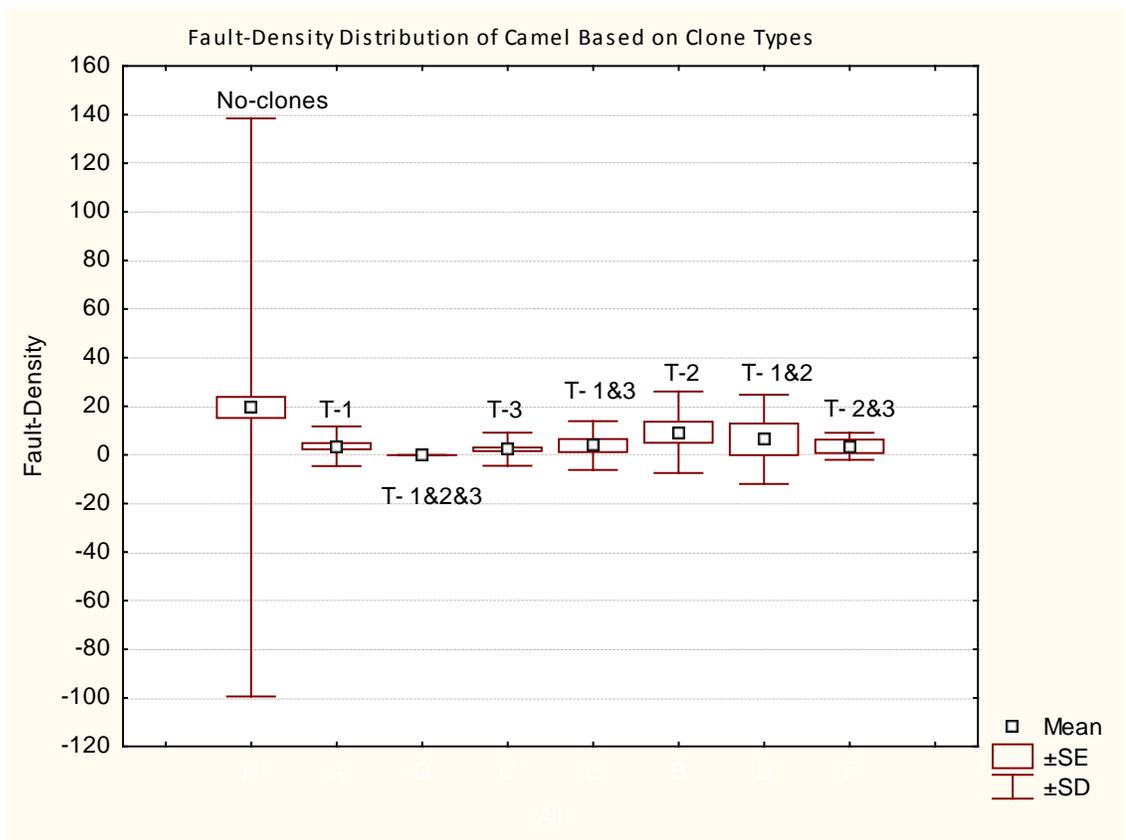
From the p-values above, there are significant differences between clones based on types with respect to fault-proneness. Similarly, this system in addition to other subject systems continues the same behavior with respect to fault-density distribution. All the data groups are less fault-density when compared with the data group of no-clones. The summary in [Table 60] has a generic view of the test results with the context of all subject systems in this study.

Camel:

**Table 57: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Camel**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.203333							
C	0.424845	0.051974						
D	0.573016	0.250723	0.753837					
E	0.857678	0.41028	0.405763	0.515296				
F	0.405052	1	0.200271	0.264996	0.527079			
G	0.175118	0.078984	0.236688	0.386477	0.156845	0.06789		
H	0.599766	0.070484	0.661659	0.692529	0.56511	0.242631	0.233054	

Mann-Whitney test shows that there is no significant difference among the individual groups of data. All data groups have less fault-density when compared with the group of data of no-clones. This cannot lead us to any conclusion for clone location relationship with fault-density. Another visit to the box-plot for fault distribution over the clone location groups will give more information that will help us in reaching a conclusion.



**Figure 36: Box-Plot of Fault-Density Based on Type for Camel System**

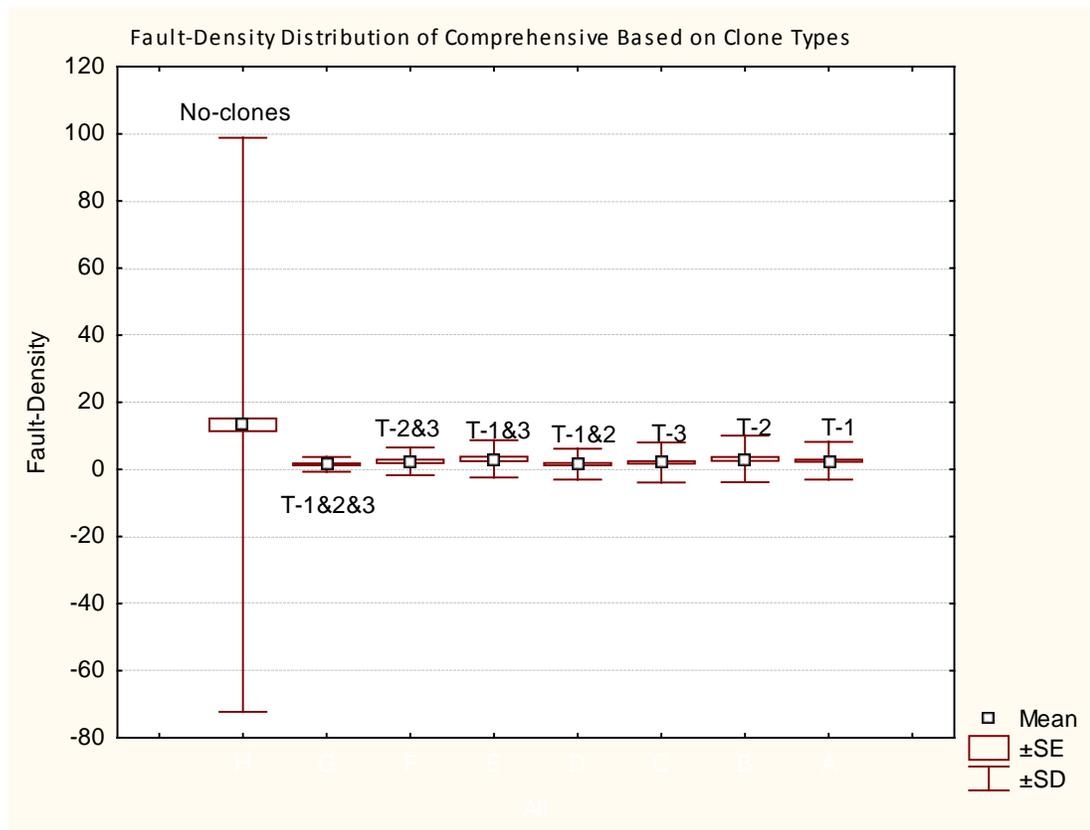
From the p-values above, there is no significant difference between clones based on types with respect to fault-density. However, the box-plot of the system shows that all clone types in classes are less fault-density than classes without clones. The summary in *Table 60* has a generic view of the test results with context of all subject systems in this study.

Total:

**Table 58: Mann-Whitney Test Result-Fault-Density Based on Clones Types for Comprehensive System**

Glossary A:	A: Type-I	B: Type-II	C: Type-III	D: Type-I&II	E: Type-I&III	F: Type-II&III	G: Type-I&II&III	H: Has No Clones
cloneTypes	A	B	C	D	E	F	G	H
A								
B	0.145004							
C	<b>0.01959</b>	<b>0.000337</b>						
D	<b>0.013149</b>	0.735931	<b>0</b>					
E	<b>0.010316</b>	0.168976	<b>0.00001</b>	0.109968				
F	0.295857	0.930752	<b>0.006878</b>	0.98264	0.291			
G	0.240978	0.985113	<b>0.001841</b>	0.878487	0.173412	0.800987		
H	<b>0.02011</b>	<b>0.000097</b>	0.554418	<b>0</b>	<b>0.000006</b>	<b>0.011262</b>	<b>0.001034</b>	

Mann-Whitney test shows that there are significant differences among some of the individual groups of data such as Type-1, Type-2, against Type-3, in addition to Type-1, Type-3 against Type-1&2. Similar to Type-1, Type-2, Type-1&2, Type-1&3, Type-2&3 and Type-1&2&3 against no-clones group. This might not lead us to any conclusion about any relationship between clone types and fault-density.



**Figure 37: Box-Plot of Fault-Density Based on Type for Comprehensive System**

From the p-values above, there are significant differences between clones based on types with respect to fault-density. However, the box-plot of the system shows that all clone types in classes have less fault-density than classes without clones. The summary in [Table 60] has a generic view of the test results with the context of all subject systems in this study.

**Table 59: Summary of Mann-Whitney Test for Fault-Density Based on Clone Location**

	Velocity	Synapse	Ant	Xalan	Camel	Total	
A: InterOnly	>	>	<*	>	<*	>	B: IntraOnly
A: InterOnly	>	>	<*	>	>	>	C: Inter&Intra
A: InterOnly	<	<*	<	<	<	<*	D: Has No Clones
B: IntraOnly	<	>	>	<	>	>	C: Inter&Intra
B: IntraOnly	<	<*	<*	<	<*	<*	D: Has No Clones
C: Inter&Intra	<	<	<*	<*	<	<*	D: Has No Clones

In the summary table [Table 59], data groups and their test results that participated in the Mann-Whitney test for clones based on location, were compared by fault-density. So, InterOnly > Has No Clones, means the classes that contain inter-clones are more fault-density than the classes with no-clones. If \* is associated with it, such as IntraOnly >\* Has No Clones, it means classes that contain intra-clones are more fault-density than classes with no-clones and this difference is significant.

