ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

ESTIMATION MODEL FOR SOFTWARE TESTING

BY
Jayakumar KAMALA RAMASUBRAMANI

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, 19 JULY, 2016.

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Prof. Alain Abran, Thesis Supervisor,
Department of Software and Information Technology Engineering,
École de technologie supérieure.

Prof. Claude Thibeult, Jury President,
Department of Electrical engineering,
École de technologie supérieure.

Prof. Abdel Gherbi, Member of the Jury,
Department of Software and IT Engineering,
École de technologie supérieure.

Prof. Hamid Mcheick, External Member of the Jury,
Université du Québec, Chicoutimi.

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON JULY 19, 2016

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENTS

I would like to first thank the management and administration of École de technologie supérieure, for admitting me as a Ph.D. student.

I am thankful to Professor Alain Abran, the co-inventor of COSMIC functional sizing method, for taking me as his student and guiding me throughout the research work. He never got tired of reviewing my work at various stages, put up with my questions and provided innumerable suggestions to progress towards completion of this work. Charles Symons, co-inventor of COSMIC and inventor of Mark II Function Point reinforced my conviction to choose the topic for research, to whom I remain thankful.

My mother Smt. Kamala and father Sri. Ramasubramani are the spirit behind my activities; it is their never ending love and blessings that continuously motivated me through the research work. Kalpana, my wife who is an integral part of my life, not only sacrificed her personal time and energy to make me complete this work, but also ensured that I was enthusiastic throughout, in spite of the various moments of ups and downs.

No doubt, the time I spent with my daughter Lakshna and son Amitesh during the period of research went down substantially when they as adolescents needed me the most. Their youthfulness brought in freshness to my life and work. My brother and sisters wanted to see me achieve my dream of acquiring Ph.D and supported in all possible ways.

I am thankful to the Investors and Board of Amitysoft Technologies who appreciated time spent on the research besides managing the affairs of the company. I would like to acknowledge my colleagues at Amitysoft, especially Srikanth Aravamudhan who served as a sounding board to discuss various ideas related to this research, and Ganapathi who shared part of my workload so that I could spare time for this research. I am thankful to my friends Sriram who helped me with statistical analyses and Susan who extended a professional hand in editing this document.

Without the connect with the Almighty, it would not have been possible for me to reach this stage and I am indebted to Him for ever.

MODÈLE D'ESTIMATION POUR LESTESTS DE LOGICIELS

Jayakumar KAMALA RAMASUBRAMANI

SOMMAIRE

Tester les applications logicielles et en assurer la conformité sont devenus une partie essentielle de la gouvernance des organisations en technologies de l'information (TI). Les tests du logiciel ont évolué vers une spécialisation avec leurs propres pratiques et connaissances.

L'estimation des tests consiste en l'estimation de l'effort pour un niveau particulier de tests, en utilisant diverses méthodes, outils et techniques. Une estimation incorrecte conduit souvent à une quantité inadéquate de tests qui, à son tour, peut conduire à des défaillances des systèmes logiciels quand ils sont déployés dans les organisations.

A partir d'un état de l'art sur l'estimation des tests de logiciel, un cadre unifié pour l'estimation des tests de logiciels a été proposé. Grâce à ce cadre, divers modèles d'estimation détaillés ont été construits pour les tests fonctionnels.

La base de données ISBSG a été utilisée pour explorer l'estimation des tests de logiciels. L'analyse des données ISBSG a révélé trois schémas de productivité de tests représentant les économies d'échelle sur la base desquelles ont été étudiées les caractéristiques des projets correspondants. Les trois groupes de projets liés aux trois modèles de productivité sont jugés statistiquement significatifs et caractérisés par domaine d'application, la taille de l'équipe, la durée de projet et la rigueur de la vérification et la validation effectuée au cours du développement.

Au sein de chaque groupe de projets, les variations dans les efforts de test peuvent être expliquées par les activités menées au cours du processus de développement et adoptées pour les tests, en plus de la taille fonctionnelle. Deux nouvelles variables indépendantes, la qualité des processus de développement (DevQ) et la qualité des processus de test (TestQ) ,ont été identifiées comme influentes dans les modèles d'estimation.

Des portfolios de modèles d'estimation ont été construits pour différents ensembles de données en utilisant des combinaisons des trois variables indépendantes. Au moment de l'estimation, un estimateur peut choisir le groupe de projets par la cartographie des caractéristiques du projet à estimer et en les comparant aux attributs du groupe de projets afin de choisir le modèle le plus proche.

La qualité de chacun des modèles a été évaluée selon les critères établis tels que R^2, R^2 Adj, MRE, MedMRE, Maslow's Cp. Les modèles ont été comparés à l'aide de leur performance prédictive en utilisant les nouveaux critères proposés dans ce travail de recherche. De plus, les modèles d'estimation de test à l'aide de la taille fonctionnelle mesurée en points de fonction COSMIC présentaient une meilleure qualité et ont abouti à une estimation plus précise par rapport à la taille fonctionnelle mesurée en points de fonction IFPUG.

Un prototype de logiciel a été développé en utilisant un langage statistique "R" de programmation intégrant les portefeuilles de modèles d'estimation. Cet outil d'estimation de test peut être utilisé par l'industrie et le milieu universitaire pour estimer les efforts de test.

Mots clés: génie logiciel, mesures de logiciels, la taille fonctionnelle, tests de logiciel, Test fonctionnel, Modèle d'estimation, estimation des tests, points de fonction COSMIC, points de fonction IFPUG, outil pour l’estimation des tests étalonnage.

ESTIMATION MODEL FOR SOFTWARE TESTING

Jayakumar KAMALA RAMASUBRAMANI

ABSTRACT

Testing of software applications and assurance of compliance have become an essential part of Information Technology (IT) governance of organizations. Over the years, software testing has evolved into a specialization with its own practices and body of knowledge.

Test estimation consists of the estimation of effort and working out the cost for a particular level of testing, using various methods, tools, and techniques. An incorrect estimation often leads to inadequate amount of testing which, in turn, can lead to failures of software systems when they are deployed in organizations

This research work has first established the state of the art of software test estimation, followed by the proposal of a Unified Framework for Software Test Estimation. Using this framework, a number of detailed estimation models have been designed next for functional testing.

The ISBSG database has been used to investigate the estimation of software testing. The analysis of the ISBSG data has revealed three test productivity patterns representing economies and diseconomies of scale, based on which the characteristics of the corresponding projects were investigated. The three project groups related to the three productivity patterns were found to be statistically significant, and characterised by application domain, team size, elapsed time, and rigour of verification and validation throughout development.

Within each project group, the variations in test efforts could be explained by the activities carried out during the development and processes adopted for testing, in addition to functional size. Two new independent variables, the quality of the development processes (DevQ) and the quality of testing processes (TestQ), were identified as influential in the estimation models.

Portfolios of estimation models were built for different data sets using combinations of the three independent variables. At estimation time, an estimator could choose the project group by mapping the characteristics of the project to be estimated to the attributes of the project group, in order to choose the model closest to it.

The quality of each model has been evaluated using established criteria such as R^2, Adj R^2, MRE, MedMRE and Maslow's Cp. Models have been compared using their predictive performance, adopting new criteria proposed in this research work. Test estimation models using functional size measured in COSMIC Function Points have exhibited better quality and resulted in more accurate estimation, compared to functional size measured in IFPUG Function Points.

A prototype software is now developed using statistical "R" programming language, incorporating portfolios of estimation models. This test estimation tool can be used by industry and academia for estimating test efforts.

Key Words: Software Engineering, Software Measurements, Functional Size, Software Testing, Functional Testing, Estimation Model, Test Estimation, COSMIC Function Point, IFPUG Function Point, Test Estimation Tool, Benchmarking.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABREVIATIONS

CMMI	Capability Maturity Model Integration
COSMIC	Common Software Measurements International Consortium
DevQ	Development Process Quality Rating
ERP	Enterprise Resource Planning
FSM	Functional Size Method
FUR	Functional User Requirements
KLOC	Kilo Lines of Code
NFR	Non Functional Requirements
IFPUG	International Function Point Users Group
ISBSG	International Software Benchmarking and Standards Group
OCL	Object Constraint Language
PG	Project Group
PSP	Personal Software Process
SPICE	Software Process Improvement and Capability Determination
SWEBOK	Software Engineering Body of Knowledge
TDR	Test Delivery Rate
TestQ	Test Process Quality Rating
TTCN-3	Testing and Test Control Notation
UML	Unified Modelling Language

INTRODUCTION

0.1 Context of Software Testing and Estimation

Software Testing, as defined by the Guide to the SWEBOK (Bourque et al., 2014), consists of the 'dynamic' verification of the behaviour of a program on a 'finite' set of test cases. These test cases are suitably selected from the usually 'infinite execution domain,' against 'expected behaviour'.

Testing is also part of software maintenance and operations, besides development. Testing activities have to be managed effectively and quantitatively, in order to meet the intended purpose.

Software testing is quite challenging, technically and managerially, for the following reasons:

1. Software testing is carried out against Functional User Requirements (FUR), where all the operational scenarios cannot be identified due to the complexity barrier. Expected behaviors must be tested in an infinite execution space (Beizer, 2007).
2. There is a lack of consistency in the factors to be considered for Non-functional Requirements (NFR) (COSMIC, 2015). It is challenging to plan for testing with incompletely described NFR because it increases the odds that the NFR testing process will be incomplete.
3. There is no scientific approach to estimating efforts for all the aspects of software testing. The existing estimation approaches, such as judgment based, factors and ratings-based methods and functional size-based methods are characterized by several limitations (Refer Section 2.2).

0.2 Processes and Test Types

The International Standard ISO 29119 on testing (ISO/IEC/IEEE 29119 – Part 1, 2013; ISO/IEC/IEEE 29119 – Part 2, 2013; ISO/IEC/IEEE 29119 – Part 3, 2013; ISO/IEC/IEEE 29119 – Part 4, 2015) subdivides the test process into Project Test Process and Test Sub-Process, as follows:

a. Component, Integration, System and Acceptance Testing are different levels/ phases of the Project Test Process.
b. Performance Testing, Security Testing and Functional Testing are different types of testing referred to as Test Sub-Processes.

The Project Test Process involves and Dynamic Test Processes and Test Management Processes:

a. The Dynamic test process consists of test design, preparing test scripts and test data, setting up test environment, executing tests and reporting test results.
b. Management processes involve planning, monitoring and control of testing activities under dynamic testing process.

This ISO 29119 Standard provides a framework to identify specific activities that would constitute the scope of testing projects. Tasks to be undertaken for testing and estimate of effort for executing those tasks can be derived based on the scope.

0.3 Early Perspectives on Testing and Effort Estimation

One of the early books on Software Testing, 'Software Testing Techniques' by Borris Beizer (Beizer, 2007), documents the goals for testing. Testing and designing of tests, as parts of quality assurance, should also focus on bug prevention apart from discovering bugs. Beizer makes a practical suggestion:

> *"The ideal test activity would be so successful at bug prevention, that actual testing would be unnecessary. Unfortunately, we can't achieve this ideal. Despite our effort there will be bugs because we are human. To the extent that testing fails to reach its primary goal, bug prevention, it must reach its secondary goal, bug discovery."*

Beizer refers to a 'Complexity Barrier' where software complexities grow to the limits of our ability to manage the complexity. In the business context, where time-to-market or on-time delivery with assured quality is a most important success criterion, demands on software testing are two-fold:

1. Test everything that carries risks of failure.
2. Minimize efforts required for testing.

The two-component testing strategy advocated by Willam Perry (Perry, 2006), another pioneer in software testing, consists of:

1. 'Test Factors' that need to be addressed to reduce the risk.
2. 'Test Phase in the Life Cycle' in which the test factors are to be considered.

Test factors are related to a number of generic functional requirements, non-functional requirements and technical / quality requirements. Perry has suggested a parametric model to estimate staff-hours using:

1. The functional size, measured in terms of function points or lines of code, further adjusted considering project specific characteristics.
2. The total effort estimate is distributed next to project phases, based on the 'percentage efforts distribution' norm established using historical data.

To meet the conflicting demands on software testing, testers must be able to use effective estimation techniques. Managing software testing without quantitative measures increases the odds of failure. For example, cost and time overruns potentially result in financial and legal consequences. Estimation techniques should be based on a sound mathematical basis, and verifiable to the satisfaction of all stakeholders (Abran, 2015).

0.4 Implications for the Industry & Economy

0.4.1 Impacts of Testing and Software Defects

World Quality Report 2015 – 16 (Capgemini, 2016) observes that 'quality assurance and testing' have failed to keep up with business needs; this is inferred from the IT spend on these activities outstripping predictions every year. This report estimates that the proportion of the IT spend allocated to ensuring application quality and performance will continue to rise, from the current figure of 35%, to reach 40% by 2018. Gartner's forecast on enterprise IT spending

across all industry market segments in 2016 is US $ 2.77 trillion (Gartner, 2015). These two figures from Capgemini and Gartner indicate the amount of budget involved in quality and testing of IT solutions.

There are several instances of software failures that have resulted in major economic impacts, including loss of human lives. A study by the National Institute of Standards and Technology - USA (Tassey, 2002) estimated the losses to the US economy alone at close to US$ 60 billion, due to defects in software applications in the Manufacturing and Service sectors. Some of the incidents, reported due to defects in Enterprise Software applications, together with their cost impact, are listed in Table 0.1 for the 2011-2012 reporting period. This table lists details such as the names of the organizations, the related defective software and the direct cost impact.

Table 0.1 Cost Impacts of Defective Software Reported in 2011 – 12

#	Company	Defective Software	Cost (mil)	Source
1	RBS	C7A batch Process	£125	http://www.computerweekly.com/news/2240160860/RBS-computer-problem-costs-125m
2	AXA Rosenburg Group	Portfolio management	$242	http://www.advfn.com/commodities/CommoditiesNews.asp?article=46297248&headline=axa-rosenberg-to-pay-242-million-over-software-glitch
3	Knight's Capital	Automatic market orders	$440	http://www.bloomberg.com/bw/articles/2012-08-02/knight-shows-how-to-lose-440-million-in-30-minutes
4	Telecom Company, New Zealand	Customer Billing	$2.7	www.comcom.govt.nz/dmsdocument/10828

There are several incidents related to defects and consequent software failures, which are periodically reported by the press. However, the economic impacts are not disclosed for most

the cases. Management processes are critical for planning and delivering testing projects successfully. It is estimated that the annual cost of poor performance of software suppliers in North America and in Europe is over US$ 200 billion (Symons, 2010). Comparing this to the approximately US$ 200 billion of losses faced by banks in the 2008 credit crisis which was a one-time event, it is apparent that the recurring annual losses due to poor performance in the software industry is multi-fold.

The poor performance of software in this context refers to cost overruns, including cancelled projects or projects finished but not deployed. A causal chain links poor performance to measurements and estimation practices (Symons, 2010). The Project Management Institute (PMI) has identified 'Estimation' as a key area in the Project Management Body of Knowledge (PMI, 2013) for successful delivery of projects.

In order to tackle performance and poor management issues, the state of Victoria in Australia has designed and implemented the 'SouthernScope' project contracting methodology (SouthernSCOPE, 2012) which resulted in less than 10% cost overrun after implementing functional size-based estimation and costing of projects. This can be compared to 84% cost overrun that prevailed when traditional methods were used (Symons, 2010).

0.4.2 Potential of a Better Estimation Model for Testing

Software projects go through the 'testing phase', which is a key phase for controlling defects within the overall development life cycle. Inadequate testing leaves defects in the software used in production, leading to failures and consequent financial impact.

A better testing process can contribute to the reduction in the number of defects. The US Study (Tassey, 2002), points out that improvements in the testing infrastructure could result in US$22 billion saving by reducing the defects by 50%.

Improved estimation techniques can aid better budgeting and resourcing of software testing. Performance measurements using International benchmarks can enable the organizations improve their test processes to become more competitive.

0.5 Motivation for the Research

Software testing has evolved into a specialization with its own practices and body of knowledge (ISTQB, 2011; QAI, 2006). Over the years, software testing has become an industry of its own, with the emergence of independent testing services firms and IT services companies in India (such as Cognizant, TCS, Accenture) establishing testing services as a separate business unit.

Test estimation consists of the estimation of effort and cost for a particular level of testing, using various methods, tools, and techniques. Test estimation techniques have often been derived from generic software development estimation techniques (Chemutri, 2012), in which testing figures as one of the phases of the software development life cycle. The incorrect estimation of testing effort often leads to an inadequate amount of testing that, in turn, can lead to failures of software systems when they are deployed in organizations. There are no international benchmarks available to verify test effort estimates.

Existing estimation techniques such as judgment based, test estimation specific methods and functional size methods used for estimating test efforts are hampered by several arbitrary factors and they lack of compliance to fundamentals of metrology (Abran, 2010).

The functional size of software is found to be a significant influencer in the estimation of development effort (Abran, 2015). Even those techniques that use functional size for estimation models do not take into consideration the mathematically correct functional size as a parameter (Refer Section 2.2).

There is a growing amount of work carried out on the use of the functional sizing method COSMIC – ISO 19761 (COSMIC, 2003; ISO, 2011), for estimation and performance measurements of software development projects. There are several complex approaches taken by the researchers for estimation, while simple and practical approaches to estimation (Abran, 2015) governed by metrology principles can fulfill the needs in industry, academia and research.

The motivation for the research arises from:

a. Difficulties in estimating the effort for software testing.
b. Opportunities emerging out of recent development in functional sizing methods.
c. Lean approaches towards building estimation models.

The research strategy will involve adapting innovations from related disciplines, to come out with practical estimation models for software testing. These estimation models will be designed to substantially overcome the limitations of existing estimation techniques. Additionally, they would comply with metrology, be simple to understand and use by the industry and academia.

0.6 Organization of the Thesis

Chapter 1 presents the research goal, objectives and the methodology adopted. This chapter also details out the research disciplines involved and the systematic steps followed as per the methodology.

Chapter 2 provides an evaluation of existing test estimation techniques followed by detailed discussions of each of the categories, analysis of their strengths and weaknesses followed by comments on experimental studies. A summary of the literature study with limitations of existing techniques is part of this chapter.

Chapter 3 proposes an 'Unified Framework for Software Test Estimation' mapped to relevant ISO standards along with proposals for measures and approaches for test estimation models for software testing.

Chapter 4 details out the estimation model for functional testing, consisting of details of data selection and analysis, design of portfolio of estimation models, their evaluation and comparison of predictive performance. This chapter also documents the design of a prototype tool to automate the estimation model for functional testing.

How this research work meets the objectives set initially, contributions arising out this work, limitations and potential for future work are presented as a part of the Conclusion chapter.

CHAPTER 1

RESEARCH GOAL, OBJECTIVES AND METHDOLOGY

1.1 Research Goal

The long term goal of this research project is to develop a practical solution for estimating software testing effort, for all types of testing. The immediate goal is to focus on designing estimation model for estimating the effort for functional testing.

1.2 Research Objectives

The objectives selected for this research project are to design an estimation model to:

a. Estimate the effort for functional testing. This comprises:
   - Identifying the 1 to 3 factors that contribute most to the relationship with effort for functional testing.
   - Arriving at a model that can be used during the early stages of software testing.

b. Serve the needs for benchmarking and measuring the performance of software testing projects.

c. Be capable of automation, which can be deployed as an estimator's tool for use by industry and academia.

1.3 Research Approach

1.3.1 Research Disciplines

The research approach selected to address the problem involves combining knowledge from four disciplines (Figure 1.1).

a. Software Testing

b. Software Engineering

c. Metrology

d. Statistics

The contexts of each of these disciplines in this research work are presented next.

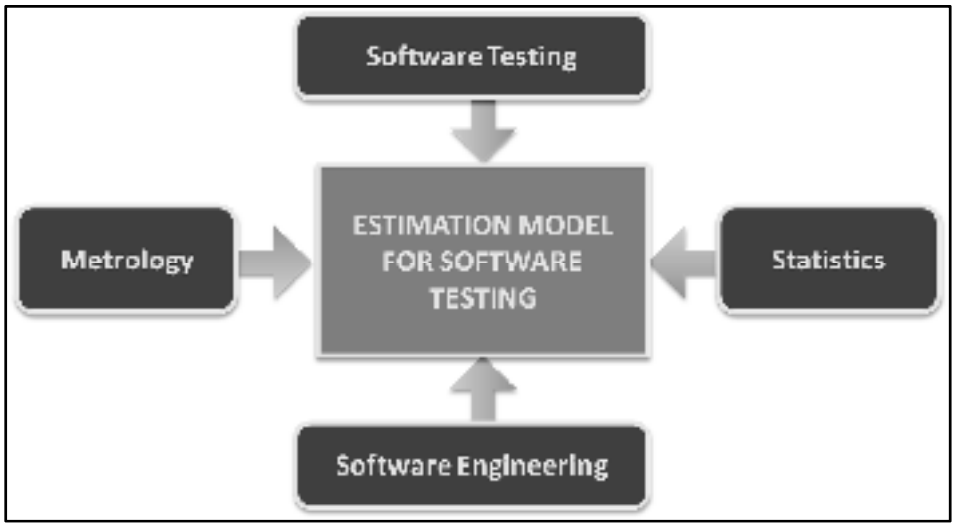


Figure 1.1 Research Disciplines

1.3.1.1 Software Testing

Software Testing has evolved into a discipline along with the development of software development methodologies: its evolution has passed through the following phases (Beizer, 2007):

- Phase 0: There is no difference between testing and debugging. Other than in support of debugging, testing has no purpose.
- Phase 1: The purpose of testing is to show that the software works.
- Phase 2: The purpose of testing is to show that the software does not work.
- Phase 3: The purpose of testing is not to prove anything, but to reduce the perceived risk of not working to an acceptable value.
- Phase 4: Testing is not an act. It is a mental discipline that results in low-risk software without much testing effort.

The two key words in Phase 4 of the evolution of software testing are ‘low-risk’ and ‘testing effort’. Analysing the product risks and designing and executing tests in such a way that effort for testing is minimal are the key characteristics of current phase. Estimating the effort required for testing and measuring performance of testing can provide a quantitative basis for managing testing projects.

1.3.1.2 Software Engineering

Software Testing is a part of software engineering. The 'Guide to the SWEBOK' (Bourque et al., 2014) identifies software testing as one of the knowledge areas related to software engineering (Table 1.1).

Table 1.1 The Knowledge Areas in the SWEBOK Body of Knowledge

No.	Knowledge Area	No.	Knowledge Area
1	Software Requirements	9	Software Engineering Models and Methods
2	Software Design	10	Software Quality
3	Software Construction	11	Software Engineering Professional Practice
4	Software Testing	12	Software Engineering Economics
5	Software Maintenance	13	Computing Foundations
6	Software Configuration Management	14	Mathematical Foundations
7	Software Engineering Management	15	Engineering Foundations
8	Software Engineering Process	-	--

Estimation of efforts for software testing cannot be looked into isolation, without considering software life cycle aspects and measurements related to various aspects of software engineering. While software testing is the key knowledge area relevant to this research work, other knowledge areas such as Software Engineering Management, Software Engineering Process, Software Engineering Models and Methods, Software Quality, Software Engineering Professional Practice, Software Engineering Economics and Mathematical Foundations contribute to this research work. This research work explored the literature on software engineering, in order to establish the state of the art of software test effort estimation.

A major problem in software engineering is the passing on of some trivial rumour from one person to the next, until it has become distorted and blown out of all proportion. The outcome is that it becomes entrenched as ‘fact’, claimed to be supported by ‘figures’, and attains an elevated status despite being merely anecdotal (Bossavit, 2015). Appealing to authority overlies such so-called ‘facts’ and ‘figures’; this is the key guideline for this research work, in order to differentiate the ‘feel good’ aspects from the ‘feel right’ approach to developing estimation models.

1.3.1.3 Metrology

Metrology deals with rigorous definitions of measurement standards and their instrumentation. This discipline helps to tackle the disparity in the units of measurements, in support to various derived measures and models.

Metrology related concepts from the ISO Vocabulary on Metrology (VIM, 2007) have been adopted as the basis for measurement terminology for future ISO standards on software measurement. Information technology and computer science have not been subjected to the metrological scrutiny that other sciences have (Gray, 1999; Kaner, 2013). According to the principles of metrology, the term measurement has to be used in the context of ‘measurement method’, ‘application of a measurement method’ or ‘measurement results’. They correspond to three steps (Figure 1.2) in the measurement context as illustrated in the Measurement Context Model from (Abran, 2010):

a. Design of the measurement method: before measuring, it is necessary to either select a measurement method if one already exists, or design one if an existing method does not fit the need.
b. Application of the measurement method: once the measurement method has been designed or selected, its rules are applied to a piece of software to obtain a specific measurement result.
c. Exploitation of the measurement results: the measurement result is exploited in a quantitative or qualitative model, usually in combination with other measurement results of different types.

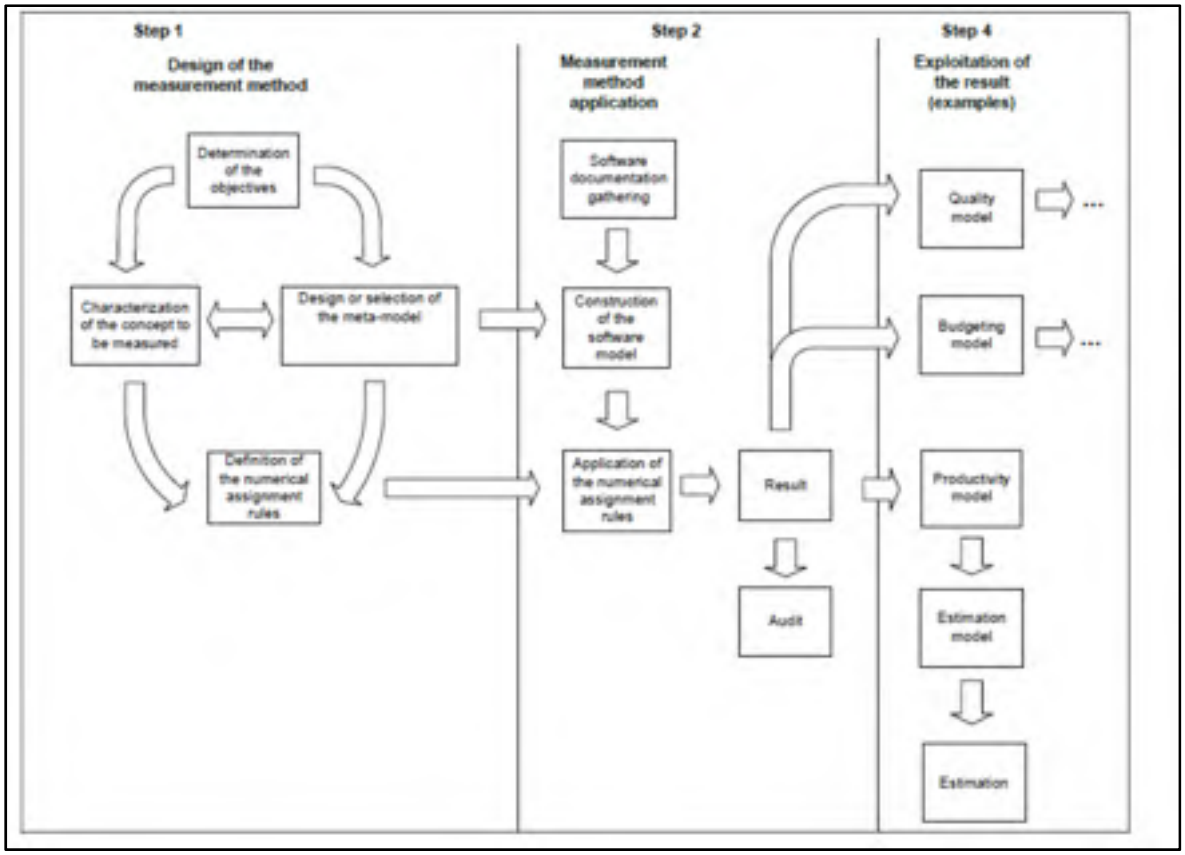


Figure 1.2 Measurement Context Model
(Taken from Abran, (2010))

While metrology is mature in other disciplines, it is yet to become a norm in software engineering. This has resulted in several flaws (Abran, 2010) in the existing 'software metrics' used in software engineering.

This research project ensures the application of measurement principles to arrive at soundly-structured estimation models.

1.3.1.4 Statistics

Statistics is broadly divided into two branches – descriptive and inferential statistics (Levine, 2013):

a. Descriptive statistics deal with quantitative data and the methods for describing them. This is the most familiar branch of statistics used in everyday life, such as in social services, business, health care and sports. For example, measures of central tendency and measures of spread are used to describe data.
b. Inferential statistics make inferences about populations by analyzing data gathered from samples and lead to conclusion from these data. Methods of inferential statistics include testing of hypotheses and estimation of parameters.

This research project uses various data sets to build test estimation models, and hence the correct application of principles and practices of statistics is of paramount importance. Even though basic statistical concepts, such as measures of central tendency, are a foundation to any analysis, there are several statistical techniques available in the process of building an estimation model.

In large data sets, wider deviations are mostly attributable to noise than to information (Taleb, 2012). Hence, if complex models are built, their relevance may become questionable and usage may become difficult. Nassim Nicholas Taleb, noted author of ‘Antifragile’ proposes antifragile models for informed decisions rather than fragile and robust models. The fragile and robust models collapse faster, while antifragile systems can change and evolve.

This research project has come out with portfolios of estimation models for different contexts, rather than attempting one single robust and complex model that can only provide an illusion of stability. It has been observed that, more the small variations in the system, the fewer would be the major surprises (Taleb, 2012). These insights form the backbone of the work carried out in this research.

1.3.2 Research Methodology

The methodology adopted for carrying out this work consists of the following five research phases:

1. Literature study
2. Designing Unified Framework for Test Estimation
3. Designing Estimation Models for Functional Testing

4. Evaluating the Estimation Models
5. Developing a Prototype Estimation Tool

Each phase is briefly described next.

1.3.2.1 Literature Study

A literature study has been conducted to understand the techniques and practices in the estimation for software testing. The study covered estimation techniques used in both industry and academia. The study reviewed the basic approaches to test estimation, including estimation techniques for functional and non-functional testing. Other estimation approaches such as neural network and case based reasoning have also been examined.

Besides, the literature study has also reviewed model-driven testing, agile testing, service oriented architecture and test automation from the perspective of estimation; the presents the state of the art on software test estimation techniques, their strengths and weaknesses. The inputs for this phase, the various steps performed and the outputs from this phase are listed in Table 1.2.

Table 1.2 Inputs, Steps and Outputs of Phase 1

Inputs	**Steps**	**Outputs**
SWEBOK Body of Knowledge ISO 29119 : Software and Systems Engineering – Software Testing	• Study of basic approaches to test estimation • Estimation techniques for Functional Testing • Estimation for Non-functional Testing • Fuzzy, Neural Network and Case-Based Reasoning for Estimation	• Grouping test estimation techniques into categories . . • Evaluation criteria for estimation techniques

Inputs	Steps	Outputs
ISO 19761 : COSMIC Functional Sizing Method	• Current key developments in Testing • Review of evaluation criteria for estimation techniques	

1.3.2.2 Designing an Unified Framework for Test Estimation

This phase comes out with a Unified Framework for test estimation, based on the learnings from the Literature Study.

The first step for designing the Unified Framework consists of characterizing the various facets of functional and non-functional testing, based on ISO Standards. This results in a qualitative model.

The next step transforms this model into a quantitative model by identifying relevant measures. This quantitative view of the Unified Framework provides a basis for building estimation models for testing. Table 1.3 presents the inputs considered for this phase, the steps performed during the phase and the outputs from this phase.

Table 1.3 Inputs, Steps and Outputs of Phase 2

Inputs	Steps	Outputs
• ISO 25010 on quality characteristics of software • ISO 29119: Software and Systems Engineering – Software Testing.	• Characterizing various facets of Functional and Non-functional Testing • Identification of measures related to individual aspects of Functional and Non-functional Testing	• Unified Framework for various facets of functional and non-functional testing with qualitative characteristics,

Inputs	Steps	Outputs
• ISO 19759: SWEBOK Guide v3 • Software Metrics and Software Metrology book	• Developing approaches to building estimation models	• Quantitative measures and approaches to estimation.

1.3.2.3 Designing Estimation Models for Functional Testing

The Functional Testing component has been taken up from the Unified Framework, for designing detailed estimation models for functional testing.

Release 12 (2013) of ISBSG database serves as the data source for building the testing estimation model. The following steps are performed during this phase 3:

a. Data preparation from ISBSG by applying relevant filters, to come out with high quality data representing testing.
b. Selection of four different data sets, consisting of relevant samples from a statistical point of view.
c. Data analysis using statistical analysis tools.
d. Identification of relevant variables for project contexts.
e. Building a portfolio of context specific estimation models.
f. Model selection approach for estimation user.

Table 1.4 provides the inputs considered for this phase 3, the various steps performed during the phase and the outputs from the phase.

Table 1.4 Inputs, Steps and Outputs of Phase 3

Inputs	Steps	Outputs
• Literature Study • Unified Framework for Functional Testing • ISBSG Release 12 Data repository	• Observation of project data & data preparation • Selection of relevant samples and ensuring adequacy of samples from a statistical point of view. • Data analysis using statistical analysis tools • Identification of relevant variables for project contexts. • Building context specific estimation models.	• Portfolio of Estimation Models based on initial data set • Portfolio of Estimation Models based on a subset of initial data set • Portfolio of Estimation Models for COSMIC Functional Size. • Portfolio of Estimation Models for IFPUG Functional Size. • Approach for selection of a model

1.3.2.4 Evaluating the Estimation Models

The quality of the estimation models for functional testing are evaluated using the criteria followed by the researchers. This phase also compares the performance of COSMIC-based estimation models versus IFPUG-based estimation models.

The inputs considered for this phase, the steps performed during the phase and the outputs are listed in Table 1.5. The conclusions are based on the performance of each of the model.

Table 1.5 Inputs, Steps and Outputs of Phase 4

Inputs	Steps	Outputs
• Portfolio of Estimation Models • Quality criteria for evaluation of Models • Criteria for performance of estimation models	• Computation of quality criteria for each of the model • Computation of model performance for each of the model • Comparison of quality of estimation models • Developing criterion for comparison of performance of models & its application	• Quality of estimation models. • Performance of estimation models. • Comparison of models.

