AN ENHANCED FORMAL INSPECTION PROCESS TO IMPROVE THE EFFICIENCY AND EFFECTIVENESS OF SOFTWARE INSPECTION PROCESS

NAVID HASHEMI TABA

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

*(Please delete this part) Depending on the medium of thesis/dissertation, pick either the English or Bahasa Malaysia version and delete the other.

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: (I.C/Passport No:)

Registration/Matric No:

Name of Degree:

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Field of Study:

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

ABSTRACT

In software development, it is important to detect and remove software defects as early as possible using a software inspection method. A formal inspection process (Process 1), was first introduced by Michael Fagan in 1976, and it was claimed that it can reduce about 38% of the defects in the software during development. Although some enhancements and improvements have been made to Process 1, and a few new inspection processes have been introduced, there are still weaknesses which include the absence of proper selection criteria for choosing the most suitable/qualified inspectors to conduct inspection on specific types of artefacts, some existing inspection processes do not have a database to help in the preparation of inspection checklists, and/or a database to store information on potential causes of each defect, which can provide a fast and easy reference to help in removing the defects. There is also a lack of inspection support tools that can help the software inspectors to conduct online inspection process which can save both time and costs for the inspection team members. This research is aimed at introducing an enhanced inspection process, ISIP (Process 2), by making some enhancements to overcome or reduce the weaknesses in Process 1. The enhancements include providing a method of selecting suitable inspectors based on their expertise and work experience; providing a defects database to store information on potential defects that are most commonly found in the requirements analysis and design phases, and which can be used as a reference for the preparation of inspection checklists; providing a database to store the possible causes of each of the defects stored in the defects database; and incorporating a few enhancements in the inspection process. To facilitate the selection of suitable inspectors and the inspection process, an inspection tool, ArSeC was developed using agile development techniques and MS SQL development environment. To determine whether the use of Process 2 with ArSeC can improve the inspection process, the process was evaluated using Paired T-test which compares the differences between Process 1 and

Process 2, in terms of the number of defects detected, inspection time, and the productivity of the inspectors. Two case studies were conducted in two companies and inspections were carried out on two software development projects to collect the data needed for the statistical test. The statistical test results show that at $\alpha = 0.05$, the mean values of the number of defects detected, the inspection time, and the productivity of the inspectors when using Process 1 and Process 2 are 19.78 and 28.13; 220.97 minutes and 213.69 minutes; and 0.0891 and 0.1328, respectively. This shows that the use of ISIP effect significant improvement in the number of defects detected (i.e. difference = 8.35; 42.21%), and productivity of the inspectors (i.e. difference = 0.0437; 49.05%). However, there is only a slight improvement in the inspection time (i.e. difference = 7.28 minutes; 3.29%). It can be concluded that ISIP together with ArSeC improve the quality of inspection process in the two case studies.

ABSTRAK

Dalam pembangunan perisian, adalah penting untuk mengesan dan menghapuskan kecacatan perisian seawal mungkin dengan menggunakan proses pemeriksaan perisian. Satu proses pemeriksaan formal (Proses 1), yang mula-mula diperkenalkan oleh Michael Fagan pada tahun 1976, dan telah didakwa bahawa ianya boleh mengurangkan kira-kira 38% kecacatan yang dalam perisian pada masa pembangunan. Walaupun beberapa tambahan dan penambahbaikan telah dibuat terhadap Proses 1, dan beberapa proses pemeriksaan baru telah diperkenalkan, masih terdapat kelemahan yang termasuk ketiadaan kriteria pemilihan yang wajar untuk memilih pemeriksa yang paling sesuai/ layak untuk menjalankan pemeriksaan pada jenis artifak tertentu, sesetengah proses pemeriksaan yang sedia ada tidak mempunyai satu pangkalan data untuk membantu dalam penyediaan senarai semakan pemeriksaan dan/atau satu pangkalan data yang menyimpan maklumat mengenai penyebab potensi bagi setiap kecacatan, yang boleh memberi satu rujukan yang cepat dan mudah untuk membantu dalam menghapuskan kecacatan. Terdapat juga kekurangan alat sokongan pemeriksaan yang boleh membantu pemeriksa perisian untuk menjalankan proses pemeriksaan dalam talian yang boleh menjimatkan keduanya perjalanan dan kos bagi ahli pasukan pemeriksaan. Penyelidikan ini adalah bertujuan untuk memperkenalkan satu proses pemeriksaan dipertingkatkan yang dikenali sebagai ISIP (Proses 2) dengan membuat beberapa penambahbaikan untuk mengatasi atau mengurangkan kelemahan yang telah dikenalpasti dalam Proses 1. Penambahbaikan yang dibuat termasuk satu proses memilih pemeriksa yang sesuai berdasarkan kepakaran dan pengalaman kerja mereka; menyediakan satu pangkalan data kecacatan yang menyimpan maklumat mengenai kecacatan potensi yang paling biasa dijumpai dalam fasa analisis keperluan dan fasa rekabentuk yang boleh digunakan sebagai rujukan bagi penyediaan senarai semakan pemeriksaan; mempunyai satu pangkalan data yang menyimpan penyebab yang mungkin bagi setiap kecacatan yang

disimpan dalam pangkalan data kecacatan itu; dan menggabungkan beberapa penambahbaikan dalam proses pemeriksaan. Bagi memudahkan pemilihan pemeriksa yang sesuai dan proses pemeriksaan, alat pemeriksaan, ArSeC dibangunkan menggunakan teknik pembangunan tangkas dan persekitaran .Net MS-SQL. Untuk menentukan sama ada penggunaan Proses 2 dengan ArSeC boleh meningkatkan proses pemeriksaan, proses dinilai menggunakan Paired-Samples T-Test untuk membandingkan perbezaan antara Proses 1 dan Proses 2, dari segi bilangan kecacatan yang dikesan, masa pemeriksaan, dan produktiviti pemeriksa. Dua kajian kes telah dijalankan di dua syarikat dan pemeriksaan telah dijalankan ke atas dua projek pembangunan perisian untuk mengumpul data yang diperlukan untuk ujian statistikal tersebut. Keputusan ujian statistikal menunjukkan bahawa pada $\alpha = 0.05$, nilai purata bilangan kecacatan dikesan; masa pemeriksaan; dan produktiviti pasukan pemeriksa apabila menggunakan Proses 1 dan Proses 2 adalah 19.78 dan 28.13; 220.97 minit dan 213.69 minit; dan 0.0891 dan 0.1328, masing-masing. Ini menunjukkan bahawa kegunaan ISIP membawa peningkatan yang ketara dalam bilangan kecacatan yang dikesan (iaitu, perbezaan = 8.35; 42.21%), dan produktiviti pasukan pemeriksa (iaitu, perbezaan = 0.0437; 49.05%). Walau bagaimanapun, hanya terdapat sedikit kemajuan dalam masa pemeriksaan (iaitu, perbezaan = 7.28 minit; 3.29%). Sehubungan itu, boleh disimpulkan bahawa ISIP bersama ArSeC meningkatkan kualiti proses pemeriksaan dalam dua kajian kes itu.

ACKNOWLEDGEMENTS

Praise be to God Almighty Who has blessed and endowed me with fortitude to reach this pinnacle of my academic pursuit. The past six years had taken me through an arduous academic journey, which I have made every endeavor to arrive at a fruitful end. Many people, in one way or another, have helped in this journey in pursuit of my doctoral degree. Problems, big and small, which I was at times desperate to solve, were resolved one after another, and as time passes, I attained greater awareness of the true aim and meaning of my research. I acknowledge Him, and I beseech Him to assist me in my service to mankind.

All along the lengthy course of my research, Associate Prof. Dr. Ow Siew Hock, my supervisor, patiently guided, encouraged me to overcome some problems, academic and non-academic, I encountered. Without her invaluable guidance, her extensive knowledge, experience, and wise words of advice, I would certainly not have been able to arrive at this juncture. I wish to express my deepest gratitude to her.

Associate Prof. Dr. Noor Haroon Abdul Karim, from the Department of Library and Information Science, very kindly provided statistical consultation for this thesis. An expert in statistical tests, he guided me on data gathering and validation, and the use of appropriate statistical tests for my research. I am truly grateful for his help in this important aspect of my thesis.

Mr. K H Teh helped to proofread this thesis. With extraordinary patience and finesse, he made the necessary corrections to the text of the thesis. I am indebted to him for the professionalism he showed in his task.

I greatly appreciate and recognize the help of the managers of the companies which allowed me to use the data from their major software projects. In particular, I would like to thank the project managers of those two companies who have kindly agreed to implement the proposed inspection process from this research. Their trust in me, yet not

seeking any remuneration of any kind, is indeed a most noble gesture which augurs well in fostering close collaboration between the industry and the university. Credit must be given to the experienced inspectors who participated in this research voluntarily, out of their professionalism. Throughout the case studies, they remained unbiased and put aside their personal opinions. As the thesis was being done, other studies were also carried out in other companies, projects, and with other people partaking. Some led to the publication of various papers, and some shed light on the main direction of this research. I owe my sincere gratitude to all those who contributed to the aforementioned studies in one way or another.

I am glad to be able to pursue my doctorate degree in the University of Malaya, which has a tradition of academic and research excellence. The University had generously provided all the software and hardware facilities as well as various services to effect an environment conducive to qualitative and quantitative research. In this regard, I am especially thankful to Professor Dr. Abdullah Gani, Dean of the Faculty of Computer Science and Information Technology (FCSIT), and Associate Prof. Dr. Rodina, Head, Department of Software Engineering, for accepting my candidacy for the PhD program and availing me all the facilities needed for my research. I would also like to acknowledge the staff of the university at various levels for their warm and friendly attitude, and the professional services which they offered, unhesitatingly. I hereby also acknowledge the Research Grant (PPP) extended to me by the University of Malaya under the Postgraduate, Account Number: PS027-2012A, which was used to carry out this research.

Almost half a century ago, my erudite father and insightful, kind mother impressed upon me the value of education and the splendor of science. For fifty years, my father humbly illuminated students' minds while teaching using the blackboard, on the importance of amalgamating science and culture. Never through all these years, have I heard from him a word of complaint about life. Since early childhood, my mother taught me that wealth

is mortal but knowledge is perpetual. She dispensed with the many conventional pleasures of life so that her children can receive proper education. My parents provided a rich cultural environment their children' talents to blossom, and they sacrificed their whole life so that their children can achieve both intellectual and spiritual development. I never have the opportunity to make up for even a small part of what they did. I am at the loss for words to express my gratitude to them for inculcating us with the ethical standards to live our life, and the unceasing quest for education to gain knowledge and wisdom. To my lovely parents, I dedicate my whole life's achievements.

My wife, Ahdieh, has been my friend, supporter, and faithful companion as I undertake my academic pursuit. She underwent great hardship; and made equally great sacrifices; but she has never ceased to support me and to care for our family. My daughter, Niki, grew from one year old to eight years old, and Hafez, my little son, set foot on this world while I was taking part in conferences and gathering and analyzing tremendous amount of data. During the past years, my studies has inevitably encroached on myfamily life to some extent, and I could not allocate enough time to my family. My dear family has always been the highest priority of my life, and their sacrifice indeed encouraged me to continue my research. I seek their forgiveness, and I hope they will allow me to make up for all their sacrifices, soon.

My brothers, Saeid and Maysam, compensated for my failure to serve my parents over the years when I was busy with my doctorate studies. They deserve my eternal gratitude.

Last but not least, the subject of this thesis is software quality improvement and defect reduction. Hence, it will benefit the children of the present generation - those who will strive to make this world a better place as engineers, managers, and experts in the sciences and other life disciplines in future.

I very much hope that this research will serve as a small but valuable illumination of the path for those who have faith in better quality of life. I would like to extend my gratitude

in advance to those who will read, and critique my thesis, and continue to improve on the findings reported, here within.

This research is dedicated to my learned father, my devoted, insightful mother, and my wife, Ahdieh, who has been unwavering in her support for me for almost a decade. I would also like to dedicate this dissertation to my two beloved children, my daughter, Niki, and my son, Hafez. I hope they will choose the path of knowledge, wisdom, and service to mankind. This work is also dedicated to all dear scholars who are preoccupied with creating a better and healthier world, where peace and kindness prevails over violence and hatred.

Navid Hashemi Taba

TABLE OF CONTENTS

Abstract iv

Abstrak vi

Acknowledgements viii

Table of Contents xii

List of Figures xxii

List of Tables xxiv

List of Symbols and Abbreviations xxvii

List of Appendices xxviii

CHAPTER 1: INTRODUCTION 1

1.1 RESEARCH BACKGROUND 1

1.2 PROBLEM STATEMENTS 3

1.3 RESEARCH OBJECTIVES 5

1.4 RESEARCH QUESTIONS 5

1.5 RESEARCH SCOPE 6

1.6 RESEARCH METHODOLOGY 7

1.7 ORGANISATION OF THE THESIS 8

CHAPTER 2: LITERATURE REVIEW 9

2.1 SOFTWARE TESTING AND SOFTWARE INSPECTION 9

2.2 SOFTWARE INSPECTION PROCESS 12

2.2.1 Formal Inspection Process Improvements 13

2.2.2 The Fagan Defect-Free Process 17

2.3 SOFTWARE INSPECTION TEAM 18

2.4 IEEE FORMAL INSPECTION PROCESS 21

2.4.1 Planning 19

2.4.2 Overview 19

2.4.3 Preparation 19

2.4.4 Examination/ Meeting/Inspection phase 19

2.4.5 Rework 20

2.4.6 Follow-up 20

2.4.7 Third Hour (Optional Step) 20

2.5 INSPECTION AND DEFECT REMOVAL 20

2.6 IMPORTANCE OF INSPECTION IN EARLY PHASES OF SOFTWARE DEVELOPMENT 22

2.7 A CHRONOLOGICAL LIST OF SOFTWARE INSPECTION RELATED ARTICLES 23

2.7.1 The subjects covered in the articles reviewed 35

2.8 SOFTWARE INSPECTION PROCESSES FOR THE REQUIREMENTS ANALYSIS AND DESIGN PHASES 36

2.8.1 Software Inspection Processes Applicable in the Requirements Analysis Phase 37

2.8.1.1 Structured Walkthroughs 37

2.8.1.2 Phased Inspection 37

2.8.1.3 Inspection without a Meeting 38

2.8.1.4 Inspection with Brainstorming Session 38

2.8.1.5 Software Inspection Process Applicable in the Design Phase 39

2.8.1.6 Active Design Review 39

2.8.1.7 Two-Person Formal Inspection Method 40

2.8.1.8 High-Level Object-Oriented Designs Inspection 40

2.8.1.9 Usage-Based Testing (UBT-i) Method 40

2.8.1.10 Multiple Team (N-fold) Inspection Method 41

2.9 PROBLEMS OF USING CURRENT INSPECTION METHODS IN THE ANALYSIS AND DESIGN PHASE 43

2.9.1 Common Problems Encountered when Using the Inspection Methods in the Requirement Analysis Phase 43

2.9.2 Specific Problems Encountered when Using the Inspection Methods in the Requirements Analysis Phase 43

2.9.3 Common Problems Encountered when Using the Inspection Methods in the Design Phase 46

2.9.4 Specific Problems Encountered when Using Inspection Methods in the Design Phase 46

2.10 STANDARD CLASSIFICATION OF SOFTWARE DEFECTS 46

2.10.1 Defect Classification in the Requirements Analysis Phase 46

2.10.2 Defects Classification in the Design Phase 50

2.11 FINDINGS OF THE DEFECT CLASSIFICATION IN THE REQUIREMENTS ANALYSIS PHASE AND DESIGN PHASE 52

2.12 MISSING ASPECTS OR GAPS 53

2.13 SUMMARY 54

CHAPTER 3: RESEARCH METHOD 56

3.1 RESEARCH METHOD USED 56

3.1.1 Activity 0: Choosing Area Subject 56

3.1.2 Activity 1: Literature Review 57

3.1.3 Activity 2: Problems Identification 57

3.1.4 Activity 3: Propose an Improved Software Inspection Process (ISIP) 58

3.1.5 Activity 4: Develop a Web-based tool (ArSeC) 58

3.1.6 Activity 5: Conduct case studies to evaluate the proposed software inspection process 59

3.1.6.1 Hypothesis 1: 61

3.1.6.2 Hypothesis 2: 61

3.1.6.3 Hypothesis 3:.....61

3.2 EVALUATION OF ISIP.....62

3.2.1 Number of real defects detected.....62

3.2.2 Inspection Time.....63

3.2.3 Productivity of Inspection Team.....64

3.3 VALIDITY OF RESEARCH.....64

3.3.1 Statistical validity.....64

3.3.2 Construct validity.....65

3.3.3 External validity.....66

3.3.4 Internal validity.....66

3.3.5 Summary.....67

CHAPTER 4: THE PROPOSED INSPECTION PROCESS.....68

4.1 INSPECTION TEAM.....68

4.2 AN IMPROVED SOFTWARE INSPECTION PROCESS (ISIP).....70

4.3 INSPECTION PROCESS IN ISIP.....71

4.3.1 Preparation.....71

4.3.2 Defect Detection 72

4.3.3 Pioneer Inspection Kernel 72

4.3.4 Process appraisal 73

4.4 UNIQUE FEATURES OF ISIP 74

4.4.1 Selection of Suitable Inspectors 74

4.4.2 Maintenance of Defects List 76

4.4.3 Preparation of Inspection Checklists 76

4.4.4 Avoidance of Defect Transition 77

4.4.5 Inspection Reference Guide 77

4.4.6 Shared Databases 77

4.5 COMPARISON BETWEEN ISIP AND THE FORMAL INSPECTION PROCESS: THE ENHANCEMENTS MADE 78

4.6 CLASSIFICATION OF DEFECTS 83

4.7 DEVELOPMENT OF A WEB-BASED TOOL 89

4.7.1 Artefacts and Session Control System (ArSeC) 89

4.7.2 The Workflow of ArSeC 89

4.7.2.1 Step 1: Project definition 89

4.7.2.2 Step 2: Preparation of Artefacts 90

4.7.2.3 Step 3: Formation of Inspection Team 90

4.7.2.4 Step 4: Preparation of checklist 91

4.7.2.5 Step 5: Preparation for inspection 91

4.7.2.6 Step 6: Conduct inspection 91

4.7.2.7 Step 7: Defect removal 92

4.8 DEFECTS AND CAUSES OF DEFECTS 93

4.9 FUNCTIONS AND FEATURES OF ARSEC 93

4.10 COMPARISON BETWEEN ARSEC AND OTHER INSPECTION PROCESS SUPPORT TOOLS 93

4.11 SUMMARY 95

CHAPTER 5: CASE STUDIES, DATA COLLECTION AND ANALYSIS 96

5.1 PILOT CASE STUDY 97

5.2 CASE STUDIES 98

5.3 DATA COLLECTION 99

5.3.1 Design of Inspection Data Recording Form 100

5.3.2 Administration of Inspection Process and Data Recording 101

5.3.3 Calculation of the total number of real (actual) defects detected 101

5.3.4 Calculation of total inspection time 105

5.3.5 Calculation of the productivity of an inspection team 107

5.4 DATA ANALYSIS 107

5.4.1 Data Screening 109

5.4.2 Test of Hypotheses 113

5.4.2.1 Hypothesis 1 113

5.4.2.2 Hypothesis 2 115

5.4.2.3 Hypothesis 3 116

5.5 INSPECTION STOPPING CRITERIA 118

5.6 OTHER FINDINGS OF THE CASE STUDIES 119

5.6.1 Defect Density 119

5.6.2 Evaluation of the Efficiency of the Inspection Processes – An Alternative Approach 123

5.6.3 Productivity of Inspectors 126

5.6.4 Most Common Defects Detected 131

5.6.4.1 Defects detected in the requirements analysis phase (Project 1 and Project 2) 131

5.6.4.2 Defects detected in the design phase (Project 1 and Project 2) 135

5.6.5 Lessons learned from the two case studies 140

5.7 SUMMARY 140

CHAPTER 6: DISCUSSION AND CONCLUSION 143

6.1 RESEARCH VALIDITY AND RELIABILITY 143

6.2 BIAS IN THE RESEARCH 145

6.3 ETHICAL ISSUES 146

6.4 PROBLEMS ENCOUNTERED 147

6.4.1 Cooperation from software companies 147

6.4.2 Number of artefacts used in the case study 148

6.4.3 Number of inspectors involved in the case study 148

6.5 CONTRIBUTIONS OF THE RESEARCH STUDY 149

6.5.1 The number of inspectors in the inspection team 149

6.5.2 The roles and responsibilities of the inspection team 150

6.5.3 Selection of inspectors based on inspector skill levels 150

6.5.4 Inspection checklist 150

6.5.5 Inspection process and inconsistencies resolution 151

6.5.6 Potential causes of each defect 152

6.5.7 Use of automated inspection tool (ArSeC) and inspection meeting.....152

6.6 RESEARCH CONCLUSION.....153

6.7 FUTURE RESEARCH.....155

References.....157

List of Publications and Papers Presented.....167

Appendix A.....169

Appendix B.....174

Appendix C.....175

Appendix D.....179

Appendix E.....183

Appendix F.....189

LIST OF FIGURES

Figure 2.1: The five phases of Fagan's formal inspection process (1976) 13

Figure 2.2: Formal Inspection Process (Six-Step) (Fagan, 1986) 14

Figure 2.3: Fagan's Formal Inspection (Seven-Step) (Fagan, 1999) 15

Figure 2.4: Formal Inspection Steps of the IEEE STD 1028-1988, rev. 1997 standard . 21

Figure 2.5: Steps in defect removal strategy 22

Figure 2.6: Cost to detect and correct defects in different development phases (Nair Suma, Kumar, 2011) 23

Figure 2.7: Inspection Session Stages in the Parnas and Weiss method (1985) 39

Figure 2.8: Techniques for Reading in Two Axes 40

Figure 2.9: The Major Steps of UBT-i Inspection Method 41

Figure 2.10: Inspector Responsibilities in UBT-i Approach 41

Figure 3.1: Research Activities 57

Figure 3.2: Research Design 60

Figure 4.1: The work flow of ISIP 71

Figure 4.2: Comparison between ISIP and Formal Inspection Process 79

Figure 4.3: The Enhancements of Inspection Process in ISIP Comparing Formal Inspection 81

Figure 4.4: The work flow of ArSeC 90

Figure 4.5: A screen shot of ArSeC 94

Figure 5.1: Case study – Two independent groups Research design 97

Figure 5.2: Histograms of data (total number of real defects detected, inspection time, and productivity of inspection team) 111

Figure 5.3: Box plot of variables 121

Figure 5.4: Histograms of variables 122

Figure 5.5: Average productivity of each inspector 131

Figure 5.6: Comparison of defects detected in Project 1 according to defect class (inspected by Team A and Team B) with the actual total number of defects in each defect class (Requirements analysis phase) 134

Figure 5.7: Comparison of defects detected in Project 2 according to defect class (inspected by Team C and Team D) with the total number of defects in each defect class (Requirements analysis phase) 135

LIST OF TABLES

Table 2.1: Comparison of the defects detected using software inspection and software testing 10

Table 2.2: A Chronological List of Software Inspection Articles from 1976 to 2015.... 23

Table 2.3: Number of articles that contain the subjects relevant to the inspection process 35

Table 2.4: Common Problems Encountered when using Inspection Methods in the Analysis Phase 42

Table 2.5: Specific Problems Encountered when using Inspection Methods in Requirements Analysis Phase 44

Table 2.6: Common Problems of Inspection Methods Used in the Design Phase..... 45

Table 2.7: Specific Problems Encountered when using Inspection Methods in the Design Phase 47

Table 2.8: Software Defects in Requirement Analysis Phase classified based on (IEEE Std 1044-2009) Standard 48

Table 2.9: Types of Defects in the Analysis Phase (Hong et al., 2008)..... 49

Table 2.10: ODC of Defects in the Analysis Phase (Chillarege et al., 1992) 49

Table 2.11: Software Defects in Design Phase (NASA, 2013)..... 51

Table 2.12: Design Defects Adopted (Chaar, 1993) 51

Table 2.13: Software Defects Classification in the Design Phase (IEEE Std 1044-2009) 53

Table 4.1: The skill levels of the inspectors..... 74

Table 4.2: Combination of weighted votes of three inspectors..... 76

Table 4.3: Classification of defects (Requirements analysis phase) 84

Table 4.4: Classification of defects (Design phase)..... 86

Table 4.5: Comparison between ArSeC and nine current inspection support tools..... 95

Table 5.1: Title and purpose of each section of the inspection data recording form 100

Table 5.2: Summary of defects classes, inconsistencies and final decision 103

Table 5.3: Profiles of inspectors of the four inspection teams 104

Table 5.4: Total number of real defects detected, total inspection time, and productivity of inspection team 108

Table 5.5: Skewness the distribution of data 112

Table 5.6: Kolmogorov-Smirnov tests of normality 113

Table 5.7: Paired-Samples Statistics 114

Table 5.8: Paired-Samples Correlations 114

Table 5.9: Paired-Samples Test 115

Table 5.10: Test of hypothesis 2 using Related-Samples Wilcoxon Signed Rank test: Summary 116

Table 5.11: Paired-Samples Statistics 117

Table 5.12: Paired-Samples Correlations 117

Table 5.13: Paired-Samples Test 117

Table 5.14: Defect density of artefacts 119

Table 5.15: Skewness in the distribution of data (Size of artefact and Defect density) 122

Table 5.16: Kolmogorov-Smirnov test of normality (Size of artefact and Defect density) 122

Table 5.17: Pearson Correlation Test 123

Table 5.18: Efficiency of inspection processes based on the actual total number of real defects present in each artefact 124

Table 5.19: Inspectors and the artefacts they are assigned to inspect 125

Table 5.20: Total number of real defects detected and total inspection time of each inspector 127

Table 5.21: Average productivity of each inspector 130

Table 5.22: Distribution of defects detected by defect classes (Requirements analysis phase) 132

Table 5.23: Actual total number of defects detected by defect classes (Requirements analysis phase) 133

Table 5.24: Distribution of defects detected by defect classes (Design phase) 136

Table 5.25: Actual total number of defects detected by defect classes (Design phase) 136

LIST OF SYMBOLS AND ABBREVIATIONS

FIP	:	Formal inspection process
ISIP	:	Improved software inspection process
TIRT	:	Total inconsistencies resolution time
SIRT	:	Scheduled inconsistencies resolution time
TIST	:	Total inspection session time
SIST	:	Scheduled inspection session time
IST	:	Inspection session time
RD	:	Real defect
ArSeC	:	Artefacts and Session Control System

LIST OF APPENDICES

Appendix A: Artefact complexity conversion ……... 197

Appendix B: The composition of five inspectors ……..... 202

Appendix C: Brief description about project 1, project 2, and each artefact…..... 203

Appendix D: Inspection data recording form ……... 207

Appendix E: Brief description about project 1, project 2, and each artefact and sample of artefact ……. 211

Appendix F: Sample cause and effects ……..... 217

CHAPTER 1: INTRODUCTION

1.1 Research Background

In the NASA-STD-2202-93 standard, software inspection is defined as, "An in-process technical review of any software work product conducted for the purpose of finding and eliminating defects." (Gregory, 1993). Software inspection was introduced about 40 years ago by Michael Fagan (1976), and is the most formal structure of peer review which can remove as much as 80% of the total defects detected. Since then, many other researchers such as Gilb and Graham (1993), have fine-tuned the inspection process to make it an even more cost-effective instrument for tackling quality deficiencies and defect costs. In a survey of articles published on software inspection technologies, Laitenberger (2002) found that many of the articles reported the successful use of software inspection, and only one article reported on its failure to obtain the expected benefits (Shirey, 1992).

Much research efforts have been made to improve the inspection methods and the activities involved; the inspected software product; the team roles as well as its optimal size and selection of team members; the technique applied to detect defects in the software product (i.e. reading technique); and the various automated tools and their efficiency in supporting a given inspection approach (Laitenberger, 2002). However, many findings on these research issues remain controversial. For example, Ackerman, Buchwald and Lewsky (1989), Fagan (1976), Gilb and Graham (1993), and Strauss and Ebenau (1993) emphasised on the planning phase to prepare for the inspection, but most of the other researchers do not. Hence, in view of many different inspection processes, it becomes difficult for software developers to determine which inspection process or refinement to use should they wish to introduce inspection or improve on their current inspection approach.

On the issue of inspection team size, Fagan (1976) recommended four people; Bisant and Lyle (1989) preferred two persons (one inspector and the author) to perform the inspection; Weller (1993)

suggested to have three to four inspectors, Madachy, Little, and Fan (1993) and Bourgeois (1996) stated that the optimal size is between three and five people; while Kaner (1998), Martin and Tsai (1990) proposed the N-fold inspection method – i.e. N teams each carrying out parallel independent inspections of the same software artefact. Thus, there is no definite agreement as to the optimal number of inspectors, and the team size. This situation creates confusion for the developers who intend to conduct software inspection.

At the same time, the early inspection process introduced by Fagan in 1976 has also been improved from Fagan Inspection to Fagan Defect-Free Process to include the following three essential and interwoven components that are required to make software inspections successful (Fagan, 2002):

(i) Formal Process Definition ensuring each member of the team is conversant with the objectives, function, and entry and exit criteria of each process phase;
(ii) Inspection Process – the seven-step process used to find defects; and
(iii) Continuous Process Improvement – removing systemic defects from the development process.

Inspections do not only consume effort, but they also have an impact on the software product development cycle time. Inspection activities are scheduled in a way in which all team members involved can participate and fulfill their roles. Thus, the interval for the completion of all activities will range from at least a few days to up to a few weeks. During this period, other tasks that rely on the inspected software product might be delayed. Thus, the duration is a crucial issue for a software project manager if time to market a software product is a critical issue (Laitenberger, 2002). Votta (1993) discussed the effects of time loss due to scheduling contention. He reported that inspection meetings account for 10% of the development interval. He advised substituting inspection meetings with other methods of defect collection, because of delays.

Similarly, Fagan (2002) also expressed the same concern that software developers who have tight project delivery schedules, are reluctant to employ the inspection process, as they fear that inspections will take too long, lengthen the schedule and delay shipment. He stated that inspections use 20%-30% of the effort during the first half of product development, and fears that this will add to the development cycle time. Also, without full understanding of what it entails to make the process work successfully, cases of partial or improper implementation of the inspection process, which are incorrect executions of the process, often produced poor results. There are developers who would rather not include inspections in their development processes because of the fear of getting mediocre results due to mediocre implementation. Some developers who find the inspection process tedious and time-consuming, tend to make changes to the inspection process, and they do not carry it out correctly and consistently. Thus, they could not get good results from the inspection process. Also, experience has shown that all variations of the inspection processes do not produce similar results. The aforementioned challenges inspired the initiation of this research to enhance the Fagan's formal inspection process to improve the quality of software inspection, particularly during the requirements analysis and design phases of the software development lifecycle.

1.2 Problem Statements

Fagan's formal inspection process (FIP) had proven to be an effective means of reducing customer-reported defects, improving product quality as well as saving untold millions of dollars in development cost (Laitenberger, 2002). Despite its achievement, variations of the inspection process have been introduced by other researchers. Some of these variations include the way to organise the defect detection phase – should defect detection be performed individually or conducted as part of a group meeting; the purpose of the inspection preparation phase, i.e. with the goal of detecting defects or just understanding the inspected artefact to detect defects later on in a meeting session; the methods of inspection such as the Phased Inspection Method suggested by Knight and Myers (1991), and the

N-fold inspection method proposed by Martin and other researchers (Martin & Tsai, 1990; Schneider, Martin & Tsai, 1992; Kaner, 1998), and the reading technique for the defect detection activity. A reading technique is a mechanism or strategy for the individual inspector to detect defects in the inspected product. There are two most popular reading techniques used today for defect detection in inspection – ad-hoc reading and checklist-based reading (Fagan, 1976; Gilb & Graham, 1993), besides the Reading by Stepwise Abstraction advocated by the Cleanroom community for inspection on code documents only (Dyer, 1992a; Dyer, 1992b, Huzooree, & Devi, 2015) and Active Design Review suggested by Parnas and others for inspection on design documents only (Parnas & Weiss, 1985; Parnas, 1987). However, there are disagreemenst on the use of checklist-based reading. The checklist provides little support for an inspector to understand the inspected artefact if the checklist questions are too general and not sufficiently tailored to a particular development environment. Secondly, concrete instructions on how to use a checklist are often missing. Thirdly, the checklist questions are often limited to the detection of defects that belong to particular types of defect (Vitharana, 2015). Since the defect types are based on past defect information (Chernak, 1996), inspectors may not focus on defect types not previously detected and thus, may miss those classes of defects. These weaknesses pertaining to the checklist were not addressed explicitly in Fagan's FIP.

Another important factor that impacts on the success of software inspection is the human factor – the number of inspectors, team size and team selection. In the FIP, Fagan recommended to keep the inspection team small, that is, four people (Fagan, 1976). However, there is no definite answers as to the optimal number of inspectors, and the team size. In the selection of members for an inspection team (Goswami, Walia, & Singh, 2015), the primary candidates for the role of inspectors are personnel involved in product development (Fagan, 1986). Outside inspectors may be brought in if they have a particular expertise that would add to the inspection (National Aeronautics and Space Administration, 1993). The selection of inspectors is often based on length of experience, and

knowledge (Fagan, 1986; Blakely & Boles, 1991; Strauss & Ebenau, 1993). This implies that inspectors with little experience are unlikely or rarely to be chosen to participate in any inspection process. This issue has not been addressed per se and reported in Fagan's FIP, hence, this important issue has to be speedily resolved.

Besides the human factor, another factor that could impact on the inspection process is the use of automated tools to support the inspection process (Laitenberger, 2002). Automated tools for code inspection are commonly available (Huzooree, & Devi, 2015). However, there is currently no automated tool to support the formal inspection process, fully. This is an issue that this research will also address as part of the overall efforts to enhance the FIP.

1.3 Research Objectives

The detection and removal of software defects at the early phases of software development can lead to substantial reduction in the cost of software development. The main objectives of this research are:

i. to identify the activities and areas for improvement in the formal inspection process;

ii. to enhance the formal inspection process in order to improve the inspection of the artefacts of the requirements analysis and design phases;

iii. to develop an inspection tool to support the enhanced formal inspection process; and

iv. to evaluate the performance of the enhanced formal inspection process.

1.4 Research Questions

The main aim of this research is to enhance the FIP to improve the quality of software. The formal inspection process which was first introduced by Michael Fagan in 1976 has been very effective in

early detection of defects during software development. The software inspection process has generally been viewed to be tedious and time consuming, hence, researchers have made efforts to improve and simplify this process. As a result, different variations of inspection processes have been introduced. However, the formal inspection process continues to be the preferred process that is often used, with some customisation, by software companies to perform inspections on artefacts in the software development process. These crucial issues have provided the motivation to enhance the FIP to improve the quality of software inspection. Specifically, this research aims to answer the following two questions:

i. How can the FIP be enhanced to improve the quality of software inspection?
ii. How would the enhanced inspection process impact on the quality of software inspection from the perspectives of the number of defects detected, inspection time, and the productivity of an inspection team?
iii. How can the performance of the enhanced software inspection process be evaluated?

The answers to the three questions would be very beneficial to software developers who need an effective method for the detection of defects, especially in the early phases of software development, so that the overall cost of development can be drastically reduced.

1.5 Research Scope

Michael Fagan introduced the formal inspection process in 1976. Over the years, he made various enhancements to the inspection process. In 1986, he added another step to the first inspection process he introduced in 1976. 1n 1999, he improved the inspection process by adding another step to the process introduced in 1986 to become a seven-step formal inspection process. In 2002, Fagan improved the inspection process again to become the defect free process comprising three

components i) Formal process definition, ii) Inspection process, and iii) Continuous process improvement. This research only focuses on the enhancements to the second component, comprising the seven activity of his formal inspection process. (Fagan, 2002)

The enhancement made to the formal inspection process can be used to inspect artefacts in all phases of software development. However, this research focuses on the inspection of artefacts of the requirements analysis and design phases, thus, the databases that store the details on the defects, their classification, inspection checklists, causes of each defect as well as the artefacts used in this research, pertain to defects of these two phases only. Also, because of resource and time constraints, only two case studies could be carried out for the empirical study to collect at least 30 sets of inspection data from the participating companies.

1.6 Research Methodology

To carry out the research, a comprehensive literature review was conducted on the formal inspection process (FIP), which covers: basic features of the inspection process; variations of software inspection process proposed by other experts; problems related to the current software inspection process of the requirements analysis and design phases; classification of commonly detected defects in the artefacts of the requirements analysis and design phases; design of inspection checklists; the problems in rework; and issues pertaining to the inspection teams. Thus, the proposed enhancements on the FIP include: features that are lacking in the FIP, as evident from the literature survey. Statistical hypotheses will be established and tested to evaluate the performance of the enhanced FIP. Case studies will be conducted to collect data for statistical analysis to prove the hypotheses. A comparison of the features of the enhanced FIP with FIP as well as with other inspection processes will also be made to evaluate its performance.

To facilitate the inspection of artefacts of the requirements analysis and design phases using the enhanced FIP, a Web-based inspection support tool will be developed using agile development

technique and with MS SQL as the selected database management system. Chapter 3 presents the research methodology in more detail.

1.7 Organisation of the Thesis

This thesis consists of six chapters. Chapter 1 introduces the background study, defines the problem statements, research objectives, and explains the research method used to carry out the research. Chapter 2 presents the literature review pertaining to software inspection. It discusses: the formal inspection process introduced by Michael Fagan, as well as other inspection processes proposed by other researchers; the issues and weaknesses in the formal inspection process; and problems pertaining to the inspection of artefacts of the requirements analysis and design phases; and the various types of defects that are commonly detected in the requirements analysis and design phase of software development.

Chapter 3 discusses: the method used to carry out the research; metrics used to measure the quality of an inspection process; the hypotheses established for the research; and the validity of the research. Chapter 4 discusses: the roles of each inspection team member; the workflow and unique features of the proposed enhanced inspection process (ISIP); the classification of defects in the requirements analysis and design phases; and the development and features of ArSeC, an inspection support tool to support ISIP. Chapter 5 discusses: the two case studies carried out to test the three hypotheses; the Paired-Samples T Test and the Related-Samples Wilcoxon Signed Rank test. Chapter 6 discusses: the validity and reliability of the research; problems encountered and the limitations of the research; research contribution, research conclusion and suggestions for future works.

CHAPTER 2: LITERATURE REVIEW

Software inspection is a static verification as well as a validation technique (Thelin, Runeson, & Wohlin, 2003, Vitharana 2015). The aim of inspection is to evaluate software quality and not the quality of software development process. Inspections are applicable to all software products as they do not need dynamic execution. Unlike manual and automatic testing which are applied only after code completion, software inspection can be conducted in any development phases. In other words, it can be conducted starting from the first documentation in the preliminary investigation phase and continued until the maintenance phase.

2.1 Software Testing and Software Inspection

Fagan introduced software inspection in 1976. Ten years later, he published his paper on the success of software inspection (Fagan, 1986). Several research findings show that 60% to 90% of software problems are discovered by software inspections, and this has impact on the eventual software quality (Denger & Shull, 2007). Between inspection and testing, some researchers found that inspection is equally good or even more effective in finding defects during the early software development phases, as shown in Table 2.1.

Table 2.1: Comparison of the defects detected using software inspection and software testing

Researchers	Defects captured by software inspection (A)	Defects captured by software testing (B)	Development phase	Comparison between (A) and (B)
Myers (1978)	15	15	Design and Coding	I = T
Chillarege et al. (1992)	333	255	Analysis, Design, and Coding	I > T
Chaar et al. (1993)	810 (Analysis and design: 477, coding: 333)	401	Analysis, Design, and Coding	I > T
Conradi, Marjara, & Skåtevik (1999)	6,300	1,502	Analysis, Design, and Coding	I > T
Berling & Thelin (2003)	169	49	Analysis	I > T
Anderson et al. (2003)	14	13	Analysis, Design, and Coding	I > T
Gopalakrishnan et al. (2012)	307	52	Analysis, and Design	I > T

Note: I = T means inspection is as efficient as testing.
I > T means inspection is more efficient than testing.

Chaar et al. (1993) reported that 810 defects were detected using software inspection of which 477 defects were found in the requirements specification and design specification documents. On the other hand, only 401 errors were found through software testing. The total number of defects found in the analysis, design, and coding phases using inspection is twice the number of defects found by testing.

Conradi (1999) expressed that software inspection is a cost-effective approach when compared with software testing. They found that inspection was able to find 6,300 defects, which is 4,798 defects

more than the 1,502 defects detected by test activities. Besides, they also reported that only 10% of the development time was spent on the inspection process to find 70% of the defects. Another research conducted by Berling and Thelin (2003) found that although the specifications of their project were clearly defined, inspection detected 169 defects, which is about three times more than the 49 defects detected by testing.. Besides these findings, Boehm and Basili (2001) reported that inspection improved the defect detection rate from 15% to 50%. Shull et al. (2002) also reviewed the data from a large project and reported that 64% of the defects were detected through inspection. Runesson et al. (2006) reviewed 12 studies on software inspection and testing, and concluded that inspection is more effective and efficient than testing especially in the requirements analysis and design phases. They found that in seven out of the 12 studies, inspection was able to find different defects that were not detected by testing. Most researchers emphasised that inspection can reveal many errors that testing could not detect. Gopalakrishnan et al. (2012) conducted a study to compare the efficiency in defect detection between inspection and testing. The study conducted on nine separate projects found that inspections discovered 151 and 156 defects in the analysis and design phases, respectively, from the total of 307 defects. On the other hand, testing only found 33 and 19 defects in the requirements analysis and design phases, respectively from the total of 52 defects.

Furthermore, a large German company found that a defect detected by testing costs 14.5 times more to correct as did one found by formal inspection, while a defect discovered by a customer through testing costs 68 times as much to fix. IBM also reported that an error found after product release costs 45 times as much to correct as one uncovered during the design phase (Wiegers, 1995). The Jet Propulsion Laboratory estimated a net savings of $7.5 million from 300 inspections performed on software they produced for NASA. Another large company estimated an annual savings of $2.5 million due to their inspection activities, based on $146 it costs to fix a major defect found by inspection, and $2,900 to fix a defect found by the customer. Although incorporating inspections into

software engineering process can incur between 5% and 15% of the total project budget, many companies have learned that the benefits gained from a good inspection process to detect and correct errors early, far outweigh the costs of performing the inspection.

This chapter describes a theoretical framework for software inspection as an effective and efficient defect detection approach. Owing to the different processes and approaches in software inspection, similarity and mutual interference of them, detailed explanations that include a comparison of common problems, and specific constraints of those techniques are presented. Finally, the outcomes of the studies discussed here set the direction for this research to develop an enhanced inspection process.

2.2 Software Inspection Process

Software inspection is a technical review that was developed by Michael Fagan (1976) to enhance the quality of software as well as its efficiency and effectiveness. Fagan stated that there are five mandatory phases and one optional operation (acceptance) phase in a formal software inspection process. Figure 2.1 shows the five phases - *Overview*, *Preparation*, *Inspection*, *Rework*, and *Follow-up*. He also defined four roles in the inspection process – that of a *moderator, designer, coder* and *tester*. The moderator is the coach of the inspection team, whose duties include scheduling the inspection meetings and reporting the results of the inspection. Moreover, he also handles the follow-up of the reworks after the inspection. In the *overview* phase, the designer will deliver the artefact to the moderator, who will then arrange the inspection process. The moderator also calls the inspector and decides on the resources that are needed for the inspection process. In the *preparation* phase, all inspectors are given the artefact as well as the necessary documents by the moderators. In the meeting or inspection phase, all inspectors participated in the meeting sessions to discuss the potential defects and specify them. In the meeting session, all inspectors participate to recognize the defects. After

specifying the defects, the rework phase follows. In this phase, the author (designer/coder) - as the owner of artefact - is responsible for working on the artefact and removing the defects discussed in the meeting. The last mandatory phase of the inspection process is the follow-up phase, and the main responsibility rests on the moderator, who will check to ensure that all defects have been fixed, and to seek clarifications, if any, from the author.