From the summary table we observed that only three groups maintained consistent results across all five subject systems, namely InterOnly < no-clones, IntraOnly < no-clones and Inter&Intra < no-clones. All results show they have less fault-density than the classes with no-clones. IntraOnly group against no-clones group show significant differences in three out of five subject systems and also consolidated by the aggregated system sharing the same significance. The Mann-Whitney test results of Inter&Intra against no-clones showed significant differences in two systems in addition to the comprehensive system in total. The box-plots show a general trend across all the subject systems, that classes with no-clones have higher fault-density, regardless of any type location. Similarly, no clear relationship between different clone location groups that is consistent for all subject systems.

We observed that classes that contain intra-clones only have higher fault-density than classes that have hybrid-clones, this is true for all systems except for ant subject system. Nevertheless, the overall trend and behavior of the aggregated system support our interpretation that classes with hybrid-clones would probably have higher fault-density than classes with only intra-clones.

This summary concludes that there is no relationship between clone location and fault-density. So, we accept the null hypothesis HYP-F3.

**Table 60: Summary of Mann-Whitney Test for Fault-Density Based on Clone Types**

	Velocity	Synapse	Ant	Xalan	Camel	Total	
A:Type-I	<	<	<	<	<	<	B:Type-II
A:Type-I	>	>*	>	>	>	>*	C:Type-III
A:Type-I	>	>	>	>*	<	>*	D:Type-I&II
A:Type-I	<	<	<	<*	<	<*	E:Type-I&III
A:Type-I	>	>	<	<*	>	>	F:Type-II&III
A:Type-I	>	>	>	>*	>	>	G:Type-I&II&III
A:Type-I	<	<*	<	<	<	<*	H:Has No Clones
B:Type-II	>	>*	>	>	>	>*	C:Type-III
B:Type-II	>	>*	>	>	>	>	D:Type-I&II
B:Type-II	<	<	>	<*	>	<	E:Type-I&III
B:Type-II	>	>	<	<	>	>	F:Type-II&III
B:Type-II	>	>*	>	>	>	>	G:Type-I&II&III
B:Type-II	<	<*	<	<	<	<*	H:Has No Clones
C:Type-III	>	>	<	>*	<	>*	D:Type-I&II
C:Type-III	<	<	<	<*	<	<*	E:Type-I&III
C:Type-III	>	>	<	<*	<	<*	F:Type-II&III
C:Type-III	>	>	>	<*	>	>*	G:Type-I&II&III
C:Type-III	<	<	<*	<	<	<	H:Has No Clones
D:Type-I&II	<	<	<	<*	>	<	E:Type-I&III
D:Type-I&II	>	>	<	<	>	<	F:Type-II&III
D:Type-I&II	>	>	<	<	>	>	G:Type-I&II&III
D:Type-I&II	<	<	<	<*	<	<*	H:Has No Clones
E:Type-I&III	>	>	<	<	>	>	F:Type-II&III
E:Type-I&III	>	>	>	>	>	>	G:Type-I&II&III
E:Type-I&III	<	<	<*	<*	<	<*	H:Has No Clones
F:Type-II&III	>	>	>	>	>	>	G:Type-I&II&III
F:Type-II&III	<	<	<	<	<	<*	H:Has No Clones
G:Type-I&II&III	<	<	<	<	<	<*	H:Has No Clones

In the summary table [Table 60], data groups and their test results that participated in the Mann-Whitney test for clones based on types, were compared by fault-density. So, Type-1 > Type-2, means the classes that contain Type-1 clones are more fault-density than the classes with Type-2 clones. If \* is associated with it, such as Type-1 >\* Type-2 Clones, it means that classes that contain Type-1 clones have higher fault-density than classes with Type-2 clones and this difference is significant. From the summary table we observed one major finding that is consistent across all the subject systems, that all

classes that have no-clones have higher fault-density than classes that contain clones regardless of type or location. The following data groups have consistent results regarding fault-density in all five subject systems in addition to the aggregated system: Type-I < Type-II, Type-I > Type-III, Type-I < Type-I&III, Type-I > Type-I&2&3, Type-II > Type-III, Type-II > Type-I&II, Type-II > Type-I&II&III. Type-III < Type-I&III, Type-I&III > Type-I&II&III and Type-II&III < Type-I&II&III. Significance of the differences resulted of Mann-Whitney test for fault-density was not dominant within the results as can be seen from the summary table. Only Type-I&III <\* no-clones was significant in two systems out of five but consolidated by the aggregated system that contain all subject systems in one. These findings conclude that there are relationships between clone types and fault-density. So, we accept the null hypothesis HYP-F2.

## 5.5 Univariate Analysis

For the five subject systems, univariate regression models for the fault-density variable were created. The goal of univariate analysis is to find out if code clones metrics are good indicators of fault-density. A summary of the results is shown in tabular format for each subject system. The following approach was followed during univariate analysis for all the systems:

- The univariate models were assessed if they were **statistically significant** for all variables. This was achieved through checking p-value of correlation results.
- **Degree and direction of impact** of independent variables were measured against fault-density, which measure the direction of increase/ decrease of fault-density if metric value was increased/ decreased.

- Univariate models were checked for **goodness-of-fit using two measures**. R Square (R<sup>2</sup>): it is a measure that shows how close/far the variation values of the response data to fitted regression models are. It explains the goodness-of-fit of the model, where a higher value means better goodness-of-fit of the model. The range of values is: 0% - 100%. Adjusted R Square (AR<sup>2</sup>): it is also a statistical measure that adjusts R-squared value relatively to the number of data inputs. Adjusted R-squared is always less than R-square, where a high value of AR<sup>2</sup> means that the model fits well. 0% - 100%. (Note: AR<sup>2</sup> can be negative)
- The **prediction performance** of these univariate models was evaluated. Three statistical measurements were used to describe the prediction performance, namely Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Standard Deviation of Absolute Error (SDAE).

For velocity system, as shown below:

**Table 61: Univariate Analysis for Fault-Density of Velocity System**

	C <sub>0</sub>	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
TCF	14.599	-3.006	0.007	0.003	0.520	20.064	46.891	42.474
CD	14.926	-0.729	0.095	0.009	0.648	19.971	46.917	42.547
CRFL	13.616	-1.493	0.003	-0.002	0.330	20.016	46.920	42.529
NBR	14.239	-4.642	0.005	0.000	0.035	20.001	46.842	42.450
RSA	14.320	-15.788	0.006	0.001	0.054	19.908	46.817	42.466
RSI	14.113	-23.418	0.006	0.001	0.342	19.889	46.833	42.493
CVR	15.393	-19.504	0.011	0.007	0.563	20.045	46.786	42.367
RNR	-20.415	36.348	0.007	0.003	0.652	20.198	46.917	42.439
Inter-clones	14.476	-6.103	0.006	0.001	0.030	19.970	46.825	42.445
Intra-clones	13.661	-2.138	0.003	-0.001	0.138	20.032	46.953	42.558
Type-1	14.281	-5.808	0.005	0.001	0.101	19.941	46.846	42.483
Type-2	13.780	-4.827	0.004	-0.001	0.530	19.852	46.867	42.548
Type-3	13.587	-3.427	0.002	-0.002	0.928	20.048	46.956	42.554

**Independent variables assessment for statistically significance:**

From the above table, NBR and Inter-class showed significant indicators of fault-density.

However, TCF, RSA, CVR, CRFL, CD, RSI, RNR, Intra, Type-1, Type-2 and Type-3 are

non-significant fault- density indicators. Based on these results, the following can be observed:

**Degree and direction of impact of the independent variables to fault-density:**

From the table, the coefficients C1 show the direction of impact whereas the independent variables that are based on clone types and location have negative impact. The remaining clone metrics have also negative impact except for RNR which has positive impact.

**Goodness-of-Fit:**

The following measures are representing the goodness-of-fit of the regression models:

For all independent variables, they have relatively very low scores either for R2 or AR2, all variables have values between 0% - 1% which result in a very poor level as goodness-of-fit of the model.

**Prediction Performance:**

Univariate analysis results for velocity system show that MAE and RMSE values are relatively high, and that they indicate poor prediction performance since they are negatively-oriented scores. The SDAE scored on average  $\approx 42$  for all independent variables which comes relatively in the midrange values when compared with other subject systems.

Synapse system:

**Table 62: Univariate Analysis for Fault-Density of Synapse System**

	C <sub>0</sub>	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
TCF	7.311	-1.058	0.005	0.001	0.011	8.787	33.590	32.484
CD	6.681	-0.128	0.002	-0.002	0.095	8.724	33.634	32.547
CRFL	6.992	-1.767	0.003	-0.001	0.039	8.693	33.615	32.535
NBR	6.656	-0.196	0.002	-0.002	0.001	8.747	33.614	32.520
RSA	6.651	-6.560	0.002	-0.002	0.042	8.724	33.637	32.550
RSI	6.749	-11.980	0.004	0.000	0.458	8.650	33.566	32.496
CVR	7.524	-8.706	0.004	0.000	0.023	8.766	-8.706	32.534
RNR	-24.781	33.352	0.007	0.003	0.331	9.103	33.687	32.497
Inter-clones	7.091	-1.369	0.004	0.000	0.042	8.829	33.607	32.489
Intra-clones	6.207	-0.924	0.002	-0.002	0.760	8.642	33.577	32.510
Type-1	6.204	-1.340	0.001	-0.003	0.022	8.663	33.582	32.509
Type-2	6.272	-1.031	0.001	-0.002	0.000	8.617	33.586	32.525
Type-3	6.723	-1.550	0.003	-0.001	0.356	8.699	33.587	32.504

**Independent variables assessment for statistically significance:**

From the above table, TCF, CRFL, NBR, RSA, CVR, Inter-class, Type-1, and Type-2 showed significant indicators of fault-density whereas RNR, RSI, Type-3, CD, and Intra are non- significant fault-density indicators.