1.3.2.5 Developing a Prototype Estimation Tool

A prototype tool will be developed, based on the estimation models. The tool will allow the estimation user to choose a project context and, based on the functional size, will provide an estimate for the test effort.

The design of the tool takes into consideration refinements to the estimation model, based on the availability of organization specific project data and regeneration of models, based on the availability of a larger multi-organizational data set.

The basic purpose of the development of this tool is to confirm the possibility of automation of the estimation using the models generated.

CHAPTER 2

LITERATURE STUDY

2.1 Evaluation of Test Estimation Techniques

2.1.1 Categories of Techniques

The test estimation techniques studied can be categorized into the following five groups:

1. Judgment and rules of thumb.
2. Analogy and work breakdown.
3. Factors and weights.
4. Size based estimation models.
5. Fuzzy, Neural and Case based models.

Approaches towards estimation adopted by these techniques can be broadly classified as formula oriented and model oriented while some of the techniques combine both. Several of these techniques identify variables relating to the project and come out with a formula to provide an estimate. They incorporate various heuristics based on the experience of the person proposing the technique. There are no established criteria to evaluate such formulae. Other techniques use those variables to build a statistical model for estimation based on the relationship between the independent variables and the dependent variable. An a-posteriori estimation model representing testing process is built with data from completed projects. The models are subsequently used to estimate new projects. These models can be evaluated using recognized criteria (see section 2.1.2).

New criterion has been proposed to evaluate existing categories of estimation techniques (see 2.1.3). A brief review of several techniques falling into each of the above groups is presented later in section 2.2.

2.1.2 Model Evaluation Criteria

Estimation models are built using past data for prediction in future, and so they are to be evaluated for fitness for the purpose. The criteria used for evaluating the estimation models (Conte, 1986) are:

a. Coefficient of determination (R^2)
b. Adjusted R^2 (Adj R^2)
c. Mean Magnitude of Relative Error (MMRE)
d. Median Magnitude of Relative Error (MedMRE)

The **coefficient of determination (R^2)** describes the percentage of variability explained by the independent variable(s). This coefficient has a value between 0 and 1. A value of R^2 close to 1 indicates that the variability in the response to the independent variable can be explained by the model and hence there is a strong relationship between the independent and dependent variables. A R^2 close to 0 indicates that the variability in the response to the independent variable cannot be explained by the model and hence there is a no relationship between the independent and dependent variables. The equation 2.1 show how the adjusted R^2 is computed.

$$R^2 = 1 - \frac{SS_{residuals}}{SS_{total}} \quad (2.1)$$

where $SS_{residuals} = \sum_i^n (\hat{y}_i - \bar{y})^2$ and $SS_{total} = \sum_i^n (y_i - \bar{y})^2$ y_i

y_i is original data value, $\hat{y}_i$ is predicted value, n is number of samples.

Adjusted R^2 (equation 2.2) is an improvement over the R^2 in revealing explanatory power of models when there are more than one independent variables used in the model.

$$Adj\ R^2 = 1 - \frac{SS_{residuals}/(n-K)}{SS_{total}/(n-1)} \quad (2.2)$$

The sum-of-squares of the residuals from the regression line or curve have n-K degrees of freedom, where n is the number of data points and K is the number of parameters fit by the regression. As the Adj R^2 increases, the model becomes more desirable. When there are more than one variable used, the value of the adjusted R^2 is always lower than of the R^2.

$$MRE = |RE| = \left|\frac{Actual - Estimate}{Actual}\right| \quad (2.3)$$

$$MMRE = \overline{MRE} = \frac{1}{n}\sum_{i=1}^{n} MRE_i \quad (2.4)$$

MRE provides an indication of the divergence between the values estimated by the model and the actual values. MMRE (or $\overline{MRE}$) is the mean magnitude of relative error across all the data points. The MMRE does not represent the extreme of the estimate errors and only the mean. There will be estimates which would be much closer to actuals as well as estimates which are quite higher compared to the mean.

Median MRE (MedMRE), calculated from Median value instead of Mean value analogous to equations 2.3 and 2.4, can provide a better indication of the error in such cases.

Another criterion, referred to as Mallow's C_p, is used (Lindsey, 2010) for evaluating linear regression models along with the Adjusted R^2.

$$C_p = (n - m - 1)\frac{RSS}{RSS_{FULL}} - (n - 2p) \quad (2.5)$$

Where $RSS\ is\ the\ SS_{residuals}$ for the model with *p* regression coefficients and RSS_{FULL} is the $SS_{residuals}$ for the full model with *m* possible predictors excluding intercept.

Mallow's Cp helps to strike an important balance with the number of predictors in the model. It compares the precision and bias of the full model to models with a subset of the predictors. Models where Mallow's Cp is small and close to the number of predictors in the model plus the constant (p) are usually preferred.

A small Mallow's Cp value indicates:

a. that the model is relatively precise (i.e., has a small variance) in estimating the true regression coefficients and predicting future responses;
b. that the model is relatively unbiased in estimating the true regression coefficients and predicting future responses.

Models with bias or improper fit will have a Mallow's Cp value larger than the number of independent variables plus a constant. Mallow's Cp values are computed with p = 1 for each of the models.
These criteria will be used to evaluate the portfolio of estimation models designed later in this research work.

2.1.3 Criteria for Evaluation of Test Estimation Techniques

In order to evaluate the test estimation techniques identified in the literature study, I propose the following criteria:

Customer view of requirements: This criterion makes it possible to determine whether the estimation technique looks at the software requirements from a customer viewpoint or from the technical/implementation viewpoint. Estimation based on the customer viewpoint provides an opportunity for customer to directly relate estimates to the requirements.

Functional size as a prerequisite to estimation: Most estimation methods use some form of size, which is either implicit or explicit in effort estimation. When size is not explicit, benchmarking and performance studies across projects and organizations are not possible. Functional size can be measured using either international standards or locally defined sizing techniques.

Mathematical validity: Several of the estimation techniques discussed in Section 2.2 have evolved over the years, mostly based on a 'feel good' approach and ignoring the validity of their mathematical foundations. This criterion looks at the metrological foundation of the proposed estimation techniques and application of statistical criteria to assess the quality of the estimation models. A valid mathematical foundation provides a sound basis for further improvements.

Verifiability: The estimate produced must be verifiable by a person other than the estimator. Verifiability makes the estimate more dependable.

Benchmarking: It is essential that estimates be comparable across organizations, as this can help later in benchmarking and verifying performance improvement. The genesis of the estimation techniques is looked at to determine whether or not benchmarking is feasible.

Using these five criteria, Table 2.1 presents summary of my high level analysis of each category of techniques for the estimation of software testing. The details supporting this analysis are presented in the following sub-sections.

Table 2.1 Evaluation of Test Estimation Techniques

Criteria → / Estimation techniques ↓	Customer view of requirements	Functional size as a prerequisite	Mathematical validity	Verifiable	Benchmarking
1- Judgment & rule of thumb	NO	NO	Not applicable	NO	NO
2- Analogy & work breakdown	NO	NO	YES	YES	Partial, and only when standards are used
3- Factor & weight	NO	NO	NO – units are most often ignored	YES	NO
4- Size	YES	YES	Varies with sizing technique selected	YES	YES
5-Neural Network & Fuzzy logic models	Partially	Most often, No	YES, in general, but at times units are ignored	Partial	Partially, and only when standards are used

2.2 Test Estimation Techniques

2.2.1 Judgement and Rule of Thumb

A description of judgement and rule of thumb techniques is presented next.

Delphi (Chemuturi, 2012): A Delphi is a classic estimation technique in which experts are involved in determining individual estimate for a particular set of requirements based on their own earlier experience. Multiple iterations take place during which the experts learn the reasoning from other experts, and rework their estimate in subsequent iterations. The final estimate is selected from the narrowed range of values estimated by experts in the last iteration.

Rule of Thumb: The rule of thumb estimates are based on ratios and rules pre-established by individuals or by experienced estimators, but without a well-documented and independently variable basis. For example: the following rules of thumb (Jones 2007) are used to estimate efforts for certain activities involved in testing:

a. Function points raised to the 1.2/ 1.15/ 1.3 power will give an approximation of the average/ minimum/ maximum number of test cases.
b. Function points raised to the power 1.25 predict the approximate defect potential for new software projects.
c. Each software test step will find and remove 30 percent of the bugs that are present.

Strengths and Weakness

Strengths are as follows:

- Simple to use.
- Perception of quick results.

Weaknesses are as follows:

- Results cannot be verified by an independent person
- Estimation is not based on the analysis of well documented historical data and hence benchmarking is not feasible.
- Estimator often takes up an implementation view of the requirements.
- Some of the techniques do not provide estimates for all activities in testing.

Experimental Studies

No experimental studies are recorded to evaluate the effectiveness of these rules of thumb techniques.

2.2.2 Analogy and Work Breakdown Techniques

A description of analogy and work breakdown techniques is presented below:

Analogy-based (Chemuturi, 2012): The analogy-based techniques involve comparisons of the components of the software under test with a set of reference components, for which test effort is known based on historical data. The total estimate of all the components of the software to be tested is further adjusted based on project-specific factors and the management effort required, such as planning and review.

Task-based (Chemuturi, 2012): It is a typical work breakdown-based estimation technique where all testing tasks are listed and three-point estimates for each task are calculated with a combination of the Delphi Oracle and Three Point techniques (Black, 2002). One of the options offered by this method for arriving at an expected estimate for each task is a beta distribution formula. The individual estimates are then cumulated to come out with the total effort for all the tasks. Variations of these techniques, such as **Bottom-Up** and **Top-Down**, are based on how the tasks are identified.

Test Case Enumeration-based (Chemuturi, 2012): This is an estimation technique which starts with the identification of all the test cases to be executed. The estimate of the expected effort for testing each test case is calculated, using a beta distribution formula.

Strengths & Weaknesses

Strengths are as follows:

- These techniques can work in a local context within an organization, where similar types of projects are executed.
- Simple to use.

- Estimates can be verified by an independent person, if historical records are maintained.

Weaknesses are as follows:

- Benchmarking is not possible, since there is no agreed definition of what constitutes a task or work breakdown.
- Implementation view of the requirements is taken while estimating.

Experimental Studies

No known experimental studies on the effectiveness of these techniques for test effort estimation are reported.

2.2.3 Factors and Weights

A description of factors and weights techniques is presented below:

Test Point Analysis (Kerstner, 2011): It is a technique in which dynamic and static test points are calculated to arrive at a test point. Dynamic test points are calculated based on function points, functionality-dependent factors, and quality characteristics. Function-dependent factors, such as user importance, usage intensity, interfacing requirements, complexity, and uniformity are given a rating based on predefined ranges of values. Dynamic quality characteristics, such as suitability, security, usability, and efficiency, are rated between 0 and 6 to calculate dynamic test points. Each applicable quality characteristic as defined in ISO 9126 is assigned a value of 16 and summed to obtain the total number of static test points. The test point total is converted to effort based on ratings to be provided for a set of productivity and environmental factors.

Use Case Test Points: Use case test points (Kerstner, 2011) is proposed as an alternative to Test Points and derived from Use Case-based estimation for software development. Unadjusted Use Case Test Points are calculated as the sum of the actors multiplied by each actor's weight from an actors' weight table and the total number of use cases multiplied by a weight factor, which depends on the number of transactions or scenarios for each use case. Weights assigned to each of the technical and environmental factors are used to convert 'unadjusted use case points' to 'adjusted use case points'. A conversion factor accounting for technology/process language is used to convert adjusted use case points into test effort.

Test Execution Points (Aranha and Borba, 2007): This technique estimates test execution effort based on system test size. Each step of the test specifications is analyzed based on characteristics exercised by the test step, such as screen navigation, file manipulation, and network usage. Each characteristic that impacts test size and test execution is rated on an ordinal scale – low, average, and high – and execution points are assigned.

Test team efficiency is factored into another variation of the estimation model for test execution effort. The **Cognitive Information Complexity Measurement Model** (Silva, Abreua and Jino, 2009) uses the count of operators and identifiers in the source code coupled with McCabe's Cyclomatic Complexity measure.

Strengths & Weaknesses

Strengths are given below:

- Partial customer view of requirements.
- Consideration of various factors which are believed to impact test effort provides a feel-good factor to estimator.
- Estimates can be verified by an independent person.

Weaknesses are given below:

- Factors used to calculate are not of the same measurement scale.
- The measures used in this model lack the basic metrological foundations for quantification (Abran, 2010) and the validity of such measurements for estimating test execution effort has not been demonstrated.
- Formulae used to calculate contain invalid mathematical operations.

Experimental studies

There are no documented studies recorded on the effectiveness of these techniques.

2.2.4 Size-based Estimation Models

The size of the software in terms of its functional size measured using any functional size measurement method is used in this class of estimation techniques. The size, along with other relevant measures, is built into an estimation model which is used to estimate effort. Some techniques use size to build a regression model using historical data, while others do not follow a statistical approach in modelling. As individual techniques differ from each other in using a particular functional size measurement method and the approach used to building estimation model, they are dealt separately in the following subsections.

2.2.4.1 Test Size-based Estimation

A description of test size-based estimation techniques is presented below.

The size of the functional requirements in Function Points using IFPUG's Function Point Analysis (FPA) (IFPUG, 2005; ISO 20926, 2009) is converted to unadjusted test points through a conversion factor in estimation using Test Size (Chemuturi, 2012). Based on an assessment of the software, the programming language and the scope of the testing, weights from a weight table are assigned to test points. Unadjusted test points are modified using a composite weighting factor to arrive at a test point size. Next, test effort in person hours is computed by multiplying Test Point Size by a productivity factor.

Strengths & Weaknesses

The strengths are given below:

- Takes a customer view of the requirements during functional size measurement
- Estimates can be verified by an independent person
- Benchmarking is feasible as test effort is computed with reference to a size measured using FPA.

The weaknesses are given below:

- The technique uses FPA. Mathematical limitations of FPA have been analyzed and documented (Abran, 2010).
- Basis of weight and composite weight factor used to arrive at test point size is not known.

- No statistical approach used in determining the productivity factors.

Experimental Studies

There are no recorded experimental studies to assess the quality of test size based estimation.

2.2.4.2 AssessQ Model

A description of AssessQ Model is presented below:
The **AssessQ model** (Mutalik, 2003) and tool built by the founder of the first independent software testing organization in India, were used within his own organization prior to getting acquired by another IT services organization. The model was built based on internal experience of executing independent testing contracts. It uses IFPUG Function Points as its basic size measure which gets multiplied by factors based on the software engineering maturity of the development organization whose product is tested. This model uses past project data and provides estimates for (i) number of test cases to be designed and (ii) the number of expected defects. The estimates are adjusted to accommodate project specific factors.

Strengths & Weaknesses

The strengths are given below:

- Takes a customer view of the requirements during functional size measurement.
- Estimates can be verified by an independent person and benchmarking is feasible.
- Provides specific estimates for test case preparation and test case execution based on domain of the software tested.
- Recognizes ‘developer’ maturity of the development organization as a major factor affecting testing efforts. Process maturity of developer is assessed and used as one of the parameters in the model.
- Maturity of verification and validation process of the testing organization is used as another parameter in the estimation model, thus taking into consideration process aspects of testing organization.

- Statistical approach using historical data for building the estimation model.

The weaknesses are given below:

- The technique uses FPA. Mathematical limitations of FPA have been documented (Abran, 2010).
- The technique was internal to the organization. Technical details of the model, and the data used for building the model are not available for public review and validation.

Experimental Studies

No known experimental studies published on the effectiveness of the usage of this model.

2.2.4.3 Estimating test volume and effort

A description of the technique is presented below:

An approach for estimating the test volume and effort is proposed (Abran, Garbajosa and Cheikhi, 2007), where a functional size of requirements is used as a basis for quantitatively estimating test volume and used later in an effort estimation model. The initial estimate based on the functional requirements is adjusted subsequently by taking into consideration non-functional requirements. This technique uses the European Space Standards (ECSS, 2003) as a reference for functional and non-functional requirements. This model uses COSMIC Function Point (COSMIC, 2007; ISO/IEC 19761, 2011) to measure functional size. Estimates for non-functional testing are arrived at based on a graphical assessment of non-functional requirements of project data.

Strengths & Weakness

The strengths are given below:

- Uses COSMIC Function Point (ISO 1976, 2011) and overcomes the limitations of first generation of functional size measurement methods [Kamala Ramasubramani, 2011].
- Mathematically valid, verifiable and benchmarking is feasible.
- Provides an approach for accommodating non-functional requirements into estimation model.

The Weaknesses are given below:

- Combines all non-functional requirements together for estimation. Testing for non-functional requirements such as Security, Performance is often carried out separately by specialist teams, which requires separate estimate.
- Assumes all variations in efforts for particular functional size is due to non functional requirements only. There are several other cost drivers and project specific factors that affect test effort.

Experimental Studies

A case study using the February 2006 release of the ISBSG repository considering 292 new development projects data and 366 enhancement projects data, reports estimation models for functional testing with R^2 value of 0.31 and R^2 value of 0.20 for enhancement projects (Abran, Garbajosa and Cheikhi, 2007).

2.2.5 Neural Network and Fuzzy Models

2.2.5.1 Artificial Neural Network (ANN) Estimation Model

A description of the technique is presented below:

An Artificial Neural Network (ANN) is a model of the functioning of the human brain. ANN consists of several layers of neurons, each of which takes inputs from other neurons in the network and fires its outputs to other neurons, if the sum of its input connections rises above some specific 'threshold value'. A typical ANN configuration involves an input layer, an output layer, and one or more 'hidden layers'. There is an implementation of Artificial Neural Network (ANN) for software testing (Chintala et al., 2010) in which two effort estimation models are proposed:

- pre-coding model based on use case point, and
- post-coding model based on a number of variables, their occurrences, complexity of the code and criticalness of the code.

Strengths & Weaknesses

The strengths are given below:

- Pre-coding model takes up a customer view of the requirements.
- Estimates can be verified by an independent person.

The weaknesses are given below:

- Use case point with its mathematical limitations (Abran, 2010) made use of in pre-coding model.
- Post coding model uses factors such as number of variables, complexity and criticality of the code with arbitrary assignment of weights.
- Implementation view is taken up for estimation using post coding model.
- Benchmarking is not possible.

Experimental Studies

It is reported in (Chintala, 2010) that an experimental studies technique resulted in estimated effort deviating not more than 8% from actual in a few real time data from projects. However, the sample size of 4 projects used is much too small for statistical inference.

2.2.5.2 Fuzzy Logic Test Estimation Model

A description of the technique is presented below:

Fuzzy logic application to estimate software testing effort has been proposed in (Srivastava, 2009). This approach uses COCOMO (Boehm, 2000) as the basis in which KLOC is used as an input and development effort is calculated using ‘Effort Adjustment Factors’ based on ‘Cost Drivers’. Four testing specific cost drivers such as Software Complexity (SC), Software Quality (SQ), Schedule Pressure (SP) and Work Effort Driver (WFD) are used as inputs to fuzzy inference system that produces ‘test effort’ as output.

Strengths & Weaknesses

The strength is as follows:

- Deals with inputs which are uncertain - testing specific cost drivers inputs to fuzzy inference.

The weaknesses are given below:

- Testing Effort Drivers such as software complexity, software quality and work force drivers depend upon several other factors and they are not well defined.
- This technique uses COCOMO with its inherent mathematical limitations (Abran, 2010)
- Benchmarking is not possible.

Experimental Studies

No experimental studies with large data sets reported.

2.3 Other Literature of Interest on Test Estimation

2.3.1 Functional Testing

Scenario based black box testing using COSMIC (Abu Talib et al., 2006) is a method to optimise the test cases. COSMIC model of a scenario can lead to a test set consisting of test cases corresponding to the scenario. Test cases are partitioned into equivalent classes based on similarity and dissimilarity between test cases. A measure of functional complexity is proposed to prioritize test cases. Test cases with higher functional complexity are chosen for execution from the possible choices within an equivalent class. This approach aims to provide the best possible coverage with optimal use of resources.

Mapping of software scenarios to COSMIC model and using the model for preparation of test cases is a new approach. COSMIC model not only facilitates mapping business scenarios to a standard reference, but also provides a quantitative basis for measuring the scenarios that can be used for estimating.

Testing for the changes made to software and carrying out regression testing involves understanding the impact of the changes across the software. These impacts are not just related to ‘functional size of the change’ and there is no way of estimating for the changes using the scenario based black box testing approach.

2.3.2 Non-functional Testing

Non-functional requirements (NFR), in addition to the functional requirements, are quite critical for testing of the software: they can skew the efforts required for testing disproportionately to the size of functional requirements. The 'Guideline on Non-Functional and Project Requirements' (COSMIC, 2015) standardizes a glossary of terms associated with NFR. The most common types of testing carried out to test against NFR (Table 2.2) are identified as a part of types of testing in ISO 29119-4 (ISO/IEC/IEEE 29119 – Part 4, 2015).

Table 2.2 Common Types of NFR Testing

Type of NFR Testing	**Type of NFR Testing**
Accessibility Testing	Localization Testing
Backup/Recovery Testing	Maintainability Testing
Compatibility Testing	Performance Testing
Conversion Testing	Portability Testing
Disaster Recovery Testing	Procedural Manual Testing
Installability Testing	Reliability Testing
Interoperability Testing	Security Testing
Stability Testing	Usability Testing

NFR specified at a high level of granularity are often ignored during the entire development life cycle until they become an issue at the stage of acceptance or during operation of the software. Lack of details in NFR specifications adversely affects the test strategy. A set of reference models of NFR defined at different levels of details on the basis of various standards opens up a new vista to view NFR in terms of functional requirements, thus enabling size measurement. NFR such as System Maintenance, System Portability (Al Sarayreh, Abran and Cuadrado-Gallego, 2013), System Configuration and System Operational Requirements (Al Sarayreh, Abran and Cuadrado-Gallego, 2013) have been explored using this approach and mapped to functional requirements that can be allocated to software.

The standards-based framework for portability NFR provides for 4 function types and 11 portability functions when portability requirements are allocated to software and implemented as functional requirements. Once the relevant portability functions are specified, their size can be measured using COSMIC FSM method, as the portability functions are 'functional' in nature.

This literature has not explored their application specifically to software testing. The reference model accounts for certain NFRs such as Portability and Maintenance as defined by standards. Application of this model to other NFRs and across application domains has to be investigated further.

2.3.3 Fuzzy Logic Estimation

A fuzzy logic estimation process has been designed by Francisco Valdés (Valdes, 2011) as a part of his doctoral thesis. The process involves six stages of which the first three stages are setting up the fuzzy rule-based estimation model and the next three stages are the application of the model to obtain an estimate for a specific project.

The model allows the experts from a software organization to decide the most significant input variables for the kind of projects for which the model will be used. Typical input variables are size, team skills and complexity. The membership function is defined for the input variables and the values are assigned based on the opinions of the expert practitioners. This creates fuzzy values which are used in inference rule execution.

His approach differs over several other estimation techniques in terms of modelling capabilities. Estimation process based on fuzzy logic resembles how experts make decisions in the context of uncertain, incomplete, imprecise and conflicting information. Unlike expert judgement-based methods where the knowledge resides with experts, here the knowledge is captured in the form of inference rules and stay within the organization. People who use the fuzzy system to estimate do not need to be experts themselves. The estimates produced by the fuzzy model can be verified, which overcomes a limitation of expert judgement-based techniques. This approach can be explored for estimation of NFR test effort.

2.3.4 Model-driven Testing

Model-driven testing is an evolving approach to testing based on 'modelling' of requirements using established notations such as UML diagrams. Model refers to the 'what' aspect of the requirements. Requirements are translated into UML Testing Profile (UTP), Testing & Test Control notation (TTCN-3) and Object Constraint Language (OCL) that enable the creation of generic test models. Model Driven Architecture (MDA) aids the conversion of abstract test cases into test models to specific test cases (Seigl, Kai-Steffen, Reinhard, 2010). This approach improves the 'Extended Automation Method' (EXAM) through 'Timed Usage Model' (TUM) based on Markov chains using a probability density function. In this approach all sequences of stimuli and their responses across the system boundary are enumerated following the principles of 'sequence based software specification'.

By systematically and unambiguously depicting all the transitions into TUM, test cases are generated. Automated tools perform several transformations to generate and execute test cases on various platforms. The major advantage of model-driven testing is that the error prone manual activity of preparation of test cases can be avoided. This approach resulted in minimizing the number of test cases and mean test case length.

However, if the sampling method is not relevant to the context, the test cases generated may not be the most appropriate ones. There is a possibility of spending time on weak test cases while critical ones are missed out due to sampling. Some of the tests are mandatory as known to designers and they are to be included in the final set of test cases. There is no such provision in this method to include compulsory test cases. Test designers create usage model which can be error prone. There is no mechanism to validate the model.

A survey of model driven testing techniques (Musa, et al., 2009) discusses various approaches used for model-driven testing. Their survey identifies modeling languages such as UML activity diagrams, Extended UML, Class diagrams, Object Diagrams, State Diagrams/FSM, UML sequence diagrams, UML Testing Profile, Testing and Test Control Notation (TTCN-3) and Object Constraint Language (OCL). Modeling Language, Automatic Test Generation, Testing Target and Tool Support have been taken up as criteria to evaluate various Model-Driven Testing techniques. While the survey mentions that testing consumes

more than 50% of the time while model based testing is introduced, there is no quantitative data provided on saving testing time using these techniques. Modeling requires complete understanding of requirements and large upfront investment in time and money is required to build the model. The survey has not captured efforts required to build the model in various techniques surveyed. The model once built needs to be verified before usage, which is also a time and resource intensive activity to be factored in estimate. There is no information available on the 'effectiveness' of the automatically generated test cases in order to judge the efficacy of the automated test cases generation.

2.3.5 Agile Testing

Agile methods have been increasingly adopted for software development over the past decade. Agile Project Management methods and Agile Software Development methods are two broad categories under which all agile methods are grouped (COSMIC, 2011):

SCRUM, Feature Driven Development and Dynamic Systems Development method are practiced for the management of projects.

Methods such as Extreme Programming, Crystal Clear, Test Driven Development and Domain Driven Development are advocated for software development.

Customer requirements take the form of User Stories, one of the three characteristics of which is referred as 'Confirmation' meaning exactly what behaviour will be verified to confirm the scope of user story leading to the test plan. 'Estimatable' and 'Testable' are two of the six 'INVEST' criteria proposed to verify the quality of a User Story (Cohn, 2005).

From the point of view of testing, activities required to be performed to deliver value to the customer in a particular iteration have to be completed as a part of the iteration: this implies execution of all test cases for the user stories identified for the iteration. Testing related activities, such as writing code to test and/or executing test cases, are part of most of the iterations during agile development.

Using COSMIC Function Point to measure the size of a User Story instead of Story Points provides a mechanism to estimate projects early in the life cycle and to carry out performance measurements (COSMIC, 2011). COSMIC size based estimation can also be explored for estimating testing specific efforts in agile projects.

2.3.6 Service Oriented Architecture and Cloud based Technologies

Service Oriented Architecture and Cloud based technologies extend the scope of testing beyond the deployment of software. Services provided by components can change any time during their production use and require testing during the operational phase. Changes to any service oriented component can affect the overall orchestration, even if other participating components remain unchanged.

Service Level Agreements of the services provided can change during the operational phase due to the performance of a collection of components. Using probabilistic customer models to estimate the cost of checking SLAs of real time systems (Cesar, Merayo and Nunez, 2012) provides insights to handling such situations. Further research work is required to build an estimation model considering these behaviours of software designed using a service oriented architecture.

2.3.7 Automated testing

Automated testing is continuously evolving as testing techniques and approaches to testing evolve in tune with the changes in technology. Test automation is a major cost driver in software testing. Functional tests are automated for the purpose of using them repeatedly for regression testing when changes are made to the software and new builds are released. Automated test tools are essential to perform load/stress testing to identify performance bottlenecks.

Most of the techniques discussed so far can be tuned for estimating efforts for test automation despite their limitations. Test automation is akin to software development and methods for estimating software development efforts can be explored for estimating test automation efforts.

The Collaborating Automation Elements Framework (CAFÉ) (Kamala Ramasubramani, 2006) provides an architecture for test automation. Test Strategy, ROI Models for Automation, Automation Frameworks, Levels of Test Automation, Modular Script & Library, COTS & Open Source Tools, Script Extension Methods and Test break-in prevention techniques form part of this architectural framework. Components of this framework and details of individual elements provide information which can aid estimating efforts for test automation. The components of the model can be explored for building an estimation model. However, CAFÉ is a theoretical architectural model, and its practical implementation and claimed benefits are yet to be verified.

2.4 Summary

This literature study has reviewed various test estimation techniques and evaluated their strengths and weakness, resulting in documenting the state of the art in software test estimation. Five factor evaluation criteria consisting of Customer view of requirements, Functional size as a prerequisite to estimation, mathematical validity, verifiability and benchmarking have been proposed to examine existing test estimation techniques. The study also noted established criteria such as R^2, Adjusted R^2, MRE, MedMRE, Mallow's Cp used to evaluate *a posteriori* estimation models built using sample dataset. These criteria will be used to evaluate estimation models to be built for testing as a part of this research work.

Test estimation techniques have been classified into four groups based on their approach to estimation:

Judgment & Rule of Thumb-based estimation techniques: are quick to produce very approximate estimates, but the estimates are not verifiable and of unknown ranges of uncertainty. They take an implementation viewpoint of requirements to come out with estimates and cannot be used for benchmarking.

Analogy & Work Break-Down estimation techniques: may be effective when they are fine-tuned for technologies and processes adopted for testing. They take an implementation view of the requirements. Estimates can be verified when the components are properly defined and a consistent approach used. They cannot be used for benchmarking across organizations.

Factor & Weight-based estimation techniques: perform most often several illegal mathematical operations, and they lose scientific credibility in the process.

Functional Size-based estimation models: are more amenable to performance studies and benchmarking provided that a proper statistical approach is used while building the estimation model. Method used for functional size measurement plays a key role: current methods, other than COSMIC, violate mathematical principles.

The literature study has also covered new approaches and technologies, such as: model based testing, agile testing, service oriented architecture and cloud based technologies from the perspective of test estimation. There is a reference to conventional techniques such as Delphi, Analogy, software Size based estimation and Test Case Enumeration in the context of Service Oriented Architecture and Regression Testing (Bharadwaj Yogesh and Kaushik Manju, 2014). However, this remains conceptual without clarity on application to the context.

Existing estimation techniques such as judgment based, work-break down, factors & weights, and functional size methods used for estimating test efforts suffer from several limitations due to arbitrary factors and lack of compliance to metrology fundamentals while arriving at final estimates. Even those techniques that use functional size for estimation models, do not take into consideration mathematically correct functional size as a parameter. Innovative approaches, such as Fuzzy Inference, Artificial Neural Networks, and Case-based Reasoning, are yet to be adopted in the industry for estimating testing effort. A review of over 150 papers spanning 30 years (Kafle, 2014) could not find any new approaches to estimation in testing and the techniques adopted by the industry are derived from software development effort by expert judgement resulting in similar error level.

CHAPTER 3

PROPOSED UNIFIED FRAMEWORK FOR SOFTWARE TEST ESTIMATION

3.1 Introduction

Phase 2 of the research methodology involves designing a Unified Framework that can characterize facets of functional and non-functional testing. A Unified Framework is proposed in this chapter consisting of basic parameters relevant for each type of testing and techniques that can be adopted for building an estimation models for software testing.

Section 3.2 presents the structure of the proposed unified framework for software testing based on the nature of the different types of testing and with the aid of ISO Standards related to testing.

Section 3.3 defines the candidate base measures in terms of size and effort as relevant for each type of testing.

Section 3.4 relates the above two sections in order to develop estimation models that can be used to estimate test effort.

Section 3.5 summarises the unified framework and discusses how the framework can be exploited to develop test estimation models for various types of testing. This section identifies the scope of the unified framework for further elaboration in the following phases of this research work.

3.2 Structure of the Framework

Software Testing consists of both Functional and Non-functional Testing. When software undergoes changes Re-testing and Regression Testing are carried out. Re-testing will be required for both functional and non-functional testing. Automated testing comes into context when tests are to be automated. While non-functional tests are mostly automated, functional testing is automated based on the need and mostly for regression testing. Functional and

Non-functional testing are core to the structure of the unified framework. Test Management involves planning and managing test activities and defects. The structure of the proposed Unified Framework is illustrated in Figure 3.1. Individual components of the structure are presented in following sections.

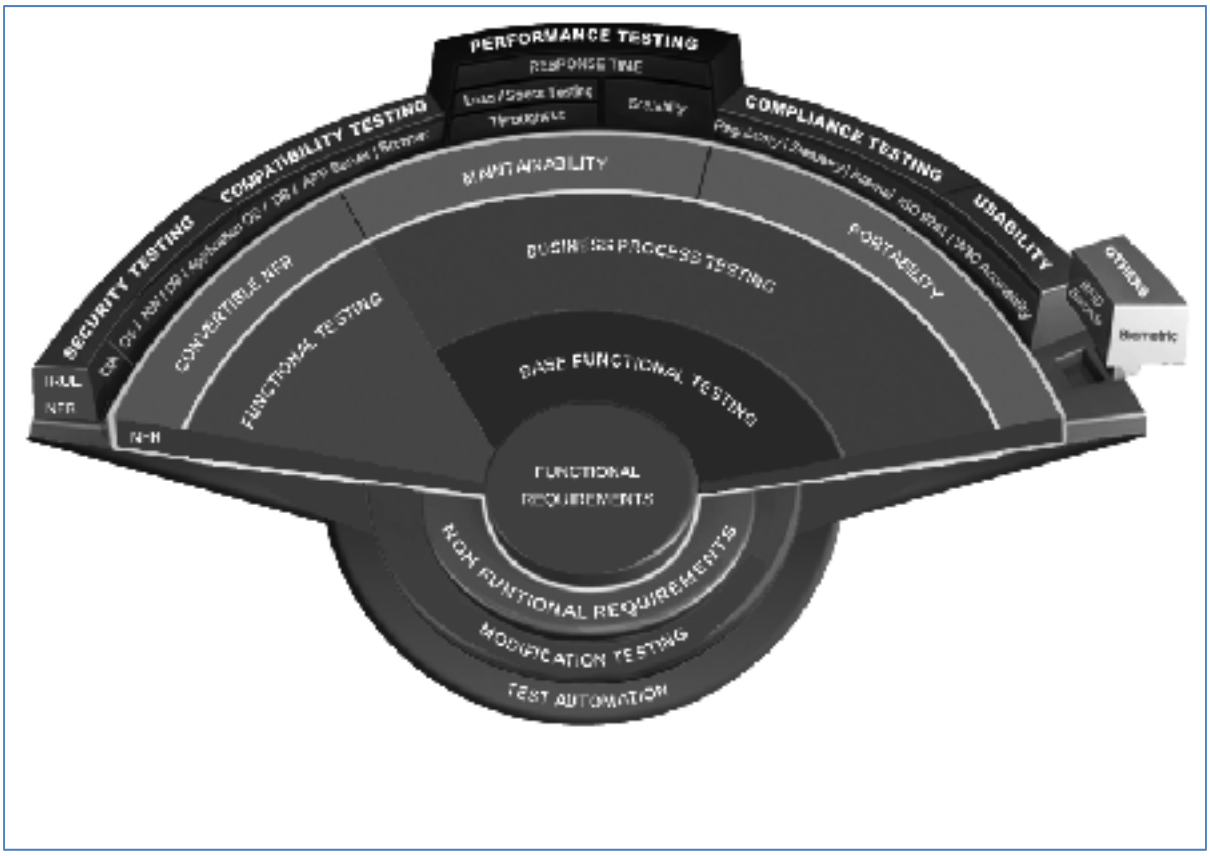


Figure 3.1 Structure of the Unified Framework

3.2.1 Functional Testing

In real-life testing, functionality as provided by the application and functionality as emerged out of operational scenarios bring in two distinct aspects to functionality testing:

1. Functional specifications serve as a primary reference for testing to ensure conformance.
2. An understanding of end to end business process help identifying various workflows within the software that can be used to validate the software for the fitness of purpose for which it is deployed.