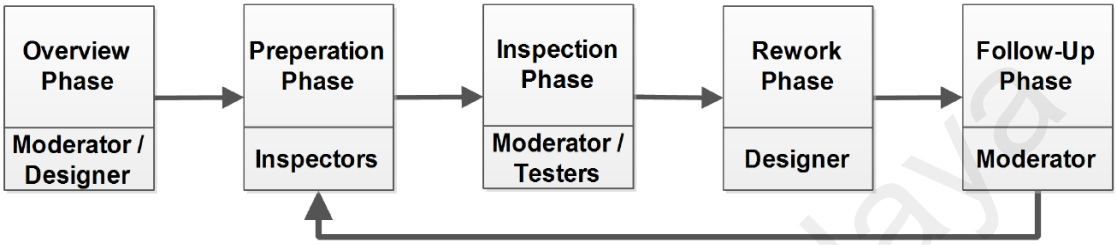


Figure 2.1: The five phases of Fagan's formal inspection process (1976)

Formal inspection plays an essential role in software quality (Perry et al., 2002). There are different review committees, various routines and frameworks to prepare the formal technical reports.

2.2.1 Formal Inspection Process Improvements

Fagan (1986) improved the software inspection process by adding one stage (planning) to the five stages previously introduced by him in 1976, as shown in Figure 2.2. The main features of this inspection process are:

(i). Clear definition of the inspectors four-fold duties of moderator, recorder, reader, or producer;

(ii). Formal collection of data for the inspection process, and product of inspection; and

(iii). An appropriate supporting infrastructure.

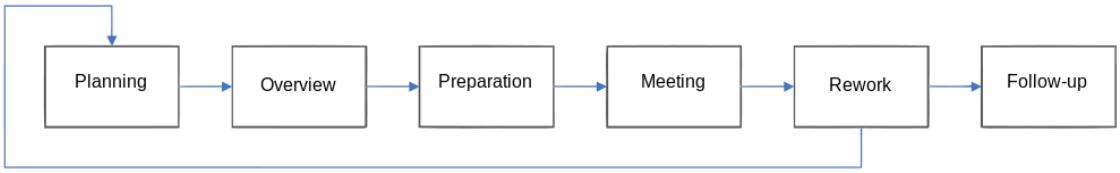


Figure 2.2: Formal Inspection Process (Six-Step) (Fagan, 1986)

The seven-step inspection is another improvement of the formal software inspection. It was introduced by Fagan in 1999 and later completed in 2002. Fagan's seven-step formal activities are shown Figure 2.3:

(a) ***Planning***

Selecting the software inspection team and sending the artefact and the related documents like defect sheet to each member of the team two or three days before the inspection meeting.

(b) ***Overview***

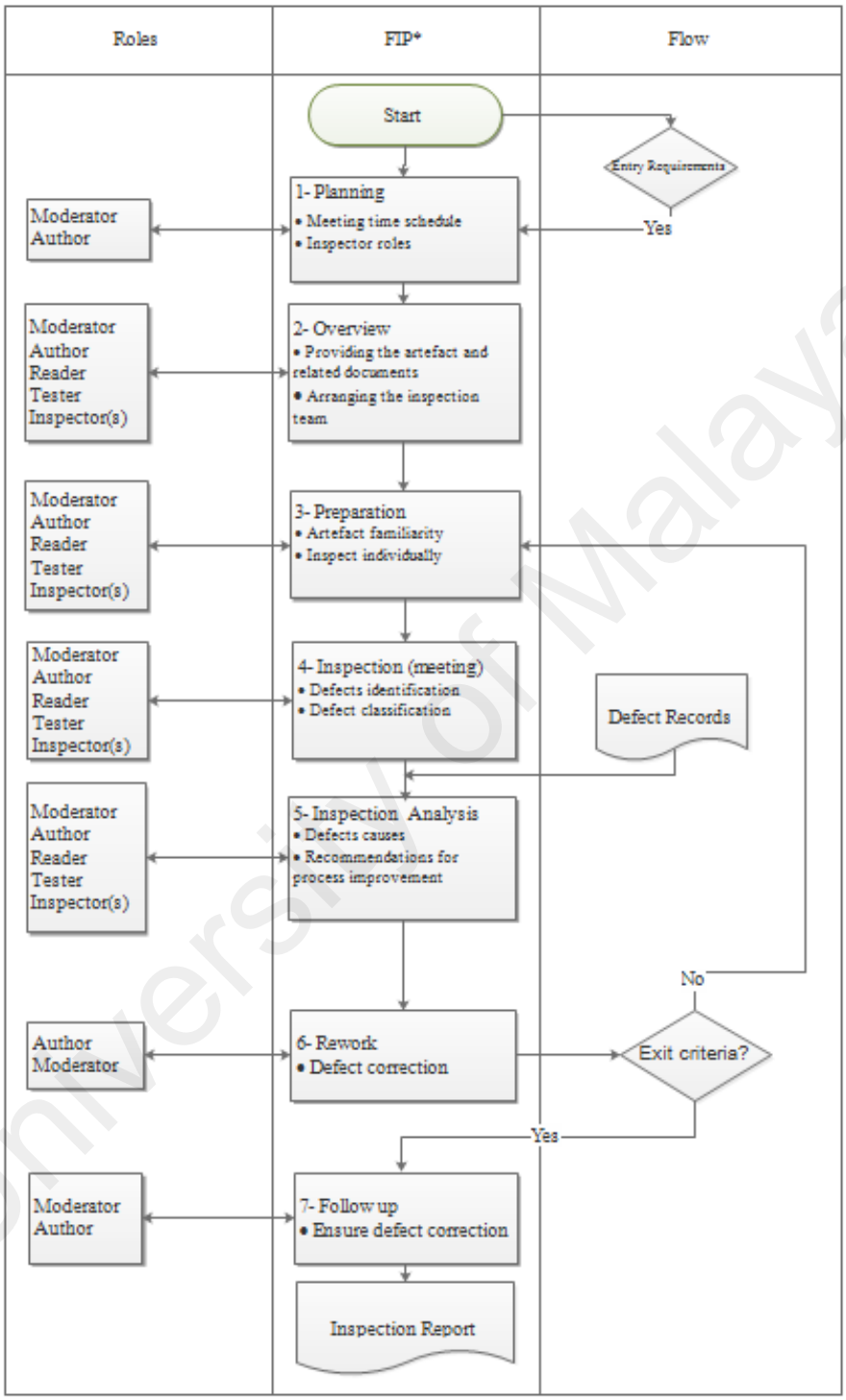
In overview meeting, the author will explain the important features of the artefact. Overview is not mandatory for those artefacts that are popular or are familiar to inspection team members.

(c) ***Preparation***

Before the inspection meeting, the artefact should be inspected by each team member to find the defect(s), individually. The correctness and completeness have to be checked. About 75% of the defects detected could be found by inspector in the preparation phase, before the inspection meeting. The checklists for recording the defect detected will be used in this phase of inspection (Barnard, & Price, 1994, Sommerville, 2013, Pressman, 2015).

The most important phase of software inspection is the individual preparation phase. Inspectors must be familiar with the development environment, development tools, projects characteristics, and the relevant software products. In their study, Van Genuchten et al. (2001) found that well–

executed inspections can discover from 60% to 80% of the defects those phases of the software development lifecycle, before the testing phase.



FIP: Formal inspection process

Figure 2.3: Fagan's Formal Inspection (Seven-Step) (Fagan, 1999)

(d) ***Inspection Meeting***

In this step, all inspectors, moderator, author and a reader gather in one place. They discuss the defects detected by each inspector during preparation phase. The explanation could be provided by author, but he/she is not allowed to make any counter-argument or to provide any reason in defense of the defects detected. The recorder will document the result of discussions.

Ackerman, Buchwald, and Lewski (1989) emphasized that providing the relevant checklists is one of the first steps of inspection. In the inspection process, they emphasized that the presence of a developer in inspection meetings can reduce the level of secrecy of the product being inspected, and this will facilitate the review.

(e) ***Inspection analysis***

Making an analysis on the defects detected, is not necessary in the current inspection. However, it can serve as a useful guide for the future inspections to avoid facing the same defects. The long-term improvement of software quality is the main goal of this step. Analyzing the data gathered from an inspection allows project managers and developers to have a better estimation of the numbers of potential defects in the forthcoming projects, and this will ensure better product quality (Biffl, 2000).

(f) ***Rework***

The corrections needed to fix the defects detected will be done in this phase by the author, who is responsible for resolving all issues raised during the inspection. It is highly recommended that the author takes into consideration the suggestions, however, the author has the prerogative to fix the defects detected in any way that he/she prefers.

(g) ***Follow-up***

In this step, the moderator will determine whether all defects detected are removed, and will follow-up with the author, if necessary. If an artefact has to be modified more than 5% or 10% (up to the moderator to decide), the inspection should be done again. Another exit criterion may clarify the completion of inspection. The correct documentation of defects in a tracking system or fixing an acceptable percentage of defects could be set as the exit criteria.

2.2.2 The Fagan Defect-Free Process

To improve the formal inspection process, Fagan (2002) introduced the Defect-Free software inspection process in 2002 and claimed that it consists of the three essential components to make the inspection process more successful: i) Formal process definition, ii) Inspection process, and iii) Continuous process improvement.

The main goal of the first component is to ensure that all team members are familiar with the process. Thus, it focuses on the defects to eliminate the systematic defects. Fagan stated that analyzing the defects can provide enough insights to avoid introducing the same fatal defects in future software developments.

However, the guidelines in all the components are still general and ambiguous in nature. Moreover, there some shortcomings in the new Defect-Free Process: the inspection session could be cancelled at any time because of the absence of one or more inspectors; the brainstorming sessions and discussion may end disagreement among the inspectors not accepting each other's views on the defects; the reader may not be professional enough as a moderator and his/her records might not be accurate; gathering the inspectors, the author, moderator in one place is very difficult, because those involved feel that the meeting sessions can be conducted even remotely through modern online telecommunication technologies ; using old classification of defects might cause the inspection to be

conducted wrongly; and even with new defect classification the potential causes are not provided and inspectors might be repeating a task such as finding the causes of a defect, which had previously been done by other inspectors.

2.3 Software Inspection Team

The inspection team can consist of more than three but less than eight members. Each team member is responsible for a specified task (role) during the inspection process. The composition of the inspection team as well as the roles of the team members could be changed or redefined during the inspection. The five main roles in the inspection process are explained below.

i) **Author** or **owner** is a professional person or a group of professionals who will prepare the artefacts. He will provide all information about the artefacts during the inspection process, and fix the defects detected.

ii) **Inspector** is a skilled person who is responsible for reviewing the artefacts created by the author. Every inspection team members who attend the inspection meeting could assume the role of an inspector (NASA-STD-8739.9).

iii) **Moderator** is a leader, manager, and controller of the whole inspection process, especially, during the inspection session. In addition, he institutes the policy to foster collaboration among the inspection team members.

iv) **Reader** is responsible for reading the artefacts if it is a document, or present the information if they are in other formats. Usually, the reader is a different person from the author in order to avoid conflict of interest.

v) **Recorder** is responsible for logging and recording all the information about the defect(s) found during the inspection process. The details of the defects recorded include the type (extracts from the pre-defined taxonomy of defects), rank (the frequency of defect type detected based on

previous similar inspections, and class or severity of defects. This information can help the inspector in estimating the expected amount of time and cost to be incurred to handle a defect of a particular type

2.3.1 Planning

This step involves scheduling the inspection activities in the other phases. The moderator will decide whether the artefact is ready for inspection, then selects the inspectors based on their ability, and subsequently invites them to participate.

2.3.2 Overview

The objectives and goals of inspection will be clarified in this phase. The moderator will explain the role of each inspector in detail, and will distribute the checklists, procedures, artefacts and any other related materials to them.

2.3.3 Preparation

In this step, each inspector makes all the preparation for the inspection meeting. It is not compulsory for the author to participate in this step. During the preparation, the potential defects, ambiguous issues, and questions raised will be documented and passed to the moderator. The inspection meeting will be held when all inspectors are ready (Fagan, 1986; IEEE STD 1028-1997).

2.3.4 Examination/ Meeting/Inspection phase

The moderator is responsible for planning, conducting, coordinating and controlling this phase. A formal meeting is held to allow the inspection team to review the artefacts together to find the potential defect(s) (NASA, 2013). At this stage, the inspector are not expected to find or suggest the solutions to remove the defects in the artefacts. However, the moderator might decide to address the causes or possible source of each defect (Fagan, 1986). In this step, a recorder is responsible for

recording all the defects discussed in the inspection meeting, noting the severity level as well as the type of defects. The moderator will decide whether to continue the meeting or arrange additional inspection session.

2.3.5 Rework

In this step, the author removes all defects that had been detected during the inspection meeting (NASA, 2013).

2.3.6 Follow-up

In this step, the moderator examines the artefacts to ensure that all the detected defects have been fixed. The moderator may verify the artefacts with other inspection team members (NASA, 2013).

2.3.7 Third Hour (Optional Step)

The third hour is optional and is decided by the author. In this step, all inspectors discuss about the defects identified during the inspection meeting. The brain-storming session will result in gathering the opinions of every inspector about the issues raised, and this will give the author a clearer idea on fixing the defects.

2.4 Inspection and Defect Removal

According to IEEE, a product has a defect when it has some shortcomings or inadequacies in providing its own requirements and attributes (IEEE Std. 2002.94130). Therefore, ‘repairing’, ‘reworking’ or ‘replacing’ must be carried out for the defect removal. Regardless of any special kind of inspection technique used to identify and remove the defect, the general defect removal strategy follows the sequence as shown in Figure 2.5.

2.5 IEEE Formal Inspection Process

According to the IEEE STD 1028-1988 rev. 1977 standard, the inspection process consists of six steps and an optional step (Third Hour), as shown in Figure 2.4, and discussed in detail in the following sub-sections.

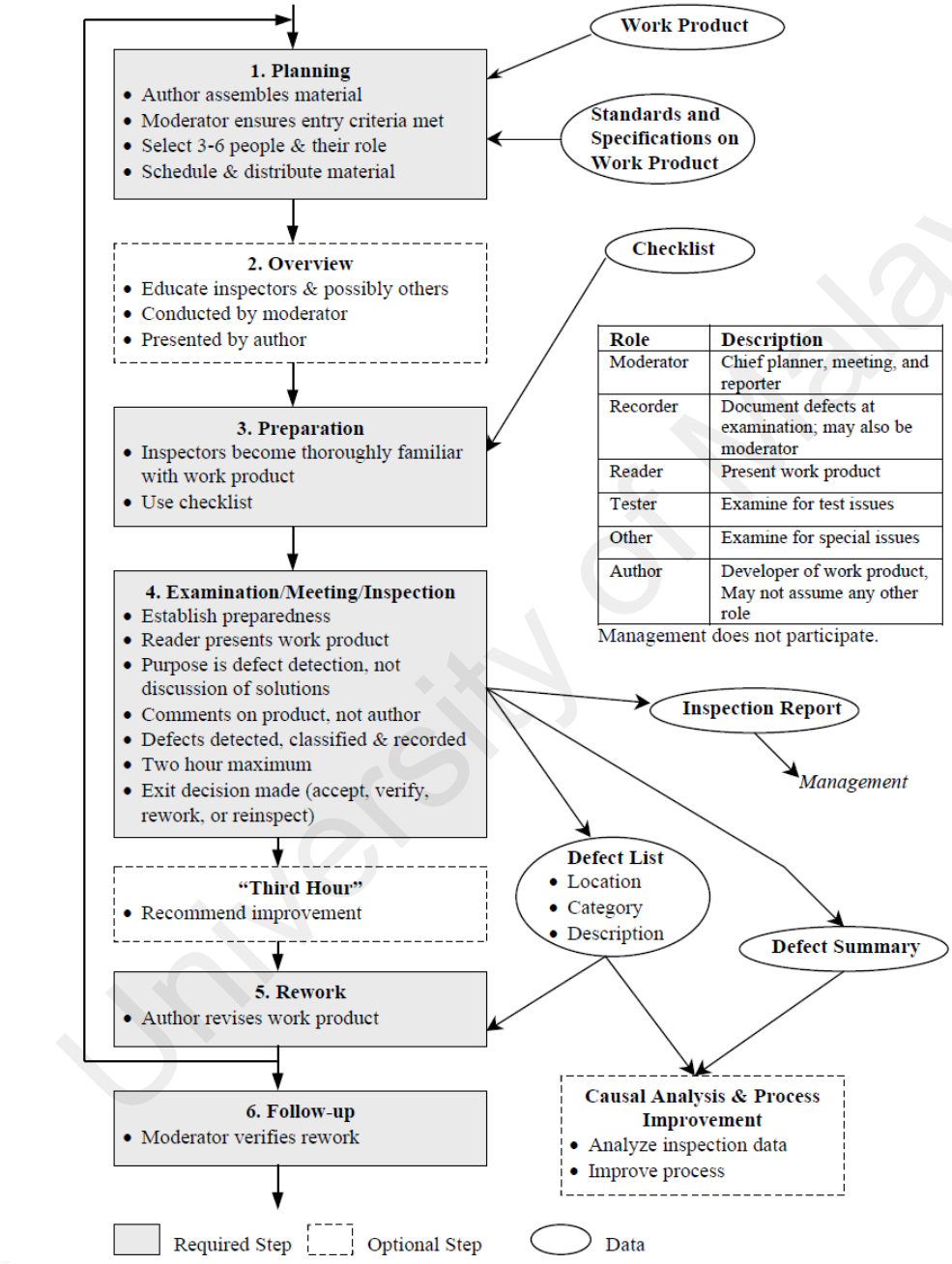


Figure 2.4: Formal Inspection Steps of the IEEE STD 1028-1988, rev. 1997 standard

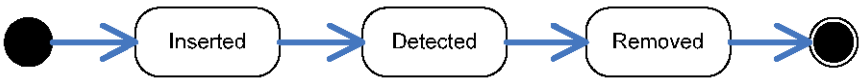


Figure 2.5: Steps in defect removal strategy

There is an IEEE standard for failure classification of the defects. However, there has been a minor improvement and change to this over the past two decades. In addition, common software inspections are usually conducted by independent inspectors using their legible software documents in their official paper, and search for defects and problems. In this way, the detected defects will be recorded in standard forms by their type and their sources. These defects are accessible only by the project managers. Usually, supplementary inspection meetings are held in the presence of individuals who have connection to the inspected documents, and the software developer presents the suggestions for eliminating the defects. However, these suggestions will only be applied upon the confirmation by the project managers.

2.6 Importance of Inspection in Early Phases of Software Development

Weinberg and Freedman (1984) emphasised that inspection must commence early in the software development lifecycle. This is because the inspection results and reports can be useful to the project manager in subsequent phases. Studies have shown that correcting a defect in the early phases of the software development lifecycle saves up to hundred times in cost, when compared to fixing a defect in the later phases of development (Nair, Suma, Kumar, 2011). Figure 2.6 shows the relative costs for correcting software defects at different phases. The detection and correction of defects in the last stages of software development can cost a hundred times more than correction of defects detected in the requirements analysis phase.

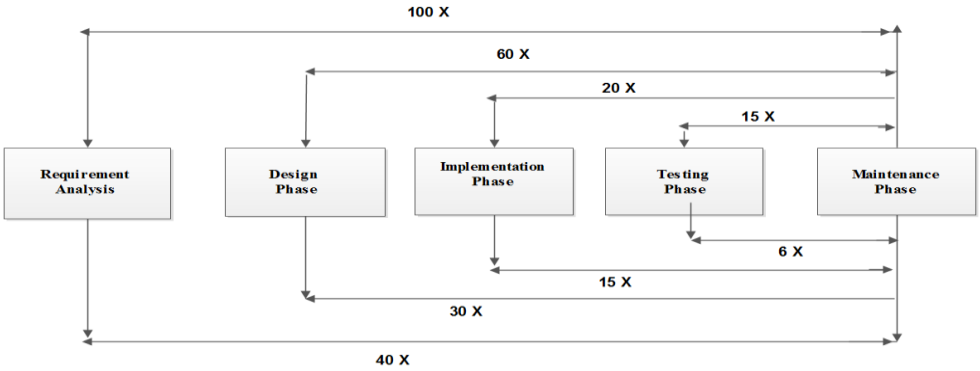


Figure 2.6: Cost to detect and correct defects in different development phases (Nair Suma, Kumar, 2011)

2.7 A Chronological List of Software Inspection Related Articles

The relevant articles on software inspection published from 1976 to present are listed chronologically in Table 2.2.

Table 2.2: A Chronological List of Software Inspection Articles from 1976 to 2015

No	Year	Author	Title of Article	Focus	Resource Title
1.	1976	Fagan	Design and Code Inspection to Reduce Errors in Program Development	Formal inspection	*IBM Systems Journal*
2.	1984	Weinberg e and Freedman	Reviews, Walkthroughs, and Inspections	Technical and project review policies, From testing to debugging	*IEEE Transactions on Software Engineering*
3.	1986	Fagan	Advances in Software Inspections	Advance in formal inspection	*IEEE Transactions on Software Engineering*
4.	1989	Dunham	V&V in the Next Decade	Technology, tools and evaluation of V&V	*IEEE Software*
5.	1989	Ackerman Ackerman, Buchwald, and Lewski	Software Inspections: An Effective Verification Process	Inspection metrics and defect types	*IEEE Software*

6.	1992	Chillarege et al.	Orthogonal Defect Classification-A Concept for In-Process Measurements	Defect types	*IEEE Transactions on Software Engineering*
7.	1995	Kaplan, Clark, & Tang	Secrets of Software Quality	Defect removal and software quality	*Innovations 40 from IBM*
8.	1997	Porter et al.	An Experiment to Assess the Cost-Benefits of Code Inspections in Large-Scale Software Development	Comparison of different approaches, Inspection experiments	*IEEE Transactions on Software Engineering*
9.	2000	Biffl	Using Inspection Data for Defect Estimation	Capture–Recapture Model (CR) detection profile method (DPM)	*IEEE Software*
10.	2001	Van Genuchten et al.	Using Group Support Systems for Software Inspections	Communication improvement for software inspection	*IEEE Software*
11.	2001	Chernak	Validating and Improving Test-Case Effectiveness	Test cases before release	*IEEE Software*
12.	2002	Houdek, F., Schwinn, T., & Ernst	Defect Detection for Executable Specifications — An Experiment	Inspection practices	*International Journal of Software Engineering & Knowledge Engineering*
13.	2002	Perry et al.	Reducing Inspection Interval in Large-Scale Software Development	Comparison of Web-based inspection using traditional inspection	*IEEE Transactions on Software Engineering*
14.	2002	Antoniol, et al.	Recovering Traceability Links between Code and Documentation	Information retrieval	*IEEE Transactions on Software Engineering*

15.	2002	Fagan	A History of Software Inspections	Recommended three components includes seven steps formal continuous process for inspection success	*Software Pioneers*
16.	2003	Thelin, Runeson, & Wohlin	Prioritized Use Cases as a Vehicle for Software Inspections	Usage-Based Reading (UBR), use cases, scenarios, instead of Checklist-Based Reading (CBR)	*IEEE Software*
17.	2003	Anderson, Reps, & Teitelbaum	Design and Implementation of a Fine-Grained Software Inspection Tool	Software inspection using dependency graph automated code inspection	*IEEE Transactions on Software Engineering*
18.	2003	Parnas & Lawford	The Role of Inspection in Software Quality Assurance	Software inspection process improvement	*IEEE Transactions on Software Engineering*
19.	2003	Xu	Making Software Timing Properties Easier to Inspect and Verify	Pre runtime schedule for the inspection of large software	*IEEE Software*
20.	2004	Miller & Yin	A Cognitive-Based Mechanism for Constructing Software Inspection Teams	Cognitive team selection for inspection process	*IEEE Transactions on Software Engineering*
21.	2004	Shore	Fail Fast	Global error handler Global exception handler	*IEEE Software*
22.	2004	Kelly & Shepard	Task-Directed Software Inspection	Light inspection process (TDI) with focus on individual work of inspectors.	*Journal of Systems and Software*
23.	2004	Yin, Dunsmore, & Miller	Self-Assessment of Performance in Software Inspection Processes	Presenting a subjective defect estimation for the number of remained defects	*Information and Software Technology*
24.	2004	Leite et al.	Scenario inspections	Inspection practices	*Requirements Engineering*

25.	2005	Armour	The Unconscious Art of Software Testing	Software test cases	*Communications of the ACM*
26.	2005	Ruhe & Saliu	The Art and Science of Software Release Planning	Software delivery planning	*IEEE Software*
27.	2005	Williams & Hollingsworth	Automatic Mining of Source Code Repositories to Improve Bug Finding Techniques	Using user report in inspection process to improve debugging as well as development	*IEEE Transactions on Software Engineering*
28.	2005	Freimut & Vollei	Determining Inspection Cost-Effectiveness by Combining Project Data and Expert Opinion	Inspection cost	*IEEE Transactions on Software Engineering*
29.	2005	Remillard	Source Code Review Systems	Code inspection tools, Check-in spot inspections, Recommended Code striker at Solaris.	*IEEE Software*
30.	2005	Huhns & Singh	Service-Oriented Computing: Key Concepts and Principles	Case tools, Framework in open environment	*Internet Computing, IEEE*
31.	2006	Faraj & Sambamurthy	Leadership of Information Systems Development Projects	Team efficiency in software development	*IEEE Transactions on Engineering Management*
32.	2006	Tyran	A Software Inspection Exercise for the Systems Analysis and Design Course	Relationship between system analysis and software inspection, inspection practices	*Journal of Information Systems Education*
33.	2006	Runeson et al.	What Do We Know about Defect Detection Methods?	Summarized empirical studies on inspection and defect detection	*Software, IEEE*
34.	2006	Zheng et al.	On the Value of Static Analysis for Fault Detection in Software	Statistical model to evaluate the defect removal efficiency, Economics of defect detection	*IEEE Transactions on Software Engineering*

35.	2006	Lange, Chaudron, & Muskens	In Practice: UML Software Architecture and Design Description	UML model defects and metrics	*Software, IEEE*
36.	2007	Denger & Shull	A Practical Approach for Quality-Driven Inspections	Developer's role in product quality, Importance of inspection techniques training	*Software, IEEE*
37.	2007	Vodde	Experiences in Software Inspection Measurements	Inspection metrics	*Software Quality Professional*
38.	2007	Jalote, Mittal, & Prajapat	On Optimum Module Size for Software Inspections	Reduce inspection cost via optimum size of modules	*International Journal of Reliability, Quality and Safety Engineering*
39.	2008	Bertrand	Design and Code Reviews in the Age of the Internet	Online Code inspection	*Communications of the ACM*
40.	2008	Carver, Nagappan, & Page	The Impact of Educational Background on the Effectiveness of Requirements Inspections: An Empirical Study	Individual's abilities in software inspection, Non-computer graduates are more efficient	*IEEE Transactions on Software Engineering*
41.	2008	Shull & Seaman	Inspecting the History of Inspections: An Example of Evidence-Based Technology Diffusion	NASA experts' experiences, improvement using professionals; New approaches	*Software, IEEE*
42.	2008	Godefroid et al.	Automating Software Testing Using Program Analysis	Dynamic Test Generation; Automated defect detection for program code	*Software, IEEE*

43.	2008	Hatton	Testing the Value of Checklists in Code Inspections	Inspection Checklist, Formal statistical analysis, weak relationship between experience and defect detection	*Software, IEEE*
44.	2008	Glass	Software: Hero or Zero?	Recommended data validation process for defect prevention	*Software, IEEE*
45.	2009	Kollanus	Experiences from using ICMM in Inspection Process Assessment	Improving inspection practices, Inspection Capability Maturity Model	*Software Quality Journal*
46.	2009	Gjerlufsen, Ingstrup, & Olsen	Mirrors of Meaning: Supporting Inspectable Runtime Models	Develop inspectable systems using hierarchical graphs	*Computer*
47.	2009	Mantyla & Lassenius	What Types of Defects Are Really Discovered in Code Reviews?	Defect type, Comparison of other research work	*IEEE Transactions on Software Engineering*
48.	2009	Koru et al.	An Investigation into the Functional Form of the Size-Defect Relationship for Software Modules	Exponential relation between software size and number of defects	*IEEE Transactions on Software Engineering*
49.	2009	Pothier & Tanter	Back to the Future: Omniscient Debugging	Log-based, breakpoint-based, and Reversible debugging	*Software, IEEE*
50.	2009	Walia & Carver	A Systematic Literature Review to Identify and Classify Software Requirement Errors	Taxonomy of errors, and requirement faults.	*Information and Software Technology*
51.	2009	Kollanus & Koskinen	Survey of Software Inspection Research	A classification of papers in inspection area with an emergent taxonomy of the inspection research	*TOSEJ*

52.	2010	Yang et al.	TESTQUAL: Conceptualizing Software Testing as a Service	Software quality aspects	*E - Service Journal*
53.	2010	Poulding & Clark	Efficient Software Verification: Statistical Testing Using Automated Search	Practical method using statistical testing for automated defect detection	*IEEE Transactions on Software Engineering*
54.	2010	Nair & Suma	Impact Analysis of the Inspection Process for Effective Defect Management in Software Development	Effectiveness of inspection using metrics	*Software Quality Professional*
55.	2010	Spinellis	Software Tracks	Discussion on the variety of automated tools for software development and reduction of defects.	*Software, IEEE*
56.	2011	Shin et al.	Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities	Web-browser case studies to improve security inspection	*IEEE Transactions on Software Engineering*
57.	2011	Shaoying et al.	Formal Specification-based Inspection for Verification of Program	Program code inspection, formal specification	*IEEE Transactions on Software Engineering*
58.	2011	Sumit & Patil	A Practical Experiment in Teaching Software Engineering Metrics	Error measurement, Systematic debugging	*Journal of Computational Simulation and Modeling*
59.	2011	Nair & Nair	Estimation of the Characteristics of a Software Team for Implementing an Effective Inspection Process Through Inspection Performance Metrics.	Inspection performance metrics, Depth of inspection	*Software Quality Professional*

60.	2012	Wilkerson, Nunamaker Jr., and Merce	Comparing the Defect Reduction Benefits of Code Inspection and Test-Driven Development	The higher efficiency of software (code) inspection compared to TTD	*IEEE Transactions on Software Engineering*
61.	2012	Gopalakrishnan et al.	Significance of depth of inspection and inspection performance metrics for consistent defect management in software industry	Comparing efficiency between testing and the inspection process. Inspection is better at finding defects.	*IET Software*
62.	2012	de Mello et al.	Checklist-Based Inspection Technique for Feature Models Review	Proposed checklist-based inspection technique (FMCheck) Software Product Line Engineering. It is configurable and applicable on several extensions of the original feature model notation.	*Software Components Architectures and Reuse (IEEE Proceeding)*
63.	2013	Souza et al.	Evidence of Software Inspection on Feature Specification for Software Product Lines	Gathering evidence about the effects of applying inspection to feature specification for SPL. Recommends using sub-domain risk as on indicator for sub-domains priority in the inspection activities.	*Journal of Systems and Software*
64.	2013	De Sousa, Coelho, Braga, & Ambrósio	System Dynamics Model for Simulation of the Software Inspection Process	Introduces a dynamic model to facilitate inspection simulation based on predefined scenarios	*ACM SIGSOFT Software Engineering Notes*
65.	2013	Ali et al.	An Improved, Efficient and Cost Effective Software Inspection Meeting Process	Recommendations for better meeting and follow-up activity of inspection	*International Journal of Modern Education and Computer Science*

66.	2013	Fernandez, Abrahão, & Insfran	Empirical Validation of a Usability Inspection Method for model-driven Web development	Inspection method for Web artefacts	*Journal of Systems and Software*
67.	2014	Misra, Fernández, & Colomo-Palacios	A Simplified Model for Software Inspection	Development of an asynchronous tool to facilitate the inspection process.	*Journal of Software: Evolution and Process*
68.	2014	Chandani & Gupta	A Survey on Effective Defect Prevention - 3T Approach	Defect analysis and defect tracking	*IJIEEB*
69.	2014	Dittrich	Software Engineering Beyond the Project – Sustaining Software Ecosystems	A qualitative research shows that in software domain, a continuous development is preferred to project management. Major improvements in SE needed, to support software ecosystems. Local designer in company and third-party (from outside of project) have collaborated in the design phase	Information and Software Technology
70.	2014	Gimpel	Software That Checks Software: The Impact of PC-lint	Development of automated tool for software test and software inspection the software with defining its behaviour via a scenario and classifying the corresponded defects	Software, IEEE

71.	2014	Shen, Zhao, & Han	On Inspection Strategy Based on Sample Inspection Reliability.	Redefine the reliability with multi aspect factors and defining the inspection strategy based on it	Applied Mechanics and Materials
72.	2014	Chen, & Agrawal	Special issue on Emerging Topics on Software Debugging.	Debugging from an intelligent mufti agent landscape	Journal of Systems and Software
73.	2015	Huzooree, & Devi Ramdoo	Evaluation of Code Inspection on an Outsourced Software Project in Mauritius.	Defining the criteria for debugging evaluation through case study	IJCA
74.	2015	Sneed, & Verhoef	From Software Development to Software Assembly	The software development approach with highlighting the importance of software test for successful implementation	*Software, IEEE*
75.	2015	Mahmoud, Haggag, & Abd	Cost Analysis of a Two-Unit Cold Standby System Considering Hardware, Software Failures and Inspection with Maximum Repair Time.	Classification the failures to find new analysis approach for cost analysis	IJCA
76.	2015	Gomes & Moita	On the Validation of a Specific Development Process for Scientific Software using the Inspection Technique	Focusing on inspection techniques for process phases validation as well as the corresponded work products.	
77.	2015	Malhotra, & Taneja	Comparative Analysis of two Stochastic Models subjected to Inspection and Scheduled Maintenance.	A pragmatic research to compare the advantages and shortcoming the software inspection and maintenance plan	IJSEIA

78.	2015	Li, X., Mutha, C., & Smidts	An automated software reliability prediction system for safety critical software	Focusing on the reliability prediction and evaluation to improve the quality of software with unique and special features	Empirical Software Engineering
79.	2015	Minetola, Iuliano, & Calignano	A customer oriented methodology for reverse engineering software selection in the computer aided inspection scenario.	The reverse approach to promote the quality of software development through the inspection technique	Computers in Industry
80.	2015	Goswami, Walia, & Singh	Using Learning Styles of Software Professionals to Improve Their Inspection Team Performance.	Focus on different learning philosophy, application, and mechanisms for improving the effectiveness and productivity of inspection teams	International Journal of Software Engineering and Knowledge Engineering
81.	2015	Vitharana	Defect propagation at the project-level: results and a post-hoc analysis on inspection efficiency	Study the potential shortcoming in inspection process specially in amplification the debugs in software life cycle from one phase to next phase	Empirical Software Engineering

One of the major problems highlighted by many researchers, is that not many enterprises know how to conduct inspection. Another problem is the lack of research pertaining to handling the inconsistencies. Some inspection processes recommend brainstorming, negotiation, and collaboration as a part of the inspection process. This means soliciting different views from the inspectors on how the potential defects are to be handled if they could not be resolved by brainstorming, the issue. Researchers have not been able to provide a definitive solution for such a case, until now.

There is no justification for companies to save cost by using traditional testing rather than by inspections to reduce defects (Radice, 2002, Mahmoud, Haggag, & Abd, 2015). In a comprehensive study, Mishra and Mishra (2009) found that" most organizations do not use inspections in the software development process as the inspections are too rigorous, and even with the support of computer, it is too complicated for organizations to implement".

Recent researches on how software inspection has evolved over the years, show that efforts to eliminate the inspection meeting in order to enhance efficiency can have adverse effects such as an increase in false positives. Although many studies have shown that no-meeting inspection processes are more efficient compared to meeting-based inspections, none of these studies had provided answers for solving the false positive issue (Misra, Fernández, & Colomo, 2014). In any occurrence of false positives, inspectors identify non-defects to be defects (Land, Sauer, & Jeffery, 1997). Also the tools developed for the new inspection processes lack coordination support (Zhang & Babar, 2013).

In software inspection, a long-standing problem that still remains after three decades concerns the choice, evaluation, and appraisal of inspectors for an inspection process (Winkler, Thurnher, & Biffl, 2007; Valentim & Conte, 2014). None of the existing

inspection processes address the selection method, or qualification needed of an inspector, or even recommend a placement test to assess their abilities. Most studies on software inspection are based on some experiments and surveys, involved small artefacts, and none on the large projects or very large artefacts. Wilkerson, Nunamaker Jr., and Mercer (2012) stated that to validate the findings from the studies as well as the inspection process, further research is necessary both inside and outside of the laboratory.

2.7.1 The subjects covered in the articles reviewed

Table 2.3 shows the subjects covered in the articles listed in Table 2.2. The subjects covered in the articles are relevant to this research. The table also shows the number of articles that include each subject in their contents.

Table 2.3: Number of articles that contain the subjects relevant to the inspection process

	Subject covered					
	Inspection	Checklist	Defect Classes	Defect detection in Requirements Phase	Defect detection in Design Phase	Defect detection in Program Code
No. of articles	51	11	6	28	34	24
Percentage	33 %	7 %	4%	18%	22%	16%

Of the total of 154 articles reviewed, only 24 (16%) focused on code inspection. Today, automated tools are used for code inspection, as well as online inspection. Inspection in the requirements analysis and design phases, and inspection efficiency have remained important issues over the past three decades. Also, 51 (33%) of the articles listed above, focus on software inspection process; 28 (18%) on defects of the requirements analysis phase artefacts, and 34 (22%) on defects detected in the artefacts of the design phase.

Only 6 (4%) of the articles reviewed discussed defects classification. It must be noted that defects classification is crucial for the success of the inspection process. Most of the studies used a sub-class of the IEEE defects classification.

Inspection is a team activity and the productivity of the team is another important issue for a successful inspection process (Sneed, & Verhoef, 2015). Team structure, and the productivity of the different compositions of semi-skilled, skilled, and highly skilled inspectors are not covered in the studies. Also, removing the inconsistencies identified are not covered. Brainstorming and negotiation sessions conducted during inspection meeting to resolve ambiguous issues also have to be addressed.

2.8 Software Inspection Processes for the Requirements Analysis and Design Phases

Ad hoc, Checklist, and Scenario are three common classes of methods for software inspection in the requirements analysis and design phases (Minetola, Iuliano, & Calignano, 2015). Ad hoc and Checklist are two frequently-used fault detection methods. Studies have proven that traditional checklist-based methods do not perform better than the Ad hoc method (Grady & Slack, 1994). Inspection meetings to discuss individual checklists have been futile, inflexible and ineffective (Malhotra, & Taneja, 2015). There is no professional cooperation, on exchange of knowledge, opinions and experiences in the checklist method.

The use of the Scenario method results in 35% better efficiency than the Checklist and Ad hoc methods (Hatton, 2008). Although this method allows the inspectors to concentrate on specific features of the defects, other studies have shown that the Scenario method is not applicable for all defects (Minetola, Iuliano, & Calignano, 2015). The success of software inspections using the Scenario method, depends on the type of Scenario designs (Gomes, & Moita, 2015). Thus, despite the inspectors' efforts, poorly-

designed Scenario methods cannot perform satisfactory during inspections (Minetola, Iuliano, & Calignano, 2015).

2.8.1 Software Inspection Processes Applicable in the Requirements Analysis Phase

This section discusses the current inspection processes for artefacts in the requirements analysis phase.

2.8.1.1 Structured Walkthroughs

Weinberg and Freedman (1984) conducted a comprehensive study on reviewing, walk-through, and inspection. They emphasized that the Formal Technical Review (FTR) must be within the scope of responsibility for those not involved in software production. Hence, the main role of FTR is to provide reliable information on the software products to managers.

In 1989, Yourdon introduced a new software inspection technique called Structured Walkthroughs that allows very short preparation time and meeting time. The method is appropriate for use in the requirements analysis phase. The important feature of Yourdon's technique is its focus on all types of documents generated during the requirement analysis phase.

2.8.1.2 Phased Inspection

The Phased inspection method could be considered a combination of traditional, active design reviews, and multiple team inspection. This technique developed by Knight and Myers (1993), is aimed that not only finding the defects but also to examine the quality features of software such as reusability and portability. The artefacts will be inspected over six or more sequential phases.

A simple inspection may consist of up to six phases, which are conducted sequentially. Each phase is designed to examine a specific property of an artefact. Completion of a phase and correction of all defects found is important in order to start the next phase. In this method, a single checklist is used by a single inspector, prior to using different checklists by several inspectors.

2.8.1.3 Inspection without a Meeting

Mashayekhi et al. (1993) stated that face-to-face communication in inspection meetings is quite expensive and does not produce any significant results in defect finding. Eick et al. (1992) and Votta (1993) found that 90% of the defects could be identified during inspection, and before the inspection meeting. Porter et al. (1995) opined that inspection meeting is too time-consuming and not popular with project managers. They also viewed that gathering several inspectors, authors, and the moderator in one place is a waste of time and resources.

Votta (1993) also stated that because of the inevitable sequential order in inspection processing, only two of the inspectors actually have face-to-face interaction in any inspection with *n* inspectors participating. He also opined that the inspection meetings are not cost effective, and recommends a replacement meeting (depositions) with an author and an inspector. His studies show that depositions are more effective when compared to traditional inspection meetings.

2.8.1.4 Inspection with Brainstorming Session

Gilb and Graham (1993) suggested adding an extra step - process-brainstorming meeting to improve the inspection process, after the inspection meeting. The weakness of short meetings is that misunderstanding will be resolved by applying the aforementioned step. Without strong leadership during inspection, confusion and anarchism might arise.

2.8.1.5 Software Inspection Process Applicable in the Design Phase

The current inspection methods for the design phase, are presented in the following sections.

2.8.1.6 Active Design Review

The active design review method was introduced by Parnas and Weiss in 1985. The method involves several small reviews instead of a big review. Each review focuses on an important part of an artefact. There are three classes of defects: i) inconsistency, ii) inefficiency, and iii) ambiguity. The artefacts are classified based on their properties. Each inspector has to pass a qualification exam to ensure that he/she is fully competent to inspect specific part of the artefact. This is because each inspector is assigned to find a particular class of defects. To guide the reviewers, the errors are classified. Figure 2.7 shows the steps of Parnas and Weiss method.