**Degree and direction of impact of independent variables to fault-density:**

From the table, coefficients C1 show direction of impact, independent variables that are based on clone types and location have negative impact. The remaining clone metrics have also negative impact except for RNR which has a positive impact.

**Goodness-of-Fit:**

Similarly, for all independent variables, they have relatively very low scores either for R2 or AR2, all the variables have values between 0% - 1% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

Univariate analysis results for synapse system show that all independent variables scored relatively low values for MAE and RMSE, which concludes that the prediction

performance of synapse system is better than velocity. The values of SDAE for all independent variables are on average  $\approx 32$

Ant system:

**Table 63: Univariate Analysis for Fault-Density of Ant System**

	C <sub>0</sub>	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
TCF	1.589	-0.053	0.000	-0.001	0.000	2.410	7.116	6.700
CD	1.689	-0.039	0.002	0.001	0.020	2.391	7.111	6.702
CRFL	1.613	-0.235	0.002	0.000	0.001	2.396	7.110	6.699
NBR	1.635	-0.104	0.002	0.000	0.018	2.404	7.110	6.696
RSA	1.694	-1.647	0.002	0.000	0.513	2.370	7.108	6.706
RSI	1.570	-1.133	0.000	-0.001	0.000	2.395	7.115	6.704
CVR	1.730	-1.534	0.002	0.001	0.009	2.380	7.109	6.703
RNR	-0.996	2.722	0.001	-0.001	0.849	2.391	7.121	6.712
Inter-clones	1.633	-0.171	0.001	0.000	0.004	2.411	7.113	6.697
Intra-clones	1.531	-0.013	0.000	-0.001	0.000	2.393	7.116	6.706
Type-1	1.563	-0.095	0.000	-0.001	0.000	2.405	7.116	6.701
Type-2	1.566	-0.107	0.000	-0.001	0.003	2.401	7.115	6.702
Type-3	1.544	-0.048	0.000	-0.001	0.000	2.399	7.117	6.705

**Independent variables assessment for statistically significance:**

All variables except RSA and RNR showed significant indicators of fault-density, whereas RSA and RNR showed non-significant fault-density indicators. These results are similar to synapse where RNR is common between the two systems as non-significant fault-density indicator.

**Degree and direction of impact of the independent variables to fault-density:**

From the table, coefficients C1 show the direction of impact, the independent variables that are based on clone types and location have negative impact. The remaining clone metrics have also negative impact except for RNR which has positive impact.

**Goodness-of-Fit:**

Similarly, for all the independent variables, they have extremely low scores either for R2 or AR2, all variables have values around 0% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

Univariate analysis results for ant system show that all independent variables scored relatively the lowest values for MAE and RMSE, which concludes that the prediction performance of ant system is the best among other subject systems. For the values of SDAE for all independent variables are on average  $\approx 6$ .

Xalan system:

**Table 64: Univariate Analysis for Fault-Density of Xalan System**

	C <sub>0</sub>	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
TCF	10.519	-0.918	0.003	0.002	0.000	14.155	59.325	57.644
CD	10.554	-0.369	0.002	0.001	0.087	14.153	59.431	57.754
CRFL	9.612	-0.097	0.001	0.000	0.363	14.112	59.333	57.663
NBR	10.161	-0.237	0.002	0.001	0.034	14.135	59.363	57.688
RSA	11.322	-11.832	0.005	0.003	0.849	14.042	59.249	57.594
RSI	9.842	-18.513	0.001	0.000	0.033	14.039	59.329	57.677
CVR	12.093	-13.616	0.006	0.005	0.681	14.211	59.208	57.510
RNR	-32.192	44.564	0.006	0.005	0.010	14.391	59.412	57.676
Inter-clones	10.660	-2.000	0.006	0.005	0.000	14.266	59.326	57.619
Intra-clones	9.728	-0.826	0.001	0.000	0.009	14.036	59.332	57.681
Type-1	10.216	-1.656	0.002	0.001	0.000	14.171	59.308	57.623
Type-2	9.967	-1.556	0.002	0.001	0.000	14.107	59.319	57.651
Type-3	9.316	-0.716	0.000	-0.001	0.002	14.109	59.389	57.721

**Independent variables assessment for statistical significance:**

From the above table, TCF, RNR, RSI, Type-3, NBR, Inter-class, Intra, Type-1, and Type-2 showed significant indicators of fault-density, whereas RSA, CVR, CD, CRFL are none significant fault-density indicators.

**Degree and direction of impact of the independent variables to fault-density:**

From the table, coefficients C1 show the direction of impact, independent variables that are based on clone types and location have negative impact. Remaining clone metrics have also negative impact except for RNR which has positive impact.

**Goodness-of-Fit:**

Similarly, for all the independent variables, they have extremely low scores either for R2 or AR2, all variables have values around 0% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

Univariate analysis results for xalan system show that all independent variables scored relatively midrange values for MAE and RMSE, which concludes that the prediction performance of xalan system is in the middle among other subject systems. For the values of SDAE for all independent variables are on average  $\approx 57$ .

Camel system:

**Table 65: Univariate Analysis for Fault-Density of Camel System**

	C <sub>0</sub>	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
TCF	18.032	-3.412	0.001	0.000	0.296	28.239	107.536	103.818
CD	17.643	-0.401	0.001	0.000	0.481	28.177	107.572	103.872
CRFL	17.614	-6.344	0.001	0.000	0.304	28.187	107.556	103.852
NBR	18.160	-6.344	0.002	0.001	0.041	28.176	107.507	103.805
RSA	17.967	-22.820	0.002	0.001	0.039	28.116	107.522	103.837
RSI	17.190	-24.927	0.000	-0.001	0.000	28.110	107.582	103.901
CVR	18.577	-24.417	0.002	0.001	0.223	28.039	107.554	103.891
RNR	-97.625	124.375	0.009	0.008	0.864	29.753	107.217	103.061
Inter-clones	17.512	-3.576	0.001	0.000	0.165	28.157	107.563	103.868
Intra-clones	17.173	-3.399	0.001	0.000	0.000	28.075	107.575	103.902
Type-1	17.360	-7.431	0.001	0.000	0.852	28.101	107.566	103.885
Type-2	16.765	-2.092	0.000	-0.001	0.171	28.035	107.599	103.938
Type-3	17.667	-4.343	0.001	0.000	0.812	28.158	107.554	103.858

**Independent variables assessment for statistically significance:**

From the above table, NBR, RSI, RSA, and Intra showed significant indicators of fault-density, whereas TCF, CD, CRFL, CVR, Inter-class, RNR, Type-1, Type-2 and Type-3 are non- significant fault-density indicators.

**Degree and direction of impact of the independent variables to fault-density:**

From the table, coefficients C1 show the direction of impact, independent variables that are based on clone types and location have negative impact. Remaining clone metrics have also negative impact except for RNR which has positive impact.

**Goodness-of-Fit:**

Similarly, for all independent variables, they have extremely low scores either for R2 or AR2, all variables have values around 0% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

Univariate analysis results for camel system show that all independent variables scored relatively the highest values for MAE and RMSE, which concludes that the prediction performance of camel system is in the lowest among other subject systems. For the values of SDAE for all independent variables, they are on average  $\approx 103$  which are the highest among all subject systems.

Comprehensive system, as shown below:

**Table 66: Univariate Analysis for Fault-Density of Comprehensive System**

	C <sub>0</sub>	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
TCF	10.855	-1.229	0.002	0.002	0.000	15.838	69.788	67.979
CD	10.826	-0.330	0.002	0.001	0.000	15.840	69.805	67.995
CRFL	9.775	-0.111	0.000	0.000	0.000	15.705	69.840	68.063
NBR	10.193	-0.301	0.001	0.001	0.000	15.799	69.819	68.019
RSA	11.095	-13.132	0.002	0.002	0.000	15.837	69.776	67.966
RSI	10.343	-19.504	0.001	0.001	0.000	15.747	69.818	68.030
CVR	11.783	-14.955	0.003	0.003	0.000	15.958	69.744	67.905
RNR	-43.851	57.922	0.005	0.004	0.040	16.336	69.720	67.791
Inter-clones	10.883	-2.328	0.002	0.002	0.000	15.866	69.794	67.978
Intra-clones	10.094	-1.078	0.001	0.000	0.000	15.717	69.825	68.044
Type-1	10.341	-2.048	0.001	0.001	0.000	15.768	69.814	68.021
Type-2	10.188	-1.818	0.001	0.001	0.000	15.740	69.823	68.037
Type-3	10.072	-1.491	0.001	0.000	0.008	15.724	69.832	68.050

**Independent variables assessment for statistically significance:**

All independent variables showed significant indicators of fault-density.

**Degree and direction of impact of the independent variables to fault-density:**

From the table, coefficients C1 show the direction of impact, independent variables that are based on clone types and location have negative impact. Remaining clone metrics have also negative impact except for RNR which has positive impact.

**Goodness-of-Fit:**

Similarly, for all the independent variables, they have extremely low scores either for R2 or AR2, all variables have values around 0% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

Univariate analysis results for comprehensive system show that all independent variables scored relatively midrange values for MAE and RMSE, which concludes that the

prediction performance of comprehensive system is in the middle among other subject systems. For the values of SDAE for all independent variables are on average  $\approx 68$ .

