Improve Software Defect Estimation with Six Sigma Defect Measures: Empirical Studies with Imputation Techniques on ISBSG Data Repository with a High Ratio of Missing Data

by

Mhammed ALMAKADMEH

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

MONTREAL, JUNE 23, 2017

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

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

Mr. Antoine Tahan, President of the Board of Examiners
Department of Mechanical Engineering, École de technologie supérieure

Mr. Alain April, Member of the jury
Department of Software Engineering & Information Technology, École de Technologie Supérieure

Mr. Hamid Mcheick, External Evaluator
Department of Computer Science and Mathematics, Université du Québec à Chicoutimi

THIS THESIS WAS PRENSENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

JUNE 19, 2017

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEUR

ACKNOWLEDGMENT

I would like to express my gratitude to all who have supported me in my doctoral studies. First and foremost: my research director Professor Alain Abran, for his incredible support at all stages of my doctoral studies. His kindness, encouragement, wise advices and knowledge have made working with him a great pleasure and honour. His professionalism and positive attitude towards scientific research have encouraged me to work day and night to finish my doctoral studies. He nicely devoted much of his time in face to face meetings and fast email feedbacks to help me pursue my research project. I feel that he is always with me to guide me to the correct path, like a torch that guides its holder through his/her way. I believe that a doctoral student like me would not find such an extra-ordinary research director other than professor Abran. Also, I would like to thank everyone in the software engineering research laboratory and the department of software engineering and information technologies at École de technologie supérieure for their support.

I am deeply grateful to the jury members; Professor Mr. Antoine Tahan, Professor Mr. Alain April, and Professor Mr. Hamid Mcheick for taking time to review my doctoral thesis.

A great love to my parents, my father (Abdelkarim) the source of strength and wisdom in my whole life, my mother (Hanan), who I want to spend my entire life to thank and honor. You have shown me the joy of intellectual pursuit ever since I was a child. Thanks to my brothers (Khaled, Ahmad, Ali, and Alaa) and my sister (Shaima). And, I will not forget to thank my grandmother (Sua'ad) who has always supported and encouraged me throughout my whole life as well as, my deep appreciation to all my friends in Jordan and Canada.

IMPROVE SOFTWARE DEFECT ESTIMATION WITH SIX SIGMA DEFECT MEASURES: EMPIRICAL STUDIES WITH IMPUTATION TECHNIQUES ON ISBSG DATA REPOSITORY WITH A HIGH RATIO OF MISSING DATA

Mhammed ALMAKADMEH

RÉSUMÉ

Cette analyse de recherche fait état d'un ensemble d'études empiriques abordant les problèmes de recherche visant à améliorer les modèles d'estimation des défauts logiciels avec des mesures de défaut Sigma (par exemple, les niveaux Sigma 3 et 4) en utilisant le référentiel de données ISBSG avec un ratio élevé de données manquantes. Trois techniques d'imputation ont été sélectionnées pour ce travail de recherche: imputation unique, imputation de régression et imputation de régression stochastique. Ces techniques d'imputation ont été utilisées pour imputer les données manquantes dans la variable 'Nombre total de défauts', et ont d'abord été comparées les unes aux autres en utilisant des critères de vérification communs. Une autre stratégie de vérification a été développée pour comparer et évaluer la performance des techniques d'imputation sélectionnées en vérifiant la précision prédictive des modèles d'estimation des défauts logiciels obtenus dans les jeux de données imputés. Une classification basée sur Sigma a été effectuée sur l'ensemble de données imputé de la meilleure technique d'imputation de performance sur l'estimation des défauts logiciels. Cette classification a été utilisée pour déterminer à quels niveaux de Sigma; Les projets logiciels peuvent être utilisés pour construire des modèles d'estimation des défauts logiciels: ce qui a entraîné des ensembles de données Sigma avec la gamme Sigma (par exemple, l'ensemble de données de projets logiciels d'une gamme allant de 3 Sigma à 4 Sigma). Enfin, les modèles d'estimation des défauts logiciels ont été construits sur les ensembles de données Sigma.

Mots-clés: Six Sigma, ISBSG, Estimation des défauts, DMAIC, Design for Six Sigma (DFSS), Imputation unique, Imputation de régression, Imputation de régression stochastique.

IMPROVE SOFTWARE DEFECT ESTIMATION WITH SIX SIGMA DEFECT MEASURES: EMPIRICAL STUDIES WITH IMPUTATION TECHNIQUES ON ISBSG DATA REPOSITORY WITH A HIGH RATIO OF MISSING DATA

Mhammed ALMAKADMEH

ABSTRACT

This research analysis work reports on a set of empirical studies tackling the research issues of improving software defect estimation models with Sigma defect measures (e.g., Sigma levels) using the ISBSG data repository with a high ratio of missing data. Three imputation techniques that were selected for this research work: single imputation, regression imputation, and stochastic regression imputation. These imputation techniques were used to impute the missing data within the variable 'Total Number of Defects', and were first compared with each other using common verification criteria. A further verification strategy was developed to compare and assess the performance of the selected imputation techniques through verifying the predictive accuracy of the obtained software defect estimation models form the imputed datasets. A Sigma-based classification was carried out on the imputed dataset of the better performance imputation technique on software defect estimation. This classification was used to determine at which levels of Sigma; the software projects can be best used to build software defect estimation models: which has resulted in Sigma-based datasets with Sigma ranging (e.g., dataset of software projects with a range from 3 Sigma to 4 Sigma). Finally, software defect estimation models were built on the Sigma-based datasets.

Keywords: Six Sigma, ISBSG, Defect estimation, DMAIC, Design for Six Sigma (DFSS), Single imputation, Regression imputation, Stochastic regression imputation.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Page

Page

Page

Page

LIST OF FIGURES

Page

Page

Page

Page

LIST OF ABREVIATIONS

ANOVA	Analysis of Variance
BBs	Black Belts
CAR	Causal Analysis and Resolution
CMMI	Capability Maturity Model Integration
COPQ	Cost of Poor Quality
Cp	Process Capability
Cpk	Process Capability Index
DFSS	Design of Six Sigma
DOE	Design of Experiments
DMAIC	Define-Measure-Analyze-Improve-Control
DMADV	Define-Measure-Analyze-Design-Verify
DPMO	Defect Per Million Opportunities
DPU	Defect per Unit
DPUO	Defect per Unit opportunities
FMEA	Failure Modes Effect Analysis
GBs	Green Belts
ISBSG	International Software Benchmarking Standards Group
LL	Lower limit
MBBs	Master Black Belts
MSE	Measurement System Evaluation
OA	Orthogonal Arrays

OPP	Organizational Process Performance
OPM	Organizational Performance Management
PA	Process Area
PCDA	Plan-Do-Check-Action
PROMISE	The PRedictOr Models In Software Engineering
QC tools	Quality Control tools
QPM	Quantitative Project Management
QFD	Quality Function Deployment
RI	Regression Imputation
SRI	Stochastic Regression Imputation
SN	Signal-to-Noise
SI	Single Imputation
UL	Upper limit
6SSP	Six Sigma Software Program

LIST OF SYMBOLS

N	Number of projects
%	Percentage
P	Probability of reject H_0 when H_0 is true, P(reject H_0 I is true) (p-value)
δ	Sigma
*	Multiply
+	Addition
=	Equal

INTRODUCTION

Six Sigma has achieved recognizable success over the past 20 years in industry in general, while only a few studies have been conducted within the software industry to explore its use and expected benefits. In particular, there is a lack of related empirical studies based on large repository of software engineering project data such as the repositories of the International Software Benchmarking Standards Group (ISBSG) and PRedictOr Models In Software Engineering (PROMISE). This research thesis describes the planning and execution of empirical studies to explore Sigma defect measures can be useful for designing defect estimation models.

The large data repositories in software engineering have a common serious data collection challenge: when the data fields are not all mandatory, this often leads to a large percentage of missing values within these data fields. Thus, the challenge of a large number of missing values in these repositories must be handled in order to derive valuable information about the variables to be used to build defect estimation models, for the benefits of both software organizations and practitioners.

In this research work, the ISBSG data repository has been identified as the most relevant for our research goal and objectives (based on the discussion in chapter 2): the data related to software quality in the ISBSG collection span the entire software life cycle, from project initiation to project completion.

The International Software Benchmarking Standards Group (ISBSG) was founded in 1994 by a number of national software measurement associations (ISBSG, 2013) to:

- Develop "the profession of software measurement by establishing a common vocabulary and understanding of terms".
- Provide "software development practitioners with industry output standards against which they can compare their aggregated or individual projects, and real data of

> international software development that can be analyzed to help improve the management of IT resources by both business and government"(Cukic, 2005).

The ISBSG dataset provides "*software development practitioners with industry output standards against which they may compare their aggregated or individual projects, and real data of international software development that can be analyzed to help improve the management of Information Technology (IT) resources by both business and govern*ment" (Menzies, 2008).

The data collected are assembled, evaluated, and stored in a database in Australia. A standardized extract of a number of data fields in this database is provided for a fee in the format of a Release; moreover, in addition to these ISBSG Releases, special extracts of additional data fields are available upon a specific request for research purposes in the form of an Excel file (Cheikhi et Abran, 2013).

The ISBSG database of software projects is a multi-organizational and multi-environment dataset with more than 100 data fields on more than 6,000 projects from industry and public organizations, the majority of which were collected after 2001; these projects are related either to software development and software enhancements and from various software industry sectors (Cheikhi et Abran, 2013).

The ISBSG repository collects a large number of independent variables and a considerable amount of descriptive information on the various characteristics of software projects, including quality-related data fields, through the software life cycle phases (Cheikhi, 2008).
The data fields include, for instance, information about project staffing, effort by phase, development methods and techniques, team work, project type, organization type, software process along with the various life cycle phases, technology and tools used for developing and carrying out the project, people and work effort for each project team member, software product, quality attributes, size attributes, and so on (Cheikhi et Abran, 2013).

In order to handle the issue of missing data within the ISBSG data repository there is a number of methods in the scientific literature providing solutions, such as imputation techniques.

This research work focuses on three imputation techniques:

- Single Imputation;
- Regression Imputation; and
- Stochastic Regression Imputation.

To verify the performance of the imputation techniques a verification strategy had to be developed: it consists in verifying the predictive accuracy of the estimation models obtained from the imputed data sets.

This thesis contains eight chapters. The current Introduction outlines the organization of the thesis itself.

Chapter 1 presents the literature review of Six Sigma, including: definitions, concepts, available tools and techniques, challenges and strengths, critical success factors, and different views on Six Sigma. It also introduces the ISBSG data repository including the ISBSG data collection process using the COSMIC data questionnaire, the common methods for treating the missing values, the techniques to deal with outliers, regression techniques, and criteria for the evaluation of defect estimation models.

Chapter 2 presents the research goal and motivations, the research objectives, and the research methodology proposed to achieve the research goal and its objectives: it consists in conducting empirical studies with imputation techniques on the ISBSG release 12 of 2013 with a high ratio of missing data for improving software defect estimation with Sigma defect measures. A number of imputation techniques are evaluated for dealing with missing data in context of defect estimation modeling.

Chapter 3 presents the quality-related information in the ISBSG questionnaire, maps the ISBSG questionnaire to the related measurement steps in Six Sigma (DMAIC and DFSS) methodologies. It also presents the data set preparation which consists of two levels of data preparations based on (Déry et Abran, 2005), and next analyzes the quality-related data fields in the ISBSG MS-Excel data extract (Release 12, 2013). Next, it presents an analysis of the extracted software projects of ISBSG dataset N=360 projects based on the development type and Sigma project type, and finally it identifies the steps of the strategy used to implement the imputation techniques and the activities to build defect estimation models (using 'Functional Size') with the associated statistical criteria.

Chapter 4 presents the use of the single imputation. The 'Total Number of Defects' data fields with missing values from the ISBSG R12, N=360 projects are imputed based on the absolute min-max seeds approach: random numbers are generated to fill out the missing TD values. The seed values selected for the full sample of 360 projects are set to the minimum and maximum values from the 'Total Number of Defects' data fields that do not have missing values within the dataset.

Chapter 5 presents next the use of the regression imputation. the missing values of the variable 'Total Number of Defects' from the dataset N=360 of software projects are imputed by the predicted values generated using an estimation model from the TD complete values (the complete values are observations reported within the same variable 'Total Number of Defects') as a dependent variable based on 'Functional Size' as an independent variable.

Chapter 6 presents the use of the stochastic regression imputation. It follows similar imputation steps of standard regression imputation where the missing values are imputed by the predicted values generated using an estimation model from the complete values within the dependent variable to be imputed (e.g., The 'Total Number of Defects'). The TD estimation step of those complete values is accomplished as previously on standard regression imputation based on the independent variable 'Functional Size' in CFP, and next then a residual term is added to the predicted values generated from the complete TD estimation model.

Chapter 7 presents the measurement of the predictive accuracy of the defect estimation models (based on the independent variable ‘Functional Size’ in CFP) obtained from the complete dataset and the imputed dataset. This involves developing a verification strategy for analyzing the defect estimation models results to verify the impact of the independent variable ‘Functional Size’ on the parameter estimates of the dependent variable ‘Total Number of Defects’.

Chapter 8 presents an analysis of the results of related Six Sigma aspects (as a measurement system and as improvement methodologies DMAIC and DFSS) based on the ISBSG imputed datasets after the imputation procedures with the three studied imputation techniques on N=360 software projects. The variables used are: number of software projects, software projects’ development type, software projects’ Functional size, software projects’ Total Number of Defects, software projects’ Defect Density, Sigma projects’ type (DMAIC and DFSS), and Sigma projects’ values. It also presents how the Sigma values of software projects of the imputed Dataset N=360 projects are used for a Sigma-based classification for defect estimation purposes.

The Conclusion chapter summarizes the research contributions and the recommendations and suggests future related research challenges.

CHAPTER 1

LITERATURE REVIEW

1.1 Introduction

Sigma, in statistics, measures how far a process deviates from its goal (Nanda et Robinson, 2011). Six Sigma focuses on reducing variations within processes, because such variations may lead to an inconsistency in achieving projects' specifications which represent 'defects', which means not meeting customers' satisfaction (Nanda et Robinson, 2011).

Since the 1980's, Six Sigma is registered as a trademark of Motorola in the USA (Motorola, 2004). It is based on the Edwards Deming's Plan-Do-Check-Act cycle (Tonini, Spinola et Laurindo, 2006). Six Sigma is considered as a data-driven suite of improvement methodologies based on a common philosophy and it is supported by tools for measurements and for process and product improvement (Nanda et Robinson, 2011). Six Sigma involves a long term commitment that requires a full commitment from upper management in the organization to change decision making strategies (Wang, 2008). In the last 20 years, the use of Six Sigma has increased in different industries (Wang, 2008).

Many different concepts and techniques have evolved through the past years to support the enhancement of quality in software organizations, including: Statistical Process Control (SPC), Total Quality Management (TQM), Six Sigma, Malcolm Baldrige National Quality Award, Quality Management Systems (ISO 9000), continuous improvement initiatives etc. Despite many process and quality frameworks and maturity models adopted in the IT industry, such as CMMI, ISO 9001, TL 9000, ITIL, and others (Nanda et Robinson, 2011), they do not provide complete methodologies and toolkits of statistical and quality tools for problem solving purposes, and in comparison to what Six Sigma offers (Nanda et Robinson, 2011). Six Sigma differs from other quality initiatives in that it is a top down driven, and a strict methodology that requires: "*detailed analysis, fact-based decisions, a control plan to ensure ongoing quality control of a process*" (Wang, 2008). One of the major differences between Six Sigma and other quality initiatives is that it involves a project by project approach of implementation (Feng, 2008). Six Sigma focuses on both management and technical components (Antony et Fergusson, 2004):

A. The management component involves: to select the right people for Six Sigma projects, select the right process measures, provide resources for Six Sigma training, provide clear direction to project selection, etc. (Antony et Fergusson, 2004).

B. The technical component focuses on process improvement by reducing variation using certain statistical tools and techniques adopted for problem solving purposes (Antony et Fergusson, 2004).

Six Sigma can help organizations to improve their business processes and bottom-line issues: Six Sigma implementation involves determining customer's requirements and defining defects in terms of their "*critical to quality*" parameters (Teng, 2008).

The success of Six Sigma in different industries over the last two decades has encouraged exploring Six Sigma applications in other industries, such as the software industry (Al-Qutaish et Al-Sarayreh, 2008), (Hong et Goh, 2003), (Tonini, Spinola et Laurindo, 2006), (Pan et al., 2007), (Motorola, 2005), and (Murugappan et Keeni, 2000). Although Six Sigma has been adopted by many industries, it still considered new in the software industry (Antony et Fergusson, 2004). Therefore, significant challenges are encountered when software organizations decide to adopt Six Sigma (Hong et Goh, 2003). Few research studies on Six Sigma have been published in the software literature, including different views raised on whether Six Sigma can be indeed relevant to software organizations (Hong et Goh, 2003). Other studies such as in (Antony et Fergusson, 2004), and (Mahanti et Antony, 2009) claimed that Six Sigma can bring large benefits to software organizations.

1.2 Definition

Six Sigma has evolved over the last two decades and its definition can have different meanings. For instance, Six Sigma has been extended to three levels in (Motorola, 2011):

- a measurement system,
- a methodology, and
- a management System.

The Six Sigma approach satisfies all the three levels at the same time. This research work focuses on Six Sigma software measurement and on the following Six Sigma quality improvement methodologies:

- DMAIC which it stands for ‘Define-Measure-Analyze-Improve-Control’, and
- Design for Six Sigma (DFSS).

1.2.1 Six Sigma as a Measurement System

Six Sigma can be defined as a statistical expression which measures the quality of meeting customer’s requirements. “*The term "Sigma" is often used as a scale for levels of 'goodness' or quality. Using this scale, 'Six Sigma' equates to 3.4 defects per one million opportunities (DPMO)*” (Motorola, 2011). Figure 1.1 illustrates how Six Sigma measures quality. In Figure 1.1 for example, when 30.9 % of products are without defects, the Sigma level is 1; and when 99.9997% of products are without defects, the Sigma level is 6. Fewer defects correspond to higher level of Sigma, and thus higher level of customer satisfaction: each additional Sigma level corresponds to an exponential reduction in defects (Nanda et Robinson, 2011).

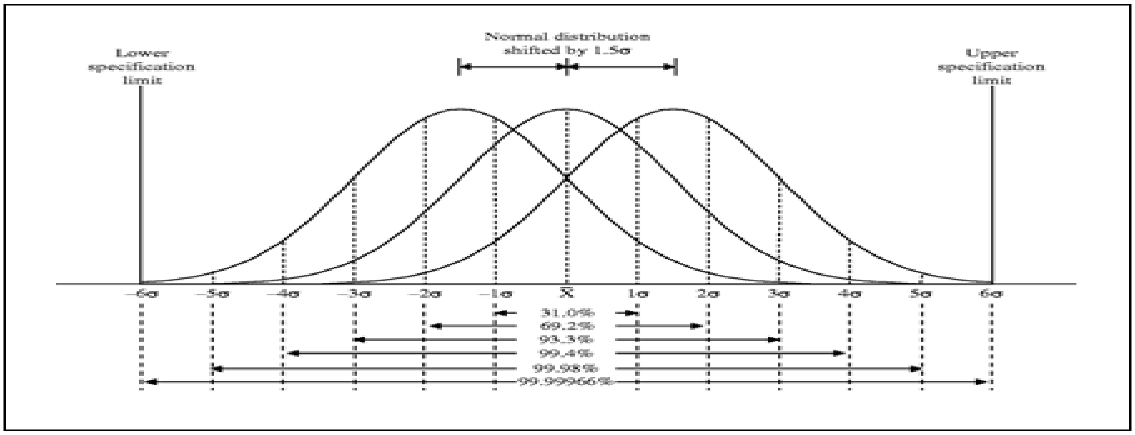


Figure 1.1 How Six Sigma measures quality (Heckl, Moormann et Rosemann, 2010)

Figure 1.1 illustrates a process that is centered with a normality distribution with mean (μ) aligned with target (T), and the specifications located six standard deviations on to the mean sides (Nanda et Robinson, 2011).

The '*sigma level'* corresponds to "*where a process or product performance falls when compared to customer specifications. In other words, the difference between the upper and lower bounds of the customer specification (denoted by the Lower Specification Limit, or LSL, and Upper Specification Limit, or USL) represents the range within which the process, product or service must fall in order to meet customer specifications, with optimal design or target (T) at the center*" (Nanda et Robinson, 2011).

The key measurements used in Six Sigma include (Nanda et Robinson, 2011):

- Critical to quality (CTQ),
- Mean (μ),
- Standard deviation (δ),
- The common Six Sigma Defect measures:
    - Defect rate: Defects Per Unit opportunities (DPU) or Defect Density (DD),
    - Sigma level,
    - Process capability indices (C_P, C_{Pk}),
    - Yield.

As result to the natural drifting that can occur in the process execution, it is observed that it over time the process mean drifts from the target by 1.5-standard deviation (Nanda et Robinson, 2011): therefore, the long-term standard deviation of the process will be greater than the observed one on the short-term (Tennant, 2001). In other words, when a process fits on '6 sigma' between the process mean and one of the nearest specification limit in a short-term data variation, it will be '4.5 sigma' in the long term fit. So the six sigma process in fact corresponds to '4.5 sigma' referred to as '6 sigma' minus the 1.5-sigma shift (Tennant, 2001). The long-term data variation, on the other hand, contains common cause variations and special cause variations (isixsigma, 2014). However the short-term data variation does not contain the special cause variation, so basically, it will have a higher process capability than the long-term data variation (isixsigma, 2014).

The calculation of Sigma level is based on the Number of Defects per Unit opportunities (DPU):

$$DPU = D/ (N*O)$$

Where:

- D: the number of defects,
- N: number of units produced, and
- O: number of opportunities per unit.

Based on this DPU formula above, the six sigma conversion scale of DPU per sigma level (Chapman, 2005) is presented in table 1.1.

Table 1.1 Six Sigma conversion scale (Chapman, 2005)

'Yield' (basically the percentage of successful outputs or operations) %	**Defects Per Unit opportunities (DPU)**	**Process Sigma**
99.99966	3.4	6
99.98	233	5
99.4	6,210	4
93.3	66,807	3
69.1	308,538	2
30.9	691,462	1

1.2.2 Six Sigma as a Problem Solving Methodology

Six Sigma provides two methodologies to solve organizations' problems: DMAIC and Design of Six Sigma (DFSS).

1.2.2.1 Six Sigma DMAIC

DMAIC stands for: 'Define-Measure-Analyze-Improve-Control' process cycle (Feng, 2008). Six Sigma DMAIC involves process improvement that can be achieved through a systematic approach for reducing variation and defects of existing processes. The DMAIC process is summarized in Table 1.2.

Table 1.2 DMAIC process (Kwak et Anbari, 2006)

Steps	**Key processes**
Define	Define the requirements and expectations of the customer. Define the project boundaries. Define the process by mapping the business flow.
Measure	Measure the process to satisfy customer's needs. Develop a data collection plan. Collect and compare data to determine issues and shortfalls.
Analyze	Analyze the causes of defects and sources of variation. Determine the variations in the process. Prioritize opportunities for future improvement.
Improve	Improve the process to eliminate variations. Develop creative alternatives and implement enhanced plan.
Control	Control process variations to meet customer requirements. Develop a strategy to monitor and control the improved process. Implement the improvements of systems and structures.

1.2.2.2 Design for Six Sigma (DFSS)

Design for Six Sigma (DFSS) is a Six Sigma approach that involves designing new or re-designing processes and products at early stages of the life cycle (Feng, 2008). Most DFSS training courses and textbooks divide the process into between four to six phases (Tayntor, 2007): they may vary within the steps included on each one (Tayntor, 2007); however, they all have similar objectives and goals (Tayntor, 2007), (Shaout et El-Haik, 2008), and (Nanda et Robinson, 2011).

This research adopts the Chowdhury's framework of IDDOV; however, it must be noted that IDDOV will be treated as five process cycle phases (Tayntor, 2007): Identification-Design-Development-Optimization-Verification. See Table 1.3.

Table 1.3 IDDOV process (Tayntor, 2007)

Steps	**Key processes**
Identification	Identify the opportunity and Define the requirements.
Design	Define initial design.
Development	Develop the high level design concepts and design alternatives to select the best design.
Optimization	Optimize the design. Develop plans for test verification; this may require simulations.
Verification	Verify the design. Implement the process in operational scale.

Besides the IDDOV framework, there are other DFSS frameworks such as:

- Define, Measure, Analyze, Design, Verify (DMADV)
- Concept, Design, Optimize, Verify (CDOV)
- Define, Measure, Analyze, Design, Optimize, Verify (DMADOV)

The Six Sigma of DMAIC and DFSS methodologies are complementary strategies and employ some of the same tools and techniques (Tayntor, 2007). However, there are differences between them and Table 1.4 (Tayntor, 2007) outlines those differences. It is important to consider when deciding whether to use DFSS techniques or the traditional Six Sigma DMAIC, whether the project involves a new process or an existing one (Tayntor, 2007): DFSS is best employed on new products and processes, while the Six Sigma DMAIC is used to improve existing ones (Tayntor, 2007).

Table 1.4 Differences between Six Sigma DMAIC and DFSS (Tayntor, 2007)

Element	**Six Sigma**	**DFSS**
Focus	Existing process	New process
Goal	Reduce variation	Reduce variation and optimize performance
Action taken	Analyze	Design
Best suited for	Maximizing current process	Developing new products or reengineering existing processes
Major effect is on	C_P (reducing variation)	C_{Pk} (centering within customer requirements)

DFSS works on the Design phase in the software life cycle, while the DMAIC comes after the Design phase of the software development life cycle (Tayntor, 2007).

DFSS share the same goals with DMAIC, and can be represented as a continuing step to Six Sigma DMAIC; it also provides a set of tools and techniques that help to reduce variation in the process design (Tayntor, 2007). The DFSS is an addition to DMAIC initiatives, not a replacement. The expected process Sigma level for a DFSS product is at least 4.5 (Tayntor, 2007), and (Shaout et El-Haik, 2008).

The goal of Six Sigma is to have processes or products that are almost defect free: achieving this goal is not as simple as it sounds (Tayntor, 2007). It requires hard working and full commitment from the organizations' top management. However, it is possible for organizations that follow the DMAIC model to adopt Six Sigma tools (Tayntor, 2007) as their statistical toolkit.

1.2.2.3 Six Sigma as a Management System

The process measurement system and the problem solving methodology are applied for process improvement which is directly related to the organization's strategy (Motorola, 2011). Motorola has found that using Six Sigma as a measurement system and as a methodology are not enough to drive the improvements in an organization (Motorola, 2011), whereas Six Sigma is used as well as a management system for achieving the organizational business strategy.

Six Sigma according to General Electric (GE): "*Six Sigma is a highly disciplined process that helps us focus on developing and delivering near-perfect products and services, the central idea behind Six Sigma is that if you can measure how many 'defects' you have in a process, you can systematically figure out how to eliminate them and get as close to 'zero defects' as possible. To achieve Six Sigma Quality, a process must produce no more than 3.4 defects per million opportunities. An 'opportunity' is defined as a chance for nonconformance, or not meeting the required specifications*" (Electronic, 2005).

Six sigma according to isixsigma: "*Six Sigma is a rigorous and disciplined methodology that uses data and statistical analysis to measure and improve a company's operational performance by identifying and eliminating 'defects' in manufacturing and service-related processes. Commonly defined as 3.4 defects per million opportunities, Six Sigma can be defined and understood at three distinct levels: measurement system, methodology and philosophy*" (isixsigma, 2011).

(Linderman et al., 2003) emphasizes the need for a common definition of Six Sigma: "*Six Sigma is an organized and systematic method for strategic process improvement and new product and service development that relies on statistical methods and the scientific method to make dramatic reductions in customer defined defect rates*".

Various other authors have provided variant definitions of Six Sigma, as illustrated in Table 1.5.

Table 1.5 Definitions of Six Sigma

Definitions	
(1)	"*Sigma is a Greek alphabet and is used in statistics as a measure to denote the standard variation in a process. More specifically sigma measures the capability of the process to perform defect free work. A defect is anything that results in customer dissatisfaction*" (Johnson et Swisher, 2003).
(2)	"*Six Sigma is a strategic, company-wide, approach by focusing on variation reduction; projects have the potential of simultaneously reducing cost and increasing customer satisfaction*" (Bendell, 2004).
(3)	"*Six Sigma is a business strategy that seeks to identify and eliminate causes of errors or defects or failures in business processes by focusing on outputs that are critical to customer*" (Seow et Antony, 2004).
(4)	"*Six Sigma is a data-driven and statistics-based approach, aims to deliver near-zero defects (as defined by customers) for the product, process, and transaction within an organization, The objective of using the Six Sigma approach is to reduce process variation, so that the process results in no more than 3.4 defects per million opportunities (DPMO) in the long term*" (Feng, 2008).
(5)	"*Six Sigma is a methodology for structured and process-oriented quality or performance improvement*" (Feng, 2008).

Table 1.5 Definitions of Six Sigma (continued)

Definitions	
(6)	"*Six sigma method is a project-driven management approach to improve the organization's products, services, and processes by continually reducing defects in the organization. It is a business strategy that focuses on improving customer requirements understanding, business systems, productivity, and financial*

	performance" (Kwak et Anbari, 2006).
(7)	"*Six Sigma is an approach provides a top-down solution to help the organization. It put the improvement efforts according to the strategy. It prepares the teams to work on the highly important projects. It drives clarity around the business strategy*" (Motorola, 2011).
(8)	"*Six Sigma is a highly disciplined, customer-oriented and bottom-line driven business improvement strategy that relies on statistical methods to make dramatic reductions in defect rates in processes; manufacturing, service or transactional*" (Antony, 2007).
(9)	"*Six Sigma has been defined as the statistical unit of measurement, a Sigma that measures the capability of the process to achieve a defect free performance*" (Wang, 2008).
(10)	"*Six Sigma has been described as a high performance data-driven approach in analyzing the root causes of business problems and solving them*" (Wang, 2008).
(11)	"*Six Sigma is an approach that improves quality by analyzing data with statistics*" (Wang, 2008).

In summary, the definition of Six Sigma in (Motorola 2011) refers to three levels: as a measurement system, as a methodology, and as a management system.

- As a measurement system, it aims to reducing defects: the highest level "6σ" equates to 3.4 defects per million opportunities.
- As a methodology, it focuses on improving the process: DMAIC and DMADV models are the most commonly used.
- As a management system, it combines the measurement system and methodologies for executing the business strategy, and aims to continuously improving product quality. Essentially, Six Sigma is all three at the same time.

1.3 Six Sigma Concepts

There are many Six Sigma aspects not accentuated in previous quality improvement initiatives, as listed in Table 1.6.

Table 1.6 Aspects of Six Sigma (Seow et Antony, 2004)

Aspects of Six Sigma	
(1)	Six Sigma strategy focuses on achieving measurable and quantifiable financial returns to the bottom-line of an organization. Six Sigma project does not get approved unless the bottom-line impact has been clearly identified and defined.
(2)	Six Sigma strategy places an importance on strong and passionate leadership and the support required for its successful deployment.
(3)	Six Sigma methodology of problem solving integrates the human elements (culture change, customer focus, belt system infrastructure, etc.) and process elements (process management, statistical analysis of process data, measurement system analysis, etc.) of improvement.
(4)	Six Sigma methodology utilizes the tools and techniques to fix problems in business processes in a sequential and disciplined way. Each tool and technique within the Six Sigma methodology has a role to play and when, where, why and how these tools or techniques should be applied is the difference between success and failure of a Six Sigma project.

Table 1.6 Aspects of Six Sigma (Seow et Antony, 2004) (continued)

Aspects of Six Sigma	
(5)	Six Sigma creates an infrastructure of Champions, Master Black Belts (MBBs), Black Belts (BBs) and Green Belts (GBs) that lead, deploy and implement the approach.
(6)	Six Sigma emphasizes the importance of data and decision making based on facts and data rather than assumptions and hunches, Six Sigma forces people to put measurements in place. Measurement must be considered as a part of the culture change.

(7)	Six Sigma utilizes the concept of statistical thinking and encourages the application of well-proven statistical tools and techniques for defect reduction through process variability reduction methods (e.g.: statistical process control and design of experiments).

According to GE Company (isixsigma, 2011), and (Mahanti et Antony, 2005): "*Six Sigma key concepts are:*

- ***Critical to Quality:*** *Attributes most important to the customer.*
- ***Defect:*** *Failing to deliver what the customer wants.*
- ***Process Capability:*** *What your process can deliver.*
- ***Variation:*** *What the customer sees and feels.*
- ***Stable Operations:*** *Ensuring consistent, predictable processes to improve what the customer sees and feels.*
- ***Design for Six Sigma:*** *Designing to meet customer needs and process capability*".

1.4 Tools and techniques in Six Sigma

Tools used in Six Sigma include qualitative and quantitative (statistical) tools for data analysis, root cause analysis, root cause validation, and identification and selection of process improvements (Nanda et Robinson, 2011):

- Qualitative tools refer to: process mapping, fishbone diagram, cause and effect matrix, five whys, failure mode effects analysis (FMEA), etc.
- Quantitative tools refer to: Kruskal-Wallis, one- and two-sample T-test, analysis of variance, confidence intervals, F-tests, one- and two-proportion tests, Monte Carlo simulation, regression, Design of Experiments (DOE), etc.

Many quality tools and techniques are adopted in Six Sigma for process improvement. They can be grouped as a combination of Quality Control (QC) tools that can be used in all phases of the improvement methodology and of all other tools and techniques that can be effective to improve process quality for software, such as (isixsigma, 2011):

- Cause-effect Diagram;

- Pareto Chart;
- Stratification
- Histogram;
- Check Sheet;
- Control Chart;
- Scatter Plot;
- Brainstorming;
- Affinity diagram;
- High level process map;
- Measurement analysis system;
- Voice of customer method;
- Kano analysis;
- Project management methods;
- Failure effect and mode analysis;
- Stakeholder's analysis;
- Process documentation
- Analyses of variance;
- Correlation and regression;
- Design of experiments;
- Process capability;
- Taguchi method; and
- Hypothesis testing.

(Antony et Fergusson, 2004) presented the results from a pilot survey on Six Sigma tools and techniques used by software industry (a total of 100 questionnaires sent to software companies: only 10 of those companies were applying the principles of Six Sigma. The majority of the respondents (85%) to the questionnaire were general managers, QA managers and business process managers. The other 15% respondents were green belts and process improvement personnel.

The following tools were the most commonly used tools/techniques used by the companies in their Six Sigma programs (Antony et Fergusson, 2004):

- Data flow diagram (DFD);
- Gap analysis;
- Process mapping; and
- Voice of the customer analysis.

The less commonly used tools/techniques include statistical process control, design of experiments, Taguchi methods, process capability analysis (PCA), COPQ analysis (Antony et Fergusson, 2004).

(Mahanti et Antony, 2009) presented the results from an empirical investigation of Six Sigma in the Indian software industry on the Six Sigma tools, metrics and techniques; a total of 100 questionnaires were sent to software companies. The criteria used to select the companies were Six Sigma certification, CMM certification, service areas and employee strength. Only twelve (12) of the 100 companies were actively applying the principles of Six Sigma. The distribution of the respondents to the questionnaire was master black belts (5 percent), green belts (10 percent), black belts (10 percent), project managers (15 percent), general managers (5 percent), Vice-President-Quality (10 percent) and others (45 percent)). The most commonly used tools/techniques included:

- Statistical process control (SPC);
- Control charts;
- Fishbone diagram;
- Gap analysis;
- Inspection;
- Regression;
- Process mapping;
- Quality function deployment (QFD);
- Failure mode and effect analysis (FMEA); and

- Process capability analysis (PCA).

The least commonly used tools/techniques included:

- SERVQUAL for measuring service quality;
- Service blueprinting; and
- Simulation.

The commonly used measures and indicators of service performance in the software industry were (Mahanti et Antony, 2009):

- Number of customer complaints;
- Defect rate;
- Cost of poor quality (COPQ);
- Defect per million opportunity (DPMO);
- Process capability indices: $C_P$and C_{Pk}; and
- Access time.

Less frequently used indicators were (Mahanti et Antony, 2009):

- Schedule variance;
- Effort variance;
- SLA compliance; and
- Schedule slippage.

(Mahanti et Antony, 2005) mentioned that the majority of the elements of the six sigma toolkit are directly applicable to every day software development data analysis:

- Fishbone diagram;
- Benchmarking;
- Pareto chart;
- Scatter diagram;
- Quality function deployment (QFD) for prioritizing requirements;
- Process mapping for work flow optimization;

- Correlation analysis;
- Failure modes effect analysis (FMEA);
- Statistical process control;
- Control charts;
- Flowcharts;
- Modeling; and
- Simulation.

Design of experiments (DOE), measurement system evaluation (MSE) tends to have less applicability to every day software development situations than they do in manufacturing applications (Mahanti et Antony, 2005).

(Janiszewski et George, 2004) also mentioned that the Six Sigma toolkit is a suite of problem-solving tools that can be used to implement the DMAIC method. The most common tools are:

- Quality function deployment (QFD);
- Process mapping, correlation analysis;
- Analysis of variance (ANOVA);
- Failure modes effect analysis (FMEA);
- Statistical process control (SPC);
- Control plans;
- Design of experiments (DOE); and
- Measurement system evaluation (MSE).

1.5 Challenges of Implementing Six Sigma: Strengths and Weaknesses

1.5.1 Weaknesses

There has been disagreement on whether or not Six Sigma should be adopted for software (Hong et Goh, 2003) and significant challenges are encountered when attempting to apply Six Sigma to the software industry.

Some challenges in implementing Six Sigma have been identified by (Seow et Antony, 2004):

- The difficulty of having available quality data in processes. Sometimes this task could take the largest proportion of the Six Sigma project time.
- The right selection and prioritization of projects is one of the critical success factors of a Six Sigma implementation is still based on subjective judgment.
- The statistical definition of Six Sigma is 3.4 defects per million opportunities. In service processes, a defect may be defined as anything which does not meet customer needs or expectations. It would be illogical to assume that all defects are equally bad when calculating the Sigma Capability level of a process.
- Measurement is required in the software development industry but it is not an easy work to measure the effectiveness of the development process. At the same time, it is quite difficult to quantify all the parameters in the software development industry (Saini et al., 2011).

Six Sigma has strong foundations in statistics (Mahanti, 2011): thus, an inadequate knowledge of statistics is another reason behind resistance to Six Sigma (Mahanti, 2011). Software professionals in software organizations who are implementing Six Sigma have identified data availability and integrity, resources and investments, and time bandwidth as potential barriers (Mahanti, 2011). Another barrier to the successful application of Six Sigma to software is a lack of adequate product and process measures (Janiszewski et George, 2004). There is also disagreement among leaders in the software industry about the need for Six Sigma (Jacowski, 2006).

Sigma level determination is a key step in the Six Sigma (Hong et Goh, 2003). This is a critical step as it is what Six Sigma is meant to offer and it matters to the success and failure of a Six Sigma project (Hong et Goh, 2003).

1.5.2 Strengths

- One of the advantages of the Six Sigma methodology over other process improvement initiatives is that the use of data analysis tools in Six Sigma projects enables the practitioners to identify process preventing problems and demonstrate the improvements using objective data (Feng, 2008). Organizations that implement Six Sigma have benefited in three major ways: reduced defect rate, reduced operational costs, and increased value for both customers and shareholders (Antony, 2007).
- Six Sigma increases quality by reducing process variability and aligning customer's expectations, providing high financial returns (Tonini, Spinola et Laurindo, 2006).
- The essence of six-sigma for software is to prevent software from producing faults in spite of their defects rather than to build software without defects (Biehl, 2004).
- The tools and methods of Six Sigma are applied to discovery and eliminate the defects of software process, which can ensure final quality of the software product (Zhao et al., 2008).
- Six Sigma provides a series of concrete steps to carry out continuous improvement during software process, which leads to high quality in the final software product (Zhao et al., 2008).
- Many leaders in the high-tech industry have used Six Sigma to improve their operational results and profitability (Nanda et Robinson, 2011).
- Six Sigma provides software organizations with the opportunity to achieve quality and cost-effective development measures may save substantial financial resources in rework and waste, and also to ensure business continuity (Hong et Goh, 2003).
- Six Sigma can be valuable in implementing an effective and sustainable software process improvement (SPI) initiative. It can be used by itself or in conjunction with a model-based approach like Capability Maturity Model Integration (CMMI) (Janiszewski et George, 2004).
- Most of the software organizations who applied Six Sigma have achieved significant financial savings and cost reductions through the application of Six Sigma (Mahanti, 2011). Other benefits include remarkable reduction of defects, greater project success

rates, cycle time reduction, and reduction in process variation and an increase in customer satisfaction (Mahanti, 2011).

1.6 Critical success factors of implementing Six Sigma

The key to implement Six Sigma successfully is aligning with its critical success factors (Feng, 2008). Some authors stated the key ingredients for the effective Six Sigma introduction and implementation in organizations are the following (Kwak et Anbari, 2006), (Antony et Banuelas, 2002), and (Coronado et Antony, 2002):

- Management commitment and involvement,
- Organizational infrastructure and Training,
- Understanding of Six Sigma methodology, tools, and techniques,
- Linking Six Sigma to customers,
- Project selection, Project management, control skills (reviews and tracking),
- Cultural change, and
- Communication.

Other researchers have been identified various critical success factors of implementing Six Sigma in software (Nanda et Robinson, 2011), (Antony et Fergusson, 2004), (Mahanti et Antony, 2009), and (Mahanti, 2011) - see Appendix XV.

1.7 Different views on applying Six Sigma in software organizations

Some sources in the scientific literature have discussed the applicability of Six Sigma in software development projects such as (Antony et Fergusson, 2004), (Binder, 1997), (Hong et Goh, 2003), (Mahanti et Antony, 2006), and (Mahanti et Antony, 2009). Researchers have been divided into two groups, with or against the applicability of it. Some researchers have also identified a number of uncertainties on applying Six Sigma to software (Al-Qutaish et Al-Sarayreh, 2008). The following summarizes such viewpoints:

(Seow et Antony, 2004) stated: "*When Six Sigma was introduced to many organizations, the initial reactions varied from a lot of enthusiasm to an absolute skepticism.*

(Fehlmann, 2004) identified a six sigma approach to software development, and mentioned three principles based on the experience of implementing Six Sigma for software:

- Principle 1: Measure customer related metrics only.
- Principle 2: Adjust to moving targets (your goals may need change; accept change and manage it accordingly).
- Principle 3: Enforce measurement (Do not enforce meeting targets).

(Siviy et Forrester, 2004) conducted a research project to investigate the use of six-sigma to accelerate the adoption of CMMI and they concluded the following:

- Six Sigma helps integrate multiple improvement approaches to create a seamless, single solution.
- Rollouts of process improvement by six-sigma adopters are mission-focused as well as flexible and adaptive to changing organizational and technical situations.
- Six Sigma is frequently used as a mechanism to help sustain (and sometimes improve) performance in the midst of reorganizations and organizational acquisitions.
- Six Sigma adopters have a high comfort level with a variety of measurement and analysis methods.
- Six Sigma can accelerate the transition of CMMI:
    - Moving from CMMI ML 3 to 5 in 9 months, or from SW-CMM Level 1 to Level 5 in 3 years (the typical move taking 12-18 months per level).
    - Underlying reasons are strategic and tactical.
- When Six Sigma is used in an enabling, accelerating, or integrating capacity for improvement technologies, adopters report quantitative performance benefits, using measures they know are meaningful for their organizations and clients.

(Hong et Goh, 2003) stated that a recurring debate is the applicability of Six Sigma methodology in the software industry.

(Al-Qutaish et Al-Sarayreh, 2008) stated that the Six Sigma concepts could be applied to the software engineering, but it needs some customization. In addition, applying Six Sigma to a software engineering process could be extended to the software product through the transformation of the product quality characteristics values into sigma values for all types of software product. Taking into account the different software product types may have different quality requirements in particular when some software products are very sensitive to the quality, such as control systems, real-time systems, etc.

There are some misconceptions that Six Sigma to software; for instance:

- A misconception is that Six Sigma is only helpful if the whole software organization has adopted it. But there are benefits of adopting Six Sigma tools and techniques and incorporating the processes into software development, even if the whole organization is not using Six Sigma (Hong et Goh, 2003).
- Other misconception is that designing a Six Sigma program is very expensive (Hong et Goh, 2003). However, if the project design is still in the early phases of development, so it has minimal cost regarding quality improvement, therefore, if organization waits until the testing phase of development in order to detect the defects, so the cost for resolving those defects and problems will be very high.

The following are some examples of using some Six Sigma concepts in software by researchers:

- (Al Qutaish, 2007) extended Six Sigma to the software product through the transformation of the product quality characteristics values into sigma values for all types of software product. Taking into account that different software product types may have different quality requirements since some software products are very sensitive to the quality, such as control systems, real-time systems, etc.
- (VanHilst, Garg et Lo, 2005) proposed that the Global software Development Environments (GDEs) can be extended with a DMAIC framework (methodology) to interactively provide required metrics and analyses.

- (Pan et al., 2007) proposed a framework to support Six Sigma projects for continuous process improvements for software developments.
- (Zhao et al., 2008) established the software process management model based on Six Sigma, and carried on a case analysis. The results indicate that this method is feasible in the software quality management.
- (Tonini, Spinola et Laurindo, 2006) suggested some improvements on the DMAIC method and proposed a specific roadmap for Six Sigma projects on software development process improvement.
- (Redzic et Baik, 2006) presented the six sigma DMAIC approach which is used for software quality improvement.
- (Shenvi, 2008) decided to deploy the Design for Six Sigma (DFSS) techniques to software development. This was the first time that the DFSS concept was used directly for improving software product quality.
- (Nanda et Robinson, 2011) presented number of case studies on implementing Six Sigma in software organizations, these case studies presented how Six Sigma methodologies (DMIC and DFSS) used for problem solving and claimed that these organizations have successfully deployed Six Sigma such as: reducing business risk, reducing cycle time of software development processes, addressing defect reduction, and achieving productivity improvement.
- (Mahanti et Antony, 2009) presented the results from an empirical investigation of Six Sigma in the Indian software industry on the Six Sigma tools, metrics and techniques; a total of 100 questionnaires were sent; data are collected by means of questionnaires or interviews, the response rate from the organizations was about 20% (e.g., 20 organizations), 41.67% of the respondents implementing Six Sigma have completed more than 30 Six Sigma projects. 33.33% have completed between five and ten Six Sigma projects. The rest, 25%, have completed less than five Six Sigma projects, also claimed that 25% of the respondents implementing Six Sigma had their core processes operating between 3 sigma and 4 sigma.

1.8 Why software organizations should choose Six Sigma?

The significance of Six Sigma is illustrated in terms of legal responsibility, mission critical systems, complex systems and the customer driven software industry in general (Hong et Goh, 2003):

- *Legal Responsibility* – Six Sigma approach helps to fulfill the legal responsibility, one of the most significant benefits of the Six Sigma: many software systems and packages are distributed and installed in identical or similar copies, all of which are vulnerable to the same failure.
- *Mission Critical Systems* – Some software organizations are developing mission critical systems. The failure of a mission critical results in a great loss to society. Six Sigma means 3.4 defects per million opportunities: this can prevent the software to fail. The benefit of Six Sigma to mission critical systems is rather significant.
- *Complex Systems* – The application of Six Sigma can be effective in case of complex systems. Consider a system that consists of modules designed to Six Sigma rather than 3 Sigma. The cumulative effects on the complete system are quite significant. For example, for a system that consists of 100 modules, if all parts are designed to 3 Sigma, the probability of getting a defect-free system is 0.001. If all parts are designed to Six Sigma, the probability of a working 100-module system is 0.9997 (Binder, 1997). The benefit of Six Sigma is more significant in the case of complex systems.
- *Software Company* – Some software size can be very large and may have a very large number of lines of code. It has more possibility to have many defects.

Summary of elements for research contribution:

Six Sigma has achieved recognizable success over the past 20 years in many industry segments but it is not much used in the software industry and few studies have been conducted on its applicability within the software industry. Thus, further empirical studies should explore how Six Sigma can be used in software industry and can help to obtain a number of Sigma information regarding software projects such as using Sigma defect measures for defect estimation models. Six Sigma has three perspectives:

- as a Sigma measurement perspective such as: Sigma level,
- as an improvement methodology perspective, and
- as a management perspective for achieving the organizational business strategy.

Therefore, this research work focuses on the two perspectives of interest: as a Sigma level and as related measurement steps in improvement methodologies (DMAIC and DFSS).