This perspective leads to two categories: *'Base Functional Testing'* and *'Business Process Testing'* (Figure 3.1).

3.2.2 Non-functional Testing

Non-functional requirements (NFR) identified for testing can be classified into two groups:

1. NFR selected for testing such as Performance, Compatibility, Usability, Portability, Security, Maintainability, Reliability are different from one another with little commonality between them and are tested under different scopes of testing. Some of these requirements can evolve into functional requirements as per standards based framework (Al Sarayreh, Abran and Cuadrado-Gallego, 2013). These requirements are referred to as **Convertible NFR**.
2. Non Functional Requirements which cannot be converted into functional requirements are referred to as **True NFR** in (COSMIC, 2015).

Non Functional Testing is classified accordingly into *Convertible NFR Testing* and *True NFR Testing* (Figure 3.1, Table 3.1).

3.2.3 Modification Testing

Re-testing and Regression testing are carried out when software undergoes modifications. Products are to be tested to ensure that changes are implemented properly and that they have not introduced any new defects. Modification testing encompasses both functional and non-functional parts of the proposed Unified Framework.

3.2.4 Test Automation

Test automation is similar to programming, where programs are generated and/or developed to test other programs using specialized test tools and various programming/ scripting languages. Functional tests are automated mostly for regression testing after manual testing. However, recent software engineering methodologies, such as Agile, advocate test automation from the early stages of development. In case of non-functional testing use of

automation tools are essential as many of the non-functional types of testing cannot be carried out manually.

3.2.5 Mapping Framework Components to ISO Standards

Dynamic testing, as referred to in International Standard ISO 29119 – Software Testing (ISO/IEC/IEEE 29119, 2013), involves testing against both functional and non-functional requirements which cover all the quality characteristics (ISO/IEC 25010, 2011) such as Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Portability and Maintainability.

Elements of the proposed Unified Framework are mapped to International Standards (Table 3.1), enabling this framework to be used as a standard reference by builders of estimation model.

Table 3.1 Unified Framework cross referenced to ISO Standards

Test Processes ISO 29119	**Test Types ISO 29119**	**Quality Characteristics ISO 25010**	**Unified Framework Components**	
Dynamic Test Processes	Functional Testing	Functional Suitability	**Functional Testing**	
			Base Functional Testing	Business Process Testing

Test Processes ISO 29119	Test Types ISO 29119	Quality Characteristics ISO 25010	Unified Framework Components	
Dynamic Test Processes	Performance Testing Security Testing Usability Testing Compatibility Testing Reliability Testing Portability Testing Maintainability Testing …. ….	Performance Security Usability Compatibility Reliability Portability Maintainability …. …. ….	**Non Functional Testing**	
			Convertible NFR	True NFR

3.3 Approach to Measurements for Estimation

It is essential to quantify various components of the proposed Unified Framework in order to build a test estimation model. 'Functional Size' measured using a Functional Sizing Method and 'Test Effort' measured in person hours are considered as the base measures in this model. Specific variants of functional size and test efforts are defined to suit the nature of the elements of the Unified Framework.

3.3.1 Functional Testing

Functional testing type carried out to meet functional suitability quality characteristic has been classified into Base Functional Testing and Business Process Testing (Table 3.1). Follows definitions for functional size and test effort for both categories:

a. **Functional Size**: Functional Size of the software functionality measured using a Functional Size Measurement Method recognized by ISO.

b. **Functional Test Effort**: Effort required to test the functionality including efforts required to manage, design and execute tests.

c. **Business Process Size**: Size of functionality scaled up considering business processes and their variations. This includes the sum of the functional sizes of all components participating in a business process scenario considering only their inputs and outputs (i.e., excluding internal functionality of individual component). There is a similar approach used in (Izak, 2012) for measuring business process size in ERP functional size measurement method delivering time and cost estimates for implementations where business processes and their sequences as used by customer become critical input to arrive at estimates: estimations based on such an approach for in a Cash to Order business process in 9 projects resulted in 8% overrun compared to earlier judgement based methods where the overrun was 39%.

d. **Business Process Test Effort:** Effort required to test the functionality represented by Business Process Size. This includes effort required to manage, design and execute the tests.

3.3.2 Non-functional Testing

According to the Unified Framework, Non-functional Testing is categorized into testing for Convertible NFR and for True NFR (Table 3.1). When some of the NFR evolve into Functional

Requirements, those requirements will be tested as a part of functional testing and the others which remain as True NFR are to be tested separately.

Functional Size and Test Effort are defined for Convertible NFR. Only Test Effort is defined for True NFR where the concept of functional size is not relevant.

a. **Convertible NFR Size:** Functional Size of the system NFR converted into software FUR, measured using a Functional Size Measurement Method.

b. **Convertible NFR Test Effort:** Effort required to manage, design and execute the tests for testing convertible NFR.

c. **True NFR Test Effort:** Effort required for testing specific NFR which are not convertible into software FUR. This effort will include effort for automation, as automation is an integral activity in NFR Testing.

3.3.3 Modification Testing

Modification Testing for functional changes involves re-testing and regression testing. Functional Size and Test Effort for functional modification testing are defined as follows:

a. **Impact Size**: Size of changes to functional requirements plus the size of those requirements which are impacted due to changes, measured using a Functional Size Measurement Method.

b. **Modification Test Effort**: Effort required to test for the changes. Modification Test Effort includes test design efforts for the changed functionality and test execution efforts for both changed and impacted functionality, apart from test management effort.

There are no specific definitions identified for non-functional testing related to modification. They would be viewed similar to non-functional testing as defined in the previous section. True

NFR effort as defined in section 3.3.2 has to be worked out for each round of Non- functional testing whenever True NFR is modified.

3.3.4 Test Automation

Definitions for size and test effort for functional test automation have been arrived at as follows:

a. **Functional Test Automation Size:** The functional size of the requirements that would be automated will be a key driver for functional test automation. This is measured using a Functional Size Measurement method.

b. **Functional Test Automation Effort**: Efforts required to manage, create test automation scripts and execute for functional test automation. Effort includes all phases of preparation of test automation scripts like specification, design, scripting, testing the automation scripts and executing those automated scripts.

c. **True NFR Test Automation Effort**: This will be part of True NFR Test Effort defined in Section 3.3.2.

When most of the non-functional tests are automated, test effort automation of non-functional tests will be part of 'True NFR Test Effort' defined earlier.

3.4 Approaches for Test Effort Estimation

This section provides approaches for test effort estimation based on the measures identified in section 3.3. Techniques for Size Measurement and Test Effort for each type of testing as discussed below enable building estimation models corresponding to those types of testing.

3.4.1 Functional Testing

Functional Size: Functional size can be measured using IFPUG Function Points (FP) or COSMIC Function Points (CFP). It has been observed (Bhardwaj, Mridul and Rana Ajay,

2015) that functional size measurement is the most accepted approach to measuring the size and change in unit size has bigger impact than effort. Appropriate estimation techniques have to be used to reduce the margin of error. COSMIC is preferred for the following reasons:

a. Compared to other functional sizing methods, the COSMIC method measures pure functional size and is ideal for carrying out performance measurements comparisons of projects using different technology and also as an input to estimating method (Gencel Cigdem, Charles Symons, 2009).
b. COSMIC is the only second generation FSM Method approved as an ISO Standard (ISO/IEC 19761, 2011).
c. COSMIC is fully compliant with ISO 14143, a meta standard for Functional Size Measurement (ISO/IEC 14143-1, 2007).
d. Is designed from the principles of metrology.
e. Applicable across Business applications and Real-time domains.
f. Recognized by the International Benchmarking and Standards Group (ISBSG) for data collection and benchmarking.

Functional Test Effort: A posteriori estimation model using Functional Size as the independent variable and Functional Test Efforts from past data can be used to estimate functional test effort. Regression technique appears to be a promising approach due the following reasons:

a. Regression analyses have been used with Functional sizes for estimation of software development efforts with a good relationship (Hill, 2010).
b. A comparison of four models namely SLIM (Putnam, 1978), COCOMO 81 (Boehm, 1981), Estimacs (Rubin, 1982) and Function Point (Albrecht, 1979) using data from 15 projects resulted in the conclusion that function point based regression model performed much better than all the other models (Kemerer, 1987).
c. Estimation models using regression trees, artificial neural network, Function Points (IFPUG, 2005), (Boehm and Abst, 2000) and the SLIM model (Putnam, 1978) using 63 COCOMO project data points from different applications as training set and tested the results on 15 projects (Srinivasan, 1995). The regression trees outperformed the COCOMO and SLIM Model. A later study (Lionel et. al., 1999) found that the function point based prediction model performed much better than regression trees.

d. Liner regression has been sued for estimation models for testing using ISBSG data based in a study (Kamala Ramasubramani J. and Abran, 2013) carried out by the researcher.
e. Parametric models are objective, repeatable, fast and easy to use which can be used early in life cycle if they are properly calibrated and validated (Galorath, 2015).

3.4.2 Modification Testing

Impact Size: Impact size is similar to functional size but takes into account the sizes of impacted functionality in addition to size of modifications to functionality. Impact size can be measured in CFP.

Modification Test Effort: Regression Model using Impact Size and Modification Test Effort can be used to estimate functional modification test effort. The approach is similar to estimation of Functional Test Effort discussed in previous section 3.4.1

3.4.3 Non-functional Testing

NFR selected for testing under different scopes of testing engagement based on contractual needs (such as: Performance, Compatibility, Usability, Portability, Security, Maintainability, Reliability) are different from one another with little commonality between them. There may not be significant relationships between functional size and efforts required to test against those specific True NFR. Separate effort model for each of the True NFR should be built due to the distinct nature of each of the NFR.

Effort model for each of the NFR can be arrived at by considering the following possibilities:

a. NFR that can be converted into suitable software functional requirements:
   Conversion of system NFR into software Functional Requirements based on generic standards-based reference models as discussed earlier for Maintainability and Portability. Regression model based on Functional size of converted NFR and corresponding test effort can be used to estimate converted NFR Test Effort.

b. True NFR that cannot be converted into software functional requirements:

This research project has reviewed estimation models such as Work-breakdown, COCOMO-like and Fuzzy inference in addition to size based estimation model. Fuzzy inference based on expert judgment appears promising for estimating test efforts for True NFR due to following reasons:

Inputs required to estimate is often based on expert judgment and fuzzy inference model can use the input variables as identified by experts.

Experts judge the relationship between inputs and test activities, which can help defining membership functions and assigning values based on expert opinion.

3.4.4 Functional Test Automation

Functional Automation size: Size of functionality measured in COSMIC would be a key parameter in determining efforts for test automation.

Functional Test Automation Efforts: Effort model based on regression analysis with functional size has worked quite well for estimation software development projects (Hill, 2010). The same approach can be explored for test automation as automation of functional testing is analogous to a software development process.

3.5 Conclusion

The proposed Unified Framework (Table 3.2) provides a high level view for software testing in terms of measures and approaches towards estimation for all types of testing. This Unified Framework, an innovative part of this research work, has taken an holistic view and, at the same time, provides a practical approach to test estimation models.

Size and effort measures along with their relevant unit of measurement applicable to different types of testing have been proposed as a part of this framework. Estimation models may include other parameters such as domain, architecture, team size, etc., based on specific contexts that can be considered while building test estimation models.

While the Unified Framework deals with estimation for all types of testing, detailed research work presented in the next chapter will focus only on the design of estimation models for

functional testing, particularly for base functional testing: base functional testing is most fundamental to software testing and is always performed, while all other types of testing are carried out based on the testing context and needs. The base functional testing specified in this framework will be referred as the functional testing in subsequent chapters for the sake of simplicity and for conforming with the convention. Proposals related to estimation of all other types of testing appearing as a part of Unified Framework will open up opportunities for further research and are not included within the scope of this research work.

Table 3.2 Unified Framework – Measurements & Estimation Approach

Test Type	**Framework Component**	**Measure**	**Unit of Measure**	**Model Technique**
Functional	Functional Testing	Functional Size	CFP	Regression
		Functional Test Effort	Hours	Regression
	Business Process Testing	Business Process Size	CFP	Regression
		Business Process Test Effort	Hours	Regression
Non-Functional	Convertible NFR Testing	Functional Size of Convertible NFR	CFP	Regression
		Converted NFR Test Effort	Hours	Regression
	Non Convertible NFR Testing	True NFR Test Effort	Hours	Fuzzy Inference
Modification	Modification Testing	Impact Size	CFP	Regression
		Modification Test Efforts	Hours	Regression
Test Automation	Automated Testing	Functional Test Automation Size	CFP	Regression
		Functional Test Automation Efforts	Hours	Regression

CHAPTER 4

ESTIMATION MODEL FOR FUNCTIONAL TESTING

4.1 Overview

This chapter deals with the phase 3 of the research work consisting of designing estimation models and phase 4 covering evaluation of estimation models.

The engineering approach adopted to design estimation models consists of:

1. Identification of project data set from repository of software projects (Section 4.2).
2. Selection of relevant project samples and data preparation (Section 4.3).
3. Analysis of data to understand productivity levels of testing (Section 4.4.1).
4. Contextualization of the projects based on attributes and categorization of projects (Section 4.4.2).
5. Identification of independent variables of significance that can influence test effort (Section 4.5).
6. Building context specific portfolio of estimation models for each category using combination of independent variables (Section 4.6).

The following steps are performed for evaluation of the estimation models:

1. Verification of the quality of estimation models based on established criteria (4.7.1).
2. Evolving a criterion for comparing predictive performance of models (4.7.2).
3. Comparison of predictive performance of models (Section 4.7.3).

The approach does not aim at designing a single model that could handle all possible conditions but aims at designing a portfolio of estimation models suiting specific contexts. A set of estimation models will be built with data chosen from the International Software Benchmarking and Standards Group (ISBSG) repository. An initial portfolio of estimation models will be built using a larger data set followed by a second portfolio of estimation models based on a more homogeneous subset of the larger data set. A third and fourth portfolio of estimation models will use only data from projects where functional size has been measured using COSMIC Function Points and IFPUG Function Points, respectively. These estimation

models will be evaluated using various criteria for measuring outputs from estimation models. The estimation models will be compared based on the measure of their predictability.

4.2 ISBSG Data for Estimation Model

The ISBSG database consists of data related to parameters of software projects reported over the last two and half decades: it provides industry and researchers with standardized data for benchmarking and estimation (Abran, 2015). Release 12 of ISBSG data published in 2013 has a repository of 6006 projects. It provides up to 126 project attributes related to information specific to each project based on the availability of the data submitted. ISBSG dataset has been extensively reviewed for applicability to build effort estimation models including effect of outliers and missing values (Bala, 2013).

The attributes that would be of interest for building test estimation models include:

a. *Functional Size* data based on international measurement standards, including IFPUG Function Point (ISO 20296, 2009) or COSMIC Function Point (ISO 19761, 2011).
b. *Schedule, Team Size and Work Effort* information in terms of project elapsed time, team size and break down of work efforts in terms of Plans, Specifications, Design, Build, Test and Install project phases.
c. *Project Processes* related data based on software life cycle activities followed such as planning, specifications, design, build, test and adoption of practices from standards or models such as ISO 9001, CMMI, SPICE, PSP etc. used while developing the software.
d. *Grouping Attributes* consisting of industry sector, application groups such as business application, real time etc., and development type in terms of new development, enhancement or re-development.
e. *Development Platform* information such as PC, Mid-Range, Main Frame or Multi-Platform.
f. *Architecture information* such as whether the application is built Stand Alone, Multi-Tier, Client/ Server or Web.
g. *Language Type* information in terms of 3GL, 4GL, Application Generators used for development.

h. *Data Quality* Rating in terms of A, B, C or D varying from Very Good to Unreliable assigned by the ISBSG data administrator.
i. Another attribute provides Unadjusted Function Points data quality rating in terms of A, B, C or D if IFPUG Function Points is used for the measurement of Functional Size.

The above data fall into one of the following scale types:

1. Nominal (Project Management Tool name, Requirements Tool name, etc.,)
2. Categorical (Platform Types such as Main Frame, PC, etc.,)
3. Numerical (Functional Size in COSMIC Function Points, Project Work Effort in Hours, etc.,)

4.3 Data Selection

4.3.1 Criteria for data selection

A set of criteria to ensure the quality of the data, relevance of the data to current industry needs, suitability of the data to the testing context and adequacy of data for statistical analysis have been defined as follows.

4.3.1.1 Data Quality

a. ISBSG Quality Rating:
   The ISBSG Data Quality Rating field is a categorical field with the following candidate rating and corresponding criteria (ISBSG, 2013):
   - A: The data submitted were assessed as sound, with nothing identified that might affect their integrity.
   - B: The submission appears fundamentally sound, but there are some factors that could affect the integrity of the submitted data.
   - C: Due to significant data not being provided, it was not possible to assess the integrity of the data.
   - D: Due to one factor or a combination of factors, little credibility should be given to the data.

In this research project, data with Data quality rating A and B are selected to reduce the risks of poor data quality and improve confidence on the result of analysis.

b. Function Point Size Quality:
   When IFPUG Function Points are used for measurement of Functional Size, only the Unadjusted Function Points value is considered. A rating has been assigned by ISBSG to the quality of Function Points data based on the integrity of data:
   - A: Data is sound with nothing identified affecting the integrity of data.
   - B: Data appears to be sound, but integrity cannot be assured as a single final figure was provided.
   - C: Due to break-down data not provided, the data may not be reliable.
   - D: Due to one or a combination of factors, little credibility should be given to the data.

   Data quality ratings of C and D are excluded from the data set in order to improve the quality of estimation models.

4.3.1.2 Data Relevance

ISBSG data consists of projects reported since the early '90s. Most of the projects during' the 90s were developed in mainframe environment, predominantly using the COBOL programming language. In order to ensure data relevant to current development environment is taken up for building model the following filters were applied:

a. Size Measurement Method chosen should be either IFPUG 4+ or COSMIC, which would represent industry data post year 2000.
b. Projects with architecture as 'Stand Alone' removed in order to eliminate trivial projects.
c. Architecture values with either Client/Server or Web based projects data were considered.
d. Architecture value with 'blanks' will be included in an initial data set.
e. Architecture value with 'blanks' will be filtered out for another data set.

4.3.1.3 Data Suitability

In order to build estimation model for software testing, effort data related to software testing should be non trivial. ISBSG data field 'Normalized Work Effort' refers to full life-cycle effort for projects. For partial life cycle projects, ISBSG fills this field with an estimate of the full development life-cycle effort. For projects covering the full development life-cycle, and

projects where life-cycle is not known, this value is the total effort in hours recorded for the projects, referred to as Summary Work Effort in ISBSG. In order to exclude trivial projects and include only projects with at least a minimum amount of test efforts expended on projects, the following filters are chosen:

a. Total normalized work effort is equal to or above 80 hours.
b. Efforts reported for testing is above or equal to 16 hours.
c. Projects referring to types of testing other than functional testing will be filtered out of the data set.

4.3.1.4 Data Adequacy

a. Application Group is chosen as 'Business Application'. Several projects are of Business Application category and this filter would result in a larger data set. As Functional size is a key parameter in estimation model, the methods used for functional sizing should be suitable to the application type. Functional Sizing methods IFPUG Function Point and COSMIC Function Point both are applicable for Business Application, while only COSMIC is applicable for other types.
b. New Development and Re-Development were chosen from the Development Type field among the values New Development, Enhancement and Re-Development. Data related to enhancement were fewer for project context specific statistical analysis.

4.3.2 Data Preparation

Applying all the filters related to the criteria for data selection resulted in 193 data points. Identification of outliers based on domain and John Tukey's Inter Quartile Range statistical criteria resulted in 170 data points (see APPENDIX I). Further based on identification of suitability of the data points (see APPENDIX I) for estimation model, 28 data points were removed, thus resulting in 142 data points for building estimation model. Details of application of individual filters and resulting data points are presented in APPENDIX I.

Four data sets are formed by applying the criteria for data selection and then removing outliers for building estimation models:

Data Set A: This data set A of the 142 data points from above contains functional size measured using IFPUG 4.1 and COSMIC FP which are not differentiated for this study considering the fact that they correlate well even though the relationship is not the same across all size ranges (Dumke and Abran, 2011).

Data Set B: While arriving at Data Set A, projects with field value for architecture 'Stand Alone' were eliminated from the original ISBSG data set, while 'blanks' were retained. In order to be very specific about the architecture type, 'blanks' were also eliminated to arrive at Data Set B. This data set is expected to be more homogeneous than Data Set A. Data Set B consists of 72 data points.

Data Set C: Data Set C is made up of only projects where functional size is measured using the COSMIC Function Points method. This data set is another subset of Data Set A and likely to be more homogeneous. Data Set C consists of 82 data points.

Data Set D: Data Set D is formed considering only projects where functional size is measured using the IFPUG method. This data set is another subset of Data Set A and consists of 60 data points.

4.4 Data Analysis

4.4.1 Strategy

The following research strategy is adopted for data analysis:

a. Identification of subset of data points exhibiting different levels of testing productivity. This is discussed in this section 4.4.2.
b. Analysis of each of these subsets to identify what could be the causes for such distinct testing levels of productivity. This will be discussed in section 4.4.3.

4.4.2 Identification of Test Productivity Levels

'Functional Size' and efforts for testing reported as 'Test Effort' are key data values from Data Set A taken up for the initial analysis. The scatter diagram in Figure 4.1 depicts an overall large dispersion in the relationship between functional size and test efforts, the respective independent and dependent variables: the pattern is closer to wedge-shaped and is typical of data from large repositories (Abran, 2015).

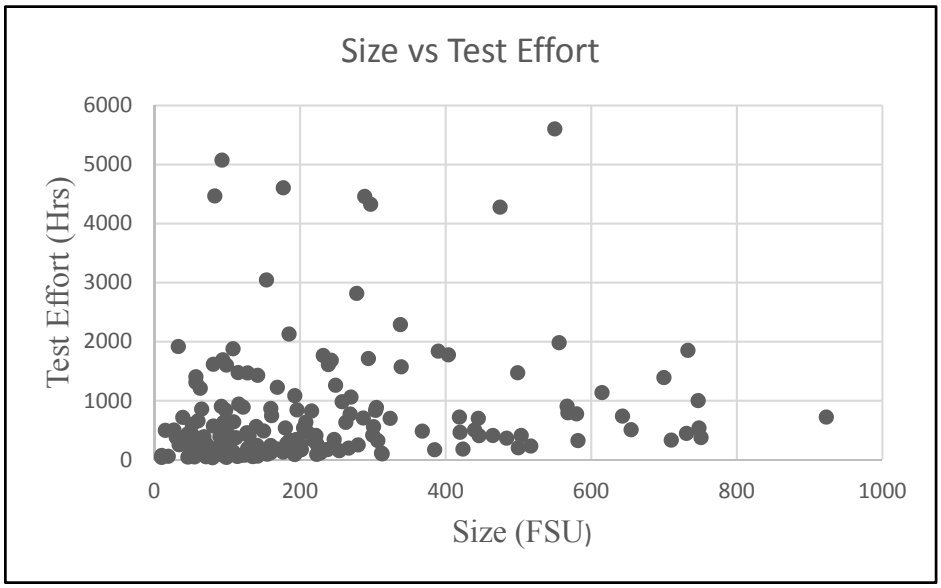


Figure 4.1 Scatter Diagram: Size versus Test Effort (N = 170)

Within this dataset of Figure 4.1, there are candidate groups of data within the data exhibiting both large economies of scale and large diseconomies of scale: the rate of increase of test effort is not same for all similar functional sizes. For those set of projects demonstrating large economies of scale, an increase in functional size does not lead to significantly larger increase in efforts. On the other hand, projects demonstrating diseconomies of scale, a small increase in size requires a much larger increase in test effort.

A further graphical analysis of data slicing along different testing productivity levels resulted in four subsets (Figure 4.2).

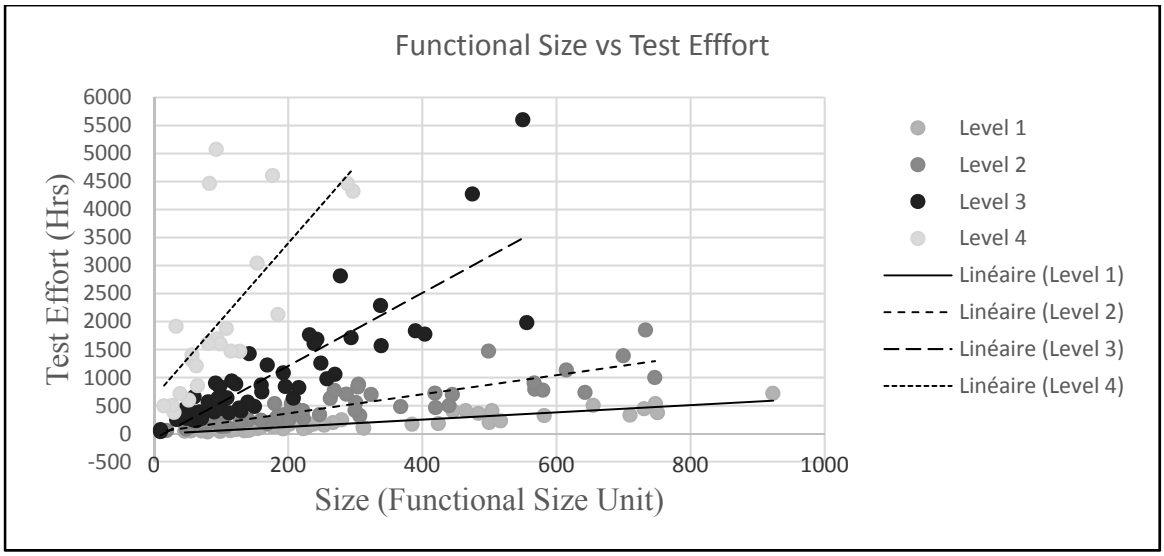


Figure 4.2 Multiple Data groups representing different economies of scale (N=170)

As economies and diseconomies of scale represent different productivity levels during testing, a new term 'Test Delivery Rate (TDR)' is defined to deal with the productivity of testing projects. Test Delivery rate is the rate at which software functionality is tested as a factor of the effort required to do so and is expressed as hours per Functional Size Unit (hr/ FSU). Functional Size Unit (FSU) refers to either IFPUG Function Point or COSMIC Function Point depending upon the sizing method used for measurement. The four varying levels of productivity will be referred as 'TDR Levels'. TDR, being the effect, characteristics of the project falling into each of the level have to be investigated for identifying underlying causes.

Review of TDR levels (Refer APPENDIX – I) resulted in the following observations:

a. TDR for projects is uniformly distributed within levels 1, 2 and 3 but not for level 4.
b. TDR level 4 projects appear quite odd as their minimum test effort is 7 – 12 times more than other levels, while the maximum functional size itself is 0.4 – 0.7 times of other levels.
c. The 23 data points in level 4 consists of small projects in terms of functional size and disproportionately very large test efforts compared to the other projects.

Due to the highly skewed nature of TDR Level 4, only TDR Levels 1 to 3 are taken up for further analysis and building estimation models.

4.4.3 Identification of Candidate Characteristics of Projects

Research study carried out at Putnam's organization (Putnam, 2005, Kate 2012) based on data from hundreds of software projects brought out the fact that team size and schedule (duration of the project) within a particular domain affect productivity of development project. As software testing is one of the phases of development projects, project attributes such as Domain, Team Size and Elapsed Time are likely candidates for software testing productivity too. Testability of the software components, i.e., quality of the software delivered for testing is found to be critical for reducing the testing cost (Ossi, 2007) and hence the effort for testing. Quality of the software delivered for testing can be determined by the extent of verification and validation activities carried out during the development process.

Candidate project characteristics that would of interest in this context of test effort estimation would be application domain of the project, size of team involved, estimated duration of the project and the extent of verification and validation activities carried out. These attributes of projects can be derived from the ISBSG data fields.

Team Size: Team sizes of 1 to 20 members have been more frequent in the team size field in the data set. I classified team size into three categories:

a. 1 – 4 (referred as 'small')
b. 5 – 8 (referred as 'medium')
c. above 8 (referred as 'large')

They represent small, medium and large team sizes typically present in the industry.

Elapsed Time: Elapsed time in calendar months is derived from the 'Project Elapsed Time' field from the data set. In order to characterize the projects as small, medium and large which are typically related to duration, this attribute has been classified into three groups:

a. 1 – 3 months (referred as 'small')
b. 4 – 6 months (referred as 'medium')
c. above 6 months (referred as 'large')

V & V Rigour: This attribute is derived from the values for data fields related to 'Documents & Techniques' category in ISBSG that can indicate the rigour followed during verification and

validation activities. Fields such as specifications, design and build representing documents/ artefacts produced during the development phases and techniques such as review carried out during each of the phase can indicate the quality of the software engineering life cycle followed during the development. Two ratings are worked out for V & V Rigour (Table 4.1). Refer to APPENDIX II for mapping of ISBSG data fields to V & V rigour for assigning rating.

Table 4.1 V & V Rigour Rating Scheme

V & V Rigour Rating	Description
Low	None or very little evidence of Reviews/Inspection to infer the rigour.
High	Reviews/ Inspection reported for at least one of the specification, design and build phases.

Application Domain: This attribute has been derived from the ISBSG data field 'Industry Sector'. Typical values found for this field are Banking, Service Industry, Government, Financial, Communication, Insurance, etc. Considering the number of data points available for different industry sectors, application domain is classified into three categories namely:

a. IT Services representing banking, financial and insurance (BFSI).
b. Educational.
c. Government.

The group of projects contributing to each TDR level will be referred to as Project Group. Project Group 1 (PG1), Project Group 2 (PG2) and Project Group 3 (PG 3) are related to TDR Levels 1, 2 and 3 respectively. The percentages of projects falling into each of the project groups for the four attributes of interest (Table 4.2) enable characterisation of the project groups.

Table 4.2 Analysis of Project Characteristics – Data Set A

Domain	No./ %	PG 1	PG 2	PG 3
BFSI	No	14	23	32
	%	20	33	46
Education	No	11	0	0
	%	100	0	0
Government	No	6	10	2
	%	33	56	11

Team Size	No./%	PG 1	PG 2	PG 3
Small	No	10	4	2
	%	63	25	13
Medium	No	18	14	7
	%	46	36	18
Large	No	5	4	8
	%	29	24	47

Elapsed Time	No./ %	PG 1	PG 2	PG 3
Small	No	18	7	5
	%	60	23	17
Medium	No	6	8	7
	%	29	38	33
Large	NO	14	20	16
	%	28	40	32

V & V Rigour	No./%	PG 1	PG 2	PG 3
Low	No	25	42	43
	%	23	38	39
High	No	21	7	4
	%	66	22	13

Closer to half of the BFSI projects (46%) fall in PG3 followed by 1/3rd in PG2. All projects of education domain fall in PG1 while slightly more than half of the government projects fall in PG2.

As far as the Team Size attribute is considered, almost 2/3rd of the projects with small team size fall in PG1, while 82% (46% + 36%) of the projects of medium team size are shared by PG1 and PG2. Little less than 50% of the projects of large team size fall in PG3.

Small Elapsed Time is closer to 2/3rd in PG1, while 71% (38% + 33%) of the projects with medium Elapsed Time are spread between PG2 and PG3. Similarly, PG2 and PG3 share 72% (40% + 32%) of the projects with large elapsed time.

Projects with higher V & V rigour has 2/3rd presence in PG1 while 77% (38% + 39%) of the lower V & V rigour projects are spread between PG2 and PG3.

The three project groups have certain distinctions with respect to Team Size, Elapsed Time, V & V Rigour and Domain apart from test productivity. In order to establish statistical significance, test of hypothesis performed and p value is computed.

Table 4.3 Statistical Significance of Attributes in Data Set A (N = 142)

Attribute/ Statistical Test	**Team Size**	**Elapsed Time**	**V & V Rigour**	**Domain**
Chi-Square P Value	0.088	0.057	< 0.001	< 0.001

The Chi-Square Test conducted (Table 4.3) on the three project groups with respect to these attributes resulted in p value of less than 0.001 for V & V Rigour and Domain and less than 0.1 for Team Size and Elapsed Time. This further establishes that variations across the three project groups are reasonably significant and the attributes identified are potential contributors to the test productivity. These project characteristics could be the causes for different TDR levels resulting in economies and diseconomies of the scale depicted in Figure 4.2.

Other general observations based on these analyses are:

a. Smaller team size projects and short duration projects exhibit higher productivity in terms of TDR and largely fall in PG1.
b. Projects executed with rigorous verification and validation have higher TDR and mostly fall in PG1.
c. Less business critical projects such as educational projects executed at universities have higher TDR, falling into PG1 as compared to projects executed for Government and BFSI domains.