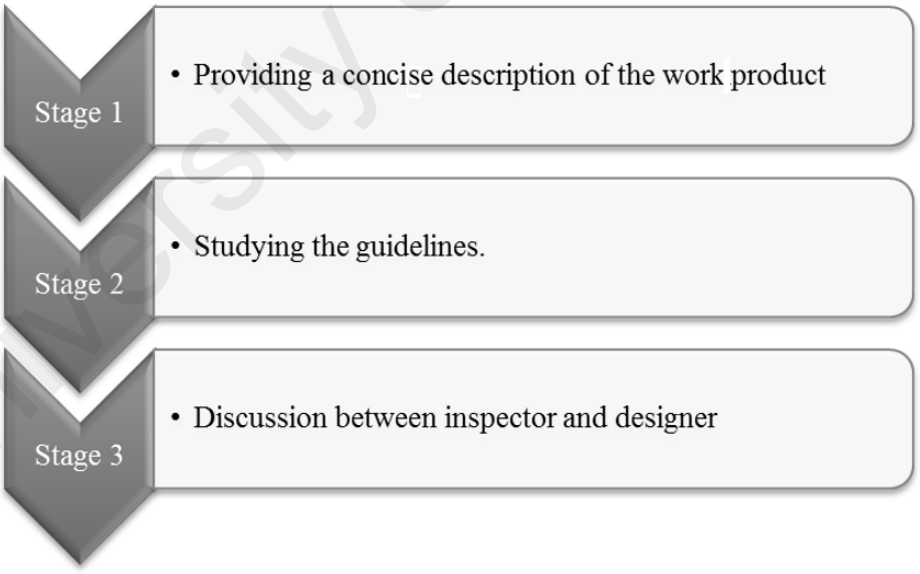


Figure 2.7: Inspection Session Stages in the Parnas and Weiss method (1985)

2.8.1.7 Two-Person Formal Inspection Method

Bisant and Lyle (1989) presented a formal method for software inspection in the design phase. In this method, the moderator's role is removed and the team size is reduced. A peer-to-peer relationship is fostered to make the review meeting more efficient. The results from researches indicated that the method is useful for semi-skilled software developers in small organizations. Fostering close relation between an author and a reviewer can improve the productivity of the inspection process.

2.8.1.8 High-Level Object-Oriented Designs Inspection

Figure 2.8 illustrates the different techniques for reading in two horizontal and vertical axes. The horizontal reading is applied within a particular software development phase, while the vertical reading compares requirements documents between the software development phases.

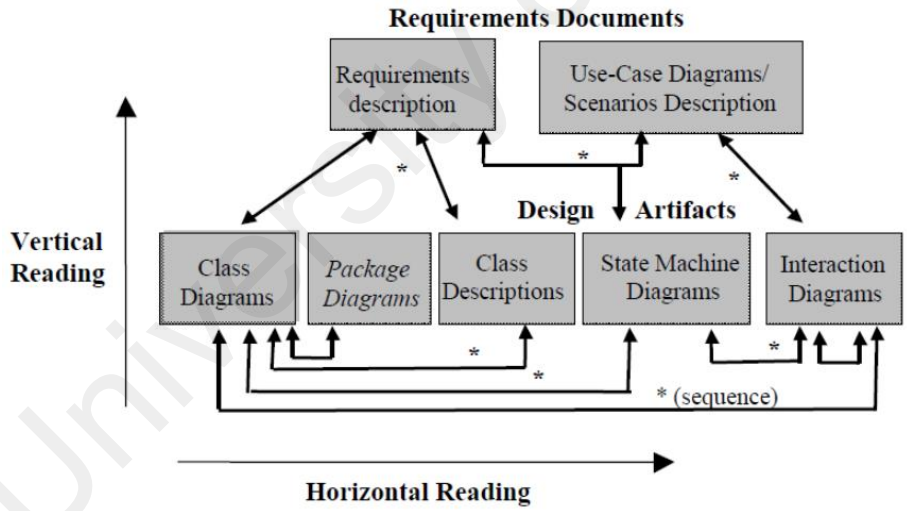


Figure 2.8: Techniques for Reading in Two Axes

2.8.1.9 Usage-Based Testing (UBT-i) Method

Winkler, Riedl, and Biffl (2005) developed a paper-based inspection process to integrate the desk test and the inspection methods. The process is applicable to the design specifications. The most important feature of UBT-i is the defect location characteristics

information that could be used in the design specifications. The major stages of the UBT-i method are shown in Figure 2.9.

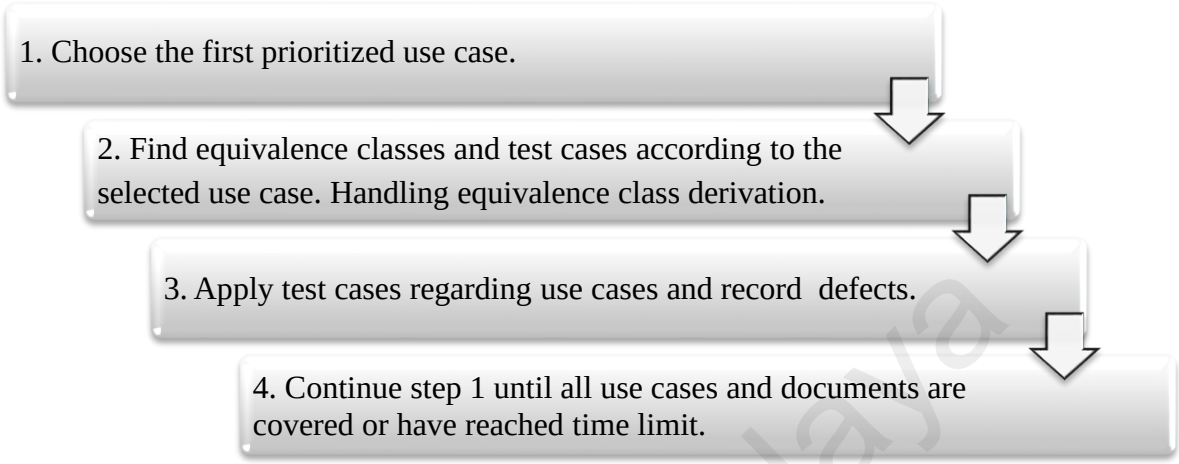


Figure 2.9: The Major Steps of UBT-i Inspection Method

Generating the Test-case is an essential task of the UBT-i process. Finally, the defects are detected in the design phase of the software development based on the priority of each use case. Figure 2.10 shows the inspector's responsibilities in the UBT-i process.

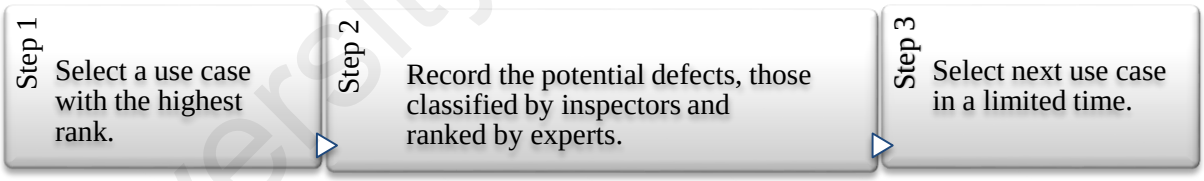


Figure 2.10: Inspector Responsibilities in UBT-i Approach

2.8.1.10 Multiple Team (N-fold) Inspection Method

The major feature of N-fold inspection is the use of N independent inspection teams. Kantorowitz, Guttman, and Arzi (1997) stated that the N-fold inspection could be used for any type of inspection and for any size of systems, provided there is an optimal number of members in the inspection team.

Using small teams provides an opportunity for finding more defects than using a single big team. Therefore, small teams are more efficient compared to big inspection teams. However, multiple teams can be used to identify defects which might not be found by a single team. Different teams with various proficiencies will find different defects of different classes.

In this method, the moderator is responsible for coordinating the teams. Each inspection team follows the formal software inspection procedures. It is important to achieve a balance between the cost of using more teams and the benefits gained from finding more defects (Tripp, Struck, & Pflug, 1991).

Table 2.4: Common Problems Encountered when using Inspection Methods in the Analysis Phase

Problem	Explanation	Reference
Checklist consequences	• Checklists facilitate the inspection process but have limitations • No flexibility	Hatton (2008)
Time measurement	• Measuring the time is sometimes important but sometimes applied by combining human forces (person–month). In both cases, the measurement process will encounter some difficulties • Measurements obtained from virtual meetings are very different from that obtained from normal sessions • E-mail inspections are less efficient but accelerate the inspection process	Vodde (2007)

Martin and Tsai (1990) opined that the N-fold inspection technique is appropriate to be applied in the requirements analysis and design phases of the software development lifecycle. Tripp, Struck, and Pflug (1991) found that this technique generates low redundancy in defects among the inspection groups. In their research, Schneider et al. (1992) found that N teams detected about 80% of potential defects whereas a single

inspection team could not find more than 35% of those defects. Other studies have found that there is no overlapping among the defects that different teams have detected. The limit on the number of inspection teams is a crucial issue that had been addressed by many researchers (Martin & Tsai, 1990).

2.9 Problems of Using Current Inspection Methods in the Analysis and Design Phase

The problems associated with the current inspection methods in the requirements analysis phase can be divided into two classes - common and specific problems - as explained in the next sections:

2.9.1 Common Problems Encountered when Using the Inspection Methods in the Requirement Analysis Phase

The traditional inspection approaches cannot meet today's inspection requirements. The size and complexities of software, especially in the early phases of development, warrant the use of more efficient inspection methods. A shortcoming of the current inspection methods is the lack of facilities to support the inspection teams. The organisation, formation, and characteristics of the inspection teams are also new concern with regard to the modern inspection approaches (Nair, Suma, Kumar, 2011). These issues have been ignored in the traditional approaches. Table 2.4 summarises some important common problems of current inspection methods in the analysis phase.

2.9.2 Specific Problems Encountered when Using the Inspection Methods in the Requirements Analysis Phase

Specific problems have been encountered when using software inspection methods in the requirements analysis phase. As a result, it restricts the use of these methods in various software types, and in projects of different disciplines (Grady & Van Slack, 1994). Table

2.5 summarises the problems encountered when using the different inspection methods in the requirements analysis phase.

Table 2.5: Specific Problems Encountered when using Inspection Methods in Requirements Analysis Phase

Method	Problem	Reference
Formal Inspection	• Failure to find defects in the product developed • Information overload in the preparation phase • Information is not relevant • Reviewers are not aware of the design objectives • Evaluation freeze • Production freeze	(Shull, 2002)
Phased Inspection	• Finding sequence of defect is complicated • Comprehensive checklists are essential, but designing is arduous • Dependencies among the defects make this technique difficult	(MacDonald, 1995)
Structured Walkthrough	• Focusing only on the positive aspects of documents while negative aspects are ignored	(Weinberg & Freedman, 1984)
Inspection with Brainstorming Session	• Confusion and anarchism could happen without strong inspection leadership • Brainstorming may result in an infinite loop with fruitless arguments	(MacDonald et al., 1995)
Inspection without a Meeting	• The method is efficient but not effective • Some serious defects will remain undetected	(Perry et al., 2002)

Table 2.6: Common Problems of Inspection Methods Used in the Design Phase

Problem	Explanations	Reference
Team Work in Software Inspection	• Finding the appropriate number of inspectors for an inspection team • No formal sharing knowledge and experience • Periodical training for the inspection team can incur unexpected costs • The current methods neglect experience sharing because no collaborative infrastructure is provided	(Boehm, Basili, 2005)
Team Characteristics	• Effective implementation of the inspection process is entirely dependent on correct software team features • Software developers employ people who have little experience in inspection and estimation, and this does not justify the cost incurred to produce quality software	(Nair et al., 2011) (Spiewak, & McRitchie, 2008)
Forming the Inspection Team	• The composition of the team is emphasised to ensure success of the inspection process • Ensuring that both the number of people in a team and their capabilities, potential, experience, and the skills meet the requirements • The composition of the team and the capabilities of each inspector does not match	(Miller & Yin, 2004)
Eliciting the Requirement and Team Training	• Paradox in qualification and requirement of inspection team members (the inspectors without computer science background discovered more defects in comparison with those with computer science qualification • Inspectors with Masters or PhD degrees or industrial experience or related expertise do not perform as remarkably as expected in the inspection • Inspectors only have experience in documenting the requirements or analyzing the systems	(Carver, Nagappan, & Page, 2008)

2.9.3 Common Problems Encountered when Using the Inspection Methods in the Design Phase

All inspection methods used for the software design phase have some common weaknesses. Table 2.6 summarises the common shortcomings of current inspection methods used in the design phase.

2.9.4 Specific Problems Encountered when Using Inspection Methods in the Design Phase

Every software inspection method used in the design phase has certain weaknesses and limitations (Grady & Slack, 1994). Table 2.7 summarises the specific problems encountered using each inspection method in the design phase.

2.10 Standard Classification of Software Defects

The defects found at the requirements analysis and design phases can be classified as defect, failure, error, bug, etc. The following section gives the definition of these defects.

2.10.1 Defect Classification in the Requirements Analysis Phase

Different researchers have proposed different classifications for defects in the requirement analysis phase of software development lifecycle. Table 2.8 shows the defects found in the requirements analysis phase during the software inspection process and classified based on the IEEE (Std. 1044-2009) standard.

Table 2.7: Specific Problems Encountered when using Inspection Methods in the Design Phase

Method	Problem	Reference
Active Design Review	• No quantitative measurement • Lack of integration • Difficulty in managing many meetings • Defect management is not effective	(Parnas & Weiss, 1987)
Two-Person Formal Inspection Method	• Not applicable for large systems • Suitable only for semi-skilled developers in small projects	(Bisant & Lyle, 1989)
Multiple Team (*N*-fold) Inspection	• Difficulties faced in multiple team arrangement • Additional team costs • Potential hidden expenses for defects not detected • Difficult in finding optimal number of members	(Tripp et al., 1991), (Schneider, Martin, & Tsai, 1992)
UBT-i	• The gap between requirements analysis and class definition • Only applicable to the class specification of UML design model • Professional prioritization of classes is needed	(Winkler, Riedl, Biffl, 2005)
High-Level Object-Oriented Designs Inspection	• Poor transition definition from one phase to the next phase • Lack of integration among the phases • Weakness in coordination, and time-management	(Travassos et al., 1999).

Table 2.8: Software Defects in Requirement Analysis Phase classified based on (IEEE Std 1044-2009) Standard

Attribute	Phase	Definition	Example
Insertion activity	Requirements	Defect inserted during requirements definition activities (e.g., elicitation, analysis, or specification)	• Function required to meet customer goals omitted from requirements specification • Incomplete use case specification • Performance requirements are missing or incorrect • Security requirements are missing or incorrect • Function incorrectly specified • Function not needed to meet customer goals specified in requirements specification
Type	Logic	Defect in decision logic, branching, sequencing, or computational algorithm, as found in natural language specifications	• Incorrect sequencing of operations
Mode	Wrong	Something is incorrect, inconsistent, or ambiguous.	• Ambiguous definition of business rule in specification
	Missing	Something is absent that should be present.	• Missing system response in sequence diagram
	Extra	Something is present that should not be present	• Some attributes (fields / variables), switches or branches are never used

Table 2.9 shows the general types of defects classification in the analysis phase, as defined by Hong et al. (2008).

Table 2.9: Types of Defects in the Analysis Phase (Hong et al., 2008)

Type of defect	Explanation
Consistency	Little consistency between the previous artefacts and the current artefacts.
Function	Defects that affect the functionality due to incorrect functional explanation,wrong algorithm, data structure, etc.
Standards	Little observance of the rules such as customer's standards, project standards methodology, coding rules, etc.
Performance	Defects that influence the performance due to incorrect design, inefficient algorithm, data structure, etc.
Miscellaneous	Defects not categorized by the aforementioned types

Chillarege et al. (1992) proposed the orthogonal defect classification (ODC). Table 2.10 shows a complete classification of potential defects in the analysis phase.

Table 2.10: ODC of Defects in the Analysis Phase (Chillarege et al., 1992)

Defect Class	Description
Missing Functionality	The system requirements specification does not contain the necessary information to clarify the system internal operational behaviour.
Missing Performance	The description of the performance specifications does not satisfy the acceptance testing criteria or is not available.

Missing Environment	Environmental resources such as database, skilled and semi-skilled staff, hardware, and software are not stated in the system requirements specification.
Missing Interface	The interaction and communication scenario, procedures, and instructions showing how the proposed system communicates with the objects out of the system boundaries are not clarified.
Incorrect Fact	The false sentence or description, or condition, or situation affects the validity of the system requirements specification.
Wrong Section	The placement of some essential data and information is not correct or misplaced in the system requirements specification.
Ambiguous Information	Misunderstanding happens because of missing a term or important sentence. Incorrect or incomplete behaviour definition could be another reason for confusion.
Inconsistent Information	Two or more sentences in the system requirements specification are contradictory or refer to some actions in an inconsistent manner.

2.10.2 Defects Classification in the Design Phase

The standard classification for software defects in the design phase adopts the NASA (2013) standard, as shown in Table 2.11.

According to Chaar et al. (1993), the design phase of the software development lifecycle could be categorised into dynamic or static classes (see Table 2.12).

Table 2.11: Software Defects in Design Phase (NASA, 2013)

Type	Description
Algorithm	Invalid logic algorithms
Cohesion	Inappropriate cohesion
Component	Bad structure of component does not meet requirement
Coupling	Inappropriate coupling
Data structure	Incomplete data structures
Functionality	Incomplete general function of module
I/O interaction	Invalid I/O interface
Inconsistency	The design decisions do not match system objectives
Interface	Incompleteness module interface
Module	Non-modular design
Notation	Non-standard notations
Pattern	Design pattern is not suitable
Relationship	Invalid modules relationships
Requirement related	Design specification is not match with its relevant requirements
Reusability	Bad selection of the reusable components
Self-instrument	Invalid fault detection
Traceability	Non-traceable design for ensuring validity

Table 2.12: Design Defects Adopted (Chaar, 1993)

Defect Type	Description Definition
Algorithm	The sequence of activities or use of the data type or the number of occurrences and loops is incorrect
Assignment	The value is missing or assigned incorrectly

Build/Package/ Merge	The predefined library objects, methods, routines, or modules have problems or are not compatible
Checking	Type mismatch of parameters or invalid parameters in a conditional expression
Documentation	Inconsistencies in the description and the real behaviour of the components
Function	The functionality requirement is not satisfied
Interface	The interaction between the components is not correctly defined or is not stated
Timing	In the sharing of resources, correctness of serialization is not met or an incorrect technique is implemented

2.11 Findings of the Defect Classification in the Requirements Analysis Phase and Design Phase

Different classification methods have been proposed by different institution or researchers. Basically defects are classified according to the source of the defect, trigger of the defect, the mode of the defect, etc. There is no standard or systematic classification for defects. Therefore, it is necessary to establish a standard for defects classification.

Based on the comparison of the different defects classifications, defects can be categorised into six major modes – missing, incorrect, inconsistent, ambiguous, wrong section, and extra. In this research, a common set of defect classes is defined, and is related to these six modes for the requirements analysis phase, as shown in Table 2.13. To facilitate the inspection process, each defect is assigned a defect code. This helps in analysing and identifying the defect types that are most common in the analysis and design phases for a specific type of system.

Table 2.13: Software Defects Classification in the Design Phase (IEEE Std 1044-2009)

Mode	Code	Definition	Example
Ambiguous	(A)	Misunderstanding caused by a missing term or important sentence.	• Definition of a discount rule in a business plan without explanation on how it is to be done.
Wrong Section	(WS)	The placement of some essential data and information is not correct or misplaced in the system requirements specification.	• Placing authentication function after authorization function in requirements specification.
Extra	(E)	Something is present that should not be present	• Asking for details of a user in a cardinality schema.
Missing	(M)	Something is absent which should be present.	• Absence of encryption method in a safe transportation mode.
Incorrect	(IC)	Something is wrong or description is not accurate under specified conditions.	• Updating a profile when the necessary information is not provided.
Inconsistent	(ICST)	Descriptions contradict each other or expressed actions that cannot both be correct or cannot both be carried out.	• Allowing a method to send a message that should be verified at the destination, and concurrently expecting to receive only verified requests from the destination.

2.12 Missing Aspects or Gaps

The literature review in this chapter shows some gaps and shortages of software inspection process which need to be focused to enhance the quality of software inspection. Some gaps and issues are listed below:

- Develop the checklists is difficult and have some limitations.

- Measuring the time will encounter some difficulties.
- Measurements obtained from virtual meetings are not considered.
- Failure to find defects in the product developed.
- Not relevant and overload information overload in the preparation phase.
- Lack of documentation of the cause and effect of the potential defects detected during the inspection process.
- Lack of comprehensive defect classification
- Brainstorming may result in an infinite loop with fruitless arguments
- Unable to determine the appropriate number of inspectors for an inspection team.
- The current methods neglect experience sharing because no collaborative infrastructure is provided.
- No formal sharing of knowledge and experience (i.e. inspector profile review).
- Preparation and development kit are not defined carefully.
- The composition of the inspection team is not considered to ensure success of the inspection process.

2.13 Summary

The formal inspection approach that had been used for more than three decades is not effective for current software and development inspection processes, especially in the requirements analysis and design phases. The current inspection tools focus on code inspection, but inspection at the early phases of software development lifecycle is more

important. Studies on software inspection and the inspection process have shown that customizing the inspection process can increase its effectiveness as well as its efficiency.

There are common problems encountered in all inspection processes in the requirements analysis and design phases (Kollanus, 2009). Inspection is not done in most of the software projects and this results in software of lower quality and lower reliability. In most software inspection processes, the productivity of inspectors is crucial to the overall productivity of the inspection team, hence, if the inspectors are not skillful or competent, the quality of inspection would be adversely affected. However, the inspection team structure as well as team productivity have not been addressed in the studies.

Some researchers opined that a software inspection process without comprehensive teamwork will not be efficient. Finding more defects in less time is the main goal of any inspection process. The studies reviewed show that developing an application to record the potential defects facilitates the inspection process. Finally, the use of defect detection performance criteria and quality metrics proposed by inspection experts have provided the motivation to design an enhanced and effective inspection process to improve the Formal inspection process.

Over the last four decades since the introduction of the formal inspection process by Fagan, different software inspection processes and tools have been introduced. However, none of the formal processes introduced is any better in efficiency. The formal process is still recognized as the standard model for NASA and IEEE. Misra Fernández, and Colomo (2014) and Kasai, Morisaki, and Matsumot (2013) introduced some prototype models and tested them in student projects (Hussain, 2007), but these prototypes were not operational processes as they have not been shown to be more efficient in detecting defects or in finding a larger number of defects compared to the formal process.

CHAPTER 3: RESEARCH METHOD

This chapter presents the methods used to carry out the research. It also discusses the research activities, research design, the metrics used to evaluate the proposed software inspection process, ISIP, and elaborates on the internal and external validity of the research.

3.1 Research Method Used

This research is aimed at proposing a new and improved software inspection process. The efficiency of the proposed process will then be evaluated using data collected from two case studies. Figure 3.1 shows the activities involved in this research.

A new software inspection process, ISIP, was proposed after conducting a thorough review of the literature pertaining to the software inspection process and as well as related issues. A tool that incorporates the proposed process was developed using agile development technique. It was used to collect data during the case studies to evaluate the efficiency of the proposed inspection process. Before conducting the case studies, the metrics to be used to evaluate the proposed process were defined. The following sections explain the research activities in detail.

3.1.1 Activity 0: Choosing Area Subject

Software inspection is an important process in software engineering. Personal interest and the experiences on software testing and software inspection have inspired the initiation to conduct a study on software inspection.

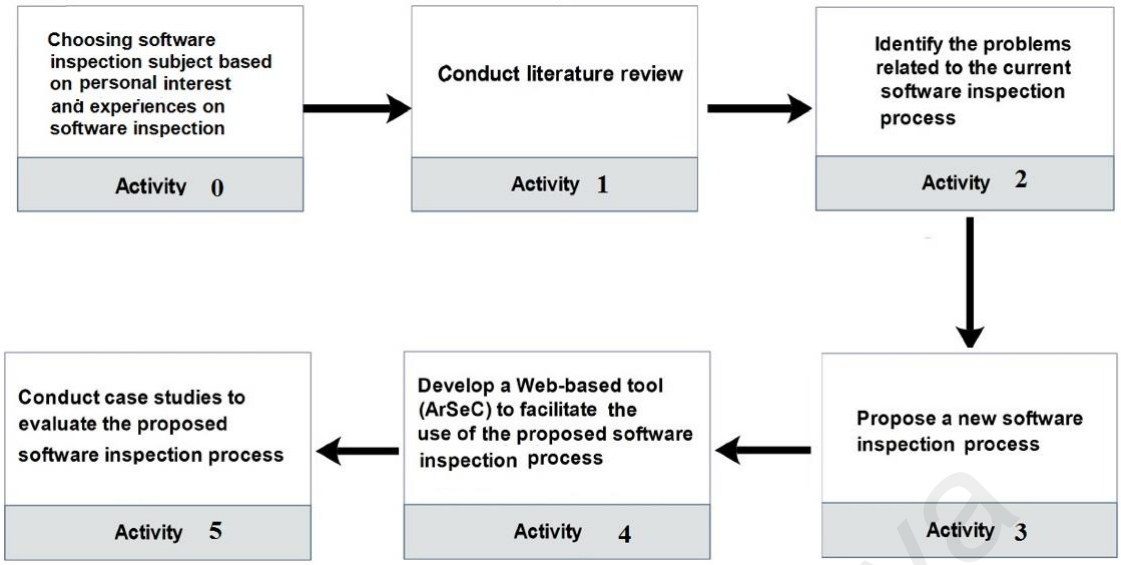


Figure 3.1: Research Activities

3.1.2 Activity 1: Literature Review

A literature review was conducted from August 2009 to July 2015, on published work of researches pertaining to software inspection process, problems encountered during inspection in the requirements analysis and design phases, and the classification of defects detected in the requirements and design artefacts. The literature review was presented in chapter 2. Altogether, 87 articles published between 1976 and 2015 were reviewed. Out of the 87 articles, 35 articles were journal papers, and 10 articles were retrieved from 30 conference proceedings, two reference books, and 10 articles were retrieved from the Internet.

3.1.3 Activity 2: Problems Identification

The literature review provides insight into the main problems and weaknesses associated with the current inspection process. Over the past three decades, software inspection had focused on program code inspection. Less attention was paid to the inspection process, as well as inspection of artefacts in the requirements analysis and design phases. Fagan (2002) highlighted the shortcomings associated with inspecting

artefacts in the early stages of software development. He stated that inspection of artefacts, especially in the requirements analysis and design phases, requires thorough and careful inspection to eliminate as many defects as possible in those early phases (Fagan, 2002). Inspection carried out using paper-based checklists and conducted manually without the use of any automated inspection tools, are very tedious and time-consuming (de Mello et al, 2012, Li, Mutha, & Smidts, 2015). In addition, there is presently no comprehensive method of categorising defects found in the requirements analysis and design phases (Alshazly, Elfatatry, & Abougabal, 2014). This research was initiated to address the aforementioned problems.

3.1.4 Activity 3: Propose an Improved Software Inspection Process (ISIP)

To overcome the shortcomings of the current software inspection process, an improved software inspection process, ISIP, was established. ISIP classifies defects detected in the requirements analysis and design phases of the software development life cycle into different categories based on IEEE's and NASA's defect classification approaches. It also supports virtual inspection meetings, and thus, allows remote inspection to be conducted. ISIP incorporates features which are aimed at improving inspections of the artefacts detected in the requirements analysis and design phases, as well as improving the productivity of the inspection teams. Chapter 4 provides more details on the features of ISIP.

3.1.5 Activity 4: Develop a Web-based tool (ArSeC)

To facilitate the implementation of ISIP, an inspection tool known as *Artefact Session Control* (ArSeC), was developed using agile development techniques in the .NET environment. ArSeC has a database that stores the various types of defects which have been classified according to the IEEE and NASA standards, and commonly detected in the requirements analysis and design phases. The database also stores information on the

potential causes of each defect. This is aimed at helping the author(s) of the artefacts in removing the defects detected quickly. The features of ArSeC are explained in chapter 4.

3.1.6 Activity 5: Conduct case studies to evaluate the proposed software inspection process

Case study is the most appropriate method to validate the proposed software inspection process, because i) the inspectors involved in the process could not be controlled by the researcher, ii) the replication cost, i.e. the cost to repeat the inspection for the other software projects, is reduced, and iii) the data that will be used to evaluate the proposed inspection process are based on the inspection outcomes of past completed projects (i.e. retrospective investigation). The limitations of using case studies to conduct this research are: 1) only two large projects and eight artefacts of the requirements analysis and design phases were inspected respectively, 2) the checklists designed and used are applicable for requirements analysis and design phases only, 3) the knowledge and experiences of the inspectors are based on the highly skilled and skilled levels, only, and 4) only three inspectors were selected to form an inspection team.

Several emails have been sent to software and industrial companies to invite them to participate in current research. Two case studies were conducted to evaluate ISIP. Two software development projects (P1 and P2) from a manufacturing company (C1) and a trading company (C2), respectively, were selected for the case studies. Several emails have been sent to software and industrial companies to participate in current research. In this research, inspection is done only on large artefacts in the requirements analysis and design phases (those had more than 15 pages). Inspectors have chosen based on their experiences of the number of artefact inspected by them and the number of their experience in software inspection, explained in detail in chapter five. The research design adopts a two-group independent design approach whereby two independent inspection

teams – Team A and Team B from C1; Team C and Team D from C2 – were formed to inspect the artefacts, as shown in Figure 3.2. In case study 1, the formal inspection process (Process 1) was first used by three inspectors from Team A to conduct manual inspections on the artefacts of P1 based on the inspection checklists prepared by a moderator. The inspection outcomes – number of defects detected and the inspection time – were recorded using MS Excel. The same sets of artefacts were also inspected by another three inspectors from Team B using ISIP (Process 2). The inspection process was facilitated using ArSeC, and the inspection outcomes were recorded using ArSeC as well. In case study 2, the inspection processes were repeated by two different inspection teams, Team C and Team D on the artefacts of P2, respectively.

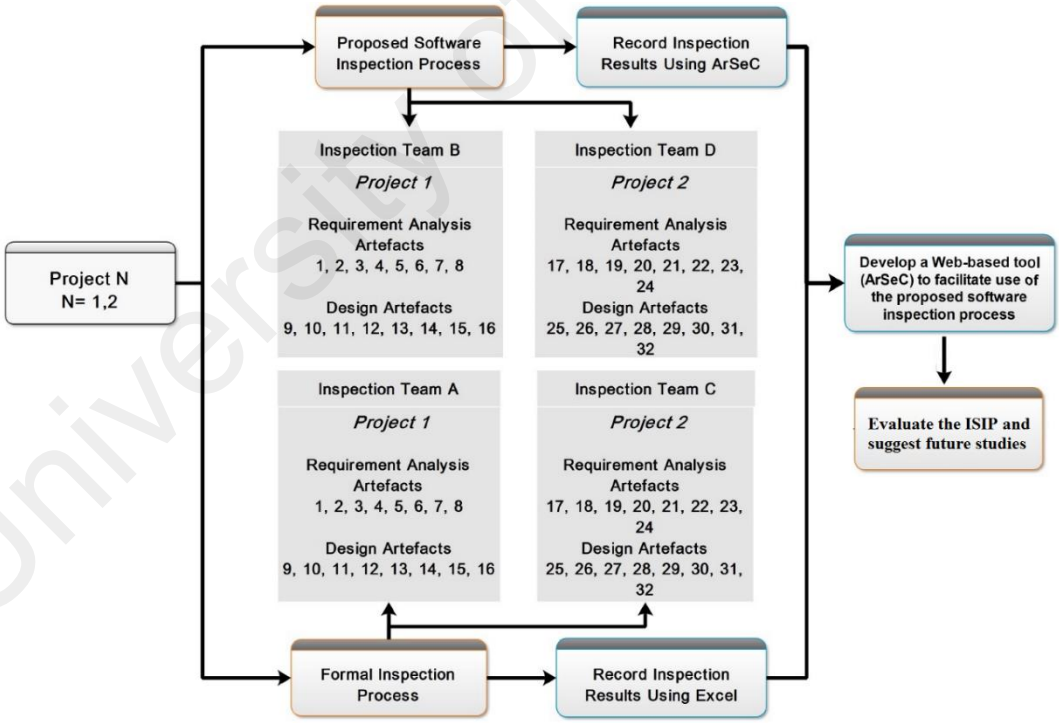


Figure 3.2: Research Design

The inspection data and outcomes of both projects were compiled and analysed using the Statistical Package for Social Sciences (SPSS) version 22 to determine whether ISIP

(Process 2) is more efficient than the formal inspection process model (Process 1). In the evaluation, these metrics were used for measuring – i) number of real (actual) defects detected in each artefact, ii) inspection time of each artefact, and iii) productivity of the inspection team, which are explained in section 3.3 below.

The following three hypotheses were formulated for this research:

3.1.6.1 Hypothesis 1:

H0: The total number of real (actual) defects detected using Process 1 is the same as the total number of real (actual) defects detected using Process 2.

H1: The total number of real (actual) defects detected using Process 2 is more than the total number of real (actual) defects detected using Process 1.

3.1.6.2 Hypothesis 2:

H0: The total inspection time on each artefact using Process 1 is the same as the total inspection time on each artefact using Process 2.

H1: The total inspection time on each artefact using Process 2 is less than the total inspection time on each artefact using Process 1.

3.1.6.3 Hypothesis 3:

H0: The productivity of the inspection team using Process 1 is the same as the productivity of the inspection team using Process 2.

H1: The productivity of the inspection team using Process 2 is higher than the productivity of the inspection team using Process 1.

3.2 Evaluation of ISIP

Three metrics are used to evaluate the efficiency of ISIP: (i) Number of defects detected – used to determine whether ISIP helps to detect more defects; (ii) Inspection time – used to determine whether ISIP reduces the inspection time; and (iii) Productivity of the inspectors – used to determine whether ISIP helps the inspectors to find more defects in a shorter inspection time. The formulas used in these measurements are explained below.

3.2.1 Number of real defects detected

As the inspections were carried out by three different inspectors, there is likelihood that inspector 2 and inspector 3 might find some real (actual) defects which have already been detected by inspector 1. Similarly, there is likelihood that inspector 2 and inspector 3 might detect some common and real defects which have not been detected by inspector 1. Thus, to eliminate double counting of the common real defects detected by the other two inspectors, the number of real (actual) defects detected in each inspection session is calculated using the following formula:

$$Total\ Number\ of\ Defects = \sum_{i=1}^{i=n} DD\ i - RD \qquad (1)$$

where,

DD_i : Total number of defects detected (DD) by all inspectors, i = 1, 2, 3

RD : Total number of common real (actual) defects detected by the second and/or third inspectors.

The size of each artefact varies depending on the type of artefact. The measurement of the size of an artefact can be the number of pages of a document, or the number of boxes in a diagram, or the number of fields in a form, etc. In the object-oriented development approach, it can be the number of use cases, or the number of classes. In this research, as the number of artefacts to be inspected are in different formats – text, diagrams, formulas,

etc. – there must be a way of determining the size of each artefact consistently and correctly. The function point was therefore used for calculating each artefact to determine its complexity. The value obtained for each artefact was then converted to the equivalent number of pages and used in the measurement of the size of each artefact. Appendix A shows details of the calculation. The defect density of each artefact is calculated using the following formula:

Defect Density = Total number of real defects in each artefact / Size of the artefact (2)

It is important to note that there is also a possibility that the defects detected and reported by the inspectors might contain defects which are not actually defects. This kind of defects is known as false positives. Hence, the number of real (actual) defects detected by an inspection team should exclude false positives. This issue is discussed in detail in section 5.3.3, and calculation of the total number of real (actual) defects detected by an inspection team is further elaborated in the section.

3.2.2 Inspection Time

The inspection time, in minutes, of an artefact is the sum of inspection preparation time, the time spent to inspect an artefact (inspection session), and the time spent to resolve inconsistencies in the artefact (explained in chapter 4), as shown below:

Inspection Time = Preparation time + Inspection session time + Inconsistencies resolution time (3)

Hence, the total inspection time taken by all the inspectors to inspect an artefact is: $\sum_{i=1}^{i=n} \mathrm{IT}$

where,

IT: Inspection time taken by an inspector to inspect an artefact.

3.2.3 Productivity of Inspection Team

As aforementioned, the total number of defects detected by an inspector may contain false positives, hence, the productivity of an inspection team – the defect finding efficiency – must be calculated based on the total number of real (actual) defects detected by the inspection team, as shown in the formula below:

Productivity of an Inspection Team = Total number of real (actua)l defects detected by the inspection team / Total inspection time on the artefact by the inspection team (4)

where,

Total number of real (actual) defects detected = Total number of defects detected - Total number of false positives

A higher defect finding efficiency implies that an inspection team is able to find more defects and in a shorter inspection time.

3.3 Validity of Research

A major concern in any research is the soundness of the processes and procedures used to carry out the research. In this research, the following four types of validity are considered:

3.3.1 Statistical validity

Statistical validity used to assess that the dependent variables are reliable as the number of defects detected (after eliminating double counting of common defects), and that the inspection time is correctly and systematically recorded either manually or using ArSeC, respectively. The correct formula is used to calculate the productivity of the inspector – false positive defects are eliminated in the counting of defects detected by an

inspector. The statistical tests used to prove the hypotheses were selected based on the advice of a statistician. The assumptions that underlie the statistical tests –test on the data distribution normality – were tested to decide whether the parametric or non-parametric tests should be used to prove the hypotheses. The only statistical error that cannot be avoided in this research is the rejecting of the null hypothesis incorrectly, in the statistical decision, α, which is set at 0.05.

3.3.2 Construct validity

Construct validity concerns the extent to which a test is measuring what it claims, or purports, to be measuring (Cronbach & Meehl, 1955). This research aims to determine whether ISIP outperforms the formal inspection process. Hence, assigning the same set of artefacts to two independent inspection teams with similar inspection experience – one inspection team using Process 1 (Formal inspection) and another team using Process 2 (ISIP) – and comparing the inspection results is a logical and suitable approach to achieve the objective. The three metrics – number of defects detected, inspection time, and productivity of inspectors – used to measure the efficiency of each process, are closely related to the performance of each process. Hence, these measurements are the most appropriate for comparing and evaluating the two processs. Moreover, all the inspectors involved do not know that they were selected to participate in the study (i.e. a blind study). They also do not know the actual total number of defects in each artefact, and that their productivity is being measured. The inspection processes – using the two processes – were conducted simultaneously and this will strengthen the construct validity of this research.

A pilot case study was conducted before the actual case study began. The pilot case study involved six inspectors from Company 1 (C1), who inspected a medium-sized artefact which had been injected with 12 additional defects. One inspection team

comprising three inspectors used the formal inspection process and MS Excel, while the other inspection team used ISIP and ArSeC, to conduct inspection. The pilot case study is aimed at assessing the feasibility of using ISIP in a full-scale case study in the company. The outcomes of the pilot study revealed that the inspectors were skeptical about the use of ISIP as it is new to them although they agreed that ISIP and ArSeC can facilitate the inspection process. Hence, necessary adjustments were made to the case study – only eight artefacts of high complexity were selected at random from a large-sized software project, and six other inspectors who were not involved in the pilot study, were selected to participate in the actual case study.

3.3.3 External validity

External validity refers to the extent to which researchers are allowed to generalise the results of a study to other participants, conditions, times, and places (Graziano & Raulin, 2014). In this research, the artefacts were selected randomly from two big software projects which had more than 50 artefacts identified in the requirements analysis and design phases, respectively. A randomised block design (i.e. complete balanced block design), with eight (equal numbers) artefacts assigned to each process (treatment), and three inspectors per inspection team were used in this research. Also, each inspection session is limited to two hours only, regardless of the process used.

3.3.4 Internal validity

Internal validity concerns the extent to which a causal conclusion based on a study is warranted. It concerns the independent variable, and not some extraneous variables that cause changes to the dependent variable (Graziano & Raulin, 2014). Although the two software projects were not developed under similar environments (i.e. using different programming languages, databases, development platforms, etc.), in this research, the artefacts selected for the case studies are of similar size. Before selecting the artefacts,

the size of each artefact is calculated using the function point approach. Hence, only artefacts of large size were used in the inspection process. This eliminates the confounding effect that would have resulted if different sizes of an artefact are used. Besides, the use of two independent inspection teams had eliminated the confounding effect – defects will be detected in a shorter time if only one inspection team setting is used in the case study to conduct inspections using Process 1 followed by Process 2. In addition, the four teams of inspectors were selected at random from both companies, based on the same number of years of work experience in software inspection. This helps to eliminate the confounding effect that would have resulted if the inspectors had different levels of inspection experience, where an experienced inspector could possibly find more defects and in shorter time than the inexperienced inspectors. This will obviously have impact on the performance and productivity of an inspector.

3.3.5 Summary

This chapter explains the research methodology used to carry out the research. It includes the literature review on the inspection processes, and problems related to these existing inspection processes. An enhanced inspection process (ISIP) was proposed to solve those problems. Two case studies were conducted to evaluate the performance of ISIP against the formal inspection process. To facilitate the use of ISIP, a Web-based inspection tool (ArSeC) was developed. In evaluating the performance of ISIP against the formal inspection process model, three metrics were used – number of defects detected, inspection time, and productivity of the inspecting teams. The statistical approach is used to prove the hypotheses established. A detailed discussion on the validity of the research was also presented.

CHAPTER 4: THE PROPOSED INSPECTION PROCESS

In view of the problems, weaknesses, and limitations identified during the inspection of the artefacts in the requirements analysis and design phases as well as the weaknesses identified during the formal inspection and other traditional inspection processes, an Improved Software Inspection Process (ISIP) was developed. In ISIP, an inspection team consists of an Author, Moderator, and Inspector, as shown in Figure 4.1. The following sections describe the responsibilities of the team and the features of ISIP, in detail.

4.1 Inspection Team

An inspection team consists of an author, a moderator, and a few inspectors, whose responsibilities are described below:

i. Author

The author is a professional person whose responsibilities include:

- Develops the artefacts, provides all information, and clarifies the ambiguities about the artefacts during the inspection process;
- Fixes the defects that have been identified during the inspection process.

ii. Moderator

This role is same as the moderator of the formal inspection process but in ISIP, the moderator has additional responsibilities of that of a reader and a recorder, and the responsibilities include:

- Records and updates profiles of inspectors into the Inspectors database – details such as telephone number and email address; inspection work experience; and inspection productivity;
- Defines the project to be inspected (Project title, project ID, inspection context, access level to manage the artefacts), and the related project constraints such as the deadline to complete an inspection process, and/or special qualifications or

certification required of an inspector to be selected to conduct inspection on a particular type of artefacts;

- Selects the inspectors and invites them to form an inspection team for a project;
- Defines the software inspection session;
- Assigns inspectors to inspect artefacts based on their specific expertise and experience;
- Provides guidelines for the inspection preparation stage;
- Conducts the inspection session - perform checklist based reading, resolve the conflicts among the inspection team members;
- Records the inspection outcomes and incidents; and
- Monitors the resolution of inconsistent defects - compile the votes from the inspectors.

iii. Inspector

In ISIP, an inspector performs the following tasks facilitated by the integrated Web-based tool, ArSeC:

- Provides his/her profiles (contacts, qualifications and inspection work experience) to the moderator to store into the Inspectors database (for the selection and assignment of inspection tasks by a moderator);
- Reads the inspection package provided by a moderator;
- Identifies the potential defects in the artefacts based on the checklists prepared by the moderator;
- Records details of the defects detected; and
- Verifies the inconsistency removal process.

ISIP is supported by an automated tool (ArSeC) and thus a system administrator is responsible for defining the users, and maintaining the databases. However, the administrator is not a member of inspection team.