1.9 The International Software Benchmarking Standards Group (ISBSG)

1.9.1 ISBSG Data repository

In software engineering, the data collected for empirical studies is very important. Data repositories such as the ISBSG provides a free set of questionnaires to collect data on software projects, including software functional size measured with standard measurement methods recognized by ISO. ISBSG collects data in a repository in Australia and provides an extract of data to practitioners and researchers in a MS-Excel file - see Figure 1.2.

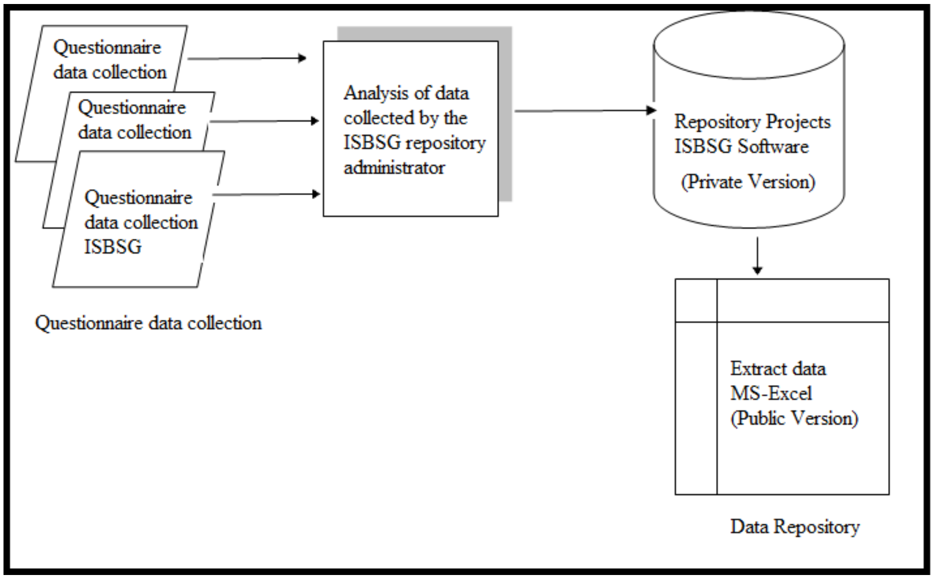

Questionnaire data collection
Questionnaire data collection
Questionnaire data collection ISBSG
Questionnaire data collection
Analysis of data collected by the ISBSG repository administrator
Repository Projects ISBSG Software (Private Version)
Extract data MS-Excel (Public Version)
Data Repository

Figure 1.2 Management of the ISBSG repository (Cheikhi, 2008)

The data collection questionnaire is available on the ISBSG website (www.isbsg.org/data-collection-questionnaires) and includes a large number of quantitative and descriptive information on the different characteristics of a software project, namely: team project effort by phase of development, the development methods and techniques, etc.

ISBSG provides to its users a dictionary of terms and measures it has defined (ISBSG, 2013) to facilitate the understanding of the questionnaire, to assist in the collection of project data in the repository and to standardize the way that the data collected are analyzed. The questionnaire consists of seven sections broken down into several sub-sections.

ISBSG offers at a modest license fee the public the data collected from various organizations around the world, with different methodologies, techniques and phases of the software life cycle, and in standard format (Cheikhi, 2008). For example, ISBSG provides useful data for multiple purposes, namely the comparison of productivity models, models for estimating the effort, etc. (Cheikhi, 2008). Such models can be used by organizations to improve their capacity in terms of planning and control of projects. In addition, the ISBSG repository collects a large number of numeric data on the different characteristics of the software project, including with its various project phases from planning to completion (Cheikhi, 2008). The ISBSG collects data related to software quality that span the entire software life cycle, from project initiation to project completion.

1.9.2 ISBSG Internal View

The internal view of the ISBSG data repository corresponds closely to their data collection questionnaire, with some additional fields added by their repository manager (Cheikhi, Abran et Buglione, 2006). The ISBSG provides a glossary of terms and measures (ISBSG, 2013) to facilitate the understanding of the data collection questionnaire, to assist the users when they collect data and to standardize the data gathering process (Cheikhi, Abran et Buglione, 2006).

The data repository of the ISBSG (ISBSG, 2013) is a publicly available multi-company data set which contains software project data collected from various organizations around the world from 1989 to 2013. This data set has been used in number of studies focusing on software estimation, such as in (ISBSG, 2013) to estimate software effort.

For example, the ISBSG provides data are related to:

- Defect prediction: such as number of defects recorded during the various software life cycle phases, effort, size in Function Points and LOC (Lines Of Code), number of requests for specification changes during the software life cycle, type of application, etc (Cheikhi et Abran, 2013).
- Effort prediction: such as effort by phases, summary work effort, normalized work effort, etc.

The ISBSG questionnaire contains six parts (Cheikhi, Abran et Buglione, 2006):

- Project attributes;
- Project work effort data;
- Project size data (function points);
- Project quality data;
- Project cost data;
- Project estimation data.

ISBSG is a not-for-profit organization and it exploits two independent repositories of IT data to help improve the management of IT globally (ISBSG, 2013):

1. Software Development and Enhancement Repository – over 7,500 projects (Release 1, 2016).
2. Software Maintenance and Support Repository – over 1100 applications (Release 2013).

For the purpose of software benchmarking, ISBSG collects, analyzes and reports data relating to products developed and processes implemented within organizational units in order to (Cheikhi, Abran et Buglione, 2006):

- Support effective management of the processes.
- Objectively demonstrate the comparative performance of these processes.

The projects have been submitted from 25 countries and the major contributors are: the United States, Japan, Australia, Finland, Netherlands and Canada (ISBSG, 2013). The data extract contains different types of projects: 61 percent are enhancements, 37 percent are new developments, and 2 percent are re-development projects. ISBSG reports that there are over 100 types of software applications available in the MS-Excel data extract (release 12, 2013) grouped into the following categories (ISBSG, 2013):

- Financial Transaction Process/Accounting (1163 projects);
- Transaction/Production System (510 projects);
- Management Information System (409 projects);
- Process control, sensor control, real time (232 projects);
- Financial (146 projects);
- Sales & Marketing (132 projects);
- Office information system, Executive information system and Decision support system (121 projects);
- Database, Catalogue/register of events or things (113 projects);
- Billing (97 projects);
- Network Management, Communications (88 projects);
- Web, E-Business (86 projects);
- Inventory / Ordering (86 projects);
- Other (897 projects).

The ISBSG offers 141 data fields in the data extract: they are not all necessarily filled out by the submitters since only a subset of the data fields is mandatory.

Software Functional Size is measured in function points. The four main function point counting approaches represented in the Repository are IFPUG, COSMIC, FiSMA and NESMA. Other approaches represented in the Repository include Mark II and Feature Points (ISBSG, 2013). Although the ISBSG Repository does include projects that are sized using Line of Code (LOC) these projects are not validated and should not be used for benchmarking (ISBSG, 2013).

These sizing methods are ISO certified (Symons et Lesterhuis, 2014).

- IFPUG (ISO/IEC 20296) stands for the International Function Point Users Group.
- COSMIC (ISO/IEC 19761) stands for the Common Software Measurement International Consortium.
- Mark-II (ISO/IEC 20298) stands for the MK II method and was used exclusively in the UK.
- NESMA (ISO/IEC 24570) stands for the Netherlands Software Metrics users Association.
- FiSMA (ISO/IEC 29881) stands for the Finnish Software Measurement Association.

There are various data collection questionnaires of ISBSG data that have the same structure with a slight difference in Section 'Functional size'. In this research work the COSMIC functional sizing method. The COSMIC method can be used to measure the size of a change (addition, modification or deletion) to software of one CFP, and it can also be used to measure the size of software that is added, changed or deleted (Symons et Lesterhuis, 2014), whereas it is not possible to measure the size of a change to a software component with the IFPUG method for example: IFPUG can only be used to measure the size of software components that are added, changed or deleted (Symons et Lesterhuis, 2014). For more details about the differences between the COSMIC and IFPUG methods - see Appendix XVII.

The ISBSG data collection questionnaire includes 7 sections divided into subsections (Symons et Lesterhuis, 2014) - see Figure 1.3:

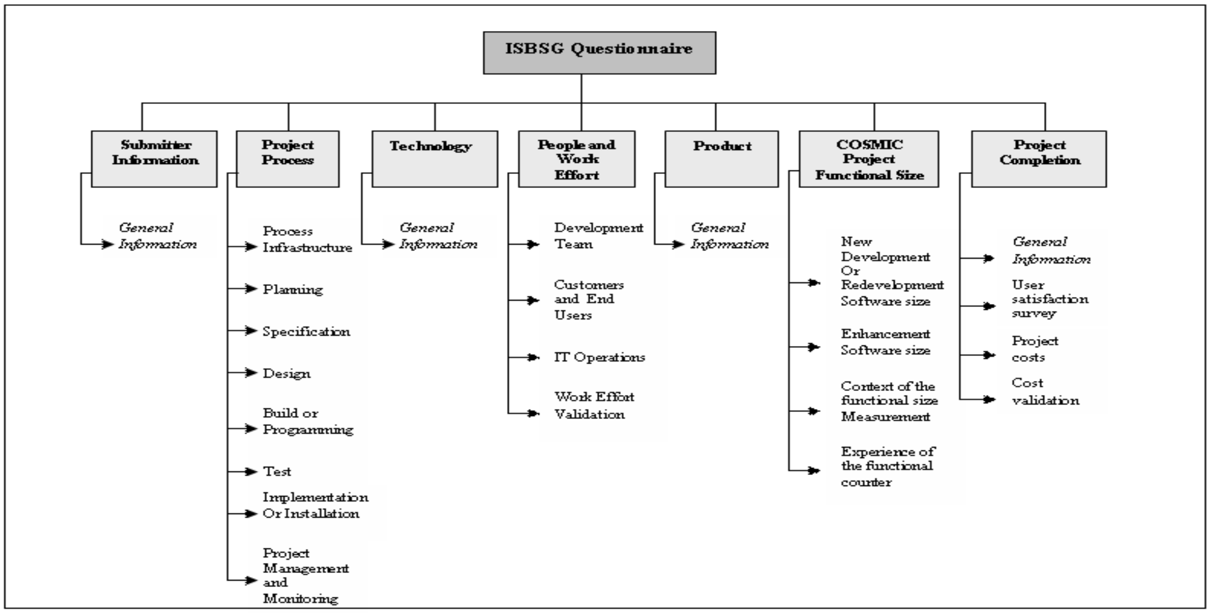


Figure 1.3 Structure of the ISBSG COSMIC Data Collection Questionnaire (Cheikhi, Abran et Buglione, 2006)

A. Submitter Information: collects the submitter's details, which are kept confidential to ISBSG.

B. *Project Process*: collects information about how the project was performed.

C. *Technology:* collects information about the technology used on the project.

D. People and Work Effort: collects descriptive information about the people who worked on the project and the effort they expended.

E. *Product:* collects description about the software product or application created or enhanced.

F. *COSMIC Project Functional Size:* collects the amount of functionality of the project delivered. The ISBSG COSMIC questionnaire collects quantitative information about data movements (ENTRIES, EXITS, WRITES and READS) by project types: new development, redevelopment software, or enhancement software.

G. *Project Completion:* collects overview information on the project completion.

1.9.3 Anonymity of the data collected

The ISBSG recognizes the imperative of guaranteeing the anonymity of the organizations that submit data to its repositories. The ISBSG carefully follows a secure procedure to ensure that the sources of its data remain anonymous. Only submitters can identify their own projects/applications in the repositories using the unique identification key provided by the ISBSG manager on receipt of a submission.

1.9.4 Extract data from the ISBSG data repository

The ISBSG assembles this data in a repository and provides a sample of the data fields to practitioners and researchers in an Excel file. All of the information on a project is reviewed by the ISBSG data administrator and rated in terms of data quality (from A to D). In particular, the ISBSG data administrator looks for omissions and inconsistencies in the data that might suggest that its reliability could be questioned.

1.10 The PRedictOr Models In Software Engineering (PROMISE) repository

The PRedictOr Models In Software Engineering (PROMISE) repository was begun in December, 2004, by Sayyad Shirabad and Tim Menzies to encourage the development of predictive models for software engineering (Menzies et al., 2012).

The first version of the PROMISE repository was created from NASA data, and hosted at the University of Ottawa (Canada). This repository contains a set of datasets and provided free of charge to the public, by the software engineering community to serve researchers and the software industry. In year 2006, the PROMISE repository was contained 23 datasets and then in 2013 it has been expanded to become 84 datasets (http://promise.site.uottawa.ca/SERepository). The datasets are grouped by PROMISE members into 5 categories, based on the addressed topic (Cheikhi et Abran, 2013) - see Table 1.7. Since 2004, the number of datasets submitted to PROMISE has been increasing, as this community now recognizes the importance of the data for conducting studies and gaining a better understanding of ways to successfully achieve their objectives (Cheikhi et Abran, 2013), such as increasing productivity, improving quality, etc.

Table 1.7 Repository and datasets (Cheikhi et Abran, 2013)

PROMISE Repository and categories	**Number of Datasets**
Defect prediction	54
Effort prediction	12
Text mining	8
Model-based software engineering	3
General	7
Total	84

The five categories are:

1. Defect prediction, with 54 datasets.
2. Effort prediction, with 12 datasets.
3. Text mining, with 8 datasets.
4. Model-based software engineering, with 3 datasets.
5. General, with 7 datasets.

1) Defect prediction: Defect prediction category has the largest number of datasets in the PROMISE repository 54 datasets out of 84. Each dataset is related to a specific purpose with locally based definitions of attributes collected and number of instances (Cheikhi et Abran, 2013). For example:
   a. The AR1 to AR6 datasets.
   b. The Bugreport dataset.

2) Effort prediction: The effort prediction category includes 12 datasets out of 84 (Cheikhi et Abran, 2013). For example:
   a. Coc81, Cocomo_sdr, and Nasa93.
   b. Kemerer and Albrecht.

3) Text Mining: The text mining category includes 8 datasets. Most of them are donated by NASA between 2005 and 2008: the MODIS dataset concerns requirements and their traceability, the NFR dataset concerns non-functional requirements (Cheikhi et Abran, 2013). The Project and Issue Tracking System (Pits) data have been collected for more than 10 years, and include issues on robotic satellite missions and human-rated systems captured by NASA's IV&V Program for software testing (Menzies, 2008).

4) Model-based software engineering: this category includes 3 datasets, provided since 2009 and collected for different purposes. The CM1-bn dataset contains data on the quality measures of 6 attributes collected are (Gay et al., 2010):
   - Change effort,
   - State,
   - Average cyclomatic complexity,
   - Average module size,
   - Probability,
   - Comment ratio.

5) General: The general category includes 7 datasets. Such as:
   - Reuse: It contains a set of 29 software project management, process, and product attributes on 24 projects (Cheikhi et Abran, 2013).
   - Nickle, XFree86, and Xorg: These datasets provide data generated from CVS archive files of the Nickel, Xorg, and XFree86 open source projects (Bart, 2005).

Because of the size PROMISE data sets, it is difficult for researchers to quickly find in them what is relevant to their work. (Cheikhi et Abran, 2013) presented a structured overview of these datasets, which will allow researchers and practitioners to find the information they need more quickly.

(Cheikhi et Abran, 2013) identified the PROMISE repositories and presented their classification framework, using the following classification criteria - see Tables (1.8, 1.9, 1.10, 1.11, and 1.12) in Appendix XVI:

- Dataset name.
- Year the dataset was originally donated.
- Dataset source.
- Availability of the descriptions of the attributes.
- Availability of the data file.

(Cheikhi et Abran, 2013) has observed only 37 datasets provide both available data file and description of attributes of the datasets; therefore many datasets cannot be used directly without contacting the dataset owners (Cheikhi et Abran, 2013).

(Cheikhi et Abran, 2013) has also observed during the survey of the PROMISE datasets that out of the 84 datasets, only 14 datasets (17%) reported past usage and the remaining 70 datasets (83%) do not, and indicated that, there are several possible reasons for such a lack of referencing to the past usage of the PROMISE datasets, such as: The data sources may not have provided this information; the only past usage might be the reference paper given; or the datasets had not been used before.

(Cheikhi et Abran, 2013) recommended that the quality of the PROMISE repository needs to be improved, and suggested that this requires a joint effort of three groups of participants (data repository owners, owners of the datasets, and researchers and practitioners):

- The owners of the repository should check the availability of the following information, before accepting the dataset:
    - the year the dataset was made available on the PROMISE website,
    - the source of the dataset or the donors' names,

  - the reference for the paper in which it was used, the number of attributes and their number of instances, the past usage of the dataset, if any, a description of the attributes and useful information about the dataset.
- The owners of the datasets should regularly check the availability of the data files through the links provided in the PROMISE repository, and update them whenever necessary.
- The researchers and practitioners of the available datasets should provide references for the published papers that are using the datasets.

Summary of elements for research contribution:

The past usage information is not readily available on the majority 70 datasets of the 84 PROMISE datasets. However, the past usage for the ISBSG dataset is available and up to date, and documented on the website (http://isbsg.org/tag/research-papers) in two ways: research papers that have used ISBSG dataset or referred to it, and research projects that have used the ISBSG dataset.

Moreover, the PROMISE datasets offer a limited number of attributes, which mainly concern source code measurements of the available software product during the final phases of the product software development cycle, not at the different stages of the software lifecycle (Cheikhi et Abran, 2013), such as the case of the ISBSG repository.

ISBSG practitioners have made their own data publicly available since 1994, while the software engineering research community only began to share their data later in 2004 (Cheikhi et Abran, 2013), although these data were available before that time (Cheikhi et Abran, 2013).

For this research work, the ISBSG data repository is selected as the appropriate one, and in particular because the ISBSG collects data on the quality of software that spans the entire life cycle of a software project, from its inception to its completion.

1.11 Methods for treating the missing values

Many studies have been conducted to tackle the missing data problem and some techniques have been proposed to handle such problem. Many of these techniques have been widely used in different sectors, such as in the medical sector; in software engineering however, only a few authors have used them only as an attempt to handle missing data, such as in research work of effort estimation models (Bala, 2013).

This section describes some of the common techniques used in the literature in general to deal with the missing data, and also shows the common limitations of their usage and impact when to decide to handle the missing data.

This section discusses:

A. The deletion methods.
B. The imputation methods.

A. The deletion methods are:

- Listwise deletion (LD), and
- Pairwise deletion (PD).

B. The imputation methods are:

- Hot-deck imputation (HDI),
- Cold-deck imputation,
- Mean imputation (MI),
- Single imputation (SI),
- Regression imputation (RI), and
- Stochastic Regression imputation (SRI).

1.11.1 Deletion methods for treatment of missing values

The missing data deletion techniques consist of deleting the fields that contain missing data, and because of their simplicity, they are widely used (Roth, 1994); but this may not lead to the most efficient utilization of the data because such handling can incur a bias in the data unless the values are Missing Completely at Random (Song et Shepperd, 2007). Consequently they should be used only in situations where the amount of missing data is very small (Song et Shepperd, 2007). Researchers have been cautioned against using the deletion methods because they have been shown to have serious limitations (Schafer, 1997).

- **Listwise deletion**

Listwise deletion is also referred as to Casewise deletion, or complete case. This method uses only the data fields that do not have missing values. Because of its simplicity, this may result in many observations that are being deleted can be desirable (Graham et Schafer, 1999). This method is generally acceptable only if there is small number of missing values and also when the data is randomly missing within the data set that is being used (Song et Shepperd, 2007). The listwise deletion method is the simplest technique where all the missing data are removed (Van Hulse et Khoshgoftaar, 2008). When the analyst discards the project with missing data on any of the variables selected and proceeds with the analysis using standard methods (Graham, 2012), then the results of the analysis will be unbiased (Graham, 2012). However, this procedure can lead to a large loss of the observations, which may result in a small data set if the number of the missing data fields are high, in particular when the original data set is small: this situation often occurs for software project estimation (Myrtveit, Stensrud et Olsson, 2001), and (Song et Shepperd, 2007). If the deleted data fields do not represent a random sample from the entire population, the inference will be biased (Mockus, 2008). Also, fewer data fields result in less efficient inference (Mockus, 2008).

- **Pairwise deletion**

Pairwise deletion is also referred to as the available case method. This method considers each data field separately where the fields that contain data will be considered and the ones that do not will be removed from the data set in order to reduce the number of data fields being removed, which may result of using the listwise deletion method (Bala, 2013); however, this approach will result in changing the sample size for each considered data field. Note that pairwise deletion becomes like the listwise deletion when all the data fields are needed for a particular analysis, e.g. multiple regression analysis (Bala, 2013). This method will result in unbiased results if the data is randomly missing (Little et Rubin, 2014). Pairwise deletion needs at least three variables for this kind of approach in order to be different from listwise deletion (Mockus, 2008).

The advantage of this method is that the sample size for each individual analysis is generally higher than with the listwise method (Song et Shepperd, 2007) It is necessary when the overall sample size is small or the number of the missing data is large (Song et Shepperd, 2007).

Pairwise deletion is a procedure that focuses on the variance-covariance matrix and each element of that matrix is estimated from all data available for that element (Graham, 2012). The pairwise deletion uses of all available data (Graham, 2012); however, there is no obvious way to estimate standard errors (Graham, 2012). It also may generate an inconsistent covariance matrix in case of multiple variables that contain missing values as mentioned before; on the other hand, the listwise deletion method always generates consistent covariance matrices (Graham et Schafer, 1999).

Since the pairwise deletion method uses all of the observed data, then, it should perform better than listwise deletion method when the missing data are completely missing at small correlations and randomness (Little et Rubin, 2014), as shown in the Kim and Curry study (Graham et Schafer, 1999). Studies have shown that when the correlations are large, the listwise deletion method performs better than the pairwise deletion method (Azen et Van Guilder, 1981).

However, these methods lead to inefficient analyses and, more seriously, commonly produce severely biased estimates (Donders et al., 2006). There are more techniques to handle missing data, such as imputation techniques, that give much better results (Donders et al., 2006): these techniques are easy accessible and available in standard statistical software, such as SAS. Nevertheless, there seems to be a general lack of understanding that has limited their use by researchers (Donders et al., 2006).

(Haitovsky, 1968) stated that imputation techniques might perform better than deletion techniques, when the data set contains large amount of missing data, or the mechanism leading to the missing data is non-random.

1.11.2 Imputation methods

Most of data analysis methods only work with a complete data set; therefore, the projects with missing data fields should be filled or such projects with missing data should be deleted, and then the resulting data set is used to perform the data analysis (Song et Shepperd, 2007).

The substitution or imputation techniques fill (impute) the data fields that are missing. Any standard statistical analysis may then be done on the completed dataset. The basic idea of the imputation methods is to replace the missing data fields with estimates that are obtained based on the reported data (Colledge et al., 1978).

In most situations, simple techniques for handling the missing data (such as listwise deletion method) produce biased results, whereas imputation techniques yield valid results without complicating the analysis once the imputations are carried out (Donders et al., 2006). Imputation techniques are based on the idea that any missing data field can be replaced by a new randomly chosen value from the same source of population (Donders et al., 2006). In other words; the imputation of a missing data on a variable is replacing that missing data by a value that is drawn from an estimation of the distribution of that variable (Donders et al., 2006). For example:

- In single imputation, an estimate is used,
- In multiple imputation technique, various estimates are used (using whether single or multiple imputation actually depends on the nature of the data field(s) to be imputed).

Under the general conditions of missing at random (MAR) and missing completely at random (MCAR), both single and multiple imputations result in unbiased estimates of study associations (Donders et al., 2006).

Mostly, data is missing at random (Donders et al., 2006). Generally, when missing data are missing at random, all simple techniques for handling missing data (e.g., complete and available case analyses, and mean imputation) give biased results (Donders et al., 2006). However, more sophisticated techniques like single and multiple imputations give unbiased results when missing data are missing at random (Donders et al., 2006).

Several researchers have examined various techniques in order to tackle the problem of incomplete multivariate data in software engineering: those researchers preferably want to avoid the use of the deletion approach (Little, 1988). The imputation methods are useful in situations where a complete data set is required for a further data analysis (Switzer, Roth et Switzer, 1998). For example, in the case of multiple regressions all observations must be complete (Little, 1988).

The reason behind using the imputation methods is that it is simple to implement and no observation is removed, as the case with the listwise deletion method (Bala, 2013). However, these techniques would typically provide underestimated standard errors (Mockus, 2008).

There are more statistically methods for handling the missing data, which have been shown to perform better than the ad-hoc methods (Schafer, 1997). These methods do not concentrate on identifying a replacement for a missing value, but on using the available data to preserve the relationships on the entire data sets, such as: regression imputation method.

The common forms of imputation techniques are as follows: Hot-deck imputation (HDI), Cold-deck imputation, Mean imputation (MI), Single imputation (SI), Regression imputation (RI), and Stochastic Regression imputation (SRI).

- **Hot-deck imputation**

The Hot-deck imputation consists of filling up the missing data fields with data taken from other data fields within the same data set, which depend on the data field that has the missing values in order to select which values to be used in such data filling. Hot-deck imputation selects a data field called 'donor' that best matches the data field that contains the missing data field (Bala, 2013).

Data fields with missing data are being imputed with values obtained from complete data field within each category (Bala, 2013). It assumes that the distribution of the observed data is the same as that of the missing values (Bala, 2013). The purpose of selecting a set of donors is to reduce the likelihood of an extreme value being imputed one or more times (Little et Rubin, 2014), (Colledge et al., 1978). The hot-deck imputation appears to be a good technique for dealing with the missing data, but it actually requires a further analysis to be done before the widespread of its use (Little, 1988).

- **Cold-deck imputation**

This method is similar to the hot-deck imputation except on the selection of a donor which comes from another data set (Little, 1992). The cold deck method imputes a data field by observed data from anything other than data values for the same item in the current data set (Bala, 2013). In contrast, it still may increase the probability of type I error due to the small standard error (McKnight et al., 2007).

- **Mean imputation**

Mean imputation (MI) is also referred to as unconditional mean imputation (Song et Shepperd, 2007). This method imputes each missing data field with the mean of that observed data. The advantage of using this method is that it is simple to implement and no observations are removed from the data set, as is the case with listwise deletion method (Bala, 2013). The disadvantage is that the measured variance for that variable will be underestimated (Little et Rubin, 2014), (Switzer, Roth et Switzer, 1998).

The mean imputation technique fills the missing data with an average value over the available data, however, this procedure underestimates the variances and, co-variances (Song et Shepperd, 2007) in case of missing completely at random (MCAR) and it is likely tend to introduce biased results (Mockus, 2008). Therefore, smaller variances may reduce p-values and, also, may provide non accurate statistical significance of some predictors (Mockus, 2008). Thus, mean imputation has very poor imputation accuracy (Van Hulse et Khoshgoftaar, 2008). Mean imputation is, therefore, not a reliable imputation technique (Van Hulse et Khoshgoftaar, 2008).

- **Single / multiple imputation**

For each missing value, an imputed randomly value is used to impute each missing value, creating a complete data set. This more sophisticated imputation procedure is also called single and multiple imputation procedure (Donders et al., 2006). The procedure is more sophisticated because the imputation is based on various known characteristics of the data field, rather than only on the estimated mean of the observed subjects (i.e., overall mean imputation described previously) (Donders et al., 2006).

The estimated distribution can be an unbiased estimate of the population distribution (Donders et al., 2006). Therefore, the associations under study estimated after missing data have been completed (imputed) by the more sophisticated single imputation and using standard analytical techniques and software are unbiased (Donders et al., 2006). The imputation might lead to a larger standard error and wider confidence intervals, and the estimated standard errors are also correct and the confidence interval has the correct coverage (Donders et al., 2006).

Moreover, using such an imputation approach leads to unbiased results with correct standard errors, in situations where missing data are MCAR or MAR (Donders et al., 2006).

Single imputation is an attractive choice as a solution to missing data problems, where it represents a good balance between quality of results and ease of use (Bala, 2013). Furthermore, single imputation has been shown to provide adequate results in the presence of a low sample size or high rates of missing data (Graham et al., 1997). The imputation technique has the advantage "*of using the complete-data methodologies for the analysis and the ability to incorporate the data collector's knowledge*" (Rubin, 2004).

- **Regression imputation**

Regression imputation involves replacing each missing value with a predicted value based on a regression model (Bala, 2013). A regression model is built using the complete observations, for each incomplete observation, each missing value is replaced by the predicted value found by replacing the observed values for that observation in the regression model (Little et Rubin, 2014). Using the regression imputation may underestimate the variance and the standard error, but it performs better than single imputation in such regards, and still gives better statistical significance results than the techniques mentioned previously (of course under the constraints of amount of the missing data in the data set, the data set sample size, and the variables to be used for an estimation).

- **Stochastic Regression Imputation**

Stochastic regression also uses regression equation to predict the incomplete variables from the complete variables, but it takes an extra step of augmenting each predicted score with normality distributed residual term (Enders, 2010). Adding the residuals to the imputed values restores lost variability to the data and effectively eliminates the biases associated with standard imputation scheme. The residual term is a random value from a normal distribution with a mean of zero and a variance equal to a residual variance from the regression of complete values (Enders, 2010).

Stochastic regression imputation is actually a fairly successful attempt to deal with the lack of the error term in the regression imputation by adding the average regression variance to the regression imputations to introduce error (Enders, 2010). Stochastic regression shows much less bias than all the mentioned techniques (Enders, 2010).

Summary of elements for research contribution:
Among all the previously discussed imputation techniques, it is quite difficult to identify the one that always performs better than the other ones; therefore, the data set simplicity takes an important role, as well as the number of variables to be used for imputation, and the missing data percentage as well. Thus, a strategy is to select candidate imputation techniques that might represent a solution for the missing data problem within the data sets variables to be used for such statistical studies. This research work focuses on three imputation techniques:

- Single Imputation.
- Regression Imputation.
- Stochastic Regression Imputation.

1.12 Techniques to deal with outliers

An outlier corresponds to a data point that is far distant from other data points in statistical analysis (Bala, 2013).

"*An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism*" (Hawkins, 1980).

The identification of outliers is an important procedure to verify the relevance of the data points in multivariate analysis: either to investigate that data points lies away from all other data points, or it comes as a proactive procedure that is taken for some multivariate method, in order to preserve the results of any statistical analysis that may possibly those data points may possibly affect the accuracy of the analysis results (Davies et Gather, 1993).

These outliers may lead to misleading results when using standard methods and also may have an indication about special events or dependencies (Kuhnt et Pawlitschko, 2005).

Outliers are defined as observations in a data set which appear to be inconsistent with the other of data points within the data set (Abran, 2015). The outliers' identification procedure is often considered as means to eliminate those data points from the data set due to disturbance (Abran, 2015) and their impact on the analysis' results.

However, outliers identification does not necessarily mean to eliminate these data points from a data set to avoid any disturbance in a statistical analysis (Bala, 2013), because ether those outliers can give certain indication about the data set structure or about special events during the sampling period (Bala, 2013). Therefore, it is important to choose the suitable outliers' detection methods.

In order to deal with the outliers' data points in software engineering, authors have introduced several techniques in order to deal with these outliers in their data set analysis; however, a number of other authors did not address at all the presence of outliers (Bala, 2013).

The presence of outliers can be analyzed using Grubbs test or Kolmogorov-Smirnov test (Bala, 2013), to verify whether the variable(s) in the data set is normally distributed: it is also it referred to as Extreme Studentized Deviate (ESD) method: these studentized values measure how many standard deviations each value is from the sample mean (Abran, 2015):

- When the P-value for Grubb' test is less than 0.05, that value is a significant outlier at the 5.0% significance level;
- Values with a modified Z-score greater than 3.5 in absolute value may well be outliers; and
- Kolmogorov-Smirnov test is used to gives a significant P-value (high value), which allows to assume that the variable is normally distributed.

This research work uses the Grubbs test to identify the presence of outliers in numerical data fields.

The three methods mentioned previously are almost the same, however, the Grubbs' test is particularly easy to implement (Bala, 2013), where, the first step is to quantify how far the outlier is from the others by calculating the ratio Z as the difference between the outlier and the mean divided by the SD (Bala, 2013). If Z is large, the value is far from the others (Bala, 2013). After calculating the mean and SD from all values, including the outlier, the Grubb's test calculates a P value only for the value furthest from the rest (Bala, 2013) of the data points within the data set. Unlike some other outlier tests, Grubbs' test only asks whether that one value is an outlier, and then the data analyst can remove that outlier, and run the test again (Bala, 2013).

1.13 Defect estimation models

Software defect is any flaw or imperfection in a software work product or software process (Clark et Zubrow, 2001):

- Software work product is any artifacts created as part of software process.
- Software process is a set of activities, methods, practices, and transformations that people use to develop and maintain software work products.

A defect is frequently referred to as a fault or bug (Clark et Zubrow, 2001). Focusing on estimating those defects is very important in software quality that may affect project and product performance.

Most defect estimation techniques used in planning rely on historical data; those techniques vary in the types of the data required (Clark et Zubrow, 2001):

- Some require little data, other require more
- Some use work product characteristics, other require defect data only.

1.13.1 Regression techniques

This sub-section presents an example of a defect estimation model built with the linear regression technique from a set of completed projects. This linear regression statistical function can be considered as the algorithm of a standard reference model (Abran, 2010).

Using linear regression models for defect estimation through empirical defect estimation, that is number of defects per software size (called Defect Density), based on historical data, where it can be implemented with minimal data (Clark et Zubrow, 2001). Estimation based on size is frequently used in the scientific literature (Nam, 2014).

$$\text{Defect Density} = \frac{Total\ Number\ of\ Defects}{Functional\ Size}\ , \text{(ISBSG, 2013)}$$

The standard reference model (linear regression): A defect estimation model built using the linear regression technique is presented in Figure 1.4. The quantitative representation from the linear regression statistical technique is of the following form, that is: Total number of defects = *f* (Functional Size). In the linear regression model, this equation takes the following quantitative form: Number of defects = a × Functional Size + b.

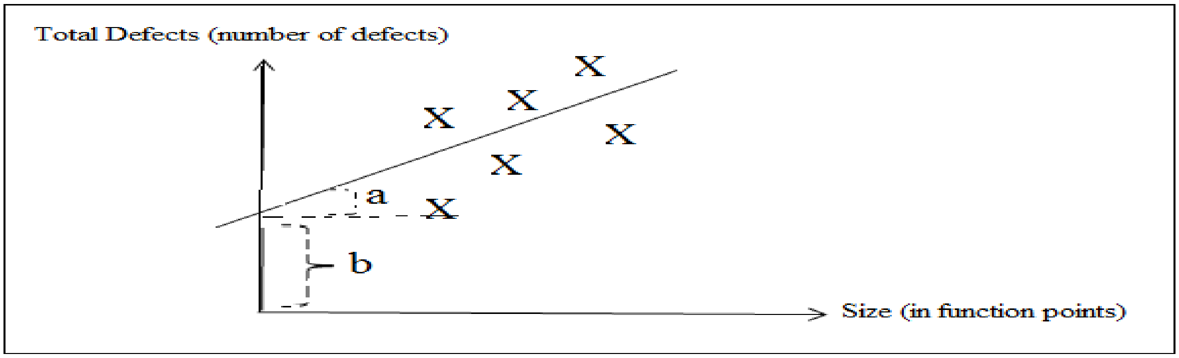


Figure 1.4 An estimation model with variables 'Total Number of Defects' and 'Functional size' (Abran, 2010)

Where, *a* represents the slope of the linear regression line, and *b* represents the point at the origin (that is, when the independent variable is = 0) (Abran, 2010).

In terms of measurement units, this equation then corresponds to:

- Total defects (defects number) = *a* (defects number / Function Point) × Functional size (in Function Points) + *b* (defects number at the origin when the functional size is 0).

A significant proportion of research on software estimation focuses on linear regression analysis; however, this is not the unique technique that can be used to develop estimation models.

Linear regression is a popular method for expressing relationship between two variables as a linear formula, but this does not mean that the determined formula will fit the data very well. Regression is based on a scatter plot, where each pair of attributes (xi, yi) corresponds to one data point when looking at a relationship between two variables. The line of best fit among the points is determined by the regression (Bala, 2013).

1.13.2 Estimation models: Evaluation criteria

For such a standard reference model (i.e. the linear regression model), a number of the well - known evaluation criteria of such statistical models are available in the literature, such as (Abran, 2010):

- *Coefficient of determination (R^2)*: the coefficient of determination (R^2) describes the percentage of variability explained by the predictive variable in the linear regression models. This coefficient has a value between 0 and 1: a R^2 close to 1 indicates that the variability in the response to the predictive variable can be explained by the model, i.e. there is a strong relationship between the independent and dependent variables.
- *Error of an estimate (Error)*: the effort of an estimate (i.e. Error = Actual − Estimate) represents the error of the estimation model on a single project. For example, the difference between the known effort of a project completed (i.e. Actual) versus the value calculated by the model (i.e. Estimate).
- *Relative Error (RE)*: The relative error (RE) corresponds to the Error divided by the Actual.
- *Magnitude Relative Error (MRE) = | Actual value – Estimate value | / Actual value.*

- *Mean Magnitude Relative Error (MMRE) for n projects = 1/n*Σ (MREi) where i= 1...n*
- *Predictive quality of the model — the prediction level of an estimation* model is: PRED (l) = $\frac{K}{N}$, where *k* is the number of projects in a specific a sample of size *n* for which MRE ≤ l.

In the software engineering literature, an estimation model is generally considered good when (Abran, 2010):

1. The MRE (Mean Relative Error) is within +/ − 25% for 75% of the observations, or
2. PRED (0.25) = 0.75.

The evaluation criteria of such statistical models are the most widely used in order to verify the performance of the software prediction models is the Mean Magnitude of Relative Error (MMRE) (Bala, 2013). The MMRE is computed from the relative error (RE), if the values of MMRE have small values, then, the results should be precise or very close to the real data (Bala, 2013). The MMRE helps to assist which the best model to select (Conte, Dunsmore et Shen, 1986).

1.14 Literature review of ISBSG-based studies dealing with missing values

- ➢ (Abran, Ndiaye et Bourque, 2007): used the ISBSG Release 6 (789 projects). An approach for building size-effort models by programming languages is presented. The relevant data for their analysis are identified by providing a description of the data preparation filtering, and the projects with no data on the programing language are removed (Abran, Ndiaye et Bourque, 2007) also removed records for programming languages with too few observations in order to form adequate samples by programming language: that left 371 records relevant to their analyses. It was then followed by a corresponding analysis that excluded 72 outliers, which left 299 projects.
- ➢ (Pendharkar, Rodger et Subramanian, 2008): used the ISBSG Release 7 (1238 projects). Projects with data quality rated (A and B) with no missing data were used to investigate

the links between team size and software size, and development effort. Only 540 projects satisfied their data quality and completeness constraint for their investigation purposes.

- ➢ (Xia, Ho et Capretz, 2015): used the ISBSG Release 8 (2027 projects). Projects with quality rated (A and B) were used. Further data filtering is applied in relation to IFPUG-sizing method, development type, effort recording and availability of all of the components of function point counting. The outliers in the ISBSG dataset are undetermined, and also the projects with missing values are removed, which left 184 projects.
- ➢ (Déry et Abran, 2005): used the ISBSG Release 9 (3024 projects). Projects with quality rated (A and B) were used. They investigated and reported the consistency of the effort data field and for each development phase. Major issues were identified in the data collection and data analysis:
  - Inconsistencies and contradiction within some data fields. The data analysts must either make an assumption on which field is the correct one or remove the projects that contain such contradictory information.
  - Number of projects with missing data in many fields lead to a few usable data samples, such data samples that have less statistical scope for analysis that corresponds to a challenge when the intended data set is to be used for such analysis. (Déry et Abran, 2005) treated the missing values within the development phases in the data set indirectly by an inference from the average values within subsets of data with similar groupings of phases without missing values. (Déry et Abran, 2005) observed and investigated outliers and the unusual patterns in terms of effort recorded in each project phase, which left 106 projects.
- ➢ (Cheikhi, Abran et Buglione, 2006) used the ISBSG Release 9 (3024 projects). Projects with quality rated (A and B) were used (2792 projects). They classified number of Projects (A and B) by Defect Severity type). 2270 projects had missing values in all the defect severity type fields and were removed, which left 522 projects with some quality-related information available for their analysis. Furthermore (Cheikhi, Abran et Buglione, 2006) removed 103 projects which had a zero value in all three types of defect fields (e.g. minor = 0, major = 0 and extreme = 0), Moreover, 55 additional projects were dropped

for the same reason; they had only zero and/or a blank in the three types of defect fields (for example, minor = 0, major = blank, and extreme = blank), and 3 projects have been removed for non-numeric values within the data set. This left 361 projects available for quality-related analysis.

- (Bala, 2013) used the ISBSG Release 9 (3024 projects). Projects with quality rated (A and B) and sized by IFPUG method were used: they extended the (Déry et Abran, 2005) work by tackling the issue of removing projects with missing values, instead; (Bala, 2013) applied an imputation technique that handles the missing values in the effort data fields and also observed and removed the outliers within the ISBSG dataset (106 projects) by using the Grubbs test as well as the Kolmogorov-Smirnov test. An analysis on the ISBSG data set with and without outliers is provided to investigate the impact analysis within the effort data fields (effort planning, effort building, effort testing, effort specification, and effort implementation).
- (Jiang, Naudé et Jiang, 2007) used the ISBSG Release 10 (4106 projects). They present an analysis of the relationships between the software size and effort. The data preparation consisted in only the software functional size in IFPUG/NESMA method function points and effort in total hours, but without any additional filtering upon the ISBSG data set, for modeling purposes; the ISBSG data set became 3433 projects, and the projects with missing values were removed: that left 540 projects for investigation purposes.

Table 1.8 Summary of ISBSG studies dealing with missing data and outliers

Paper work	**ISBSG Release**	**#No Projects in the sample**	**Missing values**	**Outliers identified and removed**
(Bala, 2013)	Release 9	3024	Observed and removed	Observed and removed

Table 1.8 Summary of ISBSG studies dealing with missing data and outliers (continued)

Paper work	**ISBSG Release**	**#No Projects in the sample**	**Missing values**	**Outliers identified and removed**
(Cheikhi, Abran et Buglione, 2006)	Release 9	3024	Observed and removed	N/A
(Déry et Abran, 2005)	Release 9	3024	Observed and investigated	Observed and removed
(Pendharkar, Rodger et Subramanian, 2008)	Release 7	1238	Observed and removed	N/A
(Jiang, Naudé et Jiang, 2007)	Release 10	4106	Observed and removed	N/A
(Xia, Ho et Capretz, 2015)	Release 8	2027	removed	N/A
(Abran, Ndiaye et Bourque, 2007)	Release 6	789	removed	Observed and removed

CHAPTER 2

RESEARCH GOAL, OBJECTIVES AND METHODOLOGY

2.1 RESEARCH GOAL AND MOTIVATION

The research goal is to improve software defect estimation (in terms of the independent variable ‘Functional Size’) with Six Sigma defect measures, based on the ISBSG data repository (software development and enhancement Repository, release 12, 2013), handling the high ratio of missing data of variable ‘Total Number of Defects’ with imputation techniques.

In the area of software engineering, the importance of data for conducting empirical and experimental studies is well recognized, as is the challenge of collecting data.

The practical research issue of predicting software defects associated with software size using collected data from previous software projects faces a set of challenges in data collection. The missing data is a common problematic issue within the existing data repositories of software projects. Therefore, such missing data problem should be handled as a prior stage for any type of data repositories as long as the amount of missing data is not high and does not affect the intended research works’ results.

This research work is in context of software defect estimation, and Six Sigma defect measures such as: Sigma level, with ISBSG data repository R12 with a high ratio of missing data in the variable ‘Total Number of Defects’, with software projects’ Functional sizes of the COSMIC sizing method. A number of imputation techniques will be evaluated for dealing with missing data in context of defect estimation modeling.

This research work focuses on three imputation techniques for missing data:

- Single Imputation;
- Regression Imputation; and

- Stochastic Regression Imputation.

The following research objectives have been selected to achieve the research goal.

2.2 REASEARCH OBJECTIVES

OBJECTIVE #1: To investigate the use of imputation techniques (Single imputation, Regression imputation, and, Stochastic Regression imputation) with the ISBSG data repository R12 for dealing with missing data within the 'Total Number of Defects' variable.

OBJECTIVE #2: To demonstrate the impact and evaluate the performance of the imputation techniques (Single imputation, Regression imputation, and, Stochastic Regression imputation) on the ISBSG data repository R12, dealing with missing data within the variable 'Total Number of Defects', for defect estimation purposes based on the independent variable 'Functional Size'.

OBJECTIVE #3: To investigate and analyze the use of the related Six Sigma aspects of defect measures and its improvement methodologies (DMAIC and DFSS) with the ISBSG data repository R12, after dealing with the missing data of variable 'Total Number of Defects'.

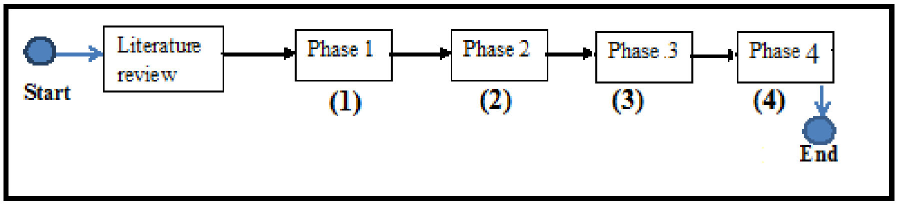
OBJECTIVE #4: To build defect estimation models (based on the independent variable 'Functional Size') along with the Six Sigma defect measures from the imputed dataset of the better imputation technique performance among: Single imputation, Regression imputation, and, Stochastic Regression imputation.

2.3 THE RESEARCH METHODOLOGY

This section presents the research methodology selected for this research work. The research methodology consists of four phases to achieve the research goal - see Figure 2.1:

1) Phase 1: Data preparation.

2) Phase 2: Implementations and comparisons for imputation techniques.
3) Phase 3: Comparisons based on complete dataset for imputation techniques.
4) Phase 4: Sigma-based defect estimation.



1). Phase 1: Data preparation.

2). Phase 2: Main contribution part 1: Implementations & Comparisons for imputation techniques

3). Phase 3: Main contribution part 2: Comparisons for imputation techniques based on complete dataset

4). Phase 4: Main contribution part 3 : Sigma-based Defect estimation

Figure 2.1 Research Methodology phases

Literature review and lessons learned

This research methodology is based on our literature review and the lessons learned in which we:

- Described and discussed the use of Six Sigma in software industry.
- Investigated the use of the two available software engineering data repositories:
  - The ISBSG data repository.
  - The PROMISE data repository.
- Discussed the available imputation techniques, and selected the most related ones to the research boundaries, for comparisons analysis purposes.
- Identified elements for candidate research problem solutions.

Phase 1 - Data preparation

- Identify the quality-related information in the ISBSG questionnaire.

- Map the quality-related information in the ISBSG questionnaire to Six Sigma (DMAIC and DFSS) methodologies.
- Analyze the quality-related data fields in the ISBSG MS-Excel data extract (Release 12, 2013).
    - Present the MS-Excel data extract preparation based on (Dery and Abran, 2005, Cheihki et al., 2007) which is consisting of two levels of dataset preparations.
- Identify a strategy for 'the imputation and defect estimation processing activities', to be followed and applied for the selected imputation techniques and the statistical analysis.

Phase 2 - Implementations and comparisons of imputation techniques

- Single imputation, Regression imputation, and, Stochastic Regression imputation are to be applied for missing values: they will be applied on the ISBSG data repository to handle the missing values.
    - Based on the identified strategy of 'the imputation and defect estimation processing activities' (in Figures 3.10 and 3.11): the imputation techniques to be implemented and compared (with and without outliers). This phase will investigate the use of the selected imputation techniques with the ISBSG repository for dealing with missing values, and will report on its use. This phase will also investigate the impact of the independent variable 'Functional Size' on the 'Total Number of Defects' parameter estimates.

Phase 3 - Comparisons based on the complete dataset for imputation techniques

- Verify the contribution of the selected imputation techniques on defect estimation models: this phase will demonstrate the impact and evaluate the performance of the imputation techniques based on an artificially missing ISBSG dataset created form the complete data within the variable 'Total Number of Defects' in order to find the best imputation technique performance on defect estimation models based on ISBSG dataset. This phase will attempt to measure the predictive accuracy of the defect estimation models (based on the independent variable 'Functional Size' in CFP) obtained from complete dataset and imputed datasets. This involves developing a verification strategy for analyzing the defect estimation models results, by verifying the impact of the independent variable 'Functional Size' on the parameter estimates of the dependent variable 'Total Number of Defects'.