### **Summary of Univariate Analysis:**

From the results of the analysis the following findings are concluded to answer the thirteen hypotheses (HYP-G1, HYP-G2, HYP-G3, HYP-G4, HYP-G5, HYP-G6, HYP-G7, HYP-G8, HYP-G9, HYP-G10, HYP-G11, HYP-G12, and HYP-G13):

- 1) If a metric scored  $p\text{-value} < 0.05$  across all the systems, then it is considered a significant indicator of fault-density. Hence, there is a clear evidence to reject the null hypothesis.
  - 2) If a metric scored  $p\text{-value} \geq 0.05$  across all the systems, then it is considered non-significant indicator of fault-density. Hence, there is a clear evidence to accept the null hypothesis.
  - 3) If a metric across systems has inconsistent results, then we don't have clear evidence to accept or to reject a hypothesis.
- TCF, there is no clear evidence to accept the null hypothesis of HYP-G1.
  - CD, there is no clear evidence to accept the null hypothesis of HYP-G2.
  - CRFL, there is no clear evidence to accept the null hypothesis of HYP-G3.
  - NBR, metric is fault-density significant indicator, so HYP-G4 is **rejected**.
  - RSA, there is no clear evidence to accept the null hypothesis of HYP-G5.
  - RSI, there is no clear evidence to accept the null hypothesis of HYP-G6.
  - CVR, there is no clear evidence to accept the null hypothesis of HYP-G7.
  - RNR, there is no clear evidence to accept the null hypothesis of HYP-G8.

- Interclass, there is no clear evidence to accept the null hypothesis of HYP-G9.
- Intra, there is no clear evidence to accept the null hypothesis of HYP-G10.
- Type-1, there is no clear evidence to accept the null hypothesis of HYP-G11.
- Type-2, there is no clear evidence to accept the null hypothesis of HYP-G12.
- Type-3, there is no clear evidence to accept the null hypothesis of HYP-G13.

## 5.6 Multivariate Analysis

In this section, we are trying to answer the questions H, I, and J in 5.1. This section shows the study of the subject systems with multivariate perspective, where Weka software used to generate features selection. The attribute evaluator used is CfsSubsetEval and the search method used is BestFirst with bi-directional setting. After getting these attributes that mostly representative of the data in a subject system, they are incorporated in multivariate analysis to study their features using weka software with default values.

For velocity system:

**Table 67: Univariate Analysis for Fault-Density of Velocity System**

	$C_1$	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
CVR	-16.385	0.014	0.006	0.563	20.111	46.769	42.316
RNR	25.461			0.652			
Intercept	-8.430						

### **Independent variables assessment for statistical significance:**

The features selection yielded two attributes to be selected which are CVR, and RNR, each one of the metrics belongs to different principal component, CVR belongs to PC-1

and RNR belongs to PC-3. Multivariate model produced no better than the scores of any univariate models as all of them are not significant indicators.

**Degree and direction of impact of the independent variables to fault-density:**

From the table, coefficients C1 show the direction of impact, the independent variable CVR has negative impact. However, RNR has positive impact which means higher RNR value in a class yields higher fault-density.

**Goodness-of-Fit:**

Similar to univariate models of velocity system, all the independent variables, have relatively very low scores either for R2 or AR2, all the variables have values between 0% - 1% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

The multivariate analysis results for velocity system show that both independent variables scored relatively low values for MAE and RMSE, which concludes that the prediction performance of velocity system is relatively poor. The values of SDAE for all independent variables are on average  $\approx 42$ .

For synapse system:

**Table 68: Univariate Analysis for Fault-Density of Synapse System**

	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
TCF	-1.052	0.012	0.004	0.011	9.210	33.682	32.462
RNR	33.199			0.331			
Intercept	-23.089						

**Independent variables assessment for statistically significance:**

The features selection yielded two attributes to be selected which are TCF, and RNR. Each one of the metrics belongs to different principal component, TCF belongs to PC-2 and RNR belongs to PC-3. The multivariate model produced no better than the scores of any univariate models in addition to RNR is not significant indicator.

**Degree and direction of impact of the independent variables to fault-density:**

Both TCF and RNR maintained the same effects on the direction of impact as in univariate models where TCF is negative and RNR is positively affecting fault-density.

**Goodness-of-Fit:**

Similar to univariate models of synapse system, they have relatively very low scores either for R<sup>2</sup> or AR<sup>2</sup>, all values are between 0% - 1% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

The multivariate analysis results for synapse system show that the model scored relatively low values for MAE and RMSE, which concludes that the prediction performance of synapse system is relatively poor. For the values of SDAE for all independent variables are on average  $\approx 32$ .

For ant system:

**Table 69: Univariate Analysis for Fault-Density of Ant System**

	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
RSA	-1.530	0.003	0.000	0.513	2.373	7.115	6.712
RNR	1.724			0.849			
Intercept	0.087						

**Independent variables assessment for statistically significance:**

The features selection yielded two attributes to be selected which are RSA, and RNR, both metrics belong to the same principal component PC-3. The multivariate model produced no better than the scores of any univariate models as both are not significant indicators.

**Degree and direction of impact of the independent variables to fault-density:**

From the table, coefficients C1 show the direction of impact, the independent variable RSA has negative impact. However, RNR has positive impact.

**Goodness-of-Fit:**

Similar to univariate models of ant system, they have relatively very low scores either for R2 or AR2, all values are between 0% - 1% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

The multivariate analysis results for ant system show that the model scored relatively low values for MAE and RMSE, which concludes that the prediction performance of ant system is relatively poor although they are the best among the other subject systems. For the values of SDAE for all independent variables are on average  $\approx 6$ .

For xalan system:

**Table 70: Univariate Analysis for Fault-Density of Xalan System**

	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
RSI	-3.552	0.009	0.006	0.033	14.451	59.175	57.416
CVR	-12.527			0.681			
RNR	38.153			0.010			
Intercept	-23.349						

**Independent variables assessment for statistically significance:**

The features selection yielded three attributes to be selected which are RSI, CVR, and RNR. Each one of the metrics belongs to different principal component, RSI belongs to PC-1, CVR belongs to PC-2 and RNR belongs to PC-4. The multivariate model produced no better than the scores of any univariate models in addition to CVR is not a significant indicator.

**Degree and direction of impact of the independent variables to fault-density:**

RNR still positively affects fault-density, whereas CVR and RSI still negatively affect fault-density.

**Goodness-of-Fit:**

Similar to univariate models of xalan system, they have relatively very low scores either for R<sup>2</sup> or AR<sup>2</sup>, all values are between 0% - 1% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

The multivariate analysis results for xalan system show that the model scored relatively low values for MAE and RMSE, which concludes that the prediction performance of xalan system is relatively poor. For the values of SDAE for all independent variables are on average  $\approx 57$ .

For camel system:

**Table 71: Univariate Analysis for Fault-Density of Camel System**

	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
NBR	-6.530	0.011	0.009	0.041	29.900	107.120	102.917
RNR	125.151			0.864			
Intercept	-96.746						

**Independent variables assessment for statistically significance:**

The features selection yielded two attributes to be selected which are NBR, and RNR. Each one of the metrics belongs to different principal component, NBR belongs to PC-1 and RNR belongs to PC-2. The multivariate model produced no better than the scores of any univariate models in addition to RNR is not significant indicator.

**Degree and direction of impact of the independent variables to fault- density:**

From the table, the coefficients C<sub>1</sub> shows the direction of impact, the independent variable NBR has negative impact. However, RNR has positive impact.

**Goodness-of-Fit:**

Similar to univariate models of camel system, they have relatively very low scores either for R<sup>2</sup> or AR<sup>2</sup>, all values are between 0% - 1% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

The multivariate analysis results for camel system show that the model scored relatively low values for MAE and RMSE, which concludes that the prediction performance of camel system is relatively poor. For the values of SDAE for all independent variables are on average  $\approx 102$ .

For comprehensive collection:

**Table 72: Univariate Analysis for Fault-Density of Comprehensive System**

	C <sub>1</sub>	R Square	Adjusted R Square	p-value	Mean absolute error	Root mean squared error	St.Dev absolute error
CVR	-12.686	0.007	0.006	0.000	16.422	69.644	67.691
RNR	51.781			0.040			
Intercept	-36.317						

**Independent variables assessment for statistically significance:**

The features selection yielded two attributes to be selected which are CVR, and RNR.

The multivariate model produced no better than the scores of any univariate models.

**Degree and direction of impact of the independent variables to fault- density:**

From the table, the coefficients C<sub>1</sub> shows the direction of impact, the independent variable CVR has negative impact. However, RNR has positive impact.

**Goodness-of-Fit:**

Similar to univariate models of comprehensive system, they have relatively very low scores either for R<sup>2</sup> or AR<sup>2</sup>, all values are between 0% - 1% which conclude very weak goodness-of-fit of the models.

**Prediction Performance:**

Multivariate analysis results for camel system show that the model scored relatively low values for MAE and RMSE, which concludes that the prediction performance of camel system is relatively poor. For the values of SDAE for all independent variables are on average  $\approx 67$ .

### 5.6.1 Validation of Fault-density Prediction Model

A cross-validation has been used to minimize the chance of getting an ideal goodness-of-fit or exact match of the explanatory model with high accuracy, as this is considered to be a threat to validity of the model. Hence, all the five subject systems are combined in one data set, and each time one system will be left out to build the model and tested against the system that was left out. The approach of using leave-one-out in building the model will help in generalizing the results and moving away from being biased to the small data set.

**Table 73: Comparison of Multivariate Analysis - Training data Vs. Cross-validation**

	Mean absolute error		Root mean squared error		Stdev. absolute error	
	Training Data	Cross-V	Training Data	Cross-V	Training Data	Cross-V
velocity	20.111	<b>16.094</b>	<b>46.769</b>	71.180	<b>42.316</b>	69.349
synapse	<b>9.210</b>	17.080	<b>33.682</b>	72.037	<b>32.462</b>	69.996
ant	<b>2.373</b>	20.637	<b>7.115</b>	79.797	<b>6.712</b>	77.099
xalan	<b>14.451</b>	17.202	<b>59.175</b>	73.544	<b>57.416</b>	71.520
camel	29.900	<b>10.379</b>	107.120	<b>43.200</b>	102.917	<b>41.944</b>
Avg.	<b>15.209</b>	16.278	<b>50.772</b>	67.952	<b>48.365</b>	65.982

Keys: Numbers in **Bold** indicate the best value among others.