These project attributes (Table 4.4) pave the way for building a portfolio of estimation models representing different contexts than a single model representing the entire data set.

Table 4.4 Characteristics of Project Groups

Attribute	PG1	PG2	PG3
Domain	Educational	Government	BFSI
Team Size	Small/ Medium	Small/ Medium	Large
Elapsed Time	Small	Medium/ Large	Medium/ Large
V & V Rigour	High	Low	Low

The statistical tests (Table 4.5) conducted to compute the P Value establishes higher level of significance across three product groups for test effort.

Table 4.5 Statistical Significance of Project Groups

Statistical Test	P Value
PG and Test Effort Significance	< 0.001

The results of analysis of project characteristics based on Data Set A and Data Sets B, C and D (APPENDIX III), with exception to Data Set D, demonstrate more or less similar behaviour. Project attributes taken up for analysis such as Domain, Team Size, Elapsed Time and V & V Rigour are reasonable causes for varying productivity level in testing. Projects can be grouped into one of the three project groups namely PG1, PG2 and PG3 based on these attributes. Separate estimation models can be built for each project group which can more closely represent the projects within that group instead of developing a single model.

4.5 Identification of Independent Variables

It has been identified earlier (Section 3.4) in the unified framework that functional size (referred as ‘size’ in later sections) is one of the independent variables that can be used in building estimation model software testing using ISBSG data set. Scatter Diagrams of Size versus

Figure 4.3 Scatter Diagrams for Size vs Test Effort: Data Set A, PG1, PG2 and PG3

Test Effort for the whole data set A and individual project groups PG1, PG2 and PG3 (Figure 4.3) depicts relationship between size and test effort within the data sets chosen for estimation model for testing.

Correlation coefficients computed between Size and Test Effort in the data sets (Table 4.6) also indicate good correlation and hence size is chosen as the primary independent variable.

Table 4.6 Correlation Coefficients for Size Vs Test Effort

Data Set	Correlation coefficient
Data Set A	0.3565
Data Set A PG1	0.9035
Data Set A PG2	0.8572
Data Set A PG3	0.8752

Size being the main independent variable, other independent variables are examined for their significance to incorporate into estimation models.

AssessQ model for software test effort estimation designed (Mutalik, 2003) for independent testing projects, discussed in the literature study, brought out 'Developer maturity', referring to the quality of the processes used for development and 'Maturity of verification and validation processes' as two of the process factors affecting estimates for testing.

In Ossi Taipale Ph. D thesis (Ossi, 2007), it has been observed that 'testability of software components' meaning the quality of the software delivered for testing and testing processes followed while testing as critical factors for reducing testing cost and improving software quality.

In order to accommodate for the process factors two new variables representing development process quality and testing process quality have been defined and investigated next.

4.5.1 Development Process Quality Rating (DevQ)

The process followed during the development can be rated considering the nature of the development life cycle followed and the artefacts produced, using the following attributes of the project:

1. Standards followed
2. Distinct development life cycle phases followed.
3. Verification activities carried out during development.

The ISBSG data field ‘software process’ has one of the values - CMMI, ISO, SPICE, PSP or any such standard followed during the development. A set of fields representing ‘Documents and Techniques’ exists in ISBSG data provide information on the life cycle phases followed and verification activities carried out during the development (Refer to APPENDIX IV). Based on these, a rating for DevQ has been arrived at as in Table 4.7.

Table 4.7 Rating for Development Process (DevQ)

Software Process	Documents & Techniques	DevQ Rating
Not reported	Very little reporting to infer	0
Reported	Very little reporting to infer	1
Not reported	One or more phases has values	1
Reported	One or more phases has values	2

4.5.2 Test Process Quality Rating (TestQ)

While reviewing the data related to the testing process followed, it is found that there were not enough fields in ISBSG data to capture the details of the testing process followed such as testing techniques adopted, levels of testing executed, test artefacts produced, reviews of test cases etc., to gauge the extent of testing. However, it is possible to classify the test process rating broadly into two categories (Table 4.8). The detailed mapping of the test process rating criteria against fields of ISBSG is provided in APPENDIX IV.

Table 4.8 Rating for Test Process (TestQ)

Test Process Criteria	Test Process Rating (TestQ)
No evidence of Test Artefacts	0
Evidence of Test Artefacts	1

4.5.3 Analysis of DevQ and TestQ

Classification of the data set in terms of DevQ and TestQ rating reveals that 36%, 49%, and 15% of the projects are of DevQ Ratings 0, 1 and 2 respectively. TestQ percentages are 80% and 20 % for TestQ ratings 0 and 1 respectively (Figure 4.4).

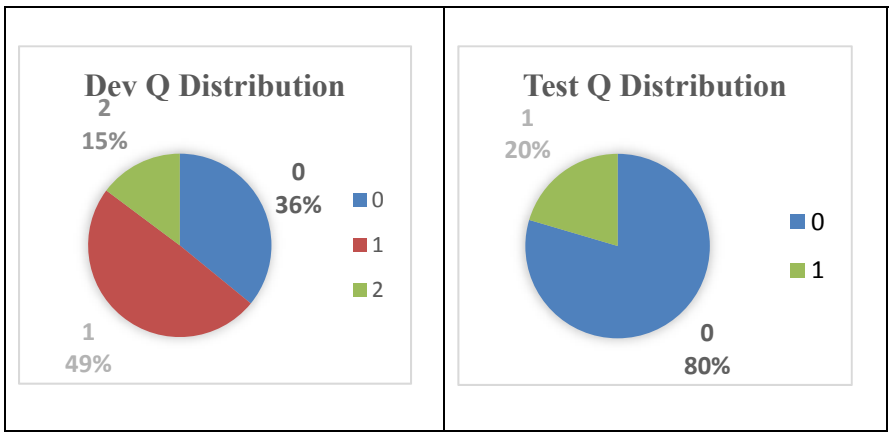


Figure 4.4 Distribution of DevQ and TestQ Ratings (N = 142)

Different median test effort and interquartile ranges for DevQ values of 0, 1 and 2 (Figure 4.5) reveals the effect of DevQ on test effort. Median test effort and the interquartile range is the lowest for the highest DevQ rating.

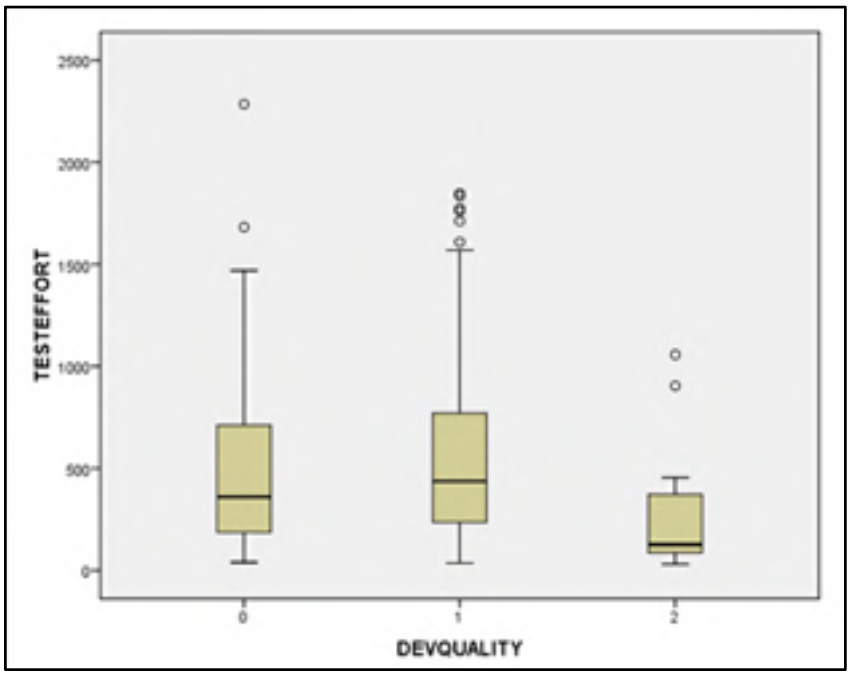


Figure 4.5 Box Plots of DevQ Ratings 0, 1 and 2

Similarly, different median test effort and interquartile ranges for TestQ values of 0 and 1 (Figure 4.6) reveals the effect of TestQ on test effort. Median test effort and the interquartile range is the lowest for the highest TestQ rating.

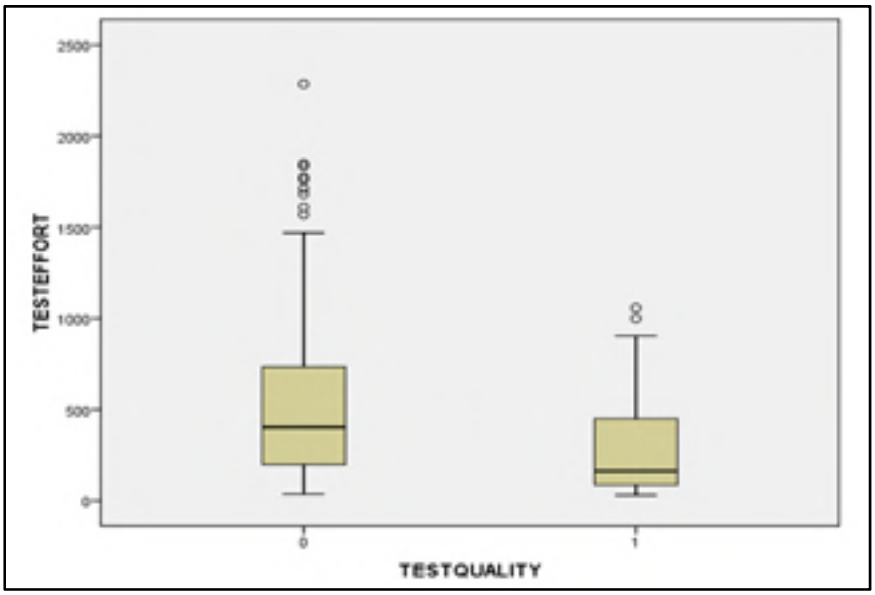


Figure 4.6 Box Plots of TestQ Ratings 0 and 1

In order to further justify the inclusion of these variables, two statistical tests were carried out to evaluate their significance:

1. the Kruskal-Walis Test was taken up for DevQ as it involved three categories, and
2. the Mann Whitney Test was applied for TestQ (Table 4.9).

Table 4.9 Test of Significance for Independent variables

Statistical Test	Variable	P Value
Chi Square P Value	Size	< 0.001
Kruskal-Wallis Test	DevQ	0.005
Mann-Whitney Test	TestQ	0.003

The P-value indicates that Size, DevQ and TestQ are statistically significant.
Analysis of DevQ and TestQ for Data Set B, C and D are detailed out in APPENDIX V. These analyses reveal that the two additional variables viz., DevQ and TestQ can play a useful role in estimation models.

4.6 Portfolio of Estimation Models for Functional Testing

4.6.1 Estimation Models – Data Set A (N = 142)

The linear regression technique has been chosen to build test effort estimation models. APPENDIX VI details out a study of fitment of different models to the data set. The linear models are better understood by the practioners and simpler to use. Further data set used for models are subdivided into three project groups exhibiting similar characteristics, making application of the linear model a reasonable choice.

A portfolio of 9 models are built using Data Set A:

a. Three models are built using functional size as an independent variable for each of the three Project Groups using simple linear regression.

b. Three models are built with Size and DevQ as independent variables for each of the Project Group using multiple regression technique.
c. Three models are developed using Size, DevQ and TestQ for each of the Project Group using multiple regression technique.

Each model in this portfolio has been identified with a Model Id (Table 4.10), Model Name, Project Group to which it corresponds to, related independent variables and the number of data points available.

Table 4.10 List of Models in Portfolio A

Model ID	Model Name	Project Group	No of Data Points	Independent Variable		
				Size	DevQ	TestQ
1	APG1S	1	46	√	-	-
2	APG2S	2	49	√	-	-
3	APG3S	3	47	√	-	-
4	APG1SD	1	46	√	√	-
5	APG2SD	2	49	√	√	-
6	APG3SD	3	47	√	√	-
7	APG1SDT	1	46	√	√	√
8	APG2SDT	2	49	√	√	√
9	APG3SDT	3	47	√	√	√

a. Model IDs 1, 2 and 3 present model for each Project Group using Size as an independent variable.
b. Model 4, 5 and 6 use both Size and DevQ as independent variables and relate to Project Groups 1, 2 and 3 respectively.
c. Models 7, 8 and 9 use Size, DevQ and TestQ as independent variables and represent Project Groups 1, 2 and 3 respectively.

Depending upon the number of independent variables, model equations will have co-efficients A, B, D1, D2, T1 and T2 (Table 4.11), which can be used to estimate a value for Test Effort for specific values of Size, DevQ and TestQ as explained next.

Using Estimation Models1 2 & 3 based on Size

Test effort for a particular functional size can be estimated from models (Table 4.11) using the following equation:

Test Effort = A + B × (Size)

Using Estimation Models 4, 5 & 6 based on Size and DevQ

Test effort for a particular values of functional size and DevQ can be estimated from models (Table 4.11) using the following equation:

Test Effort = A + (B × (Size)) + D1 + (D2 × (Size))

D1 and D2 have different values based on the value of DevQ. Appropriate values from the table to be chosen depending on whether DevQ = 0 or Dev Q = 1. For DevQ = 2, the value will be 0, the base value considered for this model.

Using Estimation Models 7, 8 & 9 based on Size, DevQ and TestQ

The equation for estimating Test Effort for particular values of Size, DevQ and TestQ from the model (Table 4.11) takes the form:

Test Effort = A + (B × (Size)) + D1 + (D2 × (Size)) + T1 + (T2 × (Size))

Values for D1 and D1 can be chosen from the table depending upon input value of DevQ either 0 or 1. Value for T1 and T2 are provided for TestQ value of 0. Values for DevQ = 2 and Test Q = 1 are zero, as they are considered as base while modelling.

The portfolio consisting of nine models generated by regression as explained above are listed in Table 4.11 in terms of the values of the coefficients of the equations representing the models.

Table 4.11 Estimation Models - Portfolio A (N = 142)

ID	Model Coefficients							
	A	B	D1		D2		T1	T2
			DevQ=0	DevQ=1	DevQ=0	DevQ=1	TestQ=0	TestQ=0
1	1.617	0.604						
2	20.69	1.705						
3	98.13	4.801						
4	16.12	0.485	19.347	-39.375	-0.23	0.214		
5	20.57	1.56	-94.1	34.077	0.562	-0.009		
6	38.85	3.734	-55.913	92.609	2.14	0.852		
7	-9.62	0.65	6.967	-41.78	0.003	0.193	38.124	-0.191
8	30.74	1.541	-19.755	62.481	-0.039	-0.338	-84.511	0.62
9	38.85	3.734	-55.913	92.609	2.14	852	0	0

4.6.2 Estimation Models - Data Set B (N=72)

In the previous sub-section, the data Set A has been considered for estimation models in order to gain the larger picture with respect to project characteristics related to various TDR levels. The Data Set A with 142 data points represents web or client server projects even though they include projects where the information about their architecture value was not filled up. In order to remove some uncertainty related to projects with blanks, the Data Set B with 72 projects was filtered out from Data Set A by eliminating projects with architecture value as blank. Therefore, the Portfolio B of estimation models identified based on Data Set B (Table 4.12) represent Web and Client/ Server projects more closely than Portfolio A models.

Table 4.12 List of Estimation Models - Portfolio B

Model ID	Model Name	Project Group	No of Data Points (N)	Indepdendent Variable		
				Size	DevQ	TestQ
10	BPG1S	1	32	√	-	-
11	BPG2S	2	24	√	-	-
12	BPG3S	3	16	√	-	-
13	BPG1SD	1	32	√	√	-
14	BPG2SD	2	24	√	√	-
15	BPG3SD	3	16	√	√	-
16	BPG1SDT	1	32	√	√	√
17	BPG2SDT	2	24	√	√	√
18	BPG3SDT	3	16	√	√	√

Portfolio B consisting of nine models are listed in Table 4.13 in terms of values of coefficients:

a. Models 10, 11 and 12 relate to Project Groups 1, 2 and 3 respectively using Size as independent variable.
b. Models 13, 14 and 15 relate to Project Groups 1, 2 and 3 respectively using Size and DevQ as independent variables.
c. Models 16, 17 and 18 relate to Project Groups 1, 2 and 3 using Size, DevQ and TestQ as independent variables.

Table 4.13 Estimation Models Portfolio B (N = 72)

ID	Model Coefficients							
	A	B	D1		D2		T1	T2
			DevQ=0	DevQ=1	DevQ=0	DevQ=1	TestQ=0	TestQ=0
10	-8.3448	0.61						
11	-30.569	1.929						
12	-157.62	6.126						

ID	Model Coefficients							
	A	B	D1		D2		T1	T2
			DevQ=0	DevQ=1	DevQ=0	DevQ=1	TestQ=0	TestQ=0
13	16.124	0.485	46.572	-52.672	-0.201	0.222		
14	20.57	1.56	-180.84	-60.58	0.973	0.313		
15	38.847	3.734	-375.38	5.027	3.881	1.449		
16	-12.583	0.68	58.608	-43.272	-0.208	0.171	16.67	-0.188
17	56.462	1.492	2.443	129.634	-0.354	-1.025	-219.18	1.395
18	38.847	3.734	-375.38	5.027	3.881	1.449	0	0

Specific model from Portfolio B can be chosen based on project context in terms of Project Group 1, 2 or 3 and based on the availability of DevQ and/ or TestQ as additional inputs.

4.6.3 COSMIC Function Point Estimation Models – Data Set C (N=82)

The models in Portfolio A and B both have been built using data of projects measured in IFPUG Function Point or COSMIC Function Point meaning that the size measure in some projects has FP (IFPUG Function Point) as unit of measure and for the rest of the projects CFP (COSMIC Function Point) as their unit of measure. In practice, size is measured using any one of the methods. However, Portfolio A and B models serve as useful reference as there is a correspondence between both functional sizes (Dumke and Abran, 2011). In order to generate models specific to COSMIC Function Points measured projects, Data Set C, a subset of Data Set A consisting of 82 projects is taken up. Portfolio C (Table 4.14) consisting of 3 models, one for each of the project group are generated, which can be used if the measurement method is known as COSMIC Function Point. Models with additional variables DevQ and TestQ are not listed due to lack of sufficient data within project groups.

Table 4.14 Estimation Models Portfolio C (N = 82)

Model ID	Model Name	PG	No. of Data Points (N)	Model Coefficients	
				A	B
19	CPG1S	1	27	-20.142	0.693

Model ID	Model Name	PG	No. of Data Points (N)	Model Coefficients	
				A	B
20	CPG2S	2	26	47.999	1.590
21	CPG3S	3	29	136.267	4.481

4.6.4 IFPUG Function Point Estimation Models – Data Set D (N = 60)

IFPUG Function Point based models are generated (Table 4.15) out of Data Set D consisting of projects reporting functional size in terms of IFPUG FP unadjusted points. Data Set D is derived as a subset of Data Set A, by choosing only projects reporting size method as IFPUG Function Points. Model ID 22 represents Project Group 1, Model ID 23 represents Project Group 2 and Model ID 24 Project Group, all of them use only size as independent variable. Due to lack of data, models using DevQ and TestQ could not be generated within project groups.

Table 4.15 Estimation Models Portfolio D (N = 60)

Model ID	Model Name	PG	No. of Data Points (N)	Model Coefficients	
				A	B
22	DPG1S	1	19	37.588	0.455
23	DPG2S	2	23	-29.939	1.917
24	DPG3S	3	18	77.585	6.087

4.6.5 Model Selection for Estimation

The models developed are categorised into the project groups based on the attributes of the project and portfolios based on the data set. Within a portfolio and project group models differ from one another based on the combination of independent variables used for modelling. Estimator can choose a particular model for estimation following the following steps:

a. Decide the Project Group

b. Decide the relevant portfolio from models built using different data sets.
c. Choose the model based on the availability of values for independent variables

Decide Project Group:

Estimator can map the project to be estimated to the project attributes – domain, team size, elapsed time and v & v rigour. Domain of the project to be tested is known prior.
Team size and elapsed time of the project with respect to 'development' would have been already estimated as a part of estimation for the development. These two attributes together indicate comparative size of the project referred as 'project size' using which projects are classified as Small, Medium and Large.

Table 4.16 Project Size Classification

Team Size	Elapsed Time	Project Size
Small	Small	Small (S)
Small	Medium	Small (S)
Small	Large	Medium (M)
Medium	Small	Small (S)
Medium	Medium	Medium (M)
Medium	Large	Medium (M)
Large	Small	Medium (M)
Large	Medium	Large (L)
Large	Large	Large (L)

Extent of project management and process requirements are determined by the project size and are factored into project planning in the Project Management Advisor tool built by University of Wisconsin (Wisconsin, 2007; 2015). Based on the values of Team Size and Elapsed Time, the variable Project Size can be figured out (Table 4.16).

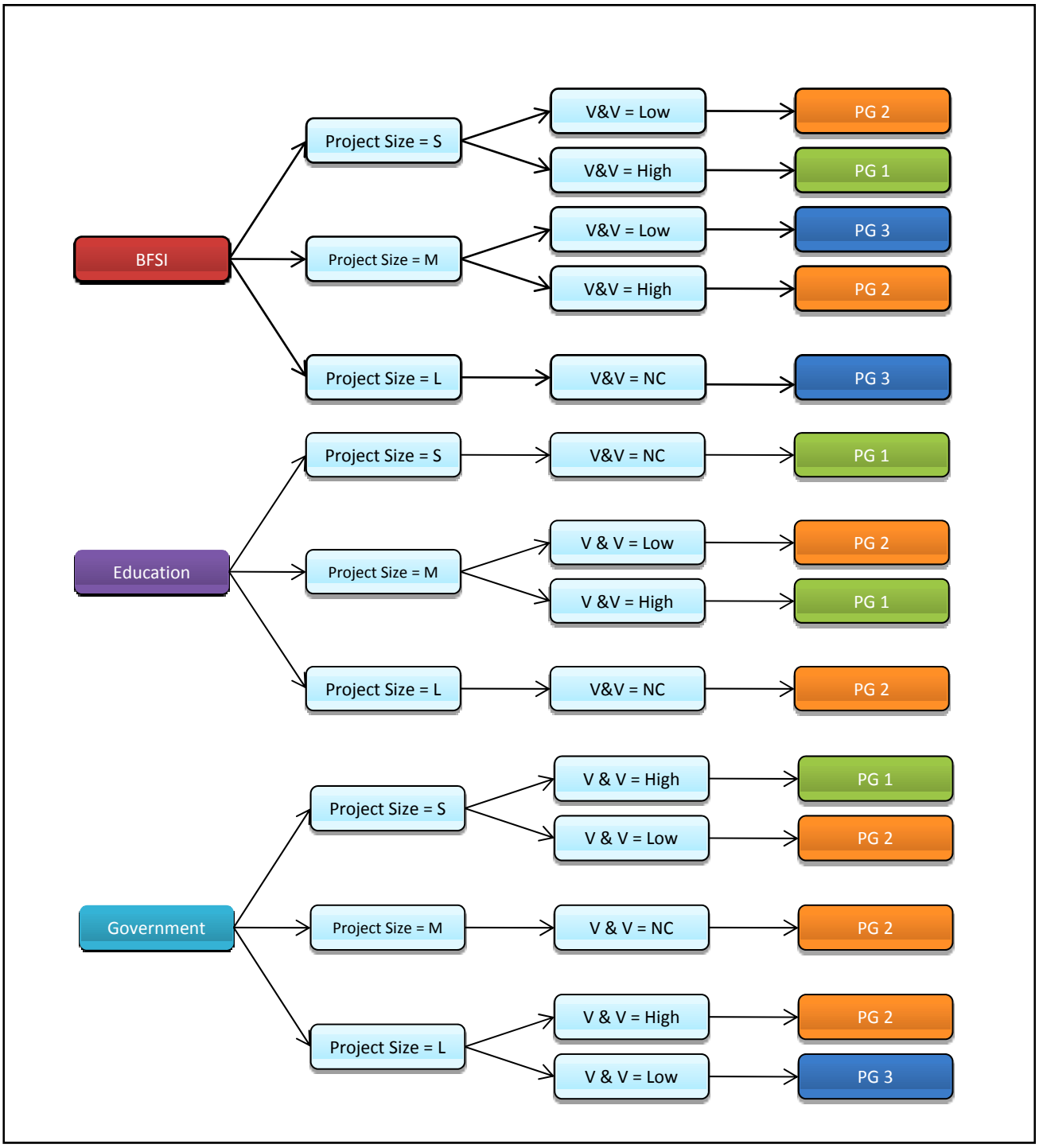


Figure 4.7 Project Group Selection Decision Tree

V & V rigour refers to the quality of the development process followed: the value may be known at the time of testing. Higher rating for V & V rigour improves the quality of the product delivered for testing. It can be observed that TDR Level is high, if V & V rigour is high as 66% of the projects (Table 4. 2) with V & V rigour 1 (i.e., High) falls into PG1.

Once a choice is made based on domain and project size, V & V rigour attribute can be used to determine the final project group as relevant. Hence this attribute can be used to promote the project group from lower to higher productivity or de-promote it based on the v& v rigour rating. Values of V & V rigour that will be used to determine project group are Low (rating '0'), High (rating '1') or NC (meaning Not a Concern) for the set of domain and project sizes. The Decision Tree in Figure 4.7 illustrates how Project Group can be identified based on Domain, Project Size and V & V rigour.

Decide Portfolio:

The models in Portfolio A were built from Data Set A, which is a larger data set compared to Data Set B, where there can be some ambiguity in terms of 'architecture' of the project. If the estimator is quite sure about the architecture, then models from Portfolio B can be chosen for estimation.
In case of Portfolios A and B, there is no difference as to which functional size method is used for measuring the functional size. If the estimator uses either COSMIC Function Point or IFPUG Function Point, then models from Portfolios C or D could be chosen for estimation.

Choose the model:

Once the estimator has narrowed down a particular portfolio and project group, then based on the availability of value(s) for independent variables, a particular model can be chosen for project group from portfolios of models (Tables 4.11, 4.13, 4.14 and 4.15). Size is a mandatory independent variable. Model Ids 1, 2,3,10,11,12,19,20,21,22, 23 and 24 uses Size as the only independent variable. Size and DevQ are used in Model Ids 4,5,6,13,14 and 15. All the three variables namely Size, DevQ and TestQ are used in Model Ids 7,8,9,16,17 and 18.

4.7 Evaluation of Estimation Models

4.7.1 Quality of Estimation Models

The quality of estimation models is evaluated using the criteria (Refer Section 3.6.2) such as Coefficient of determination (R^2), Adjusted R^2, Median Magnitude of Relative Error and Mallow's Cp (Table 4.17).

Table 4.17 Evaluation of Models in Portfolios A, B, C and D

Portfolio	Model Id	No. of Projects	R^2	Adj R^2	MedMRE	Mallow's Cp
A (N=142)	1	46	0.82	0.81	0.24	2
	2	49	0.74	0.73	0.27	2
	3	47	0.77	0.79	0.25	2
	4	46	0.85	0.83	0.24	6
	5	49	0.75	0.73	0.28	6
	6	47	0.79	0.77	0.22	6
	7	46	0.86	0.83	0.23	8
	8	49	0.78	0.74	0.24	8
	9	47	0.79	0.77	0.22	6
B (N = 72)	10	32	0.80	0.8	0.24	2
	11	24	0.67	0.66	0.26	2
	12	16	0.83	0.82	0.25	2
	13	32	0.84	0.81	0.22	6
	14	24	0.70	0.62	0.25	6
	15	16	0.91	0.86	0.10	6
	16	32	0.87	0.83	0.20	8
	17	24	0.70	0.57	0.25	8
	18	16	0.91	0.86	0.10	6
C (N = 82)	19	27	0.87	0.86	0.19	2
	20	26	0.73	0.71	0.30	2

Portfolio	Model Id	No. of Projects	R^2	Adj R^2	MedMRE	Mallow's Cp
	21	29	0.82	0.82	0.23	2
D	22	19	0.78	0.77	0.25	2
(N = 60)	23	23	0.76	0.75	0.26	2
	24	18	0.70	0.68	0.33	2

The Value of R^2 for Portfolio A ranges between 0.74 and 0.86 and that of Adj R^2 ranges between 0.72 and 0.83 indicating strong relationship between independent variables - Size, DevQ and TestQ with the dependent variable test effort in all models.

MedMRE value ranging between 0.22 and 0.28 shows that the error levels between estimate and actual are less than 30% for 50% or less of the samples, which is practical considering the multi-organizational data used for building the models. Models using DevQ and TestQ are better compared to Models using DevQ, which are in turn better than Models using just Size as independent variable.

Similar observations can be made for Portfolio B models. Project Group 3 specific models using DevQ and Test Q in Portfolio B have excellent quality with R^2 value of 0.91 and MedMRE of 0.10.

R^2 values for COSMIC based models range from 0.73 to 0.82 while that of IFPUG FP based models range from 0.70 to 0.78 demonstrating strong relationship between test effort and size in all models in both Portfolios C and D.

The MedMRE value for COSMIC based models ranging between 0.19 and 0.30 compared to IFPUG FP based models ranging between 0.25 and 0.33 demonstrate better accuracy of COSMIC FP based models.

The Mallow Cp values indicate that the model is good in predicting future responses based on the variable(s) chosen. In all the models, the Cp value computed is the value required to satisfy the Mallow's Cp Criterion.

4.7.2 Predictive Performance of Models

The criterion used to evaluate the predictive quality of an estimation model is PRED (l) = k/n, where k is the number of projects in a specific sample of size n for which MRE <= l. In the software engineering literature, an estimation model is considered good when PRED (0.25) = 0.75 (Conte, 1986) or PRED (0.30) = 0.70 and PRED (0.20) = 0.80 (Abran, 2015). PRED (0.25) = 0.75 means 75% of the samples should have MRE values less than or equal to 0.25.

While the MRE error level in 75% of the population should be less than 25% - is the expectation of the above criteria, multi-organizational data such as in ISBSG data base are less homogeneous and will exhibit larger MRE for 75% of the population. In a study conducted (Solomon et., al, 2016) to compare estimation results of models from multiple organizations data with single organization data using MedMRE, it has been found that single organizational scored better without normalization and both yielded approximately similar results when data sets were subjected to the z-score normalization technique. When the data is heterogeneous, error for 50% of the population can also be a reasonable indicator of the model predictability. The main requirement in this research context is not just look at the predictability of an individual model, but to compare between models to infer how they differ.
Instead of comparing only at a single upper arbitrary value for PRED (0.25), which relates to 'k/n= ¾ = 0.75', that is the smallest upper range of MRE for 75% of the sample, I propose a new scheme to identify and compare the smallest upper ranges of MRE that would include 50% and 25% of the samples i.e., the smallest upper range of MRE values for 'k/n= ½ = 0.5' and 'k/n= ¼ = 0.25' to gain a better understanding of the error distribution within each estimation model.

The prediction performance of estimation models (Table 4.18) provides the smallest ranges (from 0% to the % indicated in PRED (l) of MRE values for 25%, 50% and 75% of the population which are actually values for the right hand side of the PRED equation. The values within the following tables are MRE values for 25% (represented as Q25), 50% (represented as Q50) and 75% (represented as Q75) of the population.

Considering the Model Id 1 from the (Table 4.18):

Q25 = 0.13

Q50 = 0.24

Q75 = 0.38

which means:

a. MRE value of less or equal to 0.13 is found in 25% of the projects in the samples;
b. MRE value of less or equal to 0.24 is found in 50% of the samples, and
c. MRE value of less or equal to 0.38 is found in 75% of the samples.

Table 4.18 Prediction Performance of Estimation Models

Model ID	# of Data Points	MRE		
		Q25	Q50	Q75
1	46	0.13	0.24	0.38
2	49	0.13	0.27	0.38
3	47	0.11	0.25	0.4
4	46	0.14	0.24	0.39
5	49	0.13	0.28	0.4
6	47	0.08	0.22	0.39
7	46	0.12	0.23	0.35
8	49	0.08	0.24	0.36
9	47	0.08	0.22	0.39
10	32	0.17	0.24	0.35
11	24	0.18	0.26	0.37
12	16	0.05	0.25	0.42
13	32	0.11	0.22	0.42
14	24	0.17	0.25	0.38
15	16	0.03	0.1	0.39
16	32	0.07	0.2	0.38
17	24	0.1	0.25	0.36

Model ID	# of Data Points	MRE		
		Q25	Q50	Q75
18	16	0.03	0.1	0.39
19	27	0.11	0.19	0.35
20	26	0.2	0.3	0.45
21	29	0.1	0.23	0.34
22	19	0.13	0.25	0.34
23	23	0.07	0.26	0.37
24	18	0.13	0.33	0.45

4.7.3 Comparison of Performance of Models

4.7.3.1 Comparison of Models in Portfolio A

There are 9 models in Portfolio A (Table 4.19). A comparison of these nine models can reveal how predictability varies between project groups and those using different independent variables within the context of Data Set A.