Phase 4 - Sigma-based defect estimation

- Conduct Six Sigma defect measures' analysis on the imputed datasets by all selected imputation techniques: single imputation, regression imputation, and, stochastic Regression imputation.
    - Determine the Sigma values for the software projects within the imputed datasets, and determine the Six Sigma project type based on the mapped ISBSG questionnaire for collecting data with Six Sigma methodologies' steps.
    - Conduct Sigma distribution analysis followed by a discussion for the software projects within the imputed dataset of ISBSG in terms of Six Sigma defect measures purposes.
- Conduct a classification based on the Sigma values of the imputed dataset of the best performance of an imputation technique that is to be used for handling the missing data in the variable 'Total Number of Defects'. The purpose of this classification is to determine at which level of Six Sigma the datasets of software projects can be best used to build defect estimation models using the independent variable 'Functional Size' from the imputed ISBSG dataset.

The details of the research methodology phases - see Figure 2.2 - are presented next.

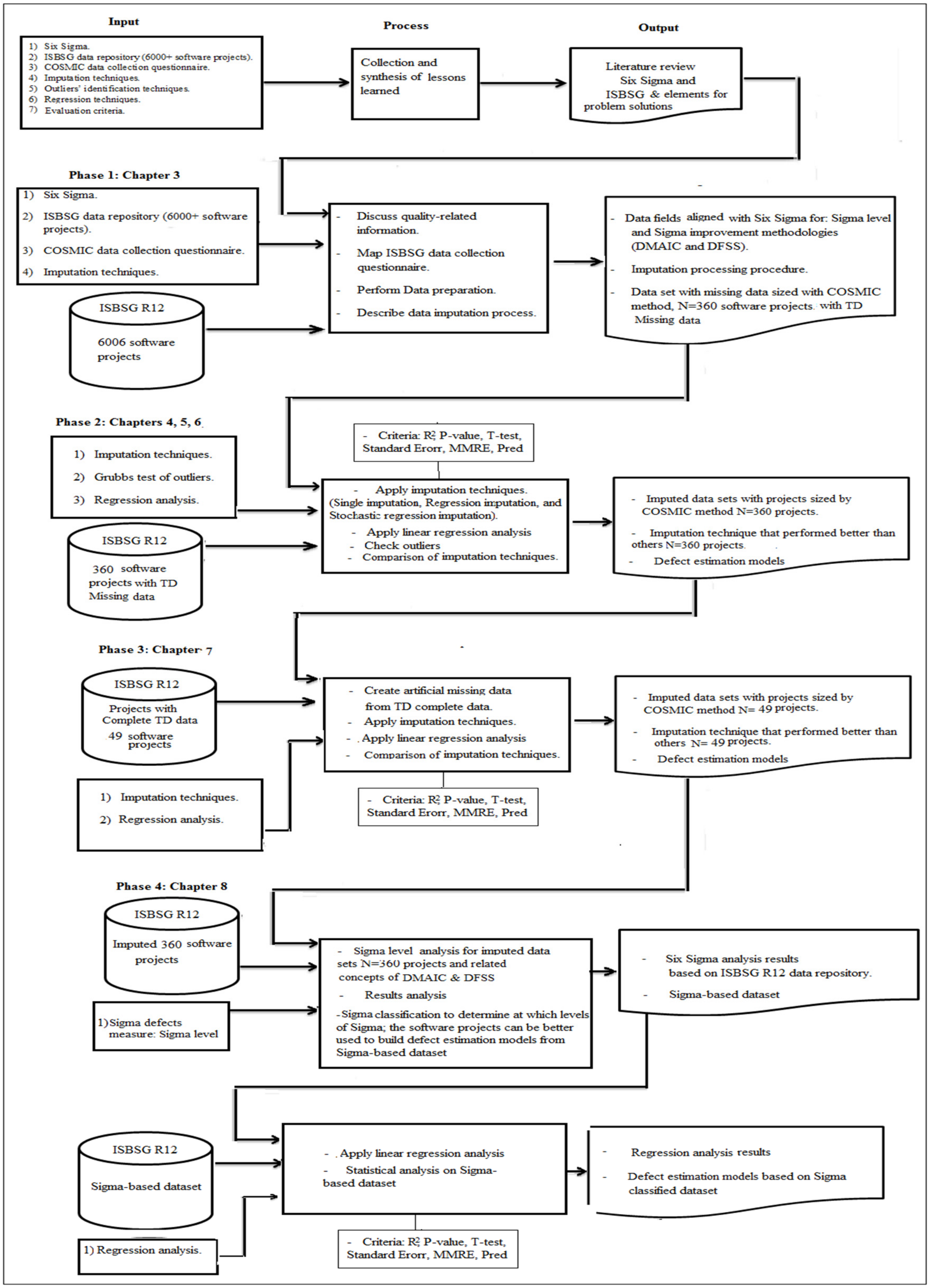

Input
Process
Output
1) Six Sigma.
2) ISBSG data repository (6000+ software projects).
3) COSMIC data collection questionnaire.
4) Imputation techniques.
5) Outliers' identification techniques.
6) Regression techniques.
7) Evaluation criteria.
Collection and synthesis of lessons learned
Literature review Six Sigma and ISBSG & elements for problem solutions
Phase 1: Chapter 3
1) Six Sigma.
2) ISBSG data repository (6000+ software projects).
3) COSMIC data collection questionnaire.
4) Imputation techniques.
ISBSG R12
6006 software projects
- Discuss quality-related information.
- Map ISBSG data collection questionnaire.
- Perform Data preparation.
- Describe data imputation process.
- Data fields aligned with Six Sigma for: Sigma level and Sigma improvement methodologies (DMAIC and DFSS).
- Imputation processing procedure.
- Data set with missing data sized with COSMIC method, N=360 software projects. with TD Missing data
Phase 2: Chapters 4, 5, 6
1) Imputation techniques.
2) Grubbs test of outliers.
3) Regression analysis.
ISBSG R12
360 software projects with TD Missing data
- Criteria: R2, P-value, T-test, Standard Erorr, MMRE, Pred
- Apply imputation techniques. (Single imputation, Regression imputation, and Stochastic regression imputation).
- Apply linear regression analysis
- Check outliers
- Comparison of imputation techniques.
- Imputed data sets with projects sized by COSMIC method N=360 projects.
- Imputation technique that performed better than others N=360 projects.
- Defect estimation models
Phase 3: Chapter 7
ISBSG R12
Projects with Complete TD data 49 software projects
1) Imputation techniques.
2) Regression analysis.
- Create artificial missing data from TD complete data.
- Apply imputation techniques.
- Apply linear regression analysis
- Comparison of imputation techniques.
- Criteria: R2, P-value, T-test, Standard Erorr, MMRE, Pred
- Imputed data sets with projects sized by COSMIC method N= 49 projects.
- Imputation technique that performed better than others N= 49 projects.
- Defect estimation models
Phase 4: Chapter 8
ISBSG R12
Imputed 360 software projects
1) Sigma defects measure: Sigma level
- Sigma level analysis for imputed data sets N=360 projects and related concepts of DMAIC & DFSS
- Results analysis
- Sigma classification to determine at which levels of Sigma; the software projects can be better used to build defect estimation models from Sigma-based dataset
- Six Sigma analysis results based on ISBSG R12 data repository.
- Sigma-based dataset
ISBSG R12
Sigma-based dataset
1) Regression analysis.
- Apply linear regression analysis
- Statistical analysis on Sigma-based dataset
- Criteria: R2, P-value, T-test, Standard Erorr, MMRE, Pred
- Regression analysis results
- Defect estimation models based on Sigma classified dataset

Figure 2.2 Detailed research methodology phases

Phase 1 - Data preparation

This phase 1 presents (Figure 2.3) the quality-related information in the ISBSG questionnaire and maps the ISBSG questionnaire to the Six Sigma DMAIC and DFSS methodologies. This phase also presents the data set preparation which consists of two levels of data preparations based on (Dery and Abran, 2005), and next analyzes the quality-related data fields in the ISBSG MS-Excel data extract (Release 12, 2013). It presents next an analysis for the extracted software projects of ISBSG dataset N=360 projects based on the development type and Sigma project type. Finally it identifies the steps of the strategy designed to implement the imputation techniques and the activities to build defect estimation models (using 'Functional Size') with the associated statistical criteria.

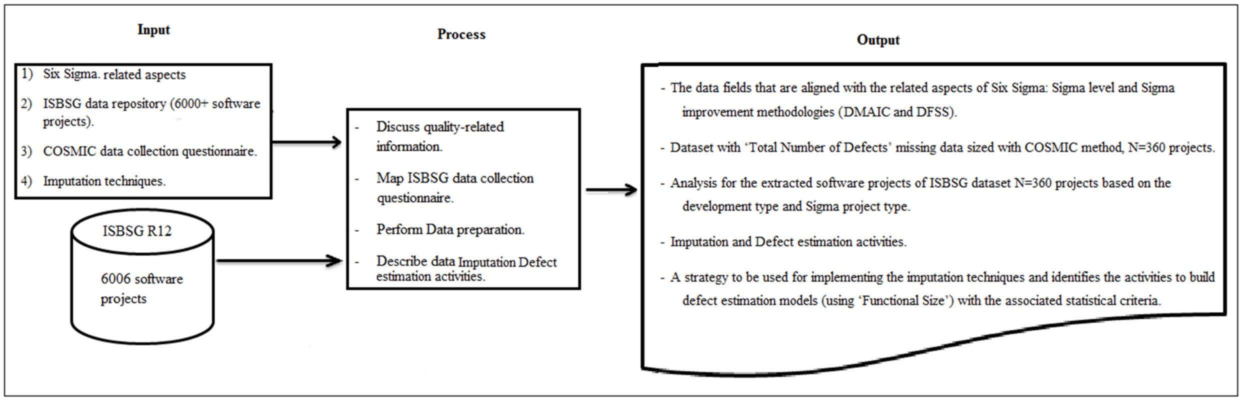

Input
1) Six Sigma. related aspects
2) ISBSG data repository (6000+ software projects).
3) COSMIC data collection questionnaire.
4) Imputation techniques.
ISBSG R12
6006 software projects
Process
- Discuss quality-related information.
- Map ISBSG data collection questionnaire.
- Perform Data preparation.
- Describe data Imputation Defect estimation activities.
Output
- The data fields that are aligned with the related aspects of Six Sigma: Sigma level and Sigma improvement methodologies (DMAIC and DFSS).
- Dataset with 'Total Number of Defects' missing data sized with COSMIC method, N=360 projects.
- Analysis for the extracted software projects of ISBSG dataset N=360 projects based on the development type and Sigma project type.
- Imputation and Defect estimation activities.
- A strategy to be used for implementing the imputation techniques and identifies the activities to build defect estimation models (using 'Functional Size') with the associated statistical criteria.

Figure 2.3 Phase 1 - data preparation

The inputs of this phase are:

- The literature review, and the lessons learned and the elements identified for research problem' solutions.
- The ISBSG data repository (6006 software projects).
- The ISBSG COSMIC data collection questionnaire.
- The imputation techniques (single imputation, regression imputation, stochastic regression imputation).

The outputs of this phase are:

- The data fields that are aligned with the related aspects of Six Sigma: Sigma level and Sigma improvement methodologies (DMIAC and DFSS).
- Dataset with 'Total Number of Defects' missing data sized with COSMIC method, N=360 projects.
- Analysis for the extracted software projects of ISBSG dataset N=360 projects based on the development type and Sigma project type.
- Imputation and Defect estimation activities.
- A strategy to be used for the implementing the imputation techniques. It also identifies the activities - statistical analysis - to build defect estimation models (using 'Functional Size') with the associated statistical criteria.

The strategy for implementing the imputation techniques and building defect estimation models consists of

1. *Create the imputed data sets*

The first step is to create the 'imputes' in order to be used to substitute the missing data. The imputation procedure needs to be identified to allow the 'imputes' to be created based on the values found across the data set for the available values of the same variable in the dataset (Bala, 2013). This involves the creation of the imputed dataset by using the three selected imputation techniques (single imputation, regression imputation, and stochastic regression imputation) in order to generate a complete dataset as an adequate representation of the data.

2. *Defect estimation modeling*

A statistical analysis is conducted on the imputed dataset. Such statistical analysis is to be achieved in order to analyze the imputed data set after accomplishing the imputation procedure - in step (1), that is to produce a complete imputed data set with no missing data within the dependent variable 'Total Number of Defects' based on an independent variable 'Functional Size' in Function Points (FP).

The modeling through a linear regression of the relationship of the dependent variable ‘Total Number of Defects’ (TD) based on an independent variable ‘Functional Size’ in Function Points is used on the imputed dataset to obtain the TD estimates and standard errors (build TD estimation models).

The statistical analysis includes:

- Estimate TD (dependent variable) based on Functional Size (independent variable).
- Analysis of TD with R^2 and P-value of the estimation results of TD using FP as the dependent variable.
- Outliers’ detection: using Grubbs test to investigate whether the outliers affects the rest of data points on TD after filling out its missing data by the three selected imputation techniques.
- Observe the values of Defect Density (DD) for each software project within the dataset of N=360 projects based on the formula of the Defect Density which measures the quality of software in terms of defects delivered in unit size of software. It is expressed as Defects per Function Points (TD⁄CFP).

The following criteria for analyzing the results of TD estimation models:

- Coefficient of determination (R^2): the coefficient has a value between 0 and 1. R^2, close to 1;
- Standard Errors (STD-E): low Standard Errors;
- Mean Magnitude Relative Error (MMRE): low values of Mean Magnitude Relative Error.
- P-value: Statistical Significance (P-value < 0.05).
- T-test: Statistical Significance (t-test > 2).
- Predictive quality of the TD estimation model: Pred(0.25) = 0.75.

Phase 2 - Implementations and comparisons for imputation techniques

This phase 2 (Figure 2.4) implements the three imputation techniques (single imputation, regression imputation, stochastic regression imputation) on the dataset N=360 of software projects with missing data in variable ‘Total Number of Defects’ data fields. This phase also conducts a modeling through a linear regression of the relationship of the dependent variable ‘Total Number of Defects’ (TD) based on an independent variable ‘Functional Size’ (in Function Points) is used on the imputed dataset to obtain the TD estimates and standard errors (build TD estimation models).

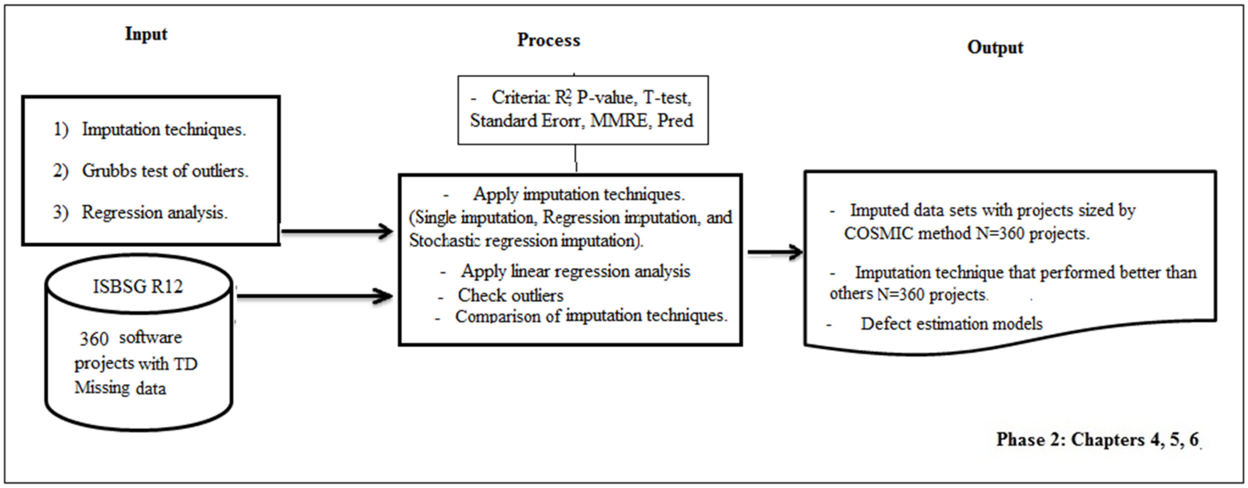

Input
Process
Output
- Criteria: R², P-value, T-test, Standard Erorr, MMRE, Pred
1) Imputation techniques.
2) Grubbs test of outliers.
3) Regression analysis.
ISBSG R12
360 software projects with TD Missing data
- Apply imputation techniques. (Single imputation, Regression imputation, and Stochastic regression imputation).
- Apply linear regression analysis
- Check outliers
- Comparison of imputation techniques.
- Imputed data sets with projects sized by COSMIC method N=360 projects.
- Imputation technique that performed better than others N=360 projects.
- Defect estimation models
Phase 2: Chapters 4, 5, 6

Figure 2.4 Phase 2 - Implementations and comparisons for imputation techniques

Single imputation, Regression imputation, and, Stochastic Regression imputation are applied for missing values on the ISBSG data repository.

- Apply the imputation procedures for handling missing values based on the imputation and defect estimation processing activities (with and without outliers): this phase will investigate the use of the selected imputation techniques with the ISBSG repository for dealing with missing values, and will report on its use. This phase also investigates the impact of the independent variable ‘Functional Size’ on the ‘Total Number of Defects’ parameter estimates.
- ‘Single imputation technique’: the ‘Total Number of Defects’ data fields with missing values from the ISBSG R12, N=360 projects are imputed based on the absolute min-max seeds approach: random numbers are generated to fill out the missing TD values. The seed values selected for the full sample of 360 projects are set to the minimum and maximum values from the ‘Total Number of Defects’ data fields that do not have missing values within the dataset.

- ‘Regression imputation technique’: the missing values of the variable ‘Total Number of Defects’ from the dataset N=360 of software projects are imputed by predicted values generated using an estimation model from TD complete values (the complete values are observations reported within the same variable ‘Total Number of Defects’) as a dependent variable based on ‘Functional Size’ as an independent variable.
- ‘Stochastic Regression imputation technique’: this step follows similar imputation steps of standard regression imputation where the missing values are imputed by: predicted values generated using an estimation model from the complete values within the dependent variable to be imputed (e.g., The ‘Total Number of Defects’). The TD estimation step of those complete values was accomplished previously on standard regression imputation based on the independent variable ‘Functional Size’ in CFP, but next a residual term is added to the predicted values generated from the complete TD estimation model.

This phase 2 uses the identified strategy (previously in phase 1) for implementing the imputation techniques and identifies the activities to build defect estimation models (using ‘Functional Size’) with the associated statistical criteria.

The inputs of this phase are:

- Imputation techniques (single imputation, regression imputation, stochastic regression imputation).
- Grubbs test for data outliers.
- A strategy to be used for implementing the imputation techniques and identifies the activities to build defect estimation models (using ‘Functional Size’) with the associated statistical criteria.
- Dataset with ‘Total Number of Defects’ missing data sized with COSMIC method, N=360 projects.

The outputs of this phase are:

- Imputed datasets N=360 software projects by imputation techniques.

- Identified Imputation technique that performed better than other imputation techniques based on the statistical criteria used.
- Defect estimation models are built based on the imputed datasets.

Phase 3 - Comparisons based on complete dataset for imputation techniques

This Phase 3 (Figure 2.5) measures the predictive accuracy of the defect estimation models (based on the independent variable ‘Functional Size’ in CFP) obtained from complete dataset and imputed datasets. This involves developing a verification strategy for analyzing the defect estimation models results and verifying the impact of the independent variable ‘Functional Size’ on the parameter estimates of the dependent variable ‘Total Number of Defects’.

This strategy for analyzing the performance of the three imputation techniques used in the empirical studies involves to work with a dataset of complete data set (e.g., it does not contain any missing value: here N=49 software projects) through creating artificially a subset by deleting the data values within the intended variable, and next, impute these artificially missing data by the selected imputation techniques, and next to generate estimation models from the original complete data set and the other imputed data sets, in order to compare and assess the estimates derived from these estimation models through evaluation criteria of such statistical models, such as: Magnitude of Relative Error (MRE).

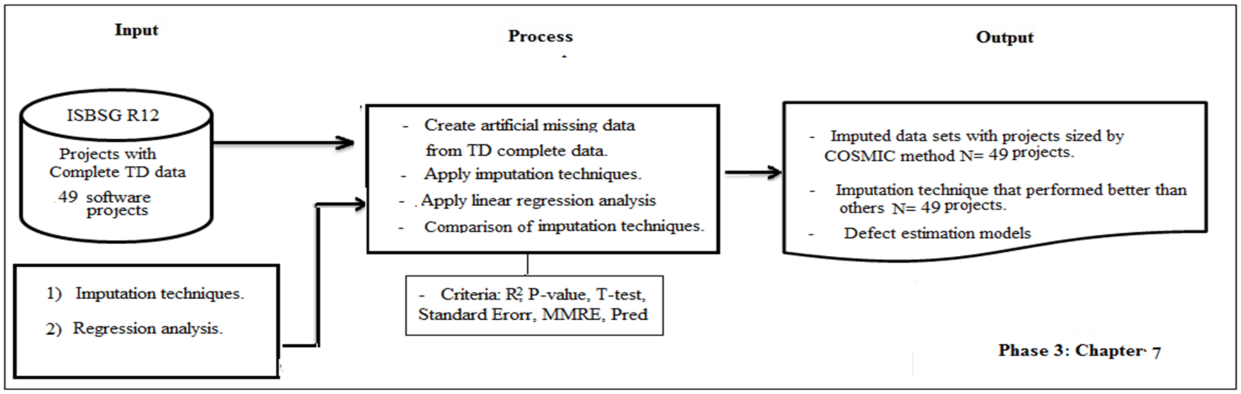

Input
Process
Output
ISBSG R12
Projects with Complete TD data
.49 software projects
1) Imputation techniques.
2) Regression analysis.
- Create artificial missing data from TD complete data.
- Apply imputation techniques.
- . Apply linear regression analysis
- Comparison of imputation techniques.
- Criteria: R², P-value, T-test, Standard Erorr, MMRE, Pred
- Imputed data sets with projects sized by COSMIC method N= 49 projects.
- Imputation technique that performed better than others N= 49 projects.
- Defect estimation models
Phase 3: Chapter 7

Figure 2.5 Phase 3 - Comparisons based on complete dataset for imputation techniques

This phase 3 uses the identified strategy (previously in phase 1) for implementing the imputation techniques and identifies the activities to build defect estimation models (using 'Functional Size') with the associated statistical criteria.

The verification strategy is designed as follows:

- Given the complete data set N sample size of projects, randomly split the data set into two subsets X and Y.
- From subset Y, delete the data values for the data field 'Total Number of Defects',
- Use *Single Imputation* (SI) technique: based on absolute seeds (min, max) for the missing values of 'Total Number of Defects' (TD) within subset Y.
- Use *Regression Imputation* (RI) technique based on replacing each missing value with a predicted value based on estimation model built using complete observations of Total Number of Defects (TD).
- Use *Stochastic Regression Imputation* (SRI) technique based on replacing each missing value with a predicted value based on estimation model built using complete observations of Total Number of Defects (TD).
- Defect estimation models (based on independent variable 'Functional Size') will be built with both the initial complete dataset N of software projects and all the imputed dataset N of software projects (N: represents number of software projects in the dataset).
- Compare the TD estimate by assessing and comparing the predictability with MMRE and Pred(25) to assess the predictability of these estimation models based on the following criteria ((Conte, Dunsmore et Shen, 1986) and (Abran, 2010):
    - Magnitude of Relative Error (MRE) = | Estimated value – Actual value | / Actual
    - Mean Magnitude of Relative Error for N projects (MMRE) = 1/n*Σ(MREi)
    - Measure of Prediction Quality = Pred(x/100)

The inputs of this phase 3 are:

- Imputation techniques (single imputation, regression imputation, stochastic regression imputation).
- Grubbs test for data outliers.

- A strategy to be used for implementing the imputation techniques and identifies the activities to build defect estimation models (using 'Functional Size') with the associated statistical criteria.
- Dataset with 'Total Number of Defects' complete data sized with COSMIC method, N=49 projects.

The outputs of this phase are:

- Imputed datasets N=49 software projects by imputation techniques.
- Identified Imputation technique that performed better than other imputation techniques based on the statistical criteria used.
- Defect estimation models are built based on the complete dataset N=49 projects and the imputed datasets N=49 projects.

Phase 4: Sigma-based defect estimation

This phase 4 (Figure 2.6) presents an analysis of the results of related Six Sigma aspects (as a measurement system and as improvement methodologies DMAIC and DFSS) based on the software projects of ISBSG imputed data sets after the imputation procedures with the three studied imputation techniques on N=360 software projects, in terms of:

- Number of software projects,
- Software projects' development type,
- Software projects' Functional size,
- Software projects' Total Number of Defects,
- Software projects' Defect Density,
- Sigma projects' type (DMAIC and DFSS), and
- Sigma projects' values.

This phase 4 also presents how the Sigma values of software projects of the imputed Dataset N=360 projects are used for a Sigma-based classification for defect estimation purposes.

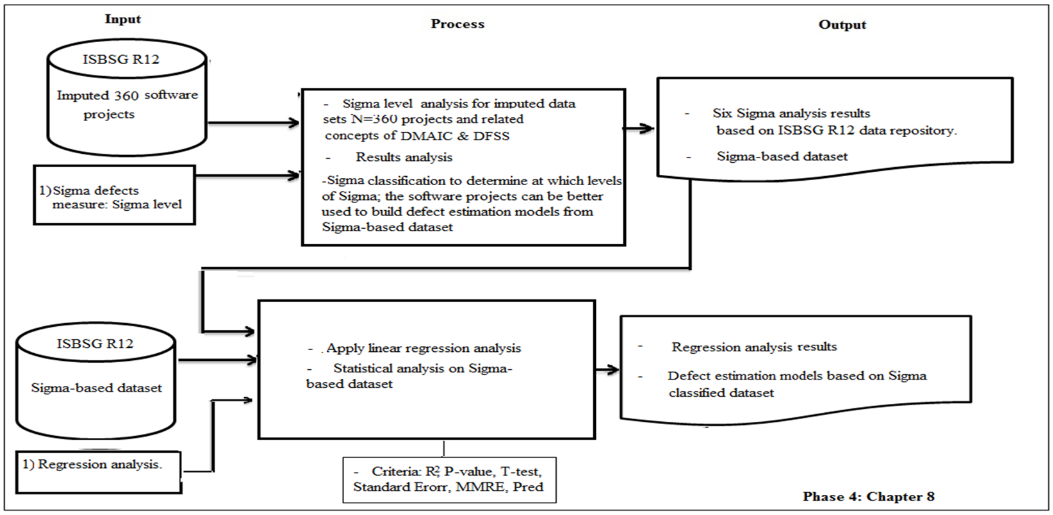

Input
Process
Output
ISBSG R12
Imputed 360 software projects
1) Sigma defects measure: Sigma level
- Sigma level analysis for imputed data sets N=360 projects and related concepts of DMAIC & DFSS
- Results analysis
-Sigma classification to determine at which levels of Sigma; the software projects can be better used to build defect estimation models from Sigma-based dataset
- Six Sigma analysis results based on ISBSG R12 data repository.
- Sigma-based dataset
ISBSG R12
Sigma-based dataset
1) Regression analysis.
- . Apply linear regression analysis
- Statistical analysis on Sigma-based dataset
- Regression analysis results
- Defect estimation models based on Sigma classified dataset
- Criteria: R2, P-value, T-test, Standard Erorr, MMRE, Pred
Phase 4: Chapter 8

Figure 2.6 Phase 4 - Sigma-based defect estimation

This phase 4 consists of three stages:

1) Sigma analysis results of software projects of ISBSG data set N=360 projects.
2) Classification of software projects based on Sigma levels of imputed Dataset N=360 projects for defect estimation purposes.
3) Statistical analysis for defect estimation.

- **Stage 1 - Sigma analysis results of software projects of ISBSG data set N=360 projects**

This stage presents the sigma values of each imputed software dataset N=360 projects that are imputed with the 3 selected imputation techniques. Software projects' sigma values are calculated through the NORMSINV Excel function, taking in consideration the 1.5 sigma shift.

- **Stage 2 - Classification of the software projects based on Sigma levels of imputed Dataset N=360 projects for defect estimation purposes**

Based on the Sigma values of the imputed dataset of software projects, N=360 projects (e.g., the Sigma values of the imputed dataset using 'stochastic regression imputation technique' with dataset N=360 projects): the software projects are classified based on their Sigma values. The purpose of this classification is to determine at which levels of Sigma; the software projects can be better used to build defect estimation models using the independent variable ‘Functional Size’.

This procedure allows producing Sigma-based datasets with software projects ranges based on Sigma levels (e.g., Sigma-based dataset with a range of software projects from 3δ to 4.5δ or more). Tables 2.1 and 2.2 illustrate **an example** of the software projects’ classification based on Sigma levels, and the datasets classification based on Sigma levels.

Table 2.1 Example of Software projects classification based on Sigma levels N=405 projects

Sigma level	**Assigned Sigma Ranges**	**Number of projects**
δ	- δ < 2	10 Projects
δ	- 2 ≤ δ < 2.5	24 Projects
δ	- 2 .5 ≤ δ < 3	66 Projects
δ	- 3 ≤ δ < 3.5	237 Projects
δ	- 3.5 ≤ δ < 4	45 Projects
δ	- 4 ≤ δ < 4.5	19 Projects
δ	- 4 .5≤ δ	4 Projects
		405 Projects

Table 2.2 Example of Datasets classification based on Sigma levels N=405 projects

Sigma Datasets	**Assigned Sigma-based Ranges**	**Total number of projects**
(1)	- From 2δ to 4.5δ and more.	398 Projects
(2)	- From 2.5δ to 4.5δ and more.	388 Projects
(3)	- From 3δ to 4.5δ and more.	335 Projects
(4)	- From 3.5δ to 4.5δ and more.	54 Projects
(5)	- From 4δ to 4.5δ and more	25 Projects
		Out of 405 Projects

The inputs of phase 4 - stages 1 and 2:

- Imputed dataset with the better performed imputation technique (e.g., stochastic regression imputation technique) N=360 software projects.
- Sigma defect measure: Sigma level.

The output of phase 4 - stages 1 and 2:

- Six Sigma analysis results of related aspects (Sigma defect measures and Sigma improvement methodologies - DMAIC and DFSS) based on imputed ISBSG datasets.
- Software projects classification based on Sigma levels, N=360 projects.
- Sigma-based datasets, N=360 projects.

❖ **Stage 3 - Statistical analysis for defect estimation.**

Based on the classified datasets of software projects based on their Sigma level; a statistical analysis is conducted in order to build defect estimation models and analyze their results. (A linear regression analysis is applied on variable 'Total Number of Defects' based on the independent variable 'Functional Size' in CFP).

This stage uses the identified strategy (previously in phase 1) of the statistical analysis activities to build defect estimation models (using 'Functional Size') with the associated statistical criteria.

The inputs of phase 4 - stage (3) are:

- Sigma-based datasets, N=360 projects.
- Strategy (previously in phase 1) of the statistical analysis activities to build defect estimation models (using 'Functional Size') with the associated statistical criteria.

The output of phase 4 - stage (3) are:

- Statistical analysis results.
- Defect estimation models based on Sigma classified datasets.

CHAPTER 3

DATASET PREPARATION

This chapter presents the phase 1 of the research project, that is the quality-related information in the ISBSG questionnaire, the mapping of the ISBSG questionnaire to the related measurement steps in Six Sigma (DMAIC and DFSS) methodologies, This chapter also presents the data set preparation which consists of two levels of data preparations based on (Déry et Abran, 2005), and next analyzes the quality-related data fields in the ISBSG MS-Excel data extract (Release 12, 2013). It also presents an analysis for the extracted software projects of ISBSG dataset N=360 projects based on the development type and Sigma project type, and finally it identifies the steps of the strategy used to implement the imputation techniques and the activities to build defect estimation models (using 'Functional Size') with the associated criteria for the selected imputation techniques.

3.1 ISBSG data collection questionnaire

The COSMIC data collection questionnaire consists of 141 questions in 7 sections (see Table 3.1). The data collection questionnaire's documentation follows the structure of such instruments: the initial summary and general description of the section, followed by a list of questions related to the section and their purpose.

Table 3.1 Number of questions within the ISBSG COSMIC questionnaire

Section	**Number of questions**
Submitter information	4
Project process	51
Technology	9
People and work effort	23
Product	7
COSMIC project functional size	30

Table 3.1 Number of questions within the ISBSG COSMIC questionnaire (continued)

Project completion	17
Total	**141**

3.2 Quality-related Information in the ISBSG Questionnaire

The ISBSG data collection questionnaire (ISBSG, 2013) was analyzed in order to identify the data fields that collect information directly related to software quality. The data quality fields among the data collected in the Project Process category and the Project Completion category are listed in Table 3.2. A number of data fields such as software size, number of defects are included in this list since they are useful for normalization purposes in order to calculate quality-related ratios, such as defect density.

Table 3.2 ISBSG data fields with information related to software quality

Category	**Phases**	**Collected Data**	**ISBSG Questionnaire**
Project process	Process Infrastructure	Type of software project	(Question: 5)
		The project consists of software that is reusable	(Question: 7)
		Process improvement program	(Question: 13)
	Planning	Rank project objectives	(Question: 15)
		Initial measure of the project's functional size made in project planning	(Question: 17)
		Estimate of total project effort made in project planning	(Question: 18)
		Estimated project completion date set in project planning	(Question: 19)
		Estimate of total project cost made in project planning	(Question: 20)
		Size of any preliminary functional model created during project planning	(Question: 21)
		Duration of project planning	(Question: 22)

Table 3.2 ISBSG data fields with information related to software quality (continued)

Category	**Phases**	**Collected Data**	**ISBSG Questionnaire**
	Specification	Size of any functional model created during the specification activity	(Question: 25)
		Number of defects recorded in the documents and other work products of this phase	(Question: 27)
		Functional size measured after the specification activity	(Question: 28)
		Duration of the specification activity	(Question: 29)
	Design	Number of defects recorded during the design phase	(Question: 32)
		Number of changes raised during design	(Question: 33)
		Functional size measured after completion of design	(Question: 34)
		Duration of the design activity	(Question: 35)
	Build or programming	Type of what produced or modified during the build activity	(Question: 36)
		Number of defects recorded and resolved during the build activity	(Question: 38)
		Number of changes raised during build	(Question: 39)
		Duration of the design activity	(Question: 40)
	Test	Number of defects recorded during the test activity	(Question: 43)
		Number of changes raised during testing	(Question: 44)
		Duration of the design activity	(Question: 45)
	Implementation	Number of distinct versions of the software delivered to the customer or end user during the projects	(Question: 47)
		Number of defects recorded during the implementation activity	(Question: 49)
		Number of changes raised during implementation	(Question: 50)

Table 3.2 ISBSG data fields with information related to software quality (continued)

Category	**Phases**	**Collected Data**	**ISBSG Questionnaire**
		Functional size measured after completion specification activity	(Question: 51)
		Duration of the implementation activity	(Question: 52)
Product	General information	Project made (or not) reuse of previous software development work	(Question: 93)
		Estimate amount of functionality provided by reused work products	(Question: 94)
Project completion	General information	Factors that have a negative impact on the project performance or outcomes	(Question: 129)
		Number of defects recorded during the first month of the software's operation	(Question: 130)
		The lines of code generated by this project The percentage of these lines of code that are not program statement	(Question: 131)
	User satisfaction survey	Did the project meet the stated objectives? Did the software meet business requirements? Quality expectation for the software? Quality expectation for user documentation? Ease of use requirements for the software? Was sufficient training or explanation given ? Schedule for planning and specification? Schedule for design, build, test, and implement?	(Question: 132)
	Project cost	Development team costs for each activity/total Customer/End-user costs for each activity/total IT operation costs for each activity/total	(Question: 135)

Form Table 3.2, it can be observed that:

- The 'Number of defects reported' is present in most of project phases (Q.27, Q.32, Q.38, Q.43, and Q.49) except the planning phase. For three ISBSG phases (e.g., build or programming, test, implementation or installation) and (Q.130) in the project completion category (e.g., the information collected for defects reported during the first month of the software operation by the users), the number of defects is classified into three defect levels (ISBSG, 2013):
  - Minor defect: "Does not make the software unusable in any way".
  - Major defect: "Causes part of the software to become unusable".
  - Extreme defect: "Failure causing the software to become totally unusable".
  - The defects data fields correspond to the quality section in the ISBSG MS-Excel data extract structure (see Table 3.3).

Table 3.3 Defect data fields in the ISBSG data extract (Cheikhi, Abran et Buglione, 2006)

Quality Fields	**Description**
Minor defects delivered	Number of minor defects reported
Major defects delivered	Number of major defects reported
Extreme defects delivered	Number of extreme defects reported
Total defects delivered	Number of total defects reported (minor, major and extreme)

- The 'Number of change requests made' is also collected for most of project phases (Q. 33, Q.39, Q.44, Q.50), that is from design to implementation or installation phases.
- The User Satisfaction Survey (Q.132) collects information about the satisfaction level as perceived by the end user, and the project cost collects information about Development team costs, Customer/End-user costs, and IT operation costs.

3.3 Analysis of the quality-related data fields in the ISBSG MS-Excel data extract (Release 12 of 2013)

This section presents the data extraction of the ISBSG MS-Excel to be used in the next research phases. As recommended by (Déry et Abran, 2005), and (Cheikhi, Abran et Buglione, 2007) two verification steps have to be carried out before using the data set for analysis: data quality verification and data completeness verification.

3.3.1 First level of data preparation

The first step of data quality verification is carried out by the ISBSG repository manager, who analyzes the data collected from the questionnaires and then rates the project data collected (Cheikhi, 2008). This rating information is recorded in a data field: the Data Quality Rating (DQR). The admissible values for this data field (Cheikhi, 2008) are:

- A: the data submitted was assessed as being sound with nothing being identified that might affect its integrity.
- B: the submission appears fundamentally sound but there are some factors which 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 submitted data.
- D: due to one factor or a combination of factors, little credibility should be given to the submitted data".

It is advisable for analysis purposes to consider only those projects having a DQR equal to A or B (e.g. the data collected have a high degree of integrity) (Cheikhi, Abran et Buglione, 2007). The number of projects, with their corresponding data quality rating, is presented in Table 3.4 for ISBSG Release 12. The 448 projects with a C or D quality rating will be dropped for our empirical analyses in the subsequent research phases: this leaves a sample of 5558 projects with an A or B data quality rating - see Table 3.4.

Table 3.4 Project Data Quality Classification(ISBSG, 2013)

Data Quality Rating	**No. of Projects**	**Percentage (%)**
A	1093	18.20
B	4465	74.34
C	255	4.25
D	193	3.21
Total	**6006**	**100**

3.3.2 Second level of data preparation

A second step is required in the data preparation. The quality-related data fields are not mandatory in the ISBSG repository: for instance, many software projects might have no data about defects.

We apply next a further data filtering and analysis to select only projects sized with the COSMIC sizing method and which have data in the field of 'Total number of defects': this leaves only 393 software projects with a data quality rating A and B.

Table 3.5 presents the number of projects with, or without, information about defects for a period of one month after of the software's operation, and categorized within (ISBSG, 2013) as: Minor Defects, Major Defects, and Extreme Defects, and Total Number of Defects.

The columns in Table 3.5 on the number of projects with defect severity type's information correspond to:

- Blank data fields: represents the number of projects without any information.
- Non-Blank data fields: represents the number of projects with defect numbers.
- Zero Defect data fields: represents the number of projects with zero defects reported.
- Max Defect data fields: represents the maximum number of defects registered in the MS-Excel data extract for a defect severity type.

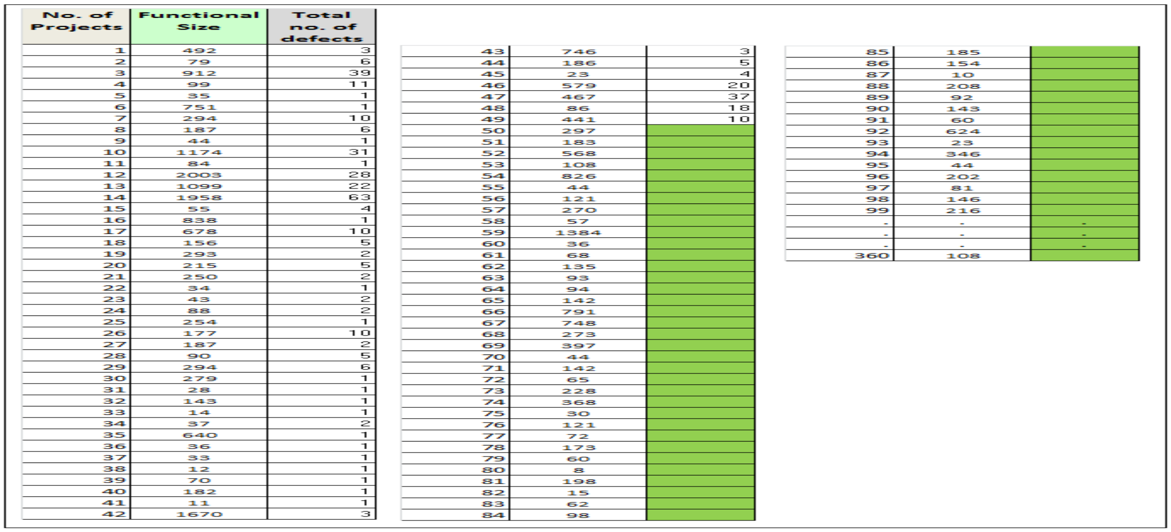
Table 3.5 Number of Projects (DQR = A and B) by Defect Severity type (ISBSG, 2013)

Quality	Blanks	Non-Blanks	Zero Defect	Max Defect	Total
Total Defects	311	49	33	63	393

In particular, from Table 3.5:

- Blank or no recorded ‘total number of defects’ = 311 software projects,
- With a ‘total number of defects’ = 49 software projects.
- A zero value in the total number of defect field (e.g. total defects = 0) = 33 software projects. This might be real information, but the zero value might also be caused by poor data entry, and some organizations might have entered a zero value instead of leaving the field blank for a missing value. To be on the safe side for this analysis, these 33 projects are dropped from further analysis. This leaves 360 projects available for further quality-related analysis.

Therefore, the data set of 360 software projects with complete and missing data with regards to the ‘total number of defects’ is selected for the empirical studies reported in this thesis - see Figure 3.1.



No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	2
28	90	5
29	294	6
30	279	1
31	28	1
32	143	1
33	14	1
34	37	2
35	640	1
36	36	1
37	33	1
38	12	1
39	70	1
40	182	1
41	11	1
42	1670	3
43	746	3
44	186	5
45	23	4
46	579	20
47	467	37
48	86	18
49	441	10
50	297	🟩
51	183	🟩
52	568	🟩
53	108	🟩
54	826	🟩
55	44	🟩
56	121	🟩
57	270	🟩
58	57	🟩
59	1384	🟩
60	36	🟩
61	68	🟩
62	135	🟩
63	93	🟩
64	94	🟩
65	142	🟩
66	791	🟩
67	748	🟩
68	273	🟩
69	397	🟩
70	44	🟩
71	142	🟩
72	65	🟩
73	228	🟩
74	368	🟩
75	30	🟩
76	121	🟩
77	72	🟩
78	173	🟩
79	60	🟩
80	8	🟩
81	198	🟩
82	15	🟩
83	62	🟩
84	98	🟩
85	185	🟩
86	154	🟩
87	10	🟩
88	208	🟩
89	92	🟩
90	143	🟩
91	60	🟩
92	624	🟩
93	23	🟩
94	346	🟩
95	44	🟩
96	202	🟩
97	81	🟩
98	146	🟩
99	216	🟩
.	.	-
.	.	-
.	.	-
360	108	🟩

Figure 3.1 A sample of software projects with missing data points within Total Number of Defects N=360 projects

Figure 3.2 shows the distribution of the software sizes of the data set of N=360 software projects sized by COSMIC method, with a software size ranging from 2 to 2090 CFP (COSMIC Function Points), with most values at the low end. The median is 133 CFP.

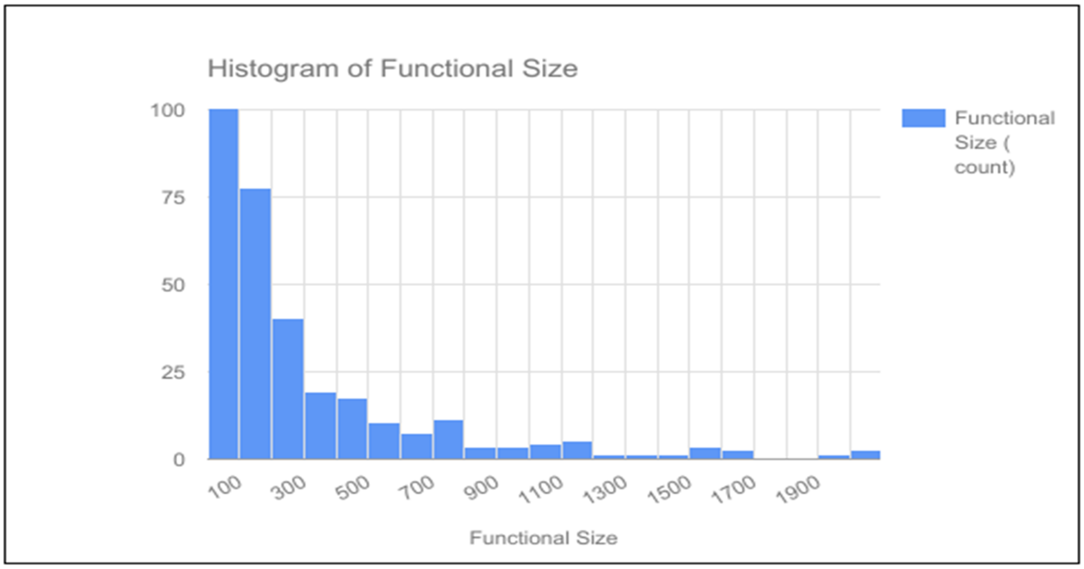


Figure 3.2 Distribution of the COSMIC functional size of data set N=360 projects

The analysis of the MS-Excel data extract indicates that only (9%) of the projects sized with COSMIC method in ISBSG R12 contains information about the quality of the software delivered by these projects.

In summary, the ISBSG MS-Excel data extract (Release 12, year 2013) has been handled using two levels of data preparations. The variables ‘Total Number of Defects delivered’ and ‘Functional Size’ measured with COSMIC will be used for further research analysis. The extracted data set consists of a sample of N=360 software projects, where it has high percentage of missing data within the variable of ‘total defects’ (more than 50%): this represents a serious challenge of the best selection of imputation techniques whereas the statistical analysis to conducted based on the obtained estimation models should be statistically significant. Therefore, it is suggested to conduct a comparison between the selected imputation techniques: single imputation, regression imputation, and stochastic imputation. Thus, this requires identifying an imputation strategy that reflects such analysis perspectives.

3.4 Mapping the of ISBSG Questionnaire to Six Sigma methodologies (DMAIC and DFSS)

This section presents the detailed mappings between the six sigma methodologies of DMAIC and DFSS (IDDOV) with the ISBSG questionnaire data. The mapping of ISBSG questionnaire sections to Six Sigma for software is presented in Tables 3.6 and 3.7: it provides a detailed Six Sigma mapping with the ISBSG data collection questionnaire.