In ISIP, it is recommended that either three or five inspectors be assigned to an inspection team, as explained in Step 3 of section 4.7.2.3

4.2 An Improved Software Inspection Process (ISIP)

The traditional formal software inspection process involves having face-to-face meetings among system developers and inspectors when inspecting the software artefacts (Fagan, 2002). Advancements in distributed systems and Web-based applications have greatly facilitated access to distributed databases for retrieving related inspection artefacts and documents to conduct online inspections remotely, to remove the defects detected, and save the updated information. The Distributor is the core component of SQL Server's replication process. ISIP incorporates these new technologies, and is therefore an enhanced formal inspection process, which can greatly improve the quality of software inspection of the artefacts in the requirements analysis and design phases. The workflow and features of ISIP are illustrated in Figure 4.1 and explained in the following sections.

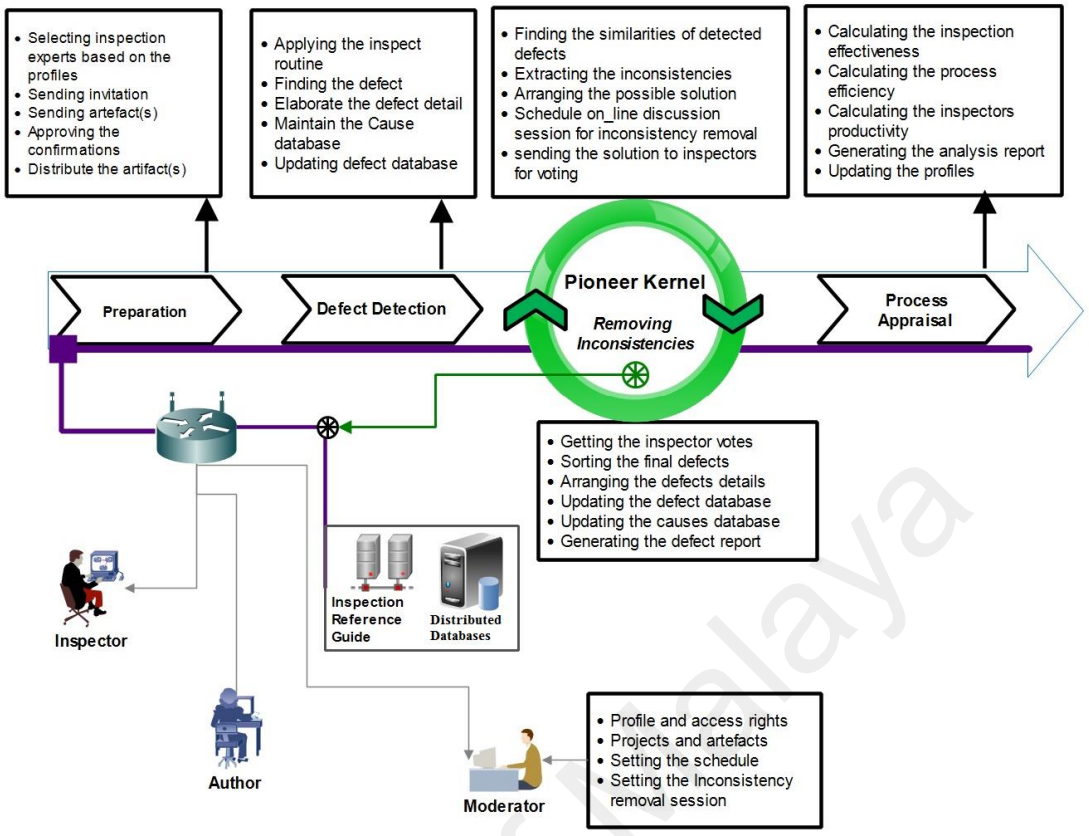


Figure 4.1: The work flow of ISIP

4.3 Inspection Process in ISIP

The inspection process in ISIP consists of four stages – preparation, defects detection, pioneer kernel, and process appraisal. The activities involved in each stage are applicable for the inspection of artefacts both in the requirements analysis and design phases.

4.3.1 Preparation

The number of inspectors to be involved is first determined. They are selected based on their expertise, inspection experience, and average past inspection productivity. Their profiles are updated and stored in the Inspectors database. A moderator will send an invitation message together with the artefacts and the related inspection documents – inspection checklists, inspection guidelines, etc. – to each selected inspector, and then wait for their confirmation of participation. There are two other databases in ISIP: (i) Defects database – stores all types of potential defects detected in the requirements

analysis and design phases; it is accessible by all members of the inspection team; software defects data are gathered from technical reports and research findings published in reputable scholarly journals; (ii) Causes database –stores the potential causes and effects of each potential defect identified in the requirements analysis and design phases; this information serves as a reference to help the author in the removing the defects.

The inspection sessions are conducted online using ArSeC, and the duration of the inspection process is also determined at this stage. The artefacts, together with the necessary guidelines and information, will be distributed to the inspectors involved in the inspection process.

4.3.2 Defect Detection

The second stage focuses on finding the defects in the artefacts. Each inspector carries out the task using the inspection routine/guidelines and checklists given, as well as referring to the Defects database to find potential defects. The details of the defects found are recorded, including the potential causes of each defect. Information on any new defects found will be added into the Defects database under the specific defect category, while the potential causes of the defects are recorded in the Causes database. This regular updating process helps in maintaining both a comprehensive Defects database as well as a Causes database, which can greatly aid the inspectors in finding defects, and the author in removing defects detected, quickly. Hence, this stage involves three crucial activities – defect detection, defect identification (sources of defect), and new defects categorization and recording.

4.3.3 Pioneer Inspection Kernel

At this stage, similar defects as well as inconsistent defects detected by each inspector are listed. All inspectors will help in making decisions to resolve all the inconsistent defects. Any inspected artefacts which require rework will be returned to the author. The

moderator is responsible for managing the rework and confirming the successful removal of defect by the inspectors.

For the inconsistent defects, the moderator will clarify the ambiguous information related to them, schedule an inconsistent defects removal session, and obtain a consensus among inspectors on appropriate action to be taken to resolve all the inconsistent defects. Similarly, information on any new defects found at this stage will be added into the Defects database under the relevant defect category, and the potential causes and effects of the defects will be recorded into the Causes database. These two databases are regularly updated, hence, they provide current information for preparing comprehensive inspection checklists to aid in the defect detection and removal processes during artefact inspections. A final inspection report is prepared by the moderator and distributed to all the members of the inspection team.

4.3.4 Process appraisal

During the inspection process, data are systematically collected and updated into the respective databases. These include information on the types and characteristics of the artefacts, total inspection time, inspectors' profiles, the number and types of defects detected during the inspection process. These data are used to measure the productivity, and efficiency of each inspector, and the overall effectiveness of the inspection process. Hence, every inspector as well as the inspection process are appraised, and an analysis report of the inspection process is prepared by the moderator. The profile of each inspector involved are updated after each inspection, thus, a moderator has the latest information for selecting the most suitable inspector to conduct future software inspections.

4.4 Unique Features of ISIP

ISIP has four unique features which are not available in the traditional formal inspection process: (i) selection of suitable inspectors; (ii) maintenance of defects list; (iii) preparation of inspection checklists; and (iv) avoidance of defect transition. The following sections describe these features in more details.

4.4.1 Selection of Suitable Inspectors

A moderator uses the Inspectors database to select suitable inspectors based on different criteria such as their expertise, past inspection experience (i.e. number of years of inspection, and the types of artefacts inspected), and average past inspection productivity (i.e. number of real (actual) defects detected per hour). During an inspection process, it is inevitable to have disputes concerning the defects found (i.e. inconsistencies) among the inspectors. For example, an inspector might deem an issue as a defect, whereas two other inspectors might disagree (i.e. it is not a defect), or vice versa. The ISIP guidelines for the selection of suitable inspectors can help in resolving such dispute, through the use of a weighted vote of each inspector. Inspectors have three skill levels – highly skilled, skilled, and semi-skilled – based on the number of years of inspection experience (criteria 1), the number of artefacts they have inspected (criteria 2), and the average past inspection productivity (criteria 3). Table 4.1 shows the classification of the skill levels of the inspectors (O'Regan, 2002).

Table 4.1: The skill levels of the inspectors

Skilled Level	No. of Years of Inspection Experience	No. of Artefacts Inspected	Average Past Inspection Productivity
Highly Skilled (HS)	≥20	≥ 900	> 40
Skilled (S)	10-19	500-899	30 - 40
Semi-Skilled (SS)	< 10	< 500	< 30

Table 4.2 shows the combined weighted votes of an inspection team that comprises three inspectors. In resolving disputes (inconsistencies) related to the defects, the vote of a highly skilled, skilled, and semi-skilled inspector is assigned 3 points, 2 points, and 1 point, respectively (O'Regan, 2002). There are 10 sets of possible combinations in the decisions made by the three inspectors. For example, in No. 2 of Table 4.2, two highly skilled and one skilled inspectors are involved, and the total weighted votes for the same decision made by the first and second inspectors, first and third inspectors, and second and third inspectors, are 6 points, 5 points, and 5 points, respectively. The values are all greater than the vote of the inspector who disagreed with their decision (i.e. 2 points, 3 points, and 3 points, respectively). The decision based on the greater vote weightage will be used to resolve the dispute. It is important to note that the decision of the values of the weighted votes can be positive as well as negative. An inspection team can consist of inspectors having different skill levels, can except for No. 4 and No. 10. These two combinations in the inspection team: one highly skilled, skilled, and semi-skilled; and one skilled together with two semi-skilled inspectors, will not be used to form an inspection team, as such combinations might result in a tie decision (as shown in red). Hence, ArSeC would not recommend forming such teams. A similar approach is used to form a five-inspector team.

An interesting situation that could arise using this selection approach is the structure of an inspection team that comprises three semi-skilled inspectors (No. 9 of Table 4.2). There will be doubts regarding the competency and expertise of this inspection team. In this regard, it is recommended that such a team should be assigned to inspect only artefacts of low priority and low complexity. However, there is also a possibility that this team might perform as well as the inspection team that has three highly skilled inspectors. This can be determined more accurately by conducting further studies on inspection team structure. Appendix B shows the combination of weighted votes of five inspectors

Table 4.2: Combination of weighted votes of three inspectors

No.	Inspector Skill Level			Decision based on Combination of Same Vote		
	1st	2nd	3rd	1st and 2nd	1st and 3rd	2nd and 3rd
1.	HS (3)	HS (3)	HS (3)	6>3	6>3	6>3
2.	HS (3)	HS (3)	S (2)	6>2	5>3	5>3
3.	HS (3)	HS (3)	SS (1)	6>1	4>3	4>3
4.	HS (3)	SS (1)	S (2)	4>2	5>1	3=3
5.	HS (3)	SS (1)	SS (1)	4>1	4>1	2<3
6.	HS (3)	S (2)	S (2)	5>2	5>2	4>3
7.	S (2)	S (2)	S (2)	4>2	4>2	4>2
8.	S (2)	S (2)	SS (1)	4>1	3>2	3>2
9.	SS (1)	SS (1)	SS (1)	2>1	2>1	2>1
10.	SS (1)	SS (1)	S (2)	2=2	3>1	3>1

Keys: HS – Highly Skilled S – Skilled SS – Semi-Skilled

4.4.2 Maintenance of Defects List

Categorising and maintaining different types of defects found in the requirements analysis and design phases of software development makes it easier for a moderator to prepare comprehensive checklists for the inspection process. In addition, newly detected defects are regularly being added into the Defects database. This helps in maintaining an updated list of defects. Besides, the Defects and the Causes databases are updated at every inspection process. Each defect is classified using a new method proposed in this thesis. Hence, this further facilitates defects detection and classification, and thereby giving better insight into the main causes (defect trigger) and sources of defects.

4.4.3 Preparation of Inspection Checklists

According to Ackerman (1989), a comprehensive checklist helps to achieve more effective software inspection regardless of the software inspection process used. In ISIP, the Defects and Causes databases are duly updated upon completion of the inspection process. New defects and their causes and elimination details are clearly described and updated into the relevant databases. The rate of occurrence of each defect is calculated and updated into the Defects database for future reference. Hence, more attention must be paid to those more frequently identified and serious defects to provide updated and

comprehensive inspection checklists for future artefact inspections. Also, a moderator can use the Checklist database to prepare inspection checklists based on the latest checklists of similar software projects. This will save preparation time and increase the comprehensiveness of inspection checklists. A sample of the checklist of P2 of artefact 6 (P2-A6-R) of the requirements analysis phase and the checklist of P2 of artefact 2 (P2-A2-D) of the design phase is shown in Appendix C.

4.4.4 Avoidance of Defect Transition

As mentioned above, suitable inspectors are selected to conduct inspection of the artefacts in the requirements analysis and design phases. The use of comprehensive checklists helps in detecting the defects in these phases faster, and thus ensures early removal of the defects. The early detection and early removal of defects will prevent propagation of defects to the later software development phases.

4.4.5 Inspection Reference Guide

In ISIP, an inspector kit is prepared by a moderator and sent to the relevant inspectors during the preparation for the inspection (Chapter 4). The inspection kit is a reference guide that provides clear insight into artefact inspection, the aim of the inspection process, as well as the use of ArSeC to support the inspection process. This inspector kit is particularly useful for new inspectors who are not familiar with the workflow of ISIP, and the use of ArSeC.

4.4.6 Shared Databases

ArSeC is a web-based inspection process support tool, which allows inspectors to conduct online inspections remotely from anywhere and at any time. Inspectors who are not available during the scheduled online inspection session(s), can access the related databases(such as the Defects and Causes databases through the shared access rights granted to them) to retrieve the information needed to conduct inspection at a time

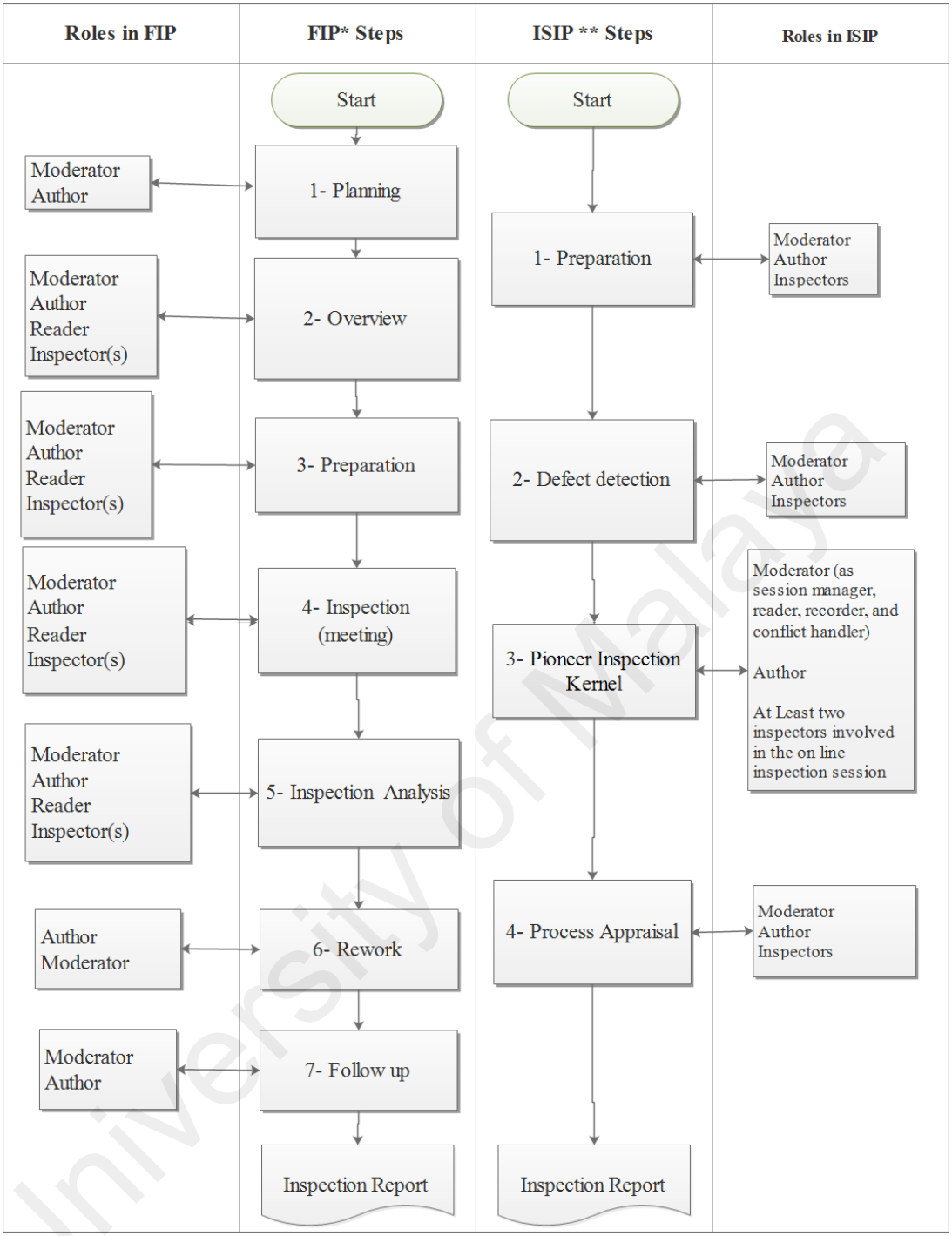
convenient to them, and submit their inspection results to the moderator using ArSeC. Furthermore, any queries or ambiguities about the artefacts can be clarified with the moderator or author using ArSeC, at any time and from anywhere.

4.5 Comparison between ISIP and the Formal Inspection Process: The Enhancements Made

ISIP was developed to improve the quality of software process by making enhancements to the formal inspection process (FIP). The major enhancements made include the roles and responsibilities of the inspection team, the selection of inspectors, and the changes to the inspection process, which are illustrated in Figure 4.2 and described below.

i) The roles and responsibilities of inspection team members, and team size

In FIP, there are four to five roles for the inspection team (team size is 4-5) with one member to assigned to each role – author, moderator, reader, recorder (a moderator may also play this role), and inspector (Fagan, 1999), as illustrated in Figure 4.2.



**** Formal inspection process ** Improved software inspection process**

Figure 4.2: Comparison between ISIP and Formal Inspection Process

However, in ISIP, the inspection team has only three roles – author, moderator and inspector, with at least one member assigned to each role except for the inspector's role, which can consist of three or five inspectors (team size is 5 or 7 members). The moderator not only prepares, arranges, and manages the inspection process, but also assumes the roles and responsibilities of a reader and a recorder during the inspection process. In FIP,

the final decisions to resolve any ambiguous issues or conflicts during an inspection process are based on the decisions of all the inspection team members. However, in ISIP, ambiguous issues and conflicts are resolved through a weighted voting process which involves the inspectors only.

ii) The selection of inspectors

In FIP, the selection of inspector(s) is based on the inspector's experience and knowledge (Laitenberger, 2002). However, in ISIP, the selection of inspectors is based on the skill levels - highly skilled, skilled or semi-skilled - and areas of expertise of each inspector. It is also recommended that the team should compose of inspectors of at least two different skill levels such as two highly skilled and one skilled inspector or one semi-skilled inspector. This will provide an opportunity for the skilled and semi-skilled inspectors to learn from the highly skilled inspector during the inspection process.

iii) Inspection process and resolution of inconsistencies

In FIP, the inspection process consists of seven activities – planning, overview, preparation, inspection (meeting), inspection analysis, rework, and follow-up. In ISIP, the inspection process consists of four main stages – preparation, defect detection, pioneer inspection kernel, and process appraisal. In FIP, face-to-face group meeting is used to detect the defects and to resolve the inconsistencies (Fagan, 1976). A reader will perform checklist-reading and a recorder will document the inspection process and outcomes during the inspection process. In ISIP, a moderator will perform the checklist-reading and also record the inspection details and outcomes. Besides, using ArSeC (a tool developed to support the inspection process), inspectors who are not available during the scheduled inspection session, can conduct inspection on the artefacts individually, and submit the inspection outcomes to the moderator for consolidation within the stipulated inspection timeline.

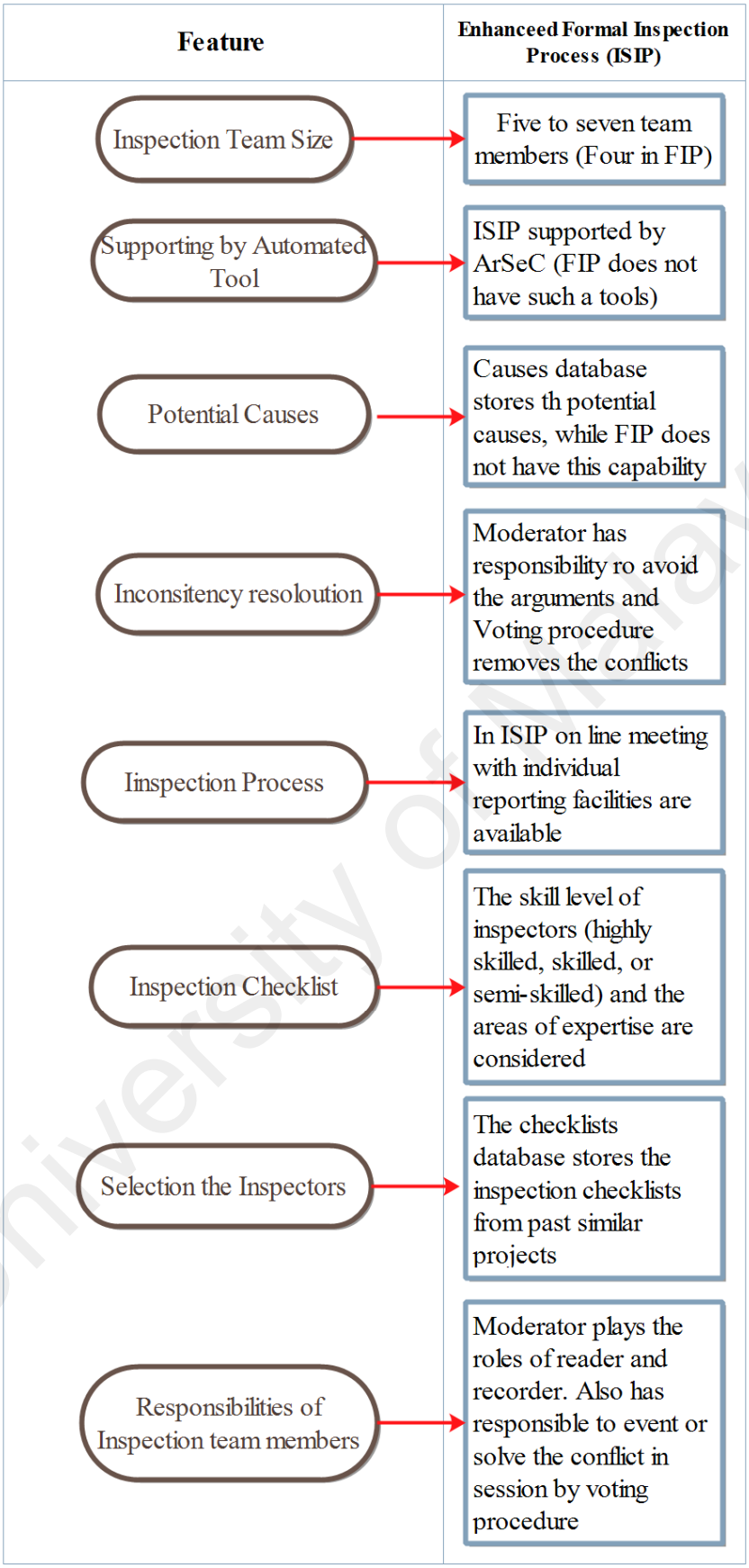


Figure 4.3: The Enhancements of Inspection Process in ISIP Comparing Formal Inspection

Also, any ambiguity pertaining to the artefacts can be clarified with the author, and inconsistencies can be resolved easily using the proposed weighted voting process, without the need to have all the inspection team members to be present for the online scheduled inspection session (but at least two inspectors must be available during the online scheduled inspection session). This feature is not available in FIP.

iv) Preparation of checklists

In FIP, a checklist is used during the inspection process. Based on the literature review, the checklists are prepared based on checklists of similar past projects (Laitenberger, 2002). However, there is no mention of the preparation of the checklists, and also no mention whether a database is available for maintaining the questions of past inspection checklists. In ISIP, a Checklists database is created to store and maintain all the checklists that were created for the past software inspection projects. This database serves as an archive and a reference source for the moderator to prepare a fairly "comprehensive and complete" checklist easily and quickly.

v) Iteration of inspection process

In FIP, the inspection team will decide whether re-inspection is needed if too many defects were detected in an artefact (Fagan, 1999). In ISIP, at least one re-inspection is conducted on an artefact. The criterion to determine if a second re-inspection process is needed is: when the total number of real defects detected during the re-inspection process exceeds 5% of the total number of defects detected during the first inspection process. The inspectors can decide on what is the threshold (percentage of the defects detected) for re-inspection and the threshold can be changed accordingly depending on the importance of the software project or the artefact.

vi) Potential causes of each defect

Providing solutions for the defects is not the focus of an inspection process. However, ISIP can reduce the rework process of the author because a Causes database is created to store and maintain all the potential causes of each defect that had been detected in an artefact. The author is responsible to maintain this Causes database whenever new defects are detected and resolved. Hence, the inspectors can retrieve the potential causes of each real defect (if the potential causes are available in the Causes database), and distribute to the author for rework together with the consolidated inspection report. This enhancement made to the FIP, is a unique feature of ISIP.

vii) Use of automated inspection tool (ArSeC) and inspection meeting

The use of ArSeC supports the inspection process by facilitating the distribution of the inspection documents to the inspection team members; recording the data and using the data in calculation inspection data; facilitating the storage and retrieval of data by the inspection team members such as the defect data, checklists, causes of each defect, etc. As it is a Web-based inspection support tool, the inspection meeting can be held at any time and at anywhere to accommodate the inspection team members who are at different geographically locations. There has been no report in the literature the use of an automated tool to support the formal inspection process (Chen, & Agrawal, 2014).

4.6 Classification of Defects

Table 4.3 and Table 4.4 show the proposed classification used in this research, for defects detected in the requirements analysis and design phases. The classification is based on the Orthogonal Defect Classification (ODC) technique and the defect classification formats used by IEEE, and NASA (NASA, 2013; IEEE Std 1044-2009). In classifying the defects found in the requirements analysis phase, it considered include the functionality, performance, environment, interface, security, and miscellaneous features. A defect code is assigned to each defect using the naming convention: AD_i, where, AD

– Analysis Defect, i – Defect class i. Each defect can be any one of the following modes: Missing (M); Incorrect (IC); Inconsistent (ICST); Ambiguous (A); Wrong Section (WS), and Extra (E).

Table 4.3: Classification of defects (Requirements analysis phase)

Defect Class	No.	Defect Code	Mode	Description
Functionality	1.	AD_F1	M	The relevant information to explain the system behaviour from the interoperational aspect is missing. Incomplete functional specification.
	2.	AD_F2	IC	Function incorrectly specified.
	3.	AD_F3	ICST	Requirement statements directly contradict each other.
	4.	AD_F4	A	Requirement statements are not clear or are confusing.
	5.	AD_F5	WS	Requirement statements are misplaced.
	6.	AD_F6	E	The function is not needed.
Performance	7.	AD_F1	M	Definition of types of defects, factors related to performance are not given.
	8.	AD_P2	IC	Defined criteria and factors are not compatible with the system specification.
	9.	AD_P3	ICST	Various aspects of performance are in contradiction.
	10.	AD_P4	A	Criteria are not stipulated clearly and quantitatively.
	11.	AD_P5	WS	Precision and attributes related to performance are juxtaposed or are wrongly defined.
	12.	AD_P6	E	Extra fields that have been included for performance criteria are not stipulated in the system specifications.
Environment	13.	AD_E1	M	Some required resources have not been provided or have been simply ignored.

	14.	AD_E2	IC	Resources have not been clearly described.
	15.	AD_E3	ICST	Amount, and type of resources provided are not compatible with actual usage.
	16.	AD_E4	A	Required resources have not been stated separately in an organised manner.
	17.	AD_E5	WS	Some resources have not been allocated to the right place or the resource needed cannot be provided.
	18.	AD_E6	E	Some resources which have been included in the system specification, are in fact, not needed.
Interface	19.	AD_I1	M	The relationship between parts, systems, and actors is not explained.
	20.	AD_I2	IC	Wrong interaction description.
	21.	AD_I3	ICST	Relationship between different sections or external entities is contradictory.
	22.	AD_I4	A	The way the relationship starts, continues, and ends is unclear.
	23.	AD_I5	WS	Relationship or a dialogue needs to be moved to another place.
	24.	AD_I6	E	A dialogue that has been defined, is in fact, not necessary.
Security	25.	AD_S1	M	Some of the security-related issues are left out from the documentation.
	26.	AD_S2	IC	Some of the codes or security-related issues do not fulfill the security objectives.
	27.	AD_S36	ICST	Security procedures and criteria contradict each other.
	28.	AD_S4	A	Good security procedures cannot be designed due to lack of clarity.
	29.	AD_S5	WS	Invalid information transmitted or received cannot be classified like other issues.
	30.	AD_S6	E	Instruction or information which will not contribute to any security improvements.
Miscellaneous	31.	AD_M1	M	Items are missing and cannot be placed in the higher classes.

	32.	AD_M2	IC	Invalid facts, diagrams or sentences which cannot be placed in the classes mentioned.
	33.	AD_M3	ICST	Other inconsistencies apart from the items mentioned.
	34.	AD_M4	A	Items that are not clear require further clarification, and cannot be placed in the higher classes.
	35.	AD_M5	WS	The items require new placements and do not belong to the classes they are assigned.
	36.	AD_M6	E	Extra items whose omission does not harm the definition or fulfillment of the requirements, and cannot be classified with the mentioned issues.

AD_i – Requirements Analysis Defect class number i

In the design phase, the defects are classified into algorithm, alignment, checking, documentation, function, interface, package/COTS, and miscellaneous. The defect code is assigned using similar naming convention as in the requirements analysis phase – DD_i, where DD – Design Defect, and i defect class. The mode of each defect is defined using the same approach as the mode of defects used in the requirements analysis phase.

Table 4.4: Classification of defects (Design phase)

Defect Class	No.	Defect Code	Mode	Description
Algorithm	1.	DD_A1	M	An algorithm or a method is totally omitted.
	2.	DD_A2	IC	An algorithm does not fulfill the defined requirement or is not compatible with the conceptual design.
	3.	DD_A3	ICST	Algorithms have incompatibilities in areas such as calling, sorting, arguments, objectives, and resources.
	4.	DD_A4	A	Execution sequence, operation, arguments or type of data exchange is not clear.
	5.	DD_A5	WS	Calling or operation is not placed in the right place.
	6.	DD_A6	E	An algorithm which is never called or its product is not needed.

Assignment	7.	DD_As1	M	An assignment or initialisation has totally been ignored.
	8.	DD_As2	IC	Method or assigned value is invalid.
	9.	DD_As3	ICST	Assignments are not compatible, and change each other's value before any usage.
	10.	DD_As4	A	Type, time, order or method of setting values is unclear.
	11.	DD_As5	WS	Assignment must be relocated.
	12.	DD_As6	E	Initialisation where the value would not be used, and it would undergo change in another operation.
Checking	13.	DD_C1	M	Lack of some controls, conditions or their parameters.
	14.	DD_C12	IC	Some conditions are not valid or their parameters have been set incorrectly or have an incorrect range.
	15.	DD_C3	ICST	Conditions contradict each other or are logically inconsistent. Parameters of each condition might be in contradiction with the order of execution in the body.
	16.	DD_C4	A	Initiation, execution or end of a condition or control is not understandable.
	17.	DD_C5	WS	The condition is in the wrong location, has an incorrect range and must be moved.
	18.	DD_C6	E	Branches that would never be entered logically.
Documentation	19.	DD_D1	M	Any description, diagram or relevant object which is missing.
	20.	DD_D2	IC	Documentations which are not prepared properly from a syntactic or lexical point of view.
	21.	DD_D3	ICST	The contents or reference of some of the documentations are contradictory.
	22.	DD_D4	A	Sentences or references do not have a clear meaning or origin or destination.
	23.	DD_D5	WS	Titles, text and [sentences or references] refer to the wrong items.
	24.	DD_D6	E	Presence or absence of sentences or facts which will not affect the development or execution.
Function	25.	DD_F1	M	Phrase or sentence essential for understanding system behaviour is missing.
	26.	DD_F2	IC	Information related to the internal operational behaviour of the system is not compatible with the requirements specification.
	27.	DD_F3	ICST	Some sentences negate each other.

	28.	DD_F4	A	The conditions set for system specification is not compatible with the included sentences.
	29.	DD_F5	WS	Exchange and use of parameters or calls are not used in the correct location.
	30.	DD_F6	E	An operation that would never be launched or is not mentioned in the system specification.
Interface	31.	DD_F1	M	An interface which has not been defined or an incomplete interaction.
	32.	DD_I2	IC	Interaction of data, information and controls are invalid.
	33.	DD_I3	ICST	Submission or reception tasks and start and end of interaction are incompatible.
	34.	DD_I4	A	Interaction methods, specifically, exchange of information is unclear.
	35.	DD_I5	WS	Dialogues are not in the right location.
	36.	DD_I6	E	Relationships defined will never be used.
Package/ COTS	37.	DD_P1	M	Some library functions or ready modules have not been incorporated.
	38.	DD_P2	IC	Ready-made classes, methods, objects, and functions have been defined or used incorrectly.
	39.	DD_P3	ICST	Tasks and responsibilities of components or their usage are in contradiction.
	40.	DD_P4	A	Purpose of a component is not clearly defined.
	41.	DD_P5	WS	Some responsibilities of classes or functions have been wrongly located. Parameters are not exchanged correctly.
	42.	DD_P6	E	Unused components have been defined or other components can carry out their tasks as well.
Miscellaneous	43.	AD_M1	M	Missing items in design documentations.
	44.	AD_M2	IC	Invalid descriptions, labels, titles or relationships that cannot be placed in any of the higher classes.
	45.	AD_M3	ICST	Design inconsistencies apart from the mentioned issues.
	46.	AD_M4	A	Implausible or non-standard issues/items that require more interpretation or explanation, and cannot be put in the same category as the issues/items mentioned.

	47.	AD_M5	WS	Definition, relationship, order or classification/categorisation that is assigned incorrectly, and are not related to the issues/items included.
	48.	AD_M6	E	Extra issues with which the requirements of the analysis phase can be fulfilled without them, and do not belong to the related classes.

4.7 Development of a Web-based Tool

Following the proposal of the new software inspection process (ISIP), and the compilation of a comprehensive list of defects identified in the requirements analysis and design phases, respectively, a Web-based tool (ArSeC) was also developed. The tool provides four databases to store details of each inspector (Inspectors database), inspection checklists of each project (Checklists database), defects details (Defects database), and the potential causes of each defect (Causes database). Hence, ArSeC facilitates the selection of the most suitable (qualified) inspectors, helps in the preparation of inspection checklists by a moderator, accelerates inspection of the artefacts by the inspectors, and removal of defects by the author, and assists in defect data collection and analysis.

4.7.1 Artefacts and Session Control System (ArSeC)

ArSeC is a Web-based software inspection support system developed using agile development techniques and ASP to facilitate the inspection of artefacts in the requirements analysis and design phases. It uses MS SQL to store details of the latest defects, thus helps to accelerate the defect removal process (rework).

4.7.2 The Workflow of ArSeC

The inspection process in ArSeC consists of seven steps, as shown in Figure 4.3, and are described in more detail below.

4.7.2.1 Step 1: Project definition

A moderator defines (creates) a project for artefact inspection, determines the phase of inspection – requirements analysis phase or design phase – and the starting and ending

dates of the inspection. Thus, the inspectors will be fully aware of what the inspection will involve, as well as the duration of the inspection process.

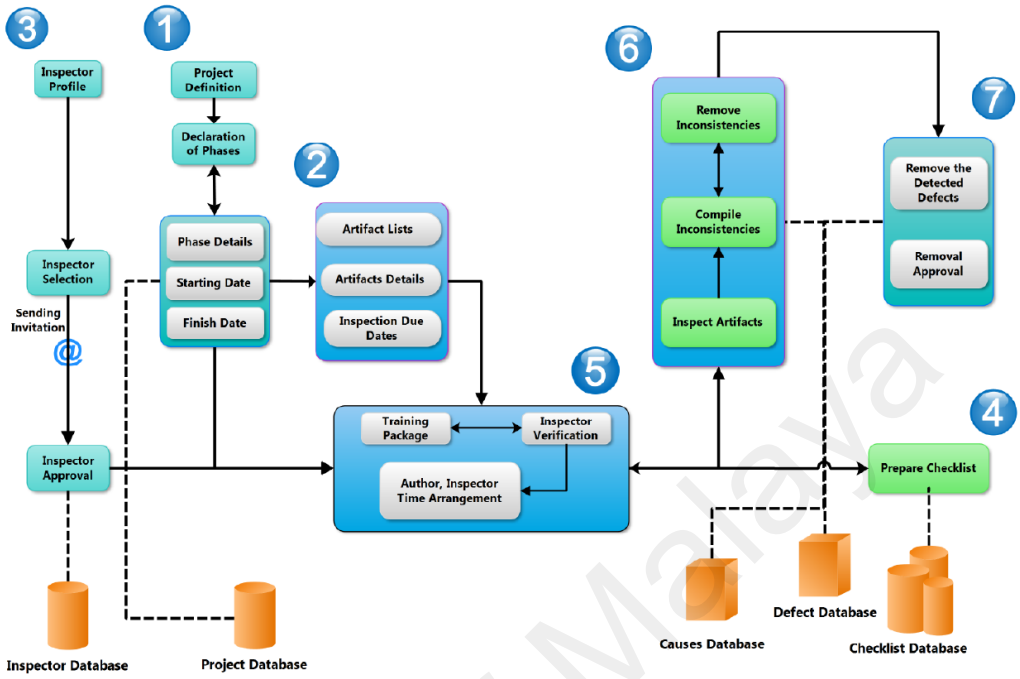


Figure 4.4: The work flow of ArSeC

4.7.2.2 Step 2: Preparation of Artefacts

In this step, a list of artefacts related to the particular inspection phase is generated. The artefacts to be inspected are categorised, prepared, and assigned inspection due dates. To meet the deadlines, artefacts which are deemed of higher priority are moved to the top of the inspection list. An Inspector Kit is also prepared to give the inspectors a clearer understanding of the artefact inspection process, the focus of the inspection process, and the use of ArSeC.

4.7.2.3 Step 3: Formation of Inspection Team

Either three or five inspectors will be selected to be members of an inspection team, depending on their profiles. The number of inspectors must be an odd number so that a decisive vote can be casted to avoid a tie in the event of disagreement during the inspection process,. The profiles of the inspectors are updated whenever they have completed inspection of the artefacts. The inspection phase and the inspection date, the

number of artefacts inspected, and the number of defects found in each artefact are recorded. The availability of the inspector to conduct an inspection is also recorded in the database. Hence, only inspectors who are available and have the necessary inspection expertise will be selected. A moderator formally invites the selected inspectors and provides them with brief description of the artefact, related documents, as well as informs them on the inspection deadline. Having a clear understanding of what the task will involve, the inspectors decide and confirm their involvement in the inspection process. In the event an inspector decides not to participate in the inspection, the moderator will assign the inspection job to another suitably qualified inspector.

4.7.2.4 Step 4: Preparation of checklist

Depending on what artefacts are to be inspected, a moderator will prepare an inspection checklist for the inspection of the specific artefacts using checklists of defects detected in similar projects conducted in the past.

4.7.2.5 Step 5: Preparation for inspection

A moderator sends the Inspector Kit together with information of the artefacts to all the inspectors. All inspectors will be given 7-14 days to conduct a preliminary review of the artefacts received, depending on the complexity of the artefacts. The inspectors will then have to indicate that they are ready and available to conduct online inspection with the author on the date determined by the moderator. Also, any inspector who is unable to conduct the inspection tasks assigned due to unforeseen circumstances, a replacement can be made at this stage. The inspection session time for each inspector will be arranged at this stage, as well.

4.7.2.6 Step 6: Conduct inspection

All inspectors will conduct online inspection using ArSeC guided by the inspection checklists prepared by the moderator. The author is also present to answer any questions

and clarify any ambiguities raised by the inspectors. An optimum online inspection session should be completed within two hours. ArSeC sends a reminder to all the inspectors ten minutes before the end of each inspection session. At the end of the inspection process, the moderator reviews and compiles all the inspection results, and prepares a list of inconsistencies, if any, using ArSeC. The list of common defects detected is sent to the author for removal (rework) of the defects as well as the information on the potential causes of each defect. To deal with the inconsistencies, the moderator distributes the list of inconsistencies to all the inspectors and calls for a vote. This voting-compilation-voting process will be iterated until all the listed inconsistencies are duly resolved. A final list of defects will be compiled and distributed to all inspection team members. The total number of defects detected is calculated and the defect details are recorded into the Defects database.

ArSeC makes it possible to conduct online inspection, therefore, inspectors who are unable to conduct inspection at the time scheduled, can seek clarification on any ambiguities with the author anytime via email and submit the inspection results to the moderator for compilation by the inspection due date. The duration of each online inspection session as well as the inspection time of the inspectors who conduct inspection on separate sessions are recorded and used to calculate the total inspection time and the productivity of each inspector.

4.7.2.7 Step 7: Defect removal

In this step, the author removes the defects and records details pertaining to the defects removal, together with the actual cause(s) of each defect. Details on the cause(s) are updated into the Causes database which will be used as reference for future inspections.

4.8 Defects and Causes of Defects

A defect could result from one or more causes, which could occur sequentially or in parallel. In the Causes database, the causes are classified into different categories such as omission, and accidental. The Causes database can be searched by using keywords. A cause may result in different defects. For example, a team of inexperienced system analysts who produced poor quality analysis specifications and reports, might include defects such as missing and ambiguous specifications. In this example, the team of inexperienced analysts is a cause, which could have resulted from inappropriate selection of system analysts. Details of the causes of each defect are recorded by the author during the defect removal process. On the other hand, details of the defects are recorded by the inspectors during the inspection process.

4.9 Functions and Features of ArSeC

ArSeC performs five main functions project definition; defect management; tracking of causes; checklist management (generates various types of checklists, and maintains the checklist database); and supports the whole inspection process (schedules inspection session; records inspection results – total number of defects detected, total inspection time, productivity of inspector, etc.). Figure 4.4 shows a snapshot of ArSeC.

4.10 Comparison between ArSeC and other inspection process support tools

Table 4.5 shows a comparison of the features and functions provided of ArSeC with nine other inspection tools (Fagan, 1986; Fagan, 1999; Fagan, 2002). Among these 10 inspection support tools, only ArSeC, Compass, and EMS provide scheduling support for managing the inspection sessions. Another important feature of ArSeC, WIP and HyperCode is that they are Web-based systems which allow inspection to be conducted online without the need to have all the parties concerned – author, moderator, and inspectors – conduct face-to-face inspections, simultaneously.