Table 4.19 Predictability of Portfolio A Models

Portfolio A Models			MRE		
ID	Name	Description	Q75	Q50	Q25
1	APG1S	PG1 with Size	0.38	0.24	0.13
4	APG1SD	PG1 with Size & DevQ	0.39	0.24	0.14
7	APG1SDT	PG1 with Size, DevQ &TestQ	0.35	0.23	0.12
2	APG2S	PG2 with Size	0.38	0.27	0.14
5	APG2SD	PG2 with Size & Dev Q	0.4	0.28	0.13
8	APG2SDT	PG2 with Size, DevQ & TestQ	0.36	0.24	0.08
3	APG3S	PG3 with Size	0.4	0.25	0.11
6	APG3SD	PG3 with Size & Dev Q	0.39	0.23	0.12
9	APG3SDT	PG3 with Size, DevQ & TestQ	0.39	0.22	0.08

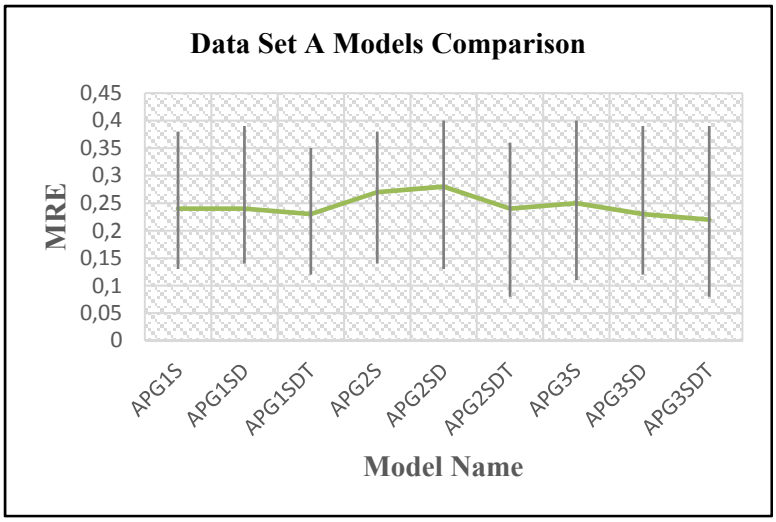


Figure 4.8 Predictability Comparison of Data Set A Models

The chart corresponding to predictability of Portfoio A models (Figure 4.8) depicts how predictability varies between models. Within Project Group 1 models with DevQ and TestQ namely APG1SD and APG1SDT are better than APG1S for Q50. Project Group 2 models APG2S, APG2SD and APG2SDT possess less predictability compared to their counter parts in PG1. However APG2SDT, the model with both DevQ and TestQ is better in PG2. PG3 models have better predictability than others, in general.

4.7.3.2 Comparison of Size Based Models

A comparison of all models using only size as an independent variable across all portfolios can provide under which context size based models provide better predictability. MRE values for these models (Table 4.20) for Q25, Q50 and Q75 are used to plot comparison chart (Figure 4.9).

Table 4.20 Predictability of Size Based Models across Portfolios

Size Based Models			MRE		
ID	**Name**	**Description**	**Q75**	**Q50**	**Q25**
1	APG1S	PG1 in Portfolio A	0.38	0.24	0.13

Size Based Models			MRE		
ID	**Name**	**Description**	**Q75**	**Q50**	**Q25**
2	APG2S	PG2 in Portfolio A	0.38	0.27	0.14
3	APG3S	PG3 in Portfolio A	0.4	0.25	0.11
10	BPG1S	PG1 in Portfolio B	0.35	0.24	0.17
11	BPG2S	PG2 in Portfolio B	0.37	0.26	0.18
12	BPG3S	PG3 in Portfolio B	0.42	0.25	0.05
19	CPG1	PG1 in Portfolio C	0.35	0.19	0.11
20	CPG2	PG2 in Portfolio C	0.45	0.3	0.2
21	CPG3	PG3 in Portfolio C	0.34	0.25	0.13
22	DPG1	PG1 in Portfolio D	0.34	0.25	0.13
23	DPG2	PG2 in Portfolio D	0.37	0.26	0.07
24	DPG3	PG3 in Portfolio D	0.45	0.33	0.13

MRE value for Q75 for the models range from 0.34 to 0.45, Q50 values range from 0.19 to 0.33 and Q25 values range between 0.05 and 0.2.

a. MRE values at Q50 level is consistently higher for Project Group 2 models across all portfolios as revealed by the slope of the line connecting the Q50 values in the comparison chart (Figure 4.5).
b. Size based models for Project Groups PG1 and PG3 are better than PG2.
c. PG3 size models, in general perform better than PG1 and PG2.

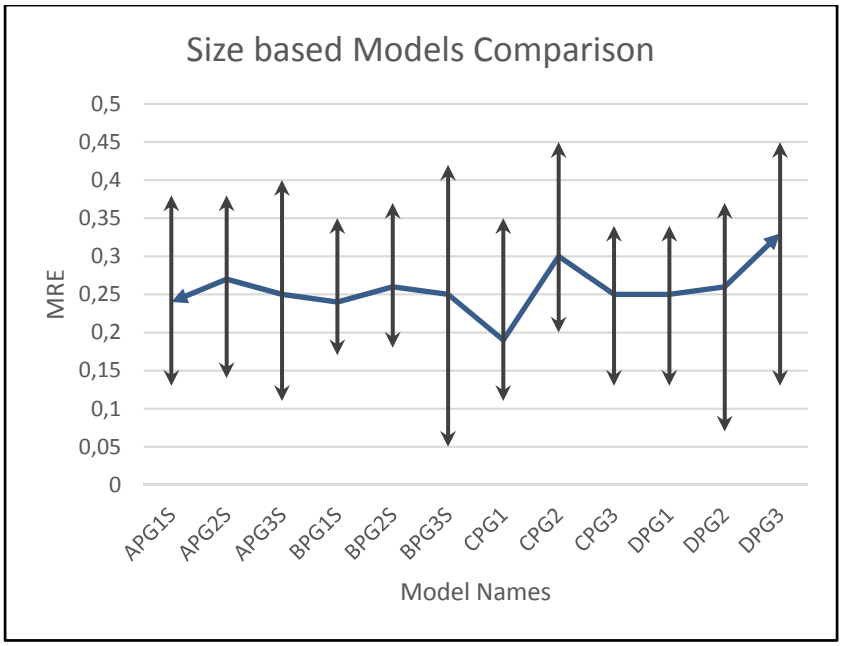


Figure 4.9 Predictability Comparison of Size Based Models

4.7.3.3 Comparison of Models in Portfolios A and B

Portfolio B models are developed using a subset of data used for Portfolio A. Portfolio B models are more specific to Web or Client/ Server architecture unlike Portfolio A models where there is an approximation due to architecture. A comparison between the models across portfolios A and B using DevQ and TestQ as independent variables along with size can help to make certain observations. MRE values for these models (Table 4.21) are used to plot a comparison chart (Figure 4.10)

Table 4.21 Predictability of Portfolio A & B Models

Models			MRE		
ID	**Portfolio**	**Name**	**Q75**	**Q50**	**Q25**
4	A	APG1SD	0.39	0.24	0.14
13	B	BPG1SD	0.42	0.22	0.11
5	A	APG2SD	0.4	0.28	0.13
14	B	BPG2SD	0.38	0.25	0.17

Models			MRE		
ID	Portfolio	Name	Q75	Q50	Q25
6	A	APG3SD	0.39	0.23	0.12
15	B	BPG3SD	0.39	0.1	0.03
7	A	APG1SDT	0.35	0.23	0.12
16	B	BPG1SDT	0.38	0.2	0.07
8	A	APG2SDT	0.36	0.24	0.08
17	B	BPG2SDT	0.36	0.25	0.1
9	A	APG3SDT	0.39	0.22	0.08
18	B	BPG3SDT	0.39	0.1	0.03

MRE values for Q50 varies between 0.1 and 0.28 and that for Q75 varies between 0.35 and 0.42 for the models identified for comparison. Lowest MRE values for Q50 occurs for PG3 model using Size and DevQ variables as well as Size, DevQ and TestQ variables.

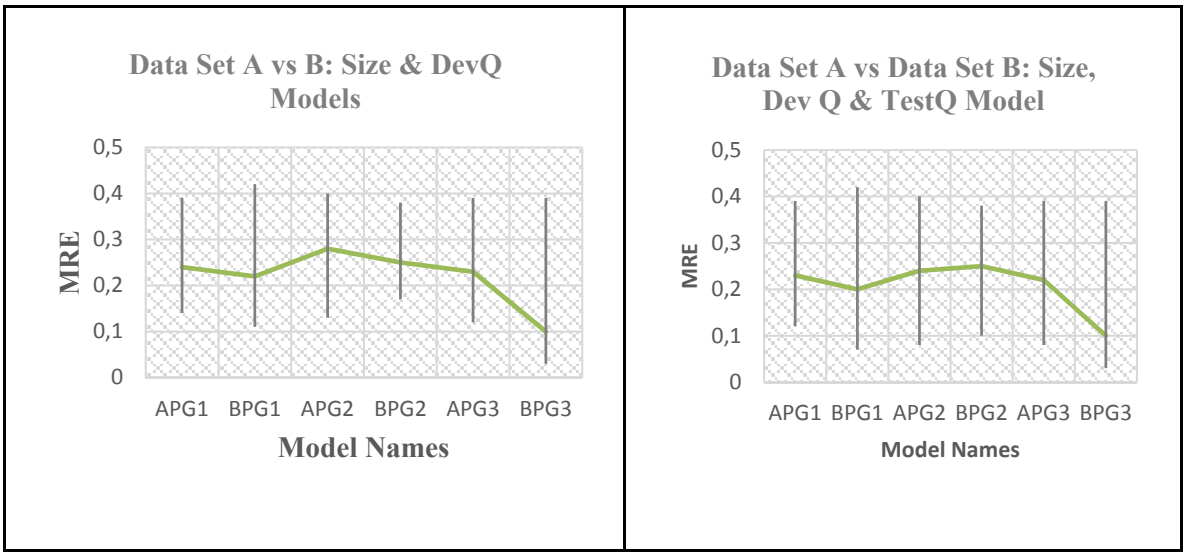


Figure 4.10 Predictability Comparison of Data Set A and B Models

Comparison chart (Figue 4.10) reveals that models from Portfolio B consistently performs better than models from Portfolio A. Models from Portfolio B can result in better prediction accuracy and can be a choice for estimation when the sizing method is not of concern. Models

with DevQ and TestQ as independent variables along with Size perform better than models using DevQ alone along with Size.

4.7.3.4 Comparison of COSMIC and IFPUG Models

Performance of COSMIC and IFPUG models can be compared taking size based models from Portfolio A as reference.

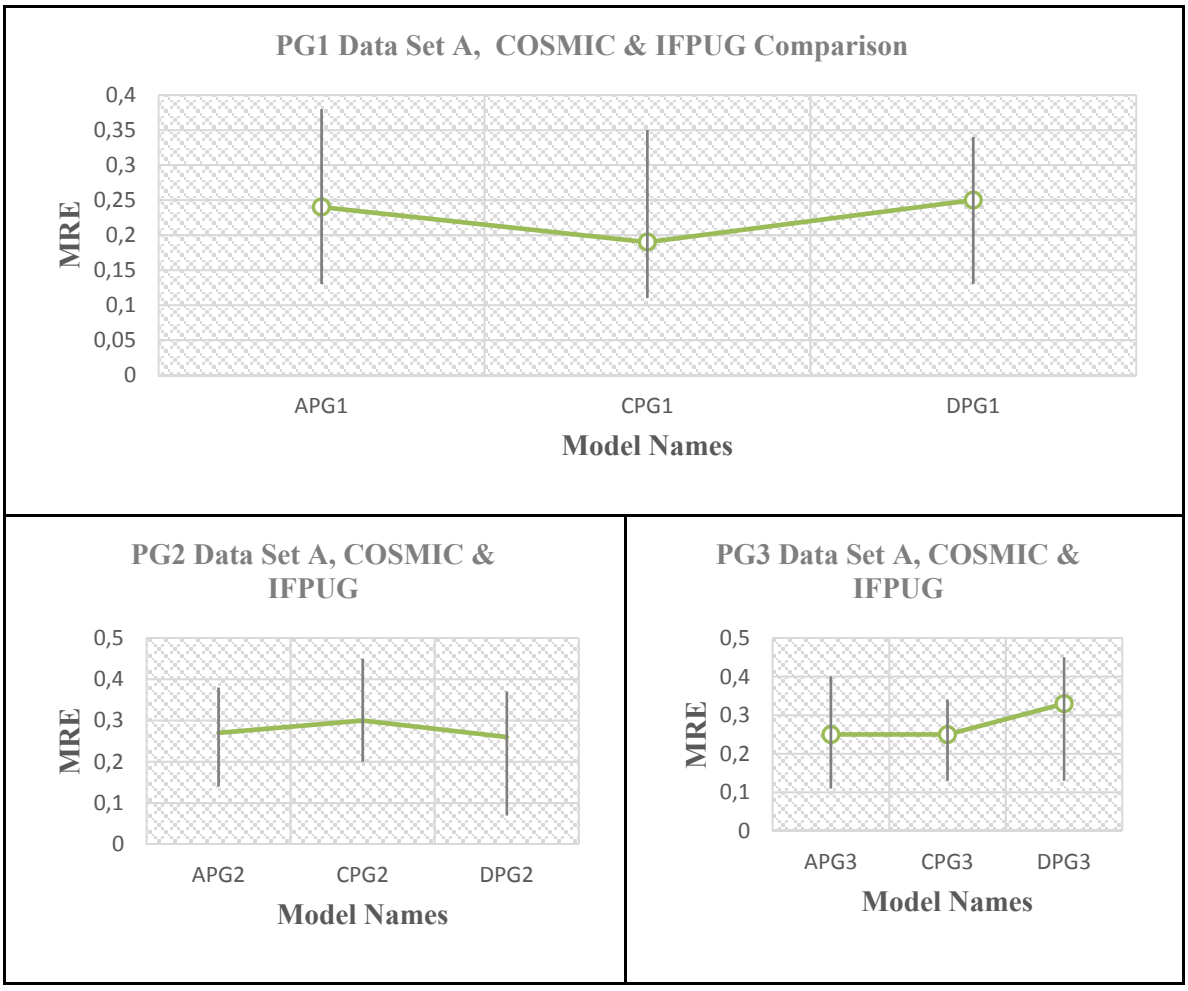


Figure 4.11 Predictability Comparison of COSMIC and IFPUG Models

Both COSMIC and IFPUG data are subsets of Data Set A used to generate models in Portfolio A. This comparison can help to evaluate prediction accuracy of COSMIC based models versus IFPUG based models.

COSMIC based estimation models using Data Set C have better performance than IFPUG based estimation models using Data Set D (Table 4.20) as can be seen from Figure 4.11. With the exception to PG2, Q50 values are always lower for COSMIC based models. COSMIC based PG3 model demonstrates the best predictability with the lowest variation between Q25 and Q75.

4.8 Estimation Tool

A prototype estimation tool has been conceptualized to automate the estimation process using the models generated. The tool, the development of which is in progress will choose a relevant model based on the inputs in terms of project context and values for the independent variables. The tool will have facility to capture actual data as and when projects are executed. Facility to refine the models based on organization specific project execution data and facility to regenerate models as and when such multi-organizational data are available are useful features of this tool. APPENDIX VII provides design details of this tool named as '***Chabroo***'.

CONCLUSION

Summary

This research work has explored the software testing discipline from the perspective of estimation of efforts for testing. The literature study carried out as a part of this work has established the state of the art of software test estimation techniques, along with their strengths and weaknesses. A criterion has been proposed to evaluate the existing techniques. While there are several research papers discussing approaches to test estimation, they have limitations as well. There is hardly any detailed work for practical application to industry use, nor is there the necessary academic rigour. Estimation techniques reviewed in the literature are often conceptual without experimental validity; otherwise they are complex techniques based on a limited data set, yet to be adopted in the industry.

Based on the test estimating components in the literature review, this work has proposed first a Unified Framework for Software Test Estimation for estimating the needs in the software testing arena. Based on this framework, detailed estimation models have been built for functional testing.

The ISBSG database, with its wealth of project data from around the globe, has been used for the first time for estimation of software testing. This data represents the software industry from different countries, and follows standard data reporting conventions. Data from ISBSG has been filtered, to represent current architecture models followed in the industry, especially web and client server. This is to make it possible for the results to be used for many of the current software testing projects.

The analysis of the data has revealed three test productivity patterns representing economies and diseconomies of scale, based on which the characteristics of the corresponding projects have been investigated. Test productivity was measured, using a new terminology defined as Test Delivery Rate. This is the rate at which software functionality is tested as a factor of the effort required to do so.

The three project groups related to the three productivity patterns have been found to be statistically significant; they are characterised by domain, team size, elapsed time and rigour

of verification and validation carried out during development. These patterns are observed in most of the data sets used for building the estimation model, which turns out to be a major finding. They provide an opportunity to build estimation models representing those project groups, instead of building a generic model for all types of projects.

Within each project group, the variations in test efforts could be explained, apart from the functional size, by (i) the processes executed during the development, and (ii) the processes adopted for testing. Two new independent variables, the quality of the development processes (DevQ) and the quality of testing processes (TestQ), were identified as influential in the estimation models. Portfolios of estimation models were built, using combinations of the three independent variables. An estimator could choose the project group, by mapping the characteristics of the project to be estimated to the attributes of project group, in order to choose the model closest to it.

Overall, four portfolios consisting of a total of 24 estimation models were generated. Models were evaluated using standard evaluation criteria, for their fitness for purpose.

The quality of each model was evaluated using established criteria such as R^2, Adj R^2, MRE, MedMRE and Maslow's Cp. As these models were built from ISBSG data, they could serve as an industry benchmark for functional test efforts. The quality of models improved when more homogeneous data from ISBSG was used, as seen during the comparison of Portfolio A and Portfolio B.

The design of these models can serve as Meta Model for building proprietary test estimation models, using data from within an organization. Such models can be more accurate than the benchmark models built using heterogeneous data from ISBSG.

Test estimation models using projects measured in COSMIC Function point exhibited better quality and resulted in more accurate estimates compared to projects measured in IFPUG Function Point. Adopting COSMIC FSM is a better choice, when accuracy of test effort estimate is of paramount importance.

A prototype test estimation software was developed using the statistical programming language “R”, incorporating portfolios of estimation models generated during this research work. This tool can be used by industry and academia for estimating test efforts.

Research Objectives Revisited

The research objectives selected for this research project were to build an estimation model for:

1. Estimating the effort for functional testing.
2. Serving the needs for benchmarking and performance measurement of software testing projects
3. Automation capability that can be deployed as an estimator’s tool, for use by industry and academia.

Estimating Test Effort

This consisted of:

1. Identification of the 1 to 3 factors that contribute most to the relationship with efforts for functional testing. The research has designed estimation models based on three parameters: (i) Functional Size, (ii) Quality of development processes – DevQ, and (iii) Quality of test process – TestQ. This means that
   a. 12 models were built using Size as independent variable,
   b. 6 Models were generated using Size and DevQ, and
   c. 6 Models were generated using Size, DevQ and TestQ.

These models were evaluated using standard evaluation criteria.

2. Arriving at a model that can be used during the early stages of software testing.

The early stage of software testing is when only the requirements of the software to be tested are known. All the models developed as a part of this research work require functional size as

a major input, which can be measured during the early stage in life cycle. Other attributes of the project related to development, such as ‘domain’ of the software, ‘size of team’, ‘duration of the project’ and ‘verification and validation rigour’, can be known at the beginning of testing. The estimation model that corresponds to the project group closely matching the attributes of the software to be tested can be chosen.

Benchmarking and performance measurement

Apart from the basic need to produce reliable estimates, an organization has to be competitive. This is required, to raise its performance to be on par with competitors in the market. Currently there are no International benchmarks available for comparing and benchmarking, except certain unreliable thumb rules like 20% to 30% of development efforts being assigned for testing.

The current work has used a data set representative of the industry from ISBSG. The estimates from models generated out this data set can serve as a benchmark reference for software test estimates, just like models made for development from ISBSG serve as benchmarking reference for development efforts. The models generated support the benchmarking needs for testing, and hence fulfil a major gap in the engineering approach to software testing.

Automation Capability

A prototype tool using statistical programming language “R” has been designed, to automate estimation. This tool will use model equations rather than the actual project data points, thus maintaining the confidentiality of project data used for generating models. To choose an appropriate model, the tool takes as inputs the project context in terms of size, domain, team size, elapsed time, and V & V rigour. Advanced features that would be part of the tool in future include generation of new models based on actual values from within an organization. It would also include re-generation of models, based on larger data sets from multiple organizations.

Contributions

The research outcome contributes positively to three primarily different segments: software organizations, researchers in estimation and the software benchmarking community.

Software Organizations

The industry follows mostly judgement based estimation techniques and rules of thumb for testing estimation. Another approach followed is the allotting of a certain percentage of software development effort to software testing; this is based on thumb rules. Because testing has become a separate discipline and often carried out independently from development, it is essential that scientific methods be designed and adopted, for estimation and performance measurement.

The current research work has provided a set of four portfolios consisting of 24 estimation models, which can be used for estimating functional test efforts for different project contexts. The models can be fine-tuned with organization specific data, to improve the accuracy of estimation. Tools can be developed based on the models, for the estimator to (i) quickly and consistently carry out estimation, and (ii) capture the actual project performance data as and when tests are carried out.

A freely downloadable version of the estimation tool to be developed based on this research work will be made available to the organizations.

Researchers in Estimation

This research has focused on the estimation of effort for software testing. The principles involved in estimation and techniques adopted can be used, in general, for estimating any specific phase of development such as software testing. For instance, ‘construction’ (or otherwise referred to as ‘coding’) is one of the phases outsourced by the industry, and any specific estimation for construction can also adopt a similar approach. Researchers can review the estimation models and parameters, to build coding specific estimation models.

The research work has proposed a Unified Framework, for addressing estimation for different types of testing. This framework has also suggested an innovative sizing and estimation approach to business process testing, which can be explored further by researchers.

Due to the lack of detailed parameters in ISBSG data with respect to testing related activities, there is no opportunity to derive TestQ rating levels in a refined manner. Researchers can further review the process ratings such as DevQ and TestQ, and modify their impact on functional test effort. The evaluation of estimation models presented provides confidence to researchers on the applicability of simpler techniques within the context of a specific group of projects, resulting in practical solution. The research work has also come out with a new criterion from the perspective of managers and estimators, to evaluate performance of models based on Q25, Q50 and Q75 values of MRE. This criterion can be adopted while dealing with multi-organizational data that tends to be heterogeneous.

Benchmarking

This research work has defined a new term – 'Test Delivery Rate' – to deal with productivity in testing. This provides scope for measuring productivity of testing in different projects within an organization, or across organizations, in order to benchmark productivity.

All current testing productivity measurements use either lines of code or other implementation aspects of software such as screens etc., to work out related measures. TDR based measurement can enable definition of metrics for test case design, test execution and test automation. ISBSG, which maintains the world's largest repository of software projects, has not yet published any benchmarks for testing related activities. ISBSG and/or other measurement organizations can initiate data capture mechanism for testing projects, which can provide a strong basis for benchmarking of testing projects.

Research Impact

The impact of this research on the community can be viewed from both short term and long term perspectives.

Short Term Impact

A consistent way to estimate efforts for functional testing, based on the project characteristics, has laid a foundation for estimation with a scientific basis. Industry can use the models for test effort estimation straightaway. The accompanying estimation tool will ease the use of models, besides providing a mechanism to capture data to improve accuracy of estimations in future within an organization.

Organizations will also be able to measure the performance of the testing projects, and compare with that of other organizations that have contributed data to the repository. It is also expected that mature organizations will use the estimation model as a reference meta model, and build proprietary models with additional/different factors that influence test efforts in their organizations.

Researchers could further classify project groups, with respect to enhancement projects or real-time projects, and generate similar estimation models with factors influencing test productivity. The software measurement community could immediately start using TDR as one of the criteria to evaluate testing projects.

Long Term Impact

The current models are generated from ISBSG data. Using these models to capture organization specific project performance data, still better models can be generated with a view to improving estimates within organizations. With wide spread adoption, the data from multiple organizations maintained in a central repository by benchmarking organizations can pave the way for periodically regenerating the models for similar and new contexts.

The future versions of the estimation tool will provide estimation as a cloud based service, making automated estimation affordable and available at any time. An online repository of global project data that can be built using this approach can open up a new era in benchmarking.

Test estimation models for Business Process Testing, Modifications Testing, Convertible NFR, True NFR testing and Test Automation proposed in the Unified Framework can be explored further by researchers, to come out with specific estimates for each type of testing. Once the models are generated with a large number of data points from multiple organizations representing different geographies, these models can serve as reference in case of disputes with respect to efforts put in testing.

Limitations of this Research and Future Work

While this research work has taken a major step in the hitherto untouched area of software test effort estimation based on ISBSG data, there are limitations due to the approach and validity threats, both internal and external.

The research project has come out with an estimation model for 'functional testing'. The models generated are applicable for both development and re-development projects, but not for enhancement projects. The models are currently applicable only for business application testing projects. These limitations can be overcome by generating specific models for enhancements or real-time projects, using an approach similar to one followed in this research work. This may require identification of additional project characteristics, as well as other variables influencing testing effort.

The current models are applicable for software whose functional sizes are within the range of functional sizes present as a part of data set, and cannot be extrapolated beyond this range. Going by the current trend in the software development, influenced by agile project management approach, the estimation is usually done for smaller scopes and hence this limitation may not affect its use in the industry.

The current work has come out with three categories of projects based on attributes such as domain, team size, elapsed time and V & V rigour. Each one of them is a categorical variable. Estimation users must be able to map the characteristics of the software project tested to a relevant category, in order to come out with a value for each of the attributes. There is a possibility that some approximation takes place during such mapping, affecting the estimates. The fourth group of project data points, which were not considered for building estimation

models in this research work, are to be analyzed further to study the possibility of building models.

The research work also has identified DevQ and TestQ as additional variables influencing testing efforts, apart from functional size. These process factors can be studied further within organizational context, and selectively used. There could be other variables that can influence test efforts in specific contexts, which require further study and analysis.

The data used for building estimation models was taken up from the ISBSG database, which has the provision to capture data related only to development projects. However, a majority of the projects have reported data related to testing activities, enabling estimation models for software testing. Applying models built from this data set to purely independent testing assignments executed by a separate contractor would not be straightforward. However, these models could provide a reference for test efforts, which can be further adjusted considering the cost drivers related to independent testing projects. There is an observation (Bareja K and Singal A, 2015) that testing techniques using machine learning and data mining help to reduce the effort required to test. The estimation models designed can be further refined considering testing techniques adopted as a parameter in order to evaluate their impact and use them in building estimation model.

Functional testing is the foundation for all other types of testing; however, models for business processes testing, with end to end integration of upstream and downstream enterprise business processes, have not yet been built. An elegant approach to handle this has been proposed, as a part of the Unified Framework. The approach can be used with organization specific data points, to build estimation models for business process testing. The estimation models generated do not address the needs of Modifications Testing, Convertible NFR Testing, True NFR testing and Test Automation. Further work can be carried out to develop solutions to these types of testing aligned with the Unified Framework for test effort estimation designed as a part of this research work.

APPENDIX I

DATA SELECTION AND OUTLIERS

This appendix provides details of how the data is selected from ISBSG data set by applying various selection criteria and, next, removing outliers.

Data Selection

The data selection criteria (Refer 4.3.1) applied on the ISBSG R12 data set consisting of 6006 projects data, resulted in a data set consisting of 193 data points (Table A I – 1).

Table A I -1 Data Selection Filters

Selection Criteria	**Filter**	**Filter Details (items removed)**	**No. of Projects Removed**	**Remains**
ISBSG R12 Data	Original Data	-	-	6006
Data Quality	ISBSG Data Quality Rating	C, D	448	5558
Data Quality	Test Effort	Blank	4080	1478
Data Quality	UFP Rating	C, D	147	1331
Data Adequacy	Application Group	Real-time, Infrastructure	91	1240
Data Adequacy	Development Type	Enhancement	643	597
Data Relevance	FSM	Other than COSMIC & IFPUG 4+	161	436
Data Suitability	Normalized effort	< 80 hrs	2	434

Selection Criteria	Filter	Filter Details (items removed)	No. of Projects Removed	Remains
Data Suitability	Test Effort	< 16 Hrs only	16	418
Data Relevance	Application Group	Blank	138	280
Data Relevance	Architecture	Stand alone	75	205
Data Suitability	Test Activity	Non Functional Testing	12	193

Identification of Outliers

Projects with very low Test Delivery Rate, such as less than 0.3 hrs/ FSU, are an indicator of inadequacy of testing for the functionality. Such projects cannot serve as good candidates for building estimation models for testing. Six projects were found to be in this category and were removed.

Identification of outliers based on John Tukey's Inter Quartile Range (IQR) statistical criteria (Below $Q_1 - 1.5 \times IQR$ or above $Q_3 + 1.5 \times IQR$) for the size value resulted in all projects with sizes above 940 FSU showing up as outliers. There were 17 project data points having sizes above 940 FSU. They were removed from the data set considered for estimation model. Overall 23 data points were eliminated (Table A I – 2) resulting in 170 data points.

Table A I - 2 Outliers

Selection Criteria	Filter	Filter Details	No. of Projects Removed	Remains
Data Suitability	Very Low TDR	TDR < 0.3	6	187
Data Suitability	Statistical outliers	Size > 940 removed	17	170

A box plot (Figure A I - 1) of size prior to the removal of outliers and after the removal of outliers depicts the distribution of size values. Outliers increase the mean value of size by 173% and the mean value of effort by 140%. Further, the maximum value of size goes up by 5.5 times and the maximum value of effort goes up by 1.6 time with the outliers.

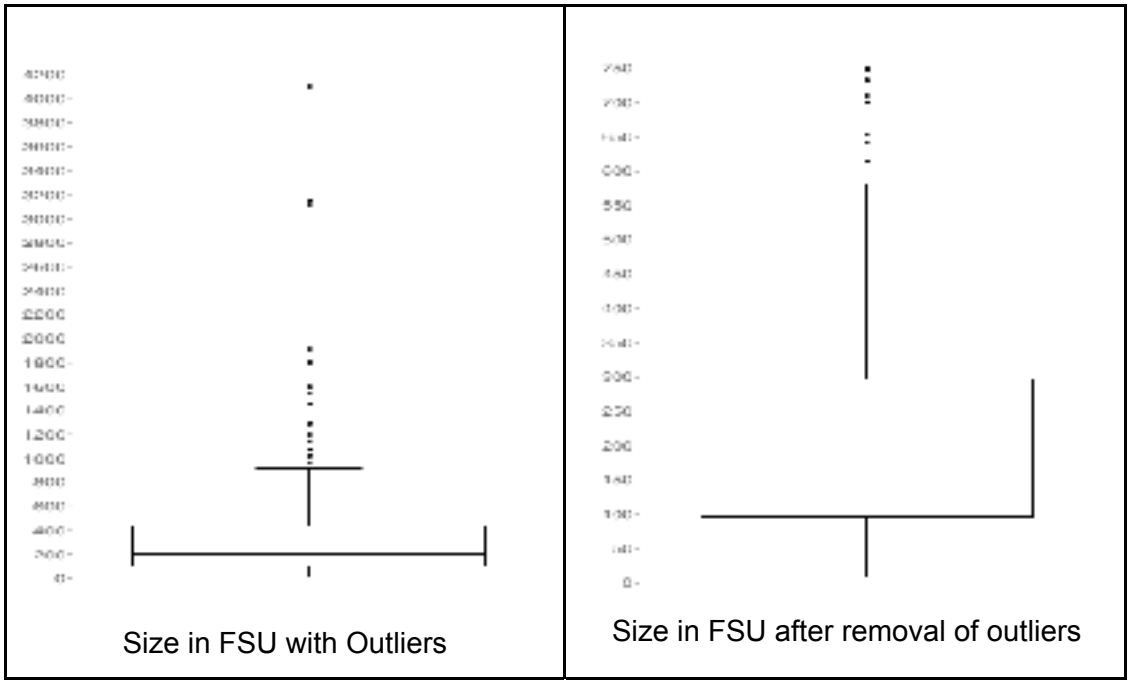


Figure A I -1 Box Plot of Size before and after removal of 23 outliers (N=193)

Identification of outliers within individual TDR Levels

Scatter diagrams of individual TDR Levels provide the dispersion of the data within each individual TDR level.

The data point (923, 718), stands out in the scatter diagram for TDR Level 1 (Figure A I - 2). There are no adequate data between size value 923 and its immediate next lower size value of 751. Absence of data between these size values will affect the accuracy of the model, hence the one data point (923, 718) is removed from the data set considered for further analysis.

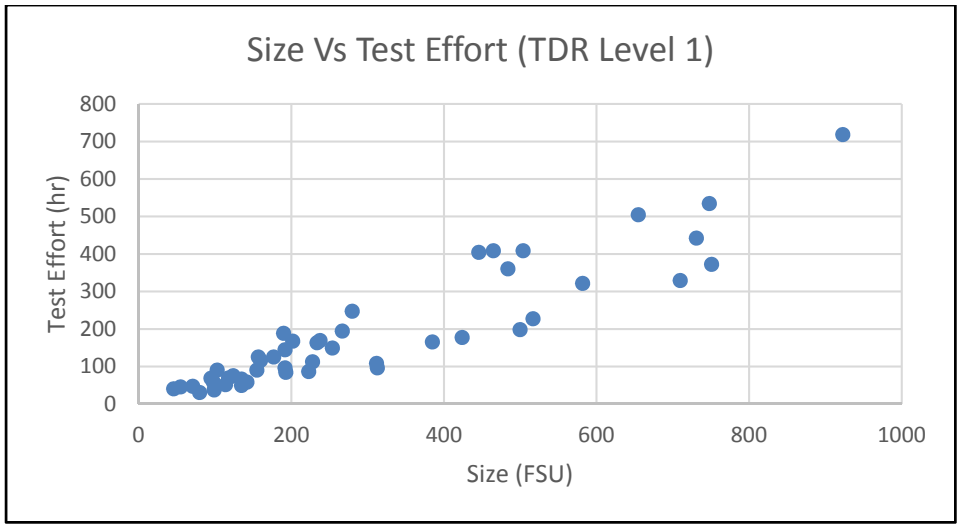


Figure A I – 2 Scatter Diagram of TDR Level 1 Data Points

Observing the scatter diagram (Figure A I - 3) corresponding to TDR Level 3, four data points which can affect the model quality are removed. An observation of size and test effort values for these four points - (278, 2813), (556,1978), (475, 4276) and (550,5600) reveal that dispersion of test effort values are quite high compared to rest of the points in the data set.