Table 3.6 Mapping ISBSG questionnaire sections to Six Sigma

Category	**Sub-sections**	**Six Sigma DMAIC**	**DFSS IDDOV**
Project process	Process infrastructure	X	X
	Planning	X	X
	Specification		X
	Design		X
	Build or Programming	X	X
	Test	X	X
	Implementation/ installation	X	X
	Project management and monitoring		
Technology	General Information		
People and Work Effort	Development Team		
	Customers and End Users		
	IT Operations		
	Work Effort validation		
Product	General Information		

Table 3.6 Mapping ISBSG questionnaire sections to Six Sigma (continued)

Category	**Sub-sections**	**Six Sigma DMAIC**	**DFSS IDDOV**
COSMIC Project Functional Size	New development or redevelopment software size		X
	Enhancement software size	X	
	Context of the functional size measurement		
	Experience of the functional counter		
Project Completion	General information	X	
	User satisfaction survey		
	Project costs		
	Cost Validation		

Table 3.7 Detailed Six Sigma views in in the ISBSG data collection questionnaire

Category	**Phases**	**Collected Data**	**ISBSG Questionnaire**	**Six Sigma DMAIC**	**DFSS IDDOV**
Project process	Process Infrastructure	Type of software project	(Question: 5)	X	X
		The project consists of software that is reusable	(Question: 7)		
		Process improvement program	(Question: 13)	X	X
	Planning	Rank project objectives	(Question: 15)	X	X
		Initial measure of the project's functional size made in project planning	(Question: 17)		X
		Estimate of total project effort made in project planning	(Question: 18)		

Table 3.7 Detailed Six Sigma views in in the ISBSG data collection questionnaire (continued)

Category	Phases	Collected Data	ISBSG Questionnaire	Six Sigma DMAIC	DFSS IDDOV
		Estimated project completion date set in project planning	(Question: 19)		
		Estimate of total project cost made in project planning	(Question: 20)		
		Size of any preliminary functional model created during project planning	(Question: 21)		X
		Duration of project planning	(Question: 22)		
	Specification	Size of any functional model created during the specification activity	(Question: 25)		X
		Number of defects recorded in the documents and other work products of this phase	(Question: 27)		X
		Functional size measured after the specification activity	(Question: 28)		X
		Duration of the specification activity	(Question: 29)		
	Design	Number of defects recorded during the design phase	(Question: 32)		X
		Number of changes raised during design	(Question: 33)		X
		Functional size measured after completion of design	(Question: 34)		X
		Duration of the design activity	(Question: 35)		

Table 3.7 Detailed Six Sigma views in in the ISBSG data collection questionnaire (continued)

Category	Phases	Collected Data	ISBSG Questionnaire	Six Sigma DMAIC	DFSS IDDOV
	Build or programming	Type of what produced or modified during the build activity	(Question: 36)		
		Number of defects recorded and resolved during the build activity	(Question: 38)	X	X
		Number of changes raised during build	(Question: 39)	X	X
		Duration of the design activity	(Question: 40)		
	Test	Number of defects recorded during the test activity	(Question: 43)	X	X
		Number of changes raised during testing	(Question: 44)	X	X
		Duration of the design activity	(Question: 45)		
	Implementation	Number of distinct versions of the software delivered to the customer or end user during the projects	(Question: 47)		
		Number of defects recorded during the implementation activity	(Question: 49)		X
		Number of changes raised during implementation	(Question: 50)		X
		Functional size measured after completion specification activity	(Question: 51)		X
		Duration of the implementation activity	(Question: 52)		

Table 3.7 Detailed Six Sigma views in in the ISBSG data collection questionnaire (continued)

Category	**Phases**	**Collected Data**	**ISBSG Questionnaire**	**Six Sigma DMAIC**	**DFSS IDDOV**
Product	General information	Project made (or not) reuse of previous software development work	(Question: 93)		
		Estimate amount of functionality provided by reused work products	(Question: 94)		
COSMIC Project Functional Size	New Development or Re-development Software Size	COSMIC functional sizing standard	(Question: 95)		
		Approach used to determine the project functional size	(Question: 96)		
		Measurement view point of the count	(Question: 97)		
		Major components of an application or of infrastructure software	(Question: 98)		X
		Size software	(Question: 99)		X
	Enhancement Software Size	COSMIC functional sizing standard	(Question: 101)		
		Approach used to determine the project functional size	(Question: 102)		
		Measurement view point of the count	(Question: 103)		
		Functional size of the software before the enhancement project	(Question: 104)	X	
		Major components of an application or of infrastructure software	(Question: 105)	X	
		Added functionality-size software	(Question: 106)	X	

Table 3.7 Detailed Six Sigma views in in the ISBSG data collection questionnaire (continued)

Category	Phases	Collected Data	ISBSG Questionnaire	Six Sigma DMAIC	DFSS IDDOV
		Changed functionality-size software	(Question: 107)	X	
		Deleted functionality-software	(Question: 108)	X	
		Software size in COSMIC function points	(Question: 109)	X	
Project completion	General information	Total duration of the project	(Question: 126)		
		Total inactivity time on the project	(Question: 127)		
		Factors that have a positive impact on the project performance or outcomes	(Question: 128)	X	X
		Factors that have a negative impact on the project performance or outcomes	(Question: 129)	X	X
		Number of defects recorded during the first month of the software's operation	(Question: 130)	X	
		The lines of code generated by this project The percentage of these lines of code that are not program statement	(Question: 131)		

From Tables 3.6, and 3.7, it can be observed that:

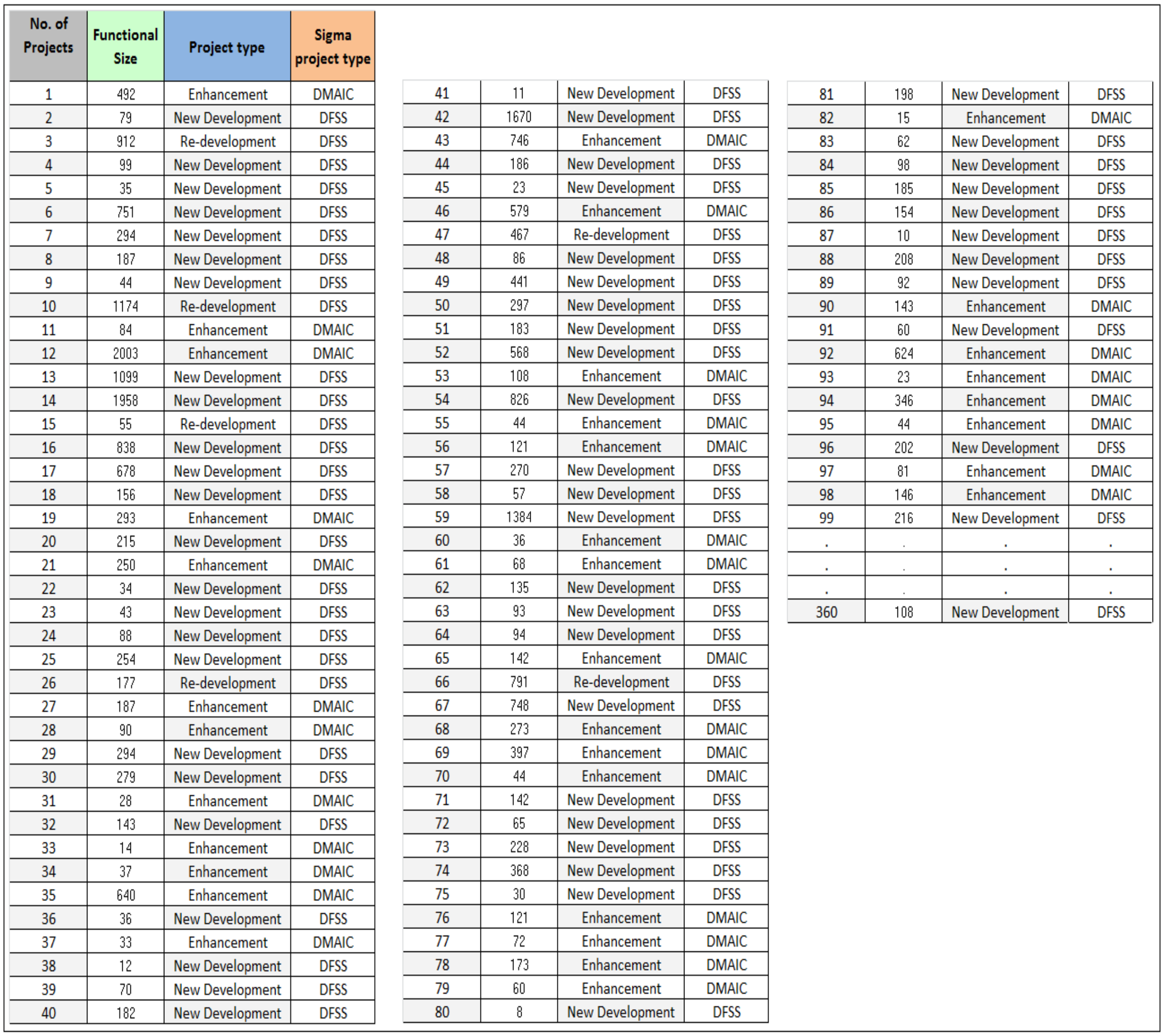
- The DMAIC for process improvement comes after the design stage of software development process, which focuses on enhancing the existed processes, whereas, the DFSS-IDDOV methodology comes before the design stage, which allows for re-designing processes before the implementation phase of projects process.

- The DMAIC approach aligns with the software enhancement' sub-section within the COSMIC Project Functional Size category.
- The DFSS-IDDOV approach aligns with the software new development and re-development' sub-section within the COSMIC Project Functional Size category.
- In contrast, questions (104, 105, 106, 107, 108, and 109) in Table 3.7 obtain information on functional size when to improve the existing processes (through adding, changing, or deleting functionalities).
- Questions (98 and 99) collect the software functional size when to re-design existing process or designing new of processes.

In summary, the ISBSG data fields with information related to software quality have been identified which gives that 39 questions are related to software quality within the COSMIC sizing method questionnaire for data release 12 of year 2013. The detailed mappings between the six sigma methodologies of DMAIC and DFSS (IDDOV) and the ISBSG data questionnaire have been conducted, which highlights that DMAIC comes after the design stage at the process life cycle, whereas, DFSS comes early; it also shows that DMAIC aligns with software enhancement of software project' type, and DFSS aligns with software new development and re-development of software project' type.

3.5 Analysis of software projects of ISBSG dataset N=360 projects

Based on sections 3.3 and 3.4, Figure 3.3 presents an example of sample results for software projects of ISBSG data set N=360 with regards to software projects' development type and Sigma projects' type with their COSMIC functional size.



No. of Projects	Functional Size	Project type	Sigma project type
1	492	Enhancement	DMAIC
2	79	New Development	DFSS
3	912	Re-development	DFSS
4	99	New Development	DFSS
5	35	New Development	DFSS
6	751	New Development	DFSS
7	294	New Development	DFSS
8	187	New Development	DFSS
9	44	New Development	DFSS
10	1174	Re-development	DFSS
11	84	Enhancement	DMAIC
12	2003	Enhancement	DMAIC
13	1099	New Development	DFSS
14	1958	New Development	DFSS
15	55	Re-development	DFSS
16	838	New Development	DFSS
17	678	New Development	DFSS
18	156	New Development	DFSS
19	293	Enhancement	DMAIC
20	215	New Development	DFSS
21	250	Enhancement	DMAIC
22	34	New Development	DFSS
23	43	New Development	DFSS
24	88	New Development	DFSS
25	254	New Development	DFSS
26	177	Re-development	DFSS
27	187	Enhancement	DMAIC
28	90	Enhancement	DMAIC
29	294	New Development	DFSS
30	279	New Development	DFSS
31	28	Enhancement	DMAIC
32	143	New Development	DFSS
33	14	Enhancement	DMAIC
34	37	Enhancement	DMAIC
35	640	Enhancement	DMAIC
36	36	New Development	DFSS
37	33	Enhancement	DMAIC
38	12	New Development	DFSS
39	70	New Development	DFSS
40	182	New Development	DFSS
41	11	New Development	DFSS
42	1670	New Development	DFSS
43	746	Enhancement	DMAIC
44	186	New Development	DFSS
45	23	New Development	DFSS
46	579	Enhancement	DMAIC
47	467	Re-development	DFSS
48	86	New Development	DFSS
49	441	New Development	DFSS
50	297	New Development	DFSS
51	183	New Development	DFSS
52	568	New Development	DFSS
53	108	Enhancement	DMAIC
54	826	New Development	DFSS
55	44	Enhancement	DMAIC
56	121	Enhancement	DMAIC
57	270	New Development	DFSS
58	57	New Development	DFSS
59	1384	New Development	DFSS
60	36	Enhancement	DMAIC
61	68	Enhancement	DMAIC
62	135	New Development	DFSS
63	93	New Development	DFSS
64	94	New Development	DFSS
65	142	Enhancement	DMAIC
66	791	Re-development	DFSS
67	748	New Development	DFSS
68	273	Enhancement	DMAIC
69	397	Enhancement	DMAIC
70	44	Enhancement	DMAIC
71	142	New Development	DFSS
72	65	New Development	DFSS
73	228	New Development	DFSS
74	368	New Development	DFSS
75	30	New Development	DFSS
76	121	Enhancement	DMAIC
77	72	Enhancement	DMAIC
78	173	Enhancement	DMAIC
79	60	Enhancement	DMAIC
80	8	New Development	DFSS
81	198	New Development	DFSS
82	15	Enhancement	DMAIC
83	62	New Development	DFSS
84	98	New Development	DFSS
85	185	New Development	DFSS
86	154	New Development	DFSS
87	10	New Development	DFSS
88	208	New Development	DFSS
89	92	New Development	DFSS
90	143	Enhancement	DMAIC
91	60	New Development	DFSS
92	624	Enhancement	DMAIC
93	23	Enhancement	DMAIC
94	346	Enhancement	DMAIC
95	44	Enhancement	DMAIC
96	202	New Development	DFSS
97	81	Enhancement	DMAIC
98	146	Enhancement	DMAIC
99	216	New Development	DFSS
.	.	.	.
.	.	.	.
.	.	.	.
360	108	New Development	DFSS

Figure 3.3 An example of sample results for software projects of ISBSG data set N=360 with regards to software projects' development type, sigma projects' type

3.5.1 Software projects' development type analysis results

Figures 3.4 and 3.5, presents next the number of software projects by type and their percentage, where:

- Enhancement projects = 149 projects, which represents 41% of projects number,
- Re-development projects = 11 projects, which represents 3% of projects number, and
- New software development projects = 200 projects, which represents the highest percentage of 56% of projects.

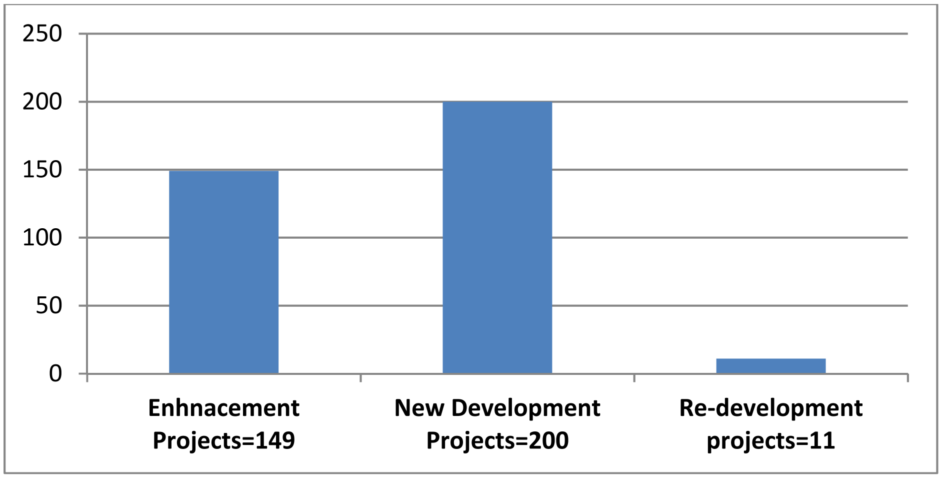


Figure 3.4 Number of software projects by type N=360 projects

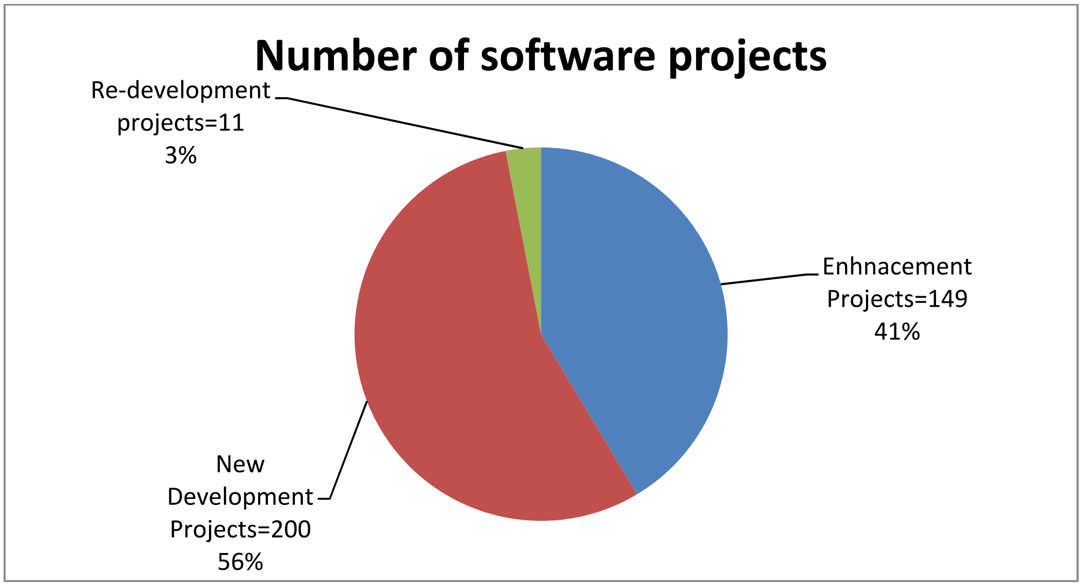


Figure 3.5 Number of software projects by type and their percentage N=360 projects

From the software projects' type distribution, it can be noted that software organizations have submitted more data on development of new processes or products (200 projects) rather than on the re-design of existing ones (11 projects). Therefore, this indicates that DFSS projects could be used for creating new processes or products (in order to prevent defects at early stages of software life cycle) more than seeking to re-design of existing ones.

3.5.2 Six Sigma projects' type analysis

Figures 3.6 and 3.7 present the number of Sigma projects by type and their percentage, where the number of the DMAIC projects is 149 projects, which represents 41.4% of projects number, and the number of DFSS projects is 211 projects, which represents the highest percentage of 58.6%.

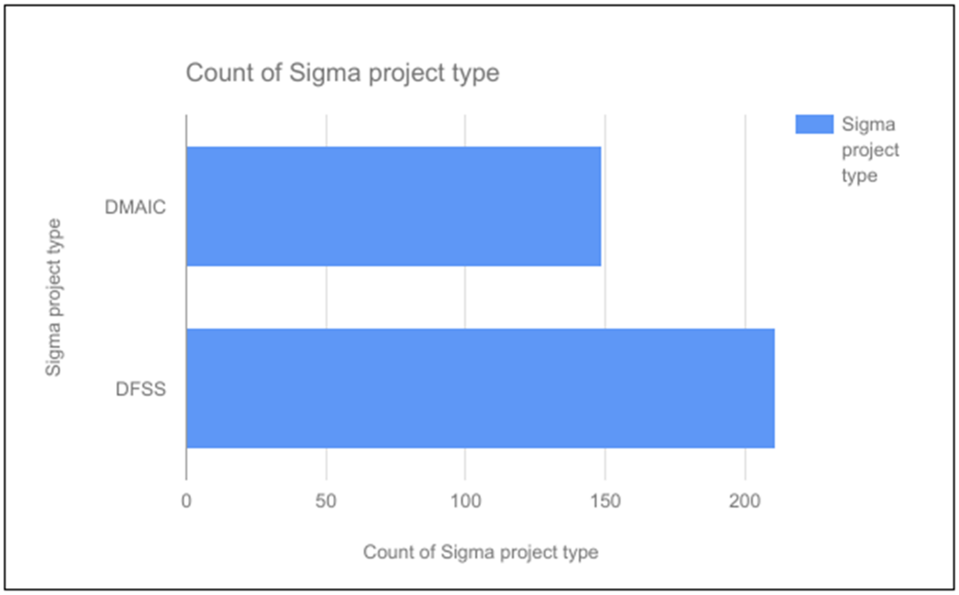


Figure 3.6 Number of Sigma projects by type N=360 projects

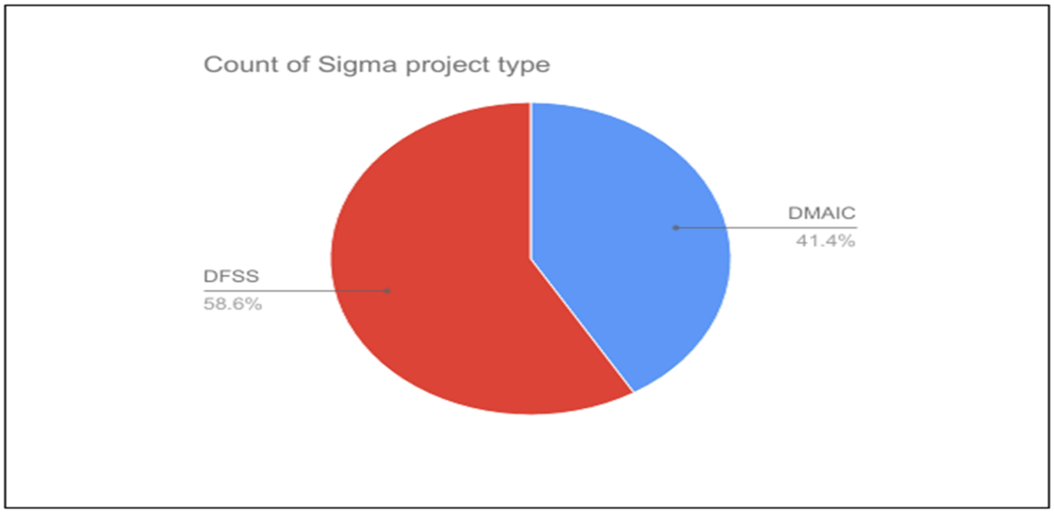


Figure 3.7 Number of Sigma projects by type and their percentage N=360 projects

Figure 3.8 shows the software sizes of DMAIC projects, with a range from 2 to 2003 CFP, with most values at the low end. The median size is 95 CFP.

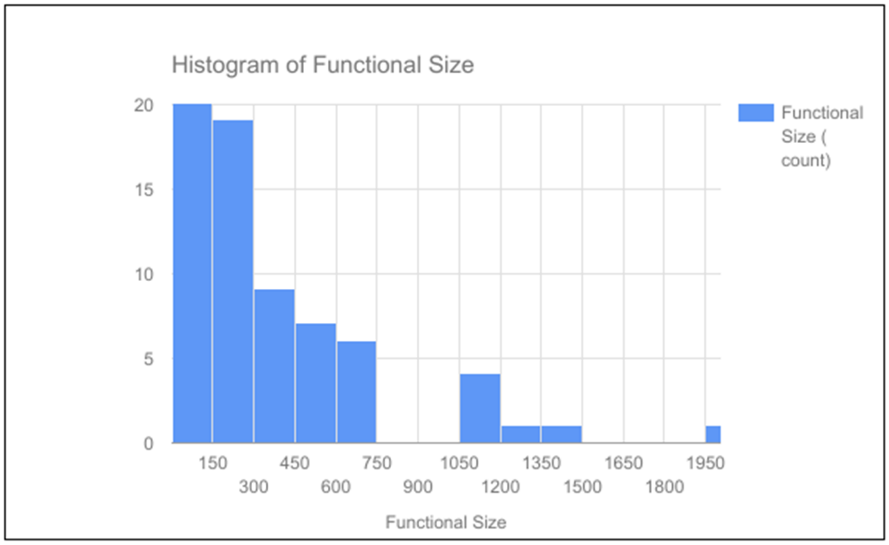


Figure 3.8 CFP software sizes of DMAIC projects N=149 projects

Figure 3.9 shows the software sizes of DFSS projects, with a range from 8 to 2090 CFP, with most values at the low end. The median size is 175 CFP.

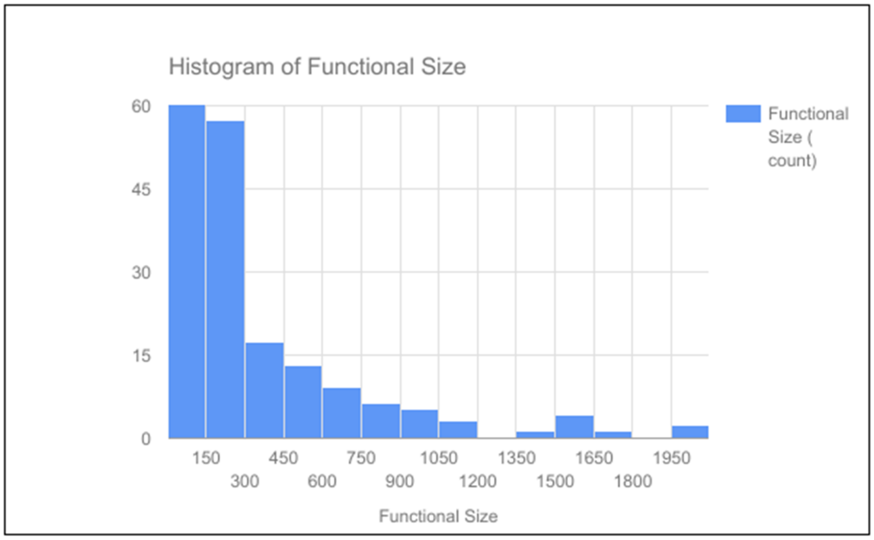


Figure 3.9 CFP software sizes of DFSS projects N=211 projects

3.6 Imputation and Defect estimation activities

Figure 3.10 shows the steps of implementing the imputation techniques and building defect estimation models.

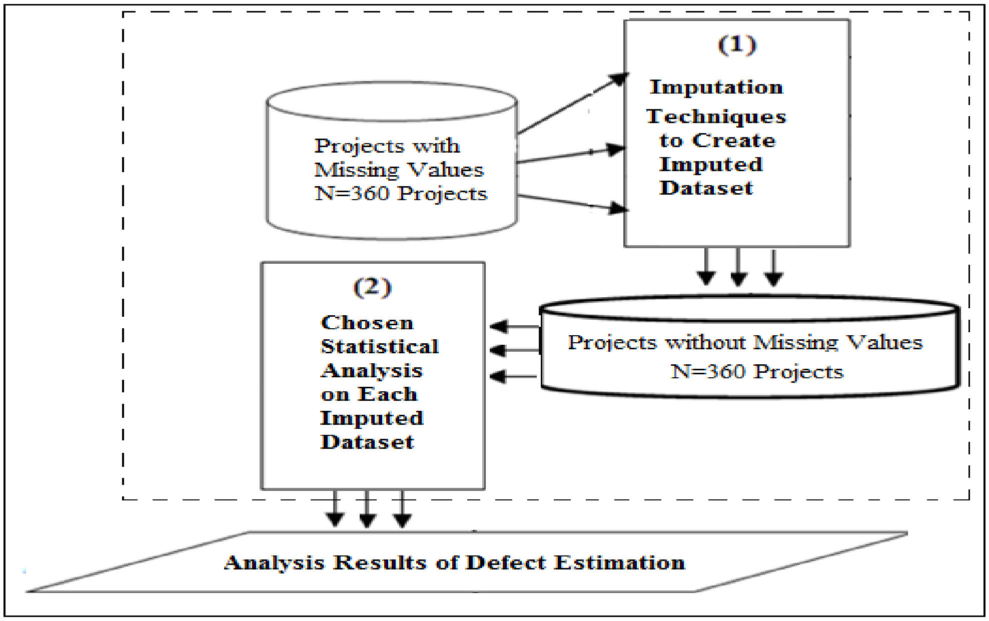

(1)
Imputation Techniques to Create Imputed Dataset
Projects with Missing Values N=360 Projects
(2)
Chosen Statistical Analysis on Each Imputed Dataset
Projects without Missing Values N=360 Projects
Analysis Results of Defect Estimation

Figure 3.10 Imputation processing and defect estimation modeling strategy

1. Create imputed data sets

The first step is to create the ‘imputes’ to substitute the missing data. The imputation procedure needs to be identified to allow the ‘imputes’ to be created based on the values found across the data set for the available values of same variable in the dataset (Bala, 2013). This involves the creation of the imputed dataset by using the three selected imputation techniques (Single imputation, regression imputation, and stochastic regression imputation) in order to generate a complete dataset, which are adequate representations of the data.

2. Defect estimation modeling

A statistical analysis is conducted on the imputed dataset. Such statistical analysis is to be achieved in order to analyze the imputed data set after accomplishing the imputation procedure - in step (1), that is to produce a complete imputed data set with no missing data within the dependent variable ‘Total Number of Defects’ based on an independent variable ‘Functional Size’ in Function Points (FP).

The modeling through a linear regression of the relationship of dependent variable ‘Total Number of Defects’ based on an independent variable ‘Functional Size’ (in Function Points) is applied on the imputed dataset to obtain the TD estimates and standard errors (build TD estimation models).

The statistical analysis includes - see Figure 3.11:

- Estimate TD (dependent variable) based on Functional Size (independent variable).
- Analysis with R^2 and P-value of the estimation results of TD using FP as the dependent variable.
- Outliers’ detection: using Grubbs test to investigate whether the outliers affects the rest of data points on TD after filling out its missing data by the three selected imputation techniques.
- Observe the values of Defect Density (DD) for each software project within dataset N=360 projects based on the formula of the Defect Density which measures the quality of software in terms of defects delivered in unit size of software. It is expressed as Defects per Function Points (TD⁄CFP).

The following criteria for analyzing the results of TD estimation models:

- Coefficient of determination (R^2): the coefficient has a value between 0 and 1. R^2, close to 1;
- Standard Errors (STD-E): low Standard Errors;
- Mean Magnitude Relative Error (MMRE): low values of Mean Magnitude Relative Error.
- P-value: Statistical Significance (P-value < 0.05).
- T-test: Statistical Significance (t-test > 2).
- Predictive quality of the TD estimation model: Pred(0.25) = 0.75.

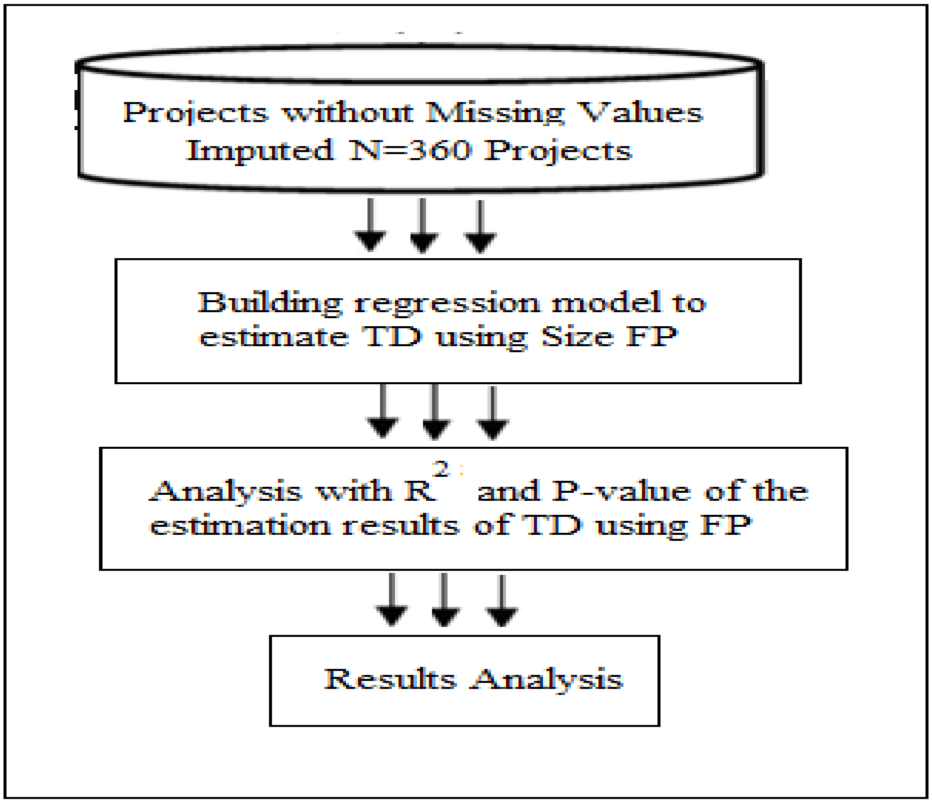

Projects without Missing Values
Imputed N=360 Projects
Building regression model to estimate TD using Size FP
Analysis with R2 and P-value of the estimation results of TD using FP
Results Analysis

Figure 3.11 Building the regression analysis for TD estimation models

CHAPTER 4

SINGLE IMPUTATION (SI)

This chapter on Phase 2 of the research methodology describes the single imputation technique, and presents how single imputation is implemented to impute the missing data fields of the 'Total Number of Defects' in order to produce a complete data set N = 360 of software projects. It presents next the linear regression modeling of the relationship of the dependent variable 'Total Number of Defects' based on the independent variable 'Functional Size' (after imputing the TD missing data using the single imputation technique).

4.1 Introduction

Single imputation consists in replacing the missing data with imputed values obtained from randomly distributed values (Bala, 2013). This process may result in valid statistical inferences that are adequately reflecting the uncertainty resulting from the missing data (Bala, 2013).

In single imputation, the predicted values, called 'imputes', are replacing the missing values resulting in a complete data set that is called an imputed data set (Bala, 2013).

Single imputation process can restore the natural variability in the missing data, and also can incorporate the uncertainty that it caused by estimating missing data (Bala, 2013).

However, single imputation does not attempt to estimate each missing value through simulated values but rather to represent a random sample of the missing values (Bala, 2013).

Single Imputation: the missing data are filled out by random selection of absolute values from min-max seeds in order to generate a complete data set of size of N = 360 software projects. The output from this process should be a completed data set of ISBSG data repository R12, with a solution of the missing data problem for the variable 'Total number of

Defects' (the adoption of Single imputation technique being considered as an attempt to resolve such issue).

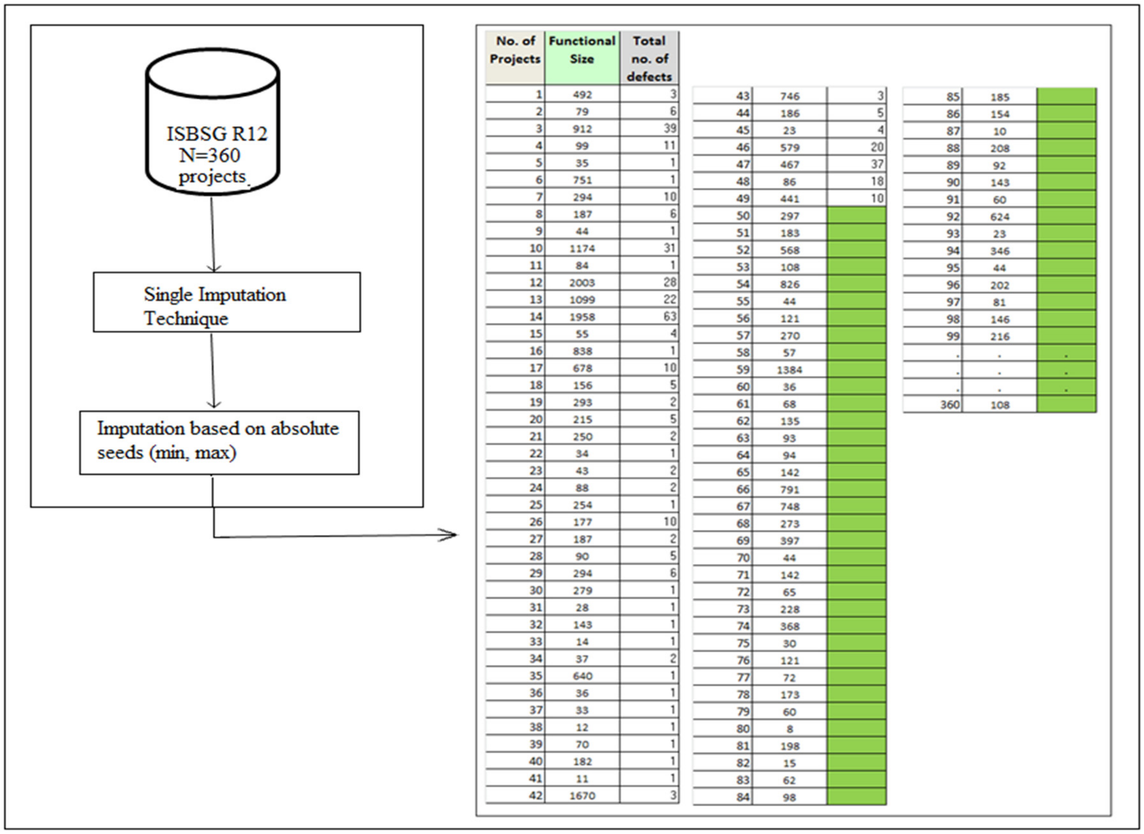
As shown in (Bala, 2013), this approach confirmed that for the data set used, the estimation model (of dependent variable and independent variable) built from imputed data based on the absolute min-max seeds obtained a statistically significant predictive accuracy.

4.2 Implement the Imputation technique for Total Defects field with missing values

This section applies Imputation processing and defect estimation modeling as structured in Figure 3.10 in the previous chapter. It presents an application of the steps of the single imputation process and a statistical inferences analysis on the dataset N = 360 of software projects (based on data quality filtering done earlier in chapter 3 of Data Preparation).

This section is structured as follows:

1. Creating the imputed data set (the imputation process). In this step (see Figure 4.1): the 'Total Number of Defects' data fields with missing values from the ISBSG R12, N=360 projects are imputed based on the absolute min-max seeds approach: random numbers are generated to fill out the missing TD values. The seed values selected for the full sample of 360 projects are set to the minimum and maximum values from the 'Total Number of Defects' data fields that do not have missing values within the dataset: here, the minimum is 1 defect and the maximum is 63 defects.



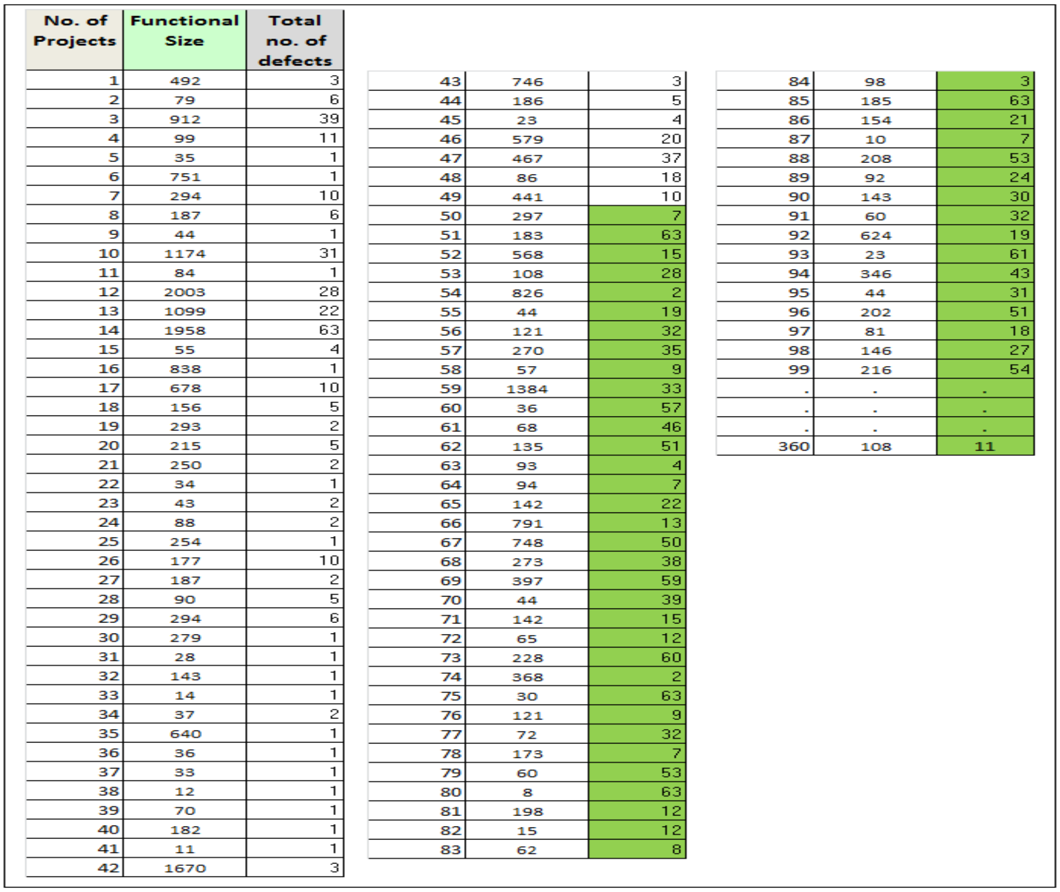
No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	2
28	90	5
29	294	6
30	279	1
31	28	1
32	143	1
33	14	1
34	37	2
35	640	1
36	36	1
37	33	1
38	12	1
39	70	1
40	182	1
41	11	1
42	1670	3
43	746	3
44	186	5
45	23	4
46	579	20
47	467	37
48	86	18
49	441	10
50	297	🟩
51	183	🟩
52	568	🟩
53	108	🟩
54	826	🟩
55	44	🟩
56	121	🟩
57	270	🟩
58	57	🟩
59	1384	🟩
60	36	🟩
61	68	🟩
62	135	🟩
63	93	🟩
64	94	🟩
65	142	🟩
66	791	🟩
67	748	🟩
68	273	🟩
69	397	🟩
70	44	🟩
71	142	🟩
72	65	🟩
73	228	🟩
74	368	🟩
75	30	🟩
76	121	🟩
77	72	🟩
78	173	🟩
79	60	🟩
80	8	🟩
81	198	🟩
82	15	🟩
83	62	🟩
84	98	🟩
85	185	🟩
86	154	🟩
87	10	🟩
88	208	🟩
89	92	🟩
90	143	🟩
91	60	🟩
92	624	🟩
93	23	🟩
94	346	🟩
95	44	🟩
96	202	🟩
97	81	🟩
98	146	🟩
99	216	🟩
.	.	.
.	.	.
.	.	.
360	108	🟩

Figure 4.1 Example of the dataset N=360 of software projects with missing data to be imputed by SI

Figure 4.1, shows an example of samples of the 360 software projects with missing data points in the field 'Total Number of Defects', where:

- 49 projects have data for 'Total Number of Defects' (projects 1 to 49) and
- 311 projects have missing data (projects 50 and over).
- 311 projects with missing TD are to be imputed based on single imputation by random selection from min-max seeds from absolute values (based on the available data within the field with missing values).

Figure 4.2 shows next an example of the sample of the 360 software projects with the imputed values on the data points in the field Total Number of Defects for the projects numbered from 50 to 360.



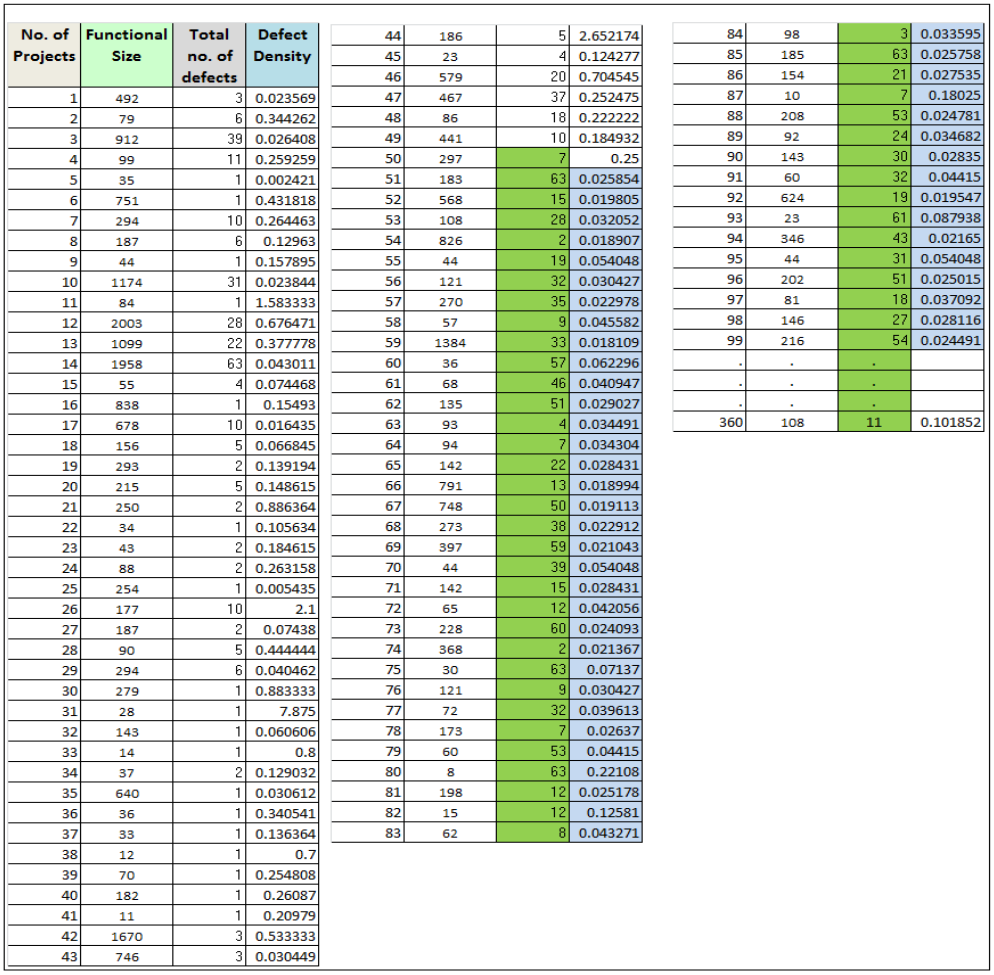
No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	2
28	90	5
29	294	6
30	279	1
31	28	1
32	143	1
33	14	1
34	37	2
35	640	1
36	36	1
37	33	1
38	12	1
39	70	1
40	182	1
41	11	1
42	1670	3
43	746	3
44	186	5
45	23	4
46	579	20
47	467	37
48	86	18
49	441	10
50	297	7
51	183	63
52	568	15
53	108	28
54	826	2
55	44	19
56	121	32
57	270	35
58	57	9
59	1384	33
60	36	57
61	68	46
62	135	51
63	93	4
64	94	7
65	142	22
66	791	13
67	748	50
68	273	38
69	397	59
70	44	39
71	142	15
72	65	12
73	228	60
74	368	2
75	30	63
76	121	9
77	72	32
78	173	7
79	60	53
80	8	63
81	198	12
82	15	12
83	62	8
84	98	3
85	185	63
86	154	21
87	10	7
88	208	53
89	92	24
90	143	30
91	60	32
92	624	19
93	23	61
94	346	43
95	44	31
96	202	51
97	81	18
98	146	27
99	216	54
.	.	.
.	.	.
.	.	.
360	108	11

Figure 4.2 Sample results of software projects with TD imputed using SI technique

Next, the values of the defect density are observed for the imputed data N=360 projects (after imputing the TD missing data by SI) that have blank fields of defect density caused by missing TD values (the software size values are all recorded within the data set N=360) - see Figure 4.3.

- Calculate the Defect Density (DD) after the imputation for the total defects, based on the formula:

$$\text{Defect Density} = \frac{\textit{Total Number of Defects}}{\textit{Functional Size}} \text{ , (ISBSG, 2013)}$$



No. of Projects	Functional Size	Total no. of defects	Defect Density
1	492	3	0.023569
2	79	6	0.344262
3	912	39	0.026408
4	99	11	0.259259
5	35	1	0.002421
6	751	1	0.431818
7	294	10	0.264463
8	187	6	0.12963
9	44	1	0.157895
10	1174	31	0.023844
11	84	1	1.583333
12	2003	28	0.676471
13	1099	22	0.377778
14	1958	63	0.043011
15	55	4	0.074468
16	838	1	0.15493
17	678	10	0.016435
18	156	5	0.066845
19	293	2	0.139194
20	215	5	0.148615
21	250	2	0.886364
22	34	1	0.105634
23	43	2	0.184615
24	88	2	0.263158
25	254	1	0.005435
26	177	10	2.1
27	187	2	0.07438
28	90	5	0.444444
29	294	6	0.040462
30	279	1	0.883333
31	28	1	7.875
32	143	1	0.060606
33	14	1	0.8
34	37	2	0.129032
35	640	1	0.030612
36	36	1	0.340541
37	33	1	0.136364
38	12	1	0.7
39	70	1	0.254808
40	182	1	0.26087
41	11	1	0.20979
42	1670	3	0.533333
43	746	3	0.030449
44	186	5	2.652174
45	23	4	0.124277
46	579	20	0.704545
47	467	37	0.252475
48	86	18	0.222222
49	441	10	0.184932
50	297	7	0.25
51	183	63	0.025854
52	568	15	0.019805
53	108	28	0.032052
54	826	2	0.018907
55	44	19	0.054048
56	121	32	0.030427
57	270	35	0.022978
58	57	9	0.045582
59	1384	33	0.018109
60	36	57	0.062296
61	68	46	0.040947
62	135	51	0.029027
63	93	4	0.034491
64	94	7	0.034304
65	142	22	0.028431
66	791	13	0.018994
67	748	50	0.019113
68	273	38	0.022912
69	397	59	0.021043
70	44	39	0.054048
71	142	15	0.028431
72	65	12	0.042056
73	228	60	0.024093
74	368	2	0.021367
75	30	63	0.07137
76	121	9	0.030427
77	72	32	0.039613
78	173	7	0.02637
79	60	53	0.04415
80	8	63	0.22108
81	198	12	0.025178
82	15	12	0.12581
83	62	8	0.043271
84	98	3	0.033595
85	185	63	0.025758
86	154	21	0.027535
87	10	7	0.18025
88	208	53	0.024781
89	92	24	0.034682
90	143	30	0.02835
91	60	32	0.04415
92	624	19	0.019547
93	23	61	0.087938
94	346	43	0.02165
95	44	31	0.054048
96	202	51	0.025015
97	81	18	0.037092
98	146	27	0.028116
99	216	54	0.024491
.	.	.	
.	.	.	
.	.	.	
360	108	11	0.101852

Figure 4.3 Sample results of observed DD for imputed TD data points by SI

Figure 4.4 shows next graphically the defect density of imputed SI data set of N=360 software projects sized by COSMIC method, with ranges from 0.0012 TD⁄CFP to 31.5 TD⁄CFP, while most values are at the low end. The median is 0.162 TD⁄CFP. The defect density largest values are 11.67 TD⁄CFP, 16 TD⁄CFP, 27.5 TD⁄CFP, and 31.5 TD⁄CFP.