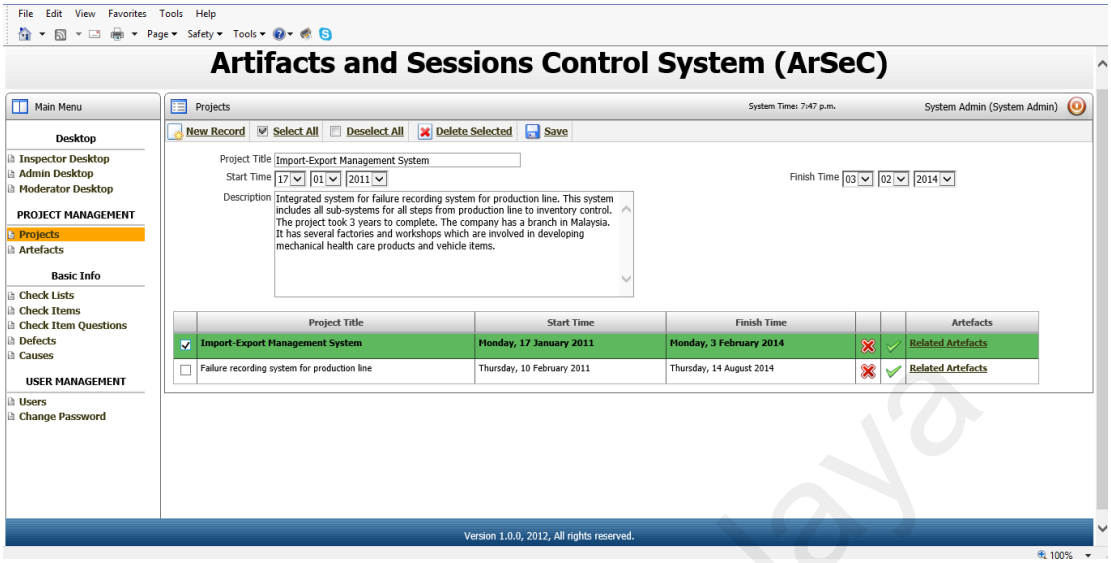


Figure 4.5: A screen shot of ArSeC

One unique feature of ArSeC, not available in the other nine tools, is the authorisation and authentication feature. Every user, regardless of his/her role – author, moderator or inspector – must be defined by an authorised person and validated through an authentication process which includes the use of passcodes (i.e. two different passwords for login and logout, respectively, for each inspection session), security questions, and IP verification. This prevents intruders from logging-in and gaining access to any important data stored in ArSeC.

Although online facilities are provided by most of the inspection tools, they are only used for discussing the inconsistencies detected. Only ArSeC allows online weighted voting, as explained in section 4.4.1, above.

Table 4.5: Comparison between ArSeC and nine current inspection support tools

Capabilities	Inspection Tools									
	ArSeC	Compass	WIP	EMS	ICICLE	InspeQ	CSI	Scrutiny	InspectA	HyperCode
Scheduling support	✓	✓	X	✓	X	X	X	X	X	X
Web-based	✓	X	✓	X	X	X	X	X	X	✓
Distributed meeting	✓	X	X	✓	X	X	✓	✓	X	X
Defect classification	✓	X	✓	✓	✓	X	✓	✓	X	X
Checklists	✓	X	✓	X	X	✓	X	X	✓	X
Data collection	✓	✓	X	X	✓	X	✓	✓	X	X
Weighted Voting	✓	X	X	X	X	X	X	X	X	X
Process support	✓	X	X	✓	X	X	X	X	X	X
Synchronous facility	✓	X	X	X	✓	X	✓	✓	X	X
Authorisation and Authentication	✓	X	X	X	X	X	X	X	X	X

4.11 Summary

This chapter discusses the enhanced formal inspection process (ISIP), the roles of the inspection team members, the four unique features of ISIP, and the development of an automated inspection support tool (ArSeC). The workflow of ArSeC, and a comparison of the features between ArSeC and nine other inspection support tools, are also discusss. This chapter also discusses two case studies conducted in two companies, and the three hypotheses formulated to determine if ISIP and ArSeC can help to improve the quality of the inspection process. Details of the case studies, data collection, and testing of the three hypotheses using statistical tests, are presented in chapter 5.

CHAPTER 5: CASE STUDIES, DATA COLLECTION AND ANALYSIS

Two case studies were conducted to evaluate the efficiency of ISIP. A software development project (P1) from one manufacturing company (C1) and another software project (P2) from a trading company (C2) were used in the two case studies. These two companies conduct software inspection using Fagan's formal inspection process. Their inspections involved projects which contain medium- or large-sized artefacts only. They do not perform inspections on small-sized artefacts – those less than three pages – which the project managers considered unnecessary as they might not uncover any serious or significant defects in these simple artefacts.

In case study 1, eight (8) artefacts of the requirements analysis phase of P1 were selected at random from a total of 30 artefacts. These artefacts were first inspected by three (3) inspectors (Team A) using the Fagan's formal inspection process (Process 1). The inspection outcomes were recorded using MS Excel and included information on the number and types of defects detected by each inspector, the total number of defects found in each artefact, the inspection time taken by each inspector, the total inspection time taken for each inspection session, the productivity of each inspector (calculated using the formula explained in chapter 3), etc. These eight artefacts were also inspected by another three inspectors (Team B) using ISIP (Process 2) together with ArSeC. In these inspections, the outcomes were recorded using ArSeC, as explained in chapter 4. Following the inspection of these artefacts, the two inspection teams conducted inspections on eight artefacts of the design phase of P1 which derived from the eight artefacts of the requirements analysis phase. Hence, two independent group research designs were used in the case study.

Similarly, in case study 2, inspections were conducted on eight randomly selected artefacts from the total of 42 artefacts of the requirements analysis phase of P2 by two inspection teams – Team C using Process 1, and Team D using Process 2, respectively.

This was followed by inspections on the eight corresponding artefacts of the design phase, as shown in Figure 5.1.

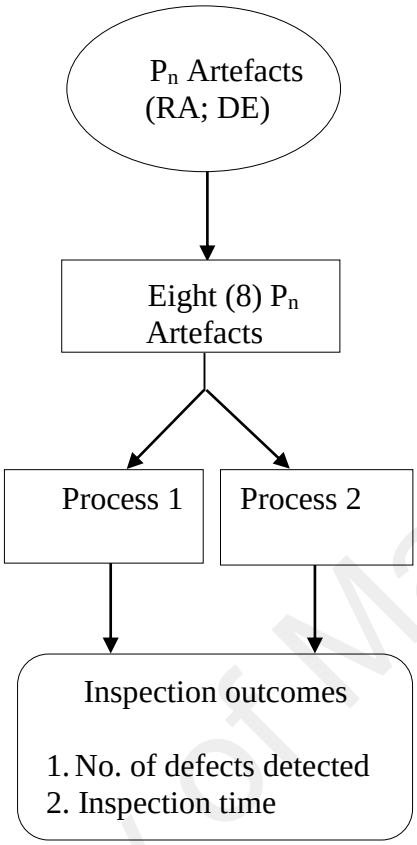


Figure 5.1: Case study – Two independent groups Research design

Keys: P_n – Project Number n, where n = 1, 2
RA – Requirements analysis phase DE – Design phase

5.1 Pilot Case Study

Before the two case studies were carried out, a pilot study using the defect injection approach was conducted. One medium-sized artefact of an in-house software project of the requirements analysis phase, which had been inspected by three inspectors earlier, were selected from Company 1 (C1), with 12 additional defects injected into the artefact. Two inspection teams were formed (excluding inspectors who had inspected the artefacts before) – one team (Team 1) with three inspectors selected at random without following any selection criteria, and another team (Team 2) with three inspectors selected using ArSeC based on the proposed inspector selection criteria of ISIP. The two artefacts (A1 and A2) were distributed to all the inspectors of both teams to prepare for inspection using

Fagan's formal inspection process (Process 1 by Team 1), and using ISIP (Process 2 by Team 2), respectively. Both teams were told to make preparation and conduct the actual inspection within two hours, respectively. Team 1 used MS Excel while Team 2 used ArSeC to record the inspection outcomes. A few problems were encountered during the inspection processes. These include: incomprehensive issues found in the design of the inspection checklist, and the inspection data recording form used to record the inspection outcomes manually; unavailability of inspectors to conduct inspection on the scheduled inspection sessions; and the use of ArSeC during the inspection process. Appropriate actions were taken to address these problems and these include: improving the design of the inspection checklists and inspection data recording form; enhancing ArSeC so as to allow inspectors to conduct inspections during the scheduled inspection sessions (i.e. discussion with other inspectors and clarification of ambiguous issues pertaining to the artefacts with the author during the inspection session) or according to the time convenient to the inspectors (i.e. clarification of ambiguities can be made later with the author via email); and organising briefing sessions for the inspectors who will use ArSeC during the actual case studies. A detailed plan of the case study was then prepared.

5.2 Case Studies

In case study 1, a moderator prepared the inspection checklists for the eight artefacts of the requirements analysis phase using MS Excel. The checklists were distributed to the three inspectors of Team A to make preparation for inspection four days before the actual inspection sessions. They were told to complete the necessary preparation within two hours. The actual face-to-face inspection processes were conducted using Process 1 at Company 1 according to the scheduled inspection sessions where each inspection session should take not more than two hours. An alarm is set 10 minutes before the end of an inspection session, and a summarized report of the inspection session is prepared. This step is followed to ensure that a consistent two-hour inspection session is applied to both

Process 1 and Process 2, as prolonged inspection time might either allow more defects to be detected by the inspectors or adversely impact on the inspectors' productivity, and thereby, affect the reliability of the case study (Shen, Zhao, & Han, 2014). Similarly, this process was repeated for inspecting the corresponding eight artefacts of the design phase of P1.

At the same time, the eight artefacts of the requirements analysis and design phases were inspected by Team B using Process 2 and ArSeC, respectively. Similarly, to ensure that each inspection session takes not more than two hours, ArSeC generates an alert 10 minutes before the end of the inspection session, and the inspection team then prepares a summarized report of the inspection session. Once the inspection session is over, all the inspectors cannot access ArSeC unless the moderator re-schedules (re-opens) a new inspection session, if necessary.

These same procedures were followed in case study 2. The following sections explain the method of data collection, and discuss the analysis of the defects detected, the inspection time taken, and calculation of the productivity of the inspection teams.

5.3 Data Collection

The ideal way of evaluating ISIP is to conduct case studies, interviews, and on-site observations. However, interviews with all the software inspectors to gather information regarding the inspection process and their inspection performance are not practical as they are time-consuming, and also likely to affect the work schedule of the inspectors (Vitharana, 2015). On-site observation is not recommended because the inspectors might feel uncomfortable or might perform differently with such an obtrusive approach. This could affect the progress of the inspection process and skew the eventual findings. Hence, an inspection data recording form was designed and used in the case studies to record the

inspection process, the defects detected, and the inspection time. In addition, The Class. Mode, and description of defects are gathered and can also be in the form.

Table 5.1: Title and purpose of each section of the inspection data recording form

Section No.	Section Title	Description
A	Project and artefacts to inspect	Project ID, project title, brief description of project, artefact ID, artefact title, and brief description of each artefact.
B	Inspector profiles (of each inspector and inspection team members)	Inspector ID, name, contact, number of years of inspection experience, number of artefacts inspected, total number of real defects detected in each artefact, total inspection time, and inspection productivity, contact details of other inspection team members.
C	Inspection process	Inspection date and time, number of inspectors involved in each inspection session, duration of each inspection session, the stopping criteria of inspection, unexpected incident(s), issues discussed, summary of the inspection outcomes.
D	Defects detected (for each artefact)	Defect ID, defect code, defect class, defect mode, brief description of each defect, inspectors who detected the defect, a summary of the total number of common and real defects detected, total number of false positives, total number of inconsistencies, and total number of ambiguous artefacts.
E	Inspection time (for each artefact)	Inspection preparation time of each inspector, total inspection preparation time of all inspectors, inspection time of each inspection session, inconsistencies resolution time, total inspection time.
F	Defect causes and effects (for each new defect)	Defect code, brief defect description, brief description of potential causes of defects, potential effects and brief description of each new defect detected, updates information on causes or effects of the existing defects.

5.3.1 Design of Inspection Data Recording Form

The inspection data recording form consists of six sections for recording data needed for evaluating ISIP, and testing the three hypotheses formulated. Table 5.1 shows the section titles and brief descriptions of these sections: Section A records details of the project and

artefacts to be inspected; Section B records the profiles of the inspectors and the contacts of other inspection team members; Section C records details of the inspection process; Section D records details of the defects detected; Section E records details of the inspection time, and Section F records details about all the new defects detected. Appendix D is a sample copy of the inspection data recording form.

5.3.2 Administration of Inspection Process and Data Recording

During the inspection process, it is important to ensure that the inspection data are accurately recorded, especially in the manual recording of the inspection outcomes by inspection Team A and Team C, respectively. On the other hand, Team B and Team D used ArSeC to record the inspection outcomes and data, thus, the productivity of each inspector was calculated, automatically.

To ensure that the inspection processes were carried out according to the planned procedures, all inspectors involved were given briefings on data collection, and the recording process, and the use of MS Excel (by Team A and Team C), and ArSeC (by Team B and Team D) to record the inspection outcomes. All inspectors were told to spend not more than two hours each to prepare for the inspection, and to conduct the actual inspection on the artefacts. Data collected from both case studies were tabulated, analysed and used to prove the hypotheses. The calculation of the number of real (actual) defects detected and the total inspection time are explained in the following sections.

5.3.3 Calculation of the total number of real (actual) defects detected

When conducting an inspection, the real defects (RD) detected could be from following groups:

RD1 – A real defect is detected and reported by all inspectors.

RD2 – A defect is detected and reported by all inspectors (i.e. reported as a defect but is actually NOT a defect). This type of defect is considered as a false positive.

RD3 – A defect is detected and reported by at least one inspector (i.e. an inspector reported a defect but it is actually NOT a defect), but it is not detected and reported by any of the other two inspectors. This type of defect is considered both as a false positive, as well as an "inconsistency", which requires re-inspection, and consolidation (voting) processes to be carried out.

RD4 – A defect is detected and reported by at least one inspector (i.e. an inspector reported a defect but it is actually NOT a defect), but it is detected and reported by any of the other two inspectors as real defect. This type of defect is also a false positive, and an "inconsistency" which requires re-inspection and consolidation (voting) processes to be carried out.

RD5 – A real defect is detected and reported by at least one inspector, but it is not detected and reported by any of the other inspectors. This type of defect is an "inconsistency" which requires re-inspection and consolidation (voting) processes to be carried out.

RD6 – A real defect is detected and reported by at least one inspector, but it is detected and reported by any of the other two inspectors as not a defect. This type of defect is an "inconsistency" which requires re-inspection and consolidation (voting) processes to be carried out.

Table 5.2 shows a summary of the six classes of defects. It is obvious that RD2-RD4 are false positives, RD3-RD6 are inconsistencies which need to go through the consolidation (voting) processes to be scheduled by a moderator. The consolidation process involves re-inspection, discussion and voting among the inspectors to resolve the inconsistencies. The final outcome of the consolidation (voting) process for RD3-RD6 could either be a correct or incorrect decision, which means a 50-50 percent decision as the decisions of the inspectors could change after re-inspection of the inconsistencies. It is therefore logical to state that an inspection team with experienced inspectors might make correct rather than incorrect decisions regarding the inconsistencies during the final consolidation

process, and at worse, identifying only a minimum number of defects as false positives (mistakes in RD2). As shown in Table 4.2 (chapter 4), the first three different compositions of inspectors are preferred in forming an experienced inspection team. Undoubtedly, the first composition which consists of three highly skilled inspectors (in a three-inspector team) is the most preferred. However, this composition of inspectors does not involve any skilled or semi-skilled inspectors in the inspection process. They have no opportunity to learn from any highly skilled inspectors and thus would not achieve the training element embedded in ISIP (Figure 4.1, chapter 4). Hence, in the two case studies, the second composition of inspectors - two highly skilled and one skilled inspectors - was chosen to form each of the four inspection teams. Table 5.3 shows the profiles of the 12 inspectors involved in the two case studies. For example, IA1 refers to the first inspector from Team A who is highly skilled, with 21 years of inspection experience, had inspected 60 projects and 1,800 artefacts, and has an average inspection productivity of detecting 55 defects per hour.

Table 5.2: Summary of defects classes, inconsistencies and final decision

Defect Group	Defect Type (T-True/real defect or F- False positive)	Inconsistency	Consolidation process needed (Voting)	Final decision (C – Correct, IC – Incorrect)
RD1	T	-	-	C
RD2	F	-	-	IC
RD3	F	✓	✓	C or IC
RD4	F	✓	✓	C or IC
RD5	T	✓	✓	C or IC
RD6	T	✓	✓	C or IC

Table 5.3: Profiles of inspectors of the four inspection teams

Inspector Code*	Skill Level	No. of years of Inspection Experience	No. of Projects Inspected	No. of Artefacts Inspected	Average Inspection Productivity (No. of defects detected/hr)
IA1	HS	21 (1994-2015)	60	1800	67
IA2	HS	26 (1989-2015)	83	1500	58
IA3	S	12 (2003-2015)	44	720	30
IB1	HS	20 (1995-2015)	76	1760	47
IB2	HS	25 (1990-2015)	102	1070	52
IB3	S	10 (2005-2015)	48	950	32
IC1	HS	22 (1993-2015)	79	930	46
IC2	HS	32 (1983-2015)	66	1620	67
IC3	S	11 (2004-2015)	93	780	31
ID1	HS	34 (1981-2015)	107	1940	64
ID2	HS	20 (1995-2015)	69	1380	45
ID3	S	14 (2001-2015)	39	650	34

* Ixi: Inspector Team x number i (x = A, B, C or D; i = 1..3) HS: Highly Skilled S: Skilled

Based on Table 5.2, if it is assumed that the final decisions made during the final consolidation processes for the defect groups, RD3-RD6, are correct (i.e. the final decisions made on RD3-RD4 are not defects, RD5-RD6 are defects), then by implication, the number of defects in RD5 and RD6 will be counted once only, and double counting would happen in RD1 only. The total number of common defects detected by the second and/or third inspectors can be determined from the defect codes, as shown in Table 4.3 and Table 4.4 (chapter 4). Based on the explanations given in section 3.2 (chapter 3), the total number of real defects detected by an inspection team is calculated using the following formula:

Total number of real defects detected by an inspection team (TNARD) = $\sum_{i=1}^{i=3}$ RD1i + $\sum$RD5 + $\sum$RD6 - ($\sum_{i=2}^{i=3}$ RD_RD1i)

Where:

$RD1_i$: Total number of real defects detected in RD1 by all three inspectors, i = 1.. 3.

RD_RD1_i : Total number of common and real defects detected by the second and/or third inspectors in RD1, i = 2, 3.

5.3.4 Calculation of total inspection time

As mentioned in section 3.2 (chapter 3), the total inspection time spent on an artefact by an inspector is calculated using the formula:

Inspection Time = Preparation time + Inspection session time + Inconsistencies resolution time

The preparation time is the time spent by each inspector to get ready for the actual inspection process. Hence, the total preparation time by all inspectors (i.e. sum of preparation time, ∑PT, for an inspection team) should be included in the calculation of the total inspection time. However, to determine the inspection session time and inconsistencies resolution time, there are two perspectives to consider based on the actual inspection situations: (i) all inspectors are available during the scheduled inspection session (for both processes), and they can resolve all the inconsistencies together. In this case, the inspection session time and inconsistencies resolution time are counted ONCE only; and (ii) one inspector (for a three-inspector team) or two-trree inspectors (for a five-inspector team), respectively, might not be available during the scheduled inspection session and/or inconsistencies resolution session. In this case, the total inspection session time and the total inconsistencies resolution time should be calculated using the following formulas:

Total inspection session time (TIST) = Scheduled inspection session time (SIST) + ∑Inspection session time of those inspector(s) who conduct inspection at their own convenient time ($\sum_{i=1}^{i=j}$ IST).

The value of $\sum_{i=1}^{i=j}$ IST is 0 if all the inspectors are available during the scheduled inspection session(s), and vice versa (i.e., SIST = 0) if all inspectors are not available for the scheduled inspection session(s). Both SIST and $\sum_{i=1}^{i=j}$ IST are > 0, when at least one or more inspectors (but not all the inspectors) are either available or not available for the scheduled inspection session.

Total inconsistencies resolution time (TIRT) = Scheduled inconsistencies resolution time (SIRT) + ∑Inconsistencies resolution time of those inspector(s) who conduct inconsistencies resolution at their own convenient time ($\sum_{i=1}^{i=k}$ IRT).

Similarly, the value of $\sum_{i=1}^{i=k}$ IRT is 0 if all the inspectors are available during the scheduled inconsistencies resolution session(s), and vice versa (i.e., SIRT = 0) if all inspectors are not available during the scheduled inconsistencies resolution session(s). Both SIRT and $\sum_{i=1}^{i=j}$ IRT are > 0, when at least one or more inspectors (but not all the inspectors) are either available or not available during the scheduled inconsistencies resolution session(s).

Hence, the total inspection time taken to inspect an artefact by an inspector (TIT-I) is: PT + SIST + IST + SIRT + IRT

The value of IST and IRT is 0 if the inspector is available during the scheduled inspection session(s) and inconsistencies resolution session(s), and vice versa (i.e. IST and IRT are ≠ 0) if the inspectors are not available during both sessions, i.e. the value of SIST and SIRT is 0. It is also possible for other combinations such as SIST ≠ 0, IST = 0, SIRT = 0, and IRT ≠ 0.

The total inspection time to inspect an artefact by an inspection team (TIT-T) is: $\sum_{i=1}^{i=n}$ PT + TIST + TIRT

where,

n: Total number of inspectors.

5.3.5 Calculation of the productivity of an inspection team

The productivity of each inspection team is calculated using the formula explained in section 3.2 (chapter 3):

Productivity of an inspection team = Total number of real (actual) defects detected by the inspection team / Total inspection time taken to inspect an artefact by the inspection team (TIT-T)

Where,

Total number of real (actual) defects detected = Total number of defects detected - Total number of false positives.

5.4 Data Analysis

The purpose of this research is to confirm a hypothesis that using ISIP (Process 2, an enhanced formal inspection process) can improve the quality of software inspection when compared with using the formal inspection process (Process 1). Both Process 1 and Process 2 are the independent variables, and they are also the treatments in this study. The number of real defects detected, the total inspection time, and the productivity of the inspection teams are dependent variables as their values depend on the type of inspection process model used.

In this research, both the number of defects detected and inspection time are directly measured and calculated during the inspection processes. However, the productivity of an inspection team is an indirect measurement as it is calculated using the formula:

Total number of real (actual) defects detected / Total inspection time.

Hence, the measurement scales of these dependent variables are ratio scales as they can have 0 value if an artefact is of zero length, and hence, will have zero defect, zero inspection time, and no measurement of the inspector productivity, respectively.

Table 5.4: Total number of real defects detected, total inspection time, and productivity of inspection team

Project ID	Artefact ID	Total No. of Real Defects Detected		Total Inspection Time (min)		Productivity of Inspection Team	
		Process 1	Process 2	Process 1	Process 2	Process 1	Process 2
P1	P1-A1-R	26	34	229	221	0.11	0.15
P1	P1-A2-R	15	26	218	218	0.07	0.12
P1	P1-A3-R	18	28	220	214	0.08	0.13
P1	P1-A4-R	15	26	217	212	0.07	0.12
P1	P1-A5-R	12	17	216	206	0.06	0.08
P1	P1-A6-R	21	23	216	201	0.10	0.11
P1	P1-A7-R	13	26	217	210	0.06	0.12
P1	P1-A8-R	21	28	223	218	0.09	0.13
P2	P2-A1-R	24	35	225	219	0.11	0.16
P2	P2-A2-R	22	33	226	219	0.10	0.15
P2	P2-A3-R	16	19	212	206	0.08	0.09
P2	P2-A4-R	7	13	209	193	0.03	0.07
P2	P2-A5-R	30	36	230	223	0.13	0.16
P2	P2-A6-R	18	27	223	213	0.08	0.13
P2	P2-A7-R	14	21	222	218	0.06	0.10
P2	P2-A8-R	31	39	234	227	0.13	0.17
P1	P1-A1-D	33	32	229	221	0.14	0.14
P1	P1-A2-D	24	36	228	220	0.11	0.16
P1	P1-A3-D	27	35	227	220	0.12	0.16
P1	P1-A4-D	8	11	210	199	0.04	0.06
P1	P1-A5-D	16	25	220	214	0.07	0.17
P1	P1-A6-D	21	31	229	219	0.09	0.14
P1	P1-A7-D	13	21	218	205	0.06	0.10
P1	P1-A8-D	22	34	220	224	0.10	0.15
P2	P2-A1-D	24	22	216	200	0.11	0.11
P2	P2-A2-D	22	28	224	218	0.10	0.13
P2	P2-A3-D	14	26	216	207	0.06	0.13
P2	P2-A4-D	27	42	221	223	0.12	0.18
P2	P2-A5-D	24	37	224	214	0.11	0.17
P2	P2-A6-D	14	24	214	208	0.07	0.16
P2	P2-A7-D	14	28	214	213	0.07	0.13
P2	P2-A8-D	27	37	224	215	0.12	0.17

Keys:
Pn – Project No. n, n = 1, 2
Am – Artefact No., m = 1,..,8
R – Requirements analysis phase
D – Design phase
Process 1 – Formal Inspection and facilitated using MS Excel
Process 2 – ISIP and facilitated using ArSeC

The data of the three dependent variables were collected from the two case studies and analysed using IBM SPSS Statistics 22. A statistician was consulted on the appropriate statistical tests to use to prove the hypotheses formulated in chapter 3. In this research, the data of the dependent variables consist of a matched pair of data: i) the number of real (actual) defects detected by two inspection teams (Team A and Team B; Team C and Team D) using Process 1 and Process 2; ii) the matched pair of total inspection time; and iii) the matched pair of the respective productivity of the Inspection teams. Hence, the Paired-Samples T Test is selected to test the hypothesis that there is no difference between the mean values of two variables. Table 5.4 shows the total number of real (actual) defects detected by each inspection team, the total inspection time to inspect each artefact by an inspection team, and the productivity of each inspection team inspecting each artefact. For example, from the artefact P1-A1-R, from the requirements analysis phase of project 1 (P1), 26 defects were detected by inspection Team A using Process 1 in 229 minutes, and 34 defects were detected by Team B using Process 2 in 221 minutes.

5.4.1 Data Screening

Before testing the hypotheses, histograms, skewness and the Kolmogorov-Smirnov test were plotted and used to determine whether the data of the dependent variables are of normal distribution, and to identify any possible departures from the normal distribution. Figure 5.2 shows the histograms of the six sets of data – total number of real defects detected by each inspection team, total inspection time taken to inspect each artefact by an inspection team, and the productivity of each inspection team. Based on this figure and Table 5.5 which shows skewness in the distribution of data, all the data seem to be distributed normally (i.e. all histograms show trend of normal distribution, and all the values of skewness are less than 1.0). However, Table 5.6, which shows the results of Kolmogorov-Smirnov test of normality indicates that the significant value (p) of the data for total inspection time using Process 2 (ISIP and ArSeC) is 0.020 (i.e. $p < 0.05$).

This indicates that the data is not distributed normally. Hence, all the data have met the data considerations – variables are ratio level of measurement, and the response for each test subject and its matched control subject is in the same case in the data file; and the assumptions that observations for each pair should be made under the same conditions, the mean differences should be normally distributed, and variances of each variable can be equal or unequal, for paired-samples T Test (Paired-Samples T Test, 2015), except for the data set – total inspection time using Process 2.

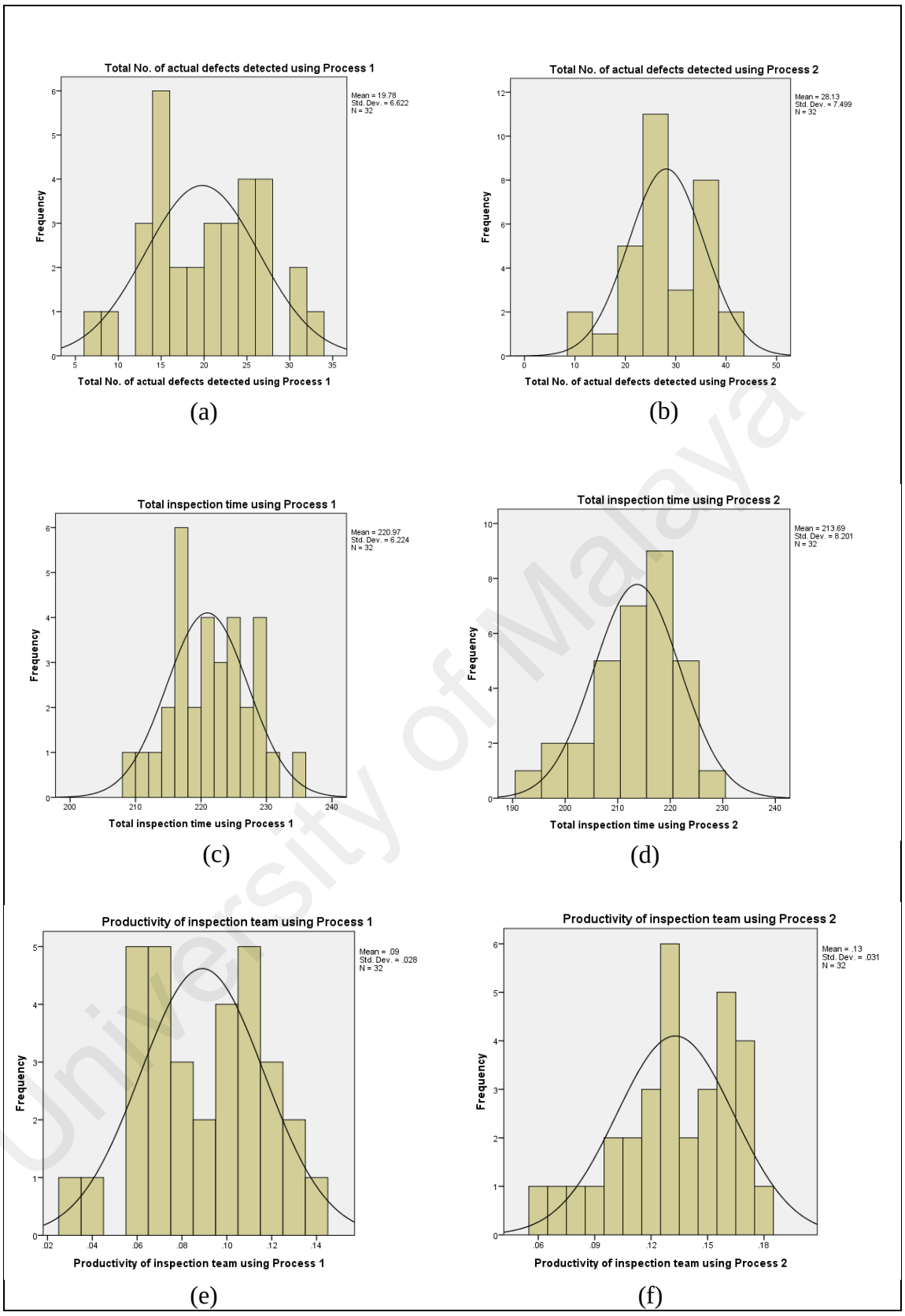


Figure 5.2: Histograms of data (total number of real defects detected, inspection time, and productivity of inspection team)

Table 5.5: Skewness the distribution of data

	Total no. of correct defects detected using Process 1	Total no. of real defects detected using Process 2	Total inspection time using Process 1	Total inspection time using Process 2	Productivity of inspection team using Process 1	Productivity of inspection team using Process 2
Valid N	32	32	32	32	32	32
Missing	0	0	0	0	0	0
Mean	19.78	28.13	220.97	213.69	.0891	.1328
Median	21.00	28.00	220.50	214.50	.0900	.1300
Std. Deviation	6.622	7.499	6.224	8.201	.2763	.3113
Variance	43.854	56.242	38.741	67.254	.001	.001
Skewness	.067	-.348	.037	-.724	-.121	-.608
Std. Error of Skewness	.414	.414	.414	.414	.414	.414
Kurtosis	-.707	-.254	-.647	-.050	-.740	-.256
Std. Error of Kurtosis	.809	.809	.809	.809	.809	.809

Note: Value of skewness < 1.0 indicates normal distribution, otherwise not normal distribution.

Table 5.6: Kolmogorov-Smirnov tests of normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Total No. of real defects detected using Process 1	.122	32	.200*	.971	32	.524
Total No. of real defects detected using Process 2	.100	32	.200*	.975	32	.646
Total inspection time using Process 1	.090	32	.200*	.981	32	.826
Total inspection time using Process 2	.169	32	**.020**	.946	32	.113
Productivity of inspection team using Process 1	.130	32	.184	.965	32	.366
Productivity of inspection team using Process 2	.121	32	.200*	.947	32	.116

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Note: Kolgomorov-Smirnov test, if sig. value, $p > 0.05$, indicates normal distribution, otherwise not normal distribution.

The Paired-Samples T Test (a parametric test) is used to test the first and third hypotheses. On the other hand, hypothesis 2 is tested using the equivalent nonparametric test - the Related-Samples Wilcoxon Signed Rank test - as the matched paired data of the total inspection time using Process 2 is not normally distributed.

5.4.2 Test of Hypotheses

5.4.2.1 Hypothesis 1

In section 3.1.5, Hypothesis 1 is stated as follows:

H0: The total number of real defects detected using Process 1 is the same as the total number of real defects detected using Process 2.

H1: The total number of real defects detected using Process 2 is more than the total number of real defects detected using Process 1.

The mean values of the total number of real defects detected using Process 1 and Process 2 are 19.78 and 28.13, respectively, as shown in Table 5.7. The Pearson correlation between these paired data is 0.838, as shown in Table 5.8. This value is close to 1.0, implying that the total number of real defects detected using Process 1 and Process 2 are closely related.

At $\alpha = 0.05$, there is a difference in the mean value of the total number of real defects detected between Process 1 and Process 2. Since the significance value for the average total number of real defects detected, at the degree of freedom, df = 31, and at 2-tailed test, is p = 0.000, which is less than 0.05, as shown in Table 5.9, we can conclude that the average increase of 8.344 in the total number of real defects detected is not due to chance variation, but is attributed to the use of ISIP and ArSeC. Thus, we reject H0 and accept H1. Also, the value of the mean difference between Process 1 and Process 2 (i.e. -8.344), as shown in Table 5.9, implies that the average total number of real defects detected using Process 2 has increased by 42.2% (i.e. 8.344 / 19.78 x 100%).

Table 5.7: Paired-Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Total no. of real defects detected using Process 1	19.78	32	6.622	1.171
	Total no. of real defects detected using Process 2	28.13	32	7.499	1.326

Table 5.8: Paired-Samples Correlations

		N	Correlation	Sig.
Pair 1	Total no. of real defects detected using Process 1 & Total no. of real defects detected using Process 2	32	.838	.000

Table 5.9: Paired-Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	Total no. of real defects detected using Process 1 - Total no. of real defects detected using Process 2	-8.344	4.108	.726	-9.825	-6.863	-11.489	31	.000

5.4.2.2 Hypothesis 2

Hypothesis 2, as stated below, was tested using the Related-samples Wilcoxon Signed Rank test.

H0: The total inspection time using Process 1 is the same as the total inspection time using Process 2.

H1: The total inspection time using Process 2 is less than the total inspection time using Process 1.

Wilcoxon Signed Rank test is the equivalent nonparametric test of the Paired-Samples T Test as data on the total inspection time using Process 2 are not normally distributed. At $\alpha = 0.05$, the results from Wilcoxon Signed Rank test suggest rejection of the null hypothesis H0, but acceptance of H1 because the value of $p = 0.000 < 0.05$. This implies that there is a difference in the outcome when using Process 2 (ISIP and ArSeC) – improvement to the inspection process reflected by a reduction in the total inspection time, as shown in Table 5.10 below. As shown in Table 5.5, the mean values of the total inspection time when using Process 1 and Process 2 are 220.97 minutes and 213.69 minutes, respectively. The mean difference is -7.31 – a 3.3% (i.e. 7.31 / 220.97 x 100%)

reduction in the average total inspection time when Process 2 was used to inspect the artefacts.

Table 5.10: Test of hypothesis 2 using Related-Samples Wilcoxon Signed Rank test: Summary

Test of Hypothesis 2: Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between Total inspection time using Process 1 and Total inspection time using Process 2 equals 0.	Related-Samples Wilcoxon Signed Rank Test	.000	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

5.4.2.3 Hypothesis 3

Hypothesis 3 was formulated as follows:

H0: The productivity of the inspection team using Process 1 is the same as the productivity of the inspection team using Process 2.

H1: The productivity of the inspection team using Process 2 is higher than the productivity of the inspection team using Process 1.

The hypothesis was tested using the Paired-Samples T Test. The mean values of the productivity of the inspection team using Process 1 and Process 2 are 0.0891 and 0.1328, respectively, as shown in Table 5.11. The correlation between these paired data is 0.723, as shown in Table 5.12. This value is fairly close to 1.0, implying that the productivity of the inspection teams using Process 1 and Process 2 are closely related.

At $\alpha = 0.05$, the test outcome indicates that there is a difference in the mean value of the productivity between the inspection team that used Process 1 and the team that used Process 2. The significance value for the average productivity of the inspection team, at the degree of freedom, df = 31, and at 2-tailed test, is p = 0.000, which is less than 0.05,

as shown in Table 5.13, hence, we can conclude that the average increase of 0.04375 in the productivity of the inspection team is not due to chance, but is attributed to ISIP and ArSeC. Hence, we reject H0 and accept H1. Also, the value of the mean difference between Process 1 and Process 2 (i.e. -0.04375), as shown in Table 5.13, implies that the average productivity of the inspection team that used Process 2 has increased by 49.1% (8.344 / 19.78 x 100%).

Table 5.11: Paired-Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Productivity of inspection team using Process 1	.0891	32	.02763	.00488
	Productivity of inspection team using Process 2	.1328	32	.03113	.00550

Table 5.12: Paired-Samples Correlations

		N	Correlation	Sig.
Pair 1	Productivity of inspector using Process 1 & Productivity of inspector using Process 2	32	.723	.000

Table 5.13: Paired-Samples Test

	Paired Differences					t	df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
Pair 1 Productivity of inspection team using Process 1 - Productivity of inspection team using Process 2	-.04375	.02211	.00391	-.05172	-.03578	-11.195	31	.000

5.5 Inspection Stopping Criteria

During each inspection session, inconsistencies and ambiguous issues are often raised by the inspectors. Furthermore, it is impossible to determine whether all or most of the serious defects have been detected by the inspection team during the first inspection session (i.e. one inspection session is needed only). Hence, a moderator might have to schedule additional inspection sessions to allow the inspection team to resolve the inconsistencies and ambiguous issues raised by the inspectors during the first inspection session. At the same time, the team has the opportunity to detect other serious defects which have not yet been detected. Thus, for the latter case, an inspection stopping condition or criterion has to be established to decide when to stop conducting further inspections.

In this research, the criterion is the percentage of real (actual) defects detected during the additional inspection session compared with the total number of real defects detected in the previous inspection session. This percentage is decided by the inspection team based on the importance and size of an artefact, for example, for an important and larged-size artefact, a small percentage, for example 5%, is used. For artefacts which are not important and of smaller size, a larger percentage of between 5%-11% is used. When the percentage of the total number of real defects detected is less than the set inspection stopping percentage, the inspection process will be considered complete, and a summarized report on all the inspection sessions and the defects detected is prepared. In the two case studies, 5% was used as the inspection stopping criterion as all the 32 randomly selected artefacts are important and large-sized artefacts.

5.6 Other Findings of the Case Studies

Besides testing the three hypotheses to evaluate the efficiency of ISIP, this research also investigates three other related issues:

i. the defect density in each artefact based on the total number of real defects present in each artefact,

ii. the efficiency of each inspection model based on the total number of real defects present in each artefact.

iii. the most common defects that are detected in the requirements analysis and design phases of the two case studies.

5.6.1 Defect Density

The two large software development projects (P1 and P2) which were used in the two case studies were in-house software development projects of company 1 (C1) and company 2 (C2) which were completed in three and half years and three years, respectively. Hence, the total number of real defects that was present in the requirements analysis and design phases was determined and documented at the completion of the two projects. The defect densities of the 32 artefacts of these two projects, are shown in Table 5.14, and had been calculated using formula (2) stated in section 3.2 (chapter 3):

Defect Density = Total number of real defects present in each artefact / Size of the artefact

Table 5.14: Defect density of artefacts

Artefact ID	Size of Artefact (N1)	Actual no. of Real Defects (N2)	Actual Defect Density, DD = N2 / N1
P1-A1-R	27	36	1.33
P1-A2-R	23	26	1.13
P1-A3-R	23	29	1.26
P1-A4-R	22	27	1.22

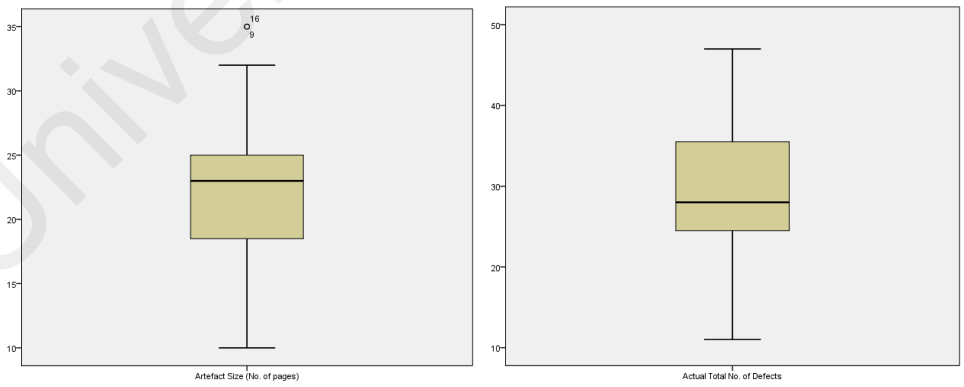
P1-A5-R	17	17	1.00
P1-A6-R	18	23	1.28
P1-A7-R	19	27	1.42
P1-A8-R	24	28	1.12
P2-A1-R	35	36	1.03
P2-A2-R	32	35	1.09
P2-A3-R	12	19	1.58
P2-A4-R	10	13	1.30
P2-A5-R	30	39	1.30
P2-A6-R	25	27	1.08
P2-A7-R	23	24	1.04
P2-A8-R	35	41	1.17
P1-A1-D	29	34	1.17
P1-A2-D	25	37	1.48
P1-A3-D	25	35	1.40
P1-A4-D	13	11	0.85
P1-A5-D	23	25	1.09
P1-A6-D	19	32	1.68
P1-A7-D	16	22	1.38
P1-A8-D	23	34	1.48
P2-A1-D	18	24	1.33
P2-A2-D	25	28	1.12
P2-A3-D	19	27	1.42
P2-A4-D	23	47	2.04
P2-A5-D	23	40	1.74
P2-A6-D	21	25	1.19
P2-A7-D	16	28	1.75
P2-A8-D	31	38	1.23

Keys:

Pn – Project No. n, n = 1, 2
Am – Artefact No., m = 1,..,8
R – Requirements analysis phase
D – Design phase
M1 (Process 1) – Formal Inspection
M2 (Process 2) – ISIP

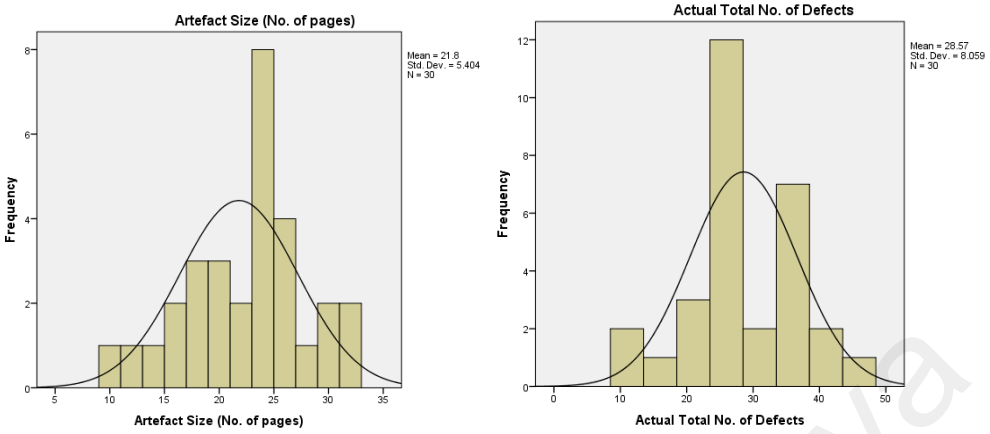
The artefact that contains the most defects from the requirements analysis and design phases are artefact 3 of P2 (P2-A3-R; size: 12 pages), and artefact 4 of P2 (P2-A4-D; size: 23 pages) with defect density of 1.58 and 2.04, respectively. On the other hand, the artefact that contains the least defects from the requirements analysis and design phases are artefact 5 of P1 (P1-A5-R; size: 17 pages), and artefact 4 of P1 (P1-A4-D; size: 13 pages) with defect density of 1.00 and 0.85, respectively.