**Training Data:** model is built over data training being part of the same data set.

**Cross-V:** model is built over data set based on cross-validation leave-one-out approach.

The above table shows the comparisons between the results of the multivariate models when the training data used within the same system and when cross validation leave-one-out is used. In this section, three measures were included in the comparison, namely Mean absolute error (MAE), Root mean squared error (RMSE), and Standard deviation of absolute error (SDAE). In general, the overall average was for the advantage of the multivariate models that the training data was part of the same system. The individual results were fluctuating from one to another, for example for MAE, xalan's value is 14.451 whereas the value when used cross-validation is 17.202 which considered relatively small difference. However, for ant system, MAE is 2.373 while 20.637 for

cross-validation model. Similarly for RMSE, the differences are not consistent between the models of both types.

To sum up, the results out of this comparison give the impression that the data used to build univariate models is dependent on the training set. Hence, it yielded better prediction accuracy.

### 5.6.2 Comparisons of Code Clones Metrics and C&K Metrics in Predicting Fault-density

In this section, comparisons between code clone metrics suite, C&K suite, and both combined in a single suite. All suites are analyzed from different perspectives, namely goodness-of-fit, Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Standard Deviation of Absolute Error (SDAE). In general, the results show that the combined suite of clone metrics and C&K and C&K individually have been interchangeably better scores in all four perspectives than clone metrics suite individually.

Starting with goodness-of-fit, the following table shows the scores for all the three suites against the five subject systems:

**Table 74: Comparison of Multivariate Analysis - Goodness-of-fit**

	R Square			Adjusted R Square		
	Both	CLN	C&K	R Square	CLN	C&K
Velocity	<b>0.026</b>	0.007	0.020	<b>0.024</b>	0.006	0.019
Synapse	<b>0.027</b>	0.007	0.008	<b>0.025</b>	0.006	0.007
Ant	<b>0.038</b>	0.009	0.028	<b>0.035</b>	0.008	0.027
xalan	<b>0.034</b>	0.005	0.030	<b>0.032</b>	0.005	0.029
camel	<b>0.012</b>	0.006	<b>0.012</b>	0.010	0.005	<b>0.011</b>
Avg.	<b>0.027</b>	0.007	0.019	<b>0.025</b>	0.006	0.019

Keys: Numbers in **Bold** indicate the best value among others.

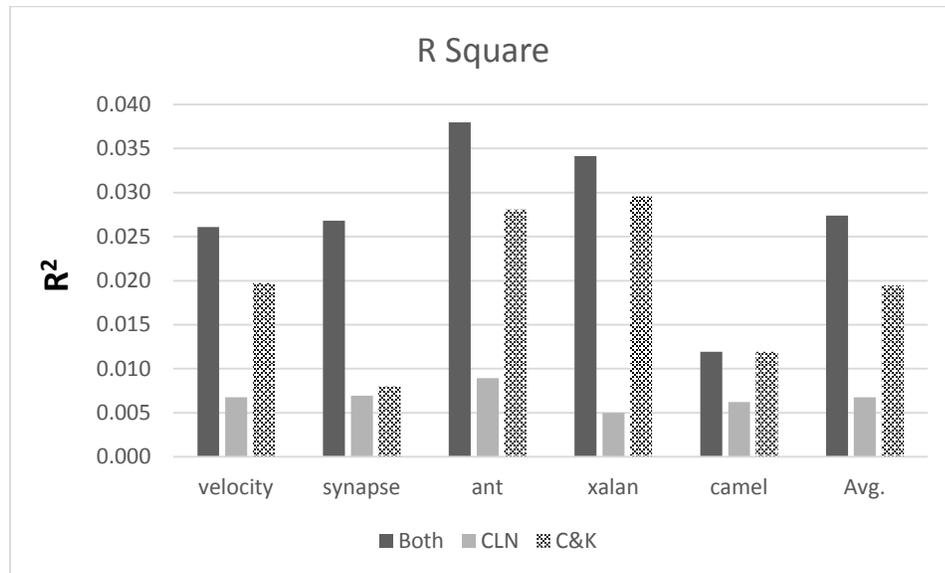
**Both:** model is built over some selected attributes of both the clone and C&K metrics

**CLN:** model is built over some selected attributes of the clone metrics only.

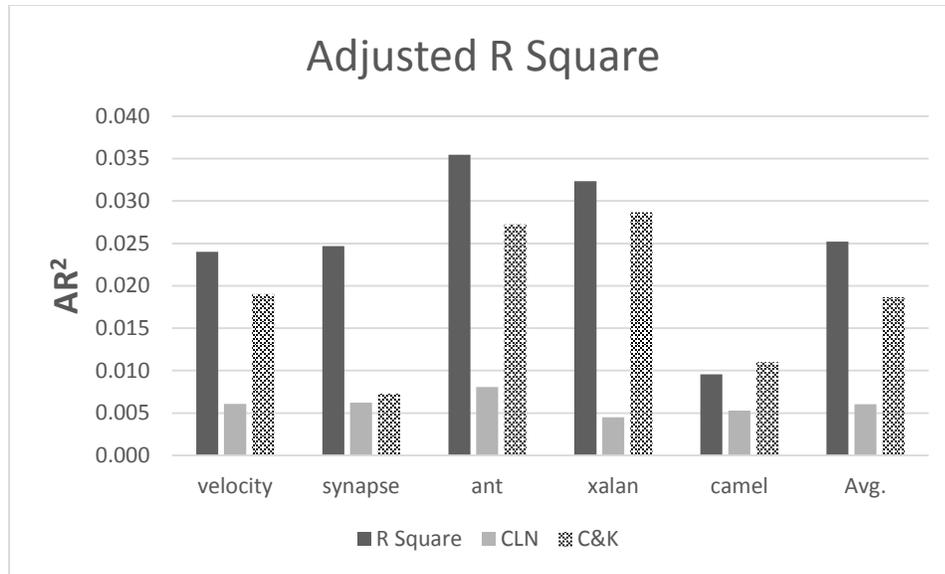
**C&K:** model is built over some selected attributes of the C&K metrics only.

From the results of the comparisons between the three suites, the difference between the clone metrics suite and C&K suite is noticeable in both measurements, namely R-square and Adjusted R-square. Similarly, the difference between clone metrics and the combined suite is also drastic. For R-square the combined suite was the highest value among other suites, similar for Adjusted R-square.

In general, all the results of the three suites are still relatively very low and this represents weak goodness-of-fit. Below are the histograms for both measurements R-square and adjusted R-square:



**Figure 38: Goodness-of-fit - R-Square**



**Figure 39: Goodness-of-fit - Adjusted R-Square**

For prediction performance, the analysis took into consideration three measurements, namely MAE, RMSE, and SDAE. According to Willmott [61], RMSE is ambiguous and could be misleading indicator. On the other hand, RMSE according to Chai [62], is good and gives non-ambiguous readings if data is normally distributed, which is not in our case. Many researchers have adopted Willmott’s approach [63-65]. So, the primary measure that will be used in our study is MAE, whereas RMSE and SDAE will be used as support measurements to MAE. It is clear that there is no specific pattern for the prediction performance behavior, although on average basis, C&K metrics suite gained the best results. The following table has further information about the comparison analysis:

**Table 75: Comparison of Multivariate Analysis -Prediction Performance**

	Mean absolute error			Root mean squared error			Stdev. absolute error		
	Both	CLN	C&K	Both	CLN	C&K	Both	CLN	C&K
velocity	19.002	<b>16.094</b>	18.125	<b>46.331</b>	71.180	46.744	<b>42.348</b>	69.349	43.182
synapse	<b>13.786</b>	17.080	14.307	<b>33.901</b>	72.037	34.134	6.926	69.996	<b>4.197</b>
ant	12.889	20.637	<b>10.819</b>	23.002	79.797	<b>20.772</b>	19.064	77.099	<b>17.744</b>
xalan	15.492	17.202	<b>15.093</b>	<b>59.867</b>	73.544	60.116	<b>57.861</b>	71.520	58.224
camel	22.058	<b>10.379</b>	22.187	107.521	<b>43.200</b>	107.549	105.290	<b>41.944</b>	105.292
Avg.	16.645	16.278	<b>16.106</b>	54.124	67.952	<b>53.863</b>	46.298	65.982	<b>45.728</b>

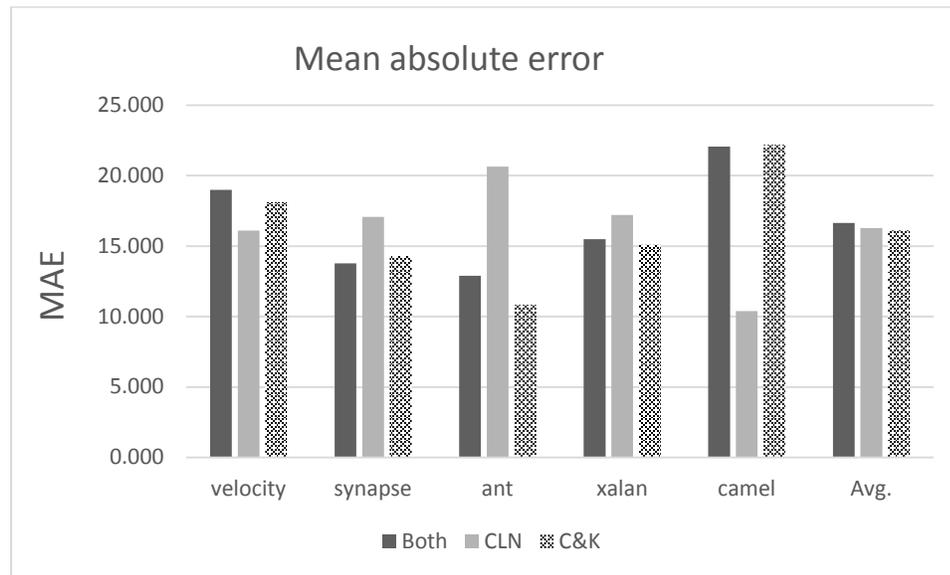
Keys: Numbers in **Bold** indicate the best value among others.