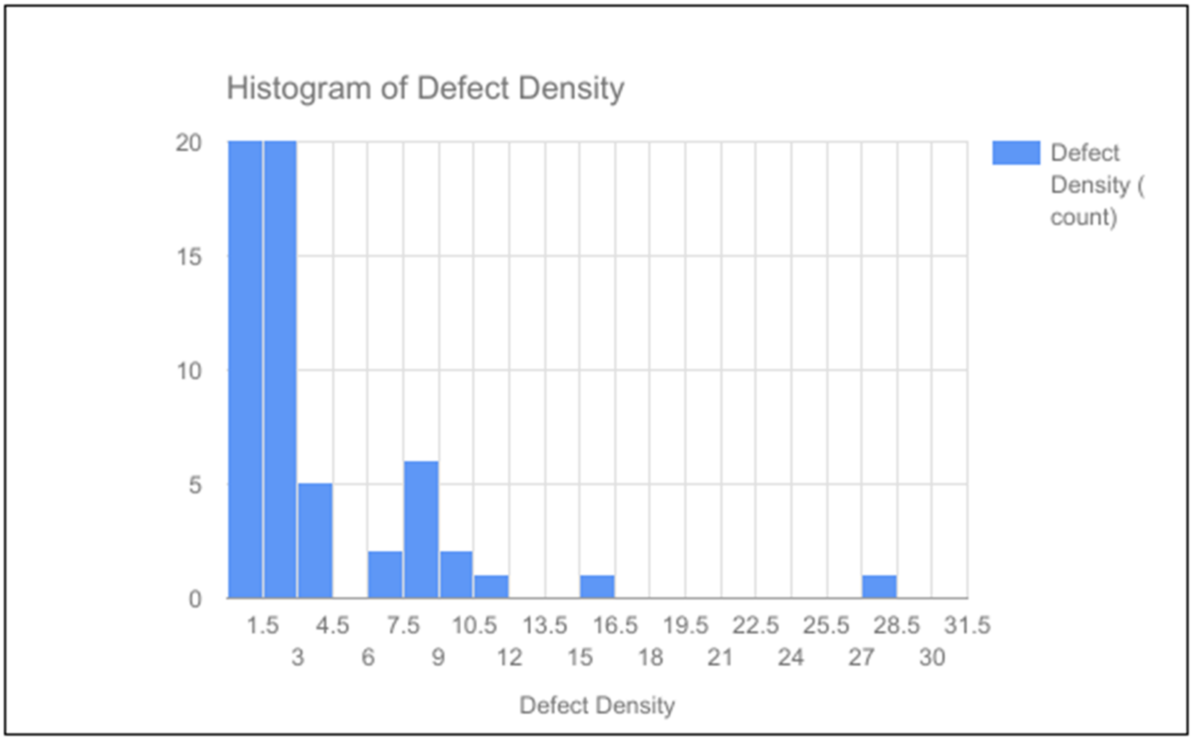


Figure 4.4 Defect density of imputed dataset by SI, N=360 projects

The next procedure to perform is the statistical analysis on the imputed dataset N=360 projects through building linear regression models: that is, to estimate the dependent variable ‘Total Number of Defects’ on the basis of the independent variables ‘Functional size’ as structured in step (2) in Figure 3.11 earlier.

In order to investigate the impact analysis of the imputation using SI, a statistical regression analysis is applied on variables ‘Total number of Defects’ and ‘Software Functional size’. If the regression results show a large number of outliers for dependent variable TD and independent variable ‘Functional Size’, then a multiple imputation technique is required to be applied in order to reduce the number of the outliers between them, and then the same regression analysis is to be applied after the imputation (see Appendix II):

- A linear regression analysis of ‘Functional size’ (independent variable) and ‘Total Number of Defects’ (dependent variable) - see Table 4.1, and also a graphical representation of the relationship of the TD based on Size in CFP - see Figure 4.5.

- To verify whether or not these data points (of TD after imputation) are truly statistical outliers, the Grubbs test is applied on the ISBSG R12 data set of N=360 projects in order to investigate the outliers on TD and on Size - see Table 4.2 and 4.3.

Table 4.1 displays a 95% mean confidence interval and a t-test with the associated P-value and verifying the impact of the independent 'Functional Size' on TD parameter estimates: the inferences are based on the t-distribution.

Table 4.1 Regression parameters and statistical tests for TD estimation model using the SI-imputed dataset, N=360 projects

Variable	Intercept	Coefficient	R2	95% Confidence Limits		T- test	Standard Error	P-value
Functional Size	26.89	0.0033	0.062	-0.0025	0.0091	1.1243	0.00293	0.26

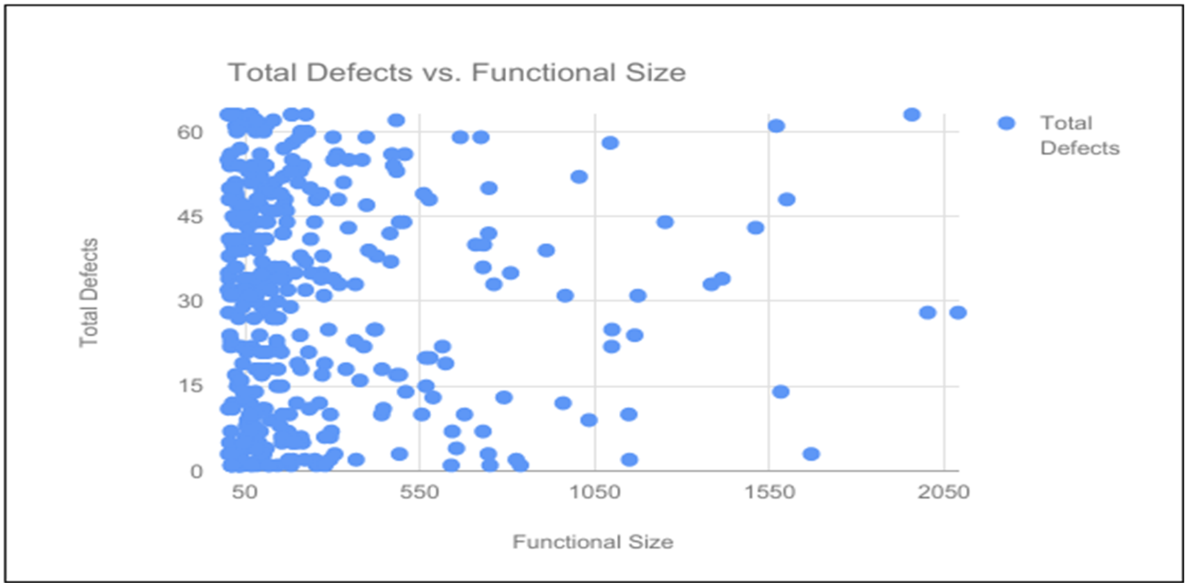


Figure 4.5 Graphical representation of functional size and total defects for imputed-SI dataset N=360 projects-with outliers

Table 4.1 presents the results of a linear regression analysis of TD estimation model for the dependent variable 'Total Number of Defects' trained by the independent variables

'Functional Size' for the imputation and based on 360 projects. Table 4.1, also shows the parameter estimates for the Total Defects model: (constant = 26.89 TD and 0.0033 TD⁄CFP). Therefore, the TD estimation model for predicting the dependent Total Number of Defects variable based on the independent software 'Functional Size' variable is:

Total Number of Defects = 26.89 + 0.0033 * Functional Size.

From Table 4.1, it can be observed from the significant t-test and the p-value that are not statistically significant. So in this case an outliers' identification is required on the data fields of total number of defects and software size, because the non-significance results after the imputation procedure could be due to the impact of the outliers on the rest of the data point in the dataset. Then, these outliers will be eliminated from that data set N = 360 software projects. The outliers in the imputation might have an undue influence on the TD estimation models. Table 4.1 also shows the coefficients of determination (R^2) for the regression model after the imputation: here, the regression model of TD, the R^2, obtained after the imputation is (0.062), which is very low and indicates that outliers might influence the TD estimations models.

The Grubbs test is applied on the 'Total Number of Defects' variable of N=360 projects after the imputation in Table 4.2 in order to investigate whether some data points are truly outliers or not, and the results show that it has no outlier.

Table 4.2 Descriptive Statistics for Grubbs' test on 'Total Number of Defects' (N=360)

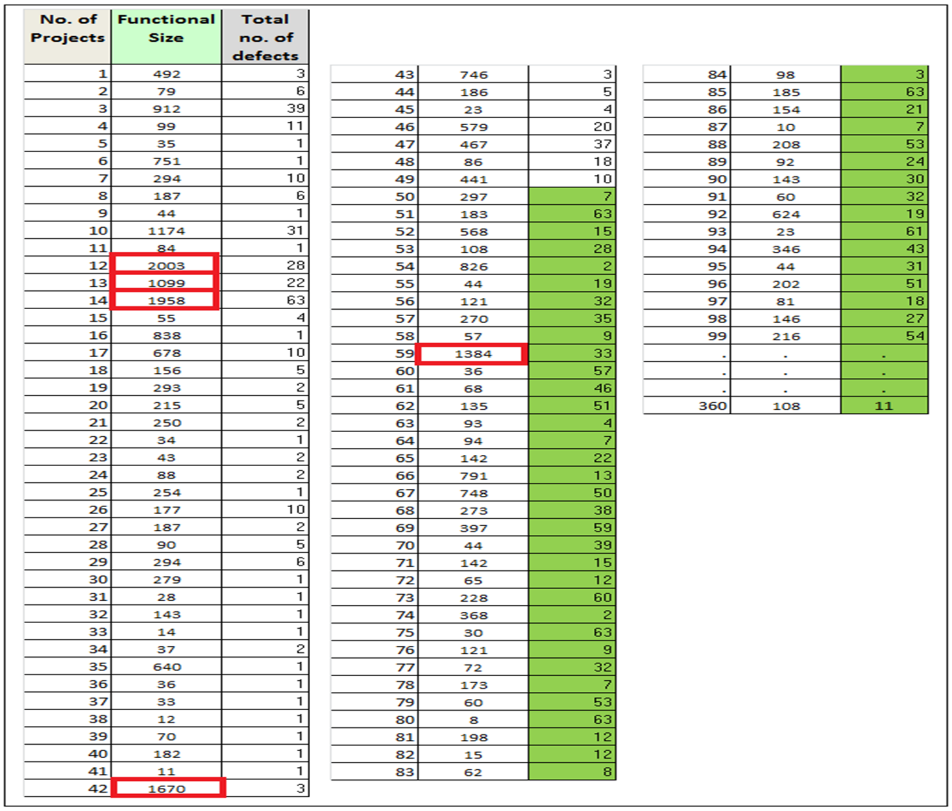
Test no.	**Mean Total Defects**	**SD**	**No. of values**	**Outlier Detected?**	**Significance level**	**Critical value of Z**
1	29.27	19.60	360	No	0.05 (two-sided)	3.774

However, when the Grubbs test is applied on the ‘Functional Size’ variable of N=360 projects, and the results show that it has 20 outliers in the ‘Functional Size’ data fields - see Table 4.3 and Figure 4.6. (See Appendix II)

Table 4.3 Descriptive Statistics for Grubbs' test on ‘Functional size’ (N=20 outliers)

Test no.	**Mean Total Defects**	**SD**	**No. of values**	**Outlier Detected?**	**Significance level**	**Critical value of Z**
1	262.06	349.97	360	Yes	0.05 (two-sided)	3.774
2	256.97	336.84	359	Yes	0.05 (two-sided)	3.773
3	252.09	324.37	358	Yes	0.05 (two-sided	3.772
4	247.32	311.96	357	Yes	0.05 (two-sided)	3.772
5	243.32	303.11	356	Yes	0.05 (two-sided)	3.771
6	239.50	294.82	355	Yes	0.05 (two-sided)	3.770
7	235.70	286.42	354	Yes	0.05 (two-sided)	3.769
8	231.92	277.84	353	Yes	0.05 (two-sided)	3.769
9	228.29	269.71	352	Yes	0.05 (two-sided)	3.768
10	224.91	262.52	351	Yes	0.05 (two-sided)	3.767
11	221.60	255.45	350	Yes	0.05 (two-sided)	3.766
12	218.64	249.76	349	Yes	0.05 (two-sided)	3.765
13	215.90	244.79	348	Yes	0.05 (two-sided)	3.765
14	213.16	239.76	347	Yes	0.05 (two-sided)	3.764
15	210.46	234.73	346	Yes	0.05 (two-sided)	3.763
16	207.74	229.56	345	Yes	0.05 (two-sided)	3.762
17	205.15	224.77	344	Yes	0.05 (two-sided)	3.761
18	202.54	219.84	343	Yes	0.05 (two-sided)	3.761
19	199.93	214.77	342	Yes	0.05 (two-sided)	3.760
20	197.48	210.26	341	Yes	0.05 (two-sided)	3.759
21	195.11	205.93	340	**No**	0.05 (two-sided)	3.758

Figure 4.6 presents an example of sample results of outliers within the variable of ‘Functional Size’. For instance, the data points highlighted with red are outliers (2003, 1958, 1099, 1384, and 1670 CFP).



No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	2
28	90	5
29	294	6
30	279	1
31	28	1
32	143	1
33	14	1
34	37	2
35	640	1
36	36	1
37	33	1
38	12	1
39	70	1
40	182	1
41	11	1
42	1670	3
43	746	3
44	186	5
45	23	4
46	579	20
47	467	37
48	86	18
49	441	10
50	297	7
51	183	63
52	568	15
53	108	28
54	826	2
55	44	19
56	121	32
57	270	35
58	57	9
59	1384	33
60	36	57
61	68	46
62	135	51
63	93	4
64	94	7
65	142	22
66	791	13
67	748	50
68	273	38
69	397	59
70	44	39
71	142	15
72	65	12
73	228	60
74	368	2
75	30	63
76	121	9
77	72	32
78	173	7
79	60	53
80	8	63
81	198	12
82	15	12
83	62	8
84	98	3
85	185	63
86	154	21
87	10	7
88	208	53
89	92	24
90	143	30
91	60	32
92	624	19
93	23	61
94	346	43
95	44	31
96	202	51
97	81	18
98	146	27
99	216	54
.	.	.
.	.	.
.	.	.
360	108	11

Figure 4.6 Examples of outliers found within functional size data fields

The Grubbs test is applied on the ‘Total Number of Defects’ (data fields) of N=340 projects in order to check the outliers after removing software projects (outliers) from the ‘Functional Size’ to verify whether after discarding projects from ‘Functional Size’ (as outliers) has affected the ‘Total Number of Defects’ projects (data fields). Table 4.4 indicates that it has no outliers with in its data points - see Table 4.4.

Table 4.4 Descriptive Statistics for Grubbs' test on Total Defects (after removing projects with software size outliers) (N=340)

Test no.	Mean Total Defects	SD	No. of values	Outlier Detected?	Significance level	Critical value of Z
1	29.13	19.67	340	No	0.05 (two-sided)	3.7586

Thus, after eliminating all the outliers from the independent variable 'Functional Size'; a linear regression analysis on TD estimation model is carried out in order to investigate how whether the outliers affected other data points within the data set - see Table 4.5. A graphical representation of data set N=340 project (with outliers removed) is presented - see Figure 4.7.

Table 4.5 Regression parameter analysis and statistical tests on TD estimation model based on the imputed dataset (N=340 projects - without outliers within functional size)

Variable	Intercept	Coefficient	R2	95% Confidence Limits		T-test	Standard Error	P-value
Functional Size	27.14	0.00213	0.063	-0.008	0.0124	0.41	0.0051	0.45

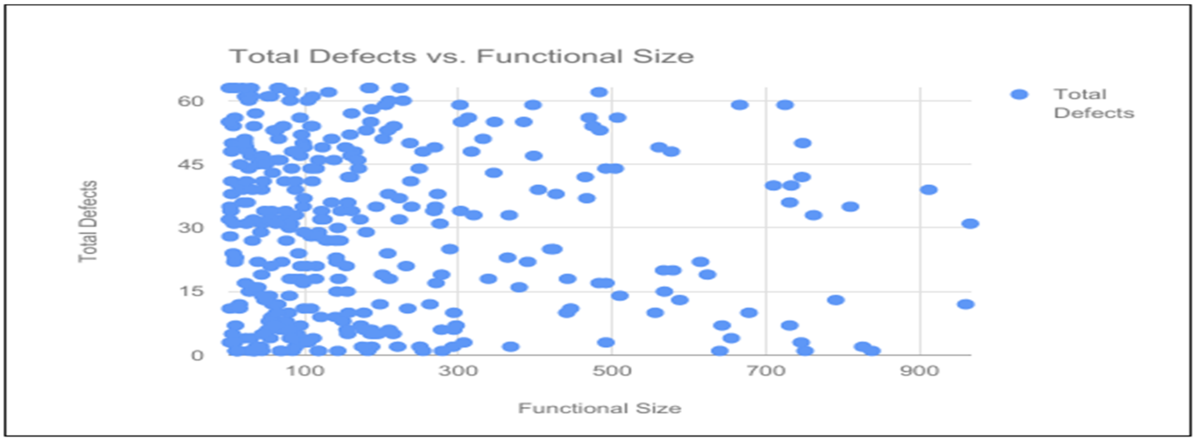


Figure 4.7 Graphical representation of functional size and total defects for imputed-SI dataset N=340 projects - without outliers

Tables 4.1 and 4.5 present the statistical analysis results (the results of the regression of TD estimation model for the variable ‘Total Number Defects’ trained with the variable ‘Functional Size’ for the imputation and based on 360 projects and 340 projects (with and without outliers within variables). It can be observed that the R^2 has changed very slightly from 0.062 (with outliers on size) to 0.064 (without outliers on size) has increased very little for the dataset without outliers, indicating that the outliers did not influence the estimation models.

The MMRE is 290% and the Pred(25) is 20% for assessing the TD estimation model (based on CFP) derived from the imputed SI dataset N=360 projects - see Table 4.6. The value of MMRE is very high after using Single imputation technique and the Pred is very low at 20%; therefore, this result explains the very low value of R^2.

Table 4.6 MMRE and Pred(25) for the TD estimation model
based on the imputed-SI dataset (N=360 projects)

MMRE	**Pred(25)**
290%	20%

Furthermore, considering the p-values in Tables 4.1 and 4.5, the results are not statistically significant at t-test and P-values with outliers for the total defects estimates, and in case of without outliers the p-value turns to statistically significant, while the t-test is not statistically significant.

Based on the above, using the ‘single imputation technique’ for the analysis does not represent the appropriate solution for a high ratio within TD missing data issue based on dataset N=360 projects.

Thus, this requires exploring more alternative imputation techniques to fit the need to handle the missing data issue.

Table 4.7 Summary of statistical tests for imputed-SI datasets (with and without outliers)

Variable	**Before removal of outliers within functional size N=360 projects**		**After removal of outliers within functional size N=340 projects**	
	Significant T-test	Significant P-values	Significant T-test	Significant P-values
Total Defects	No	No	No	Yes

Table 4.8 presents the averages of the Total Defects values imputed based on the seeds selected, and for the complete data, which included the outliers, and without outliers within the functional size data filed in the data set N=360 projects.

Table 4.8 Average Total Defects after SI imputation with and without outliers in Software Size

Variable	**Before removal of outliers within functional size N=360 projects**		**After removal of outliers within functional size N=340 projects**	
	Average completed data	**Average imputed data**	**Average completed data**	**Average imputed data**
Total Defects	6	32.613	8	32.614
# Projects	49	311	44	297

4.3 Summary

The missing values from the ISBSG R12 N=360 software projects sized by COSMIC are first imputed using: random numbers generated to provide the values that are missing from the selected variable ‘Total Number of defects’.

The Grubbs test was used next to identify the presence of outliers were performed on the 'Total Number of Defects', and 'Functional size' variables in the ISBSG repository. This analysis was applied to a sample of 360 observations of projects from the repository. When estimation models are built using data samples with outliers, these models distort only in a minor way the estimation models for future projects.

The obtained results were not statistically significant at t-test and P-values with outliers for the total defects estimates; coefficient of determination (R^2) was very low at 0.062 with outliers and 0.06346 without outliers. The standard error was 0.00293 and 0.0051 with and without outliers respectively, and in the case of without outliers the p-value turns to statistically significant. However, the t-test was not statistically significant, and the standard error is little increased. Based on the analysis results, using single imputation technique for the analysis does not provide a solution for the problem of missing data upon this research analysis. Thus, these results indicate to explore more appropriate imputation techniques which should fit the need to handle the missing data issue.

CHAPTER 5

REGRESSION IMPUTATION (RI)

This chapter describes the second step of phase 2 of the research methodology, by presenting how the regression imputation is implemented to impute the missing data fields of the 'Total Number of Defects' in order to produce a complete data set N = 360 of software projects. It also presents the linear regression modeling of the relationship of dependent variable 'Total Number of Defects' based on the independent variable 'Functional Size' (after imputing the TD missing data using Regression imputation).

5.1 Introduction

Regression imputation involves replacing each missing value with a predicted value based on a regression model (Bala, 2013). This estimation model is built using the complete observations of the variable to be estimated. For each missing observation, each missing value is replaced by the predicted value found using this estimation model (Little et Rubin, 2014).

Regression imputation is also referred to as a conditional mean imputation: it replaces missing values with the predicted scores from a linear regression equation (Saunders et al., 2006). Regression imputation is relatively straightforward if missing values are isolated on a single variable (i.e., there is a single, univariante missing data pattern). In this case, the incomplete variable is regressed on other measured variables, and missing values are replaced with the predicted scores from this analysis (Saunders et al., 2006). If there are two or more incomplete variables, a multivariate regression model has to be implemented in order to perform the imputation stage (Little et Rubin, 2014).

Consider: X_1, …., X_{k-1} are fully observed, and the X_k is observed for the (*r*) observations, and missing for the (n – r) observations, where (*n*) sample size, and *k* = 1, ..., r (Eurostat, 2016). Regression imputation computes the regression of X_k on X_1, …., X_{k-1} based on the

(r) complete observations and then fills in the missing values as predictions from the regression model (Eurostat, 2007), Assume the case where i has X_{ik} missing and X_{i1} ,…., $X_{i,k-1}$ observed, then, the missing value is imputed using the fitted regression equation (Eurostat, 2007):

$$\hat{X}_{ik}= \hat{\beta}_0 \ + \hat{\beta}_1 X_{i1} + \dots \ \hat{\beta}_{k-1} \, X_{i,k-1}$$

Where, β_0 is the intercept (which may be zero, leading to a regression through the origin), and β_1, …, β_{k-1} are respectively the regression coefficients of X_1, …., X_{k-1} based on the r complete observations (estimated parameters or predicted values of a variable are denoted by a ^) (Eurostat, 2007).

Regression imputation often gives reasonable estimates of means, particularly when the data are normally distributed. Empirical studies indicate that the regression imputation is more accurate than the previously described techniques (Raymond et Roberts, 1987), (Baraldi et Enders, 2010).

When regression imputation is used to fill in the missing data fields on a dependent variable, these missing fields can be admissibly predicted (García-Laencina, Sancho-Gómez et Figueiras-Vidal, 2010). On the other hand, if regression imputation is used to fill-in missing values on independent variables, the imputed values or data points can be admissibly correlated with the other variables (García-Laencina, Sancho-Gómez et Figueiras-Vidal, 2010).

In regression analysis, if an independent variable has a large percentage of missing values, then the slope estimates are less affected than the case when dependent variable has a comparable percentage of missing values (Saunders et al., 2006). The imputed data will preserve deviations from the mean as well as the shape of the distribution (Little, 1988), and (Wood, White et Thompson, 2004).

“*Regression imputation is well suited when the missing variables of interest are correlated with the data that are available in the complete sample*” (García-Laencina, Sancho-Gómez et Figueiras-Vidal, 2010).

Regression imputation has an advantage over the mean imputation: it preserves the variance and covariance of variables with missing data (García-Laencina, Sancho-Gómez et Figueiras-Vidal, 2010). It takes into account the relationships among the variables. Thus, this imputation approach by regression is more statistically efficient (García-Laencina, Sancho-Gómez et Figueiras-Vidal, 2010).

Regression imputation can produce parameter estimates that are consistent under MAR (Peugh et Enders, 2004). (Raymond et Roberts, 1987) suggested that regression is most useful when data are 10% - 40% incomplete. This imputation procedure may underestimate the data variability; however, to minimize this issue, using only the best predictor or set of predictors in the regression model can contribute to the largest percentage of variance in the regression model (Fox-Wasylyshyn et El-Masri, 2005). In our research work, verification strategies and statistical analysis are conducted in order to verify the predictive accuracy of estimation models.

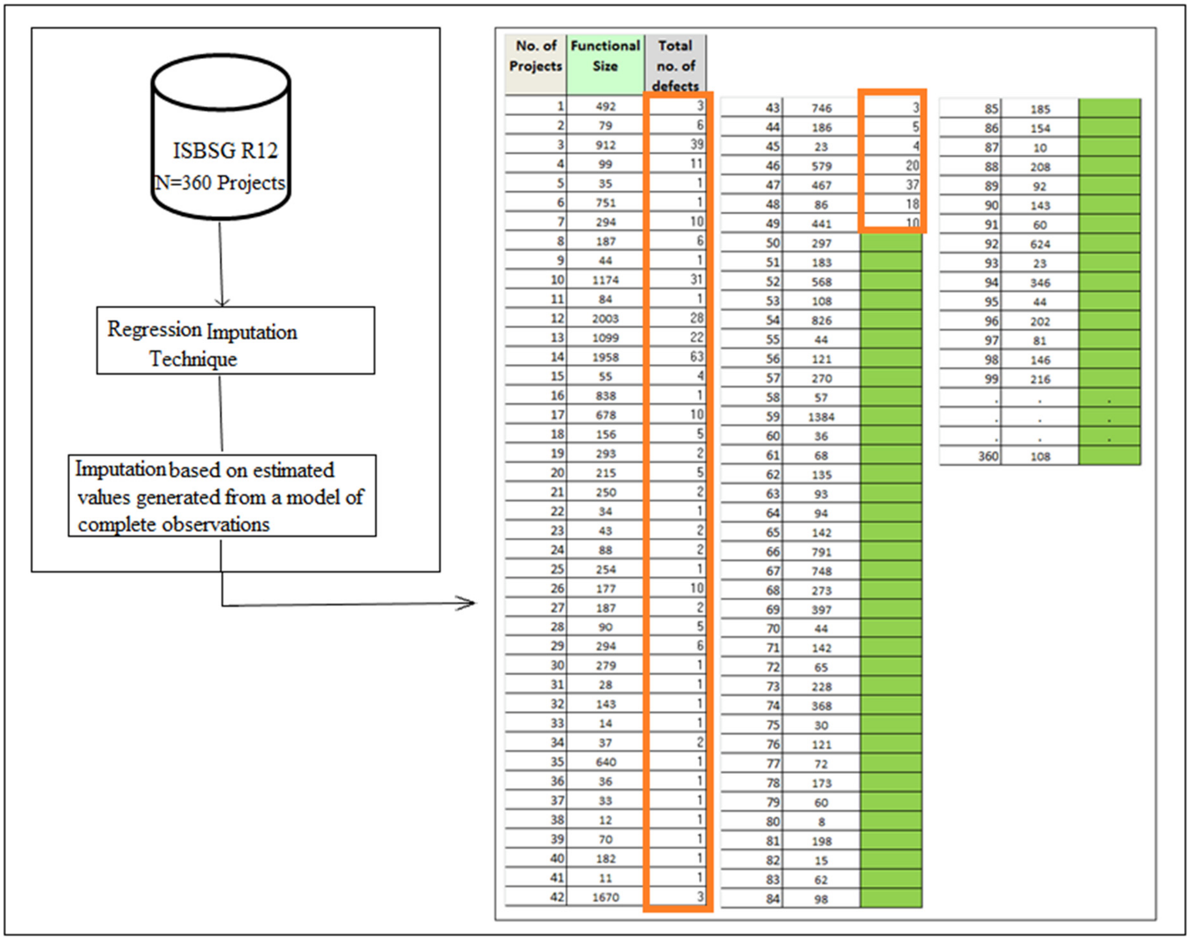
Although other regression imputation techniques exist (e.g., stepwise or iterative regression) (Wood, White et Thompson, 2004), only the single iteration will be illustrated here, because of the simplicity of missing data field to be imputed.

5.2 Implement the Imputation technique for Total Defects field with missing values

This section applies the imputation processing and defect estimation modeling as structured in Figure 3.10. It presents an application of the steps of the regression imputation process and a statistical inferences analysis on the dataset N = 360 of software projects.

This section is structured as follows:

1. Creating the imputed dataset (the imputation process). In this step (see Figure 5.1), the missing values of the variable ‘Total Number of Defects’ from the dataset N=360 of software projects are imputed by the predicted values generated using an estimation model from TD complete values as a dependent variable based on ‘Functional Size’ as an independent variable.


ISBSG R12
N=360 Projects
Regression Imputation Technique
Imputation based on estimated values generated from a model of complete observations

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	2
28	90	5
29	294	6
30	279	1
31	28	1
32	143	1
33	14	1
34	37	2
35	640	1
36	36	1
37	33	1
38	12	1
39	70	1
40	182	1
41	11	1
42	1670	3
43	746	3
44	186	5
45	23	4
46	579	20
47	467	37
48	86	18
49	441	10
50	297	🟩
51	183	🟩
52	568	🟩
53	108	🟩
54	826	🟩
55	44	🟩
56	121	🟩
57	270	🟩
58	57	🟩
59	1384	🟩
60	36	🟩
61	68	🟩
62	135	🟩
63	93	🟩
64	94	🟩
65	142	🟩
66	791	🟩
67	748	🟩
68	273	🟩
69	397	🟩
70	44	🟩
71	142	🟩
72	65	🟩
73	228	🟩
74	368	🟩
75	30	🟩
76	121	🟩
77	72	🟩
78	173	🟩
79	60	🟩
80	8	🟩
81	198	🟩
82	15	🟩
83	62	🟩
84	98	🟩
85	185	🟩
86	154	🟩
87	10	🟩
88	208	🟩
89	92	🟩
90	143	🟩
91	60	🟩
92	624	🟩
93	23	🟩
94	346	🟩
95	44	🟩
96	202	🟩
97	81	🟩
98	146	🟩
99	216	🟩
.	.	.
.	.	.
.	.	.
360	108	🟩

Figure 5.1 An example of sample results of Software projects to be imputed by RI on Total Number of Defects

- Data set with sample size N = 360 projects: 49 projects have completed total defects, and 311 projects have missing data.

- 311 projects with missing TD are to be imputed based on Regression imputation technique by estimated values that generated are from a TD estimation model of complete observations within 'Total Number of Defects' and the independent variable 'Functional Size'. (Based on the available data within the field with missing values).

Given the complete data N=49 projects, a TD estimation model (based on the independent variable 'Functional size') will be built with both the initial complete data set N=49 projects - see Table 5.1.

Table 5.1, displays a 95% mean confidence interval and a t-test with the associated P-value and whether the independent variable 'Functional size' has impact on the TD parameter estimates (of complete observations, N=49 projects): the inferences are based on the t-distribution, and followed by a graphical representation of 'Total Number of Defects' based on 'Functional Size' - see Figure 5.2.

Table 5.1 Regression parameter analysis and statistical tests for TD estimation model based on the completed dataset, N=49 projects

Variable	**Intercept**	**Coefficient**	**R2**	**95% Confidence Limits**		**T-test**	**Standard Error**	**P-value**
Functional Size	1.63	0.017	0.5	0.0113	0.0225	6.1	0.0028	1.95801E-07

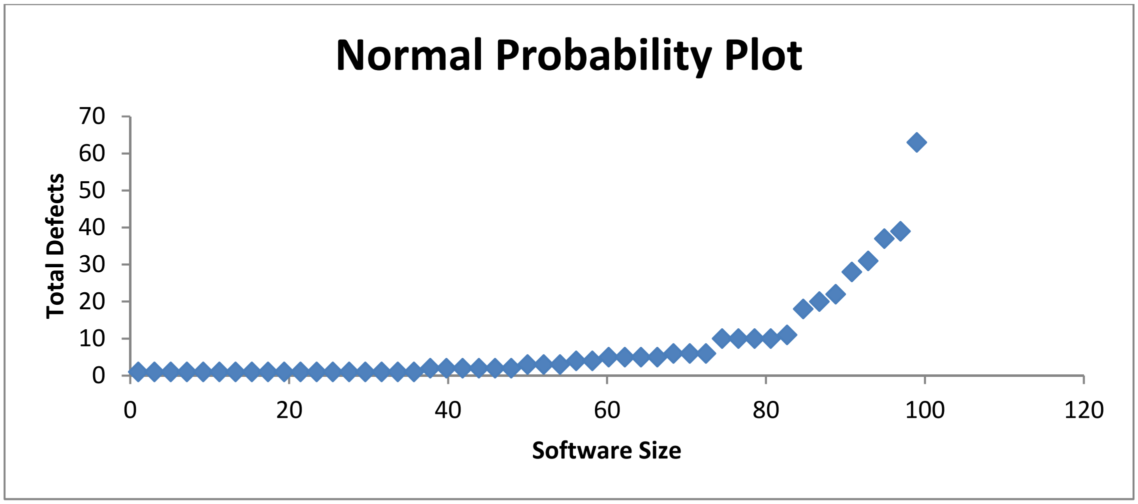


Figure 5.2 Normal probability plot of Total Defects and Functional Size based on the complete dataset N=49 projects

Table 5.1 presents the results of the TD estimation model (to be used for generating predicted values as ‘imputes’ for the missing TD) for the variable ‘Total Number of Defects’ trained with the independent variables ‘Functional Size’ for the imputation and based on the reported total defects of 49 projects. Table 5.1, also shows the parameter estimates for the ‘Total Number of Defects’ estimation model are: (constant = 1.63 TD and 0.017 TD/CFP). Therefore, the TD estimation model based on the complete subset X, N=49 projects which to be used for the ‘imputes’ estimation is:

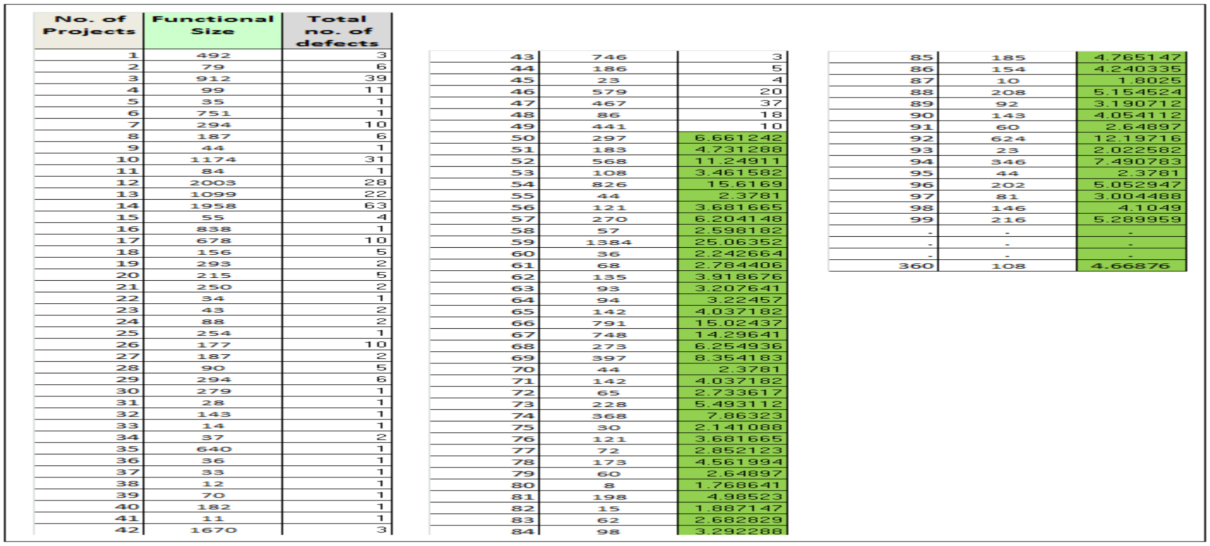
Total Number of Defects = 1.63 + 0.017 * Functional Size

It also can be observed from Table 5.1 that the t-test and the p-value are statistically significant - see Table 5.2 as well. Table 5.1 also shows the coefficients of determination (R^2) which is (0.5) for the TD estimation model (based on ‘Functional Size’) that to be used for the imputation procedure.

Table 5.2 Summary of the statistical tests of TD estimation model based on complete dataset N=49 projects

Variable	**Complete dataset** **N=49 projects**	
	Significant **T-test**	**Significant** **P-values**
Functional Size	Yes	Yes

Figure 5.3 presents an example of sample results of software projects with imputed data fields of 'Total Number of Defects' by the regression imputation technique.



No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	2
28	90	5
29	294	6
30	279	1
31	28	1
32	143	1
33	14	1
34	37	2
35	640	1
36	36	1
37	33	1
38	12	1
39	70	1
40	182	1
41	11	1
42	1670	3
43	746	3
44	186	5
45	23	4
46	579	20
47	467	37
48	86	18
49	441	10
50	297	6.661242
51	183	4.731288
52	568	11.24911
53	108	3.461582
54	826	15.6169
55	44	2.3781
56	121	3.681665
57	270	6.204148
58	57	2.598182
59	1384	25.06352
60	36	2.242664
61	68	2.784406
62	135	3.918676
63	93	3.207641
64	94	3.22457
65	142	4.037182
66	791	15.02437
67	748	14.29641
68	273	6.254936
69	397	8.354183
70	44	2.3781
71	142	4.037182
72	65	2.733617
73	228	5.493112
74	368	7.86323
75	30	2.141088
76	121	3.681665
77	72	2.852123
78	173	4.561994
79	60	2.64897
80	8	1.768641
81	198	4.98523
82	15	1.887147
83	62	2.682829
84	98	3.292288
85	185	4.765147
86	154	4.240335
87	10	1.8025
88	208	5.154524
89	92	3.190712
90	143	4.054112
91	60	2.64897
92	624	12.19716
93	23	2.022582
94	346	7.490783
95	44	2.3781
96	202	5.052947
97	81	3.004488
98	146	4.1049
99	216	5.289959
.	.	.
.	.	.
.	.	.
360	108	4.66876

Figure 5.3 Examples results of software projects with data points of Total Number of Defects generated by regression imputation N=360 projects

2. The next procedure is to perform a linear regression analysis on the imputed dataset N=360 projects through building linear regression models that is: to estimate the dependent variable 'Total Number of Defects' on the basis of the independent variables 'Functional size' as structured in step (2) in Figure 3.11 earlier.

Table 5.3 displays a 95% mean confidence interval and a t-test with the associated P-value and verifying the impact of 'Functional Size' on TD parameter estimates: the inferences are based on the t-distribution and followed a graphical representation of the relationship of TD based on Size in CFP - see Figure 5.6, which indicates that for every increase of 1000 CFP in “Functional Size”, “Total Defects” increases by almost 16.9 TD.

Table 5.3 Regression parameters and statistical tests analysis of TD estimation model based on the imputed-RI dataset, N=360 projects

Variable	**Intercept**	**Coefficients**	**R2**	**95% Confidence Limits**		**T- test**	**Standard Error**	**P-value**
Functional Size	1.4	0.017	0.8	0.01606	0.01815	32.15	0.0005322	1.8672E-107

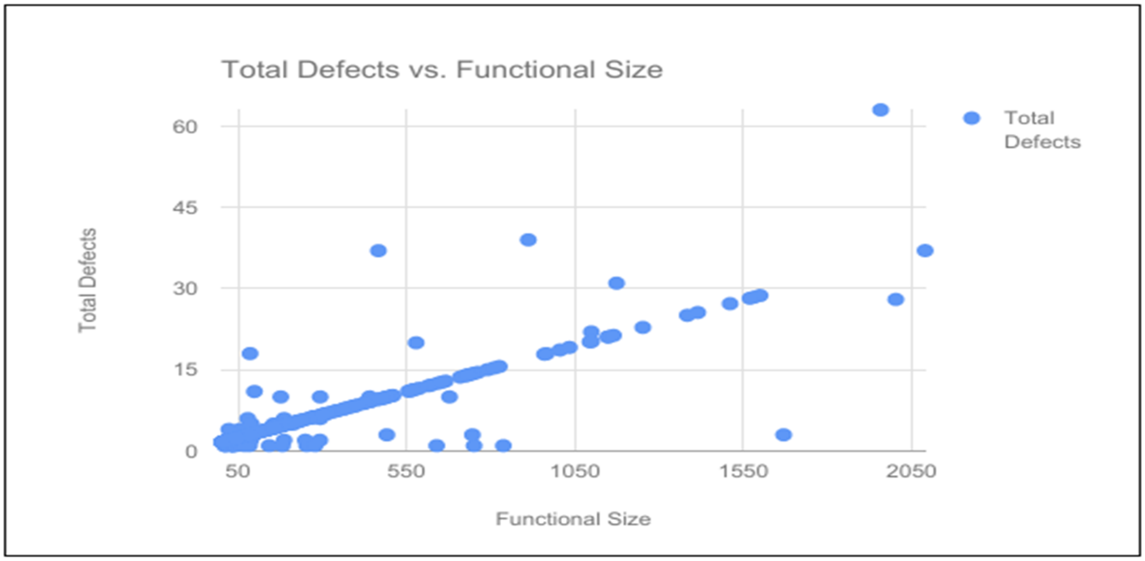


Figure 5.4 Graphical representation of the relationship of TD based on Size in CFP for imputed-RI dataset, N=360 projects

Table 5.3 presents the results of the regression analysis of TD estimation model for the variable ‘Total Defects’ trained by the independent variable ‘Functional Size’ for the imputation of 360 projects. Table 5.3, also shows the parameter estimates for the ‘Total Number of Defects’ estimation model: (constant = 1.4 TD and 0.017 TD/CFP). Therefore, the TD estimation model based on the imputed data set of N=360 projects of dependent ‘Total Number of Defect’ variable based on the independent variable ‘Functional Size’ is:

Total Number of Defects = 1.4 + 0.017 * Functional Size

It also can be observed from Table 5.3 that the significant t-test and the p-value imply are statistically significant (32.15 and 1.8672E-107 respectively) (see Table 5.4). Thus, in this case, the outliers' identification is not 'mandatory' required on the data fields of total number of defects and software size, because the statistical analysis indicates to significance results after the imputation procedure. However, it is preferable to check whether the imputed data fields contain outliers: this has been verified for such purpose (see Appendix III).

Table 5.3 also shows the coefficients of determination (R^2) for the regression model for the imputation. The R^2 obtained after the imputation is 0.8, which is large and it has significantly increased compared with the value of R^2 when the Single imputation technique used in chapter 4. The confidence interval is shaped upward (LL is 0.01606, and UL is 0.01815).

Table 5.4 Summary of the statistical tests analysis of TD estimation model from imputed-RI dataset N=360 projects

Variable	**RI-imputed dataset N=360 projects**	
	Significant T-test	**Significant P-values**
Functional Size	Yes	Yes

The MMRE is 31% and the Pred(25) is 85% for assessing the TD estimation model (based on CFP) derived from the imputed RI dataset N=360 projects - see Table 5.5. The value of MMRE is very low after using Regression imputation technique when it compared with the 290% when the Single imputation has used.

Table 5.5 MMRE and Pred(25) for TD estimation model based on the imputed-RI dataset N=360 projects

MMRE	Pred(25)
31%	85%

Table 5.6 presents the averages of the TD value imputed based on Regression imputation in the data set N=360 projects.

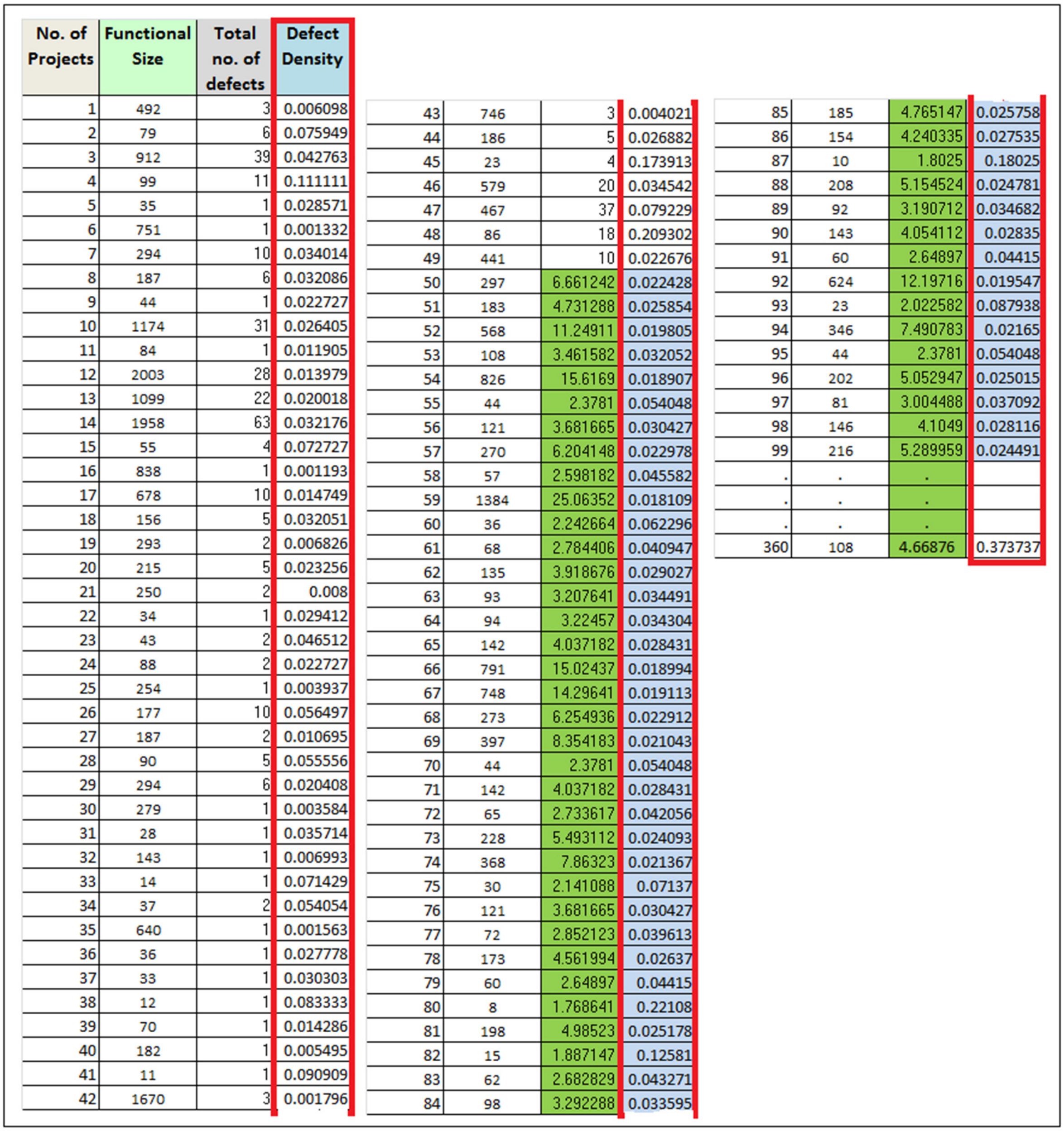
Table 5.6 Average of Total Number of Defects after RI imputation of TD

Variable	Averages of TD N=360 projects	
	Average complete data	Average imputed data
Total Defects	8	6
# Projects	49	311

Next, the values of the defect density are observed for the imputed data N=360 projects (after imputing the TD missing data by RI) that have blank fields of defect density caused by missing of TD values (the software size values are all recorded within the data set N=360) - see Figure 5.5.