To determine whether the actual number of defects present in an artefact is correlated with the size of an artefact, a test of correlation was performed on these two variables. The box plot was first used to identify any outliers in the variables. Figure 5.3 (a) shows that there is no outlier in the variable, actual total number of defects. However, there are two outliers in the variable, size of artefact, as shown in Figure 5.3 (b). Hence, these two outliers were removed. Histograms of the remaining 30 sets of data of the two variables were plotted to show the trend in data distribution (Figure 5.4). Table 5.15 shows the values of skewness of the two variables (-0.053 and -0.207), both less than 1, indicating that the data are distributed normally. Table 5.16 shows the results of Kolmogorov-Smirnov test of normality, which indicates that the significant value (p) of the data for the actual total number of defects and size of artefact are 0.065 and 0.200, respectively, both values are > 0.05. This again indicates that the data are distributed normally. As shown in Table 5.17, the Pearson Correlation coefficient between actual total number of defects and size of artefact is 0.756, significant at the 0.01 level. This shows that there is a strong correlation between the total number of defects present in an artefact and the size of the artefact, as the value is fairly close to 1.



(a): Total number of defects **(b): Size of artefact (Number of pages)**

Figure 5.3: Box plot of variables



(a) Total number of defects (b): Size of artefact (Number of pages)

Figure 5.4: Histograms of variables

Table 5.15: Skewness in the distribution of data (Size of artefact and Defect density)

	N	Mean	Std. Deviation	Variance	Skewness	
	Statistic	Statistic	Statistic	Statistic	Statistic	Std. Error
Actual Total No. of Defects	30	28.57	8.059	64.944	-.053	.427
Artefact Size (No. of pages)	30	21.80	5.404	29.200	-.207	.427
Valid N (list wise)	30					

Note: If value of skewness < 1.0, implies normal distribution, otherwise not normal distribution.

Table 5.16: Kolmogorov-Smirnov test of normality (Size of artefact and Defect density)

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Actual Total No. of Defects	.128	30	.200*	.979	30	.808
Artefact Size (No. of pages)	.155	30	.065	.972	30	.590

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Table 5.17: Pearson Correlation Test

		Actual Total No. of Defects	Artefact Size (No. of pages)
Actual Total No. of Defects	Pearson Correlation	1	.756**
	Sig. (1-tailed)		.000
	N	30	30
Artefact Size (No. of pages)	Pearson Correlation	.756**	1
	Sig. (1-tailed)	.000	
	N	30	30

**. Correlation is significant at the 0.01 level (1-tailed).

5.6.2 Evaluation of the Efficiency of the Inspection Processes – An Alternative Approach

Besides using a statistical approach to evaluate the efficiency of the inspection processes (Process 1 and Process 2), an alternative approach is to calculate and compare the total number of real defects detected by each inspection Process in relation to the total number of real defects present in each artefact, as shown in Table 5.18 below. Table 5.19 shows the inspectors (i.e. the inspection teams) and the artefacts to be inspected. Based on the figures shown in these two tables, it is obvious that Team B which used ISIP and ArSeC to inspect the eight artefacts – P1-A2-R, P1-A5-R, P1-A6-R, P1-A8-R (four artefacts from the requirements analysis phase); P1-A3-D, P1-A4-D, P1-A5-D, P1-A8-D (four artefacts from the design phase) – detected all (100%) real defects present in these eight artefacts. Similarly, Team D which also used ISIP and ArSeC to inspect the five artefacts – P2-A3-R, P2-A4-R, P2-A6-R (three artefacts from the requirements analysis phase); P2-A2-D, P2-A7-D (two artefacts from the design phase) - also detected all (100%) real defects present in these five artefacts. These findings show that using ISIP and ArSeC to inspect 40.6% (13 / 32 artefacts x 100%) of the artefacts, has succeeded detecting all the defects present in each of these artefacts. On the other hand, Team C inspected only one artefact (P2-A1-D) and succeeded in detecting all (100.0%) of the defects present in this artefact. This shows that using the formal inspection process and

MS Excel to inspect only 3.1% (1 / 32 artefact x 100%) of the artefacts succeeded in detecting all the defects present in the artefacts.

Further analysis shows that the efficiency of Process 1 (formal inspection process and MS Excel) and Process 2 (ISIP and ArSeC) ranges from 48.1%-100.0%, and 87.5%-100.0%, respectively. The difference in the minimum efficiency level of these two processes is 39.4% (i.e. 87.5% - 48.1%) in the inspection of artefacts. The average efficiency of each process is 68.0% and 96.8%, respectively, thus, ISIP shows 28.8% better efficiency. Hence, it is obvious that the use of ISIP and ArSeC can improve the quality of software inspection, markedly.

Table 5.18: Efficiency of inspection processes based on the actual total number of real defects present in each artefact

Artefact ID	Size of Artefact (N1)	Actual No. of Real Defects Detected (N2)	Total No. of Real Defects Detected (N3)		Efficiency of Inspection Model, N4 = N3 / N2 x 100%	
			M1 (N3a)	M2 (N3b)	M1 (N4a = N3a / N2 x 100%)	M2 (N4b = N3b / N2 x 100%)
P1-A1-R	27	36	26	34	72.2	94.4
P1-A2-R	23	26	15	26	57.7	**100.0**
P1-A3-R	23	29	18	28	62.0	96.6
P1-A4-R	22	27	15	26	55.6	96.3
P1-A5-R	17	17	12	17	70.6	**100.0**
P1-A6-R	18	23	21	23	91.3	**100.0**
P1-A7-R	19	27	13	26	48.1	96.3
P1-A8-R	24	28	21	28	75.0	**100.0**
P2-A1-R	35	36	24	35	66.7	97.2
P2-A2-R	32	35	22	33	62.3	94.3
P2-A3-R	12	19	16	19	84.2	**100.0**
P2-A4-R	10	13	7	13	53.8	**100.0**
P2-A5-R	30	39	30	36	76.9	92.3
P2-A6-R	25	27	18	27	66.7	**100.0**
P2-A7-R	23	24	14	21	66.7	87.5
P2-A8-R	35	41	31	39	75.6	95.1
P1-A1-D	29	34	33	32	97.1	94.1
P1-A2-D	25	37	24	36	64.9	97.3
P1-A3-D	25	35	27	35	77.1	**100.0**
P1-A4-D	13	11	8	11	72.7	**100.0**
P1-A5-D	23	25	16	25	64.0	**100.0**
P1-A6-D	19	32	21	31	65.6	96.9

P1-A7-D	16	22	13	21	59.1	95.5
P1-A8-D	23	34	22	34	64.7	**100.0**
P2-A1-D	18	24	24	22	**100.0**	91.7
P2-A2-D	25	28	22	28	78.6	**100.0**
P2-A3-D	19	27	14	26	51.9	96.3
P2-A4-D	23	47	27	42	57.4	89.4
P2-A5-D	23	40	24	37	60.0	92.5
P2-A6-D	21	25	14	24	56.0	96.0
P2-A7-D	16	28	14	28	50.0	**100.0**
P2-A8-D	31	38	27	37	71.1	97.4
Average					67.99%	96.78%

Keys:

Pn – Project No. n, n = 1, 2
Am – Artefact No., m = 1,..,8
R – Requirements analysis phase
D – Design phase
M1 (Process 1) – Formal Inspection
M2 (Process 2) – ISIP

Table 5.19: Inspectors and the artefacts they are assigned to inspect

Inspector Code*	**Skill Level**	**Artefact Code (Requirements Analysis Phase)**	**Artefact Code (Design Phase)**
IA1	H	[Use MS Excel]	[Use MS Excel]
IA2	H		
IA3	S	P1-A1-R, P1-A2-R, P1-A3-R, P1-A4-R, P1-A5-R, P1-A6-R, P1-A7-R, P1-A8-R.	P1-A1-D, P1-A2-D, P1-A3-D, P1-A4-D, P1-A5-D, P1-A6-D, P1-A7-D, P1-A8-D.
IB1	HS	[Use ArSeC]	[Use ArSeC]
IB2	HS		
IB3	S	P1-A1-R, P1-A2-R, P1-A3-R, P1-A4-R, P1-A5-R, P1-A6-R, P1-A7-R, P1-A8-R.	P1-A1-D, P1-A2-D, P1-A3-D, P1-A4-D, P1-A5-D, P1-A6-D, P1-A7-D, P1-A8-D.
IC1	HS	[Use MS Excel]	[Use MS Excel]
IC2	HS		
IC3	S	P2-A1-R, P1-A2-R, P2-A3-R, P1-A4-R, P2-A5-R, P1-A6-R, P2-A7-R, P1-A8-R.	P2-A1-R, P1-A2-D, P2-A3-R, P1-A4-D, P2-A5-R, P1-A6-D, P2-A7-R, P1-A8-D.
ID1	HS	[Use ArSeC]	[Use ArSeC]
ID2	HS		
ID3	S	P2-A1-R, P1-A2-R, P2-A3-R, P1-A4-R, P2-A5-R, P1-A6-R, P2-A7-R, P1-A8-R.	P2-A1-R, P1-A2-D, P2-A3-R, P1-A4-D, P2-A5-R, P1-A6-D, P2-A7-R, P1-A8-D.

* Ixi: Inspector Team x number i (x = A, B, C or D; i = 1..3) HS: Highly Skilled S: Skilled

5.6.3 Productivity of Inspectors

The productivity of each inspector is calculated to determine which inspector, among the 12 inspectors, has the highest, or the lowest productivity, respectively. The productivity of each inspector is calculated using the formula below:

Productivity of an inspector = Total No. of real defects detected by the inspector / Total inspection time

Where,

Total No. of real defects detected = Total No. of defects detected – Total No. of false positives

The calculation of the total inspection time of an inspector is different from the total inspection time of an inspection team. As the scheduled inspection session time is incurred from team efforts in finding the new defects which were not detected during the preparation stage, hence, this scheduled inspection time is not included in the calculation. However, if an inspector conducts on inspection session at his/her own convenient time (i.e., IST, to find other defects which were not detected during the preparation stage), then the sum of these independent inspection session times (i.e. $\sum$ IST) are considered in the calculation of the total inspection time. However, the scheduled inconsistencies resolution time is not considered because it is the time incurred to resolve inconsistencies and ambiguous defects detected by all the inspectors, and not to find new defects. Hence, the total inspection time to inspect an artefact by an inspector is calculated as follows:

The total inspection time (in min) = Total preparation time the inspector spent to find the defects + $\sum$ IST

where,

$\sum$ IST = Sum of inspection session times of the inspector who conducted the session inspections at his/her own convenient times.

Table 5.20 shows the total number of real defects detected, the total inspection time (in min), and the productivity of each inspector calculated using the formulas explained above. The average productivity of each inspector was calculated and tabulated in Table 5.21 together with the respective inspection details from Table 5.3 – inspection skill level, number of years of inspection experience (NYIE), and the number of artefacts inspected (NAI), previously.

Table 5.20: Total number of real defects detected and total inspection time of each inspector

Inspection Team A									
Requirements Analysis Phase (Using Process 1 and MS Excel)									
Artefact ID	**Inspector Code**								
	IA1			**IA2**			**IA3**		
	TNCD	**TIT**	**Prod-A1**	**TNCD**	**TIT**	**Prod-A2**	**TNCD**	**TIT**	**Prod-A3**
P1-A1-R	26	154	0.168	27	147	0.183	25	160	0.156
P1-A2-R	13	134	0.097	10	127	0.078	15	169	0.088
P1-A3-R	22	132	0.166	21	120	0.175	27	178	0.151
P1-A4-R	18	130	0.138	14	132	0.106	20	151	0.132
P1-A5-R	14	135	0.103	8	124	0.064	9	165	0.054
P1-A6-R	17	121	0.14	16	117	0.136	20	170	0.117
P1-A7-R	23	137	0.167	16	134	0.119	15	148	0.101
P1-A8-R	17	145	0.117	15	139	0.107	20	159	0.125
Sub-Total-1	-	-	1.096	-	-	0.968	-	-	0.924
Design Phase (Using Process 1 and MS Excel)									
Artefact ID	Inspector Code								
	IA1			IA2			IA3		
	TNCD	TIT	Prod-A1	TNCD	TIT	Prod-A2	TNCD	TIT	Prod-A3
P1-A1-D	20	149	0.134	13	151	0.086	17	163	0.104
P1-A2-D	28	149	0.187	25	138	0.181	31	157	0.197
P1-A3-D	28	161	0.173	30	138	0.217	23	166	0.138
P1-A4-D	7	136	0.051	10	129	0.077	8	145	0.055
P1-A5-D	14	144	0.097	16	131	0.122	21	159	0.132
P1-A6-D	26	159	0.163	24	140	0.171	27	160	0.168
P1-A7-D	16	143	0.111	10	119	0.084	9	152	0.059
P1-A8-D	23	145	0.158	21	136	0.154	22	153	0.143

Sub-Total-2	-	-	1.074	-	-	1.092	-	-	0.996
Total	-	-	2.170	-	-	2.060	-	-	1.920
Average	-	-	0.134	-	-	0.129	-	-	0.120

Inspection Team B

Requirements Analysis Phase (Using Process 2 and ArSeC)

Artefact ID	Inspector Code								
	IB1			IB2			IB3		
	TNCD	TIT	Prod-B1	TNCD	TIT	Prod-B2	TNCD	TIT	Prod-B3
P1-A1-R	26	115	0.226	27	143	0.188	25	153	0.163
P1-A2-R	13	119	0.109	10	140	0.071	15	143	0.104
P1-A3-R	22	128	0.171	21	138	0.152	27	158	0.17
P1-A4-R	18	129	0.139	14	114	0.122	20	145	0.137
P1-A5-R	14	125	0.112	8	136	0.058	9	147	0.061
P1-A6-R	17	130	0.13	16	126	0.126	20	144	0.138
P1-A7-R	23	123	0.186	16	130	0.123	15	147	0.102
P1-A8-R	17	140	0.121	15	140	0.107	20	140	0.142
Sub-Total-1	-	-	1.194	-	-	0.947	-	-	1.017

Design Phase (Using Process 2 and ArSeC)

Artefact ID	Inspector Code								
	IB1			IB2			IB3		
	TNCD	TIT	Prod-B1	TNCD	TIT	Prod-B2	TNCD	TIT	Prod-B3
P1-A1-D	20	120	0.166	13	143	0.09	17	144	0.118
P1-A2-D	28	136	0.205	25	142	0.176	31	146	0.212
P1-A3-D	28	133	0.21	30	142	0.211	23	134	0.171
P1-A4-D	7	135	0.051	10	132	0.075	8	149	0.053
P1-A5-D	14	137	0.102	16	138	0.115	21	139	0.151
P1-A6-D	26	132	0.196	24	141	0.17	27	148	0.182
P1-A7-D	16	136	0.117	10	96	0.104	9	140	0.064
P1-A8-D	23	146	0.157	21	150	0.14	22	163	0.134
Sub-Total-2	-	-	1.204	-	-	1.081	-	-	1.085
Total	-	-	2.398	-	-	2.028	-	-	2.102
Average	-	-	0.150	-	-	0.127	-	-	0.131

Inspection Team C

Requirements Analysis Phase (Using Process 1 and MS Excel)

Artefact ID	Inspector Code								
	IC1			IC2			IC3		
	TNCD	TIT	Prod-C1	TNCD	TIT	Prod-C2	TNCD	TIT	Prod-C3
P2-A1-R	22	147	0.149	23	133	0.172	24	155	0.154
P2-A2-R	27	151	0.178	26	143	0.181	28	160	0.175
P2-A3-R	13	126	0.103	11	123	0.089	13	147	0.088
P2-A4-R	11	155	0.07	8	129	0.062	8	163	0.049
P2-A5-R	33	155	0.212	30	136	0.22	31	169	0.183
P2-A6-R	21	152	0.138	23	140	0.164	19	157	0.121
P2-A7-R	12	143	0.083	13	131	0.099	16	152	0.105
P2-A8-R	27	148	0.182	25	142	0.176	25	172	0.145

Sub-Total-1	-	-	1.115	-	-	1.163	-	-	1.02
Design Phase (Using Process 1 and MS Excel)									
Artefact ID	Inspector Code								
	IC1			IC2			IC3		
	TNCD	TIT	Prod-C1	TNCD	TIT	Prod-C2	TNCD	TIT	Prod-C3
P2-A1-D	17	140	0.121	11	138	0.079	10	146	0.068
P2-A2-D	18	142	0.126	21	138	0.152	19	156	0.121
P2-A3-D	13	137	0.094	17	131	0.129	18	144	0.125
P2-A4-D	32	133	0.24	31	137	0.226	39	165	0.236
P2-A5-D	25	150	0.166	29	147	0.197	31	158	0.196
P2-A6-D	17	140	0.121	20	127	0.157	15	143	0.104
P2-A7-D	23	138	0.166	22	125	0.176	27	153	0.176
P2-A8-D	30	144	0.208	27	135	0.2	28	153	0.183
Sub-Total-2	-	-	1.242	-	-	1.316	-	-	1.209
Total	-	-	2.357	-	-	2.479	-	-	2.229
Average	-	-	0.147	-	-	0.155	-	-	0.139
Inspection Team D									
Requirements Analysis Phase (Using Process 2 and ArSeC)									
Artefact ID	Inspector Code								
	ID1			ID2			ID3		
	TNCD	TIT	Prod-D1	TNCD	TIT	Prod-D2	TNCD	TIT	Prod-D3
P2-A1-R	22	132	0.166	23	141	0.163	24	148	0.162
P2-A2-R	27	130	0.207	26	141	0.184	28	150	0.186
P2-A3-R	13	132	0.098	11	138	0.079	13	152	0.085
P2-A4-R	11	133	0.082	8	137	0.058	8	139	0.057
P2-A5-R	33	135	0.244	30	143	0.209	31	144	0.215
P2-A6-R	21	133	0.157	23	139	0.165	19	141	0.134
P2-A7-R	12	136	0.088	13	142	0.091	16	144	0.111
P2-A8-R	27	141	0.191	25	147	0.17	25	153	0.163
Sub-Total-1	-	-	1.233	-	-	1.119	-	-	1.113
Design Phase (Using Process 2 and ArSeC)									
Artefact ID	Inspector Code								
	ID1			ID2			ID3		
	TNCD	TIT	Prod-D1	TNCD	TIT	Prod-D2	TNCD	TIT	Prod-D3
P2-A1-D	17	101	0.168	11	118	0.093	10	131	0.076
P2-A2-D	18	99	0.181	21	131	0.16	19	150	0.126
P2-A3-D	13	132	0.098	17	133	0.127	18	137	0.131
P2-A4-D	32	136	0.235	31	143	0.216	39	165	0.236
P2-A5-D	25	127	0.196	29	115	0.252	31	149	0.208
P2-A6-D	17	99	0.171	20	134	0.149	15	138	0.108
P2-A7-D	23	129	0.178	22	144	0.152	27	145	0.186
P2-A8-D	30	105	0.285	27	139	0.194	28	134	0.208
Sub-Total-2	-	-	1.512	-	-	1.343	-	-	1.279
Total	-	-	2.745	-	-	2.462	-	-	2.392
Average	-	-	0.172	-	-	0.154	-	-	0.150

As shown in Table 5.21 and Figure 5.5, it is obvious that among the 12 inspectors, ID1 from Inspection Team D achieved the highest productivity (0.172), and inspector IA3 recorded the lowest productivity. The inspector with the highest productivity from each inspection team is IA1, IB1, IC2 and ID1, respectively. All these inspectors are highly skilled inspectors with at least 20 or more years of inspection experience, and had inspected more than 1,600 artefacts. The inspectors with the lowest productivity from each inspection team are IA3, IB2, IC3 and ID3, respectively. All these inspectors are skilled inspectors with less than 15 years of inspection experience and had inspected less than 1,000 artefacts except for inspector IB2 who is highly skilled, possesses 25 years of inspection experience and had inspected 1,070 artefacts. There are two highly skilled inspectors (IA2 and IB2) whose average productivity is lower than the average productivity of the skilled inspector IB3. The average productivity of IC3 and ID3 was not compared as their respective average productivity was calculated based on Project 2 and not Project 1 (i.e. incorrect comparison if it is not based on the same project).

Table 5.21: Average productivity of each inspector

Inspection Team	Inspector Code	Average Productivity (No. of real defects detected / min)	Skill Level	NYIE	NAI
A	**IA1**	**0.134**	**HS**	**21**	**1800**
	IA2	0.129	HS	26	1500
	IA3	0.120	S	12	720
B	**IB1**	**0.150**	**HS**	**20**	**1760**
	IB2	0.127	HS	25	1070
	IB3	0.131	S	10	950
C	IC1	0.147	HS	22	930
	IC2	**0.155**	**HS**	**32**	**1620**
	IC3	0.139	S	11	780
D	**ID1**	**0.172**	**HS**	**34**	**1940**
	ID2	0.154	HS	20	1380
	ID3	0.150	S	14	650

Key: NYIE – No. of Years of Inspection Experience NAI – No. of Artefacts Inspected

These outcomes show that the number of artefacts inspected might affect the individual productivity of the inspectors. Generally, the more artefacts inspected, the higher the productivity of the inspectors. On the other hand, as each inspector from the four inspection teams inspected 16 artefacts only, it is not appropriate to make any inferences and comparison of their productivity based on the inspection process and the tools they used.

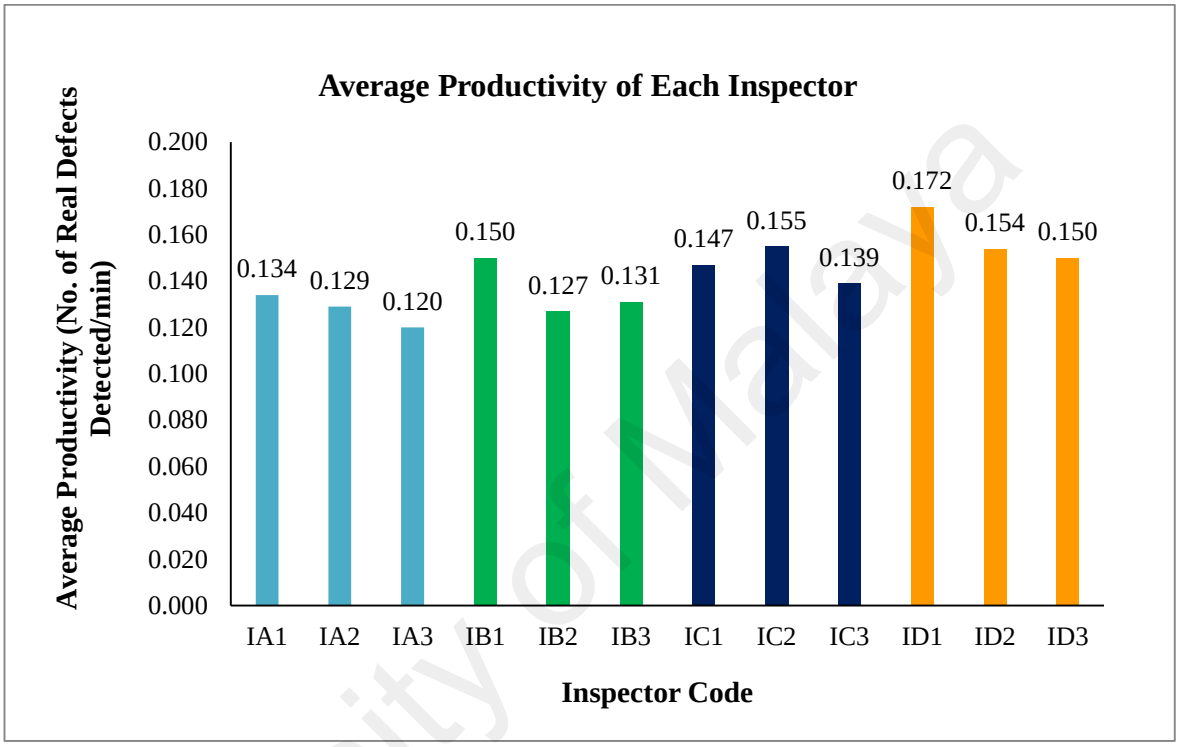


Figure 5.5: Average productivity of each inspector

5.6.4 Most Common Defects Detected

In the two case studies, different classes of defects were detected. This prompted further investigation to determine the classes of defects which were more frequently detected by the inspection teams using the two different inspection processes in compared with the number of defects detected in these classes.

5.6.4.1 Defects detected in the requirements analysis phase (Project 1 and Project 2)

Actual figures on classes of defects detected in the 32 artefacts of P1 (inspected by Team A and Team B) and P2 (inspected by Team C and Team D), and the actual total

number of defects detected during the requirements analysis phase are tabulated in Table 5.22 and Table 5.23, respectively.

Table 5.22: Distribution of defects detected by defect classes (Requirements analysis phase)

Project 1 (P1) (Inspected by Team A and Team B)														
Artefact ID	**Defect Classes**													
	Func		Perf		Envi		Inte		Secu		Misc		Total	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
P1-A1-R	13	17	4	4	2	5	5	6	1	1	1	1	26	34
P1-A2-R	8	9	1	6	3	5	1	2	2	3	0	1	15	26
P1-A3-R	8	12	4	7	1	3	3	3	2	3	0	0	18	28
P1-A4-R	4	8	3	7	3	4	4	6	1	1	0	0	15	26
P1-A5-R	3	4	3	3	3	4	2	4	0	0	1	2	12	17
P1-A6-R	8	8	3	4	2	3	1	1	6	6	1	1	21	23
P1-A7-R	2	6	2	2	8	15	1	3	0	0	0	0	13	26
P1-A8-R	3	5	7	12	5	5	3	3	2	2	1	1	21	28
Total	49	69	27	45	27	44	20	28	14	16	4	6	141	208
Project 2 (P2) (Inspected by Team C and Team D)														
Artefact ID	**Defect Classes**													
	Func		Perf		Envi		Inte		Secu		Misc		Total	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
P2-A1-R	5	9	3	6	7	9	3	3	4	6	2	2	24	35
P2-A2-R	6	8	3	5	7	8	5	8	1	4	0	0	22	33
P2-A3-R	2	2	7	8	3	4	3	4	1	1	0	0	16	19
P2-A4-R	5	9	0	0	0	0	1	2	1	1	0	1	7	13
P2-A5-R	9	12	1	3	3	3	8	8	5	5	4	5	30	36
P2-A6-R	2	2	7	8	3	5	1	7	3	3	2	2	18	27
P2-A7-R	2	2	6	7	3	6	3	6	0	0	0	0	14	21
P2-A8-R	4	4	5	5	6	10	7	10	7	7	2	3	31	39
Total	35	48	32	42	32	45	31	48	22	27	10	13	162	223

Keys:

Func – Functionality Perf – Performance Envi – Environment
Inte – Interface Secu – Security Misc – Miscellaneous
M1 (Process 1) – Formal inspection process and MS Excel
M2 (Process 2) – ISIP and ArSeC

Table 5.23: Actual total number of defects detected by defect classes (Requirements analysis phase)

Actual No. of Defects Detected: Project 1 (P1)							
Artefact ID	**Defect Classes**						
	Func	Perf	Envi	Inte	Secu	Misc	Total
P1-A1-R	17	5	5	6	1	2	36
P1-A2-R	9	6	5	2	3	1	26
P1-A3-R	12	7	3	4	3	0	29
P1-A4-R	8	7	4	7	1	0	27
P1-A5-R	4	3	4	4	0	2	17
P1-A6-R	8	4	3	1	6	1	23
P1-A7-R	6	2	16	3	0	0	27
P1-A8-R	5	12	5	3	2	1	28
Total	69	46	45	30	16	7	213
Actual No. of Defects Detected: Project 2 (P2)							
Artefact ID	**Defect Classes**						
	Func	Perf	Envi	Inte	Secu	Misc	Total
P2-A1-R	9	6	9	3	7	2	36
P2-A2-R	8	6	9	8	4	0	35
P2-A3-R	2	8	4	4	1	0	19
P2-A4-R	9	0	0	2	1	1	13
P2-A5-R	13	3	3	8	6	6	39
P2-A6-R	2	8	5	7	3	2	27
P2-A7-R	2	9	7	6	0	0	24
P2-A8-R	5	5	10	10	8	3	41
Total	50	45	47	48	30	14	234

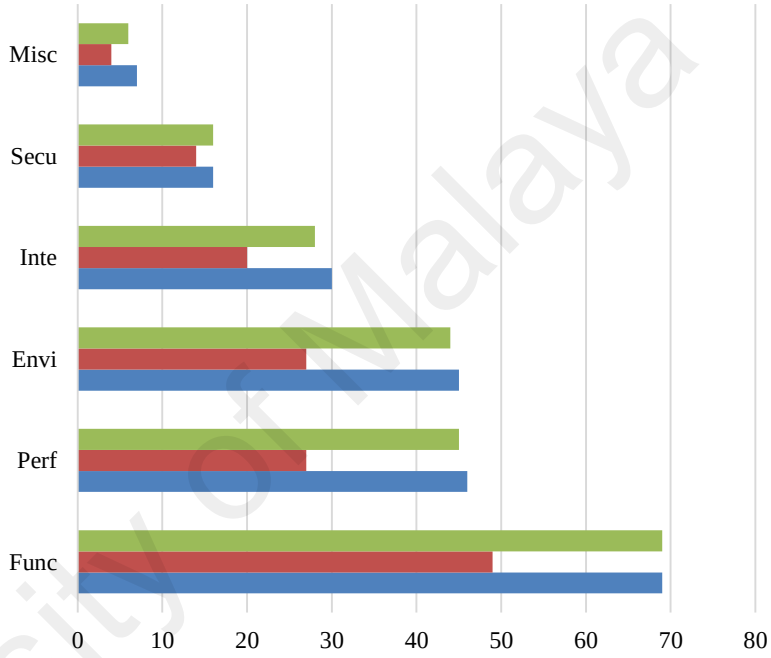
Keys:

Func – Functionality Perf – Performance Envi – Environment
Inte – Interface Secu – Security Misc – Miscellaneous

Figure 5.6 shows that of the 213 defects (Table 5.23), the two most commonly detected defects in P1 pertain to functionality, 32.4% (69 / 213 x 100%), and performance, 21.6% (45 / 213 x 100%), respectively. For these two classes of defects, inspection Team B using Process 2 detected 100% (functionality: 69 / 69 x 100%), and 97.8% (performance: 45 / 46 x 100%) of the defects, respectively. On the other hand, Team A which used Process 1, detected 71% (functionality: 49 / 69 x 100%), and 58.7% (performance: 27 / 46 x

100%) of the defects of these two classes, respectively. It is clear that the inspectors who used ISIP succeeded in detecting more functionality-related defects, as well as defects in the other classes, than the inspectors who used the formal inspection process. Team B also detected 100% of the defects pertaining to security issues.

Comparison of Defects Detected According to Defect Class (Project 1 Inspected by Team A and Team B)



	Func	Perf	Envi	Inte	Secu	Misc
No. of Defects Detected (using Process 2)	69	45	44	28	16	6
No. of Defects Detected (using Process 1)	49	27	27	20	14	4
Actual No. of Defects	69	46	45	30	16	7

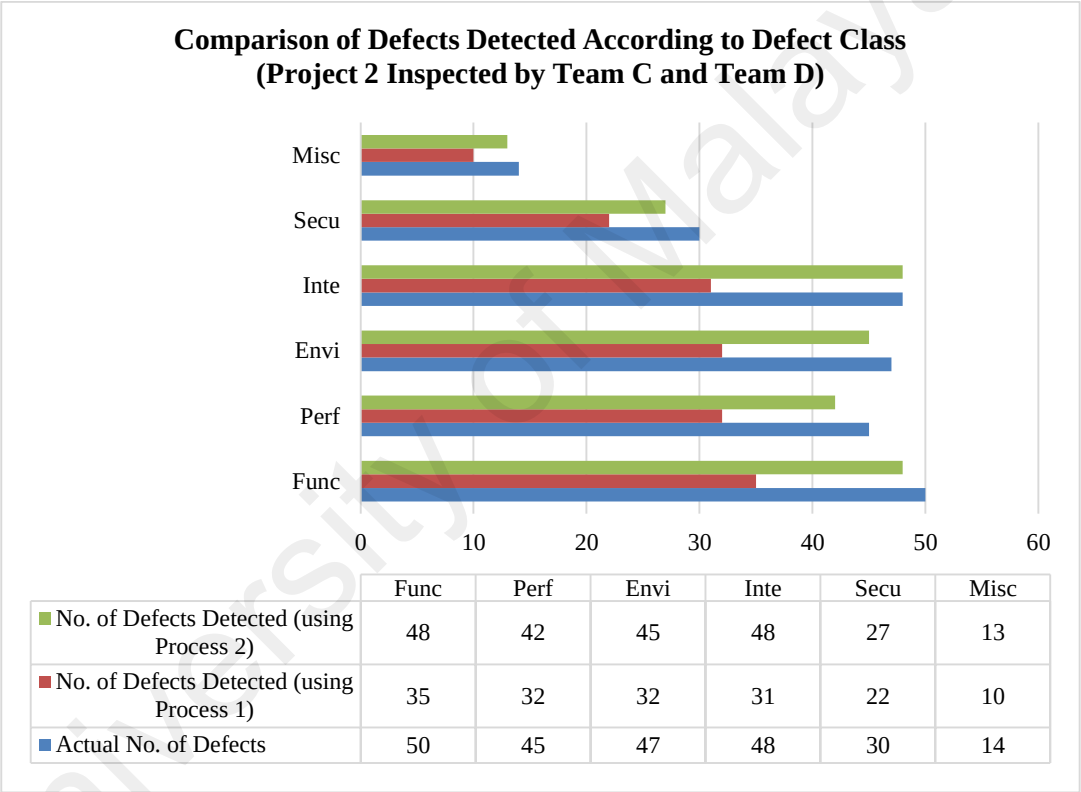
Keys:

Func – Functionality
Perf – Performance
Envi – Environment
Inte – Interface
Secu – Security
Misc – Miscellaneous

Figure 5.6: Comparison of defects detected in Project 1 according to defect class (inspected by Team A and Team B) with the actual total number of defects in each defect class (Requirements analysis phase)

As shown in Figure 5.7, of the 234 defects (Table 5.23), the two defects most commonly detected in P2 pertain to functionality, 21.4% (50 / 234 x 100%), and interface, 20.5% (48 / 234 x 100%), respectively. From these two classes of defects, inspection Team D using Process 2 detected 96.0% (functionality: 48 / 50 x 100%), and 100.0%

(interface: 48 / 48 x 100%) of the defects, respectively. However, Team C, which used Process 1, detected defects pertaining to functionality, 70.0% (35 / 50 x 100%), and pertaining to interface, 54.6% (31 / 48 x 100%) of the defects detected from these two classes, respectively. Again, it is clear that inspectors who used ISIP succeeded in detecting more functionality-related defects, as well as defects in other classes, than inspectors who used the formal inspection process. Team D also detected 100% of the defects pertaining to interface issues.



	Func	Perf	Envi	Inte	Secu	Misc
No. of Defects Detected (using Process 2)	48	42	45	48	27	13
No. of Defects Detected (using Process 1)	35	32	32	31	22	10
Actual No. of Defects	50	45	47	48	30	14

Keys:

Func – Functionality Perf – Performance Envi – Environment
Inte – Interface Secu – Security Misc – Miscellaneous

Figure 5.7: Comparison of defects detected in Project 2 according to defect class (inspected by Team C and Team D) with the total number of defects in each defect class (Requirements analysis phase)

5.6.4.2 Defects detected in the design phase (Project 1 and Project 2)

Actual figures on classes of defects detected in the 32 artefacts of P1 (inspected by Team A and Team B), and P2 (inspected by Team C and Team D), and the actual total

number of defects detected during the design phase, are tabulated in Table 5.24 and Table 5.25, respectively.

Table 5.24: Distribution of defects detected by defect classes (Design phase)

Project 1 (P1) (Inspected by Team A and Team B)																		
Artefact ID	**Defect Classes**																	
	Algo		Assi		Chec		Docu		Func		Inte		Pack		Misc		Total	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
P1-A1-D	5	5	3	3	7	7	7	7	3	3	2	2	5	4	1	1	33	32
P1-A2-D	1	1	4	5	2	9	5	6	7	10	3	4	2	1	0	0	24	36
P1-A3-D	5	5	5	6	2	2	7	8	0	1	5	9	1	1	2	3	27	35
P1-A4-D	1	2	1	1	2	3	1	2	2	2	0	0	1	1	0	0	8	11
P1-A5-D	3	6	2	3	2	3	1	1	3	4	3	4	1	3	1	1	16	25
P1-A6-D	6	9	2	4	5	6	1	3	3	4	2	2	2	3	0	0	21	31
P1-A7-D	3	8	1	2	2	2	5	5	1	3	1	1	0	0	0	0	13	21
P1-A8-D	3	5	4	6	3	4	5	6	2	4	3	5	1	3	1	1	22	34
Total	27	41	22	30	25	36	32	38	21	31	19	27	13	16	5	6	164	225
Project 2 (P2) (Inspected by Team C and Team D)																		
Artefact ID	**Defect Classes**																	
	Algo		Assi		Chec		Docm		Func		Inte		Pack		Misc		Total	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
P2-A1-D	3	3	3	3	6	6	5	5	5	3	2	2	0	0	0	0	24	22
P2-A2-D	3	3	2	2	4	7	4	6	3	3	4	5	2	2	0	0	22	28
P2-A3-D	1	2	2	3	3	3	4	5	1	4	0	1	2	7	1	1	14	26
P2-A4-D	6	7	5	5	4	3	3	7	5	9	3	7	1	3	0	1	27	42
P2-A5-D	2	2	5	8	4	6	2	4	4	8	1	4	3	3	3	2	24	37
P2-A6-D	0	0	4	4	2	2	3	6	3	4	0	5	2	3	0	0	14	24
P2-A7-D	2	2	4	5	2	4	1	5	3	5	0	4	2	3	0	0	14	28
P2-A8-D	3	5	4	8	3	3	3	4	5	5	5	7	3	4	1	1	27	37
Total	20	24	29	38	28	34	25	42	29	41	15	35	15	25	5	5	166	244

Keys:

Algo – Algorithm Assi – Assignment Chec – Checking
Docu – Documentation Func – Function Inte – Interface
Pack – Package Misc – Miscellaneous
M1 (Process 1) – Formal inspection process and MS Excel
M2 (Process 2) – ISIP and ArSeC

Table 5.25: Actual total number of defects detected by defect classes (Design phase)

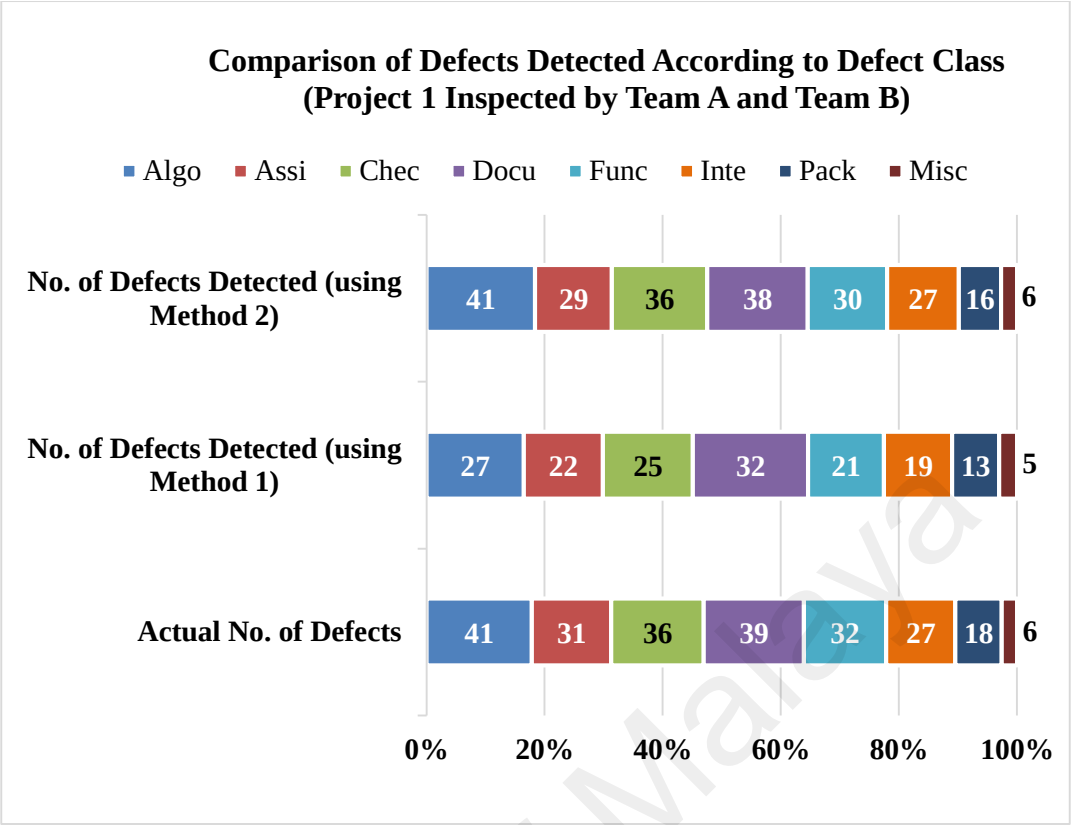
Actual No. of Defects Detected: Project 1 (P1)									
Artefact ID	**Defect Classes**								
	Algo	Assi	Chec	Docu	Func	Inte	Pack	Misc	Total
P1-A1-D	5	3	7	8	3	2	5	1	34
P1-A2-D	1	5	9	6	10	4	2	0	37
P1-A3-D	5	6	2	8	1	9	1	3	35

P1-A4-D	2	1	3	2	2	0	1	0	11
P1-A5-D	6	3	3	1	4	4	3	1	25
P1-A6-D	9	4	6	3	5	2	3	0	32
P1-A7-D	8	3	2	5	3	1	0	0	22
P1-A8-D	5	6	4	6	4	5	3	1	34
Total	41	31	36	39	32	27	18	6	230
Actual No. of Defects Detected: Project 2 (P2)									
Artefact ID	**Defect Classes**								
	Algo	Assi	Chec	Docu	Func	Inte	Pac	Misc	Total
P2-A1-D	3	3	6	5	5	2	0	0	24
P2-A2-D	3	2	7	6	3	5	2	0	28
P2-A3-D	2	3	3	6	4	1	7	1	27
P2-A4-D	7	8	5	7	9	7	3	1	47
P2-A5-D	2	8	6	4	9	4	4	3	40
P2-A6-D	0	5	2	6	4	5	3	0	25
P2-A7-D	2	5	4	5	5	4	3	0	28
P2-A8-D	5	8	3	4	5	8	4	1	38
Total	24	42	36	43	44	36	26	6	257

Keys:

Algo – Algorithm
Assi – Assignment
Chec – Checking
Docu – Documentation
Func – Function
Inte – Interface
Pack – Package
Misc – Miscellaneous

Figure 5.8 shows that of the 230 defects (Table 5.26), the two most commonly detected defects pertain to algorithm, 17.8% (41 / 230 x 100%), and documentation, 17.0% (39 / 230 x 100%), respectively. From these two classes of defects, inspection Team B using Process 2 detected defects that portion to algorithm, 100% (41 / 41 x 100%), and documentation, 97.4%: (38 / 39 x 100%), respectively. On the other hand, Team A which used Process 1, detected defects that pertain to algorithm 65.9% (27 / 41 x 100%) and documentation 82.1% (32 / 39 x 100%), in these two classes, respectively. It is clear that the inspectors who used ISIP succeeded in detecting more algorithm-related defects as well as defects in the other classes, than the inspectors who used the formal inspection process. Team B also detected 100% of the defects pertaining to Interface, as well as those pertaining to miscellaneous issues.