**Both:** model is built over some selected attributes of both the clone and C&K metrics

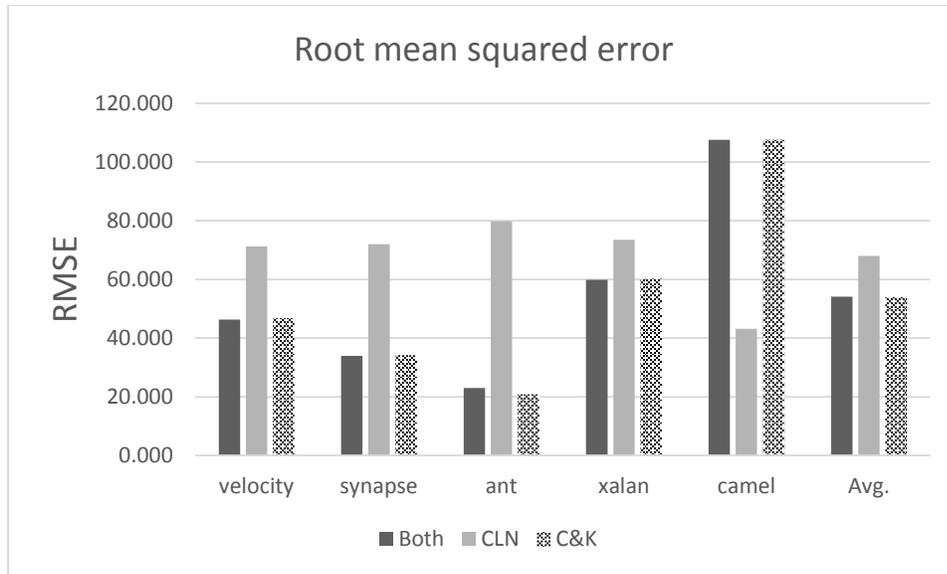
**CLN:** model is built over some selected attributes of the clone metrics only.

**C&K:** model is built over some selected attributes of the C&K metrics only.

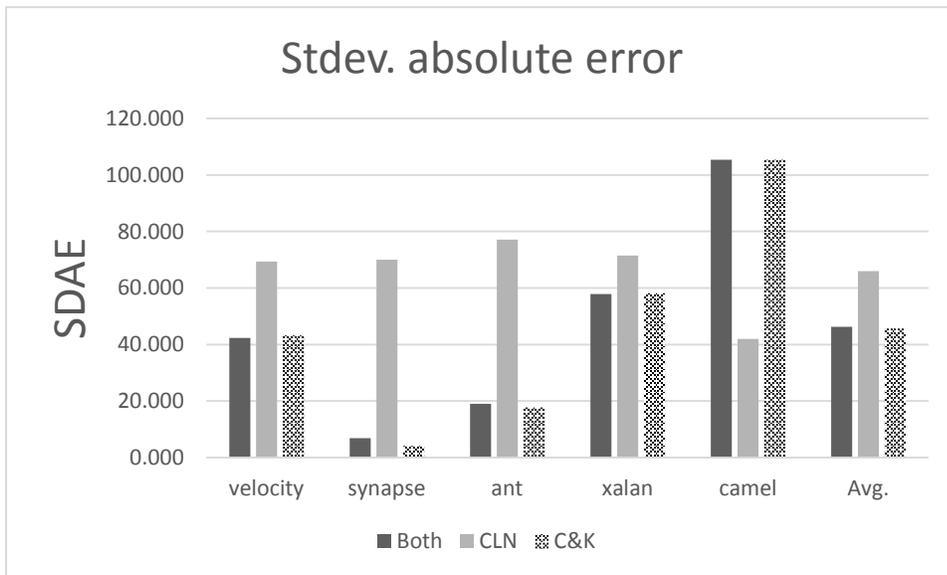
The following are the histograms that describe the results in the above table:



**Figure 40: Prediction Performance - MAE**



**Figure 41: Prediction Performance - RMSE**



**Figure 42: Prediction Performance - SDAE**

To sum up, the three measures of prediction performance showed that C&K metrics suite performed better than clone metrics suite in predicting fault-density. For MAE, C&K suite performed better in three subject systems when compared with clone metrics suite, in addition, the average is better for C&K as well. For RMSE and SDAE, C&K models were better in four out of five subject systems. Hence, we accept the null hypothesis

HYP-H. For the comparison between the multivariate models using combined suite versus the ones using clone metrics suite, there is no clear trend over the behavior of prediction performance based on the MAE results. Although in two systems out of the five systems, the performance of clone metrics suite was better and three systems is the opposite, but the average of MAE was for the sake of the models that are based on clone metrics suite. However, the results of RMSE and SDAE support the decision of considering the combined metrics suite models are performing better than the ones based on clone metrics, as for both RMSE and SDAE the combined suite is performing better in prediction in four out of five systems in addition to the average is in favor of the combined suite. Hence, we reject the null hypothesis HYP-I. The last part of comparing the multivariate models based on different metrics suites, is between the combined metrics suite versus C&K metrics suite. The RMSE and SDAE had similar results in four out of five systems, the C&K metrics suite based models were performing better in predicting fault-density in addition the overall average. However, MAE had two out five systems plus the average for the sake of C&K suite based models versus the remaining three systems where the MAE readings were better for the interest of the combined suite based models. Since the differences are very marginal in addition to supporting results from RMSE and SDAE readings, we accept the null hypothesis of HYP-J.

### **5.6.3 Comparisons of Regression models of Code Clones Metrics and Neural Network Models in Predicting Fault-density**

The Neural Network (NN) models that are used in this study are Multi-Layer Perceptron (MLP) models. MLP is an artificial neural network of type feed-forward that use back-propagation as supervised learning technique [54]. The results of linear regressions are compared with MLP models results. After generating the MLP models with code clones

metrics as inputs, better performance was noticed for the advantage of linear regression models in all three measurements MAE, RMSE, and SDAE as can be seen in the following table:

**Table 76: Comparison of Multivariate Analysis -Prediction Performance versus ANN-MLP**

	Mean absolute error		Root mean squared error		Stdev. absolute error	
	Regression	NN:MLP	Regression	NN:MLP	Regression	NN:MLP
velocity	<b>16.094</b>	31.5093	71.180	<b>64.5597</b>	69.349	<b>56.47164493</b>
synapse	17.080	<b>8.6745</b>	72.037	<b>33.402</b>	69.996	<b>32.3191725</b>
ant	20.637	<b>3.0256</b>	79.797	<b>7.6706</b>	77.099	<b>7.053487107</b>
xalan	<b>17.202</b>	19.0186	73.544	<b>61.225</b>	71.520	<b>58.22945684</b>
camel	<b>10.379</b>	23.506	<b>43.200</b>	108.4614	<b>41.944</b>	105.940306
Avg.	<b>16.278</b>	17.147	67.952	<b>55.064</b>	65.982	<b>52.003</b>

There are contradicting readings out of the general behavior of the prediction performance between the MLP models based and models based on clone metrics suite. MAE suggests that regression models performing better than MLP models, whereas RMSE and SDAE suggests the opposite. Hence, we don't have clear evidence to decide which is better in predicting fault-density.

## 5.7 Threats to validity

Threats to validity are discussed in (Sec.4.8) from four perspectives, namely threats to construct validity, threats to internal validity, threats to statistical conclusion validity, and threats to external validity.

## CHAPTER 6 CONCLUSION AND FUTURE WORK

Our study of code clones with respect to functional correctness of object-oriented classes yielded the following findings, three clone metrics are found to be good indicators of fault-proneness, namely TCF, the percentage of cloned LOC in class (CRFL), and coverage (CVR). On the other hand, we found only one clone metric to be a good indicator of fault-density which is neighbor (NBR).

For the software developer, the following general suggestions are passed with regard to incorporating code cloning in software development. Minimize the use of code clones when developing software, since the relationships that were found with fault-proneness of object-oriented classes. Detecting code clones and categorize them based on types will assist in identifying potential fault-proneness and fault-density in the product. When using code clones, it is advisable to test any code before it is been cloned and used somewhere else. Also, tracking the clones chains while development cycle, could minimize efforts to maintain and debug the product before/after release. Classes that contain Type-2 clones increase the probability of being of being fault-proneness and increase level of fault-density when compared with other types (Type-1, Type-2) individually. Similarly, a high risk of increasing fault-proneness and fault-density when incorporating Type-1 and Type-3 in the same class.

### Research Contributions

The following are the contribution of this research:

- *Surveyed the literature for existing clone related metrics and proposed a suite of metrics that we expect it helps in studying the impact of functional correctness of object-oriented classes.*

- *Compared the classification performance of three models built based on the three suites, namely clone metrics suite, C&K metrics suite and the combined clone and C&K metrics suites in studying fault-proneness.*
- *Compared the prediction performance of three models built based on the three suites, namely clone metrics suite, C&K metrics suite and the combined clone and C&K metrics suites in studying fault-density.*
- *Built logistic-regression models for three suites of metrics, namely software clone metrics, C&K metrics, and combined suite of both clone and C&K metrics to study impact of software clones on fault-proneness of object-oriented classes.*
- *Built linear-regression models for three suites of metrics, namely software clone metrics, C&K metrics, and combined suite of both clone and C&K metrics to study impact of software clones on fault-density of object-oriented classes.*

There was an unexpected general observation over both empirical studies, although a system that has higher fault-proneness in the cloned classes, it doesn't imply that the cloned classes have higher fault-density. After investigating the subject systems, we found out that the average LOC of the cloned classes are much higher than the average LOC of non-cloned classes, which in turn affected fault-density as LOC is an attribute of calculating its value.

