

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	v
	<b>LIST OF TABLES</b>	xvi
	<b>LIST OF FIGURES</b>	xvii
	<b>LIST OF ABBREVIATIONS</b>	xix
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 OVERVIEW	1
	1.2 PROBLEM DEFINITION	1
	1.3 OBJECTIVES	2
	1.4 MOTIVATION FOR THE RESEARCH WORK	2
	1.5 METRIC SUITES VALIDATION USING FAULT PRONENESS	4
	1.6 ADAPTIVE FAULT TOLERANCE IN OPEN SOURCE SOFTWARE	5
	1.7 DEDUCING AND MAPPING THE CLASS PATH OF JAR FILE	6
	1.8 SCOPE OF THE RESEARCH WORK	6
	1.9 ORGANIZATION OF THE THESIS WORK	7
<b>2</b>	<b>BASIC TERMINOLOGIES</b>	8
	2.1 INTRODUCTION ABOUT SOFTWARE AND ITS METRICS SOFTWARE	8
	2.2 SOFTWARE CHARACTERISTICS	8

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
2.2.1	Software Engineering	8
2.2.2	Software Measurement	9
2.2.3	The Need for Software Metrics	10
2.2.4	Definition of Software Metrics	11
2.2.5	Classification of Software Metrics	13
	2.2.5.1 Product metrics	15
	2.2.5.2 Size metrics	15
	2.2.5.3 Complexity metrics	17
	2.2.5.4 Halstead's product metrics	19
	2.2.5.5 Quality metrics	22
2.2.6	Measurement Scales for Software Metrics	24
2.3	TRADITIONAL AND OBJECT ORIENTED METRICS	26
2.3.1	Traditional Metrics Applied to OO Systems	26
2.3.2	Object Oriented Metrics	28
	2.3.2.1 Chidamber and Kemerer's metrics suite	28
	2.3.2.2 Metrics for object oriented design	29
	2.3.2.3 Lorenz and Kidd's suite of design metrics	29
2.4	SOFTWARE QUALITY	30
	2.4.1 Quality Control	30
	2.4.2 Quality Assurance	30
2.5	SOFTWARE QUALITY METRICS	31
	2.5.1 Properties of Metrics	34

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
2.6	AN INVESTIGATION OF THE APPLICABILITY OF DESIGN OF EXPERIMENTS TO SOFTWARE TESTING	37
2.7	FAULT TOLERANCE IN OPEN SOURCE SOFTWARE	42
	2.7.1 Mapping JAR File	44
	2.7.2 Formal Methods	47
2.8	THE STATE OF OPEN SOURCE	57
	2.8.1 Reports Based on Rate of Usage of the Open Source Software	61
	2.8.2 Comparison of Open Source Softwares	66
	2.8.3 Report showing the Reduced Error rate while Mapping the Class Path	67
2.9	CONCLUSION	69
<b>3</b>	<b>LITERATURE REVIEW</b>	<b>70</b>
3.1	INTRODUCTION	70
3.2	REVIEW OF VALIDATING THE OBJECT- ORIENTED SOFTWARE METRICS	70
	3.2.1 Object Oriented Function Points	71
	3.2.2 Three Object Oriented Metric Suites	72
3.3	MOTIVATION OF THE RESEARCH WORK	74
	3.3.1 Multiple Programming Interface	77
	3.3.2 Open Source Software System	78
3.4	ORIGINALITY OF THE RESEARCH WORK	85
3.5	CONCLUSION AND REMARKS	86