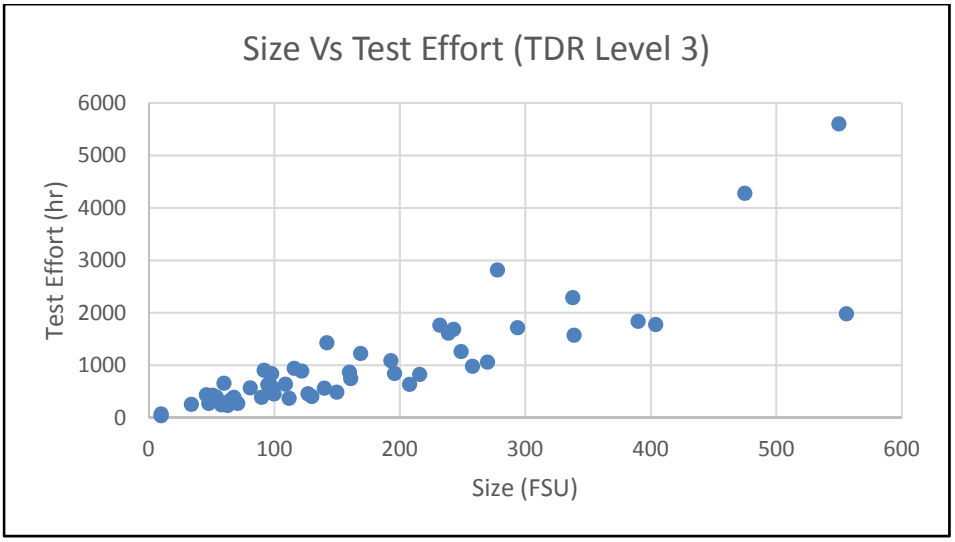


Figure A I – 3 Scatter Diagram of TDR Level 3

An analysis of the TDR distribution of TDR Level 4 (Figure A I - 4) in terms of percentile values– 10th Percentile (P10), 25th Percentile (P25), 50th Percentile (P50), 75th Percentile (P75) and 90th Percentile (P90) in the data set within each TDR level (Table A I - 3).

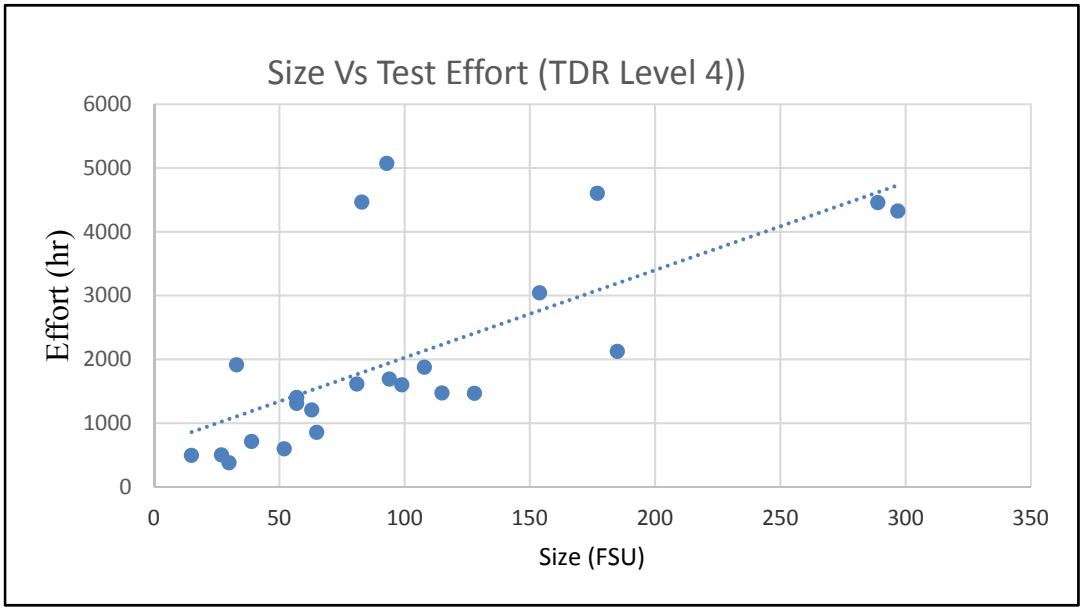


Figure A I – 4 Scatter Diagram of TDR Level 3

a. TDR is uniformly distributed in TDR levels 1, 2 and 3. Median TDR for these levels are more or less 50% of the maximum TDR for the level except for TDR level 4.
b. TDR level 4 projects appear quite odd as their minimum test effort is 7 – 12 times more than other levels and maximum effort is 2 – 10 times more than that of other levels, while the maximum functional size is 0.4 – 0.7 times of other levels.
c. The 23 data points in level 4 category consists of small projects in terms of functional size and consuming very large test efforts compared to the other projects.

Table A I - **3** TDR Distribution for various percentiles (N = 170)

TDR Level	TDR						
	Min	**P10**	**P25**	**P50**	**P75**	**P90**	**Max**
ALL	0.31	0.49	0.87	2.28	6.29	14.90	57.97
1	0.31	0.37	0.44	0.60	0.75	0.87	0.99
2	1.04	1.14	1.34	1.67	2.33	2.87	2.97
3	3.04	3.52	4.00	5.61	7.23	8.71	10.88
4	11.47	11.48	13.15	18.26	24.60	54.23	57.97

TDR Level 4 points are not considered for further analysis and building estimation model.

One data point from TDR Level 1 and 4 data points from TDR Level 3 and all the 23 data points related to TDR Level 4 totalling 28 data points were removed (Table A I - 4) resulting in 142 data points.

Table: A I – 4 Data points removed in within TDR Levels

Selection Criteria	**Filter**	**Filter Details**	**No. of Projects Removed**	**Remaining**
Data Suitability	TDR Level 1 Outlier	Size > 923	1	169
Data Suitability	TDR Level 3 Outliers	Effort > 1978	4	165
Data Suitability	PG4 Data Points	TDR > 11	23	142

APPENDIX II

MAPPING OF V & V RIGOUR TO ISBSG DATA FIELDS

This appendix details out how the V & V Rigour rating is derived from relevant fields of ISBSG data repository.

V & V Rigour refers to the extent of verification and validation activities carried out during development. This attribute of project is derived from the values for data fields related to 'Documents & Techniques' category in ISBSG that can indicate the rigour followed during verification and validation activities. The Document and Techniques grouping of ISBSG data fields (Table A II – 1) deals with methodologies and techniques used during development, phase wise artefacts produced and activities carried out.

Table A II – 1 Extract of ISBSG Documents & Techniques Data Fields

ISBSG Grouping	**Document and Techniques**		
ISBSG Column No	BL	BN	BP
Column Title	Specification Techniques	Design Techniques	Build Activity
Typical V & V related value	Specification Review	Design Review/ Inspection	Code Review/ Inspection; Unit Testing

Specification review, Design review/ Inspection, Code Review/ Inspection and Unit Testing activities are the verification and validation activities carried out in a project. It has been observed that for some projects one or two of these fields carry values, for a few projects most of these fields have values, and for several projects these fields remain blank.

A scheme has been devised (Table A II – 2) to rate the rigour of V & V based on how many of the verification and validation activities are carried out in a project.

Table A II – 2 V & V Rigour Rating Scheme

V & V Rigour Rating	**Description**
Low	None or very little evidence of Reviews/Inspection to infer the rigour.
High	Reviews/ Inspection reported for at least one of the specification, design and build phases.

APPENDIX III

PROJECT CHARACTERISTICS ANALYSIS - DATA SET B, C & D

This appendix provides details of the analysis of project characteristics based on Data Set B, C and D. The project attributes, such as Domain, Team Size, Elapsed Time and V & V Rigour (Section 4.4.3) taken up for the analysis, are the same as those considered for Data Set A.

Analysis of Data Set B

Data Set B is derived from Data Set A by eliminating data where architecture type did not have any value. This data set represents Web and Client/Server projects. Data Set B consists of 72 data points. Analysis of this data set (Table A III-1) with respect to Project Groups and Project attributes provides a picture of the characteristics of individual project groups.

Table A III-1 Analysis of project characteristics – Data Set B

Domain	No./ %	PG 1	PG 2	PG 3	Team Size	No./%	PG 1	PG 2	PG 3
BFSI	Count	4	5	5	Small	Count	7	3	1
	%	29	36	36		%	64	27	9
Education	Count	11			Medium	Count	12	5	4
	%	100	0	0		%	57	24	19
Government	Count	4	9	2	Large	Count	3	4	5
	%	27	60	13		%	25	33	42
Elapsed Time	**No./ %**	**PG 1**	**PG 2**	**PG 3**	**V & V Rigour**	**No./%**	**PG 1**	**PG 2**	**PG 3**
Small	Count	16	5	3	Low	Count	12	19	12
	%	67	21	13		%	28	44	28
Medium	Count	5	3	4	High	Count	20	5	4
	%	42	25	33		%	69	17	14
Large	Count	6	12	7					
	%	24	48	28					

The number of projects and percentage of projects falling within each of the Project Group for all the four attributes are reviewed (Table A III – 1).

All the projects in the educational category fall into PG1, while 60% of government projects fall into PG2. 72% of the BFSI projects are equally spread between PG2 and PG3.

As far as Team Size is considered, small team size contributes close to 2/3rd of the projects in PG1. A little less than 60% of the projects of medium team size also fall into PG1 and 3/4th of the projects with large team size are spread between PG 2 and PG3.

Almost two third of projects in PG1 fall into the low elapsed time category. 67% of medium elapsed time projects fall into PG1 and PG2. PG 2 and PG3 share the majority of the projects with a large elapsed time.

Closer to 2/3rd of the projects are in V & V rigour rating high. Low V & V rigour rating is predominant in PG2 and PG3 project groups.

The three project groups within Data Set B have certain distinctions with respect to Team Size, Elapsed Time, V & V Rigour and Domain.

Table A III – 2 Statistical Significance of Project Attributes within Data Group B

Attribute/ Statistical Test	**Team Size**	**Elapsed Time**	**V & V Rigour**	**Domain**
Chi-Square P Value	0.258	0.039	0.003	< 0.001

The Chi-Square Test conducted (Table A III - 2) on the three project groups with respect to these attributes demonstrates good significance for domain and reasonably significant for Elapsed Time and V & V Rigour.

Analysis of Data Set C

Data Set C is made up of only projects where functional size is measured using the COSMIC Function Points method. This data set is subset of Data Set A and consists of 82 data points. This data set was analysed (Table A III - 3) to understand the characteristics of project groups

with respect to the project attributes such as domain, team size, elapsed time and V & V Rigour.

Table A III - 3 Analysis of project characteristics – Data Set C

Domain	No./ %	PG 1	PG 2	PG 3
BFSI	Count	4	14	27
	%	9	31	60
Education	Count	11	0	0
	%	100	0	0
Government	Count	1	5	0
	%	17	83	0

Team Size	No./%	PG 1	PG 2	PG 3
Low	Count	6	2	0
	%	75	25	0
Medium	Count	10	0	0
	%	100	0	0
Large	Count	1	1	3
	%	20	20	60

Elapsed Time	No./ %	PG 1	PG 2	PG 3
Low	Count	13	1	3
	%	76	6	18
Medium	Count	4	4	2
	%	40	40	20
Large	Count	6	11	7
	%	25	46	29

V & V Rigour	No./%	PG 1	PG 2	PG 3
Low	Count	9	22	28
	%	15	37	47
High	Count	18	4	1
	%	78	17	4

While most of the projects in the educational domain fall into PG1, closer to two thirds of BFSI projects fall into PG3 and majority of the Government projects fall into PG2.

Most of the small and medium team size projects fall into PG1 and 60% of the large team size fall into PG3. Over 3/4th of small elapsed time projects fall in PG1, while 75% of the projects with medium and large elapsed time are spread between PG2 and PG3.

78% of the projects with high rating for V & V fall in PG1 and lower rating are spread between PG2 and PG3.

Table A III – 4 Statistical Significance of Project Attributes within Data Group C

Attribute/ Statistical Test	**Team Size**	**Elapsed Time**	**V & V Rigour**	**Domain**
Chi-Square P Value	0.003	0.018	< 0.001	<0.001

The test of significance carried out (Table A III – 4) reveal very high significance for attributes Domain, V & V Rigour and Team Size and comparatively less significance for Elapsed Time. As in other Data Sets A and B, these attributes are potential contributors for varying productivity levels across project groups.

Analysis of Data Set D

Data Set D is formed considering only projects where the functional size is measured using the IFPUG method. This data set is another subset of Data Set A and consists of 60 data points. Distribution of the projects into the three project groups based on the project attributes is analysed (Table A III – 5) to understand the pattern.

Unlike the previous data sets IFPUG project data do not demonstrate a clearly visible pattern with respect to project attributes. All project attributes are spread across all project groups without concentration on any one of the project groups. In many cases they are more or less equally spread across all project groups. Chi Square test (Table A III – 6) also do not conform statistical significance.

Data Sets B and C demonstrate pattern similar to Data Set A, while Data Set D does not fall into that pattern. By reviewing the pattern, a common approach to mapping characteristics of a project to be estimated to a Project Group can be arrived.

Table A III - 5 Analysis of project characteristics – Data Set D

Domain	No./ %	PG 1	PG 2	PG 3	Team Size	No./%	PG 1	PG 2	PG 3
BFSI	Count	10	9	5	Small	Count	4	2	2
	%	42	38	21		%	50	25	25
Education	Count	5	5	2	Medium	Count	8	14	7
	%	42	42	17		%	28	48	24
Government	Count	15	14	7	Large	Count	4	3	5
	%	42	39	19		%	33	25	42
Elapsed Time	No./ %	PG 1	PG 2	PG 3	V & V Rigour	No./%	PG 1	PG 2	PG 3
Small	Count	5	6	2	Low	Count	16	20	15
	%	38	46	15		%	31	39	29
Medium	Count	2	4	5	High	Count	3	3	3
	%	18	36	45		%	33	33	33
Large	Count	8	9	9					
	%	31	35	35					

Table A III – 6 Statistical Significance of Project Attributes within Data Group D

Attribute/ Statistical Test	**Team Size**	**Elapsed Time**	**V & V Rigour**	**Domain**
Chi-Square	0.466	0.568	0.943	0.948
P Value				

APPENDIX IV

MAPPING OF DEVQ, TESTQ RATINGS TO ISBSG DATA FIELDS

DevQ and TestQ are quality rating for the development processes and test processes. These ratings can be arrived at by reviewing the values from the 'Documents and Techniques' grouping of ISBSG data fields. This appendix details out how ISBSG fields are mapped to these rating.

Development Process Quality Rating (DevQ)

The process followed during the development can be rated considering the nature of the development life cycle followed and the artefacts produced, using the following attributes of the project:

- Standards followed,
- Distinct development life cycle phases followed, and
- Verification activities carried out during development.

The ISBSG data grouping 'software process' (Table A IV - 1) provides an indication of the application of CMMI, ISO, SPICE, PSP or any such standards/models followed during the development.

Table A IV – 1 ISBSG Grouping Software Process (Fields AR to AV)

Column No	Column Title	Nature
AR	CMM	Quality of Dev Process
AS	SPICE	Quality of Dev Process
AT	ISO	Quality of Dev Process
AU	TickIT	Quality of Dev Process

A set of fields representing 'Documents and Techniques' (Table A IV - 2) in ISBSG data provide information on the life cycle phases followed and verification activities carried out during the development, an indication of development processes.

Table A IV – 2 ISBSG Grouping Documents & Techniques (Fields BK to BP)

Column No	**Column Title**	**Nature**
BK	Specification Document	Engineering artefact
BL	Specification Techniques	V & V techniques
BM	Design Document	Engineering artefact
BN	Design Techniques	V & V techniques
BO	Build Products	Engineering artefact
BP	Build Activities	V & V techniques

A DevQ rating between 0 - 2 has been derived (Table A IV – 3) by reviewing the values for both Software Process and Document & Techniques grouping.

Table A IV - 3 Rating for Development Process (DevQ)

Software Process	**Documents & Techniques**	**DevQ Rating**
Not reported	Very little reporting to infer	0
Reported	Very little reporting to infer	1
Not reported	One or more phases has values	1
Reported	One or more phases has values	2

Test Process Quality Rating (TestQ)

While reviewing the data related to the testing process followed, it was observed that there were not enough fields in ISBSG data to capture the details of the testing process followed (such as testing techniques adopted, levels of testing executed, test artefacts produced, reviews of test cases etc.) to gauge the extent of testing. However, data fields BQ and BL (Table A IV – 4) provided information to infer quality of testing process.

Table A IV – 4 ISBSG Grouping Documents & Techniques (Fields BQ & BR)

Column No	Column Title	Nature
BQ	Test Document	Testing artefact
BL	Test Activity	Testing Techniques

It was possible to classify the test process rating broadly into two categories (Table A IV - 5).

Table A IV - 5 Rating for Test Process (TestQ)

Test Process Criteria	Test Process Rating (TestQ)
No evidence of Test Artefacts	0
Evidence of Test Artefacts	1

APPENDIX V

ANALYSIS OF DEVQ AND TESTQ FOR DATA SET B, C & D

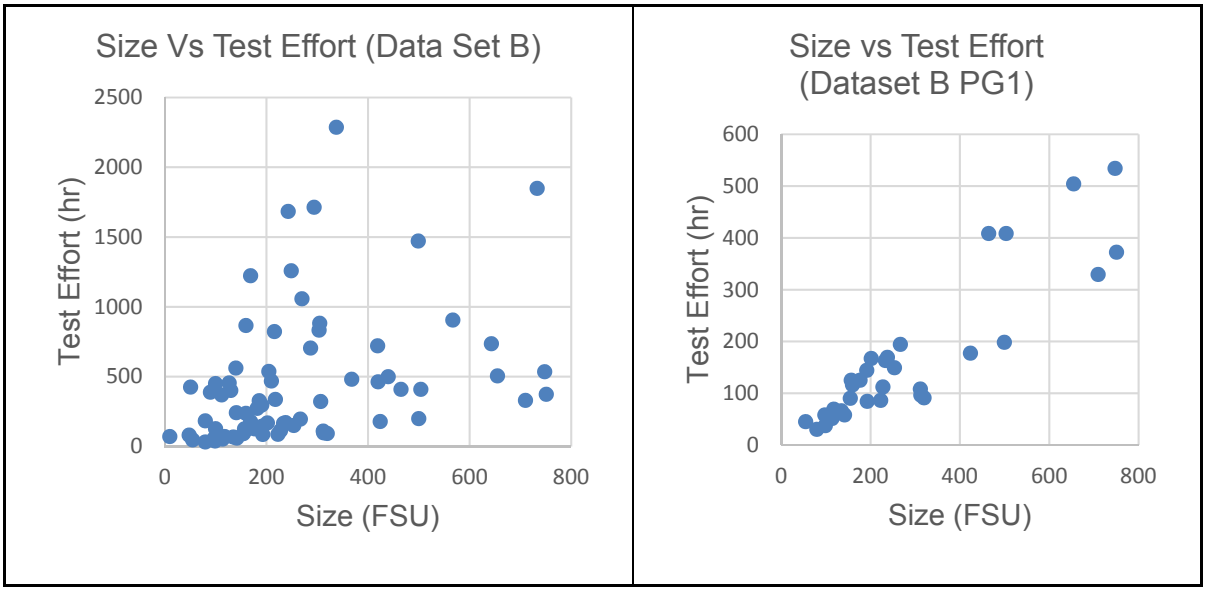
This Appendix V discusses the analysis of the independent variables - Size, DevQ and TestQ - for incorporating them into estimation models based on Data Sets B, C and D.

Size refers to the functional size measured using either IFPUG Function Points or COSMIC Function Points.

DevQ refers to the rating of the process followed during the development based on the nature of the development life cycle followed and the artefacts produced. TestQ refers to the rating of the process followed for testing based on test artefacts produced.

Data Set B:

Scatter Diagrams of Size versus Test Effort for the whole Data Set B and individual project groups PG1, PG2 and PG3 (Figure A V - 1) illustrate the relationship between size and test effort for each of these data sets.



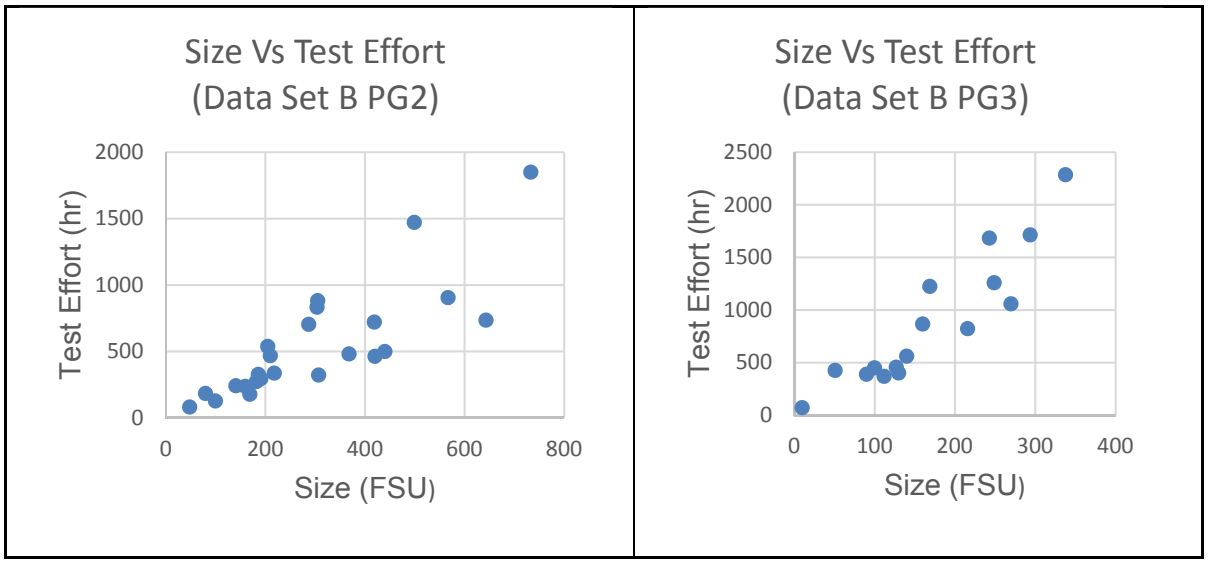


Figure A V - 1 Scatter Diagrams for Size vs Test Effort: Dataset B - PG1, PG2 and PG3

A correlation coefficient computed between Size and Test Effort in the data sets (Table A V - 1) especially for PG1, PG2 and PG3 indicate good correlation; size can be included as the primary independent variable.

Table A V - 1 Correlation: Size Vs Test Effort – Dataset B

Data Set	Correlation Coefficient
Data Set A	0.3591
Data Set A PG1	0.8952
Data Set A PG2	0.8212
Data Set A PG3	0.9134

Size being the main independent variable, other independent variables are examined next for their significance to incorporate them into an estimation model.

Analysis of DevQ and TestQ for Data Set B

The classification of the data set in terms of DevQ and TestQ rating reveals that 29%, 42%, and 29% of the projects are of DevQ Ratings 0, 1 and 2 respectively. TestQ percentages are 64% and 36% for TestQ ratings 0 and 1 respectively (Figure A V - 2).

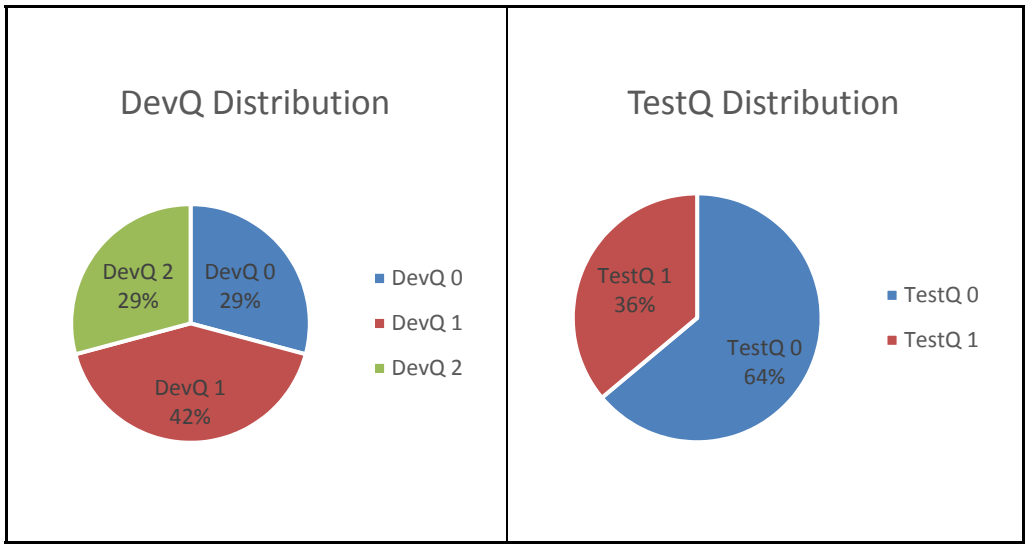


Figure A V - 2 Distribution of DevQ and TestQ Ratings (N = 72)

The different median test effort and interquartile ranges for DevQ values of 0, 1 and 2 (Figure A V - 3) reveals the effect of DevQ on test effort. Median test effort and the interquartile range is also the lowest for the highest DevQ rating.

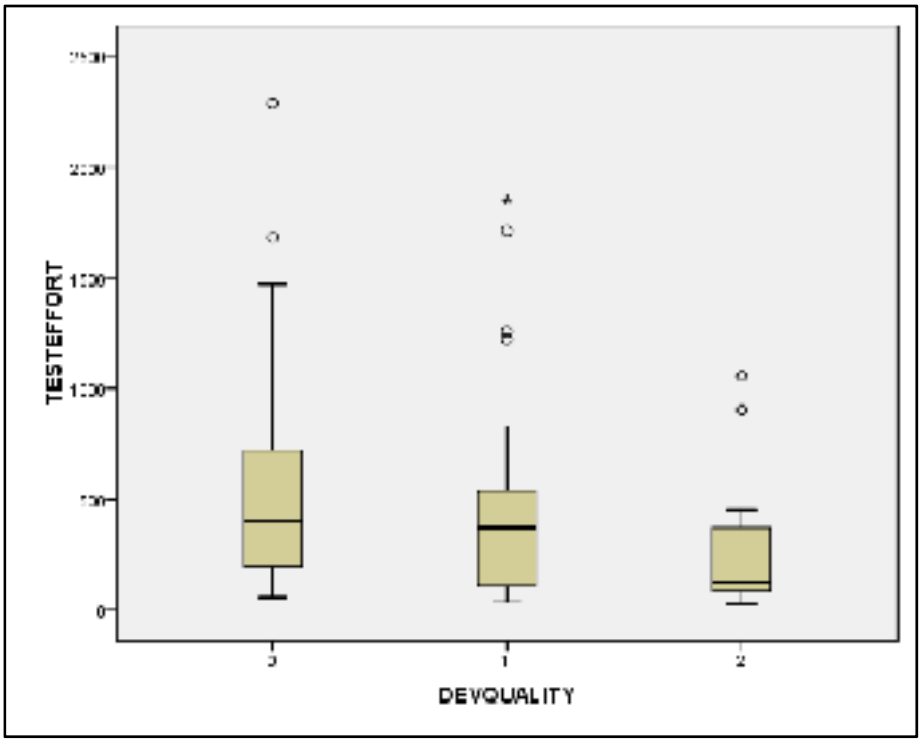


Figure A V - 3 Box Plots of DevQ Ratings 0, 1 and 2

Similarly, different median test effort and interquartile ranges for TestQ values of 0 and 1 (Figure A V - 4) reveal the effect of TestQ on test effort.

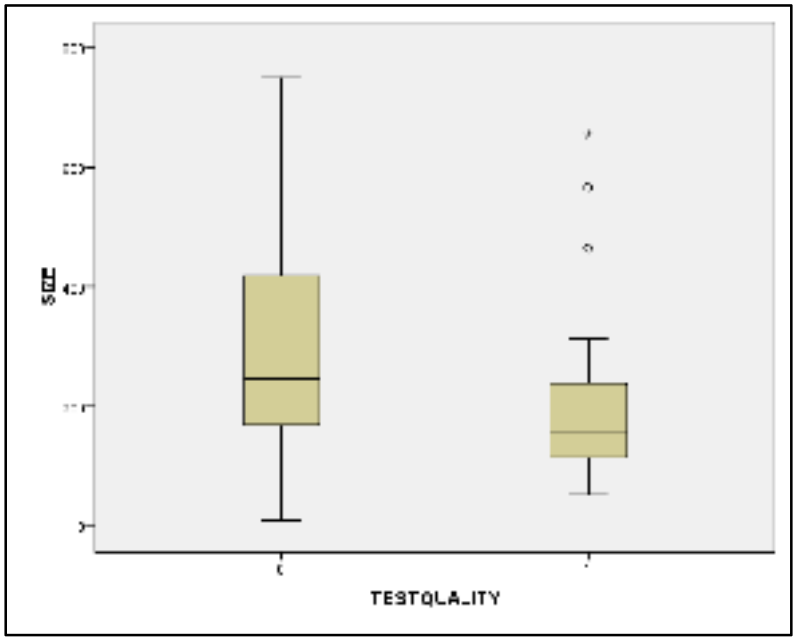


Figure A V - 4 Box Plots of TestQ Ratings 0 and 1

In order to further justify the inclusion of these variables, two statistical tests were carried out to evaluate their significance:

- the Kruskal-Wallis Test was taken up for DevQ as it involved three categories, and
- the Mann-Whitney Test was applied for TestQ (Table A V - 2).

Table A V - 2 Test of Significance for Independent variables

Statistical Test	**Variable**	**P Value**
Chi Square P Value	Size	< 0.001
Kruskal-Wallis Test	DevQ	0.018
Mann-Whitney	TestQ	0.001

The P-value indicates that Size, DevQ and TestQ are statistically significant.

Data Set C (COSMIC):

Scatter Diagrams of Size versus Test Effort for the whole Data Set C (COSMIC subset of Data Set A) and individual project groups PG1, PG2 and PG3 (Figure A V - 5) depict relationship between size and test effort.

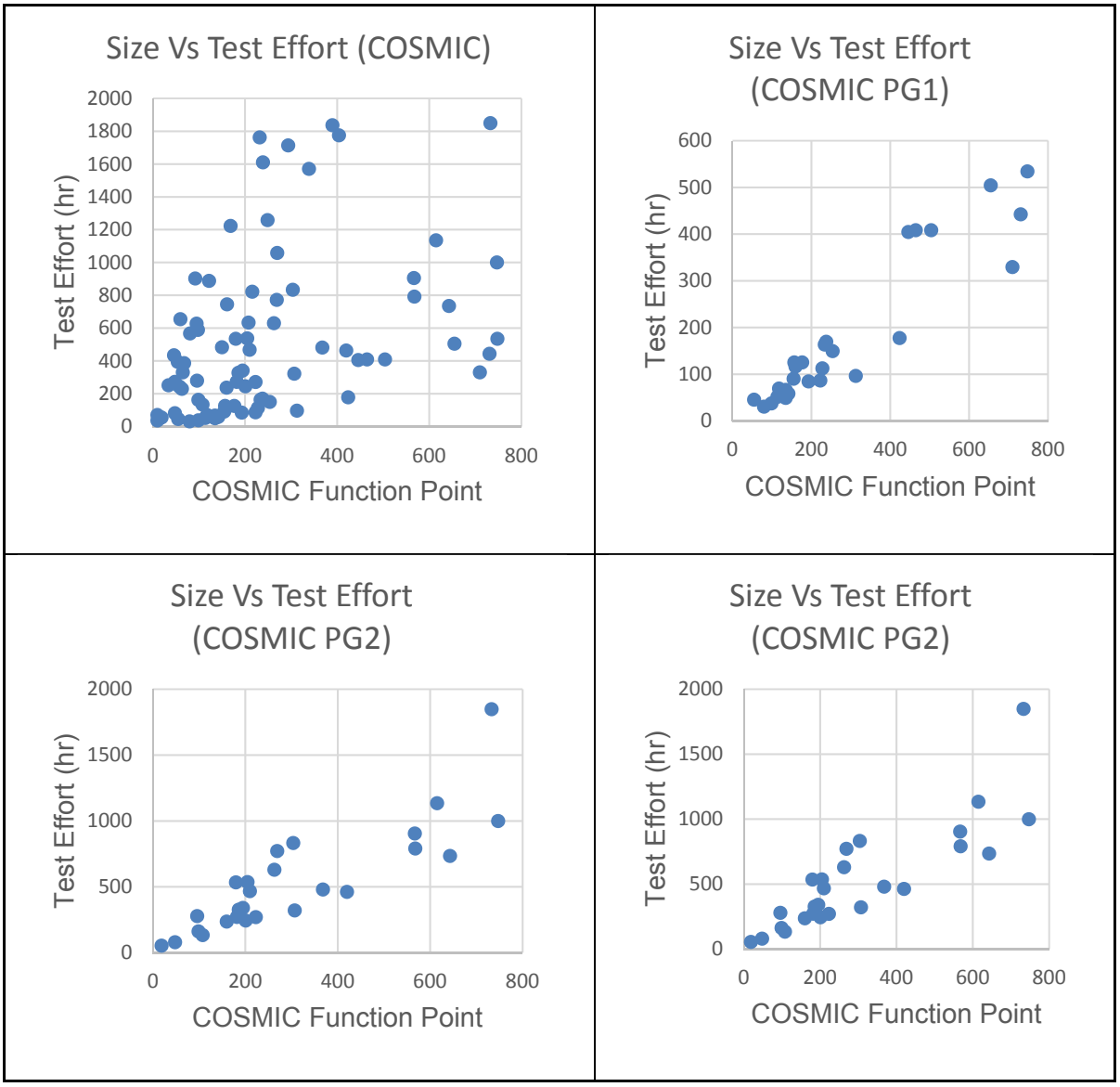


Figure A V - 5 Scatter Diagrams for Size vs Test Effort: COSMIC - PG1, PG2 and PG3

The correlation coefficient computed between Size and Test Effort in the data sets (Table A V - 3) also indicates a good correlation, and hence size is chosen as the primary independent variable.

Table A V - 3 Correlation: Size Vs Test Effort – COSMIC

Data Set	Correlation Coefficient
Data Set A	0.3980
Data Set A PG1	0.9307
Data Set A PG2	0.8520
Data Set A PG3	0.9077

Analysis of DevQ and TestQ for Data Set C

Classification of the data set in terms of DevQ and TestQ rating reveals that 83%, and 17% of the projects are of DevQ Ratings 1 and 2 respectively. There are no projects qualifying for DevQ rating 0 in the COSMIC data set. TestQ percentages are 71% and 29% for TestQ ratings 0 and 1 respectively (Figure A V - 6).