## **6.2 Future Work**

The following suggestions are regarded to future researches that might be an extension of this thesis:

- Explore and study other quality attributes using code clones such as change-proneness and effort estimation.
- Explore and study none object-oriented systems for functional correctness using clone metrics.
- Explore and study other software domains and systems that are written in other languages.
- Use other detection tools could also have different results than what we have concluded.
- Explore relationships between clone metrics and test-case adequacy prediction.

## REFERENCES

- [1] R. Koschke, *et al.*, "Clone detection using abstract syntax suffix trees," in *Reverse Engineering, 2006. WCRE'06. 13th Working Conference on*, 2006, pp. 253-262.
- [2] C. K. Roy and J. R. Cordy, "A survey on software clone detection research," Citeseer2007.
- [3] J. Krinke, "Identifying similar code with program dependence graphs," in *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*, 2001, pp. 301-309.
- [4] R. Komondoor and S. Horwitz, "Effective, automatic procedure extraction," in *Program Comprehension, 2003. 11th IEEE International Workshop on*, 2003, pp. 33-42.
- [5] A. Monden, *et al.*, "Software quality analysis by code clones in industrial legacy software," in *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, 2002, pp. 87-94.
- [6] G. M. Selim, *et al.*, "Studying the impact of clones on software defects," in *Reverse Engineering (WCRE), 2010 17th Working Conference on*, 2010, pp. 13-21.
- [7] M. Mondal, *et al.*, "An empirical study of the impacts of clones in software maintenance," in *Program Comprehension (ICPC), 2011 IEEE 19th International Conference on*, 2011, pp. 242-245.
- [8] D. Kozlov, *et al.*, "Exploratory analysis of the relations between code cloning and open source software quality," in *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, 2010, pp. 358-363.
- [9] C. Kapser and M. W. Godfrey, "" Cloning considered harmful" considered harmful," in *Reverse Engineering, 2006. WCRE'06. 13th Working Conference on*, 2006, pp. 19-28.
- [10] A. Lozano and M. Wermelinger, "Assessing the effect of clones on changeability," in *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, 2008, pp. 227-236.
- [11] J. Krinke, "Is cloned code more stable than non-cloned code?," in *Source Code Analysis and Manipulation, 2008 Eighth IEEE International Working Conference on*, 2008, pp. 57-66.
- [12] J. Mayrand, *et al.*, "Experiment on the automatic detection of function clones in a software system using metrics," in *Software Maintenance 1996, Proceedings., International Conference on*, 1996, pp. 244-253.
- [13] T. Kamiya, *et al.*, "CCFinder: a multilinguistic token-based code clone detection system for large scale source code," *IEEE Trans. Softw. Eng.*, vol. 28, pp. 654-670, 2002.
- [14] ISO/IEC, "Systems and software engineering Systems and software Quality Requirements and Evaluation (SQuARE) System and software quality models," ed, 2010.
- [15] S. Bellon, *et al.*, "Comparison and evaluation of clone detection tools," *Software Engineering, IEEE Transactions on*, vol. 33, pp. 577-591, 2007.
- [16] F. J. van der Linden, *et al.*, *Software product lines in action*: Springer, 2007.
- [17] K. E. Wiegers, *Software requirements*: O'Reilly Media, Inc., 2009.
- [18] A. Rawashdeh and B. Matalkah, "A New Software Quality Model for Evaluating COTS Components," *Journal of Computer Science*, vol. 2, 2006.
- [19] J. M. Beaver, *et al.*, "Predicting software suitability using a bayesian belief network," in *Machine Learning and Applications, 2005. Proceedings. Fourth International Conference on*, 2005, p. 7 pp.
- [20] T. Kroeger and N. Davidson, "A Perspective-Based Model of Quality for Software Engineering Processes," in *Software Engineering Conference, 2009. ASWEC '09. Australian*, 2009, pp. 152-161.

- [21] A. Cechich and M. Piattini, "Early detection of COTS component functional suitability," *Information and Software Technology*, vol. 49, pp. 108-121, 2007.
- [22] A. Cechich and M. Piattini, "Balancing stakeholder's preferences on measuring COTS component functional suitability," in *Enterprise Information Systems VI*, ed: Springer, 2006, pp. 177-184.
- [23] L. Jiang, *et al.*, "Deckard: Scalable and accurate tree-based detection of code clones," in *Proceedings of the 29th international conference on Software Engineering*, 2007, pp. 96-105.
- [24] Z. Li, *et al.*, "CP-Miner: Finding copy-paste and related bugs in large-scale software code," *Software Engineering, IEEE Transactions on*, vol. 32, pp. 176-192, 2006.
- [25] N. Bettenburg, *et al.*, "An empirical study on inconsistent changes to code clones at the release level," *Science of Computer Programming*, vol. 77, pp. 760-776, 2012.
- [26] N. Gode and J. Harder, "Clone stability," in *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*, 2011, pp. 65-74.
- [27] F. Rahman, *et al.*, "Clones: What is that smell?," in *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, 2010, pp. 72-81.
- [28] S. Baba, *et al.*, "Application of code clone information to fault-prone module prediction," ed: IEICE-INST ELECTRONICS INFORMATION COMMUNICATIONS ENG KIKAI-SHINKO-KAIKAN BLDG MINATO-KU SHIBAKOEN 3 CHOME, TOKYO, 105, JAPAN, 2008, pp. 2542-2542.
- [29] Y. Kamei, *et al.*, "An Empirical Study of Fault Prediction with Code Clone Metrics," in *Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 2011, pp. 55-61.
- [30] K. Aggarwal, *et al.*, "Investigating the effect of coupling metrics on fault proneness in object-oriented systems," *Software Quality Professional*, vol. 8, p. 4, 2006.
- [31] B. S. Baker, "On finding duplication and near-duplication in large software systems," in *Reverse Engineering, 1995., Proceedings of 2nd Working Conference on*, 1995, pp. 86-95.
- [32] I. D. Baxter, *et al.*, "Clone detection using abstract syntax trees," in *Software Maintenance, 1998. Proceedings., International Conference on*, 1998, pp. 368-377.
- [33] J. H. Johnson, "Visualizing textual redundancy in legacy source," in *Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research*, 1994, p. 32.
- [34] M. Kim, *et al.*, "An ethnographic study of copy and paste programming practices in OOPL," in *Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on*, 2004, pp. 83-92.
- [35] F. V. Rysselberghe and S. Demeyer, "Evaluating clone detection techniques from a refactoring perspective," in *Proceedings of the 19th IEEE international conference on Automated software engineering*, 2004, pp. 336-339.
- [36] G. Antoniol, *et al.*, "Analyzing cloning evolution in the linux kernel," *Information and Software Technology*, vol. 44, pp. 755-765, 2002.
- [37] R. Geiger, "Evolution Impact of Code Clones," *Evolution*, 2005.
- [38] R. Geiger, *et al.*, "Relation of code clones and change couplings," in *Fundamental Approaches to Software Engineering*, ed: Springer, 2006, pp. 411-425.
- [39] J. Krinke, "Is cloned code older than non-cloned code?," in *Proceedings of the 5th International Workshop on Software Clones*, 2011, pp. 28-33.
- [40] E. Juergens, *et al.*, "Do code clones matter?," in *Proceedings of the 31st International Conference on Software Engineering*, 2009, pp. 485-495.

- [41] E. Juergens, *et al.*, "Can clone detection support quality assessments of requirements specifications?," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, 2010, pp. 79-88.
- [42] M. Balazinska, *et al.*, "Measuring clone based reengineering opportunities," in *Software Metrics Symposium, 1999. Proceedings. Sixth International*, 1999, pp. 292-303.
- [43] N. Davey, *et al.*, "The development of a software clone detector," *International Journal of Applied Software Technology*, 1995.
- [44] K. Kontogiannis, "Evaluation experiments on the detection of programming patterns using software metrics," in *Reverse Engineering, 1997. Proceedings of the Fourth Working Conference on*, 1997, pp. 44-54.
- [45] C. Kapsner and M. W. Godfrey, "Aiding comprehension of cloning through categorization," in *Software Evolution, 2004. Proceedings. 7th International Workshop on Principles of*, 2004, pp. 85-94.
- [46] R. Fanta and V. Rajlich, "Removing clones from the code," *Journal of Software Maintenance*, vol. 11, pp. 223-243, 1999.
- [47] G. G. Koni-N'Sapu, "A scenario based approach for refactoring duplicated code in object oriented systems," *Master's thesis, University of Bern*, 2001.
- [48] E. Burd and J. Bailey, "Evaluating clone detection tools for use during preventative maintenance," in *Source Code Analysis and Manipulation, 2002. Proceedings. Second IEEE International Workshop on*, 2002, pp. 36-43.
- [49] C. Roy, "Detection and Analysis of\\ Detection and Analysis of Near-Miss Software Clones," 2009.
- [50] (Feb 2014). *Tutorial of GUI front-end GemX*. Available: <http://www.ccfinder.net/doc/10.2/en/tutorial-gemx.html>
- [51] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," presented at the Proceedings of the 6th International Conference on Predictive Models in Software Engineering, Timișoara, Romania, 2010.
- [52] S. Lessmann, *et al.*, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *Software Engineering, IEEE Transactions on*, vol. 34, pp. 485-496, 2008.
- [53] M. Asaduzzaman, *et al.*, "VisCad: flexible code clone analysis support for NiCad," presented at the Proceedings of the 5th International Workshop on Software Clones, Waikiki, Honolulu, HI, USA, 2011.
- [54] M. Kim, *et al.*, "An empirical study of code clone genealogies," in *ACM SIGSOFT Software Engineering Notes*, 2005, pp. 187-196.
- [55] I. Jolliffe, *Principal component analysis*: Wiley Online Library, 2005.
- [56] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, pp. 433-459, 2010.
- [57] N. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models," *The Journal of Machine Learning Research*, vol. 6, pp. 1783-1816, 2005.
- [58] H. Zou, *et al.*, "Sparse principal component analysis," *Journal of computational and graphical statistics*, vol. 15, pp. 265-286, 2006.
- [59] E. J. Candès, *et al.*, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, p. 11, 2011.
- [60] H. P. Wijnand and R. van de Velde, "Mann-Whitney/Wilcoxon's nonparametric cumulative probability distribution," *Computer Methods and Programs in Biomedicine*, vol. 63, pp. 21-28, 2000.