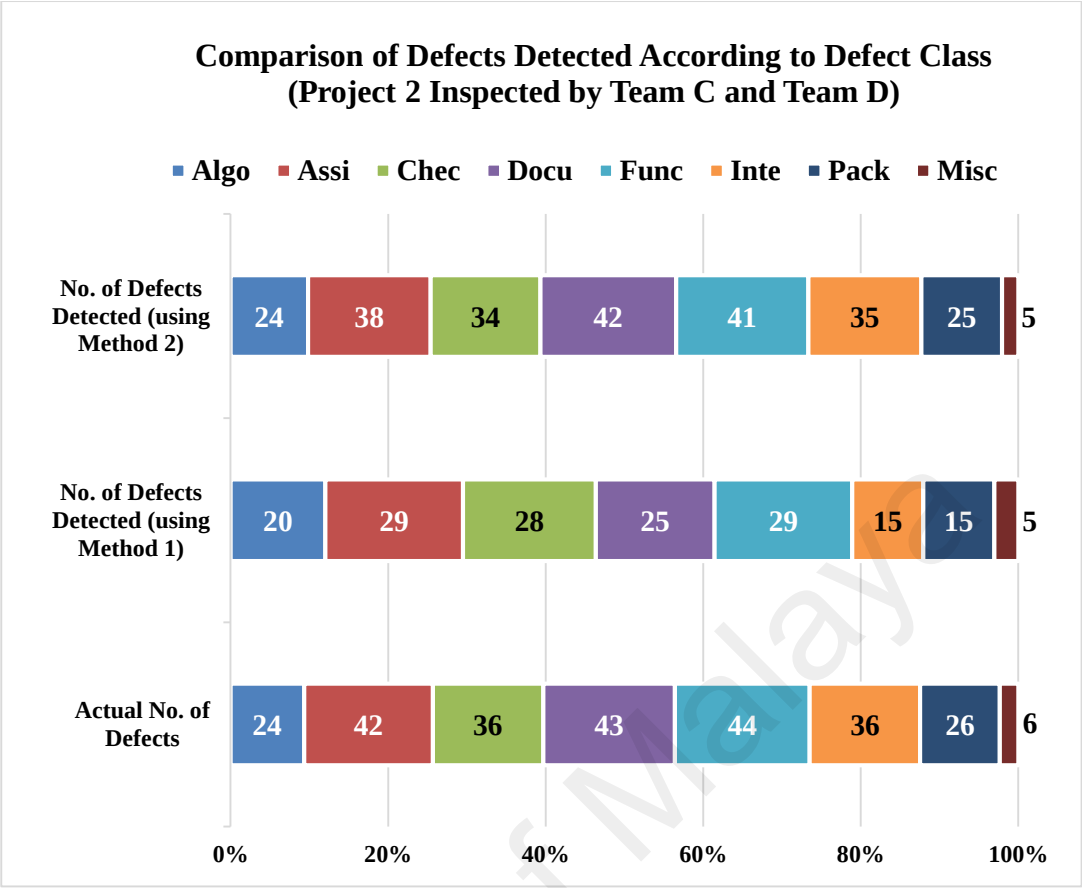


Keys:

Algo – Algorithm	Assi – Assignment	Chec – Checking
Docu – Documentation	Func – Function	Inte – Interface
Pack – Package	Misc – Miscellaneous	

Figure 5.8: Comparison of defects detected in Project 1 according to defect class (inspected by Team A and Team B) with the actual total number of defects in each defect class (Design phase)

As shown in Figure 5.9, of the 257 defects (Table 5.26), the two most commonly detected defects in P2 pertain to functionality, 17.1% (44 / 257 x 100%), and documentation, 16.7% (43 / 257 x 100%), respectively. From these two classes of defects, inspection Team D using Process 2 detected defects that pertain to functionality, 93.2% (41 / 44 x 100%), and interface, 97.7% (42 / 43 x 100%), respectively.



Keys:

Algo – Algorithm
Assi – Assignment
Chec – Checking
Docu – Documentation
Func – Function
Inte – Interface
Pack – Package
Misc – Miscellaneous

Figure 5.9: Comparison of defects detected in Project 2 according to defect class (inspected by Team C and Team D) with the actual total number of defects in each defect class (Design phase)

However, Team C which used Process 1, detected defects pertaining to functionality, 65.9% (29 / 44 x 100%) and defects pertaining to documentation, 58.1% (25 / 43 x 100%) in these two classes, respectively. Again, it is clear that the inspectors who used ISIP succeeded in detecting more functionality-related defects, as well as defects in other classes, than the inspectors who used the formal inspection process, in the design phase. Team D detected 100% of the defects pertaining to algorithm.

5.6.5 Lessons learned from the two case studies

In order to evaluate Improved Software Inspection Process (ISIP), two case studies were conducted. The lessons learned from these two case studies are summarised as follows:

i. arranging the inspection session time is very difficult and even with prior agreements some inspectors may not be attended on time. So, sending the opinion or vote after the session is an appropriate solution;

ii. guidelines for using the system with mandatory requirement to use ArSeC (like secure Internet connection) have to b included in preparation kit.;

iii. the constraints of virtual session for inspection session especially the time limitation for each inspector has to be define adequately; and

iv. complete familiarity with the web based tool (ArSeC) and answering the possible questions prior to start the inspection process is mandatory.

5.7 Summary

This chapter presents the evaluation of ISIP using the data collected from two case studies on two software development projects. Statistical tests were used to prove the three hypotheses formulated in chapter 3. Before testing the hypotheses, data screening techniques that include histogram, test of skewness, and Kolmogorov-Smirnov test of normality, were employed to determine whether the data of the dependent variables are distributed normally. Also, boxplot was used to determine and identify any outliers in the data set. Using Paired-Samples T Test, the total number of real defects detected using Process 2 is 8.344 (mean) more than the total number of real defects detected using Process 1, indicating a 42.2% increase in the average total number of real defects detected using Process 2. As the distribution of the matched paired data of the total inspection time

using Process 2 is not normally distributed, the Related-Samples Wilcoxon Signed Rank test was used to test hypothesis 2. The test result shows that the average total inspection time taken when using Process 2 is 7.31 minutes less than the average total inspection time taken when using Process 1. This is a 3.3% reduction in the average total inspection time taken to inspect the artefacts. Hypothesis 3 was tested using Paired-Samples T Test. The test result shows that the productivity of the inspection teams using Process 2 is higher by 0.04375 than the productivity of the inspection teams using Process 1, indicating a 49.1% increase in the average productivity of the inspection teams using Process 2.

Other findings from the case studies include i) a strong correlation between the total number of defects present in an artefact, and the size of the artefact, as shown by the Pearson Correlation coefficient of 0.756; ii) the efficiency of ISIP and ArSeC to improve the quality of software inspection can be determined by calculating and comparing the total number of real defects detected by each inspection process in relation to the total number of real defects present in each artefact. The result shows that using ISIP and ArSeC improves the quality of inspection by 28.8% when compared with using the formal inspection process and MS Excel, based on the total number of real defects detected in the artefacts; iii) among the 12 inspectors from inspection teams A-D, the most productive inspector is ID1 who achieved an average productivity of 0.172 real defects detected per minute ; iv) the three most commonly detected defects in the requirements analysis phase of the two case studies are the functionality, performance and interface defect classes; v) the three most commonly detected defects in the design phase of the two case studies are the algorithm, documentation, and functionality defect classes.

Although a sample size of 32 sets of data is used in these case studies, the findings show that using the proposed inspection process, ISIP and its support tool, ArSeC,

software inspectors can improve their productivity by detecting more real defects, while taking less inspection time. The findings on the most commonly detected defects in the requirements analysis and design phases will be useful to software inspectors who should be more alert to these classes of defects when inspecting the artefacts of similar projects in future.

CHAPTER 6: DISCUSSION AND CONCLUSION

This chapter explains the research validity and reliability, the different ways to minimise bias as well as ethical issues involved in the research. The problems encountered and limitations of the research are also discussed. The contributions of the research are highlighted and suggestions for future investigations based on the problems and limitations encountered are also presented.

6.1 Research Validity and Reliability

In any research, it is important to check the validity and reliability of the data gathered, and the measurement method used, and eventual findings. The concept of validity answers the question, "Does your measurement process and assessment, actually measures what you intend to measure?" The concept of reliability is an indication of whether repeated measurements or assessments provide a consistent output under the same initial circumstances (Cooper, Hedges, & Valentine, 2009).

A pilot case study was conducted on one medium-sized artefact by two inspection teams (three inspectors per team) from Company 1 (C1) prior to the two case studies. Problems encountered and weaknesses found during the pilot study were identified and appropriate corrective measures were taken to ensure that the actual case studies will be carried out in a systematic manner to ensure research validity and reliability.

In this research, four types of validity (statistical validity, construct validity, external validity, and internal validity) have been considered and described in detail in section 3.3 (chapter 3). The inspectors who participated in the case studies had not previously inspected the artefacts used in the case studies. Furthermore, the two case studies involved four different and independent inspection teams consisting of inspectors with the same skill levels – each team consisting of two highly-skilled inspectors and one skilled

inspector, respectively. The time for inspection preparation and the inspection session is limited to two hours. Hence, the two case studies were conducted by the four inspection teams using the same procedures, same tools, and under the same conditions using both the formal inspection process (by Team A and Team C) and ISIP (by Team B and Team D), respectively. Also, in the test of hypothesis, data screening, histograms, skewness test, Kolmogorov-Smirnov test and box-plot were used to determine the normality of data distribution of the dependent variables, and to identify any possible departure from normality. The Paired-Samples T Test was applied, as it is the most appropriate test to conduct a statistical test that involves two matched pairs of dependent variables (Paired-Samples T Test, 2015).

In the software inspection process, the inspection metrics considered include the total number of defects detected, the total number of false positives, and the total inspection time of each inspection process. The data were recorded using MS Excel (for those using process 1) and ArSeC (for those using process 2), and based on an inspection data recording form specially designed for the case studies to ensure that inspection data were recorded systematically and correctly. The total number of real (actual) defects identified was compared with historical records from past inspections to determine whether ISIP has effected improvement on the quality of the software inspection process. In both case studies, the ratio scale is used in the measurement of the three dependent variables – the total number of defects detected, the total inspection time, and the productivity of each inspection team. The inspection data recording form and the use of MS Excel and ArSeC with process 1 and process 2, respectively, have contributed to the accuracy of the measurements.

6.2 Bias in the Research

Besides validity and reliability, this study has also addressed the issue of biasness (Graziano & Raulin, 2014). Randomisation was used in the selection of projects (P1 and P2) from the two companies (C1 and C2); the selection of the 16 artefacts of P1 and P2 of the requirements analysis and design phases; and the selection of the 12 inspectors based on their inspection work experience and inspection competency. The random assignment of artefacts to the randomly selected inspectors, helps to avoid bias and confirm the validity of the inspection results. Also, the two case studies were carried out using the same experimental design and procedures, and under similar conditions, instead of repeating the inspections based on ISIP on the same sets of artefacts by the same team of inspectors. This replication avoids bias during the inspection of defects, and makes it possible to estimate the mean effect of the inspection process (i.e. ISIP) evaluated in this research.

Furthermore, a complete balanced block was used in assigning the artefacts and inspectors in the two case studies, as shown in Figure 6.1, below. In each case study, eight artefacts from the requirements analysis and design phases, respectively, from each company, were selected at random for the inspection process. Each inspection team has three inspectors comprising two highly skilled inspectors and one skilled inspector. They were selected at random from the two groups of highly skilled and skilled inspectors of C1 and C2. The assignment of an equal number of artefacts and an equal number of inspectors of the same skill level for each case study will eliminate the confounding effect which would have resulted if different number of artefacts had been assigned. In addition, having inspectors of different skill levels will also result in having different number of real defects being detected by the inspectors with different inspection capabilities. The pilot case study helped in reducing the potential impact of limitations associated with his

the research study by using the feedback of the inspectors to improve the inspection kit and enhanced the virtual inspection session.

Table 6.1: Complete balanced block design in assigning artefacts and inspectors to case study

Artefacts	**Case Study 1**		**Case Study 2**	
	S1	**S2**	**S1**	**S2**
Requirements Analysis Phase	8	8	8	8
Design Phase	8	8	8	8
Total No. of Artefacts	16	16	16	16
Skill Level	**Inspectors**			
	Team A	**Team B**	**Team C**	**Team D**
Highly Skilled	2	2	2	2
Skilled	1	1	1	1
Total No. of Inspectors	3	3	3	3

Keys: S1 (Process 1) – Formal Inspection Process S2 (Process 2) – ISIP

Furthermore, the inspectors were informed that there will be no evaluation on their individual productivity. This was done to avoid or pre-empt possible "dishonest" behaviours of the inspectors in the case study. An inspector who is aware that his/her productivity will be evaluated to determine the inspector with the highest or lowest productivity, will try to perform well in the inspection process by putting in extra efforts/time to detect more defects than what he/she would normally have detected. This will affect the outcome of the inspection data analysis, and thus, the validity of this research.

6.3 Ethical Issues

The 12 inspectors (human) were selected by their respective project manager, and they participated in the case study voluntarily, therefore, appropriate measures were taken to ensure their rights of privacy and confidentiality. The profiles of the inspectors, profiles of the companies, project managers, etc., involved in the two case studies were not revealed throughout this research to any other third party. Besides, all the inspectors were

not aware that their individual inspection productivity was being measured (to ensure research validity) and reported in this thesis, and that the findings are not reported to their respective project manager. This is to ensure that their personal interests are protected, and their career path in the company will not be affected in any way.

6.4 Problems Encountered

Throughout the research, a few problems were encountered. These include difficulty in getting the cooperation from the software companies to participate in the case studies, the number of artefacts used, and the number of inspectors permitted to participate in the case studies. These problems are discussed in detail, below.

6.4.1 Cooperation from software companies

To test the three hypotheses formulated in this research, more than 12 software development companies in Malaysia were initially invited to participate in the case studies. Sadly, all the companies were reluctant to participate as their software projects are commercial systems, which means that they the contents of the systems cannot be disclosed to any external parties. Furthermore, they have tight project deadlines to meet, and hence, unable to commit any inspection teams to participate in the research, even though the software development projects might even be in-house software development projects. However, through the assistance from friends, the project managers from a manufacturing company and a trading company agreed to participate in the case studies. The companies were concerned about their reputation and image, how their inspectors will benefit, and the quality of their artefacts to be inspected. To reassure them, a non-disclosure agreement was signed to ensure that all privacy and confidentiality requirements are complied with as well as address their other concerns. The inspection data collected and used for analysis were returned to the company upon completion of the research.

6.4.2 Number of artefacts used in the case study

The number of defects detected in an artefact is a reflection of the quality and the competency of the author of the artefact. Hence, the project managers of the two participating companies were very concerned that the quality of those artefacts used in the case studies will be 'opening known' to an external party (i.e. the researcher of this study). The companies feared that the researcher might have a negative impression of the competency of the authors who produced those artefacts as well as the quality of the software systems, which may contain many defects in the artefacts. Hence, the project manager of each company only allowed eight artefacts of the requirements analysis and design phases, selected at random, to be used in the case study, respectively.

A statistical test often requires 30 or more sets of data for the test of hypothesis, thus, another case study should be conducted to collect more data to test the three hypotheses formulated for this research. Owing to time constraints, only two case studies were carried out to collect sufficient inspection data needed for the statistical tests. Hence, in this research, only the defects from the 32 artefacts were used in data analysis and to prove the hypotheses (i.e. defects detected from 16 artefacts of the requirements analysis, and design phases, respectively, of the two different software projects).

6.4.3 Number of inspectors involved in the case study

Most of the inspectors were also involved as software engineers in the development of other software systems for in-house applications. After negotiating with the project managers of the two participating companies, only six inspectors - selected at random and based on their inspection skill level and experience - were allowed to participate in the case studies. However, most of the selected inspectors expressed concern that their productivity will be measured and compared with other inspectors in the inspection team, and thus, were reluctant to participate in the case study. It was only after they were assured

that their work performance will not be affected by their participation in the case study, and that their productivity will not be measured, then all the 12 inspectors agreed to participate. Hence, in the two case studies, only two inspection teams - with three inspectors in each team comprising two highly skilled inspectors and one skilled inspector - were formed to inspect the artefacts. Owing to time constraints (the two project managers allowed not more than three months for the case studies), and the number of inspectors allowed to participate in the case studies, other team structures such as teams with three skilled inspectors, two skilled inspectors with one semi-skilled inspector, etc., were not formed to conduct inspections on the 16 artefacts, to determine the impact of inspection team structure on software inspection.

6.5 Contributions of the Research Study

This research had proposed an enhanced formal inspection process comprising four stages which is aimed at improving the quality of software inspection by introducing enhancements to the inspection activity and other areas that had been identified for improvement. The enhancements that contributed to the increase in the number of real defects detected, decrease in the inspection time, and an increase of the productivity of the inspection team are described below.

6.5.1 The number of inspectors in the inspection team

In Fagan's formal inspection process (FIP), it is recommended that an inspection team should minimally consists of four to five members comprising one moderator, one author, one reader, one recorder (this responsibility can be assigned to the moderator) and one inspector (Laitenberger, 2002). In this case, only one inspector is involved. In ISIP, the inspection team should minimally have one moderator, one author, and a minimum of three inspectors up to a maximum of five inspectors. In the two case studies, the team has three inspectors. The average increase in the total number of real defects detected is 8.344

(42.2%) more than the total number of real defects detected using FIP. Hence, it is obvious that having more inspectors can help to detect more real defects in an artefact.

6.5.2 The roles and responsibilities of the inspection team

As mentioned in i) above, in FIP, an inspection team may have to perform four to five roles – moderator, author, reader, recorder (this responsibility can be assigned to the moderator) and inspector (Laitenberger, 2002). Each of them has specific responsibilities. However, in ISIP, there are only three roles – moderator, author, and inspector, in which the moderator also assume the role of a reader, as well as a recorder. This team composition optimises the usage of human resources.

6.5.3 Selection of inspectors based on inspector skill levels

The selection of inspectors is often based on their inspection experience and knowledge (Fagan, 1986). A drawback of this selection approach is that inspectors who are inexperienced will not be chosen to inspect artefacts although they could learn and benefit from their involvement in the inspection process (Laitenberger, 2002). In ISIP, the selection of inspectors is based on the skill levels - highly skilled, skilled, and semi-skilled - and the expertise of each inspector, which is evaluated using the three parameters – number of years of inspection, number of artefacts inspected, and average inspection productivity. In ISIP, it is recommended that the team structure should be composed of inspectors with different skill levels such as two highly skilled and one skilled inspectors (as in the two case studies), or two highly skilled and one semi-skilled inspectors, or any other combinations so that the skilled and semi-skilled inspectors also have the opportunity to learn from the highly skilled inspectors in the inspection process.

6.5.4 Inspection checklist

In the traditional formal inspection process (FIP), checklist and checklist-reading were used during the inspection process (Laitenberger, 2002). However, from the literature

review, it is found that there is no mention on the construction of the questions in the checklists, and also no mention of whether a database is created to maintain the checklist questions of past inspection projects. In ISIP, a Checklists database is created to maintain all the checklists used in past software inspection projects. This database can serve as a reference source for the moderator to create a fairly "comprehensive and complete" checklist easily and quickly based on the checklists of similar past software development projects. In addition, each question or statement in the checklists is specific and clearly phrased and devoid of any ambiguities, and this already facilitates decision making on whether an item that falls under a question/statement is truly a defect. Obviously, a fairly "comprehensive and complete" checklist would help in increasing the productivity of the inspection team as reflected by the detection of more defects and within shorter inspection time, when compared to a newly prepared inspection checklist.

6.5.5 Inspection process and inconsistencies resolution

Fagan's FIP supports the use of face-to-face group meeting as it promotes synergy leading to the detection of most of the defects, and resolution of the inconsistencies (Fagan, 1976; Fagan, 1986). On the other hand, ISIP allows inspectors who are not available during the scheduled inspection session to conduct inspection on the artefacts individually, and submit the inspection outcomes to the moderator for consolidation by the stipulated inspection timeline. Also, any ambiguity pertaining to the artefacts can be clarified with the author, and inconsistencies can be resolved easily using the proposed weighted voting process. Al these can be done online without the need to have all the inspection team members to be present at a fixed scheduled time. This feature eases the inspection process and resolution of the inconsistencies, and thus, helps to reduce the number of iterated inspection sessions, and reduce the total inspection time.

6.5.6 Potential causes of each defect

Providing solutions to the defects is not the focus of an inspection process. However, ISIP can ease the rework process of the author, by allowing inspectors to retrieve the potential causes of each real (actual) defect (if the potential causes are available in the Causes database), and distribute to the author for reference and resolution together with the inspection report. Logically, this enhancement will help in reducing the rework time even though it was not recorded and considered in the two case studies. There ahs been no report in the literature about this feature in the FIP.

6.5.7 Use of automated inspection tool (ArSeC) and inspection meeting

The use of ArSeC supports the inspection process by facilitating the distribution of the inspection documents to the inspection team members; recording the data and using the data in calculation of inspection data (project details, inspector details, number of defects detected by each inspector, inspection time, automatic calculation of the total number of real (actual) defects detected, total inspection time, etc.); facilitating the selection of inspectors; sending reminders to the related parties concerned; facilitate the storage and retrieval of data by the inspection team members such as defect data, checklists, defect causes, etc.; etc. As it is a Web-based inspection support tool, the inspection meetings can be held at anytime and at anywhere to accommodate inspection team members who might be at different geographical locations. This has helped to improve the productivity of the inspection team and reduce the total inspection time. There has been no report in the literature on the availability of an automated tool to support the FIP (Gimpel, 2014, Yogi, , Yatna, & Raharno, 2014).

Table 6.2 shows a summary of the comparison between the features of the formal inspection process and ISIP.

Table 6.2: Comparison between the formal inspection process and ISIP

Features	Formal Inspection Process	Enhanced Formal Inspection Process (ISIP)
Size of inspection team	4 or 5 members.	5 or 7 members.
Roles and responsibilities of inspection team: moderator, author, reader, recorder, and inspector	4 or 5 roles, with respective responsibilities.	3 roles, with respective responsibilities, but moderator also assumes the roles of reader and recorder.
Selection of inspection team members	Experience and knowledge.	Inspector skill level (highly skilled, skilled, or semi-skilled) and areas of expertise.
Inspection checklist	No checklist database.	Checklists database – allows storing and retrieving of inspection checklists from similar past projects; questions/statements in the checklist are specific and clearly phrased.
Inspection process and inconsistencies resolution	Face-to-face group inspection meeting.	Allows online group inspection meeting, and individual inspection and reporting.
Potential causes of each defect	No defect causes database to store potential causes for each defect.	A Causes database is available to store the potential causes for defects detected in the requirements analysis and design phases.
Use of automated inspection tool and inspection meeting	No automated tool to support FIP.	Availability of an automated tool (ArSeC) to support ISIP

6.6 Research Conclusion

This research focuses on improving the quality of software inspection process by making enhancements to the 7-step formal inspection process introduced by Michael Fagan. The enhanced inspection process (ISIP), has only four main inspection stages – inspection preparation, defect detection, pioneer kernel, and process appraisal. ISIP enhances the formal inspection process by introducing new features that are not found in

the latter process, as evident in the literature review. The enhancements made are confined to the inspection of the artefacts of the requirements analysis and design phases. To support the inspection process, an inspection tool, ArSeC, was developed to facilitate: the selection of inspectors; distribution of artefacts and related inspection documents to the inspection team members; online inspection process; storage of important inspection data (inspector profiles, checklists, defects, and causes of the defects); recording of inspection process, data and results; compilation of inspection outcomes; and distribution of inspection reports to all the parties concerned.

The results of the case studies show that the quality of software inspection is improved using ISIP (RQ1). The inspection time is reduced and the number of defects detected using the ISIP was more than the defects detected using the formal process, therefore the quality of software could be improved (RQ2). To answer the RQ3, the quality of the ISIP was evaluated based on three measurements: i) the total number of real defects detected in the artefacts, ii) the total inspection time to inspect an artefact, and iii) the productivity of the inspection teams. An increase in i) and iii), and a decrease in ii) reflects improvement in the quality of software inspection. In this research, two case studies were carried out, and the results from the inspection of 32 artefacts (16 requirements analysis and design artefacts, respectively) were used to prove the three hypotheses pertaining to the quality of software inspection, established in this study. The result of Paired-Samples T Test show that on an average, there is an increase in the total number of real defects detected (i.e. 8.344, or 42.2%) and an average increase of 0.04375 in the productivity of the inspection teams that used ISIP to conduct inspection process on the 16 artefacts assigned to them. Similarly, using the Related-samples Wilcoxon Signed Rank test, the average total inspection time has been reduced by 7.31, i.e. a reduction of 3.3%. As only 16 artefacts were used in the inspection process, the findings are only valid in the two case studies, and they cannot be used to generalise the efficiency of ISIP in improving the

quality of the software inspection process. More case studies have to be conducted to collect sufficient inspection data to test the three hypotheses, and to arrive at more cogent inferences. Maintaining the ArSeC databases incurs some overhead costs and hence, a limitation of using the proposed software inspection process. Despite this limitation, this research has achieved the four objectives defined in chapter 1 and contributed new insights to the area of software inspection by answering the three research questions.

6.7 Future Research

In this research, the problems encountered and the research limitations were discussed in section 6.4. The shortcomings pointed out should provide the motivation for further research to expand the scope of future studies to address the problems, as well as other issues as follows:

- Design inspection checklists for other development phases;
- Review the defect classification and enhance the defects list for the requirements analysis and design phases, and prioritise them according to severity level and frequency of occurrence;
- Analyse and classify the defects for other development phases;
- Conduct comparative studies on the efficiency of the 3-inspector and 5-inspector team structures, comprising inspectors with different skill levels and areas of expertise;
- Conduct more case studies to evaluate ISIP by involving software companies to participate in the study;
- Increase the number of artefacts, and use a variety of artefacts with different complexities levels to determine the quality of ISIP in software inspection process;

- Conduct a case study on one large software development project involving artefacts in all phases of the development lifecycle to determine the efficiency of ISIP in detecting defects with sufficient number of artefacts (i.e. more than 30 artefacts) to eliminate the confounding effects when artefacts from different software projects are used in data analysis and to prove the hypotheses established in this research.
- Investigate the productivity of other inspection team members such as the moderator and author;
- Investigate the productivity of authors in the rework process; and
- Providing three (or more) possible answers (such as Pass/Partial/ Fail options) to allow more flexible inspection process.

REFERENCES

Ackerman, A. F., Buchwald, L. S., & Lewsky, F. H. (1989). Software Inspections: An effective verification process. IEEE Software, 6(3): 31-36.

Alshazly, A. A., Elfatatry, A. M., & Abougabal, M. S. (2014). Detecting defects in software requirements specification. Alexandria Engineering Journal, 53(3), 513–527.

Anderson, P., Reps, T., & Teitelbaum, T. (2003). Design and implementation of a fine-grained software inspection tool. IEEE Transactions on Software Engineering, 29(8), 721-721-733. doi:10.1109/TSE.2003.

Anderson, P., Reps, T., Teitelbaum, T., & Zarins, M. (2003). Tool support for fine-grained software inspection. IEEE Software, 20(4), 42-42-50. doi:10.1109/MS.2003.

Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., & Merlo, E. (2002). Recovering traceability links between code and documentation. IEEE Transaction on Software Engineering, 28(10), 970–983.

Armour P. G. (2005). Communications of the ACM, 48(1), 15-18, 2005.

Barnard, J. and Price, A., 1994. Managing Code Inspection Information. IEEE Software, 11(2):59-69.

Berling, T, Thelin, T. (2003). An industrial case study of the verification and validation activities. Proceedings 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat No03EX717). IEEE Comput. Soc; 2003.

Bernd F., Lionel C. B., & Ferdinand V.ollei (2005). Using multiple adaptive regression splines to support decision making in code inspections. Journal of Systems and Software, 73(2), 205–217. doi:10.1016/j.jss.2004.01.015.

Biffl, S. (2000). Using inspection data for defect estimation. IEEE Software.17(6):36–43.

Bisant, D. B. & Lyle, J. R. (1989). A two-person inspection method to improve programming productivity. IEEE Transactions on Software Engineering, 15(10):1294-1304.

Blakely, F. W. & Boles, M. E. (1991). A case study of code inspections. Hewlett-Packard Journal, 42(4):58-63.

Boehm, B., & Basili, V. R. (2000). Gaining intellectual control of software development [Perspectives]. Computer, 33(5), 27–33.

Boehm, B., & Basili, V.R. (2005). Software defect reduction top-10 list. Foundations of Empirical Software Engineering, 426–431.

Boehm, B.; Basili, V.R. (2001). Top 10 list [software development], Computer, 34(1), 135-137.

Bourgeois, K. V. (1996). Process insights from a large-scale software inspections data analysis. Cross Talk, Journal of Defense Software Engineering, 17-23.

Carver, J. C., Nagappan, N., & Page, A. (2008). The Impact of Educational Background on the Effectiveness of Requirements Inspections: An Empirical Study. IEEE Transactions on Software Engineering, 34(6), 800–812.

Chaar, J. K., Halliday, M. J., Bhandari, I. S., & Chillarege, R. (1993). In-process evaluation for software inspection and test. IEEE Transactions on Software Engineering, 19(11), 1055-1055-1070.

Chandani, P., & Gupta, C. (2014). A Survey on Effective Defect Prevention - 3T Approach. International Journal of Information Engineering and Electronic Business, 6(1), 32–41.

Chen, Z., & Agrawal, H. (2014). Special issue on Emerging Topics on Software Debugging. Journal of Systems and Software, 90, 1–2. doi:10.1016/j.jss.2014.01.032

Chernak, Y. (1996). A statistical approach to the inspection checklist formal synthesis and improvement. IEEE Transactions on Software Engineering, 22(12):866-874.

Chernak, Y. (2001). Validating and improving test-case effectiveness, IEEE Software, 18(1),81-86, Jan/Feb 2001.

Chillarege, R, Bhandari, IS, Chaar, JK, Halliday, MJ, Moebus, DS, & Ray, BK. (1992). Orthogonal defect classification-a concept for in-process measurements. IEEE Transactions on Software Engineering. 18(11):943–56.

Conradi, R., Marjara, A., & Skåtevik, B. (1999). An empirical study of inspection and testing data at Ericsson, Norway.

Cooper, H., Hedges, L., & Valentine, J. (2009). The handbook of research synthesis and meta-analysis. New York, NY: Russell Sage Foundation Publications.

Cronbach, L. J. & Meehl, P. E. (1955). construct validity in psychological tests. Psychological Bulletin, 52 (4): 281–302.

De Sousa Coelho, J. J., Braga, J. L., & Ambrósio, B. G. (2013). System dynamics model for simulation of the software inspection process. ACM SIGSOFT Software Engineering Notes, 38(5), 1.

Denger, C., & Shull, F. (2007). A practical approach for quality-driven inspections. IEEE Software, 24(2), 79-79-86.

Dittrich, Y. (2014). Software engineering beyond the project – Sustaining software ecosystems. Information and Software Technology, 56(11), 1436–1456.

Dunham, J. R. (1989). V&V in the next decade [software validation]. IEEE Software, 6(3), 47–53.

Dyer, M. (1992a). The cleanroom approach to quality software development. New York: John Wiley and Sons, Inc.

Dyer, M. (1992b). Verification-based inspection. Proceedings of the 26th Annual Hawaii International Conference on System Sciences, pp. 418-427.

Eick, S. G., Loader, C. R., Long, M. D., Votta, L. G., & Vander Wiel, S. (1992,June). Estimating software fault content before coding. In Proceedings of the 14th international conference on Software engineering (pp. 59-65). ACM.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. IBM Systems Journal, 15(3):182-211.

Fagan, M. E. (1986). Advances in software inspections. IEEE Transactions on Software Engineering, 12(7):744-751.

Fagan, M. E. (1999). Design and code inspections to reduce errors in program development. IBM Systems Journal 38(2.3): 258–287.

Fagan, M. E. (2002). Advances in software inspections. Software Pioneers, 609–630.

Fagan, M. E. (2002). Reviews and Inspections. PP. 214-225.

Faraj, S., & Sambamurthy, V. (2006). Leadership of information systems development projects. IEEE Transactions on Engineering. Management, 53(2), 238–249.

Fernandez, A., Abrahão, S., & Insfran, E. (2013). Empirical validation of a usability inspection method for model-driven Web development. Journal of Systems and Software, 86(1), 161–186.

Freimut, B., Briand, L. C., & Vollei, F. (2005). Determining inspection cost-effectiveness by combining project data and expert opinion. IEEE Transactions on Software Engineering, 31(12), 1074–1092.

Gilb, T. & Graham, D. (1993). Software inspection. New York: Addison-Wesley Publishing Company.

Gimpel, J. (2014). Software That Checks Software: The Impact of PC-lint. IEEE Software, 31(1), 15–19. doi:10.1109/ms.2014.13

Gomes, J. O., & Moita, G. F. (2015). On the Validation of a Specific Development Process for Scientific Software using the Inspection Technique Abakós, 3(2). doi:10.5752/p.2316-9451.2015v3n2p3

Gopalakrishnan, R., Nair, T., Suma, V., and Kumar, T. (2012). “Significance of depth of inspection and inspection performance metrics for consistent defect management in software industry,” IET Software,. 6(6), 524, 2012.

Goswami, A., Walia, G., & Singh, A. (2015). Using Learning Styles of Software Professionals to Improve Their Inspection Team Performance. International Journal of Software Engineering and Knowledge Engineering, 25(9), 1721–1726. doi:10.1142/s0218194015710060.

Grady, R. B., & Slack, T. V. (1994). Key lessons in achieving widespread inspection use. IEEE Software, 11(4), 46–57.

Graziano, A. M., & Raulin, M. L. (2014). Research methods – A process of inquiry. 8th ed. Essex: England: Pearson Education Limited.

Gregory, F. D. (1993) Software formal inspections standard. Technical Report NASA-STD-2202-93, NASA Office of Safety and Mission Assurance, Washington, D.C.: NASA.

Hatton L. (2008). Testing the Value of Checklists in Code Inspections. IEEE Software. 25(4):82–8.

Hong, Y.• Baik, J., Ko, I. Y., & Choi, H. J. (2008, May). A Va lue- Added Predictive Defect Type Distribution Model based on Project Characteristics. lnComputer and Information Science, 2008.ICIS 08. Seventh IEEE/ACIS International Conference on (pp. 469-474). IEEE.

Houdek, F., Schwinn, T., & Ernst, D. (2002). Defect detection for executable specifications — An experiment. International Journal of Software Engineering and Knowledge Engineering, 12(6), 637–655.

Huhns, M. N., & Singh, M. P. (2005). Service-oriented computing: key concepts and principles. IEEE Internet Computing, 9(1), 75–81.

Hussain, F., & Shehzad, M. S. (2007). "Robust and Flexible Software Inspection model" for Software Re-Engineering Process: Abstraction phase. 14th Asia-Pacific Software Engineering Conference (APSEC'07).

Huzooree, G., & Devi Ramdoo, V. (2015). Evaluation of Code Inspection on an Outsourced Software Project in Mauritius. IJCA, 113(10), 39–44. doi:10.5120/19864-1827

IEEE Standard for Software Reviews and Audits (IEEE STD 1028-1988), IEEE Computer Society, 1988.

IEEE Standard for software reviews and audits. IEEE Std 1028-1988, Soft. Eng. Tech. Comm. of the IEEE Computer Society.

IEEE Std 1044-2009, 2009, IEEE Standard Classification for Software Anomalies IEEE standard classification for software anomalies. IEEE Std 1044-2009.

IEEE Std 730-2002 (Revision of IEEE Sid 730-1998), IEEE Standard for Software Qualit Assurance Plans, 2002.

J. W. Wilkerson, J. F. Nunamaker, and R. Mercer, "Comparing the Defect Reduction Benefits of Code Inspection and Test-Driven Development," IEEE Transactions on Software Engineering, vol. 38, no. 3, pp. 547–560, May 2012.

Jalote, P., Mittal, A. K., & Prajapat, R. G. (2007). On Optimum Module Size for Software Inspections. Int. J. Rel. Qual. Saf. Eng., 14(03), 283–295.

Jia Xu. (2003). Making software timing properties easier to inspect and verify. IEEE Software, 20(4), 34–41.

Kaner, C. (1998). The Performance of the N-Fold Requirement Inspection Method, Requirements Engineering Journal, 2(2): 114-116.

Kantorowitz, E., Guttman. A., & Arzi, L. (1997). The performance of the N-Fold requirement inspection method. Requirements Engineering, 2(3), 152-164.

Kaplan, C., Clark, R., & Tang, V. (1995). Secrets of Software Quality: 40 Innovations from IBM. New York: McGraw Hill, Inc.

Kasai, N., Morisaki, S., & Matsumoto, K. (2013). Fault-Prone Module Prediction Using a Prediction Model and Manual Inspection. 2013 20th Asia-Pacific Software Engineering Conference (APSEC).

Kelly D. & Shepard, T. (2004). "Task-directed software inspection," Journal of Systems and Software, 73(2), 361–368.

Knight JC, Myers EA. (1993). An improved inspection technique. Communications of the ACM. Association for Computing Machinery (ACM); 1993 Nov 1;36(11):51–61.

Knight, J. C. & Myers, E. A. (1991). Phased Inspections and their Implementation. ACM SIGSOFT Software Engineering Notes, 16(3):29-35.

Kollanus S. (2009). Experiences from using ICMM in inspection process assessment. Software Quality Journal. Springer Science + Business Media; 2009 Jan 10;17(2):177–87.

Kollanus, S., & Koskinen, J. (2009). Survey of Software Inspection Research. The Open Software Engineering Journal, 3(1), 15–34.

Koru AG, Dongsong Zhang, El Emam K, Hongfang Liu. (2009). An Investigation into the Functional Form of the Size-Defect Relationship for Software Modules. IEEE Transactions on Software Engineering. 35(2):293–304.

Laitenberger, O. (2002). A Survey of Software Inspection Technologies. Handbook on Software Engineering and Knowledge Engineering. In 2 Volumes, 517–555.

Land, L. P. W., Sauer, C., & Jeffery, R. (1997). Validating the defect detection performance advantage of group designs for software reviews. ACM SIGSOFT Software Engineering Notes, 22(6), 294–309.

Lange, C.F.J.; Chaudron, M.R.V.; Muskens, J. (2006). "In practice: UML software architecture and design description," Software, IEEE , vol.23, no.2, pp. 40- 46, March-April 2006.

Leite, J. C. S. do P., Doorn, J. H., Hadad, G. D. S., & Kaplan, G. N. (2004). Scenario inspections. Requirements Eng, 10(1), 1–21.

Li, X., Mutha, C., & Smidts, C. S. (2015). An automated software reliability prediction system for safety critical software. Empir Software Eng. doi:10.1007/s10664-015-9412-6

MacDonald F, Miller J, Brooks A, Roper M, Wood M. (1995). A review of tool support for software inspection. Proceedings Seventh International Workshop on Computer-Aided Software Engineering. IEEE Comput. Soc. Press; 1995.

Madachy, R., Little, L., & Fan, S. (1993). Analysis of a Successful Inspection Program. Proccedings of the 18th Annual NASA Software Engineering Laboratory Workshop, 176-198.

Mahmoud, M. A. W., Haggag, M. Y., & Abd, A. E. B. (2015). Cost Analysis of a Two-Unit Cold Standby System Considering Hardware, Software Failures and Inspection with Maximum Repair Time. IJCA, 129(5), 1–8. doi:10.5120/ijca2015906910

Malhotra, R., & Taneja, G. (2015). Comparative Analysis of two Stochastic Models subjected to Inspection and Scheduled Maintenance. IJSEIA, 9(10), 179–188. doi:10.14257/ijseia.2015.9.10.18

Mantyla, M. V., & Lassenius, C. (2009). What Types of Defects Are Really Discovered in Code Reviews? IEEE Transactions on Software Engineering. 35(3), 430–448.

Martin, J. & Tsai, W.T. (1990). N-fold Inspection: A Requirements Analysis Technique. Communications of the ACM, 33(2):225-232.

Mashayekhi, V., Drake, J. M., Tsai, W., & Riedl, J. (1993). Distributed, collaborative software inspection. IEEE Software, 10(5), 66-66-75.

Mello, R. M. de, Teixeira, E. N., Schots, M., Werner, C. M. L., & Travassos, G. H. (2012). Checklist-Based Inspection Technique for Feature Models Review. 2012 Sixth Brazilian Symposium on Software Components, Architectures and Reuse.

Meyer, B. (2008). Design and code reviews in the age of the internet. Communications of ACM, 51(9), 66.

Miller, J., & Yin, Z. (2004). A cognitive-based mechanism for constructing software inspection teams. IEEE Transactions on Software Engineering, 30(11), 811-825.

Minetola, P., Iuliano, L., & Calignano, F. (2015). A customer oriented methodology for reverse engineering software selection in the computer aided inspection scenario. Computers in Industry, 67, 54–71. doi:10.1016/j.compind.2014.11.002

Mishra, D., & Mishra, A. (2009). Simplified software inspection process in compliance with international standards. Computer Standards & Interfaces, 31(4), 763–771.

Misra, S., Fernández, L., & Colomo-Palacios, R. (2014). A simplified model for software inspection. J. Softw. Evol. and Proc., 26(12), 1297–1315.

Myers W. (1978). The Need for Software Engineering. Computer. 1978 Feb;11(2):12–26.

Nair T .R. G., Suma V, Kumar NRS. (2011). An analytical approach for project managers in effective defect management in software process. 2011 Malaysian Conference in Software Engineering. IEEE; 2011 Dec.