- Calculate the Defect Density (DD) after the imputation for the total defects, based on the formula:

$$\text{Defect Density} = \frac{Total\ Number\ of\ Defects}{Functional\ Size} \text{ , (ISBSG, 2013)}$$



No. of Projects	Functional Size	Total no. of defects	Defect Density
1	492	3	0.006098
2	79	6	0.075949
3	912	39	0.042763
4	99	11	0.111111
5	35	1	0.028571
6	751	1	0.001332
7	294	10	0.034014
8	187	6	0.032086
9	44	1	0.022727
10	1174	31	0.026405
11	84	1	0.011905
12	2003	28	0.013979
13	1099	22	0.020018
14	1958	63	0.032176
15	55	4	0.072727
16	838	1	0.001193
17	678	10	0.014749
18	156	5	0.032051
19	293	2	0.006826
20	215	5	0.023256
21	250	2	0.008
22	34	1	0.029412
23	43	2	0.046512
24	88	2	0.022727
25	254	1	0.003937
26	177	10	0.056497
27	187	2	0.010695
28	90	5	0.055556
29	294	6	0.020408
30	279	1	0.003584
31	28	1	0.035714
32	143	1	0.006993
33	14	1	0.071429
34	37	2	0.054054
35	640	1	0.001563
36	36	1	0.027778
37	33	1	0.030303
38	12	1	0.083333
39	70	1	0.014286
40	182	1	0.005495
41	11	1	0.090909
42	1670	3	0.001796
43	746	3	0.004021
44	186	5	0.026882
45	23	4	0.173913
46	579	20	0.034542
47	467	37	0.079229
48	86	18	0.209302
49	441	10	0.022676
50	297	6.661242	0.022428
51	183	4.731288	0.025854
52	568	11.24911	0.019805
53	108	3.461582	0.032052
54	826	15.6169	0.018907
55	44	2.3781	0.054048
56	121	3.681665	0.030427
57	270	6.204148	0.022978
58	57	2.598182	0.045582
59	1384	25.06352	0.018109
60	36	2.242664	0.062296
61	68	2.784406	0.040947
62	135	3.918676	0.029027
63	93	3.207641	0.034491
64	94	3.22457	0.034304
65	142	4.037182	0.028431
66	791	15.02437	0.018994
67	748	14.29641	0.019113
68	273	6.254936	0.022912
69	397	8.354183	0.021043
70	44	2.3781	0.054048
71	142	4.037182	0.028431
72	65	2.733617	0.042056
73	228	5.493112	0.024093
74	368	7.86323	0.021367
75	30	2.141088	0.07137
76	121	3.681665	0.030427
77	72	2.852123	0.039613
78	173	4.561994	0.02637
79	60	2.64897	0.04415
80	8	1.768641	0.22108
81	198	4.98523	0.025178
82	15	1.887147	0.12581
83	62	2.682829	0.043271
84	98	3.292288	0.033595
85	185	4.765147	0.025758
86	154	4.240335	0.027535
87	10	1.8025	0.18025
88	208	5.154524	0.024781
89	92	3.190712	0.034682
90	143	4.054112	0.02835
91	60	2.64897	0.04415
92	624	12.19716	0.019547
93	23	2.022582	0.087938
94	346	7.490783	0.02165
95	44	2.3781	0.054048
96	202	5.052947	0.025015
97	81	3.004488	0.037092
98	146	4.1049	0.028116
99	216	5.289959	0.024491
.	.	.	
.	.	.	
.	.	.	
360	108	4.66876	0.373737

Figure 5.5 Example of sample results of observed DD after imputing TD data points by RI N=360 projects

Figure 5.6 shows the defect density of imputed RI data set of N=360 software projects sized by the COSMIC method, with a range from 0.0012 to 0.834 TD/CFP, with most values at the low end. The median is 0.0294 TD/CFP. The values of defect density peaks at 0.561 and 0.834 TD/CFP.

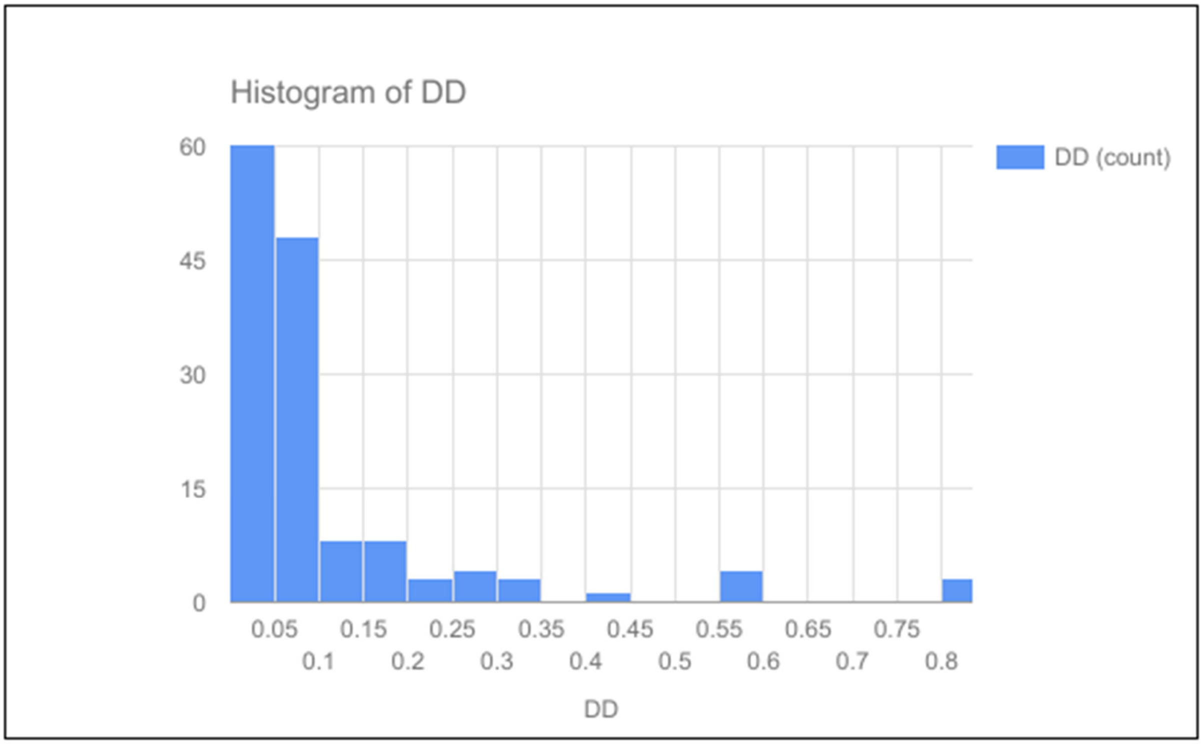


Figure 5.6 Defect density of imputed dataset by RI, N=360 projects

5.3 Summary

The missing values from the ISBSG R12 N = 360 software projects sized by COSMIC were imputed using Regression imputation: the imputed values were generated from an TD estimation model of complete observations (N=49 projects) within the dependent variable of 'Total Number of Defects' (e.g., the available data within the TD with missing values) based on an independent variable 'Functional Size' in CFP.

The obtained results were statistically significant at t-test and P-values of 'Total Number of Defects' estimates; the coefficients of determination (R^2) was 0.8 for the TD estimation model for the imputation, the confidence interval does contain true parameters for the estimates as well as the standard error has decreased. The value of MMRE was very low 31% and the Pred has increased to 85% after using 'regression imputation technique' compared with the 290% of MMRE and the 20% of Pred when the 'single imputation technique' was used. Based on that, using Regression imputation technique for the analysis provided a solution for the problem of missing data where it performs better than the technique of Single imputation for this specific dataset. However, these interesting results did not prevent to explore for more appropriate imputation techniques since the regression imputation might underestimate the standard error and lacks the variability of data which may lead to bias. Even with this efficient performance of the regression imputation, the variability among data and standard errors need to be adjusted. In an attempt to resolve this issue, some researchers use a modified version of regression imputation technique that is called ‘stochastic regression imputation technique’ - see next chapter.

CHAPTER 6

STOCHASTIC REGRESSION IMPUTATION

This chapter presents the last step of Phase 2 by describing how the stochastic regression imputation technique is implemented to impute the missing data fields of the 'Total Number of Defects' in order to produce a complete data set N = 360 of software projects. It also presents the linear regression modeling of the relationship of dependent variable 'Total Number of Defects' based on the independent variable 'Functional Size' (after imputing the TD missing data using stochastic regression imputation).

6.1 Introduction

Stochastic regression imputation represents a successful attempt of correcting the lack of an error term in regression imputation, by adding the average regression variance to the regression imputations to reform error (Enders, 2010). Stochastic regression shows much less bias than the previously mentioned techniques (Enders, 2010).

Stochastic regression also uses regression equation to predict the missing data from the complete data within the used dependent variable with missing data, but it takes an extra step to enhance each predicted value with a normality distributed residual term (Enders, 2010). Adding the residuals to the imputed values restores that lost variability to the data and effectively eliminates the bias linked with imputation procedures. This residual term, is a random value from a normal distribution with a mean of zero and a variance equal to a residual variance from the regression of complete data (Enders, 2010).

Consider that X_1, …., X_{k-1} are fully observed, and the X_k is observed for the (r) observations, and missing for the (n – r) observations, where (n) sample size, and k = 1, ..., r (Eurostat, 2016). Regression imputation computes the regression of X_k on X_1, …., X_{k-1} based on the (r) complete observations and then fills in the missing values as predictions from the regression model (Eurostat, 2016), Assume the case where i has X_{ik} missing and X_{i1}

,…., $X_{i,k-1}$ observed, then, the missing value is imputed using the fitted regression equation (1) (Eurostat, 2007):

$$\hat{X}_{ik}= \hat{\beta}_0 \ + \hat{\beta}_1 X_{i1} + \dots \ \hat{\beta}_{k-1} \, X_{i,k-1} + z_i$$

Where, β_0 is the intercept (which may be zero, leading to a regression through the origin), and β_1, …, β_{k-1} are respectively the regression coefficients of X_1, …., X_{k-1} based on the *r* complete observations (estimated parameters or predicted values of a variable are denoted by a ^), and z_i is the residual term (Eurostat, 2007).

6.2 Implement the Imputation technique for Total Defects field with missing values

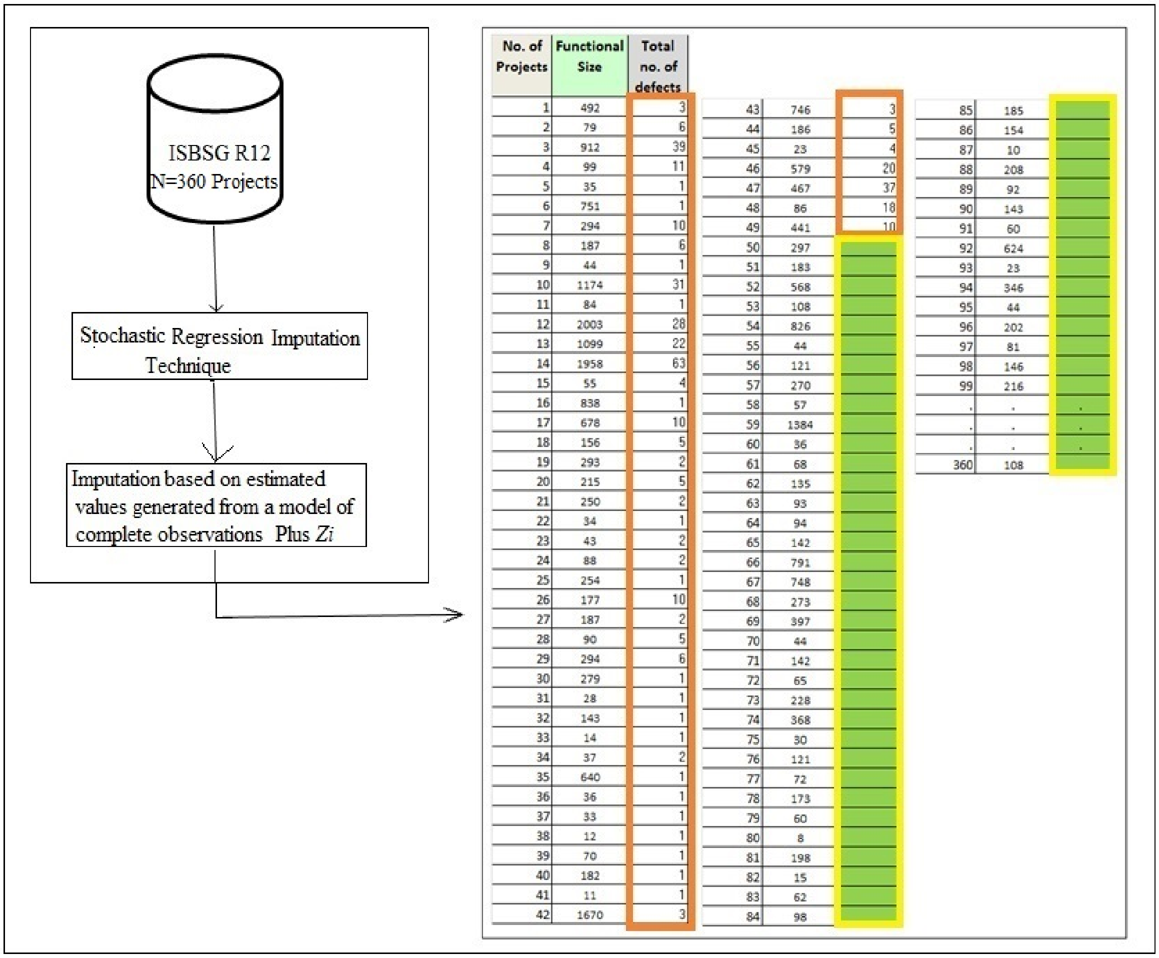
This section applies the imputation processing and defect estimation modeling activities as structured in Figure 3.10. It presents an application of the steps of the stochastic regression imputation process and a statistical inferences analysis on the dataset N = 360 of software projects.

This section is structured as follows:

1. Creating the imputed data set (the imputation process). This step follows similar imputation steps of the standard regression imputation where the missing values are imputed: the predicted values are generated using an estimation model from the complete values within the dependent variable to be imputed (e.g., The ‘Total Number of Defects’). The complete values are observations reported within the same variable of total defects (the TD estimation step of those complete values is accomplished previously on standard regression imputation based on the independent variable ‘Functional Size’), with a residual term added. The TD estimation model based on the complete subset X, N=49 software projects:

$$\text{Total Number of Defects} = 1.63 + 0.017 * \text{Functional Size} + z_i$$

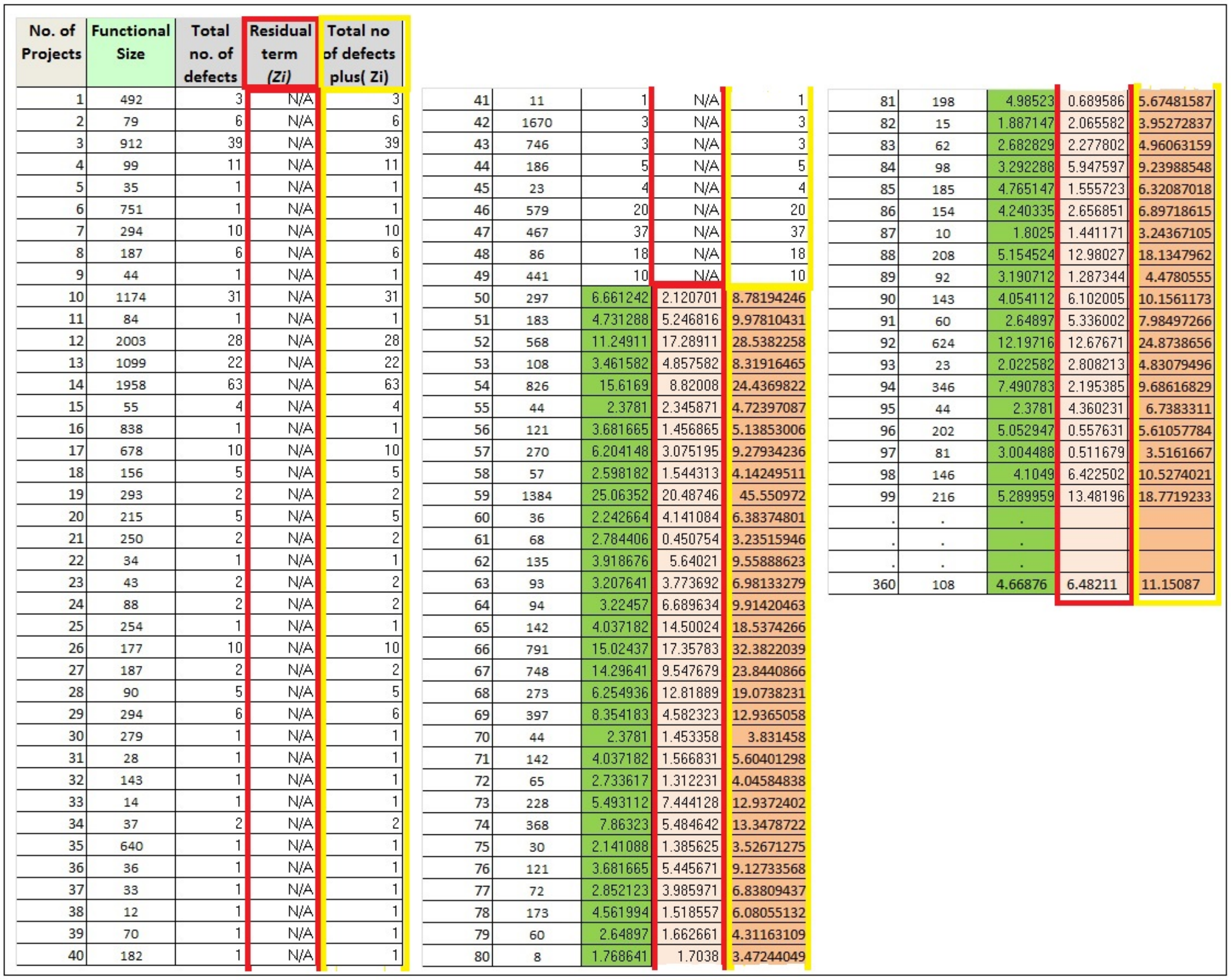
In addition to that, the normality distributed residual term is to be added to the predicted values from the estimated model generated from the reported values - see Figure 6.1.


ISBSG R12
N=360 Projects
Stochastic Regression Imputation Technique
Imputation based on estimated values generated from a model of complete observations Plus *Zi*

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	2
28	90	5
29	294	6
30	279	1
31	28	1
32	143	1
33	14	1
34	37	2
35	640	1
36	36	1
37	33	1
38	12	1
39	70	1
40	182	1
41	11	1
42	1670	3
43	746	3
44	186	5
45	23	4
46	579	20
47	467	37
48	86	18
49	441	10
50	297	🟩
51	183	🟩
52	568	🟩
53	108	🟩
54	826	🟩
55	44	🟩
56	121	🟩
57	270	🟩
58	57	🟩
59	1384	🟩
60	36	🟩
61	68	🟩
62	135	🟩
63	93	🟩
64	94	🟩
65	142	🟩
66	791	🟩
67	748	🟩
68	273	🟩
69	397	🟩
70	44	🟩
71	142	🟩
72	65	🟩
73	228	🟩
74	368	🟩
75	30	🟩
76	121	🟩
77	72	🟩
78	173	🟩
79	60	🟩
80	8	🟩
81	198	🟩
82	15	🟩
83	62	🟩
84	98	🟩
85	185	🟩
86	154	🟩
87	10	🟩
88	208	🟩
89	92	🟩
90	143	🟩
91	60	🟩
92	624	🟩
93	23	🟩
94	346	🟩
95	44	🟩
96	202	🟩
97	81	🟩
98	146	🟩
99	216	🟩
.	.	.
.	.	.
.	.	.
360	108	🟩

Figure 6.1 An example of Software projects with TD missing data to be imputed by SRI, N=360 projects

This involves generating random values from a normal distribution with a mean of zero and a variance equal to a residual variance from the regression of complete data (Enders, 2010): these generated values as mentioned earlier are to be added to the estimated values, as an attempt to restore the lost variability to the data - see Figure 6.2.



No. of Projects	Functional Size	Total no. of defects	Residual term *(Zi)*	Total no of defects plus(Zi)
1	492	3	N/A	3
2	79	6	N/A	6
3	912	39	N/A	39
4	99	11	N/A	11
5	35	1	N/A	1
6	751	1	N/A	1
7	294	10	N/A	10
8	187	6	N/A	6
9	44	1	N/A	1
10	1174	31	N/A	31
11	84	1	N/A	1
12	2003	28	N/A	28
13	1099	22	N/A	22
14	1958	63	N/A	63
15	55	4	N/A	4
16	838	1	N/A	1
17	678	10	N/A	10
18	156	5	N/A	5
19	293	2	N/A	2
20	215	5	N/A	5
21	250	2	N/A	2
22	34	1	N/A	1
23	43	2	N/A	2
24	88	2	N/A	2
25	254	1	N/A	1
26	177	10	N/A	10
27	187	2	N/A	2
28	90	5	N/A	5
29	294	6	N/A	6
30	279	1	N/A	1
31	28	1	N/A	1
32	143	1	N/A	1
33	14	1	N/A	1
34	37	2	N/A	2
35	640	1	N/A	1
36	36	1	N/A	1
37	33	1	N/A	1
38	12	1	N/A	1
39	70	1	N/A	1
40	182	1	N/A	1
41	11	1	N/A	1
42	1670	3	N/A	3
43	746	3	N/A	3
44	186	5	N/A	5
45	23	4	N/A	4
46	579	20	N/A	20
47	467	37	N/A	37
48	86	18	N/A	18
49	441	10	N/A	10
50	297	6.661242	2.120701	8.78194246
51	183	4.731288	5.246816	9.97810431
52	568	11.24911	17.28911	28.5382258
53	108	3.461582	4.857582	8.31916465
54	826	15.6169	8.82008	24.4369822
55	44	2.3781	2.345871	4.72397087
56	121	3.681665	1.456865	5.13853006
57	270	6.204148	3.075195	9.27934236
58	57	2.598182	1.544313	4.14249511
59	1384	25.06352	20.48746	45.550972
60	36	2.242664	4.141084	6.38374801
61	68	2.784406	0.450754	3.23515946
62	135	3.918676	5.64021	9.55888623
63	93	3.207641	3.773692	6.98133279
64	94	3.22457	6.689634	9.91420463
65	142	4.037182	14.50024	18.5374266
66	791	15.02437	17.35783	32.3822039
67	748	14.29641	9.547679	23.8440866
68	273	6.254936	12.81889	19.0738231
69	397	8.354183	4.582323	12.9365058
70	44	2.3781	1.453358	3.831458
71	142	4.037182	1.566831	5.60401298
72	65	2.733617	1.312231	4.04584838
73	228	5.493112	7.444128	12.9372402
74	368	7.86323	5.484642	13.3478722
75	30	2.141088	1.385625	3.52671275
76	121	3.681665	5.445671	9.12733568
77	72	2.852123	3.985971	6.83809437
78	173	4.561994	1.518557	6.08055132
79	60	2.64897	1.662661	4.31163109
80	8	1.768641	1.7038	3.47244049
81	198	4.98523	0.689586	5.67481587
82	15	1.887147	2.065582	3.95272837
83	62	2.682829	2.277802	4.96063159
84	98	3.292288	5.947597	9.23988548
85	185	4.765147	1.555723	6.32087018
86	154	4.240335	2.656851	6.89718615
87	10	1.8025	1.441171	3.24367105
88	208	5.154524	12.98027	18.1347962
89	92	3.190712	1.287344	4.4780555
90	143	4.054112	6.102005	10.1561173
91	60	2.64897	5.336002	7.98497266
92	624	12.19716	12.67671	24.8738656
93	23	2.022582	2.808213	4.83079496
94	346	7.490783	2.195385	9.68616829
95	44	2.3781	4.360231	6.7383311
96	202	5.052947	0.557631	5.61057784
97	81	3.004488	0.511679	3.5161667
98	146	4.1049	6.422502	10.5274021
99	216	5.289959	13.48196	18.7719233
.	.	.		
.	.	.		
.	.	.		
360	108	4.66876	6.48211	11.15087

Figure 6.2 Example of Software projects with imputed- SRI data points of Total Number of Defects with residual term added, N=360 projects

2. The next procedure is to perform regression analysis on the imputed dataset N=360 projects through building linear regression models that is: to estimate the dependent variable ‘Total Number of Defects’ on the basis of the independent variables ‘Functional size’ as structured in step (2) in Figure 3.11 earlier.

Table 6.1 displays a 95% mean confidence interval and a t-test with the associated P-value and verifying the impact of ‘Software Size’ on TD parameter estimates: the inferences are based on the t-distribution and followed a graphical representation of the relationship of TD based on Size in CFP - see Figure 6.3, which indicates that for every increase of 1000 CFP in ‘Functional Size’, ‘Total Defects’ increases by 16.5 TD.

Table 6.1 Regression parameters analysis and statistical tests for TD estimation model of 'Total number of Defects' and 'Functional Size' based on the SRI-imputed dataset, N=360 projects

Variable	**Intercept**	**Coefficient**	**R2**	**95% Confidence Limits**		**T-test**	**Standard Error**	**P-value**
Functional Size	3.8	0.025	0.7	0.02317	0.0269	25.9	0.00096	3.05032 E-84

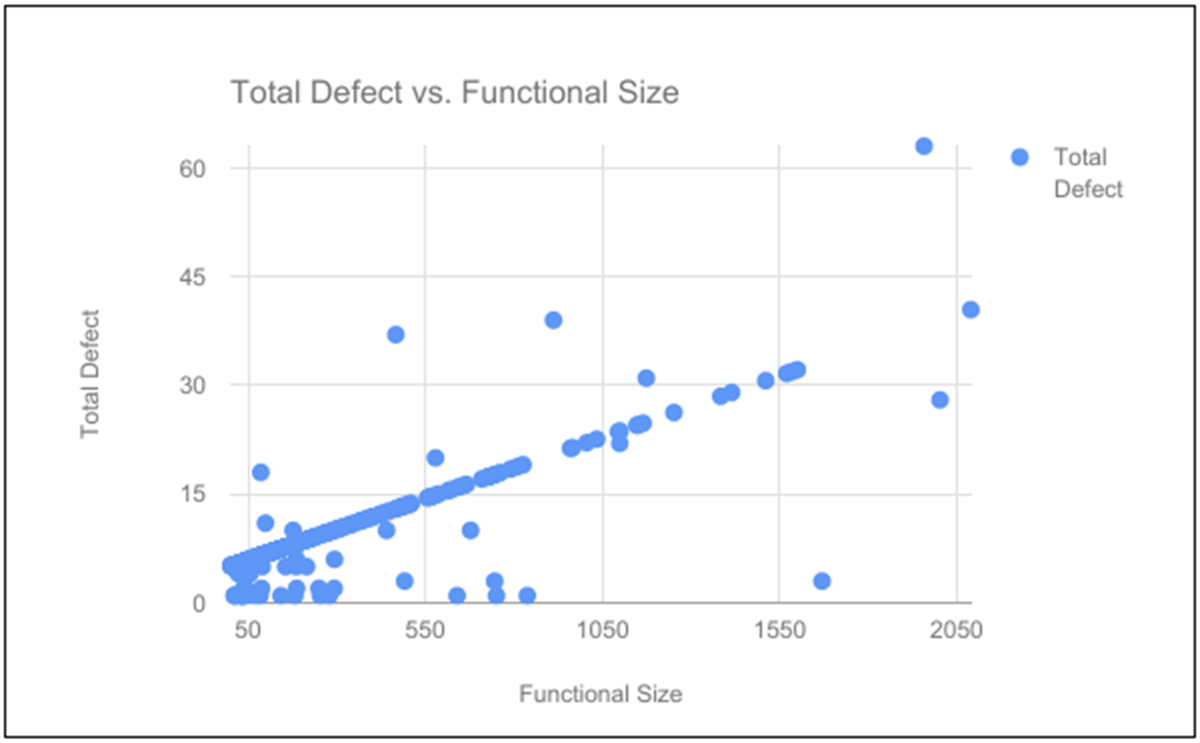


Figure 6.3 Graphical representation of functional size and total defects based on the SRI-imputed dataset N=360 projects

Table 6.1 presents the results of the regression estimation model for the variable 'Total Number of Defects' trained with the independent variable 'Functional Size' for the imputation of 360 projects. Table 6.1 also shows the parameter estimates for the Total Defects model: (constant = 3.8 TD and 0.025 TD/CFP). Therefore, the TD estimation model based on the imputed data set of N=360 projects of dependent 'Total Number of Defect' variable based on the independent variable 'Functional Size' is:

Total Number of Defects = 3.8 + 0.025 * Functional Size

It also can be observed from Table 6.1 that the t-test and the p-value are statistically significant (25.9, and 3.05032E-84 respectively). Table 6.1 also shows the coefficients of determination (R^2) is 0.7 for the TD estimation model after the imputation. The confidence interval is shaped more upward (LL is 0.02317, and UL is 0.02697).

Table 6.2 shows that the MMRE is 77% and the Pred(25) is 50% for assessing the TD estimation model (based on CFP) derived from the imputed SRI dataset N=360 projects.

Table 6.2 MMRE and Pred(25) for TD estimation model
based on the SRI-imputed dataset (N=360 projects)

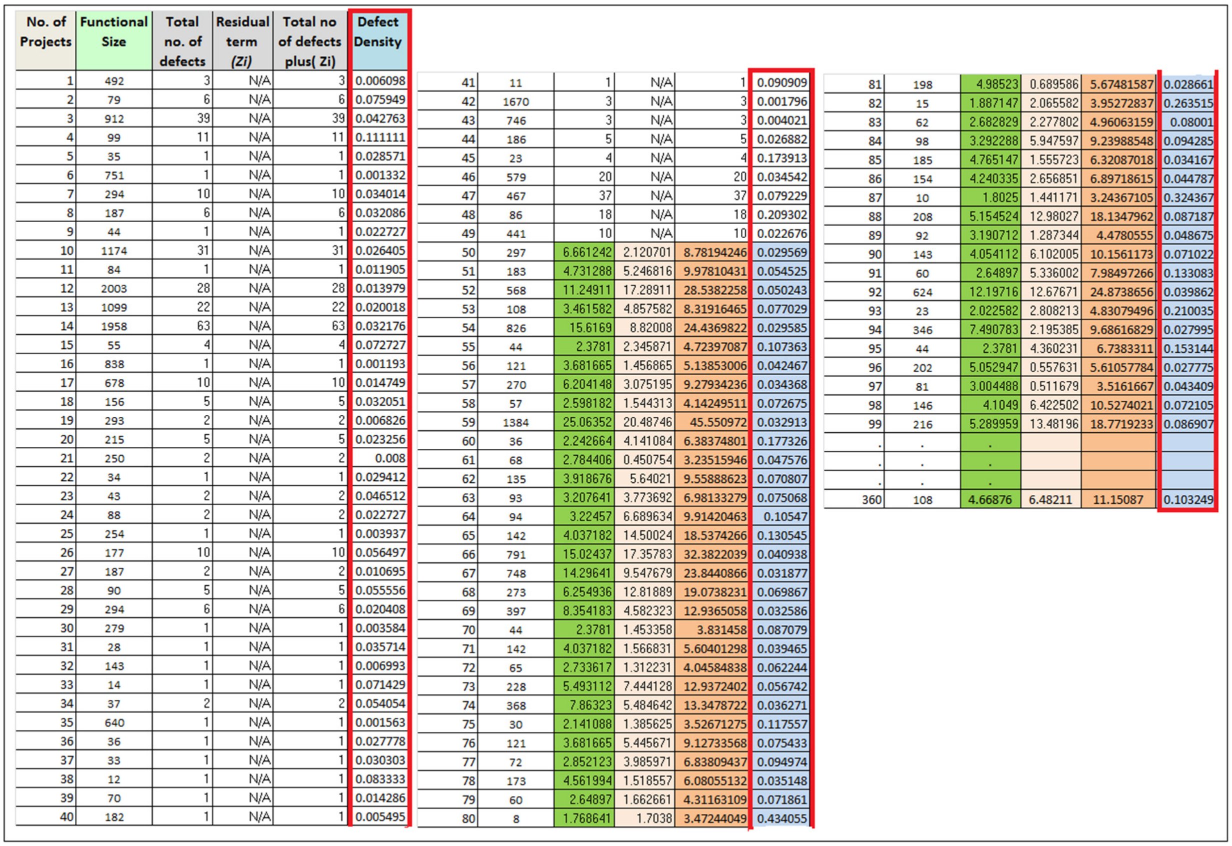
MMRE	**Pred(25)**
77%	50%

The MMRE is (77%) and the Pred(25) is 50% for the performance of the estimation model of TD based on CFP derived from the imputed dataset N=360 projects - see Table 6.2. The value of MMRE has increased after using Stochastic regression imputation technique, compared with the (31%) when the Regression imputation technique has used, which indicates that it restores that data variability with high value of the R^2.

Next, the values of the Defect Density (DD) are observed for the SRI imputed dataset N=360 projects (after imputing the TD missing data by SRI) that have blank fields of defect density caused by missing of TD values (the software size values are all recorded within the data set N=360) - see Figure 6.4. DD is to measure the TD with the 'Functional size' in CFP.

➢ Calculate the Defect Density after the imputation for the total defects, based on the formula:

$$\text{Defect Density} = \frac{Total\ Number\ of\ Defects}{Functional\ Size} \text{ , (ISBSG, 2013)}$$



No. of Projects	Functional Size	Total no. of defects	Residual term *(Zi)*	Total no of defects plus(Zi)	Defect Density
1	492	3	N/A	3	0.006098
2	79	6	N/A	6	0.075949
3	912	39	N/A	39	0.042763
4	99	11	N/A	11	0.111111
5	35	1	N/A	1	0.028571
6	751	1	N/A	1	0.001332
7	294	10	N/A	10	0.034014
8	187	6	N/A	6	0.032086
9	44	1	N/A	1	0.022727
10	1174	31	N/A	31	0.026405
11	84	1	N/A	1	0.011905
12	2003	28	N/A	28	0.013979
13	1099	22	N/A	22	0.020018
14	1958	63	N/A	63	0.032176
15	55	4	N/A	4	0.072727
16	838	1	N/A	1	0.001193
17	678	10	N/A	10	0.014749
18	156	5	N/A	5	0.032051
19	293	2	N/A	2	0.006826
20	215	5	N/A	5	0.023256
21	250	2	N/A	2	0.008
22	34	1	N/A	1	0.029412
23	43	2	N/A	2	0.046512
24	88	2	N/A	2	0.022727
25	254	1	N/A	1	0.003937
26	177	10	N/A	10	0.056497
27	187	2	N/A	2	0.010695
28	90	5	N/A	5	0.055556
29	294	6	N/A	6	0.020408
30	279	1	N/A	1	0.003584
31	28	1	N/A	1	0.035714
32	143	1	N/A	1	0.006993
33	14	1	N/A	1	0.071429
34	37	2	N/A	2	0.054054
35	640	1	N/A	1	0.001563
36	36	1	N/A	1	0.027778
37	33	1	N/A	1	0.030303
38	12	1	N/A	1	0.083333
39	70	1	N/A	1	0.014286
40	182	1	N/A	1	0.005495
41	11	1	N/A	1	0.090909
42	1670	3	N/A	3	0.001796
43	746	3	N/A	3	0.004021
44	186	5	N/A	5	0.026882
45	23	4	N/A	4	0.173913
46	579	20	N/A	20	0.034542
47	467	37	N/A	37	0.079229
48	86	18	N/A	18	0.209302
49	441	10	N/A	10	0.022676
50	297	6.661242	2.120701	8.78194246	0.029569
51	183	4.731288	5.246816	9.97810431	0.054525
52	568	11.24911	17.28911	28.5382258	0.050243
53	108	3.461582	4.857582	8.31916465	0.077029
54	826	15.6169	8.82008	24.4369822	0.029585
55	44	2.3781	2.345871	4.72397087	0.107363
56	121	3.681665	1.456865	5.13853006	0.042467
57	270	6.204148	3.075195	9.27934236	0.034368
58	57	2.598182	1.544313	4.14249511	0.072675
59	1384	25.06352	20.48746	45.550972	0.032913
60	36	2.242664	4.141084	6.38374801	0.177326
61	68	2.784406	0.450754	3.23515946	0.047576
62	135	3.918676	5.64021	9.55888623	0.070807
63	93	3.207641	3.773692	6.98133279	0.075068
64	94	3.22457	6.689634	9.91420463	0.10547
65	142	4.037182	14.50024	18.5374266	0.130545
66	791	15.02437	17.35783	32.3822039	0.040938
67	748	14.29641	9.547679	23.8440866	0.031877
68	273	6.254936	12.81889	19.0738231	0.069867
69	397	8.354183	4.582323	12.9365058	0.032586
70	44	2.3781	1.453358	3.831458	0.087079
71	142	4.037182	1.566831	5.60401298	0.039465
72	65	2.733617	1.312231	4.04584838	0.062244
73	228	5.493112	7.444128	12.9372402	0.056742
74	368	7.86323	5.484642	13.3478722	0.036271
75	30	2.141088	1.385625	3.52671275	0.117557
76	121	3.681665	5.445671	9.12733568	0.075433
77	72	2.852123	3.985971	6.83809437	0.094974
78	173	4.561994	1.518557	6.08055132	0.035148
79	60	2.64897	1.662661	4.31163109	0.071861
80	8	1.768641	1.7038	3.47244049	0.434055
81	198	4.98523	0.689586	5.67481587	0.028661
82	15	1.887147	2.065582	3.95272837	0.263515
83	62	2.682829	2.277802	4.96063159	0.08001
84	98	3.292288	5.947597	9.23988548	0.094285
85	185	4.765147	1.555723	6.32087018	0.034167
86	154	4.240335	2.656851	6.89718615	0.044787
87	10	1.8025	1.441171	3.24367105	0.324367
88	208	5.154524	12.98027	18.1347962	0.087187
89	92	3.190712	1.287344	4.4780555	0.048675
90	143	4.054112	6.102005	10.1561173	0.071022
91	60	2.64897	5.336002	7.98497266	0.133083
92	624	12.19716	12.67671	24.8738656	0.039862
93	23	2.022582	2.808213	4.83079496	0.210035
94	346	7.490783	2.195385	9.68616829	0.027995
95	44	2.3781	4.360231	6.7383311	0.153144
96	202	5.052947	0.557631	5.61057784	0.027775
97	81	3.004488	0.511679	3.5161667	0.043409
98	146	4.1049	6.422502	10.5274021	0.072105
99	216	5.289959	13.48196	18.7719233	0.086907
.	.	.			
.	.	.			
.	.	.			
360	108	4.66876	6.48211	11.15087	0.103249

Figure 6.4 An example of sample results of observed DD after imputing TD data points of by SRI, N=360 projects

Figure 6.5 shows the defect density of imputed SRI data set of N=360 software projects sized by COSMIC method, with a ranges from 0.0012 to 2.549 TD/CFP, with most values at the low end. The median is 0.0526 TD/CFP. The values of defect density peaks at 1.705 TD/CFP and 2.549 TD/CFP.

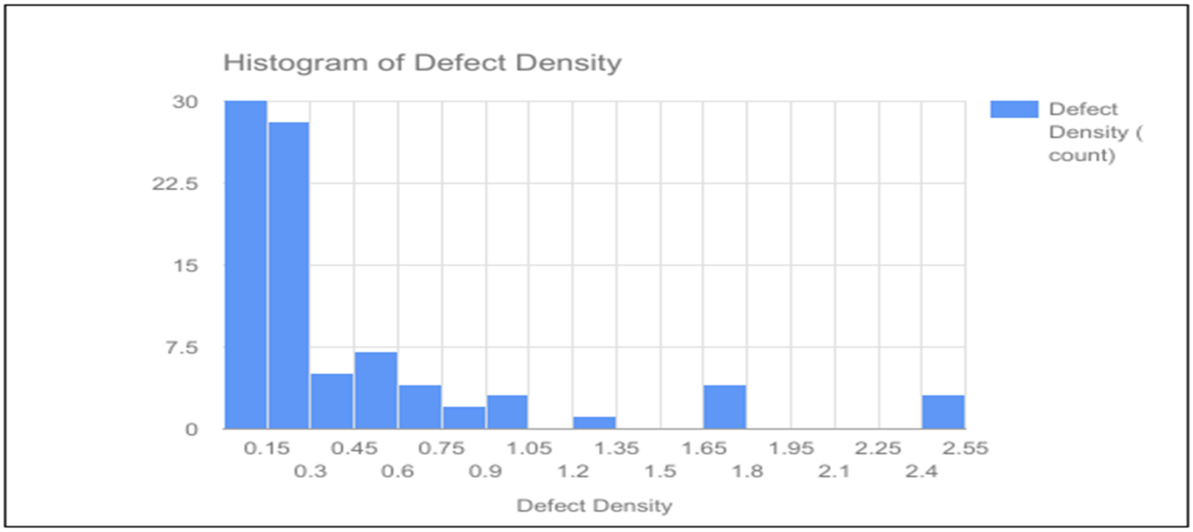


Figure 6.5 Defect density of SRI-imputed data set by SRI, N=360 projects

Table 6.3 presents the averages of the value imputed based on Stochastic regression imputation by fill-in missing values on the dependent variable, the Total Defects data filed in the data set N=360 projects.

Table 6.3 Average Total Number of Defects after SRI imputation N=360 projects

Variable	**Averages of TD** **N=360 projects**	
	Average complete data	**Average imputed data**
Total Defects	8	10.6
# Projects	49	311

6.3 Summary

The missing values from the ISBSG R12 N = 360 software projects sized by COSMIC were imputed by Stochastic regression imputation technique: estimated values generated from an TD estimation model of complete observations (N=49 projects) within the dependent variable of 'Total Number of Defects' based on an independent variable 'Functional Size' in

CFP with a residual term added (based on, available data within the variable with missing values).

The obtained results were statistically significant at t-test and P-values of 'Total Number of Defects' estimates. The coefficients of determination (R^2) was 0.7 for the TD estimation model after the imputation by the Stochastic regression imputation technique, the confidence interval does contain true parameters for the estimates as well as the standard error and the MMRE (77%) has increased compared with the (31%) of MMRE when the Regression imputation technique was used. Based on that, using stochastic regression imputation technique for the analysis provided a better solution for the problem of missing data where it performed better than the previous technique 'single imputation technique', and 'regression imputation technique' upon this research analysis, where, it corrected and restored the lack of variability, and made the obtained results more unbiased compared with two previously studied imputation techniques (SI and RI).

CHAPTER 7

VERFICATION STRATEGY FOR THE IMPUTATION TECHNIQUES

7.1 Introduction

This chapter presents phase 3 of the research methodology by verifying the contribution of the three previously implemented imputation techniques on defect estimation.

Therefore, this chapter attempts to measure the predictive accuracy of the defect estimation models (based on the independent variable 'Functional Size' in CFP) obtained from complete dataset and imputed datasets. This involves developing a verification strategy for analyzing the defect estimation models results to verify the impact of the independent variable 'Functional Size' on the parameter estimates of the dependent variable 'Total Number of Defects'.

7.2 Verification strategy: creating artificially missing values from a complete dataset

This strategy for analyzing the performance of the three imputation techniques used in the empirical studies involves to work with a dataset of complete data set (e.g., it does not contain any missing value: here N=49 projects) through creating artificially a subset by deleting the data values within the intended variable, and next, impute these artificially missing data by the selected imputation techniques, and next to generate estimation models from the original complete data set and the other imputed data sets, in order to compare and assess the estimates derived from these estimation models through evaluation criteria of such statistical models, such as: Mean Magnitude of Relative Error (MMRE).

In this section, the dataset selected consists of the 49 projects with complete data points. The specific verification strategy adopted in this research consists of:

- Given the complete data N=49 projects, randomly split the data set into two subsets X and Y.

- From subset Y, delete the data values for the Total Number of Defects data fields,
- Use *Single Imputation* (SI) technique: based on absolute seeds (min, max of subset X) for the missing values of Total Number of Defects (TD) within subset Y.
- Use *Regression Imputation* (RI) technique based on replacing each missing value with a predicted value based on estimation model built using complete observations of Total Number of Defects (TD), trained by the independent variable 'Functional Size'.
- Use *Stochastic Regression Imputation* (SRI) technique based on replacing each missing value with a predicted value based on estimation model built using complete observations of Total Number of Defects (TD), trained by the independent variable 'Functional Size' with a residual term added to the predicted values.
- Defect estimation models (based on the independent variable 'Functional Size' in CFP) will be built with both the initial complete data set N=49 projects and the imputed datasets of N=49 projects.
- Compare the TD estimate by assessing and comparing the predictability with MMRE and Pred(25) based on the following criteria (Conte, Dunsmore et Shen, 1986) and (Abran, 2010):

  (1) Magnitude of Relative Error (MRE) = | Estimated value – Actual value | / Actual

  (2) Mean Magnitude of Relative Error for n projects (MMRE) = 1/n*Σ(MREi)

  (3) Measure of Prediction Quality = Pred(x/100)

Figure 7.1 illustrates a specific strategy for investigating the contribution of SI, RI, and SRI on TD estimation (based on the independent variable 'Functional Size' in CFP) using dataset N=49 software projects, and then compares the results of the TD estimation models:

- ➢ Given the complete dataset (N= 49 projects without missing values), split it randomly into two subsets X, and Y: N=26 projects, N=23 projects, respectively.
- ➢ Delete the TD values from subset Y, N=23 projects (with artificial missing values).
- ➢ Combine the two subsets X and Y, N=49 projects (including the artificial missing values).

- Given the combined X and Y subsets (N=49 projects), apply Single Imputation procedure based on absolute seeds (min, max) for the missing values of Total Number of Defects (TD) in subset Y (Select Seeds Min & Max from subset X).
- Given the combined X and Y subsets (N=49 projects), apply Regression Imputation procedure: replace the missing values with the predicted values from a linear regression equation based on complete N= 23 projects of TD in basis of 'Functional Size' in CFP as an independent variable.
- Given the combined X and Y subsets (N=49 projects), apply Stochastic Regression Imputation procedure: replace missing values with the predicted values from the same linear TD estimation equation based on complete N= 23 projects (in basis of 'Functional Size' in CFP). Then add a residual term to the predicted values.
- Build linear regression analysis models to estimate TD based on variable 'Functional Size', for the initial complete dataset N=49 projects, and for all the imputed datasets, N=49 projects.
- Analyze the TD estimation with MMRE and Pred(25) to assess the predictability, N=49 projects.
- Compare the results of the MMRE and Pred(25) of the dataset with complete values N=49 projects, with all the imputed datasets (by SI, RI, and SRI) N=49 projects.