- [61] H. M. Park, "Univariate analysis and normality test using SAS, Stata, and SPSS," *The University Information Technology Services (UITs) Center for Statistical and Mathematical Computing, Indiana University*, 2008.
- [62] D. W. Hosmer Jr and S. Lemeshow, *Applied logistic regression*: John Wiley & Sons, 2004.
- [63] J. F. Hair, "Multivariate data analysis," 2009.
- [64] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2 ed ed.: Prentice Hall, 1998.
- [65] W. R. Shadish, *et al.*, *Experimental and quasi-experimental designs for generalized causal inference*: Wadsworth Cengage learning, 2002.
- [66] J. J. Baroudi and W. J. Orlikowski, "The problem of statistical power in MIS research," *MIS Quarterly*, pp. 87-106, 1989.
- [67] K. R. Murphy, *et al.*, *Statistical power analysis: A simple and general model for traditional and modern hypothesis tests*: Routledge, 2009.
- [68] L. V. Hedges and C. Rhoads, "Statistical Power Analysis," in *International Encyclopedia of Education (Third Edition)*, P. Peterson, *et al.*, Eds., ed Oxford: Elsevier, 2010, pp. 436-443.
- [69] L. Adelman, "Experiments, quasi-experiments, and case studies: A review of empirical methods for evaluating decision support systems," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 21, pp. 293-301, 1991.
- [70] H. C. Kraemer and S. Thiemann, *How many subjects?: Statistical power analysis in research*: Sage, 1987.
- [71] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, p. 79, 2005.
- [72] T. Chai and R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?," *Geoscientific Model Development Discussions*, vol. 7, pp. 1525-1534, 2014.
- [73] M. Taylor, *et al.*, "On the sensitivity of field reconstruction and prediction using Empirical Orthogonal Functions derived from gappy data," *Journal of Climate*, vol. 26, pp. 9194-9205, 2013.
- [74] A. Chatterjee, *et al.*, "Background error covariance estimation for atmospheric CO2 data assimilation," *Journal of Geophysical Research: Atmospheres*, vol. 118, pp. 10,140-10,154, 2013.
- [75] S. Jerez, *et al.*, "A multi-physics ensemble of present-day climate regional simulations over the Iberian Peninsula," *Climate dynamics*, vol. 40, pp. 3023-3046, 2013.

## Appendix

Principal components analysis:

<b>Rotated Component Matrix<sup>a</sup></b>			
velocity	Component		
	1	2	3
Totalcinefr agments	.348	.926	.074
Clone100 Oloc	.744	.397	.168
CRFL	.227	-.073	.819
NBR	.899	.010	-.038
RSA	.928	-.004	.050
RSI	-.037	.395	.846
CVR	.761	.217	.521
RNR	.063	-.045	-.765
Interclass	.910	.070	-.034
Intra	-.064	.980	.098
TypeI	.567	.474	-.028
TypeII	.150	.690	.191
TypeIII	.057	.798	.002

Extraction Method: Principal Component Analysis.  
Rotation Method: Varimax with Kaiser Normalization.

**Rotated Component Matrix<sup>a</sup>**

synapse	Component		
	1	2	3
Totalcinefr agments	.465	.848	.196
Clone100 Oloc	.868	.259	.129
CRFL	.888	.103	.194
NBR	.721	.158	-.165
RSA	.941	.025	.027
RSI	.036	.305	.874
CVR	.788	.164	.506
RNR	-.016	.198	-.789
Interclass	.776	.397	-.036
Intra	-.103	.861	.330
Typel	.157	.630	-.124
Typell	.285	.739	-.002
Typelll	.422	.289	.446

Extraction Method: Principal Component Analysis.  
Rotation Method: Varimax with Kaiser Normalization.

**Rotated Component Matrix<sup>a</sup>**

ant	Component		
	1	2	3
Totalcinefr agments	.849	.496	.101
Clone100 Oloc	.212	.304	.776
CRFL	.093	.652	.470
NBR	.101	.840	.223
RSA	-.082	.444	.834
RSI	.726	.037	.286
CVR	.191	.368	.862
RNR	-.101	.129	-.449
Interclass	.267	.827	.220
Intra	.953	.089	-.018
Typel	.468	.654	-.086
Typell	.655	.355	-.002
Typelll	.711	.064	.303

Extraction Method: Principal Component Analysis.  
Rotation Method: Varimax with Kaiser Normalization.

Rotated Component Matrix <sup>a</sup>				
xalan	Component			
	1	2	3	4
Totalcnefr agments	.888	.050	.444	-.034
Clone100 0loc	.157	.071	.746	.456
CRFL	.074	.880	-.139	-.147
NBR	.024	.950	.004	-.120
RSA	-.163	.835	.404	.179
RSI	.596	-.064	.197	.517
CVR	.026	.809	.414	.317
RNR	-.132	-.014	.117	-.712
Interclass	.343	.223	.766	-.213
Intra	.948	-.070	.116	.085
TypeI	.423	.046	.702	-.269
TypeII	.753	.099	.072	.107
TypeIII	.650	-.039	.137	.092

Extraction Method: Principal Component Analysis.  
Rotation Method: Varimax with Kaiser Normalization.

Rotated Component Matrix <sup>a</sup>			
camel	Component		
	1	2	3
Totalcnefr agments	.376	.577	.702
Clone100 0loc	.663	.471	.231
CRFL	.737	.379	.095
NBR	.768	-.014	.340
RSA	.922	-.103	.083
RSI	.082	.882	.066
CVR	.855	.297	.106
RNR	-.089	-.291	.225
Interclass	.527	.007	.758
Intra	-.029	.887	.210
TypeI	.199	.039	.684
TypeII	.027	.180	.766
TypeIII	.417	.699	.236

Extraction Method: Principal Component Analysis.  
Rotation Method: Varimax with Kaiser Normalization.

**Table 77: Literature summary of studies on code clones**

Research	Granularity	Modeling approach	# of modules	# of systems	Clone metrics	Quality Attribute(s)	Cloning impact results
Baba et al. (2008)	Component	Logistic Regression	40, 32	2	2 metrics	Fault-Prone	Clone metrics improved fault-prediction
Kamei et al. (2011)	File	Logistic Regression	(8,313),(9,663), (11,525)	one (3releases)	5 metrics	Fault-Density	For large modules, clone metrics improved fault-prediction
Monden et al. (2002)	File	None	2000	1	2 metrics	Reliability and maintainability	Increase reliability and decrease maintainability
Lozano et al. (2007)	Methods	None	N.A	1	None	Changeability	Clones increase maintenance efforts
Kasper and Godfrey (2008)	Function, Free blocks	None	783, 530	2	None	Maintainability	Positive impact on maintainability of software system
Krinke (2008)	Free blocks	None	N.A	5	None	Stability	Cloned code are more stable than non-cloned code
Juergens et al. (2009)	Free blocks/Tokens	None	N.A	5	None	Program correctness	Inconsistent changes of clone increase faults
Juergens et al. (2010)	SRS Documents	None	8,667 pages	28	None	Requirement redundancy	Lower the quality of SRS which implies increase development and maintainance efforts
Rahman et al. (2010)	Free blocks	None	N.A	4	3 metrics	Fault-proneness	Cloning is not harmful
Selim et al. (2010)	Function	Cox hazard model	N.A	2	9 metrics	Fault-proneness	Risk of clones is system dependent
Bettenburg et al. (2010)	Free blocks	None	N.A	3	2 metrics	Defect proneness/Post-release quality	No significant impact on post-release level
Göde&Harder (2011)	Free blocks/Tokens	None	N.A	2	4 metrics	Stability	Cloned code are more stable than non-cloned code and might require less maintainance efforts
Krinke (2011)	Free block, File	None	2202, 6406, 552	3	None	Stability	Cloned code are more stable than non-cloned code
Our study	Class	Logistic/Linear Regression, ANN	229, 256, 741, 875, 935	5 systems	13 metrics	Functional correctness	Cloning increases fault-proneness, non-cloned code has lower fault-density

## Vitae

Yasser Al-Ghamdi was born in 1981, Al-Khobar, Saudi Arabia. He received his Bachelor Degree (BS) with honors in Software Engineering from King Fahd University of Petroleum and Mineral (KFUPM), Dhahran Saudi Arabia in 2005.

In October 2005 Yasser joined Saudi Aramco, the world largest oil company, as Exploration systems analyst. Yasser involved in a variety of different projects including specialized web applications, Geological & Geophysical process integration, scientific applications deployment. Yasser is certified as Java web component developer.

Yasser wrote two papers, first for International Journal of Computer Science and Network Security, “Mining Software Repositories – A Comparative Analysis” 2010. Second, he wrote for MEOS 2011 “Unified, Automated and Map-based E&P Data Management”.

Yasser joined KFUPM as part-time Graduate student and received a Master of Science degree (MS) in Computer Science in May 2014. Yasser Al-Ghamdi can be contacted at [yasser.alghamdi@gmail.com](mailto:yasser.alghamdi@gmail.com)