Nair, T. R. G., & Nair, N. G. (2011). Estimation of the characteristics of a software team for implementing an effective inspection process through inspection performance metric. Software Quality Professional, 13(2), 14-14-24.

Nair, T.R., Suma V. (2010). Impact Analysis of Inspection Process for Effective Defect Management in Software Development. Software Quality Professional Journal, American Society for Quality (ASQ). (March 2010) 4-14.

NASA-STD-8739.9 (2013). https://standards.nasa.gov/documents/detail/3315679

National Aeronautics and Space Administration. (1993). Software Formal Inspection Guidebook. Technical Report NASA-GB-A302, National Aeronautics and Space Administration. http://satc.gsfc.nasa.gov/fi/fipage.html.

O'Regan, G. (2002). Formal Methods and Design, A Practical Approach to Software Quality, 239–277, Springer. 2002.

Paired-samples T Test. (2015). Paired-samples T Test. (2015). retrieved from http://10.100.21.18:54604/help/index.jsp?topic=%2Fcom.ibm.spss.statistics.help%2Fspss%2Fbase%2Fovervw_auto_0.htm. Date Accessed: 2 March 2015.

Parnas DL, Lawford M. (2003). The role of inspection in software quality assurance. IEEE Transactions on Software Engineering. 29(8):674–6.

Parnas, D. L. & Weiss, D. (1985). Active Design Reviews: Principles and Practices. Proceedings of the 8th International Conference on Software Engineering, 132-136. Also Available as NRL Report 8927, 18 November 1985.

Parnas, D. L. (1987). Active Design Reviews: Principles and Practice. Journal of Systems and Software, 7:259-265.

Parnas, D. L., & Lawford, M. (2003). Inspection's role in software quality assurance. IEEE Software, 20(4), 16-16-20.

Parnas, D., & Weiss, D. M. (1987). Active design reviews: Principles and practices. Journal of Systems and Software, 7(4), 259–265.

Perry, D. E., Porter, A., Wade, M. W., Votta, L. G., & Perpich, J. (2002). Reducing inspection interval in large-scale software development. IEEE Transactions on Software Engineering, 28(7), 695-705.

Porter AA, Votta LG, Basili VR. (1995). Comparing detection methods for software requirements inspections: a replicated experiment. IEEE Transactions on Software Engineering.;21(6):563–75.

Porter, A. A., Siy, H. P., Toman, C. A., & Votta, L. G. (1997). An experiment to assess the cost-benefits of code inspections in large scale software development. IEEE Transactions on Software Engineering 23(6), 329–346.

Pothier, G.; Tanter, E. (2009). Back to the Future: Omniscient Debugging. Software, IEEE , 26(6), 78-85, Nov.-Dec. 2009.

Poulding, S., & Clark, J. A. (2010). Efficient Software Verification: Statistical Testing Using Automated Search. IEEE Transactions on Software Engineering, 36(6), 763–777.

Radice, R. A. (2002). Software Process Assessments. Encyclopedia of Software Engineering.

Remillard, J. (2005). "Source code review systems," Software, IEEE , 22(1), 74- 77. Jan.-Feb. 2005.

Ruhe, G., & Saliu, M. O. (2005). The Art and Science of Software Release Planning. IEEE Software, 22(6), 47–53.

Runeson, P., Andersson, C., Thelin, C., Andrews, A., & Berling, T. (2006). What do we know about defect detection methods? IEEE Software, 23(3), 82-90.

Schneider, G. M., Martin, J., & Tsai, W. T. (1992). An experimental study of fault detection in user requirements documents. ACM Transactions on Software Engineering and Methodology, 1(2):188-204.

Shaoying Liu, Yuting Chen, Nagoya, F., & McDermid, J. A. (2012). Formal Specification-Based Inspection for Verification of Programs. IEEE Transactions on Software Engineering. 38(5), 1100–1122.

Shen, W. G., Zhao, X. T., & Han, J. W. (2014). On Inspection Strategy Based on Sample Inspection Reliability. Applied Mechanics and Materials, 487, 639–642. doi:10.4028/www.scientific.net/amm.487.639

Sherif, Y. S., & Kelly, J. C. (1992). Improving software quality through formal inspections. Microelectronics Reliability, 32(3), 423–431.

Shin, Y., Meneely, A., Williams, L., & Osborne, J. A. (2011). Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities. IEEE Transactions on Software Engineering. 37(6), 772–787.

Shirey, G. C., (1992). How Inspections Fail. Proceedings of the 9th International Conference on Testing Computer Software, 151-159.

Shore, J. (2004). Fail Fast. IEEE Software, 21(05), 21–25.

Shull, F., & Seaman, C. (2008). Inspecting the History of Inspections: An Example of Evidence-Based Technology Diffusion. IEEE Software, 25(1), 88–90.

Shull, F., Basili, V., Boehm, B., Brown, A. W., Costa, P., Lindvall, M., Port, D., Rus, I., Tesoriero, R., Zelkowitz, M. (2002). What we have learned about fighting defects. Proceedings Eighth IEEE Symposium on Software Metrics. 249-258.

Shull, F., Melnik, G., Turhan, B., Layman, L., Diep, M., & Erdogmus, H. (2010). What do we know about test-driven development? IEEE Software, 27(6), 16–19.

Sneed, H., & Verhoef, C. (2015). From Software Development to Software Assembly. IEEE Software, 1–1. doi:10.1109/ms.2015.78

Sommerville, "Teaching cloud computing: A software engineering perspective," Journal of Systems and Software, vol. 86, no. 9, pp. 2330–2332, Sep. 2013.

Souza, I. S., da Silva Gomes, G. S., da Mota Silveira Neto, P. A., do Carmo Machado, I., de Almeida, E. S., & de Lemos Meira, S. R. (2013). Evidence of software inspection on feature specification for software product lines. Journal of Systems and Software, 86(5), 1172–1190.

Spiewak, R.,McRitchie, K. (2008). Using Software Quality Methods to Reduce Cost and Prevent Defects. CrossTalk, The Journal of Defense Software Engineering, 21(12), 23-27. (December, 2008)

Spinellis, D. (2010). Software Tracks. IEEE Software, 27(2), 10–11.

Strauss, S. H. & Ebenau, R. G. (1993). Software Inspection Process. McGraw Hill Systems Design & Implementation Series.

Suma, V., & Nair, T. R. G. (2014). Impact of test effort in software development life cycle for effective defect management. International Journal of Productivity and Quality Management, 13(3), 251.

Suma, V., & Nair, T. R. G. (2014). Impact of test effort in software development life cycle for effective defect management. International Journal of Productivity and Quality Management, 13(3), 251.

Sumit, B., Patil S.M. (2011). A Practical Experiment In Teaching Software Engineering Metrics. Journal Of Computational Simulation And Modeling, 1 (1), 10-16.

Thelin, T., Runeson, P., & Wohlin, C. (2003). Prioritized use cases as a vehicle for software inspections. IEEE Software, 2003; 20(4), 30-33.

Travassos, G., Shull, F., Fredericks, M., & Basili, V. R. (1999). Detecting defects in object-oriented designs. Proceedings of the 14th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '99.

Tyran, C. K. (2006). A software inspection exercise for the systems analysis and design course. Journal of Information Systems Education, 17(3), 341-341-351.

Van Genuchten, M., van Dijk, C., Scholten, H., & Vogel, D. (2001). Using group support systems for software inspections. IEEE Software, 18(3), 60–65.

Vitharana, P. (2015). Defect propagation at the project-level: results and a post-hoc analysis on inspection efficiency. Empir Software Eng. doi:10.1007/s10664-015-9415-3

Vodde, B. (2007). Experiences in software inspection measurements. Software Quality Professional, 9(2), 27-27-35.

Votta, L. G. (1993). Does Every Inspection Need a Meeting? ACM Software Engineering Notes, 18(5):107-114.

Walia, G. S., & Carver, J. C. (2009). A systematic literature review to identify and classify software requirement errors. Information and Software Technology, 51(7), 1087–1109.

Weinberg, G. M., & Freedman, D. P. (1984). Reviews, Walkthroughs, and Inspections. IEEE Transactions on Software Engineering. SE-10(1), 68–72.

Weller, E. F., (1993). Lessons from Three Years of Inspection Data. IEEE Software, 10(5):38-45.

Wiegers, K. E. (1995). Improving Quality Through Software Inspections.

Williams, C. C., & Hollingsworth, J. K. (2005). Automatic mining of source code repositories to improve bug finding techniques. IEEE Transactions on Software Engineering. 31(6), 466–480.

Winkler, D., Riedl, B., & Biffl, S. (2005). Improvement of Design Specifications with Inspection and Testing. 31st EUROMICRO Conference on Software Engineering and Advanced Applications.

Winkler, D., Thurnher, B., & Biffl, S. (2007). Early software product improvement with sequential inspection sessions: An empirical investigation of inspector capability and learning effects. 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007).

Yang, Y., Onita, C., Zhang, X., & Dhaliwal, J. (2010). TESTQUAL: Conceptualizing software testing as a service. E - Service Journal, 7(2), 46-65,101-102.

Yin, Z., Dunsmore, A., & J. Miller, J.(2004). “Self-assessment of performance in software inspection processes,” Information and Software Technology, 46(3), 185–194.

Zhang, H., & Ali Babar, M. (2013). Systematic reviews in software engineering: An empirical investigation. Information and Software Technology, 55(7), 1341–1354.

Zheng, J., Williams, L., Nagappan, N., Snipes, W., Hudepohl, J. P., & Vouk, M. A. (2006). On the value of static analysis for fault detection in software. IEEE Transactions on Software Engineering, 32(4), 240–253.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Hashemitaba, N., & Ow, S.H. (2014). A new model for software inspection at the requirements and design phases of software development. Accepted for publication by *The International Arab Journal of Information Technology* (IAJIT). (*ISI-Cited Publication*)

Hashemitaba, N., & Ow, S.H. (2012). Defect management using a comprehensive software inspection model. Software Engineering Journal, 2(5): 160-164.

Hashemitaba, N., Khatavakhotan, A. S. & Ow, S.H. (2012). A Comprehensive model to improve the efficiency of software inspection: A case study. *International Journal of Information Technology & Computer Science.* 5:30-37.

Hashemitaba, N., & Ow, S.H. (2012). *Generative inspection: an intelligent model to detect and remove software defects.* Proceedings of the 3rd International Conference on Intelligent Systems, Modelling and Simulation (ISMS2012), February 8-10, 2012, Sabah, Malaysia (*ISI-Cited Publication*).

Hashemitaba, N., & Ow, S.H. (2012). A scenario- based model to improve the quality of software inspection process. Proceedings of the 4th International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM), 25-26 September, 2012, Kuantan, Pahang. pp. 194-198. (*ISI-Cited Publication*).

Hashemitaba, N., Khatavakhotan, A. S. & Ow S.H. (2012). *Prosperity in software inspection: Improvement of software development process using an inspection smart boost.* Conference on Computer Modelling and Simulation. CSSIM 2012. 3-5 September 2012, Brno University of Technology.

Hashemitaba, N., & Ow, S.H. (2012). *Software Defect Management Using a Comprehensive Software Inspection Model*, Software Engineering Journal, 2012, 2(5): 160-164 DOI: 10.5923/j.se.20120204.09

Hashemitaba, N., & Ow, S.H. (2012). *Improving software quality using a defect management-oriented (DEMAO) software inspection model.* Asia Modeling Symposium 2012, Sixth Asia International Conference on Mathematical Modeling and Computer Simulation Publication. 46 – 49.

Hashemitaba, N., Khatavakhotan, A. S., & Ow, S.H. (2012). *A comprehensive model to improve the efficiency of software inspection: A case study.* Proceedings of the International Conference on Information Integration and Computing Applications Singapore. Singapore, August 14-15, 2012. (*ISI-Cited Publication*).

Hashemitaba, N., Khatavakhotan, A. S., & Ow, S.H. (2012). *A scenario-based model to improve the quality of software inspection process.* Computational Intelligence, Modeling and Simulation Fourth International Conference on Topic(s): Computing & Processing (Hardware/Software). 59: 194 – 198.

Hashemitaba, N., Khatavakhotan, A. S., & Ow, S.H. (2012). A new model to improve the quality of software inspection Postgraduate Research Excellence Symposium - 25 Sep 2012 - Kuala Lumpur Malaysia.

Hashemitaba, N., Khatavakhotan, A. S., & Ow, S.H. (2012). ICCSA 2012. *An integrated model to improve the role of inspection in software quality process,* International Conference on Computer Science and Applied Computing ,Vienna, Austria. 6-7 September 2012.

Hashemitaba, N., Khatavakhotan, A. S., & Ow, S.H. (2011). A novel four-faceted telecommunication model based on distributed connectivity technology systems International Conference on Computer and Software Modeling. IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore.

APPENDIX A

ARTEFACT COMPLEXITY CONVERSION

In software development, no code is written at the software requirements analysis and design phases, hence, measurement of the complexity an artefact cannot be based on the size of software or the number of lines of codes. In this case, the function point must be used as a complexity-related indirect method of measurement for software work products [IFP, 2005]. Boehm demonstrated the method of categorization and measurement in 1996. Artefacts are divided into two groups - Data Entries, and Transactions artefacts. Transactions artefacts are often derived from DFD graphs, which are used in both the analysis and design phases, and have a separate categorization, as shown in Table A.1. Entity relationship diagrams are classic methods and class and object and collaboration are used in object-oriented methods for data entities artefact categorization [Uem99]. Table A.1 also shows the data entities sub-classes.

Table A.1. Transactions and Data entities sub-classes

Function Point			
Transactions		*Data entities*	
(EI[1]), (EO[2]), (EQ[3])		(EIF[4]), (ILF[5])	
No. of data entity types (FTR)	No. of attribute types accessed (DET).	No. of attribute types (DET)	No. of data entity subtypes (RET)

1: External Inputs, 2: External Outputs, 3: External Queries, 4: External Files 5: External Entities

Each of these sub-categories can have high, average or low level of complexity. The relevant weight of each sub-category can be seen in the table. In the present research, the complexity of selected artefacts is measured using Boehm's formula. The first step is to determine whether each artefact is simple, medium or complex. The type is also

determined and based on Table A.2 their weight quotient (weight factor) was calculated, and finally the function point is obtained.

Table A.2 Complexity Table

Domain Value	Complexity / Weighting Factor		
	Low	Medium	High
External Inputs EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Following the preliminary calculation of complexity, the figures obtained are the Unjustified Function Points (UFP). After considering the 14 complexity factors and using Equation A.1, the Adjusted Function Points are obtained. The complexity factors will each have a value of between 0 and 5 and their sum is the Value Adjusted Factors (VAF) [Lon, 2002]. The final complexity factor is obtained by considering all 14 factors (See Table A.3), and it can range from 0 to 70 (14*5). This indicates that the final function point or AFP can vary from 0.65 to UFP 1.35. Data on the functionality of artefacts have been included in Table A.4 together with explanations by the project managers and experts. Since the projects are still on-going, it is not possible to compare the artefacts, and also not possible to answer a large number of questions related to them. Thus, the preliminary FPs (UFP) are considered equal to the AFP in this research. Eventually, considering the time taken and the defects detected based on the function point of each artefact, the quality of the inspection and the productivity of the inspection teams as well as the inspectors were assessed and compared.

Table A.3. Function point adjustment questions [Lon, 2002]

1	The necessity for crucial backup and recovery	8	On-line updating of the ILFs
2	Requirements of information transferred by special data.	9	Complexity of components.
3	Distributed processing functions requirements.	10	Complexity of internal processing.
4	The expected level of performance.	11	Need for reusability.
5	Implementation in on operational platform.	12	Need for conversion of detailed design.
6	Need for online data entry.	13	Different implementation and setup requirements.
7	Need for multiple processes and concurrent operations.	14	Need for flexibility.

Equation A.1 Adjusted Function Point calculation

$$VAF = \sum F_i \ (i = 1 \text{ to } 14)$$

$$AFP = UFP * ((TDI * 0.01) + 0.65)$$

Table A.4 shows the page numbers and calculation of FP related to all 32 artefacts selected for the two projects. Items related to the product complexity are also mentioned in this table and are sorted in the relevant column according to their weight. The page number of each artefact is registered in the second column. In this research, the relationship between the FP (last column) as the metrics for complexity of the artefact, and the page number as the size of the artefacts, is as shown in Figure A.1. This Figure also shows a direct positive relationship between those two metrics. Therefore, the use of the page number as the size of artefact is justified, and there is no need to use the function point. This is also because the use of the function point only shows the same results but with different values and in different ranges.

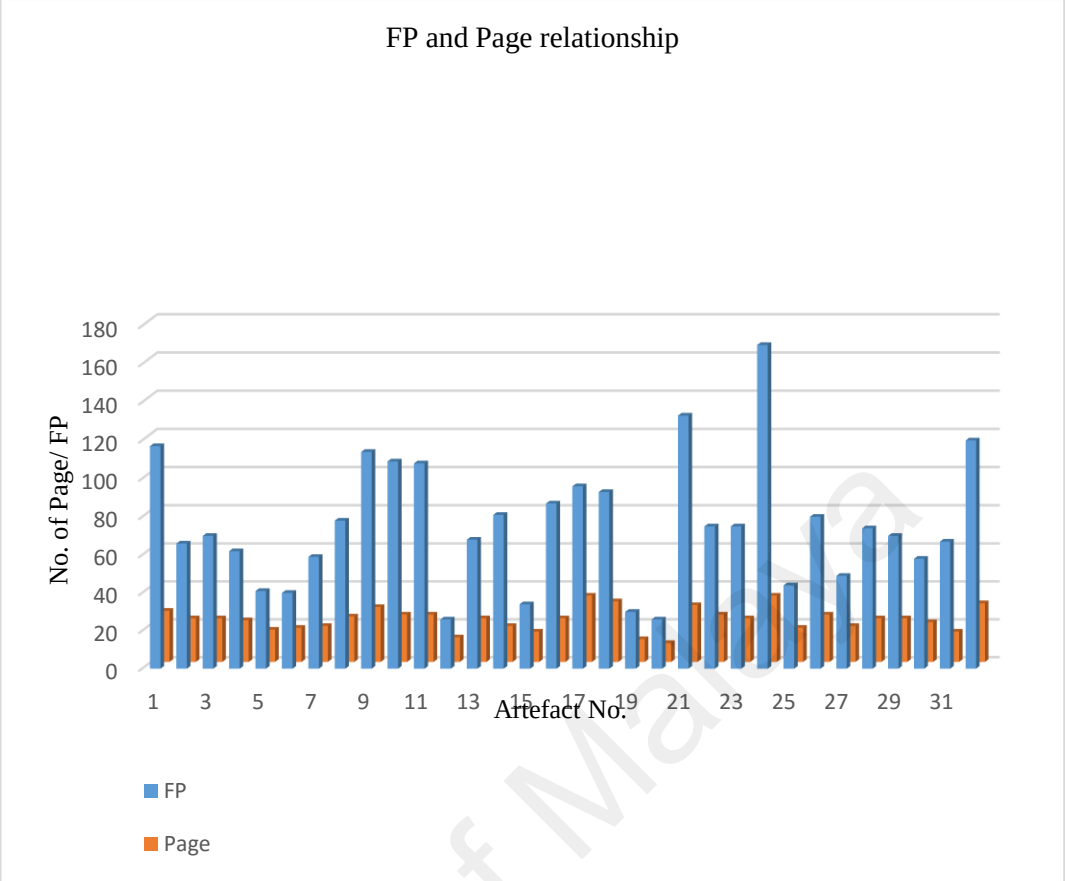
Table A.4 FP and number of Pages of Artefacts

Project 1								Project 2							
	AC	Pages	DM	CPX	WF	RFC	FP		AC	Pages	DM	CPX	WF	RFC	FP
Requirements Analysis Phase	1.	**27**	D1	H	15	5	**117**	Requirements Analysis Phase	17.	**35**	D1	L	7	8	**96**
			D2	M	7	6					D2	H	10	4	
	2.	**23**	D1	H	15	3	**66**		18.	**32**	D1	L	7	9	**93**
			D2	M	7	3					D2	L	5	6	
	3.	**23**	D1	M	10	4	**70**		19.	**12**	D1	-	-	0	**30**
			D2	L	5	6					D2	H	10	3	
	4.	**22**	D1	L	7	6	**62**		20.	**10**	D1	L	2	3	**26**
			D2	H	10	2					D2	L	5	4	
	5.	**17**	D1	M	2	3	**41**		21.	**30**	D1	L	7	9	**133**
			D2	M	7	5					D2	H	10	7	
	6.	**18**	D1	-	-	0	**40**		22.	**25**	D1	M	10	4	**75**
			D2	H	10	4					D2	L	5	7	
	7.	**19**	D1	L	7	7	**59**		23.	**23**	D1	L	7	5	**75**
			D2	L	5	2					D2	H	10	4	
	8.	**24**	D1	M	1	8	**78**		24.	**35**	D1	H	15	9	**170**
			D2	H	10	7					D2	L	5	7	
Design Phase	9.	**29**	D3	M	4	3	**114**	Design Phase	25.	**18**	D3	M	4	8	**44**
			EO	M	5	6					EO	-	-	0	
			D4	H	6	4					D4	L	3	4	
	10.	**25**	D3	H	6	8	**109**		26.	**25**	D3	M	6	4	**80**
			EO	H	7	6					EO	H	7	4	
			D4	M	4	5					D4	M	4	7	
	11.	**25**	D3	H	6	7	**108**		27.	**19**	D3	H	6	3	**49**
			EO	M	5	5					EO	L	4	4	
			D4	L	3	7					D4	L	3	5	
	12.	**13**	D3	L	1	2	**26**		28.	**23**	D3	L	3	2	**74**
			EO	-	-	0					EO	M	5	8	
			D4	M	4	6					D4	M	4	7	
	13.	**23**	D3	H	6	7	**68**		29.	**23**	D3	H	6	3	**70**
			EO	M	5	2					EO	M	4	6	
			D4	M	4	4					D4	M	4	7	
	14.	**19**	D3	L	3	6	**81**		30.	**21**	D3	H	6	5	**58**
			EO	M	5	9					EO	M	5	4	
			D4	H	6	3					D4	H	2	4	
	15.	**16**	D3	M	4	6	**34**		31.	**16**	D3	H	6	7	**67**
			EO	L	2	5					EO	L	4	4	
			D4	-	-	0					D4	L	1	9	
	16.	**23**	D3	H	6	8	**87**		32.	**31**	D3	M	4	9	**120**
			EO	L	4	6					EO	H	7	6	
			D4	L	3	5					D4	H	6	7	

AC: Artefact, D1: ILF, D2:ELF, D3:EI, D4:EQ, D5: EO

Requirement Function Counts, H: High M: Medium L: Low, FP: Function Point

AC: Artefact Code, WF: Weighting factor, DM: Domain Value, CPX: Complexity



i < 9 & 17<i<25 Artefact i: P1-Ai-R 9<i<17 & 24<i<=32 : Artefact i: P1-Ai-D

Figure A.1 The relationship between complexity of artefacts (FP) and number of pages

APPENDIX B

THE COMPOSITION OF FIVE INSPECTORS

The inspection teams formed in this research consist of three-inspector teams. However, five-inspector or seven-inspector teams could also be formed. Table C.1 shows the combination of weighted votes of an inspection team that consists of five inspectors. Any combination of four votes will not give rise to any problem. In a three-vote and a two-vote situation, some problems concerning the weight of votes arise. As shown, some combinations in row 5 and row 10 are not suitable and could give rise to problems. It is clear that any combination that include three semi-skilled the worst combination. A combination of highly skilled and semi-skilled inspectors could give rise to more votes of semi-skilled inspectors with less vote of highly skilled inspector with higher weightage. Combination of highly skilled inspectors is preferred.

Table C.1: Combination of weighted votes of five inspectors

	Combination of three similar vote	**Two similar vote**	**Decision**	**Two similar vote**	**Decision**	**wo similar vote**	**Decision**	**Two similar vote**	**Decision**	**Two similar vote**	**Decision**	**Two similar vote**	**Decision**
1	HHH	HH	9>6	HN	9>5	HS	9>4	NN	9>4	NS	9>3	SS	9>2
2	HHN	HH	8>6	HN	8>5	HS	8>4	NN	8>4	NS	8>3	SS	8>2
3	HHS	HH	7>6	HN	7>5	HS	7>4	NN	7>4	NS	7>3	SS	7>2
4	HNN	HH	7>6	HN	7>5	HS	7>4	NN	7>4	NS	7>3	SS	7>2
5	HNS	HH	6=6	HN	6>5	HS	6>4	NN	6>4	NS	6>3	SS	6>2
6	**HSS**	HH	5<6	HN	5=5	HS	5>4	NN	5>4	NS	5>3	SS	5>2
7	**NNN**	HH	6=6	HN	6>5	HS	6>4	NN	6>4	NS	6>3	SS	6>2
8	**NNS**	HH	5<6	HN	5=5	HS	5>4	NN	5>4	NS	5>3	SS	5>2
9	**NSS**	HH	4<6	HN	4<5	HS	4=4	NN	4=4	NS	4>3	SS	4>2
10	**SSS**	HH	3<6	HN	3<5	HS	3<4	NN	3<4	NS	3=3	SS	3>2

H: Highly skilled, N: Normal (Skilled), S: Semi-Skilled

APPENDIX C

SAMPLE CHECKLIST

(A) Requirements Analysis - Use Case Sample Checklist

The purpose of this document is to provide an inspection checklist of the use case diagram(s) for (Artefact **P2-A6-R)**. Two inspection teams will inspect the documents and provide a report on the result of their inspection. Team C will inspect the artefact using the by Formal inspection process (FIP) together with Microsoft Excel, and Team D will inspect the artefact using the enhanced software inspection process together with ArSeC online system. Some reference documents and resources will be provided to give some background information of the system to the inspectors.

No.	Inspection Question	Pass/Fail ☑, ☒
1.	Are all the actors and steps cited in the use case relevant to the execution of the task?	
2.	Are areas of uncertainty documented as assumptions and issues?	
3.	Are system boundaries and scope clear?	
4.	Can the system meet that objective?	
5.	Do all actors and use cases have descriptive names?	
6.	Does it "validate" as opposed to "check" a condition?	
7.	Does it have less than 10 steps?	
8.	Does it run from trigger to delivery of the success guarantee?	
9.	Does the use case contain a complete step-by-step plan?	
10	Has each step in the scenario(s) been clearly, unambiguously and completely described?	
11	Have all the alternatives been described?	

12	Have all the known exceptions been described?	
13	Is it an active-verb phrase that mentions the objective of the primary actor?	
14	Is it clear for which actors the use case is intended?	
15	Is it phrased as a goal that can be achieved?	
16	Is the aim of the use case clear?	
17	Is the goal level of the step lower than the goal level of the overall use case?	
18	Is the intent of the actor clear?	
19	The symbols in the use case diagram conform to the UML diagram.	
20	Use case diagrams and descriptions are clear and well organized.	

General Comments, Recommendations, and Suggestions (if any):

(B) Requirements Analysis – Class Diagram Sample Checklist

The purpose of this document is to provide an inspection checklist of the class diagram(s) for (Artefact **P2-A2-D)**. Two inspection teams will inspect the documents and provide a report on the result of their inspection. Team C will inspect the artefact using the by Formal inspection process (FIP) together with Microsoft Excel, and Team D will inspect the artefact using the enhanced software inspection process together with ArSeC online system. Some reference documents and resources will be provided to give some background information of the system to the inspectors.

No.	Inspection Question	Pass/Fail ☑, ☒
1.	All kinds of identification like 'id' or 'name' attribute, provided?	
2.	All needed associations from the requirements are considered.	
3.	All needed attributes are added?	
4.	All needed classes from the requirements (mostly they appear as nouns)?	
5.	Are all class names written in the singular and starting with a capital letter?	
6.	Are there direct associations (arrows), and are they correct?	
7.	Class diagram and descriptions are clear and well organized.	
8.	Do all associations have multiplicity at both ends?	
9.	Do the attributes represent simple data that each instance must have String, Integer, Float, Date, Time, Boolean etc.?	
10.	Do role names start with lower case letters and have a meaning that is a role the class plays?	
11.	Does the 'isa rule' apply to all subclasses and their superclass?	
12.	Does everything in each superclass also apply to every one of its subclasses? Are there any classes with huge numbers of attributes (>10) or associations (>5)?	
13.	Each message in sequence diagram is a method in class diagram.	
14.	Each message passing in sequence diagram is the method in class diagram.	
15.	If there is message passing between objects in sequence diagram, association relationship in class diagram is defined.	
16.	Is it correct that any unlabeled associations can be read as 'has'?	
17.	Is there an open diamond to specify an aggregation, is this really the 'whole' end of a proper part-whole relationship?	
18.	Is there any classes that should be split into an 'Abstraction' and an 'Occurrence' class using a * -- 1 association?	

19.	Is there any instance hierarchies in the model and used either an asymmetric reflexive association or the general hierarchy (composite) pattern?	
20.	Is there plural attributes but could be replaced by associations instead.	
21.	Names used in class diagram indicate their meaning. Example: class Multicast	
22.	The symbols using in class case diagram conform to UML diagram.	
23.	Using proper open triangle symbol, pointing to the superclass?	
24.	All kinds of identification like 'id' or 'name' attribute, provided?	
25.	All needed associations from the requirements are considered.	
26.	All needed attributes are added?	
27.	All needed classes from the requirements (mostly they appear as nouns)?	
28.	Are all class names written in the singular and starting with a capital letter?	
29.	Are there direct associations (arrows), and they are correct?	
30.	Class diagram and descriptions are clear and well organized.	
31.	Do all associations have multiplicity at both ends?	

General Comments, Recommendations, and Suggestions (if any):

APPENDIX D

INSPECTION DATA RECORDING FORM

Section A: Project and Artefact Details

1. Project ID: _____
2. Project title: _____

_____

3. Project brief description: _____

_____

4. Artefact details:

Requirements analysis phase			
No.	Artefact ID	Artefact title	Artefact brief description
1.			
2.			
:			
Design phase			
1.			
2.			
:			

Section B: Inspector Profiles and Inspection Team Members

1. Inspector ID: _____
2. Inspector Details:

No.	Inspector ID	Inspector name	Contact		No. of years of work experience	No. of artefacts inspected
			Tel.	Email		
1.						
2.						
3.						
4.						
5.						

3. Inspection Team Member Details:

No.	Role (Author, Moderator, System Administrator, etc.)	Name	Contact	
			Tel.	Email
1.				
2.				
3.				

4. Details about artefacts inspected (current project):

Requirements analysis phase				
Project ID:				
Artefact ID:				
No.	Inspector ID	Actual total No. of real defects detected	Total inspection time (min)	Inspector productivity
1.				
2.				
3.				
4.				
5.				
Artefact ID:				
1.				
2.				
:				
Design phase				
Project ID:				
Artefact ID:				
1.				
2.				
3.				
4.				
5.				
Artefact ID:				
1.				
2.				
:				

Note: Inspection productivity = Total No. of real defect detected / inspection time (hr)

Section C: Inspection Process

Requirements analysis phase						
Project ID:						
Number of inspectors:						
Inspector ID:						
Inspection stopping criteria:						
No.	Artefact ID	Inspection date and time	Inspection duration	Unexpected incident(s)	Issues discussed	Inspection outcomes summary
1.						
2.						
:						
Design phase						
Project ID:						
Number of inspectors:						
Inspector ID:						
Inspection stopping criteria						
1.						
2.						
:						

Section D: Defects Detected

Requirements analysis phase						
Project ID:						
Artefact ID						
No.	Defect code	Defect class	Mode	Severity	Description	Detected by (Inspector ID)
1.						
2.						
:						
Design phase						
Project ID:						
Artefact ID						
No.	Defect code	Defect class	Mode	Severity	Description	Detected by (Inspector ID)
1.						
2.						
:						

Summary of Defects Detected					
Requirements analysis phase					
Project ID:					
No.	Artefact ID	Total number of common real defects detected	Total number of false positives	Total number of inconsistencies	Total number of ambiguous artefacts
1.					
2.					

:					
Design phase					
Project ID:					
1.					
2.					
:					

Section E: Inspection Time

Requirements analysis phase																
Project ID:																
No.	Artefact ID	Inspection preparation time of each inspector					Total inspection preparation time (All inspectors)	Inspection time of each inspection session				Inconsistencies resolution time				Total inspection time
		1	2	3	4	5		1	2	3	4	1	2	3	4	
1.																
2.																
:																
Design phase																
Project ID:																
No.	Artefact ID	Inspection preparation time of each inspector					Total inspection preparation time (All inspectors)	Inspection time of each inspection session				Inconsistencies resolution time				Total inspection time
		1	2	3	4	5		1	2	3	4	1	2	3	4	
1.																
2.																
:																

Section F: Defect Causes and Effects

Requirements analysis phase						
Project ID:						
No.	Defect code	Description	Potential causes	Description	Potential effects	Description
1.						
2.						
:						
Design phase						
Project ID:						
1.						
2.						
:						

Note: Update details of existing defects and record details of new defects detected.

APPENDIX E

BRIEF DESCRIPTION ABOUT PROJECT 1, PROJECT 2, AND EACH ARTEFACT AND SAMPLE OF ARTEFACT

Brief description about Project 1, and Project 2

Project ID	Description
P1	Integrated system for failure recording system for production line. This system includes all sub-systems for all steps from production line to inventory control. The project took 3 years to complete. The company has a branch in Malaysia. It has several factories and workshops which are involved in developing mechanical health care products and vehicle items.
P2	A Malaysian trading company. The project was a web-based system as total solution for gathering and reporting the import and export orders. The projects also involved tracking the shipments from the place of origin to the second and third and final destinations.

Brief description of each artefact

Project ID: P1	Project title: Integrated system for failure recording system for production line	
Artefact ID	Artefact title	Description of artefact
P1-A1-R	Process specification for failure mode and effects analysis (FMEA) sub system	This artefact is a part of total quality management system for spotting and evaluating failures in the production process. There is description of each process, the potential hazards, the defects, the patterns, severity and frequency of occurrence.
P1-A2-R	DFD for process specification for failure mode and effects analysis (FMEA) sub system	This artefact shows the sequence of steps and data relationship between the processes which are specified through FMEA artefacts. The focus is on the data which have to be stored and retrieved.
P1-A3-R	Use cases for process specification for failure mode and	The actors and their roles and relationship for implementing the failure mode. Effects analysis is illustrated by diagrams.

	effects analysis (FMEA) sub system	
P1-A4-R	Process specification for Measurement system analysis (MSA) subsystem	This artefact shows the measurement analysis and evaluations. The instructions, criteria and flowcharts are defined to ensure that the current measurement system is reliable enough and performs to effect improvement and change (if any) to the operators ID fields, profiles, roles, equipment specifications, the parts and the components feature defined.
P1-A5-R	DFD for Process specification for Measurement system analysis (MSA) subsystem.	This artefact shows the data flow in the processes as well as data storage and retrieval.
P1-A6-R	Process specification for production planning and control	This artefact shows the necessary data, information and standards for controlling and planning the product in production line.
P1-A7-R	DFD for Specification for production planning and control.	The data to be stored permanently or temporarily are defined and the processes relationships are shown.
P1-A8-R	ER and DFD diagrams for order registration.	This artefact involves order registration and tracing of orders. The different dates from order to delivery, and order details and customer details of as well as the steps to produce the whole or part of an order are included.
P1-A1-D	Complaint resolution class diagrams.	The new and current class diagrams focus on reusability of the CRM system. This artefact includes the details of complaints and the resolution results, and the legal and police reports details (if any).
P1-A2-D	Web interface design for complaint resolution.	The artefact contains the complaints about the quality or other issues pertaining to products and the delivery process. A strong relationship between this sub-system and the customer relationship management system (artefact P1-A1-D).

P1-A3-D	Failure removal class diagrams.	Class diagrams which show the formula, comparison methods, and details of the mode and effects analysis for failures.
P1-A4-D	Web interface design failure removal.	The artefact includes the details of potential failures and defect occurrence details, and comparison reports on severity estimation and actual occurrence (if any).
P1-A5-D	Order tracing class diagrams.	The order tracing class diagrams include the classes and methods for tracing the orders, estimation information, and evaluation formulas.
P1-A6-D	Web interface design for order registration and tracing.	The artefact shows the input details and reports for each step, and current status of the production process for an order.
P1-A7-D	Class diagrams for production control.	The procedures classes and methods, local and global data used to control the production step-by-step.
P1-A8-D	Web interface design for production control.	The interface for inputting the measured features like sizes, and test results details for each work product status of the order. It is possible to change the standards and the comparison methods. It is an artefact of the detailed design of a production control system.
Project ID: P2	Project title: Import-Export Management System	
P2-A1-R	Import customer request specification.	The customer details and its shipment information and the port of origin are included in this artefact. It also includes special information like urgent shipment requests. The cost estimation and basic account information and the credit limit of customer (New or loyal customer) are also included.
P2-A2-R	Use case for import customer requests.	The artefact shows the role and interactions between the actors and systems for an import shipment. The type of relationship and description are explained and shown by diagrams.

P2-A3-R	Export customer request specification.	The export shipment to a predetermined port or destination is described. The customs process and information on the destination are not included. Details on type of transportation and the due dates are provided.
P2-A4-R	Use case for export customer requests.	The artefact shows the role and interactions between the actors for exporting a shipment. The type of relationship and the descriptions are explained and shown by diagrams. However, the customs procedures and other procedures are not mentioned.
P2-A5-R	Shipment tacking-monitoring specification (Handle by other companies).	The specifications for shipment tracking that is handled by other companies but monitored to ensure correct and on-time delivery. The import or export shipments which are monitored step-by-step include the payments and customs procedures.
P2-A6-R	Use case for tracking monitoring subsystem.	The roles and actors for tracking shipments (import or export) are illustrated by diagrams.
P2-A7-R	Custom subsystem specification (export shipments).	If requested by customer, the customs procedures will be handled by company. The detailed information for pre-evaluation and actual evaluation and payments are considered.
P2-A8-R	Use case for Custom subsystem.	This includes the various agents, the transfer of money, customs or customers details, customer credit account, the LCs.
P2-A1-D	Class diagrams for Import customer request.	The artefact shows the classes and methods for an import shipment. The customs information is considered.
P2-A2-D	Class diagrams for Export customer request.	The artefact shows the class diagrams and methods for exporting a shipment. The customs information is not included.
P2-A3-D	Class diagrams for Shipment tacking-	The class diagrams and methods necessary for tracking and monitoring the shipments.

	monitoring (Handle by other companies).	
P2-A4-D	Class diagrams for Custom subsystem.	This artefact includes the class diagrams for export shipment that the company has requested for customs procedures.
P2-A5-D	Class diagrams for integrated report generator.	The artefact includes the reused and the new class diagrams and methods for generating dynamic reports for shipments.
P2-A6-D	Web page design for integrated reports.	The details of the web design for the reports for various subsystems (customer details, profile, shipments, source and destinations, income and payments for each order).
P2-A7-D	Class diagrams for statistical data analysis outputs.	The artefact shows the class diagrams and methods for statistical data analysis (shipment details and account information).
P2-A8-D	Web page design for statistical data analysis outputs.	This artefact shows the necessary scenarios as well as fields, controls and algorithms for data analysis of the shipments and customer accounts. The status transitions are considered.

Requirements Analysis Sample Artefact

Project ID: P2	**Company: Malaysia**
Project title: Import-Export Management System	
Artefact ID: P2-A6-R	
Artefact title: Use case for tracking monitoring subsystem	THE FIRST PAGE OF ARTEFACT 6

Sun shine Company
Lot ... M KL
Kuala Lumpur, Malaysia
Phone: 03-
Fax: 03-
E-mai l: info@s

Date: 9/6/2013
Reference # 14/25/E17
Page 1/18

Use Case ID	Use Case name	Developer(s) Name and ID
UC-6/11	Shipment track	

Seq.	Component	Description
1	Main Goal	Providing the necessary information about the status of a shipment and the necessary actions
2	Sub-goal	Customer trust and satisfaction by up-to-date information - avoiding late payment or late arrival of shipment
3	Scope:	Core - Jointed to accounting, auditing, and customer order and invoice
4	Level	Detailed - Fish Level
5	Primary Actor Role	Senior tracker, Customer
6	Preconditions	Tracker has connected through a safe terminal
7	The state of the system upon successful	Update the current status and required action(s) in related databases
8	The state of the system not success	Connection failed. Shipment lost. Recall the complaint resolution, investigation, and follow-up
9	Success End conditions	Getting and logging the reference and details of legal documents regarding new status
10	Failed End Condition	Shipment stuck, captured or returned to the source, dead line passed, connections not safe or terminated
11	Trigger	Sending track request from the Tracker or Cutomer
12	Main Success Scenario	a) A: Open the corresponded URL and provide the security access code b) S: Validate the access code c) A: Request for current status d) S: Provide the current Status (in case of any changes) e) A: Update the shipment records
13	Extensions	b1) Invalid code : log - Message d1) Final destination - trigger order checkout
14	Sub-Variations	The actions are possible through the telephone call
15	Priority	High
16	Response Time	Variable on customer request , order priority - event driven
17	Frequently	Anytime until arrival of shipment at final destination and issuance the corresponded documents
18	Channels to primary Actors	Shipment database - customer database, agent database-company profile database - Invoice database
19	Secondary Actors	Complaint resolution database, quality assurance dept., risk management dept.
20	Channel to Secondary Actors	Complaint database - Risk management database, QA dept. database

APPENDIX F

SAMPLE CAUSE AND EFFECTS

a) Sample causes and effects for defects in requirements analysis phase

Sample causes and effects of potential defects in use case diagrams		
Cause Code	**Cause**	**Effect(s)**
US-1	Defining the same activity for different actors	The data security ; Unreliable data damaging the access rights
US-2	Weakness in distinguish between the responsibilities and duties of different actors	The customer actions are not comprehensive
US-3	The differences between the action of staff (system actor) and customer (Primary actor) in same situation cannot be distinguished	The scenarios those have to be different, may be planed same.
US-4	There is a delete option while (usually) after confirmation could not be deleted	Essential data will be destroyed Data security will be lost.
US-5	The similar operations with different actor not clarified (Complaint of company and customer not separated)	The sequences of actions will be undetermined.
US-6	Date and time details of the frequently (like tracking) is not adequate	The operations do not have exact timeline.
US-7	The scenario of actors (for tracking- Customer and staff) are not different.	The actions of actors will be mixed up.

b) Sample causes and effects for defects in Design phase

Sample causes and effects of potential defects in class diagrams		
Cause Code	**Cause**	**Effect(s)**
CL-1	Defining the name of elements in general like 'data', 'record', or 'info'.	Weakness in specifications and adequate operations.
CL-2	Do not understand the meaning of superclass	Defining many subclass with same attributes and operations
Cl-3	Not focus on common elements of the classes	effect missing correct super classes an instance may need to change the class
CL-4	Defining a class instead of defining an attribute in a superclass	
CL-5	Defining an attribute in associate role class, as a class	Any change in an instance may be changes the class
CL-6	Mixed up aggregation and composition relationship	Destroy the part when not necessary and not destroy when must.
CL-7	Mixed up the public, private, and protected attributes	The access to attributes in subclass within class and outside class not correct
CL-8	Multiple inheritance	Not supported by the programming language and make the code not executable.
CL-9	All kinds of identification like 'id' or 'name' attribute, are not provided	Problem in access to with some key attributes