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>4</b>	<b>RELIABILITY FRAMEWORK FOR OSS</b>	<b>87</b>
4.1	INTRODUCTION	87
4.2	OBJECTIVE	88
4.3	RESEARCH EXPERIMENTS	88
4.4	PROPOSED SYSTEM FOR OSS	90
4.5	CONCLUDING REMARKS	94
<b>5</b>	<b>METRIC SUITES VALIDATION USING FAULT PRONENESS</b>	<b>95</b>
5.1	INTRODUCTION	95
5.2	QUALITY METRIC SUITES VALIDATED IN THE RESEARCH	96
5.2.1	Chidamber and Kemerer (CK) Metrics	97
5.2.2	Robert C Martin's Metric Suite	99
5.2.3	McCabe's Metric Suite	100
5.3	DESIGN OF THE RESEARCH METHODOLOGY	102
5.4	RESULTS AND DISCUSSIONS	105
5.4.1	Empirical Validation of Selected Quality Metric Suites using Rhino Software	105
5.4.1.1	Chidamber and Kemerer metric suites vs fault proneness-statistical analysis	106
5.4.1.2	Robert C Martin metric suites vs fault proneness-statistical analysis	108

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	5.4.1.3 McCabe's metric suites Vs fault proneness - statistical analysis	110
	5.4.2 Data Analysis	113
	5.4.3 Correlation Analysis	114
5.5	CONCLUSION	117
<b>6</b>	<b>ADAPTIVE FAULT TOLERANCE IN OPEN SOURCE SOFTWARE</b>	<b>119</b>
6.1	INTRODUCTION	119
6.2	PROPOSED METHOD OF FAULT TOLERANT	120
6.2.1	Methodology	121
6.2.2	Algorithm for Detecting Error Rate	123
6.2.3	Explanation of the Algorithm	124
6.2.4	Results and Discussion	125
6.3	CONCLUDING REMARKS	127
<b>7</b>	<b>DEDUCING AND MAPPING THE CLASS PATH OF JAR FILE</b>	<b>128</b>
7.1	INTRODUCTION	128
7.2	PROPOSED AUTOMATED TECHNIQUE	129
7.2.1	Methodology	129
7.2.2	Algorithm for Deducing the JAR File	131
7.2.3	Experimental Results	131
7.3	CONCLUDING REMARKS	136

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>8</b>	<b>CONCLUSION AND FUTURE</b>	
	<b>SCOPE OF WORK</b>	<b>137</b>
8.1	OVERVIEW	137
8.2	FINDINGS AND CONTRIBUTIONS	138
	8.2.1 Empirical validation of metric suites	138
	8.2.2 Fault Tolerance in OSS	138
	8.2.3 Mapping Class Path File	139
	8.2.4 Reliability Framework for OSS	139
8.3	APPLICABILITY OF THE RESEARCH	140
8.4	LIMITATIONS	140
8.5	FURTHER RESEARCH	140
	<b>APPENDIX 1</b>	<b>142</b>
	<b>APPENDIX 2</b>	<b>151</b>
	<b>APPENDIX 3</b>	<b>192</b>
	<b>REFERENCES</b>	<b>197</b>
	<b>LIST OF PUBLICATIONS</b>	<b>208</b>
	<b>CURRICULUM VITAE</b>	<b>209</b>

## LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
2.1	Number and percent of faults triggered by n-way conditions	38
2.2	Explanation of various features	60
2.3	Usage rate of open source software apache	61
2.4	Rate of usage of open source software Linux	62
2.5	Rate of usage of open source software MYSQL	63
2.6	Rate of usage of open source software PHP	64
2.7	Rate of usage of open source software java	65
2.8	Comparison of open source softwares	66
2.9	Error rate while mapping the class path manually	68
5.1	Correlation Values obtained after Analysis for WMC	106
5.2	CK Metric Suites of the Rhino Versions	107
5.3	Robert C. Martin metric suites of Rhino Versions	109
5.4	McCabe's metric suites of the rhino Versions	111
5.5	Characteristics of Rhino software versions used for empirical validation	114
5.6	Results of correlation analysis	116
6.1	Error Report	126
7.1	Execution rate (Before implementation)	132
7.2	Execution rate (After implementation)	133
7.3	Error rate after Automated	135

## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
2.1	Features of open source softwares	59
2.2	Rate of usage of open source software apache (Bar diagram)	62
2.3	Rate of usage of open source software linux(Bar diagram)	63
2.4	Rate of usage of open source software MYSQL (Bar diagram)	64
2.5	Rate of usage of open source software PHP(Bar diagram)	65
2.6	Rate of usage of open source software java (Bar diagram)	66
2.7	Comparison chart for open source software (Bar diagram)	67
2.8	Error rate while mapping the class path manually (Bar chart)	68
4.1	Empirical Validation of Metric Suite for OSS	91
4.2	Reliability Checking System for OSS	93
5.1	Metric suites validated in the proposed approach	96
5.2	Research design of the empirical validation of metric suites	103
5.3	Metric Suites and their Relevance	104
6.1	Methodology of Adaptive Fault tolerance for OSS	122
6.2	Algorithm for detecting error rate	123



<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
7.1	Methodology for Mapping and Deducing Class path	130
7.2	Algorithm for automated technique	131
7.3	Execution Rate (Before implementation)	133
7.4	Execution rate (After implementation)	134
7.5	Error rate after automated (Bar chart)	135

## LIST OF SYMBOLS AND ABBREVIATIONS

Ab	-	Abstractness
AFTM	-	Adaptive fault tolerance management
ACE	-	Advanced computing environment
Ca	-	Afferent coupling
API	-	Application programming interface
AHF	-	Attribute hiding factor
AIF	-	Attribute inheritance factor
AEOSS	-	Automated execution of open source software
BLCR	-	Berkeley lab checkpoint/restart
BLOC	-	Blank lines of code
BFT	-	Byzantine fault tolerant
CK	-	Chidamber and Kemerer
CC	-	Class complexity
CGI	-	Common gateway interface
CLOC	-	Common lines of code
CORMA	-	Common object request broker architecture
C	-	Complexity
CMP	-	Configuration management plan
$\rho$	-	Correlation coefficient between ranks
CBO	-	Coupling between object classes
CO	-	Coupling factor
VG	-	Cyclomatic complexity
DIT	-	Depth of inheritance tree
T	-	Development time
Ce	-	Efferent coupling
E	-	Effort
ECC	-	Elliptic curve cryptography

XP	-	Extreme programming
FTFI	-	Failure triggering fault interaction
FP	-	Fault proneness
FDA	-	Food and drug administration
GUI	-	Graphical user interface
In	-	Instability
IDE	-	Integrated development environment
JAR	-	Java archive file
LCOM	-	Lack of cohesion in methods
LOC	-	Line of code
LAM	-	Local area multicomputer
SLOC-L	-	Logical source lines of code
l	-	Lower bound
MTTI	-	Mean time to interrupt
MC	-	Method complexity
MHF	-	Method hiding factor
MIF	-	Method inheritance factor
MLOC	-	Method lines of code
MPI	-	Multiple programming interface
NBD	-	Nested block depth
Dn	-	Normalized distance from main sequence
NOC	-	Number of children
CWORDS	-	Number of comment words
e	-	Number of edges
OOFP	-	Object oriented function points
ORB	-	Object request broker
OSS	-	Open source software
OS	-	Operating system
OASIS	-	Organization for the advancement of structured information standards

PC	-	Package complexity
PF	-	Polymorphism factor
POSIX	-	Portable operating system interface for UNIX
N	-	Program length
n	-	Program vocabulary
QA	-	Quality assurance
QMOOD	-	Quality metrics for object oriented design
RAX	-	Remote agent experiment
RFC	-	Response for a class
SLOC	-	Source lines of code
SQL	-	Structured query language
SDI	-	System design instability
TAO	-	The advanced computing environment (ACE)
n2	-	The number of unique operands in the program
n1	-	The number of unique operators in the program
N2	-	The total number of operations in the program
N1	-	The total number of persons in the program
TV	-	Threshold value
TL	-	Tolerance level
TLOC	-	Total lines of code
UML	-	Unified modeling language
URL	-	Uniform resource locator
u	-	Upper bound
V&V	-	Verification & validation
V	-	Volume
WS-BPEL	-	Web services business process execution language
WMC	-	Weighted methods per class