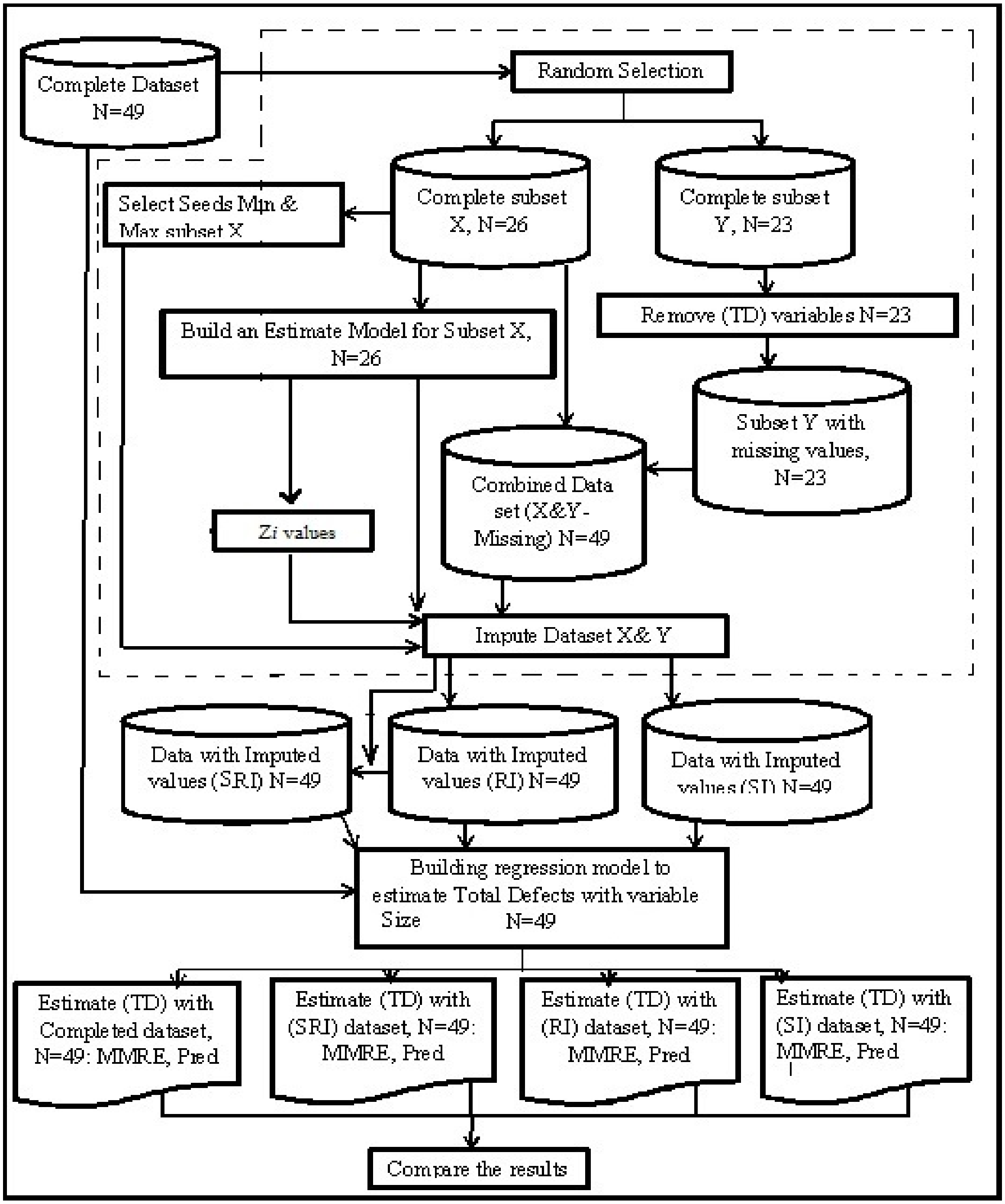
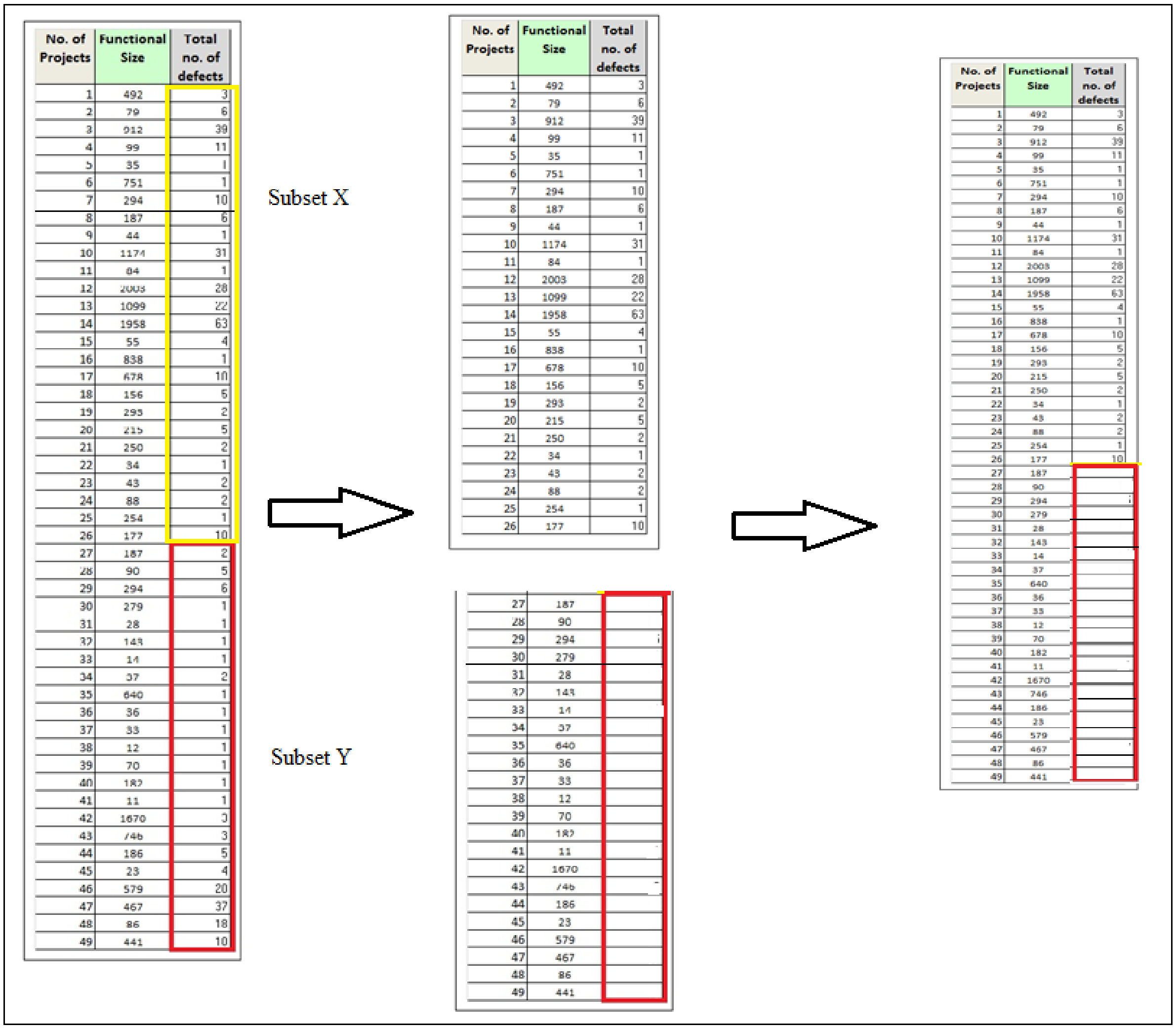

Complete Dataset N=49
Random Selection
Select Seeds Min & Max subset X
Complete subset X, N=26
Complete subset Y, N=23
Remove (TD) variables N=23
Build an Estimate Model for Subset X, N=26
Subset Y with missing values, N=23
Combined Data set (X&Y-Missing) N=49
Zi values
Impute Dataset X& Y
Data with Imputed values (SRI) N=49
Data with Imputed values (RI) N=49
Data with Imputed values (SI) N=49
Building regression model to estimate Total Defects with variable Size N=49
Estimate (TD) with Completed dataset, N=49: MMRE, Pred
Estimate (TD) with (SRI) dataset, N=49: MMRE, Pred
Estimate (TD) with (RI) dataset, N=49: MMRE, Pred
Estimate (TD) with (SI) dataset, N=49: MMRE, Pred
Compare the results

Figure 7.1 A strategy for analyzing the predictive accuracy of TD estimation models using SI, RI, and SRI imputed datasets

7.3 Dataset preparation for verification strategy analysis: artificially initiate subset with missing data

This section involves to randomly split the complete data set N=49 projects into two subsets X and Y, to be X=26 projects, and Y=23 projects, delete the data points of total number of defects within subset X, and then re-combine the two subsets X, and Y to be as one data set N=49 projects again, in order to be ready for imputation procedures by the selected imputation techniques - see Figure 7.2.


Subset X
Subset Y

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	2
28	90	5
29	294	6
30	279	1
31	28	1
32	143	1
33	14	1
34	37	2
35	640	1
36	36	1
37	33	1
38	12	1
39	70	1
40	182	1
41	11	1
42	1670	3
43	746	3
44	186	5
45	23	4
46	579	20
47	467	37
48	86	18
49	441	10

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10

27	187	
28	90	
29	294	
30	279	
31	28	
32	143	
33	14	
34	37	
35	640	
36	36	
37	33	
38	12	
39	70	
40	182	
41	11	
42	1670	
43	746	
44	186	
45	23	
46	579	
47	467	
48	86	
49	441	

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	
28	90	
29	294	
30	279	
31	28	
32	143	
33	14	
34	37	
35	640	
36	36	
37	33	
38	12	
39	70	
40	182	
41	11	
42	1670	
43	746	
44	186	
45	23	
46	579	
47	467	
48	86	
49	441	

Figure 7.2 An example of complete data set N=49 projects

7.4 Verification analysis for original complete data set N=49 projects

The complete data set N=49 software projects is to be used for a linear regression analysis modeling to estimate the 'Total Number of variable' variable along with the independent variable 'Functional Size', for the initial complete dataset N=49 projects (this procedure is already accomplished previously in chapter. 5). See Table 7.1.

Table 7.1 Regression parameter analysis and statistical tests for TD estimation model of 'Total Number of Defects' and 'Functional Size' based on the complete dataset, N=49 projects

Variable	Intercept	Coefficient	R2	95% Confidence Limits		T-test	Standard Error	P-value
Functional Size	1.63	0.017	0.5	0.0113	0.0225	6.1	0.0028	1.95801E-07

Therefore, the TD estimation model based on the complete of 'Functional Size' in CFP, N=49 software projects is: (*Total Number of Defects* = 1.63 + 0.017 * *Functional Size*) which to be used with comparing and assessment of 'Total Number of Defects' estimates throughout the evaluation criteria: MMRE, and Pred(25) as a part of the verification process. Table 7.1 also shows the coefficients of determination (R^2) for the TD estimation model based on 'Functional Size in CFP' is 0.5. The confidence interval is (LL is 0.0113, and UL is 0.0225), with statistical significant P-value and T-test.

From Table 7.2 it can be observed that: the MMRE is 167% and the Pred(25) is 21% for TD estimation model based on 'Functional Size' that is derived from the complete dataset N=49 projects (with no missing data with its variables: 'Total Number of Defects' and 'Functional Size').

Table 7.2 MMRE and Pred(25) for TD estimation model based on the complete dataset, N=49 projects

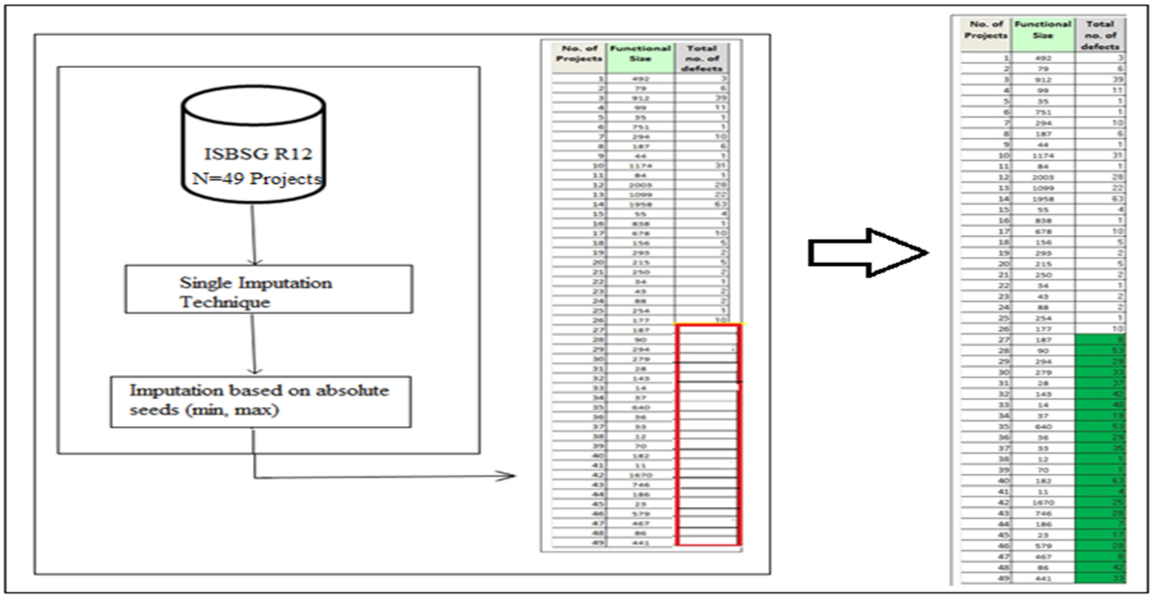
MMRE	Pred(25)
167%	21%

7.5 Verification analysis for imputed data set of N=49 software projects by Single imputation technique

From the dataset prepared in Section 7.3, N=49 projects - see Figure 7.2: this section shows the verification analysis of using Single imputation technique based on ‘imputes’ obtained from absolute seeds (min, max) within the reported data points in ‘Total Number of Defects’ variable (sub-dataset X, N =26 software projects) Therefore, this is applied on the artificially missing total defects values of sub-dataset Y, N=23 software projects - see Figure 7.3.

Then, a linear regression analysis is applied on total defects, and software size, followed by an analysis of the MMRE and Pred (based on the identified statistical analysis in step (2) - in Figure 3.11) on the SI-imputed dataset (X and Y) of N=49 software projects for comparison and assessment of the estimation model for the imputed dataset.

- ➢ In this step (Figure 7.3), the artificially missing values in subset Y of 23 projects are imputed by the random numbers generated from the min-max from the subset X of 26. The minimum is 1 TD and the maximum is 63 TD.


ISBSG R12
N=49 Projects
Single Imputation Technique
Imputation based on absolute seeds (min, max)

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	
28	90	
29	294	
30	279	
31	28	
32	143	
33	14	
34	37	
35	640	
36	36	
37	33	
38	12	
39	70	
40	182	
41	11	
42	1670	
43	746	
44	186	
45	23	
46	579	
47	467	
48	86	
49	441	

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	8
28	90	53
29	294	29
30	279	33
31	28	37
32	143	42
33	14	40
34	37	19
35	640	53
36	36	29
37	33	35
38	12	1
39	70	1
40	182	63
41	11	4
42	1670	25
43	746	28
44	186	7
45	23	17
46	579	28
47	467	8
48	86	42
49	441	33

Figure 7.3 Example of sample results for using the single imputation technique on dataset N=49 using the seeds from the subset X of 26 projects

To verify the impact of the independent variable 'Functional Size' on TD parameter estimates, a linear regression analysis is carried out: based on the regression analysis results in Table 7.3, the p-value and the t-statistic are statistically significant.

Table 7.3 Regression parameter analysis and statistical tests for TD estimation model of 'Total Number of Defects' and 'Functional Size' based on the SI-imputed dataset (X and Y), N=49 projects

Variable	Intercept	Coefficient	R2	95% Confidence Limits		T-test	Standard Error	P-value
Functional size	13.91	0.0119	0.1033	0.0016	0.0223	2.327	0.0051	0.0243

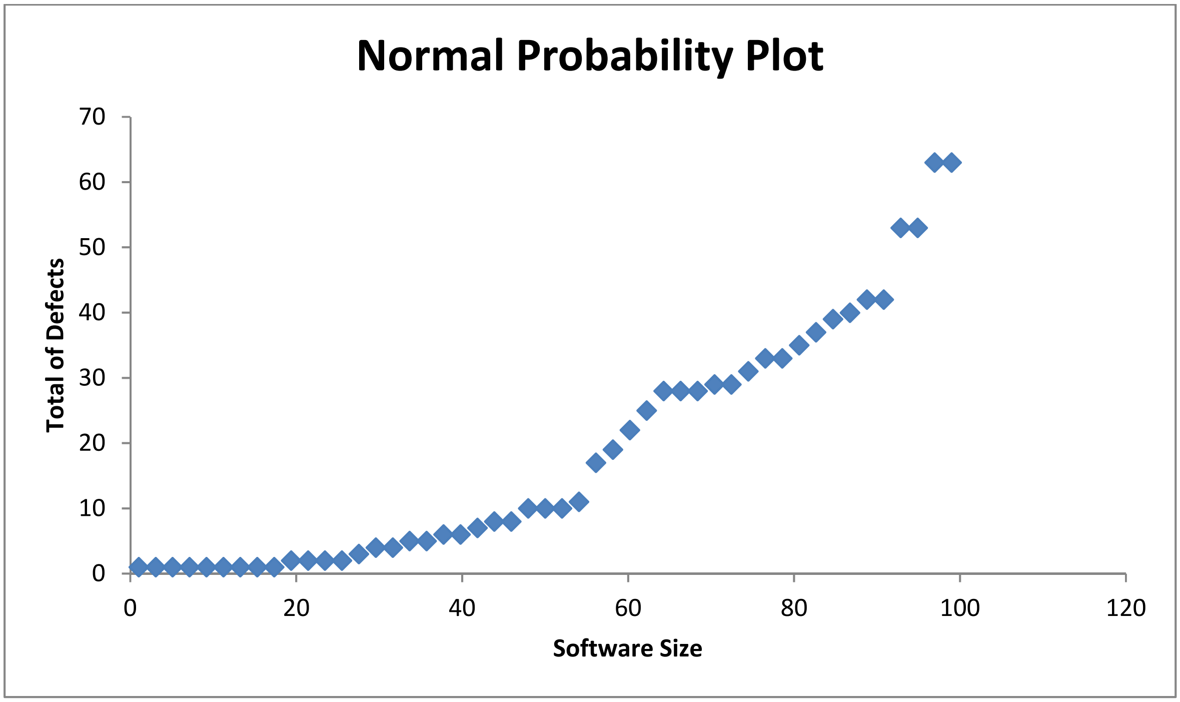


Figure 7.4 Graphical representation of total defects based on functional Size using the SI-imputed dataset (X and Y) N=49 projects

Table 7.3 also shows the parameter estimates for the variable 'Total Number of Defects' model are: (constant = 13.91 TD and 0.0119 TD/CFP). Therefore, the TD estimation model based on independent variable 'Functional Size' in CFP using the imputed dataset (X and Y) of N=49 projects:

Total number of Defects = 13.91+ 0.0119 * Functional Size

Table 7.3 also shows the coefficients of determination (R^2) for the TD estimation model (obtained dataset N=49 imputed by Single imputation technique) is very low at 0.1033. The confidence interval is (LL is 0.0016, and UL is 0.0223).

Table 7.4 Summary of the statistical tests analysis for TD estimation model of SI-imputed dataset N=49 projects

Variable	**Dataset (X and Y) N=49 projects**	
	Significant **T-test**	**Significant** **P-values**
Functional Size	Yes	Yes

The TD estimates are compared by assessing and comparing the predictability with MMRE and Pred(25). Form Table 7.5, it can be observed that:

- The MMRE is 291% which is very high and the Pred(25) is 18% for assessing the TD estimation model (based on 'Functional Size' in CFP) derived from the imputed dataset (X and Y), N=49 software projects.

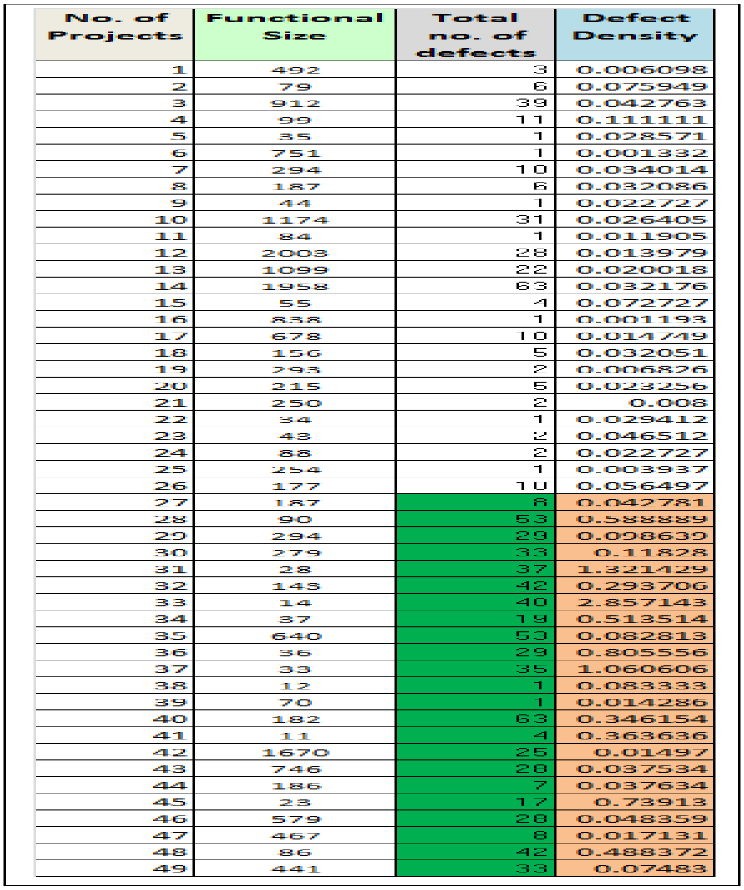
Table 7.5 MMRE and Pred(25) for TD estimation model
based on the SI-imputed dataset (X and Y) , N=49 projects

MMRE	**Pred(25)**
291%	18%

Next, the values of the Defect Density (DD) are observed for the imputed data N=49 projects (after imputing the TD missing data by SI) that have blank fields of defect density caused by missing of TD values (the software size values are all recorded within the dataset N=49 projects) - see Figure 7.5. DD is to measure the TD with the Functional size in CFP. The 'Defect Density' has a range of values from 0.0012 TD/CFP to 2.857143 TD/CFP. The median is 0.0376 TD/CFP.

- Calculate the Defect Density (DD) after the imputation for the total defects, based on the formula:

$$\text{Defect Density} = \frac{\mathit{Total\ Number\ of\ Defects}}{\mathit{Functional\ Size}} \text{ , (ISBSG, 2013)}$$



No. of Projects	Functional Size	Total no. of defects	Defect Density
1	492	3	0.006098
2	79	6	0.075949
3	912	39	0.042763
4	99	11	0.111111
5	35	1	0.028571
6	751	1	0.001332
7	294	10	0.034014
8	187	6	0.032086
9	44	1	0.022727
10	1174	31	0.026405
11	84	1	0.011905
12	2003	28	0.013979
13	1099	22	0.020018
14	1958	63	0.032176
15	55	4	0.072727
16	838	1	0.001193
17	678	10	0.014749
18	156	5	0.032051
19	293	2	0.006826
20	215	5	0.023256
21	250	2	0.008
22	34	1	0.029412
23	43	2	0.046512
24	88	2	0.022727
25	254	1	0.003937
26	177	10	0.056497
27	187	8	0.042781
28	90	53	0.588889
29	294	29	0.098639
30	279	33	0.11828
31	28	37	1.321429
32	143	42	0.293706
33	14	40	2.857143
34	37	19	0.513514
35	640	53	0.082813
36	36	29	0.805556
37	33	35	1.060606
38	12	1	0.083333
39	70	1	0.014286
40	182	63	0.346154
41	11	4	0.363636
42	1670	25	0.01497
43	746	28	0.037534
44	186	7	0.037634
45	23	17	0.73913
46	579	28	0.048359
47	467	8	0.017131
48	86	42	0.488372
49	441	33	0.07483

Figure 7.5 An example of sample results of the observed DD using SI-imputed dataset N=49 projects

Summary

The obtained results were statistically significant at t-test and P-values for the 'Total Number of Defects' estimates. The coefficients of determination (R^2) was 0.1033 for the TD estimation model after using the Single imputation technique, the confidence interval did contain true parameters for the estimates, as well as the standard error was increased by fractions compared the standard error of the complete dataset as shown previously in Table 7.1 and 7.3; the artificially missing data' percentage was 50% of the data. Based on that, using single imputation technique for the analysis may provide a solution for the problem of missing data where it performed better when the percentage of the missing data was more than 50% of the data set upon this research analysis of defect estimation from projects sized by COSMIC method. However, this is not an indication that single imputation preforms the best solution for missing data upon this analysis, whereas, the MMRE is very high.

7.6 Verification analysis for imputed data set of N=49 software projects by Regression imputation technique

From the dataset prepared in Section 7.3, N=49 projects - see Figure 7.2: this section shows the verification analysis of using Regression imputation technique that uses an estimation model from variable 'Total Number of Defects' and independent variable 'Functional Size', which is built using the complete subset X dataset N=26, in order to perform the imputation on the missing TD values of the subset Y of missing TD (N=23 projects).

A linear regression analysis is applied on total defects, and software size, (based on the identified statistical analysis in step (2) - in Figure 3.11) on subset X, N=26 software projects. Based on the regression analysis results in Table 7.6, the p-value and the t-statistic are statistically significant.

Table 7.6 Regression parameters analysis and statistical tests for TD estimation model of 'Total Number of Defects' and 'Functional Size' based on the subset X (N=26 projects)

Variable	Intercept	Coefficient	R2	95% Confidence Limits		T- test	Standard Error	P-value
Functional size	0.13	0.022	0.66	0.0150	0.0280	6.82	0.0032	4.7051E-07

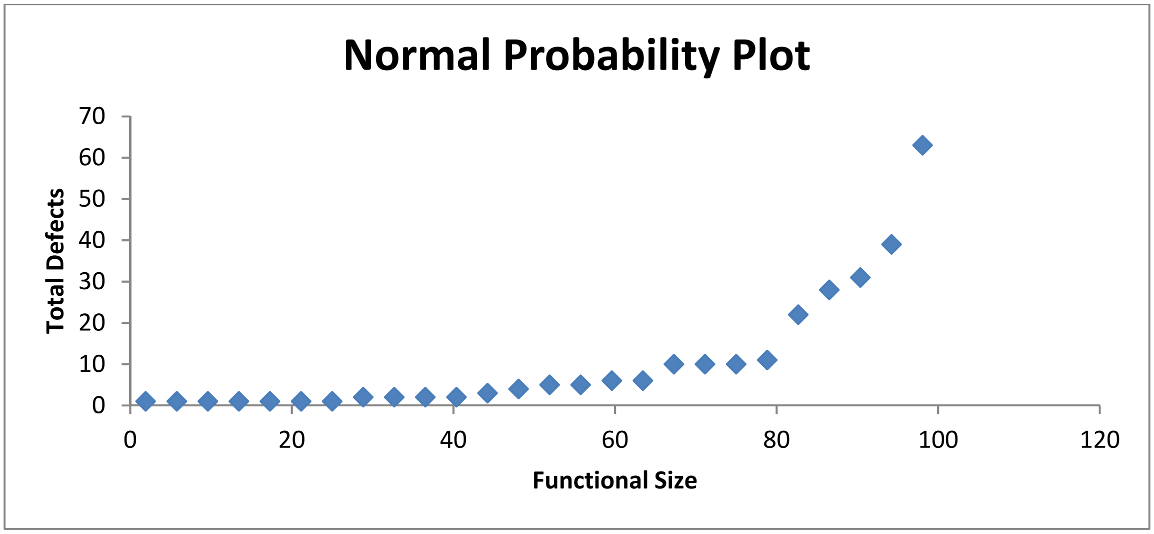


Figure 7.6 Graphical representation of 'Total Number of Defects' and 'Functional Size' based on the subset X, N=26 projects

Table 7.7 Summary of statistical tests for TD estimation model using subset X=26 projects

Variable	**Subset X Dataset N=26 projects**	
	Significant **T-test**	**Significant** **P-values**
Functional Size	Yes	Yes

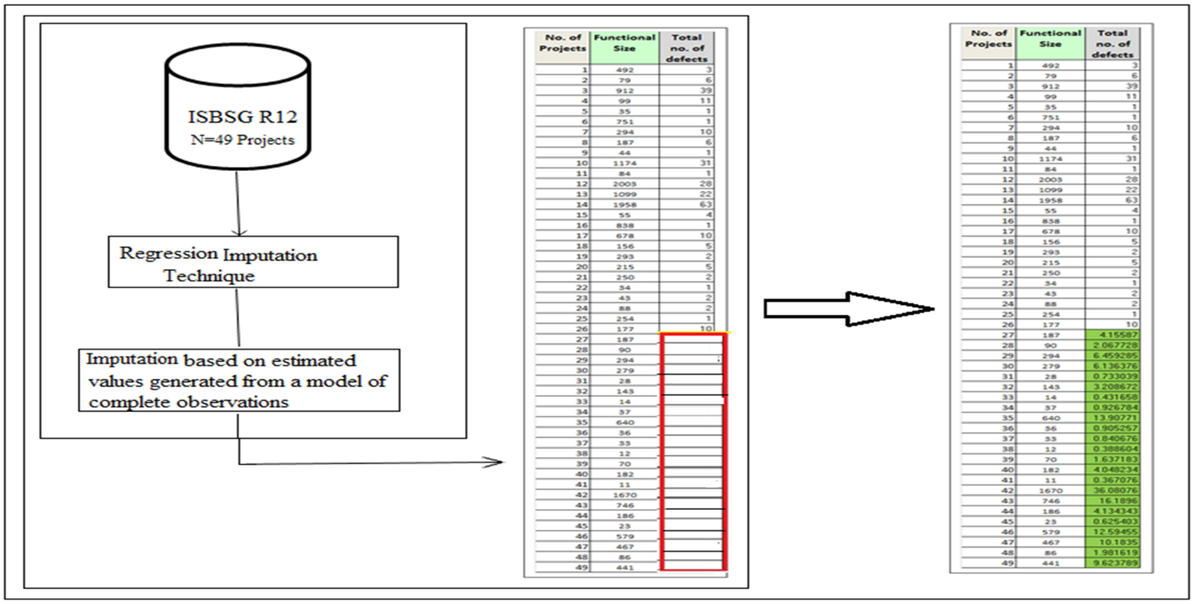
Table 7.6 also shows the parameter estimates for the 'Total Number of Defects' model are: (constant = 0.13 TD and 0.022 TD/CFP). Therefore, the TD estimation model based on independent variable 'Functional Size' in CFP from the subset X (N=26 projects) dataset is:

Total number of defects = 0.13+ 0.022* Functional Size

Therefore, this is applied on the artificially 'Total Number of Defects' missing data of sub-data set Y, N=23 software projects - see Figure 7.7.

Next, a linear regression analysis (based on the identified statistical analysis in step (2) - in Figure 3.11) is applied on total defects, and software size, followed by an analysis of the

MMRE and Pred on the RI-imputed dataset (X and Y) of N=49 software projects for comparison and assessment of the obtained TD estimation models.


ISBSG R12
N=49 Projects
Regression Imputation Technique
Imputation based on estimated values generated from a model of complete observations

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	
28	90	
29	294	
30	279	
31	28	
32	143	
33	14	
34	37	
35	640	
36	36	
37	33	
38	12	
39	70	
40	182	
41	11	
42	1670	
43	746	
44	186	
45	23	
46	579	
47	467	
48	86	
49	441	

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	4.15587
28	90	2.067728
29	294	6.459285
30	279	6.136376
31	28	0.733039
32	143	3.208672
33	14	0.431658
34	37	0.926784
35	640	13.90771
36	36	0.905257
37	33	0.840676
38	12	0.388604
39	70	1.637183
40	182	4.048234
41	11	0.367076
42	1670	36.08076
43	746	16.1896
44	186	4.134343
45	23	0.625403
46	579	12.59455
47	467	10.1835
48	86	1.981619
49	441	9.623789

Figure 7.7 Example of sample results of regression imputation technique on imputed data set N=49 of software projects

To verify the impact of the independent variable 'Functional Size' on TD parameter estimates, a linear regression analysis is carried out: based on the regression analysis results in Table 7.8, the p-value and the t-statistic are statistically significant.

Table 7.8 Regression parameters analysis and statistical tests for TD estimation model of 'Total Number of Defects' and 'Functional Size' based on the RI-imputed dataset (X and Y) N=49 projects

Variable	**Intercept**	**Coefficient**	**R2**	**95% Confidence Limits**		**T-test**	**Standard Error**	**P-value**
Functional size	0.13	0.022	0.74	0.0177	0.0252	11.52	0.0019	2.742E-15

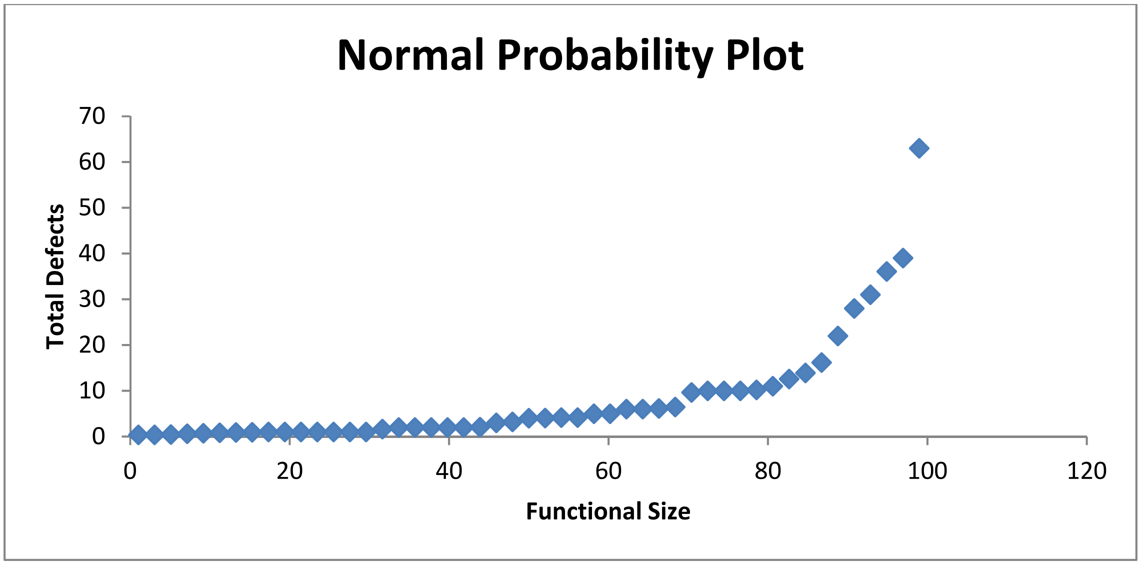


Figure 7.8 Graphical representation of Total Defects based on Functional Size using RI-imputed dataset (X and Y) N=49 projects

Table 7.8 also shows the parameter estimates for the 'Total Number of Defects' estimation model: (constant = 13.91 TD and 0.0119 TD/CFP), based on the imputed dataset by Regression imputation technique (X and Y), N=49 projects:

Total number of defects = 0.13 + 0.022 * Functional Size

Table 7.8 also shows that the coefficients of determination (R^2) is (0.74) for the TD estimation model (obtained dataset N=49 imputed by Regression imputation technique), which it has increased. The confidence interval is (LL is 0.0177, and UL is 0.0252).

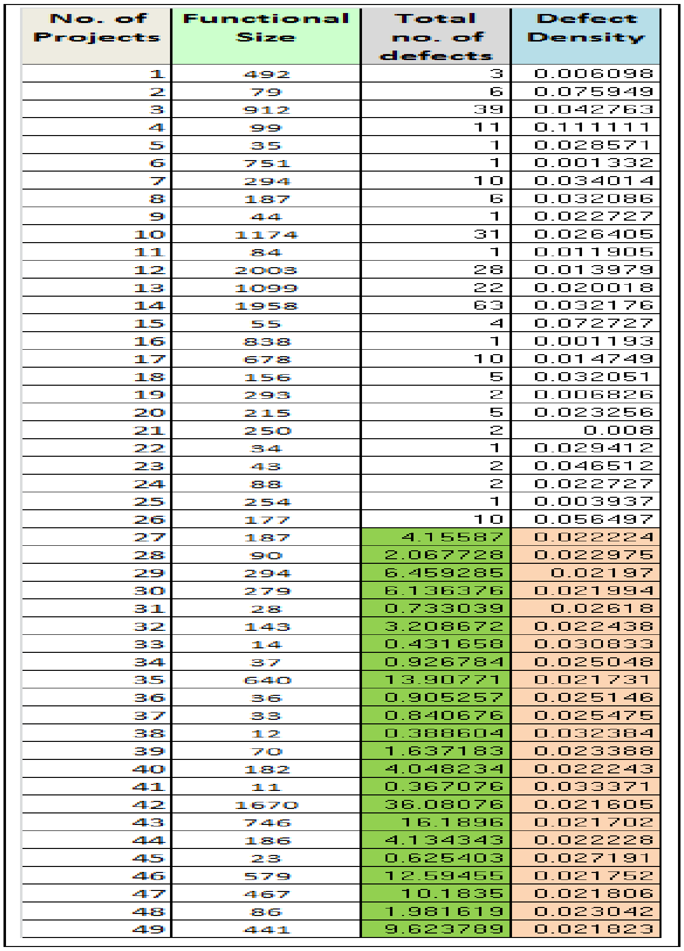
Table 7.9 Summary of the statistical tests for TD estimation model of RI-imputed dataset N=49 projects

Variable	**Dataset (X and Y) N=49 projects**	
	Significant **T-test**	**Significant** **P-values**
Size	Yes	Yes

Next, the values of the Defect Density (DD) are observed for the imputed data N=49 projects (after imputing the TD missing data by RI) that have blank fields of defect density caused by missing of TD values (the software size values are all recorded within the dataset N=49 projects) - see Figure 7.9. DD is to measure the TD with the Functional size in CFP. The 'Defect Density' has a range of values from 0.0012 TD/CFP to 0.1111 TD/CFP. The median is 0.023 TD/CFP.

- ➢ Calculate the Defect Density (DD) after the imputation for the total defects, based on the formula:

$$\text{Defect Density} = \frac{Total\ Number\ of\ Defects}{Functional\ Size}\ \text{, (ISBSG, 2013)}$$



No. of Projects	Functional Size	Total no. of defects	Defect Density
1	492	3	0.006098
2	79	6	0.075949
3	912	39	0.042763
4	99	11	0.111111
5	35	1	0.028571
6	751	1	0.001332
7	294	10	0.034014
8	187	6	0.032086
9	44	1	0.022727
10	1174	31	0.026405
11	84	1	0.011905
12	2003	28	0.013979
13	1099	22	0.020018
14	1958	63	0.032176
15	55	4	0.072727
16	838	1	0.001193
17	678	10	0.014749
18	156	5	0.032051
19	293	2	0.006826
20	215	5	0.023256
21	250	2	0.008
22	34	1	0.029412
23	43	2	0.046512
24	88	2	0.022727
25	254	1	0.003937
26	177	10	0.056497
27	187	4.15587	0.022224
28	90	2.067728	0.022975
29	294	6.459285	0.02197
30	279	6.136376	0.021994
31	28	0.733039	0.02618
32	143	3.208672	0.022438
33	14	0.431658	0.030833
34	37	0.926784	0.025048
35	640	13.90771	0.021731
36	36	0.905257	0.025146
37	33	0.840676	0.025475
38	12	0.388604	0.032384
39	70	1.637183	0.023388
40	182	4.048234	0.022243
41	11	0.367076	0.033371
42	1670	36.08076	0.021605
43	746	16.1896	0.021702
44	186	4.134343	0.022228
45	23	0.625403	0.027191
46	579	12.59455	0.021752
47	467	10.1835	0.021806
48	86	1.981619	0.023042
49	441	9.623789	0.021823

Figure 7.9 An example of sample results of observed DD using the RI-imputed dataset N=49 projects

Estimates are compared by assessing and comparing the predictability with MMRE and Pred(25). Form Table 7.10, it can be observed that:

- The MMRE is (124%), which it has decreased compared with the MMRE (291%) when the Single imputation technique has been used in section 7.5 and the Pred(25) is 30% for assessing the TD estimation model (based on ‘Functional Size’ in CFP) derived from the imputed dataset (X and Y), N=49 software projects.

Table 7.10 MMRE and Pred(25) for TD estimation model
based on the RI-imputed dataset N=49 projects

MMRE	**Pred(25)**
124%	30%

Summary

The obtained results were statistically significant at t-test and P-values for the ‘Total Number of Defects’ estimates. The increased coefficients of determination (R^2) was 0.74 for the TD estimation model after using the Regression imputation technique, the confidence interval did contain true parameters for the estimates, as well as the Standard Error was decreased by fractions as shown in Table 7.8. The artificially missing data’ percentage was 50% of the data. Based on that, using Regression imputation technique for the analysis did provide a solution for the problem of missing data where it performed better than single imputation when the percentage of missing data is 50% of the data set upon this research analysis of defect estimation from projects sized by COSMIC method.

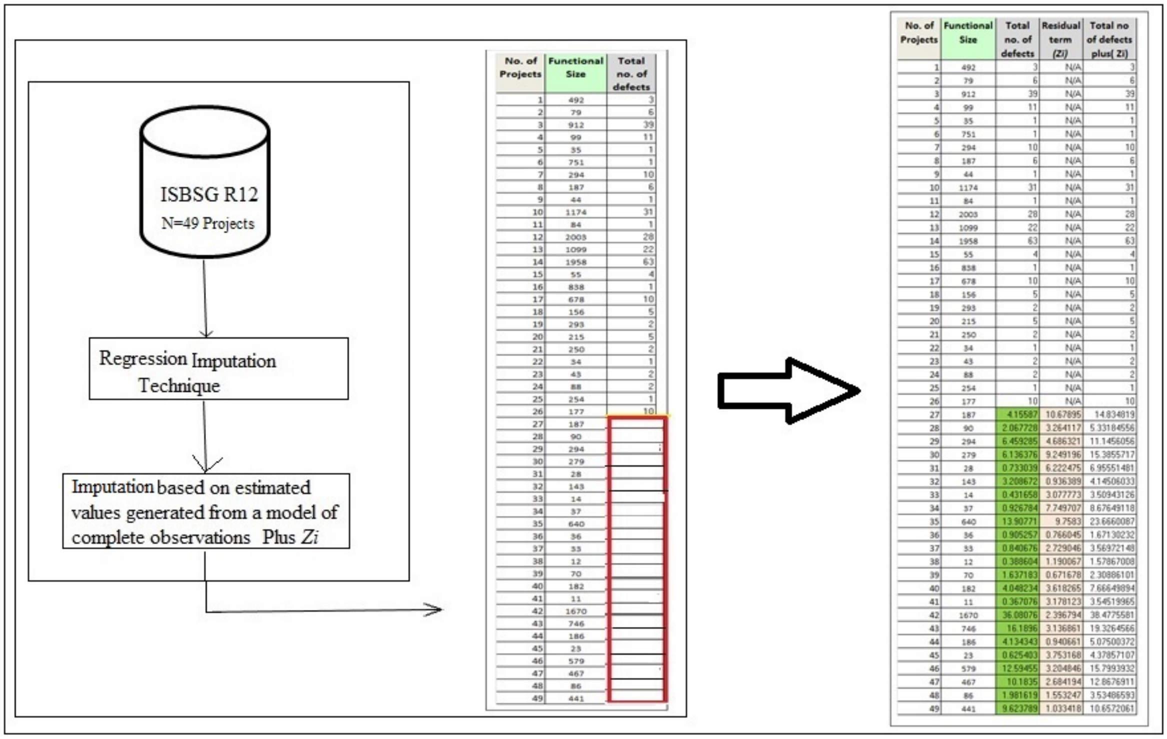
However, this is a good indication that regression imputation technique preforms a much better solution for missing data upon this analysis. Whereas, the MMRE was less than the MMRE value with complete data set, it is close to it: however, its MMRE value was also very good, which gives an obvious indication that even through the regression performs very well; it might underestimate or lack the data variation. Thus, the need to explore the stochastic regression imputation to clarify the case is important.

7.7 Verification analysis for imputed data set of N=49 software projects by Stochastic regression imputation technique

From the dataset prepared in Section 7.3, N=49 projects - see Figure 7.2: this section presents the verification analysis of using Stochastic regression imputation technique based on the similar imputation steps of the Regression imputation technique, and with a residual term (z_i) added to the predicted values obtained from TD estimation model of 'Total Number of Defects' and independent variable 'Functional Size' using the complete subset X, of N=26 software projects (this procedure is already done in section 7.6). The TD estimation model to generate the 'imputes' is:

$$\text{Total Number of Defects} = 0.13 + 0.022 * \text{Functional Size} + z_i$$

The next step is to add a residual term to the predicted values - see Figure 7.10.


ISBSG R12
N=49 Projects
Regression Imputation Technique
Imputation based on estimated values generated from a model of complete observations Plus *Zi*

No. of Projects	Functional Size	Total no. of defects
1	492	3
2	79	6
3	912	39
4	99	11
5	35	1
6	751	1
7	294	10
8	187	6
9	44	1
10	1174	31
11	84	1
12	2003	28
13	1099	22
14	1958	63
15	55	4
16	838	1
17	678	10
18	156	5
19	293	2
20	215	5
21	250	2
22	34	1
23	43	2
24	88	2
25	254	1
26	177	10
27	187	
28	90	
29	294	
30	279	
31	28	
32	143	
33	14	
34	37	
35	640	
36	36	
37	33	
38	12	
39	70	
40	182	
41	11	
42	1670	
43	746	
44	186	
45	23	
46	579	
47	467	
48	86	
49	441	

No. of Projects	Functional Size	Total no. of defects	Residual term *(Zi)*	Total no of defects plus(Zi)
1	492	3	N/A	3
2	79	6	N/A	6
3	912	39	N/A	39
4	99	11	N/A	11
5	35	1	N/A	1
6	751	1	N/A	1
7	294	10	N/A	10
8	187	6	N/A	6
9	44	1	N/A	1
10	1174	31	N/A	31
11	84	1	N/A	1
12	2003	28	N/A	28
13	1099	22	N/A	22
14	1958	63	N/A	63
15	55	4	N/A	4
16	838	1	N/A	1
17	678	10	N/A	10
18	156	5	N/A	5
19	293	2	N/A	2
20	215	5	N/A	5
21	250	2	N/A	2
22	34	1	N/A	1
23	43	2	N/A	2
24	88	2	N/A	2
25	254	1	N/A	1
26	177	10	N/A	10
27	187	4.15587	10.67895	14.834819
28	90	2.067728	3.264117	5.33184556
29	294	6.459285	4.686321	11.1456056
30	279	6.136376	9.249196	15.3855717
31	28	0.733039	6.222475	6.95551481
32	143	3.208672	0.936389	4.14506033
33	14	0.431658	3.077773	3.50943126
34	37	0.926784	7.749707	8.67649118
35	640	13.90771	9.7583	23.6660087
36	36	0.905257	0.766045	1.67130232
37	33	0.840676	2.729046	3.56972148
38	12	0.388604	1.190067	1.57867008
39	70	1.637183	0.671678	2.30886101
40	182	4.048234	3.618265	7.66649894
41	11	0.367076	3.178123	3.54519965
42	1670	36.08076	2.396794	38.4775581
43	746	16.1896	3.136861	19.3264566
44	186	4.134343	0.940661	5.07500372
45	23	0.625403	3.753168	4.37857107
46	579	12.59455	3.204846	15.7993932
47	467	10.1835	2.684194	12.8676911
48	86	1.981619	1.553247	3.53486593
49	441	9.623789	1.033418	10.6572061

Figure 7.10 Example of sample results of stochastic regression imputation technique on imputed data set N=49 of software projects

To verify the impact of the independent variable ‘Functional Size’ on TD parameter estimates, a linear regression analysis is carried out: based on the regression analysis results in Table 7.11, the p-value and the t-statistic are statistically significant.

Table 7.11 Regression parameters analysis and statistical tests for TD estimation model of ‘Total Number of Defects’ and ‘Functional Size’ using the SRI-imputed dataset N=49 projects

Variable	**Intercept**	**Coefficient**	**R2**	**95% Confidence Limits**		**T-test**	**Standard Error**	**P-value**
Functional size	3.62	0.0201	0.7	0.0159	0.0243	9.7	1.284	2.742E-15

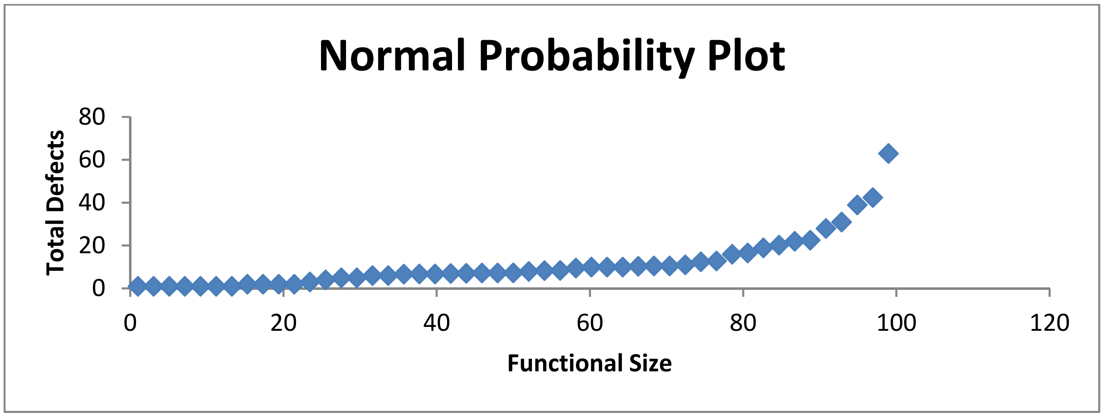


Figure 7.11 Graphical representation for ‘Total Defects’ based on ‘Functional Size’ using the SRI-imputed dataset (X and Y) of N=49 projects

Table 7.8 also shows the parameter estimates for the ‘Total Number of Defects’ estimation model: (constant = 3.62 TD and 0.02 TD/CFP), based on the imputed dataset by Stochastic regression imputation technique (X and Y), N=49 projects - see Figure 7.11:

Total number of defects = 3.62 + 0.02* Functional Size

Table 7.11 also shows that the coefficients of determination (R^2) is (0.7) for the TD estimation model (obtained dataset N=49 imputed by Stochastic regression imputation technique). The confidence interval is (LL is 0.01597 and UL is 0.0243).