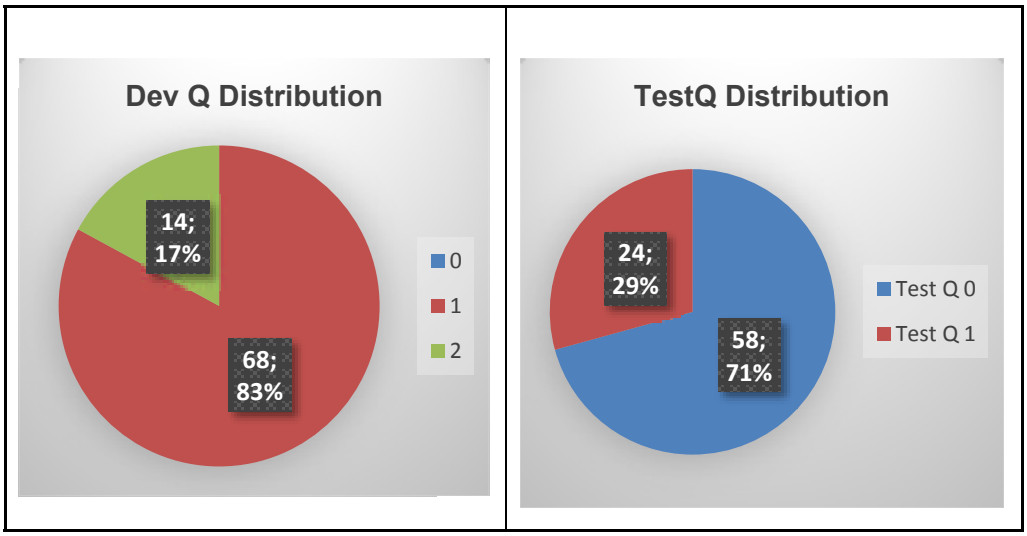


Figure A V - 6 Distribution of DevQ and TestQ Ratings (N = 82)

Different median test effort and interquartile ranges for DevQ values of 1 and 2 (Figure A V - 7) reveal the effect of DevQ on test effort. Median test effort and the interquartile range are also the lowest for the highest DevQ rating.

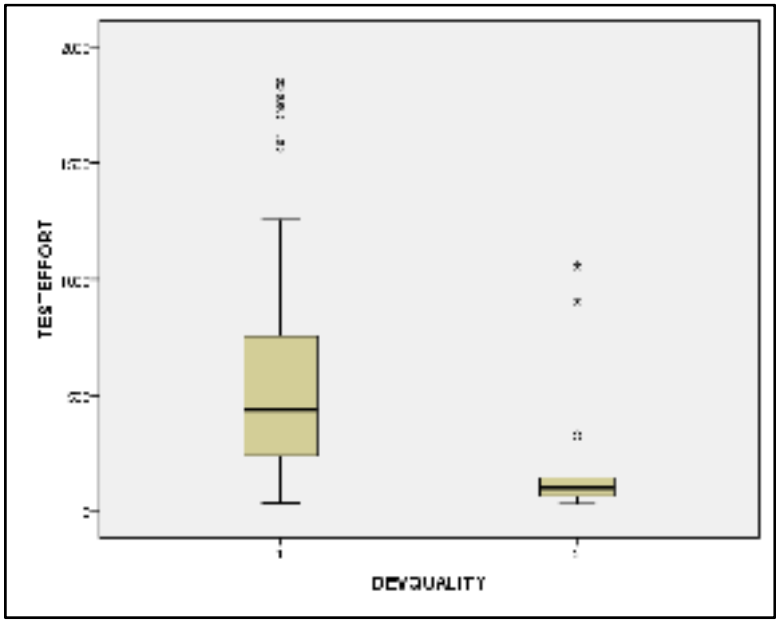


Figure A V - 7 Box Plots of DevQ Ratings 0,1 & 2

Similarly, different median test effort and interquartile ranges for TestQ values of 0 and 1 (Figure A V - 8) reveals the effect of TestQ on test effort. Median test effort and the interquartile range is also the lowest for the highest TestQ rating.

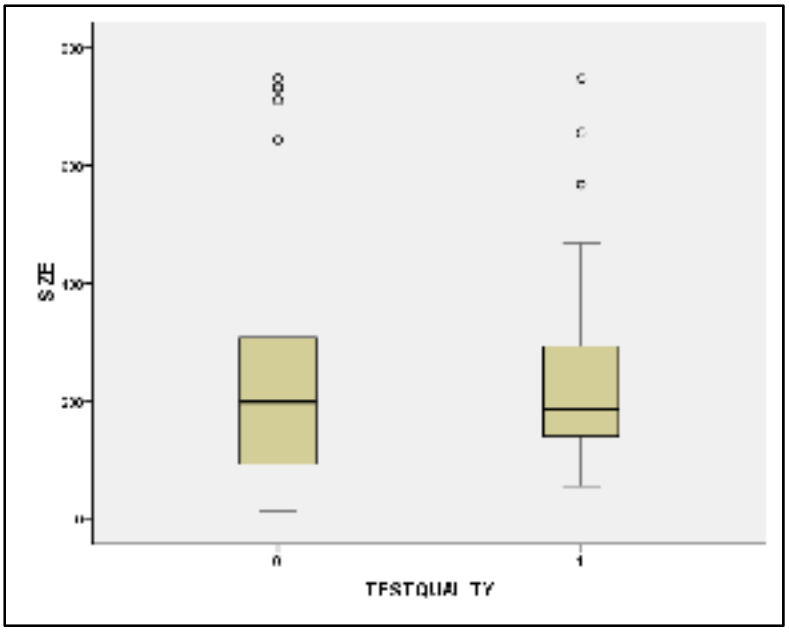


Figure A V - 8 Box Plots of TestQ Ratings 0 and 1

In order to further justify the inclusion of these variables, two statistical tests were carried out to evaluate their significance:

- the Kruskal-Walis Test was taken up for DevQ as it involved three categories, and
- the Mann-Whitney Test was applied for TestQ (Table A V - 4).

Table A V - 4 Test of Significance for Independent variables

Statistical Test	**Variable**	**P Value**
Chi Square P Value	Size	< 0.001
Kruskal-Wallis Test	DevQ	0.001
Mann-Whitney Test	TestQ	0.001

The P-value indicates that Size, DevQ and TestQ are statistically significant.

Data Set D (IFPUG):

The scatter diagrams of Size versus Test Effort for the whole IFPUG subset of Data Set A (referred as Data Set D) and individual project groups PG1, PG2 and PG3 (Figure A V - 9) depict the relationship between size and test effort.

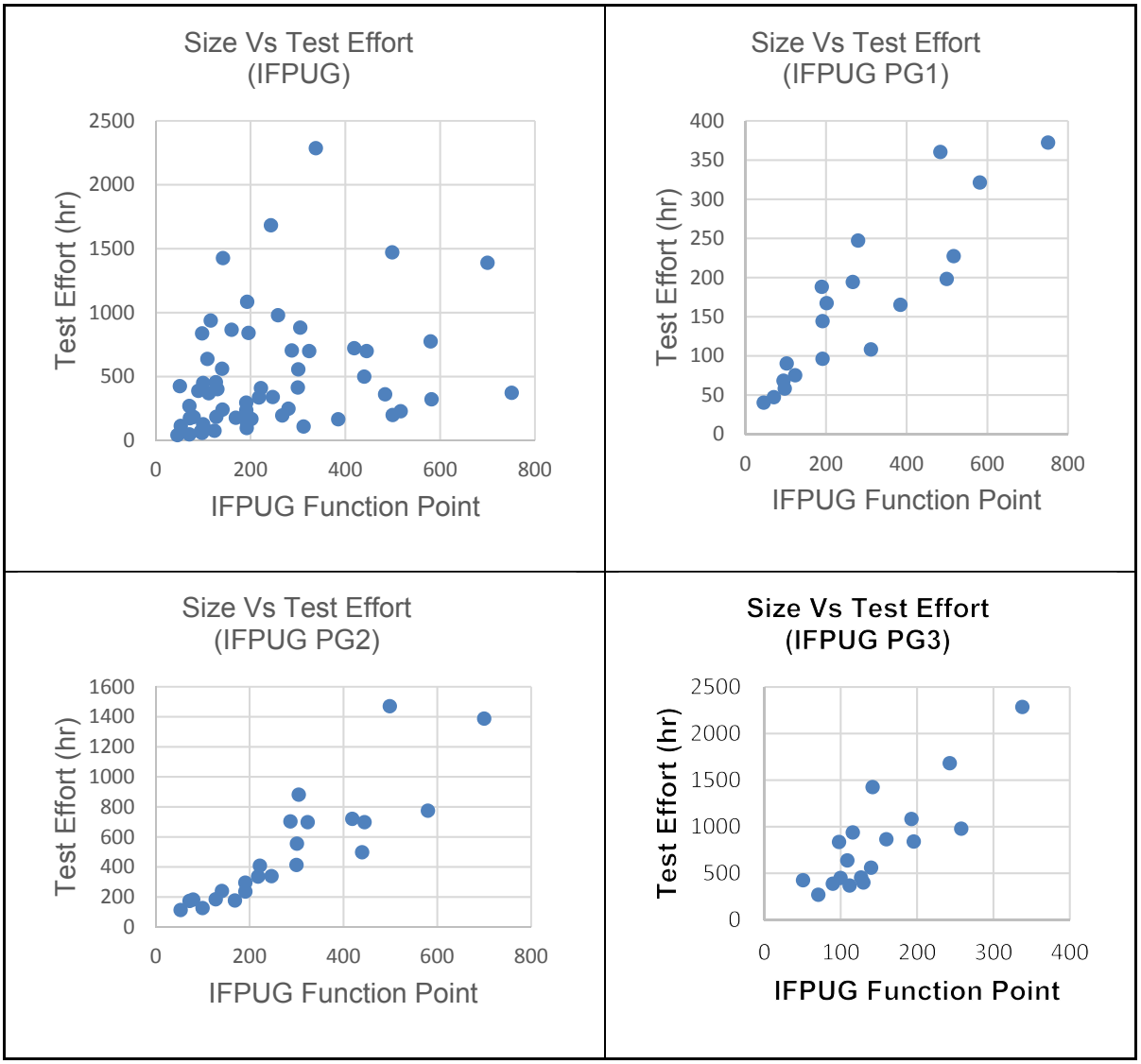


Figure A V - 9 Scatter Diagrams for Size vs Test Effort: IFPUG - PG1, PG2 and PG3

The correlation coefficient computed between Size and Test Effort in the data sets (Table A V - 5) also indicates a good correlation and, hence, size is chosen as the primary independent variable.

Table A V - 5 Correlation: Size Vs Test Effort – IFPUG

Data Set	Correlation Coefficient
Data Set A	0.2863
Data Set A PG1	0.8846
Data Set A PG2	0.8724
Data Set A PG3	0.8359

Analysis of DevQ and TestQ for Data Set D

The classification of the data set in terms of DevQ and TestQ rating reveals that 51%, 3%, and 12% of the projects are of DevQ Ratings 0, 1 and 2 respectively. TestQ percentages are 92% and 8% for TestQ ratings 0 and 1 respectively (Figure A V - 2).

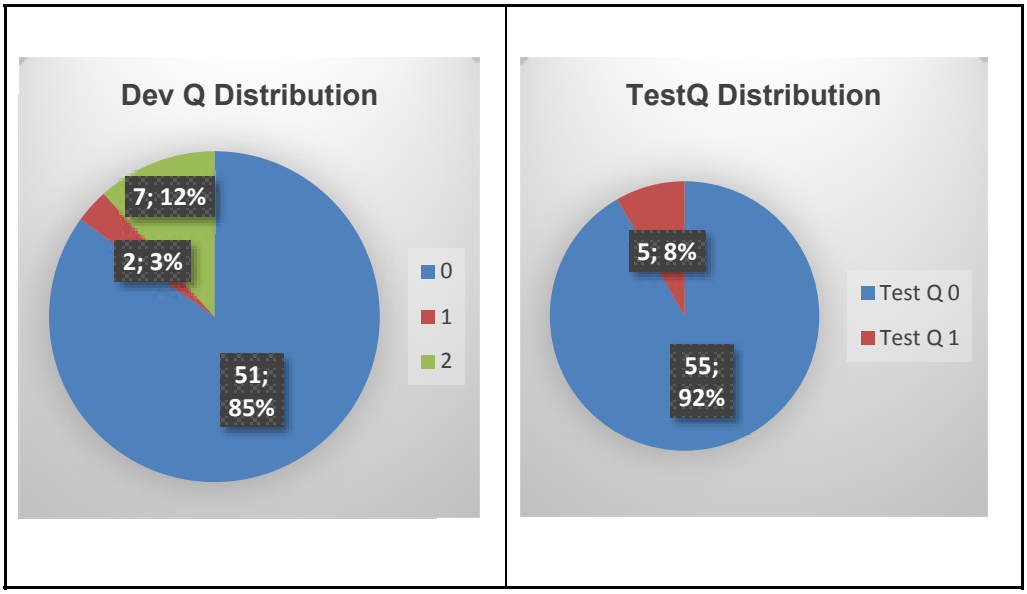


Figure A V - 10 Distribution of DevQ and TestQ Ratings (N = 60)

Different median test effort and interquartile ranges for DevQ values of 0, 1 and 2 (Figure A V - 10) reveal the effect of DevQ on test effort.

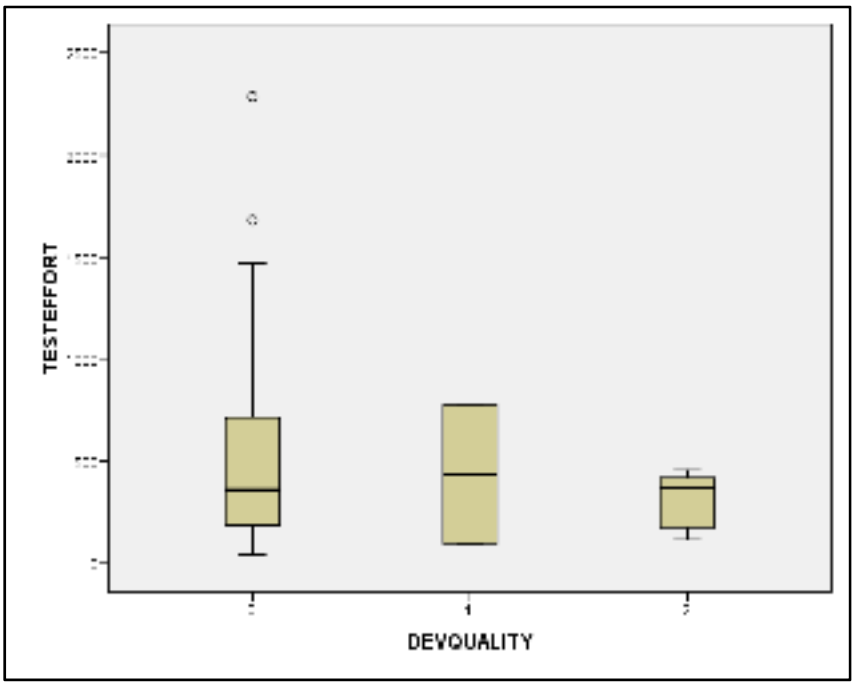


Figure A V - 11 Box Plots of DevQ Ratings 0, 1 and 2

Similarly, different median test effort and interquartile ranges for TestQ values of 0 and 1 (Figure A V - 6) reveal the effect of TestQ on test effort.

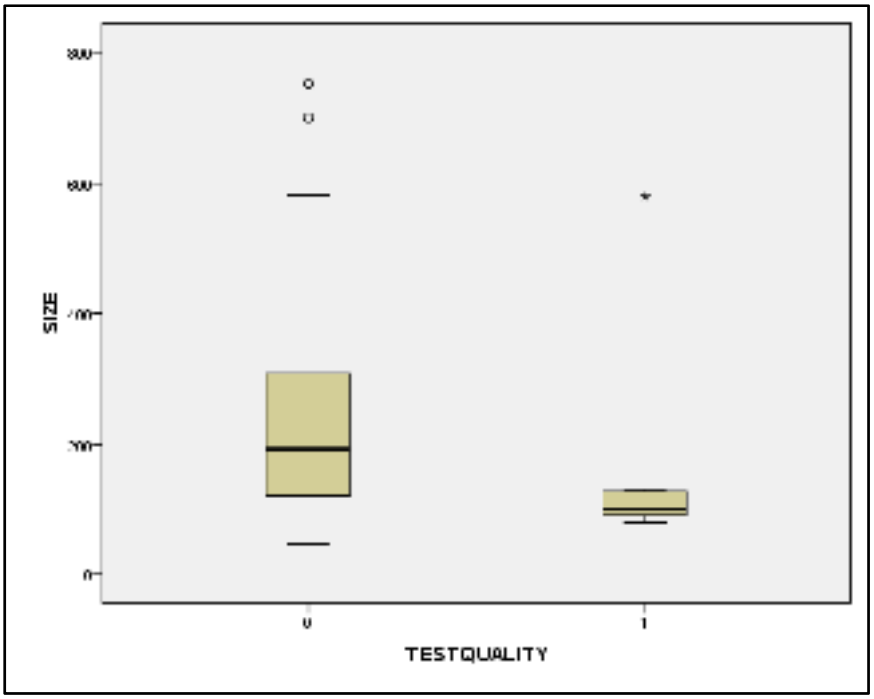


Figure A V - 12 Box Plots of TestQ Ratings 0 and 1

In order to further justify the inclusion of these variables two statistical tests were carried out to evaluate their significance:

- the Kruskal-Walis Test was taken up for DevQ as it involved three categories, and
- the Mann-Whitney Test was applied for TestQ (Table A V - 6).

Table A V - 6 Test of Significance for Independent variables

Statistical Test	**Variable**	**P Value**
Chi Square P Value	Size	< 0.001
Kruskal-Wallis Test	DevQ	0.723
Mann-Whitney Test	TestQ	0.565

The P-value indicates that DevQ and TestQ are not statistically significant.

Based on the analysis of the data sets in this Appendix it can be concluded that, to the exception of the IFPUG data set, Size, DevQ and TestQ can serve as independent variables for estimation models.

APPENDIX VI

REGRESSION MODEL FIT ANALYSIS

This appendix analyses regression models such as linear, exponential, power, logarithmic and polynomial models as to their fitment to the data set selected for constructing estimation models.

Data Set A has been taken up with its subsets PG1, PG2 and PG3 for this study: the objective is to provide estimation model specific to project contexts such as PG1, PG2 and PG3.

Linear model (Figure A VI – 1), Exponential & Power model (Figure A VI – 2) and Logarithmic and Polynomial model (Figure A VI – 3) are fitted to the PG1 data set.

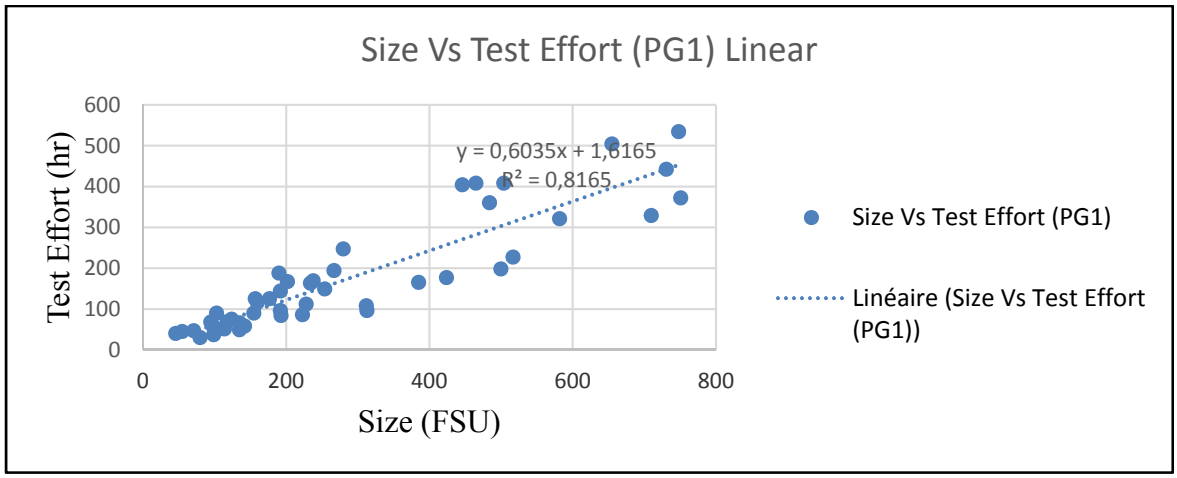


Figure A VI – 1 Linear Model for PG1

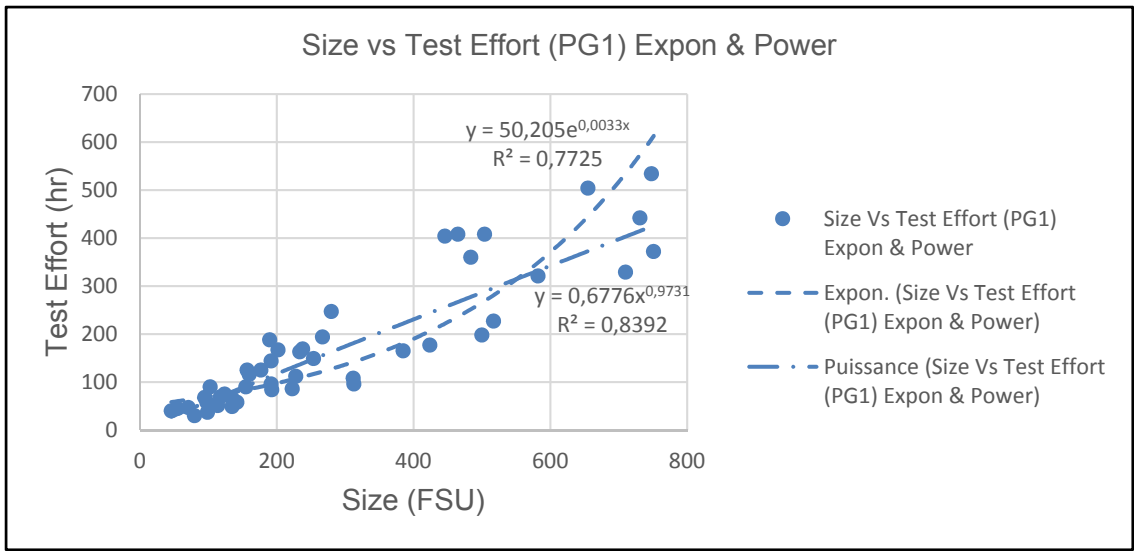


Figure A VI – 2 Exponential and Power Models for PG1

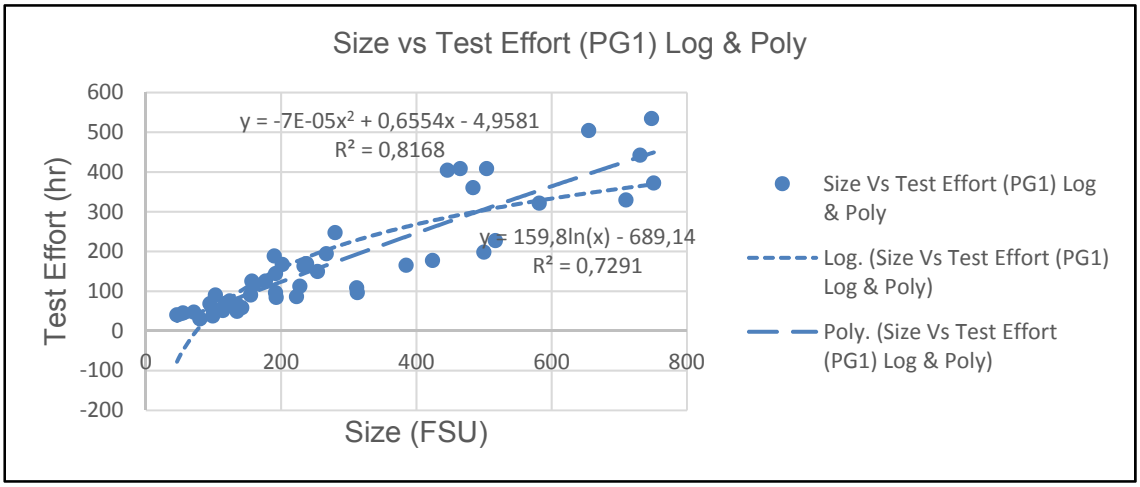


Figure A VI – 3 Logarithmic and Polynomial Models for PG1

Linear, Power and Polynomial models closely match with respect to R^2 values for PG1 while Exponential and Logarithmic models lag behind.

Based on PG2 data set, Linear model (Figure A VI – 4), Exponential & Power model (Figure A VI – 5) and Logarithmic and Polynomial model (Figure A VI – 6) are fitted.

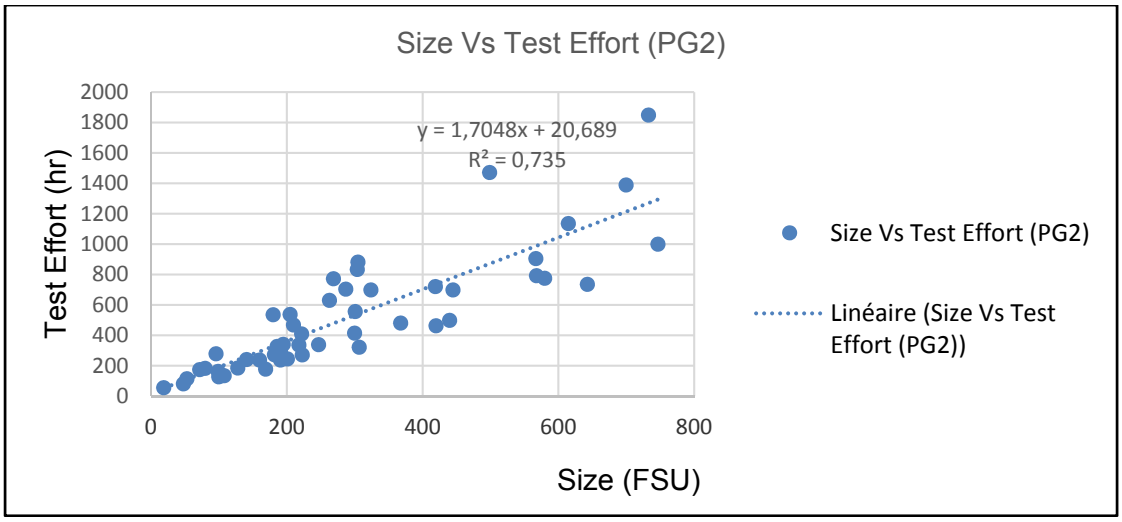


Figure A VI – 4 Linear Model for PG2

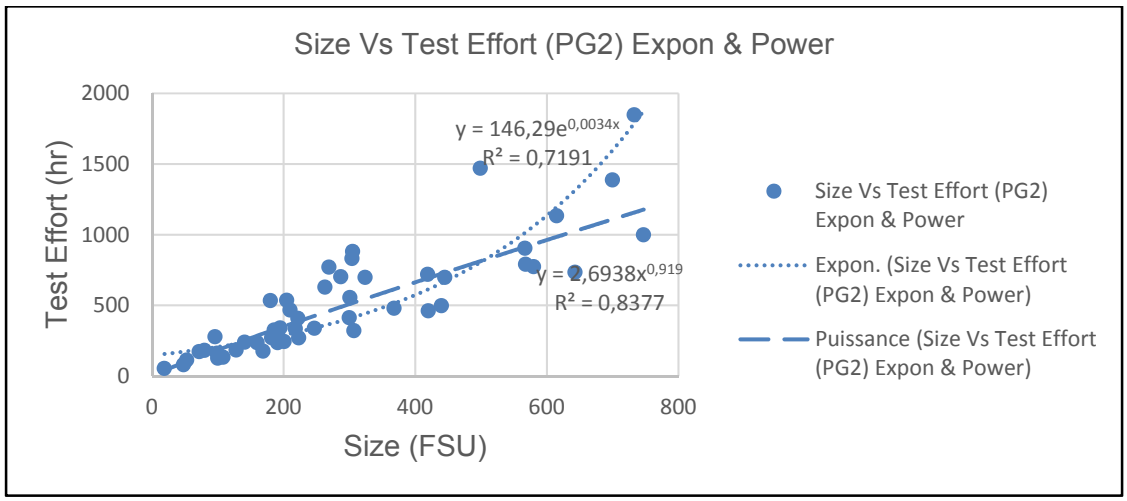


Figure A VI – 5 Exponential and Power Models for PG2

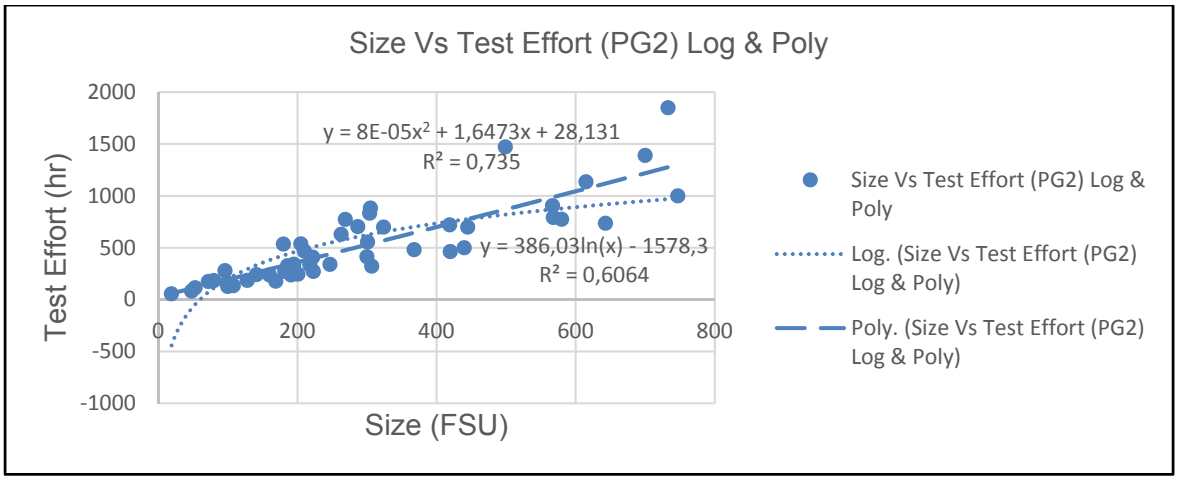


Figure A VI – 6 Logarithmic and Polynomial Models for PG2

A review of models fitted for PG2 reveals that Linear and Polynomial closely match with respect to R^2 Values while the power model results in better R^2 value, while the Exponential and Logarithmic models are poor.

Linear model (Figure A VI – 7), Exponential & Power model (Figure A VI – 8) and Logarithmic and Polynomial model (Figure A VI – 9) are explored for PG3.

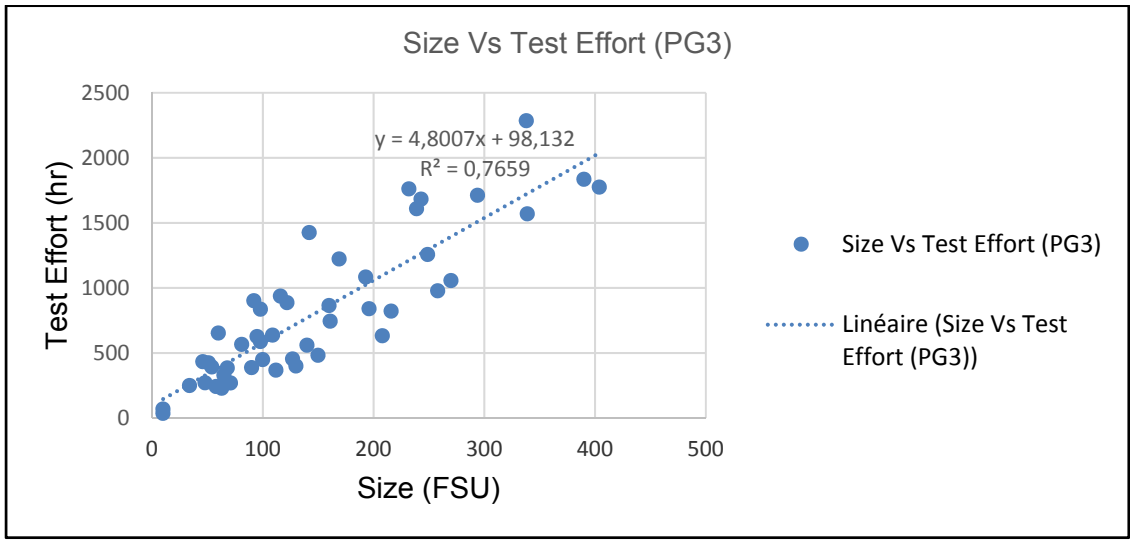


Figure A VI – 7 Linear Model for PG3

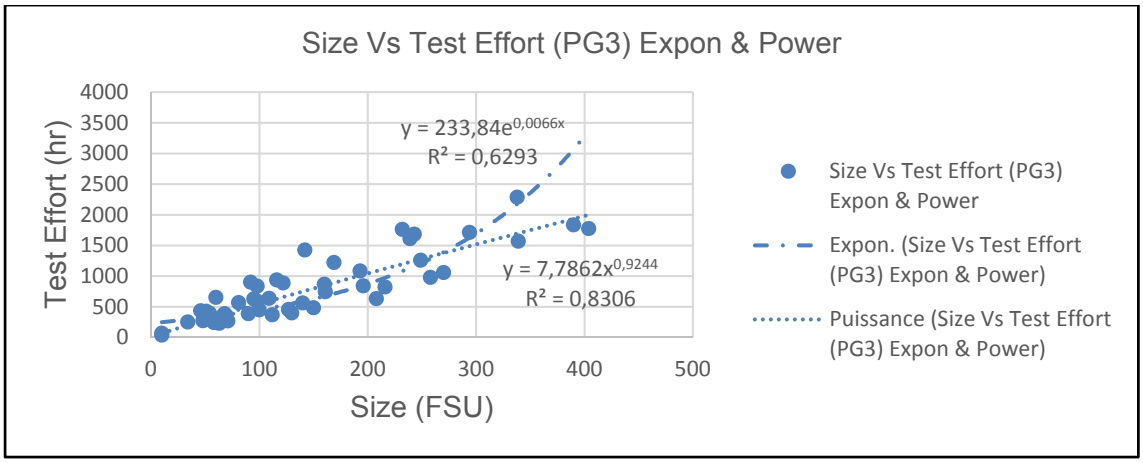


Figure A VI – 8 Exponential and Power Models for PG3

Behaviour of models fit for PG3 is similar to PG2 with Linear and Polynomial sharing the same R^2 value while Power model providing better value, and Exponential and Logarithmic models resulting in poorer values.

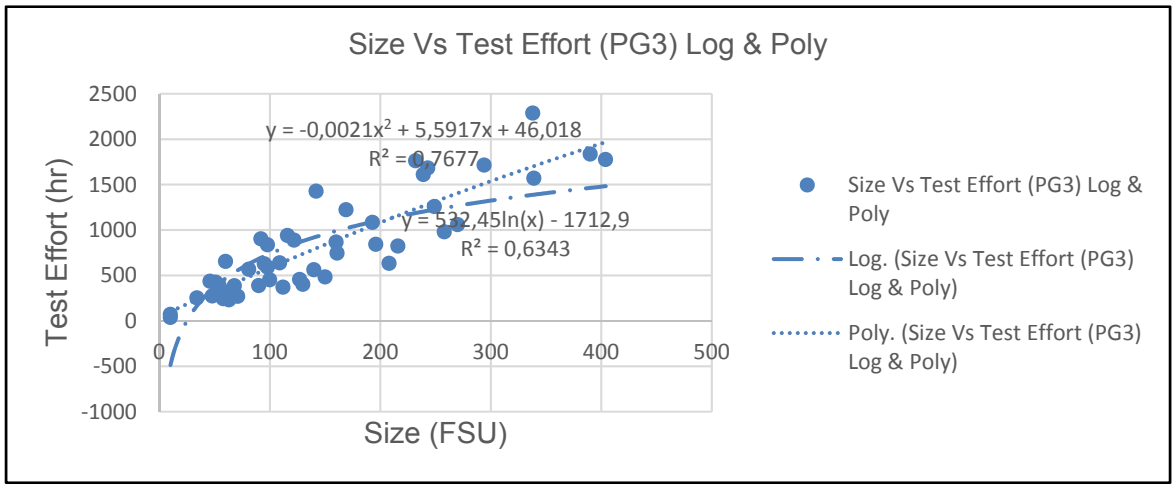


Figure A VI – 9 Logarithmic and Polynomial Models for PG3

Review of R^2 values of models across PGs (Table A VI – 1) indicate that Linear and polynomial models are similar, while power model appears better. Between Linear and Power models the difference is not significant for the PG1 data set.

Table A VI – 1 Values of R^2 for Models for each PG

Model	PG1	PG2	PG3
Linear	0.82	0.74	0.77
Exponential	0.77	0.72	0.63
Power	0.83	0.84	0.83
Logarithmic	0.73	0.61	0.63
Polynomial	0.82	0.74	0.77

Linear models have better interpretability as equal increments of the independent variable yield equal increments of the dependent variable. The same cannot be said with respect to other models. One of the major intention of this research work is to enable project managers to use estimation models without bothering too much about the mathematics behind it. The complex equations scare them and they do away with those coming with such equations. Some of them blindly follow and adjust them to suit project needs based on their judgements without really understanding the mathematics behind them and making major mistakes in that process.

APPENDIX VII

DESIGN OF A PROTOTYPE TOOL FOR ESTIMATION

The prototype estimation tool named as '***Chabroo***' is being developed to confirm that the estimation models for testing built during this research work can be automated. The first version of the tool will be a direct implementation of models and subsequent upgrades will involve more sophistications. This Appendix briefly describes the design approach used for the current prototype and its future versions.

The tool provides an option to use it to estimate or enter actual project data (Figure A VII – 1).

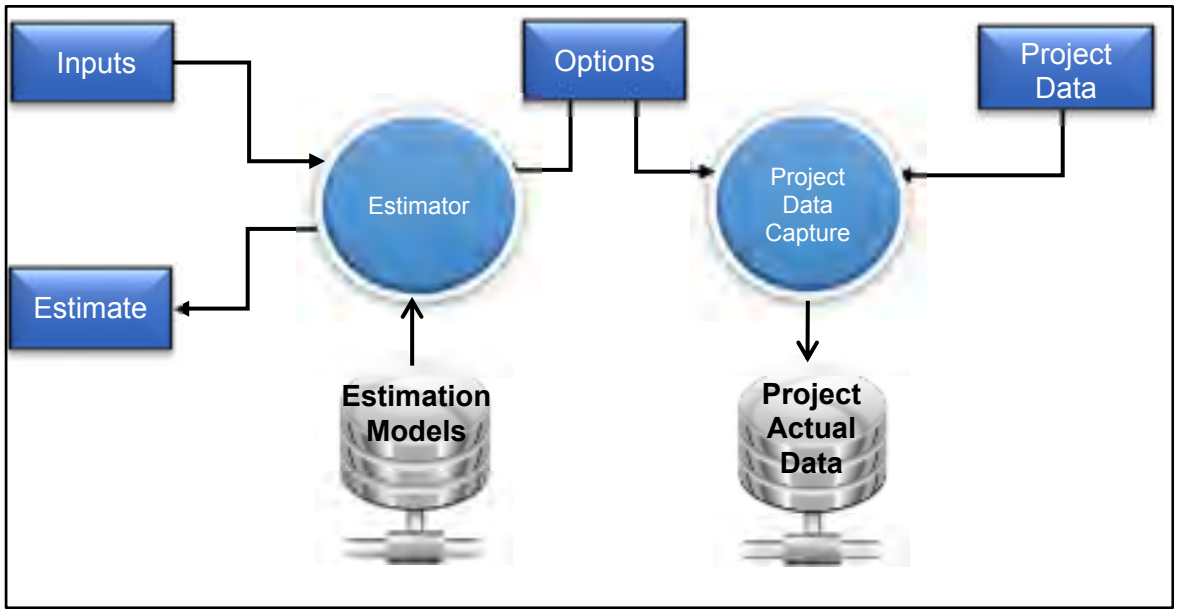


Figure A VII – 1 '*Chabroo*' Prototype

Inputs and outputs of estimate feature (Figure A VII – 2):

(1) Project Context: Project Identification, Project Name, Project Owner, Project Estimator, Date Start, Estimated End Date, Domain, Development Team Size, Development Duration, Rating of rigour of verification and validation activities during development

(2) Independent Variables: Measure of the functional size of the requirements, Development Process Rating (DevQ) and Test Process Rating (TestQ)

(3) Other inputs for Estimation: Prediction Interval

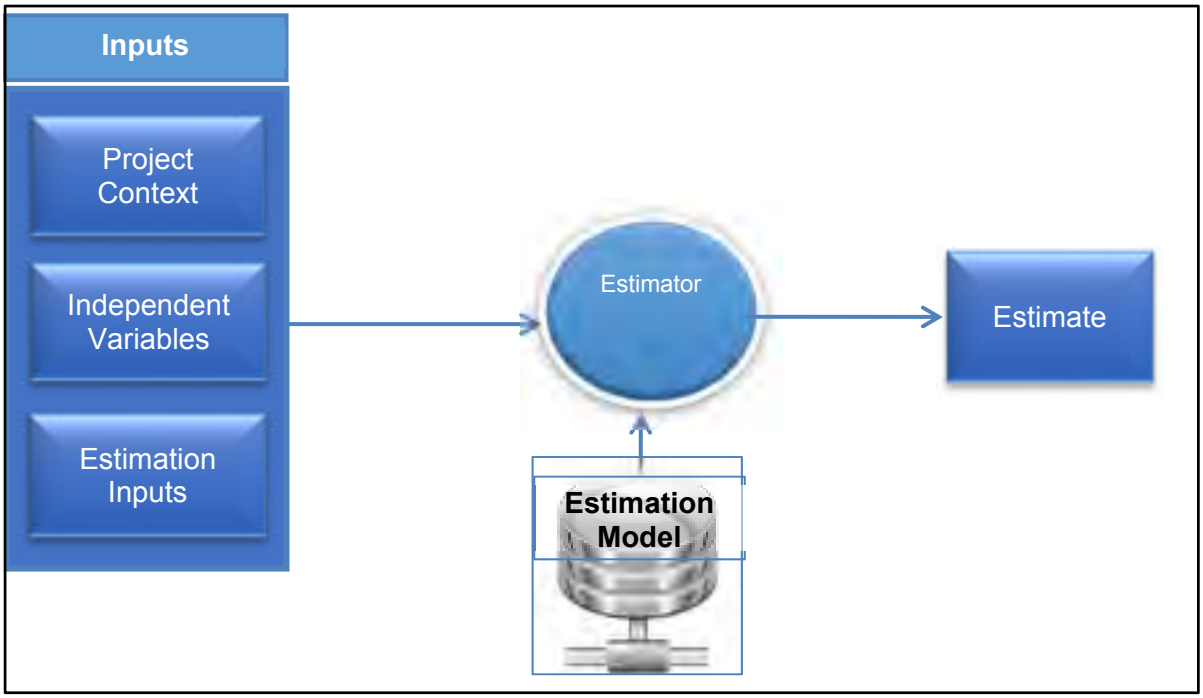


Figure A VII – 2 Inputs and outputs for Estimation

Original data from the projects based on which the models were generated will not be part of the data base as these data may be sensitive and/or not allowed to be shared. The models generated out of these data set will only be available as a part of the tool to be used within an organizational context.

Project data capture functionality (Figure A VII – 3) will facilitate for the tool user to enter actual data during the execution of the project. The data captured includes Project id, Details of testing phase, efforts expended, changes to any of the original inputs such as functional size, team size, duration, v & v rigour. These details will be stored in database organized in terms

of project domain, initial estimate, actual efforts, changes to project context inputs – functional size, development team size, development duration, v & v rigour and revised estimate.

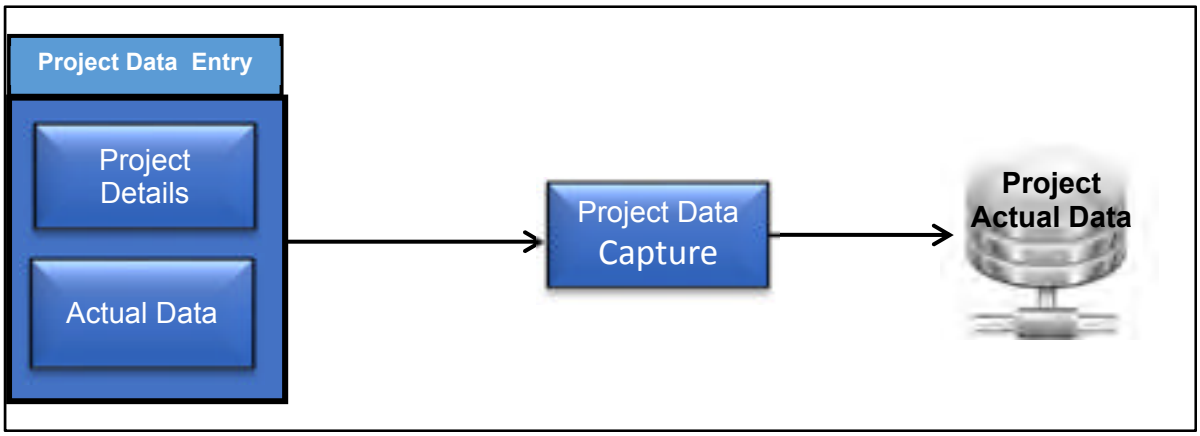


Figure A VII – 3 Project Actual Data Capture

Model Refinement (Figure A VII – 4) has been conceptualized to refine the existing models based on the actual data from several projects within an organizational context.

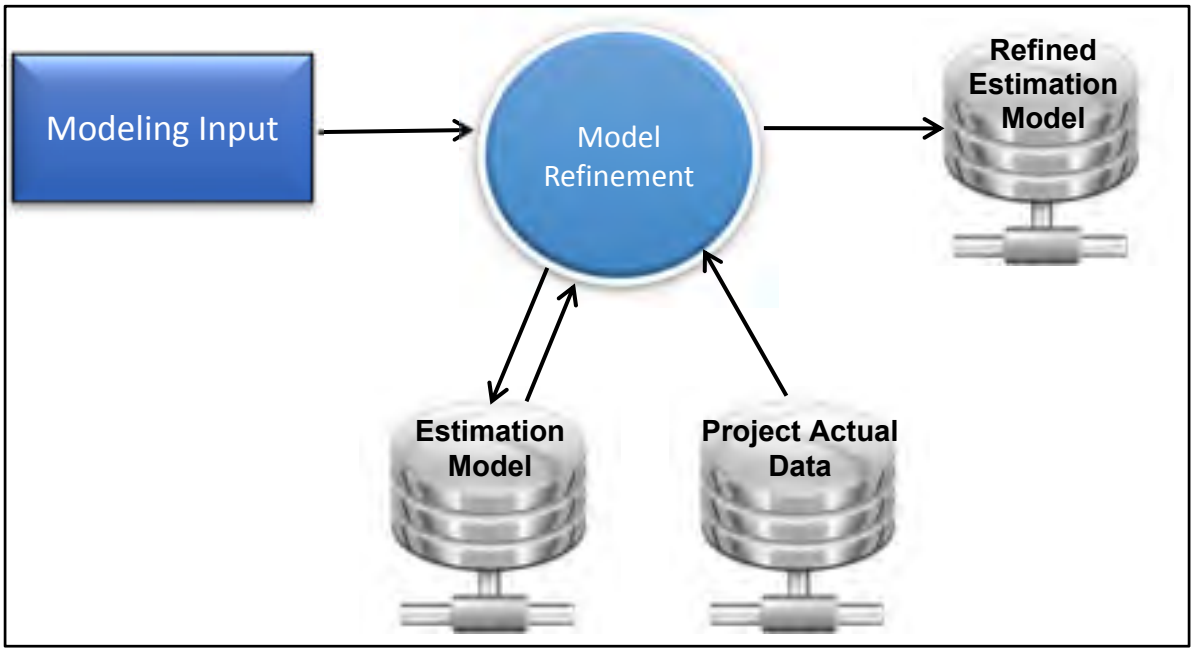


Figure A VII – 4: Model Refinement based on actual data from projects

There are two kinds of data available to refine the model. The model equations based on the original project data from where these models are generated and actual data generated during the execution of projects are used to regenerate models. Regeneration will involve certain user interventions in terms of modelling inputs to select and classify data from actual project data repository to suit the existing contexts. As there are no data points available as a part of the data repository related to the original models, simulation will be used to generate data points which will be merged with actual data from projects to generate new model equations. These new model equations are referred as 'Refined Estimation Model'. The refined estimation model will be available to estimator to use for prediction from this stage. As the data is a closer representative of the organizational context, estimates from this stage will provide better confidence to the user.

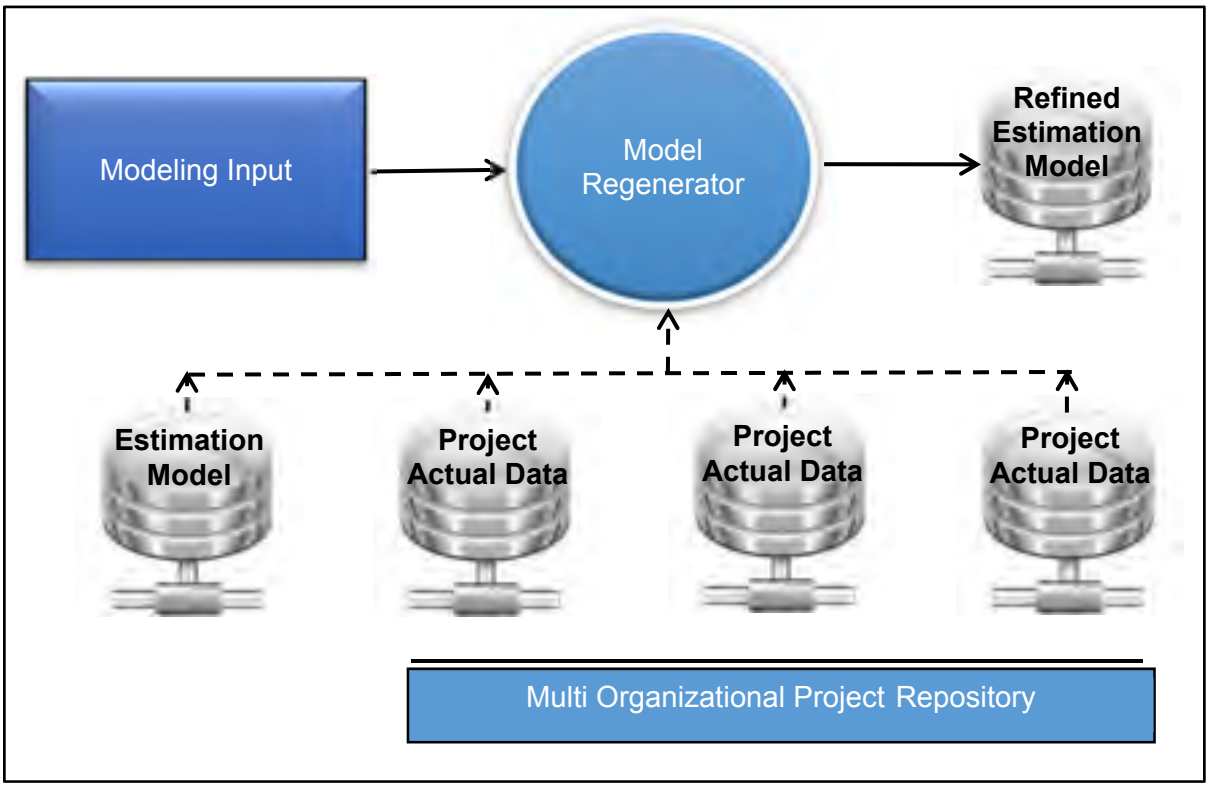


Figure A VII – 5 Model Regeneration based on multi-organizational data

The advanced implementation of the tool will involve (Figure A VII – 5) regeneration of models based on actual data from multiple organizations. When this tool is used by multiple

organizations and are willing to share the actual project data related to the organization, the tool can be used to regenerate a completely new set of models using the multi organizational data. Organizational data will be sufficiently sanitized and only the relevant parameters will be used for regeneration of models.

This advanced feature will enable international data repositories to be built and shared with confidence without the conventional hardship of data compilations for submission to international benchmarking agencies.

BIBLIOGRAPHY

Abu Talib, Olga Ormandjieva, Alain Abran, Luigi Buglione, Adel Khelifi, 2006, Scenario Based Blank Box Testing in COSMIC FFP, Software Quality Professional.

Abran, Alain, 2010, Software Metrics and Software Metrology, New Jersey: Wiley & IEEE Computer Society Press, pp. 328.

Abran, Alain, Juan Garbajosa, Laila Cheikhi, 2007, Estimating the Test Volume and Effort for Testing and Verification & Validation: International Workshop on Software Measurement - IWSM-Mensura Conference, Nov. 5-9, 2007, Publisher: UIB-Universitat de les Illes Baleares, Spain, 2007, p. 216-234.

Abran, Alain 2015, Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers, IEEE Computer Society, New Jersey: John Wiley & Sons.

Al-Sarayreh, Abran, Alain, Khalid T., Juan J. Cuadrado-Gallego, 2013, 'A Standards Based Framework for System Portability Requirements, Computer Standards & Interfaces,' Amsterdam: Elsevier Science Publishers, Volume 35, Issue 4, June 2013, p 380 – 395.

Al-Sarayreh, Khalid T., Alain Abran, Juan J. Cuadrado-Gallego, 2013, A Standards Based Reference Framework for System Operations Requirements, Journal Computer Standards & Interfaces, B V Amsterdam : Elsevier Science Publishers, Volume 47, Issue 4, July 2013, pp. 351–363.

Albrecht, A. J., 1979, Measuring Application Development Productivity, Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17, IBM Corporation (1979), pp. 83–92.

Amrinder Sing Grewal, Vishal Gupta, Rohit Kumar, 2013, Comparative Analysis of Neural Network Techniques for Estimation, International Journal of Computer Applications Volume 67-No. 11, April 2013.

Bala, Abdalla, 2013, Impact Analysis of a Multiple Imputation Technique for Handling Missing value in the ISBSG Repository of software project, Ph. D Thesis, Ecole de technologie superieure – University of Quebec, Montreal (Canada), October 17, 2013.

Bareja, K and Singal A, 2015, A Review of Estimation Techniques to Reduce Testing Efforts in Software Development, Fifth international Conference on Advanced Computing & Communication Technologies (ACCT), IEEE, 2015.

Beizer, Boris, 2007, Software Testing Techniques, 2nd Edition, Reprint Edition: 2007, New Delhi : Dreamtech Press, 550 p.

Bharadwaj, Mridul and Rana, Ajay, 2015, Estimation of Testing and Rework Efforts for software Development Projects, Asian Journal of Computer science and Information Technology, v.5, n.5, May 2015, pp 33 – 37.

Bharadwaj, Yogesh, Kaushik Manju, 2014, A Review paper on Effort Estimation and Model Based Regression Testing with SOA, International Journal of Computer Science and Mobile Computing, IJCSMC, Vol. 3, Issue. 4, April 2014, pg. 1116 - 1121.

Black, Rex, 2002, Test Estimation - Tools & techniques for realistic predictions of your test effort,; www.rbcs-us.com/documents/TestEstimation (article) .pdf, Access date 9 May 2012.

Boehm, B. W, 1981, Software Engineering Economics, New York, Prentice Hall.

Boehm, B. W, Abst, C. 2000, Software Cost Estimation with COCOMO II, New Jersey: Prentice Hall, 502 p.

Bossavit, Laurent, 2015, 'The Leprechauns of Software Engineering - How folklore turns into fact and what to do about it,' https://leanpub.com/leprechauns

Bourque, P., J. Moore, A. Abran, R. Dupuis, L. Tripp, 2014,Guide to the Software Engineering Body of Knowledge -- SWEBOK v3.0, 2014 Version, IEEE Computer Society, Los Alamitos, http://www.swebok.org

Capgemini, 2015, World Quality Report 2015 – 16, Seventh Edition, Published by Capgemini, 2015, www.worldqualityreport.com, 80p.

Cesar Andres, Mercedes G Merayo, Alberto Nunez, 2012 Using probabilistic customer models to estimate the cost of checking SLAs of real time systems, Departamento Sistemas Informaticos y Computacion Universidad Computense de Madrid E-28040 Madrid. Spain, Research Report, (TIN2009-14312-C02-01).

Chemuturi, Murali, 2012, Test Effort Estimation, http://chemuturi.com/Test %20Effort%20Estimation.pdf, Access date : 9 May 2012.

Chintala Abhishek, Veginati Pavan Kumar, Harish Vitta, Praveen Ranjan Srivastava, 2010, 'Test Effort Estimation Using Neural Network', Journal of Software Engineering & Application, 2010, 3, p 331-340.

Cohn, M., 2005, Agile Estimating and Planning, Prentice Hall, 2005.

Conte, SD, Dunsmore DE, Shen VY,1986, Software Engineering Metrics and Models, Menlo Park: The Benjamin/ Cummings Publishing Company, Inc.

COSMIC, 2015, The COSMIC Functional Size Measurement Method, Version 4.0.1, Measurement Manual, (The COSMIC Implementation Guide for ISO/IEC 19761:2011), Common Software Measurements International Consortium, Canada, April 2015, 98p., www.cosmic-sizing.org.

COSMIC, 2011, Guideline for usage of COSMIC to manage Agile projects, Common Software Measurements International Consortium, Canada, September 2011, 35p., www.cosmic-sizing.org.

COSMIC, 2015, Guideline on Non-Functional and Project Requirements v1.0, Common Software Measurements International Consortium, Canada, November 2015, 56p., www.cosmic-sizing.org.

Daniel Guerreiro E Silva, Bruno Teixeira De Abreu, Mario Jino, 2009, 'A Simple Approach for Estimation of Execution Effort of Functional Test Cases,' International Conference on Software Testing Verification and Validation, IEEE Computer Society: Denver, Colorado, USA, , 2009, Pages 289-298.

David F. Stephan, David M. Levine, 2013, Even You Can Learn Statistics: A Guide for Everyone, by David M. Levine, David F. Stephan, Pearson 2013.

Delany, Sarah Jane, Cunningham, Padraig, 2000, 'The Application of Case-Based Reasoning to Early Software Project Cost Estimation and Risk Assessment'. - Dublin, Trinity College Dublin, Department of Computer Science, TCD-CS-2000-10, 2000, p 20.

Dirk Basten, Ali Sunyaev, 2011, Guidelines for Software Development Effort Estimation, IEEE Computer, October 2011, p 88 - 90.

Dumke, Reiner and Abran, Alain (Edr), 2011, COSMIC Function Points, Theory and Advanced Practices, Chapter 3.5: Measurement Convertibility - From Function Points to COSMIC FFP, New York: CRC Press, p ,214- 225.

ECSS-E-40 Part 1B, 2003, European Consortium for Space Standardisation: Software – Part1 : Principles and Requirements, November 28, 2003.

Eduardo Aranha and Paulo Borba, 2007, Sizing system tests for estimating test execution effort, The 22nd International Annual Forum on COCOMO and Systems/ Software Cost Modelling, USC Campus: Los Angeles, CA, October 29 – November 2, 2007, 8p., http://csse.usc.edu/events/2007/CIIForum.

Galorath, Dan, 2015, Why Can't People Estimate: Estimation Bias and Mitigation, Conference presentation at IT Confidence October 2015, ISBSG.

Gartner, 2016, Forecast: Enterprise IT Spending by Vertical Industry Market, Worldwide 2013 – 2019, 4Q15 Update, 27 January 2016, www.gartner.com/doc/3191919.

Gencel Cigdem, Charles Symons, 2009, From performance measurement to project estimating using COSMIC functional sizing, Software Measurement European Forum 2009 (SMEF 2009), January 2009, 14p.

Gray Martha M., 1999, 'Applicability of Metrology to Information Technology', Journal of Research of the National Institute of Standards and Technology, Vol. 104, Number 6, November-December.

Hill. Peter, ISBSG, 2010, Practical Software Project Estimation: A Toolkit for Estimating Software Development Effort and Duration, McGraw Hill, 312 p.

IFPUG, 2005, Function Point Counting Practices Manual, Version 4.2.1, International Function Points Users Group, 500p.

ISBSG, 2013, Repository Data Release 12 – Field Descriptions, 'e.Field Descriptions – Data Release 12. Pdf' document provided as a part of data set, International Software Benchmarking and Standards Group, 14 p.

ISO/IEC 14143-1, 2007, Information technology – Software measurement – Functional size measurement Part 1: Definition of concepts, International Organization for Standards, Geneva.

ISO/IEC 19761, 2011, Software Engineering – COSMIC – A Functional Size Measurement Method, International Organization for Standardization, ISO, Geneva.

ISO/IEC 20926:2009, Software and systems engineering - software measurement - IFPUG Functional size measurement method, 2009, International Organization for Standardization, Geneva.

ISO/IEC 25010 – Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models, (2011), International Organization for Standardization, Geneva.

ISO/IEC/IEEE 29119 – Part 1(2013): Software and Systems Engineering – Software Testing – Concepts and Definitions, 2013, International Organization for Standardization, Geneva.

ISO/IEC/IEEE 29119 – Part 2 (2013): Software and Systems Engineering – Software Testing – Test Processes, 2013, International Organization for Standardization, Geneva.

ISO/IEC/IEEE 29119 – Part 3(2013): Software and Systems Engineering – Software Testing – Test Documentation, 2013, International Organization for Standardization, Geneva.

ISO/IEC/IEEE 29119 – Part 4 (2015): Software and Systems Engineering – Software Testing – Test Techniques, 2015, International Organization for Standardization, Geneva.

ISTQB, 2011, Certified Tester, Foundation Level Syllabus, Released Version 2011, International Software Testing Qualifications Board, www.istqb.org, 78p.

Izak Pierre Eramus, 2012, The COSMIC EPC Method: An ERP functional size measurement method delivering time and cost estimates, University of Gothenburg, Sweden, April 2012.

Jaswinder Kaur, Satwinder Sing, Dr. Karanjeet Singh Kahlon, Pourush Bassi, 2010, 'Neural Network – A Novel Technique for Software Estimation', International Journal of Computer Theory and Engineering, Vol. 2, No.1, February 2010, p17-19.

Jones, Capers, 2007, Estimating Software Costs- Bringing Realism to Estimating, Second Edition, New Delhi: Tata McGraw-Hill. 653 p.

Kafle, Lava, 2014, An Empirical study on software test effort estimation, International Journal of Soft Computing and Artificial Intelligence, ISSN: 2321-404X, Volume-2, Issue-2, Nov-2014, pp 96-106.

Kamala Ramasubramani J., 2006, CAFÉ – A framework for Effective Automation of Software Testing, M.Phil. Thesis, 2006, Madurai Kamaraj University, Madurai, India.

Kamala Ramasubramani J., Alain Abran, 2013, Analysis of ISBSG Data for understanding Software Testing Efforts, 1st International Conference on IT data collection, Analysis and Benchmarking, Rio (Brazil) – Oct 3, 2013.

Kamala Ramasubramani J., 2011, 'Why you must change to COSMIC for Sizing and Estimation', International Conference on Software Engineering CONSEG - 2011, 17-19 February, 2011, Bangalore, India, p 86 - 92.

Kaner Cem, 2013, Practical Approaches to software metrics, Power Point Presentation, March 2013, http://kaner.com/pdfs/PracticalApproachToSoftwareMetrics.pdf

Kate Armel, 2012, Top Performing Projects Use Small Teams Deliver Lower Cost, Higher Quality, Blog Posting, Quantitative Software Management, Inc, USA, accessed on 18 January 2016.

Kemerer, Chris F. 1987, An Empirical Validation of Software Cost Estimation Models, Communications of the ACM, May 1987, Volume 30, Number 5, p 416-429.

Kerstner, Matthias, 2011, Software Test Effort Estimation Methods, Graz University of Technology, Austria; www.kerstner.at/en/2011/02/software-test-effort-estimation-methods/; Accessed on:18 May 2012.

Krishnamurthy Srinivasan, 1995, Machine Learning Approaches to Estimating Software Development Effort, IEEE Transactions on Software Engineering, Volume 21, Issue 2, February 1995, pp. 126-137.

Lindsey Charles, Simon Sheather, 2010, 'Variable selection in linear regression,' The Stata Journal, 10, Number 4, pp 650-669.

Lionel C Briand, Khaled El Emam, Dagmar Surmann, Isabella Wieczorek, Katrina D Maxwell, 1999, 'An Assessment and Comparison of Common Software Estimation Modeling Techniques', 21st International Conference on Software Engineering' pp 313-322, 1999.

Mussa, Mohamed, Samir Ouchani, Waseem Al Sammane, and Abdelwahab Hamou-Lhadj, 2009, A survey of model-driven testing techniques, 9th International Conference on Quality Software - QSIC'09., pp. 167-172. IEEE, 2009. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5381477

Mutalik, Prakash, 2003, Test Metrics and Estimation Model, Power point presentation, National Summit on Software Testing (NSST), Chennai, India.

Ossi Taipale, 2007, Observations on Software Testing Practice, PhD Thesis, Lappeenranta University of Technology, Finland, 2007.

Perry, William, 2006, Effective Methods for Software Testing, 3rd Edition, Singapore: John Wiley & Sons (ASIA) Pte Ltd, 812 p.

PMI, 2013, A Guide to the Project Management Body of Knowledge (PMBOK Guide), Fifth Edition, PMI Standards Committee, Project Management Institute, USA, 2013.

Putman, L., 1978, A general empirical solution to the macro software sizing and estimating problem. IEEE Transactions on Software Engineering, Volume 4, No 4, pp 345-61, April 1978.

Putnam Doug 2005, Team Size can be the Key to a successful software project, Quantitative Software Management Inc., USA. www.qsm.com.

QAI 2006, Guide to the CSTE Common Body of Knowledge, Version 6.2, Software Testing Body of Knowledge, QAI Global Institute, USA, 561p.

Rubin, H.A, 1982, Macro estimation of software development parameters: The Estimacs system. In SOFTFAIR Conference on Software Development Tools, Techniques and Alternatives (Arlington, Va., July 25-28), IEEE Press, New York, 1982, pp. 109-118.

Siegl, Sebastian; Hielscher, Kai-Steffen; German, Reinhard, 2010, "Model driven testing of embedded automotive systems with timed usage models," Vehicular Electronics and Safety (ICVES), 2010 IEEE pp.110-115, 15-17 July 2010, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5550938&isnumber=5550921.

Solomon Mensah, Jacky Keung, Kwabena Ebo Bennin and Michel Franklin Bosu, 2016, Multi-Objective Optimization for Software Testing Effort Estimation, International Conference on Software Engineering & Knowledge Engineering, SEKE 2016, June 30 – July 3, 2016, San Francisco, California, USA, http://ksiresearchorg.ipage.com/seke/seke16paper/seke16paper_163.pdf.

southernSCOPE, 2012, Avoiding Software Budget Blow outs, eGovernment Resource Center, State of Victoria, Australia, http://www.egov.vic.gov.au/ victorian-government-resources/e-government-strategiesvictoria/ southern scope /southernscope-avoiding-software-budget-blowouts.html.

Srivastava, Praveen Ranjan, 2009, Estimation of Software Testing Effort: An Intelligent Approach, 20th International Symposium on Software Reliability Engineering (ISSRE), Mysore, India.

Symons, Charles, 2010, Webminar presentation on 'Effective use of Software Metrics', Software Measurement Services, http://www.smsexemplar.com.

Symons, Charles, Alain Abran, Jean-Marc Desharnais, Serge Oligny, Denis St-Pierre, 2009, The COSMIC Functional Size Measurement Method Version 3.0.1, Measurement Manual, (The COSMIC Implementation Guide for ISO/IEC 19761:2003), Common Software Measurements International Consortium.

Taleb, Nassim Nicholas, 2012, Anti Fragile, London: Penguin Books Limited, 519 p.

Tassey, Gregory, 2002, 'The Economic Impacts of Inadequate Infrastructure for Software Testing', National Institute of Standards and Technology, USA, 309 p.

Valdes, Francisco, 2011, Design of Fuzzy Logic Estimation Process, Ph. D., Thesis, Ecole de technologie supérieure - ETS, University of Quebec, Canada.

VIM ISO/IEC Guide 99, 2007, International vocabulary of metrology – Basic and general concepts and associated terms (VIM), International Organization for Standardization – ISO, Geneva.

Wisconsin, 2007; 2015, Do IT's Project Management Advisor (PMA), PM Best Practice Guide, Board of Regents, University of Wisconsin, www.pma.doit.wisc.edu, 2007, site last updated in November 3, 2015, accessed on 27 December 2015.