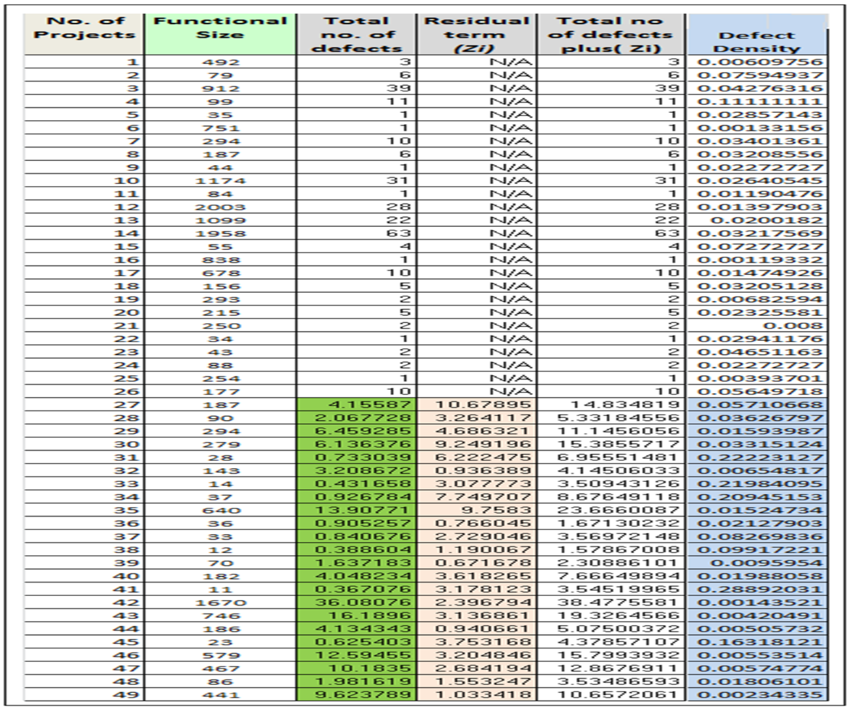
Table 7.12 Summary of the statistical tests for TD estimation model from SRI-imputed dataset N=49 projects

Variable	**Dataset X and Y N=49 projects**	
	Significant **T-test**	**Significant** **P-values**
Functional Size	Yes	Yes

Next, the values of the Defect Density (DD) are observed for the imputed data N=49 projects (after imputing the TD missing data by SRI) that have blank fields of defect density caused by missing of TD values (the software size values are all recorded within the dataset N=49 projects) - see Figure 7.9. DD is to measure the TD with the Functional size in CFP.

- Calculate the Defect Density (DD) after the imputation for the total defects; the 'Defect Density' values have a range from 0.001193 TD/CFP to 0.322291 TD/CFP, with most values at the low end. The median is 0.0321 TD/CFP - see Figure 7.12, based on the formula:

$$\text{Defect Density} = \frac{Total\ Number\ of\ Defects}{Functional\ Size}\ \text{, (ISBSG, 2013)}$$



No. of Projects	Functional Size	Total no. of defects	Residual term *(Zi)*	Total no of defects plus(Zi)	Defect Density
1	492	3	N/A	3	0.00609756
2	79	6	N/A	6	0.07594937
3	912	39	N/A	39	0.04276316
4	99	11	N/A	11	0.11111111
5	35	1	N/A	1	0.02857143
6	751	1	N/A	1	0.00133156
7	294	10	N/A	10	0.03401361
8	187	6	N/A	6	0.03208556
9	44	1	N/A	1	0.02272727
10	1174	31	N/A	31	0.02640545
11	84	1	N/A	1	0.01190476
12	2003	28	N/A	28	0.01397903
13	1099	22	N/A	22	0.0200182
14	1958	63	N/A	63	0.03217569
15	55	4	N/A	4	0.07272727
16	838	1	N/A	1	0.00119332
17	678	10	N/A	10	0.01474926
18	156	5	N/A	5	0.03205128
19	293	2	N/A	2	0.00682594
20	215	5	N/A	5	0.02325581
21	250	2	N/A	2	0.008
22	34	1	N/A	1	0.02941176
23	43	2	N/A	2	0.04651163
24	88	2	N/A	2	0.02272727
25	254	1	N/A	1	0.00393701
26	177	10	N/A	10	0.05649718
27	187	4.15587	10.67895	14.834819	0.05710668
28	90	2.067728	3.264117	5.33184556	0.03626797
29	294	6.459285	4.686321	11.1456056	0.01593987
30	279	6.136376	9.249196	15.3855717	0.03315124
31	28	0.733039	6.222475	6.95551481	0.22223127
32	143	3.208672	0.936389	4.14506033	0.00654817
33	14	0.431658	3.077773	3.50943126	0.21984095
34	37	0.926784	7.749707	8.67649118	0.20945153
35	640	13.90771	9.7583	23.6660087	0.01524734
36	36	0.905257	0.766045	1.67130232	0.02127903
37	33	0.840676	2.729046	3.56972148	0.08269836
38	12	0.388604	1.190067	1.57867008	0.09917221
39	70	1.637183	0.671678	2.30886101	0.0095954
40	182	4.048234	3.618265	7.66649894	0.01988058
41	11	0.367076	3.178123	3.54519965	0.28892031
42	1670	36.08076	2.396794	38.4775581	0.00143521
43	746	16.1896	3.136861	19.3264566	0.00420491
44	186	4.134343	0.940661	5.07500372	0.00505732
45	23	0.625403	3.753168	4.37857107	0.16318121
46	579	12.59455	3.204846	15.7993932	0.00553514
47	467	10.1835	2.684194	12.8676911	0.00574774
48	86	1.981619	1.553247	3.53486593	0.01806101
49	441	9.623789	1.033418	10.6572061	0.00234335

Figure 7.12 An example of sample results of the observed DD
using the SRI-imputed dataset N=49 projects

Estimates are compared by assessing and comparing the predictability with MMRE and Pred(25). From Table 7.13, it can be observed that: the MMRE is 173% and the Pred(25) is 33% for the performance of the estimation model derived from the imputed dataset N=49 projects.

Estimates are compared by assessing and comparing the predictability with MMRE and Pred(25). Form Table 7.13, it can be observed that:

- The MMRE is (173%), which it has increased compared with the MMRE (124%) when the Regression imputation technique has been used in section 7.6 and the Pred(25) is 33% for assessing the TD estimation model (based on ‘Functional Size’ in CFP) derived from the imputed dataset (X and Y), N=49 software projects.

Table 7.13 MMRE and Pred(25) for TD estimation model
from the SRI-imputed dataset N=49 projects

MMRE	Pred(25)
173%	33%

Summary

The obtained results were statistically significant at t-test and P-values for the 'Total Number of Defects' estimates. The coefficient of determination (R^2) was 0.7 for the TD estimation model after using the Stochastic regression imputation technique, the confidence interval did contain true parameters for the estimates, as well as the Standard Error was obviously increased compared Standard Error on the Single and the Regression imputation techniques, which it has restored the data variation with preserving the value of coefficients of determination high. The artificially missing data' percentage was 50% of the data. Based on that, using stochastic regression imputation technique for the analysis provided the best solution for the problem of missing data where it performed better than single and standard regression imputation techniques when the percentage of missing data is 50% of the data set upon this research analysis of defect estimation from projects sized by COSMIC method. The MMRE was increased by fractions than the MMRE value with complete dataset, which gives a clear indication that it performed much better than the previously discussed techniques (SI and RI) within this research work.

7.8 Summary of comparison of performance of SI, RI and SRI imputation techniques on TD estimation models, N=49 projects

Table 7.14 Summary of verification strategy analysis results on dataset N=49 projects

No.	Data set Type	MMRE	Pred(25)	No. of Projects
1	Complete dataset	167%	21%	49
2	Single-Imputed dataset	291%	18%	49
3	Regression-Imputed dataset	124%	30%	49
4	Stochastic-Regression Imputed dataset	173%	33%	49

Table 7.15 Comparison of the analysis results

No.	Comparison results	MMRE	Pred
1	1 vs. 2	+124%	-3%
2	1 vs. 3	-43%	+9%
3	1 vs. 4	+6%	+12%

From Table 7.14 and 7.15, it can be concluded that comparing the performance of the TD estimation models built based on ‘Functional Size’ from four datasets (Complete and the SI, RI, SRI imputed datasets) of N=49 projects indicates that:

- The performance of the dataset with single imputed-values against the complete dataset represents an increase in the MMRE of 124%, and represents a decrease in the Pred(25) of 3% – See line 1 in Table 7.15.
- The performance of the dataset with estimated imputed-values against the complete dataset represents a decrease in the MMRE of 43%, and represents an increase in the Pred(25) of 9% – See line 2 in Table 7.15.
- The performance of the dataset with stochastic estimated imputed-values against the complete dataset represents an increase in the MMRE of 6%, and represents an increase in the Pred(25) of 12% – See line 3 in Table 7.15.

CHAPTER 8

SIX SIGMA ANALYSES FOR SOFTWARE PROJECTS OF ISBSG DATASET

8.1 Introduction

This chapter presents phase 4 of the research methodology which consist of the analysis of the related Six Sigma aspects (as a measurement system and as improvement methodologies DMAIC and DFSS) based on the software projects of ISBSG imputed data sets after the imputation procedures with the three studied imputation techniques on N=360 software projects, in terms of the following variables:

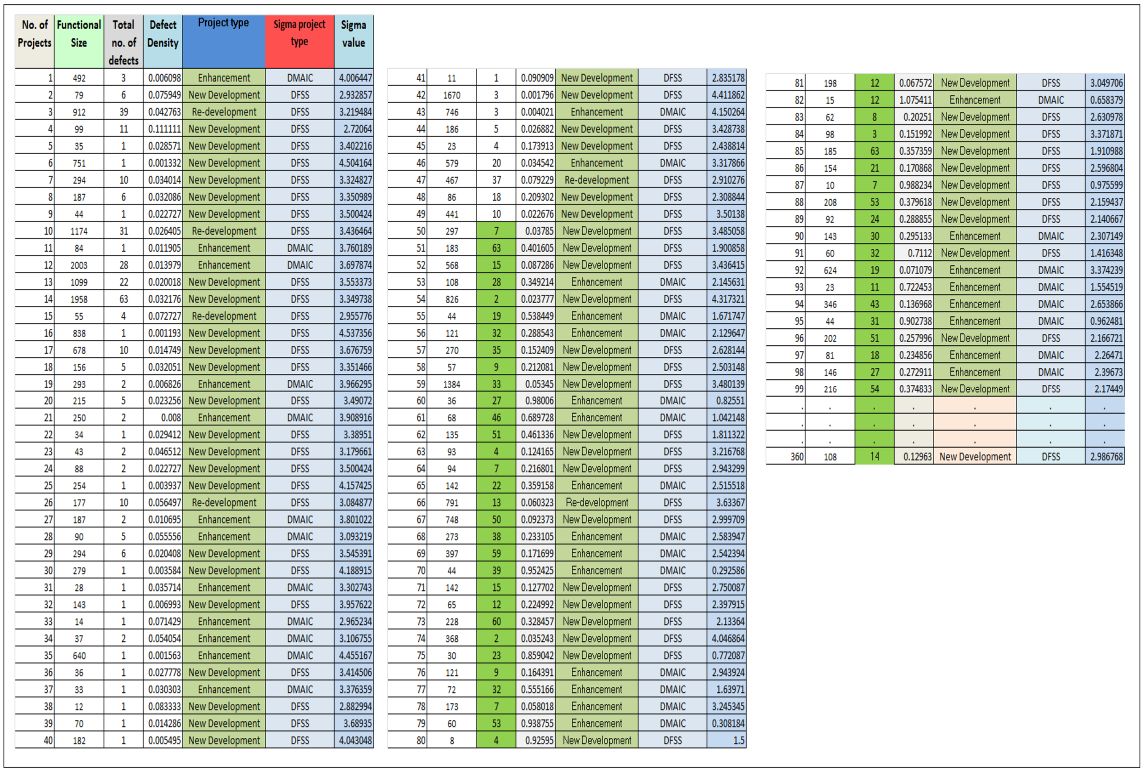
- Number of software projects,
- Software projects’ development type,
- Software projects’ Functional size,
- Software projects’ Total Number of Defects,
- Software projects’ Defect Density,
- Sigma projects’ type (DMAIC and DFSS), and
- Sigma projects’ values.

This chapter also presents how the Sigma values of software projects of the imputed SRI Dataset N=360 projects are used for a Sigma-based classification for defect estimation purposes.

8.2 Sigma analysis results of software projects of ISBSG data set N=360 projects

This section presents the sigma values of each imputed software data set N=360 projects that were imputed with the three selected imputation techniques. Software projects’ sigma values are calculated through the NORMSINV Excel function, taking in consideration the 1.5 sigma shift.

- Figure 8.1 presents an example of sample TD sigma values' for data set N=360 projects imputed with the single imputation technique, while Table 8.1 presents the corresponding averages of Sigma values for both the complete data (N=49 projects) and of the imputed data (N=311 projects).



No. of Projects	Functional Size	Total no. of defects	Defect Density	Project type	Sigma project type	Sigma value
1	492	3	0.006098	Enhancement	DMAIC	4.006447
2	79	6	0.075949	New Development	DFSS	2.932857
3	912	39	0.042763	Re-development	DFSS	3.219484
4	99	11	0.111111	New Development	DFSS	2.72064
5	35	1	0.028571	New Development	DFSS	3.402216
6	751	1	0.001332	New Development	DFSS	4.504164
7	294	10	0.034014	New Development	DFSS	3.324827
8	187	6	0.032086	New Development	DFSS	3.350989
9	44	1	0.022727	New Development	DFSS	3.500424
10	1174	31	0.026405	Re-development	DFSS	3.436464
11	84	1	0.011905	Enhancement	DMAIC	3.760189
12	2003	28	0.013979	Enhancement	DMAIC	3.697874
13	1099	22	0.020018	New Development	DFSS	3.553373
14	1958	63	0.032176	New Development	DFSS	3.349738
15	55	4	0.072727	Re-development	DFSS	2.955776
16	838	1	0.001193	New Development	DFSS	4.537356
17	678	10	0.014749	New Development	DFSS	3.676759
18	156	5	0.032051	New Development	DFSS	3.351466
19	293	2	0.006826	Enhancement	DMAIC	3.966295
20	215	5	0.023256	New Development	DFSS	3.49072
21	250	2	0.008	Enhancement	DMAIC	3.908916
22	34	1	0.029412	New Development	DFSS	3.38951
23	43	2	0.046512	New Development	DFSS	3.179661
24	88	2	0.022727	New Development	DFSS	3.500424
25	254	1	0.003937	New Development	DFSS	4.157425
26	177	10	0.056497	Re-development	DFSS	3.084877
27	187	2	0.010695	Enhancement	DMAIC	3.801022
28	90	5	0.055556	Enhancement	DMAIC	3.093219
29	294	6	0.020408	New Development	DFSS	3.545391
30	279	1	0.003584	New Development	DFSS	4.188915
31	28	1	0.035714	Enhancement	DMAIC	3.302743
32	143	1	0.006993	New Development	DFSS	3.957622
33	14	1	0.071429	Enhancement	DMAIC	2.965234
34	37	2	0.054054	Enhancement	DMAIC	3.106755
35	640	1	0.001563	Enhancement	DMAIC	4.455167
36	36	1	0.027778	New Development	DFSS	3.414506
37	33	1	0.030303	Enhancement	DMAIC	3.376359
38	12	1	0.083333	New Development	DFSS	2.882994
39	70	1	0.014286	New Development	DFSS	3.68935
40	182	1	0.005495	New Development	DFSS	4.043048
41	11	1	0.090909	New Development	DFSS	2.835178
42	1670	3	0.001796	New Development	DFSS	4.411862
43	746	3	0.004021	Enhancement	DMAIC	4.150264
44	186	5	0.026882	New Development	DFSS	3.428738
45	23	4	0.173913	New Development	DFSS	2.438814
46	579	20	0.034542	Enhancement	DMAIC	3.317866
47	467	37	0.079229	Re-development	DFSS	2.910276
48	86	18	0.209302	New Development	DFSS	2.308844
49	441	10	0.022676	New Development	DFSS	3.50138
50	297	7	0.03785	New Development	DFSS	3.485058
51	183	63	0.401605	New Development	DFSS	1.900858
52	568	15	0.087286	New Development	DFSS	3.436415
53	108	28	0.349214	Enhancement	DMAIC	2.145631
54	826	2	0.023777	New Development	DFSS	4.317321
55	44	19	0.538449	Enhancement	DMAIC	1.671747
56	121	32	0.288543	Enhancement	DMAIC	2.129647
57	270	35	0.152409	New Development	DFSS	2.628144
58	57	9	0.212081	New Development	DFSS	2.503148
59	1384	33	0.05345	New Development	DFSS	3.480139
60	36	27	0.98006	Enhancement	DMAIC	0.82551
61	68	46	0.689728	Enhancement	DMAIC	1.042148
62	135	51	0.461336	New Development	DFSS	1.811322
63	93	4	0.124165	New Development	DFSS	3.216768
64	94	7	0.216801	New Development	DFSS	2.943299
65	142	22	0.359158	Enhancement	DMAIC	2.515518
66	791	13	0.060323	Re-development	DFSS	3.63367
67	748	50	0.092373	New Development	DFSS	2.999709
68	273	38	0.233105	Enhancement	DMAIC	2.583947
69	397	59	0.171699	Enhancement	DMAIC	2.542394
70	44	39	0.952425	Enhancement	DMAIC	0.292586
71	142	15	0.127702	New Development	DFSS	2.750087
72	65	12	0.224992	New Development	DFSS	2.397915
73	228	60	0.328457	New Development	DFSS	2.13364
74	368	2	0.035243	New Development	DFSS	4.046864
75	30	23	0.859042	New Development	DFSS	0.772087
76	121	9	0.164391	Enhancement	DMAIC	2.943924
77	72	32	0.555166	Enhancement	DMAIC	1.63971
78	173	7	0.058018	Enhancement	DMAIC	3.245345
79	60	53	0.938755	Enhancement	DMAIC	0.308184
80	8	4	0.92595	New Development	DFSS	1.5
81	198	12	0.067572	New Development	DFSS	3.049706
82	15	12	1.075411	Enhancement	DMAIC	0.658379
83	62	8	0.20251	New Development	DFSS	2.630978
84	98	3	0.151992	New Development	DFSS	3.371871
85	185	63	0.357359	New Development	DFSS	1.910988
86	154	21	0.170868	New Development	DFSS	2.596804
87	10	7	0.988234	New Development	DFSS	0.975599
88	208	53	0.379618	New Development	DFSS	2.159437
89	92	24	0.288855	New Development	DFSS	2.140667
90	143	30	0.295133	Enhancement	DMAIC	2.307149
91	60	32	0.7112	New Development	DFSS	1.416348
92	624	19	0.071079	Enhancement	DMAIC	3.374239
93	23	11	0.722453	Enhancement	DMAIC	1.554519
94	346	43	0.136968	Enhancement	DMAIC	2.653866
95	44	31	0.902738	Enhancement	DMAIC	0.962481
96	202	51	0.257996	New Development	DFSS	2.166721
97	81	18	0.234856	Enhancement	DMAIC	2.26471
98	146	27	0.272911	Enhancement	DMAIC	2.39673
99	216	54	0.374833	New Development	DFSS	2.17449
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
360	108	14	0.12963	New Development	DFSS	2.986768

Figure 8.1 Example of Sigma analysis for Single imputed dataset

Table 8.1 Average sigma values after SI imputation within TD (N=360 projects)

Variable	**TD** **N=360 projects**		
	Average complete data	**Average imputed data**	**Total**
Total Defects	3.49	2.54	2.52
# Projects	49	311	360

Figure 8.2 shows the Sigma values for imputed software projects by single imputation, with a range from 0.105827 Sigma to 4.537356 Sigma, and the average is 2.52 Sigma.

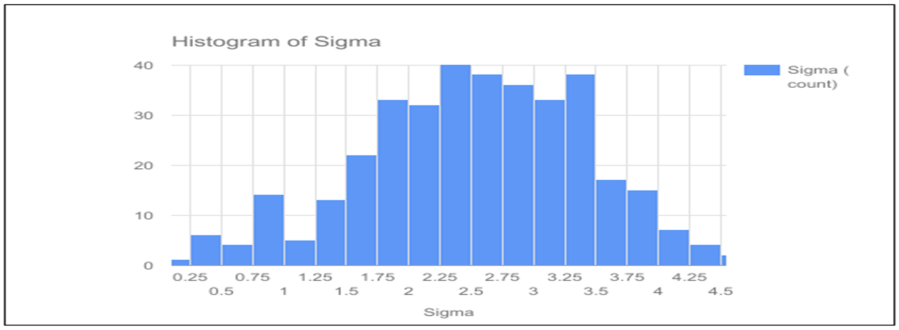
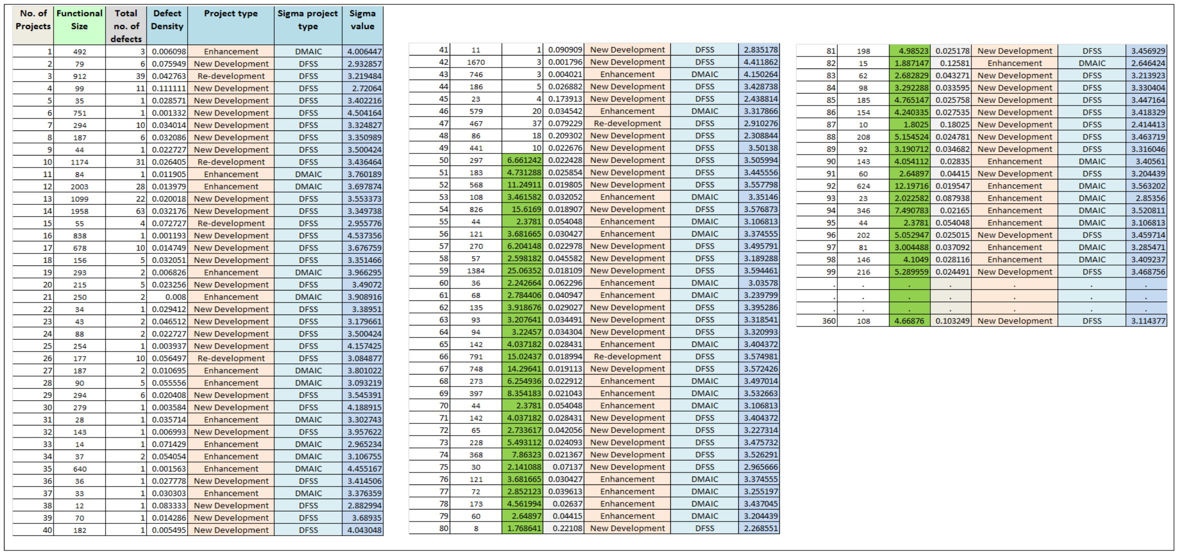


Figure 8.2 Sigma values of imputed data set by SI, N=360 projects

Figure 8.3 presents an example of sample TD sigma values' results for software projects of ISBSG data set N=360 imputed with regression imputation technique, while Table 8.2 present the corresponding averages of the TD for both the complete data (N=49 projects) and of the imputed data (N=311 projects).



No. of Projects	Functional Size	Total no. of defects	Defect Density	Project type	Sigma project type	Sigma value
1	492	3	0.006098	Enhancement	DMAIC	4.006447
2	79	6	0.075949	New Development	DFSS	2.932857
3	912	39	0.042763	Re-development	DFSS	3.219484
4	99	11	0.111111	New Development	DFSS	2.72064
5	35	1	0.028571	New Development	DFSS	3.402216
6	751	1	0.001332	New Development	DFSS	4.504164
7	294	10	0.034014	New Development	DFSS	3.324827
8	187	6	0.032086	New Development	DFSS	3.350989
9	44	1	0.022727	New Development	DFSS	3.500424
10	1174	31	0.026405	Re-development	DFSS	3.436464
11	84	1	0.011905	Enhancement	DMAIC	3.760189
12	2003	28	0.013979	Enhancement	DMAIC	3.697874
13	1099	22	0.020018	New Development	DFSS	3.553373
14	1958	63	0.032176	New Development	DFSS	3.349738
15	55	4	0.072727	Re-development	DFSS	2.955776
16	838	1	0.001193	New Development	DFSS	4.537356
17	678	10	0.014749	New Development	DFSS	3.676759
18	156	5	0.032051	New Development	DFSS	3.351466
19	293	2	0.006826	Enhancement	DMAIC	3.966295
20	215	5	0.023256	New Development	DFSS	3.49072
21	250	2	0.008	Enhancement	DMAIC	3.908916
22	34	1	0.029412	New Development	DFSS	3.38951
23	43	2	0.046512	New Development	DFSS	3.179661
24	88	2	0.022727	New Development	DFSS	3.500424
25	254	1	0.003937	New Development	DFSS	4.157425
26	177	10	0.056497	Re-development	DFSS	3.084877
27	187	2	0.010695	Enhancement	DMAIC	3.801022
28	90	5	0.055556	Enhancement	DMAIC	3.093219
29	294	6	0.020408	New Development	DFSS	3.545391
30	279	1	0.003584	New Development	DFSS	4.188915
31	28	1	0.035714	Enhancement	DMAIC	3.302743
32	143	1	0.006993	New Development	DFSS	3.957622
33	14	1	0.071429	Enhancement	DMAIC	2.965234
34	37	2	0.054054	Enhancement	DMAIC	3.106755
35	640	1	0.001563	Enhancement	DMAIC	4.455167
36	36	1	0.027778	New Development	DFSS	3.414506
37	33	1	0.030303	Enhancement	DMAIC	3.376359
38	12	1	0.083333	New Development	DFSS	2.882994
39	70	1	0.014286	New Development	DFSS	3.68935
40	182	1	0.005495	New Development	DFSS	4.043048
41	11	1	0.090909	New Development	DFSS	2.835178
42	1670	3	0.001796	New Development	DFSS	4.411862
43	746	3	0.004021	Enhancement	DMAIC	4.150264
44	186	5	0.026882	New Development	DFSS	3.428738
45	23	4	0.173913	New Development	DFSS	2.438814
46	579	20	0.034542	Enhancement	DMAIC	3.317866
47	467	37	0.079229	Re-development	DFSS	2.910276
48	86	18	0.209302	New Development	DFSS	2.308844
49	441	10	0.022676	New Development	DFSS	3.50138
50	297	6.661242	0.022428	New Development	DFSS	3.505994
51	183	4.731288	0.025854	New Development	DFSS	3.445556
52	568	11.24911	0.019805	New Development	DFSS	3.557798
53	108	3.461582	0.032052	Enhancement	DMAIC	3.35146
54	826	15.6169	0.018907	New Development	DFSS	3.576873
55	44	2.3781	0.054048	Enhancement	DMAIC	3.106813
56	121	3.681665	0.030427	Enhancement	DMAIC	3.374555
57	270	6.204148	0.022978	New Development	DFSS	3.495791
58	57	2.598182	0.045582	New Development	DFSS	3.189288
59	1384	25.06352	0.018109	New Development	DFSS	3.594461
60	36	2.242664	0.062296	Enhancement	DMAIC	3.03578
61	68	2.784406	0.040947	Enhancement	DMAIC	3.239799
62	135	3.918676	0.029027	New Development	DFSS	3.395286
63	93	3.207641	0.034491	New Development	DFSS	3.318541
64	94	3.22457	0.034304	New Development	DFSS	3.320993
65	142	4.037182	0.028431	Enhancement	DMAIC	3.404372
66	791	15.02437	0.018994	Re-development	DFSS	3.574981
67	748	14.29641	0.019113	New Development	DFSS	3.572426
68	273	6.254936	0.022912	Enhancement	DMAIC	3.497014
69	397	8.354183	0.021043	Enhancement	DMAIC	3.532663
70	44	2.3781	0.054048	Enhancement	DMAIC	3.106813
71	142	4.037182	0.028431	New Development	DFSS	3.404372
72	65	2.733617	0.042056	New Development	DFSS	3.227314
73	228	5.493112	0.024093	New Development	DFSS	3.475732
74	368	7.86323	0.021367	New Development	DFSS	3.526291
75	30	2.141088	0.07137	New Development	DFSS	2.965666
76	121	3.681665	0.030427	Enhancement	DMAIC	3.374555
77	72	2.852123	0.039613	Enhancement	DMAIC	3.255197
78	173	4.561994	0.02637	Enhancement	DMAIC	3.437045
79	60	2.64897	0.04415	Enhancement	DMAIC	3.204439
80	8	1.768641	0.22108	New Development	DFSS	2.268551
81	198	4.98523	0.025178	New Development	DFSS	3.456929
82	15	1.887147	0.12581	Enhancement	DMAIC	2.646424
83	62	2.682829	0.043271	New Development	DFSS	3.213923
84	98	3.292288	0.033595	New Development	DFSS	3.330404
85	185	4.765147	0.025758	New Development	DFSS	3.447164
86	154	4.240335	0.027535	New Development	DFSS	3.418329
87	10	1.8025	0.18025	New Development	DFSS	2.414413
88	208	5.154524	0.024781	New Development	DFSS	3.463719
89	92	3.190712	0.034682	New Development	DFSS	3.316046
90	143	4.054112	0.02835	Enhancement	DMAIC	3.40561
91	60	2.64897	0.04415	New Development	DFSS	3.204439
92	624	12.19716	0.019547	Enhancement	DMAIC	3.563202
93	23	2.022582	0.087938	Enhancement	DMAIC	2.85356
94	346	7.490783	0.02165	Enhancement	DMAIC	3.520811
95	44	2.3781	0.054048	Enhancement	DMAIC	3.106813
96	202	5.052947	0.025015	New Development	DFSS	3.459714
97	81	3.004488	0.037092	Enhancement	DMAIC	3.285471
98	146	4.1049	0.028116	Enhancement	DMAIC	3.409237
99	216	5.289959	0.024491	New Development	DFSS	3.468756
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
360	108	4.66876	0.103249	New Development	DFSS	3.114377

Figure 8.3 Example of Sigma analysis for Regression imputed dataset N=360 projects

Table 8.2 Average Sigma values after RI imputation

Variable	Sigma for RI-imputed TD N=360 projects		
	Average complete data	**Average imputed data**	**Total**
Total Defects	3.49	3.32	3.4
# Projects	49	311	360

Figure 8.4 shows the sigma values for imputed software projects by regression imputation, with a range from 0.531782 Sigma to 4.537356 Sigma, with most values at the high end. The median is 3.39 Sigma.

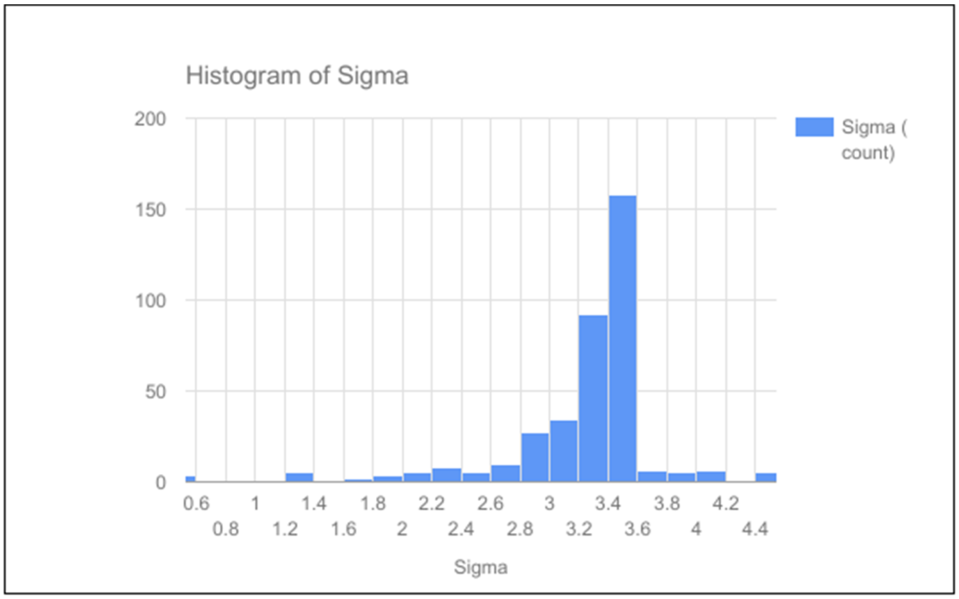
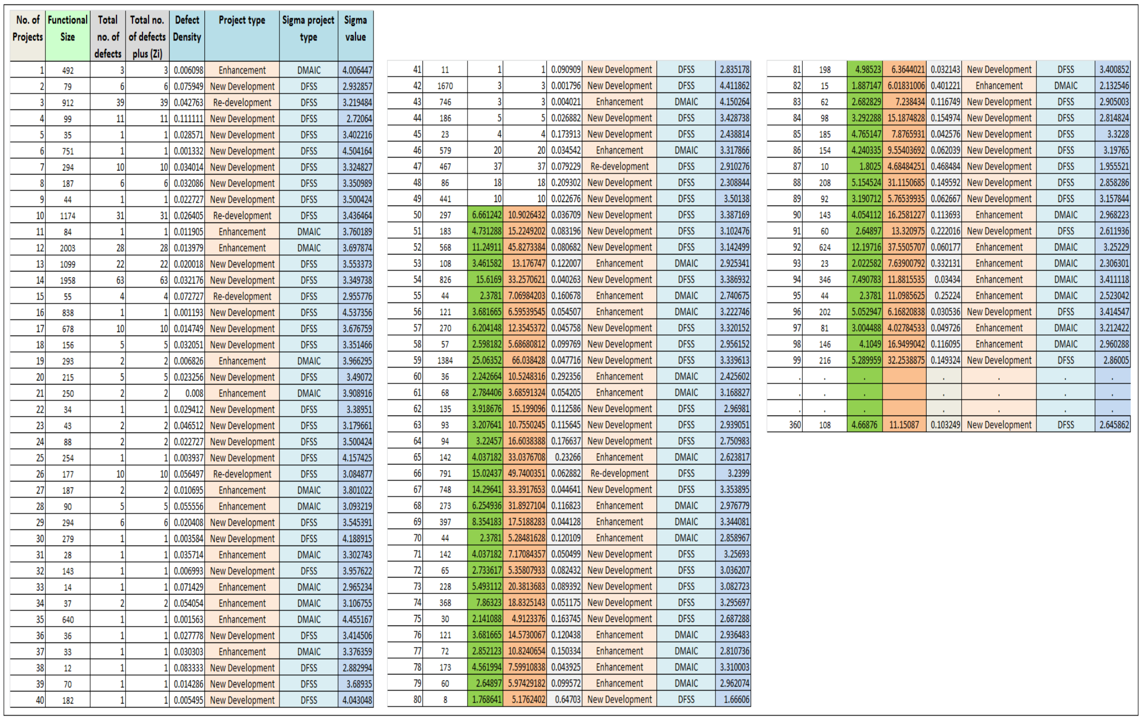


Figure 8.4 Sigma values of imputed data set by RI, N=360 projects

- Figure 8.5 presents an example of sample TD sigma values' results for software projects of imputed with stochastic regression imputation, while Table 8.3 present the corresponding averages of the TD for both the complete data (N=49 projects) and of the imputed data (N=311 projects).



No. of Projects	Functional Size	Total no. of defects	Total no. of defects plus (Zi)	Defect Density	Project type	Sigma project type	Sigma value
1	492	3	3	0.006098	Enhancement	DMAIC	4.006447
2	79	6	6	0.075949	New Development	DFSS	2.932857
3	912	39	39	0.042763	Re-development	DFSS	3.219484
4	99	11	11	0.111111	New Development	DFSS	2.72064
5	35	1	1	0.028571	New Development	DFSS	3.402216
6	751	1	1	0.001332	New Development	DFSS	4.504164
7	294	10	10	0.034014	New Development	DFSS	3.324827
8	187	6	6	0.032086	New Development	DFSS	3.350989
9	44	1	1	0.022727	New Development	DFSS	3.500424
10	1174	31	31	0.026405	Re-development	DFSS	3.436464
11	84	1	1	0.011905	Enhancement	DMAIC	3.760189
12	2003	28	28	0.013979	Enhancement	DMAIC	3.697874
13	1099	22	22	0.020018	New Development	DFSS	3.553373
14	1958	63	63	0.032176	New Development	DFSS	3.349738
15	55	4	4	0.072727	Re-development	DFSS	2.955776
16	838	1	1	0.001193	New Development	DFSS	4.537356
17	678	10	10	0.014749	New Development	DFSS	3.676759
18	156	5	5	0.032051	New Development	DFSS	3.351466
19	293	2	2	0.006826	Enhancement	DMAIC	3.966295
20	215	5	5	0.023256	New Development	DFSS	3.49072
21	250	2	2	0.008	Enhancement	DMAIC	3.908916
22	34	1	1	0.029412	New Development	DFSS	3.38951
23	43	2	2	0.046512	New Development	DFSS	3.179661
24	88	2	2	0.022727	New Development	DFSS	3.500424
25	254	1	1	0.003937	New Development	DFSS	4.157425
26	177	10	10	0.056497	Re-development	DFSS	3.084877
27	187	2	2	0.010695	Enhancement	DMAIC	3.801022
28	90	5	5	0.055556	Enhancement	DMAIC	3.093219
29	294	6	6	0.020408	New Development	DFSS	3.545391
30	279	1	1	0.003584	New Development	DFSS	4.188915
31	28	1	1	0.035714	Enhancement	DMAIC	3.302743
32	143	1	1	0.006993	New Development	DFSS	3.957622
33	14	1	1	0.071429	Enhancement	DMAIC	2.965234
34	37	2	2	0.054054	Enhancement	DMAIC	3.106755
35	640	1	1	0.001563	Enhancement	DMAIC	4.455167
36	36	1	1	0.027778	New Development	DFSS	3.414506
37	33	1	1	0.030303	Enhancement	DMAIC	3.376359
38	12	1	1	0.083333	New Development	DFSS	2.882994
39	70	1	1	0.014286	New Development	DFSS	3.68935
40	182	1	1	0.005495	New Development	DFSS	4.043048
41	11	1	1	0.090909	New Development	DFSS	2.835178
42	1670	3	3	0.001796	New Development	DFSS	4.411862
43	746	3	3	0.004021	Enhancement	DMAIC	4.150264
44	186	5	5	0.026882	New Development	DFSS	3.428738
45	23	4	4	0.173913	New Development	DFSS	2.438814
46	579	20	20	0.034542	Enhancement	DMAIC	3.317866
47	467	37	37	0.079229	Re-development	DFSS	2.910276
48	86	18	18	0.209302	New Development	DFSS	2.308844
49	441	10	10	0.022676	New Development	DFSS	3.50138
50	297	6.661242	10.9026432	0.036709	New Development	DFSS	3.387169
51	183	4.731288	15.2249202	0.083196	New Development	DFSS	3.102476
52	568	11.24911	45.8273384	0.080682	New Development	DFSS	3.142499
53	108	3.461582	13.176747	0.122007	Enhancement	DMAIC	2.925341
54	826	15.6169	33.2570621	0.040263	New Development	DFSS	3.386932
55	44	2.3781	7.06984203	0.160678	Enhancement	DMAIC	2.740675
56	121	3.681665	6.59539545	0.054507	Enhancement	DMAIC	3.222746
57	270	6.204148	12.3545372	0.045758	New Development	DFSS	3.320152
58	57	2.598182	5.68680812	0.099769	New Development	DFSS	2.956152
59	1384	25.06352	66.038428	0.047716	New Development	DFSS	3.339613
60	36	2.242664	10.5248316	0.292356	Enhancement	DMAIC	2.425602
61	68	2.784406	3.68591324	0.054205	Enhancement	DMAIC	3.168827
62	135	3.918676	15.199096	0.112586	New Development	DFSS	2.96981
63	93	3.207641	10.7550245	0.115645	New Development	DFSS	2.939051
64	94	3.22457	16.6038388	0.176637	New Development	DFSS	2.750983
65	142	4.037182	33.0376708	0.23266	Enhancement	DMAIC	2.623817
66	791	15.02437	49.7400351	0.062882	Re-development	DFSS	3.2399
67	748	14.29641	33.3917653	0.044641	New Development	DFSS	3.353895
68	273	6.254936	31.8927104	0.116823	Enhancement	DMAIC	2.976779
69	397	8.354183	17.5188283	0.044128	Enhancement	DMAIC	3.344081
70	44	2.3781	5.28481628	0.120109	Enhancement	DMAIC	2.858967
71	142	4.037182	7.17084357	0.050499	New Development	DFSS	3.25693
72	65	2.733617	5.35807933	0.082432	New Development	DFSS	3.036207
73	228	5.493112	20.3813683	0.089392	New Development	DFSS	3.082723
74	368	7.86323	18.8325143	0.051175	New Development	DFSS	3.295697
75	30	2.141088	4.9123376	0.163745	New Development	DFSS	2.687288
76	121	3.681665	14.5730067	0.120438	Enhancement	DMAIC	2.936483
77	72	2.852123	10.8240654	0.150334	Enhancement	DMAIC	2.810736
78	173	4.561994	7.59910838	0.043925	Enhancement	DMAIC	3.310003
79	60	2.64897	5.97429182	0.099572	Enhancement	DMAIC	2.962074
80	8	1.768641	5.1762402	0.64703	New Development	DFSS	1.66606
81	198	4.98523	6.3644021	0.032143	New Development	DFSS	3.400852
82	15	1.887147	6.01831006	0.401221	Enhancement	DMAIC	2.132546
83	62	2.682829	7.238434	0.116749	New Development	DFSS	2.905003
84	98	3.292288	15.1874828	0.154974	New Development	DFSS	2.814824
85	185	4.765147	7.8765931	0.042576	New Development	DFSS	3.3228
86	154	4.240335	9.55403692	0.062039	New Development	DFSS	3.19765
87	10	1.8025	4.68484251	0.468484	New Development	DFSS	1.955521
88	208	5.154524	31.1150685	0.149592	New Development	DFSS	2.858286
89	92	3.190712	5.76539935	0.062667	New Development	DFSS	3.157844
90	143	4.054112	16.2581227	0.113693	Enhancement	DMAIC	2.968223
91	60	2.64897	13.320975	0.222016	New Development	DFSS	2.611936
92	624	12.19716	37.5505707	0.060177	Enhancement	DMAIC	3.25229
93	23	2.022582	7.63900792	0.332131	Enhancement	DMAIC	2.306301
94	346	7.490783	11.8815535	0.03434	Enhancement	DMAIC	3.411118
95	44	2.3781	11.0985625	0.25224	Enhancement	DMAIC	2.523042
96	202	5.052947	6.16820838	0.030536	New Development	DFSS	3.414547
97	81	3.004488	4.02784533	0.049726	Enhancement	DMAIC	3.212422
98	146	4.1049	16.9499042	0.116095	Enhancement	DMAIC	2.960288
99	216	5.289959	32.2538875	0.149324	New Development	DFSS	2.86005
.	.	.		.	.	.	.
.	.	.		.	.	.	.
.	.	.		.	.	.	.
360	108	4.66876	11.15087	0.103249	New Development	DFSS	2.645862

Figure 8.5 Example of sigma analysis for stochastic regression imputed dataset N=360 projects

Table 8.3 Average sigma values after SRI imputation within TD, N=360 projects

Variable	Sigma for SRI-imputed TD N=360 projects		
	Average complete data	Average imputed data	Total
Total Defects	3.49	3.11	3
# Projects	49	311	360

Figure 8.6 shows the sigma values for imputed software projects by single imputation, with a range from 0.032724 Sigma to 4.537356 Sigma, with most values at the high end. The median is 3.1 Sigma.

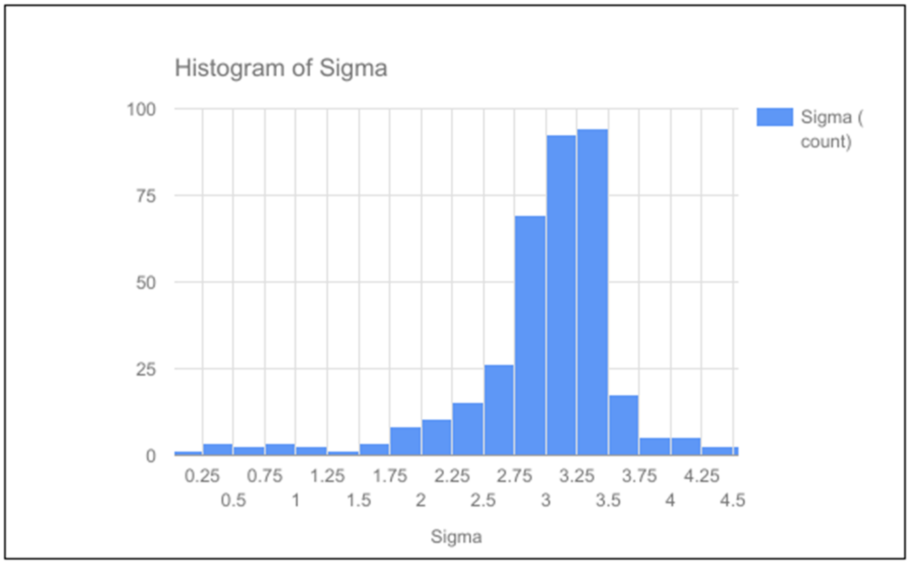


Figure 8.6 Sigma values of imputed data set by SRI, N=360 projects

Figure 8.7 shows the Sigma values of DMAIC projects, with a range from 0.27936 Sigma to 4.455167 Sigma, and the average is 2.31 Sigma.

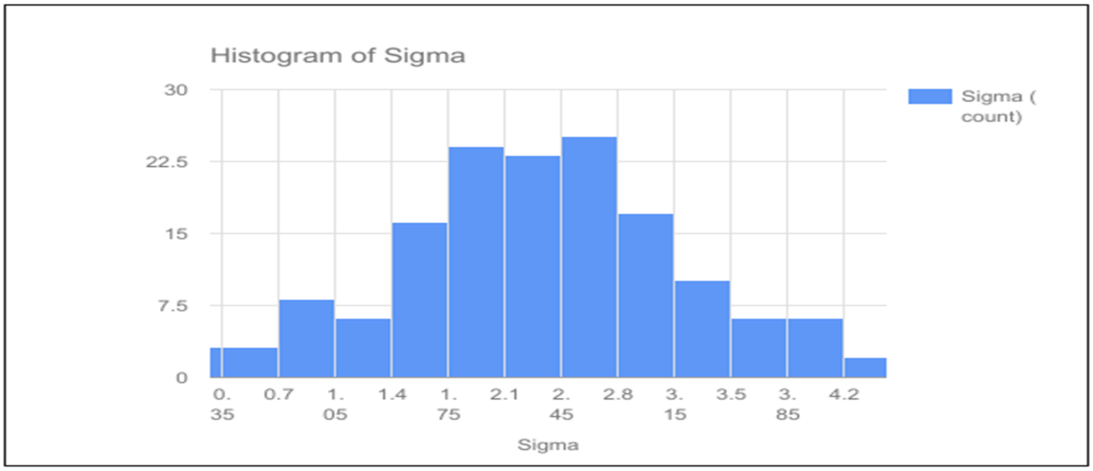


Figure 8.7 Sigma values of DMAIC projects of SRI-imputed dataset N=149 projects

Figure 8.8 shows the Sigma values of DFSS projects, with a range from 0.105827 Sigma to 4.537356 Sigma, the average is 2.67 Sigma.

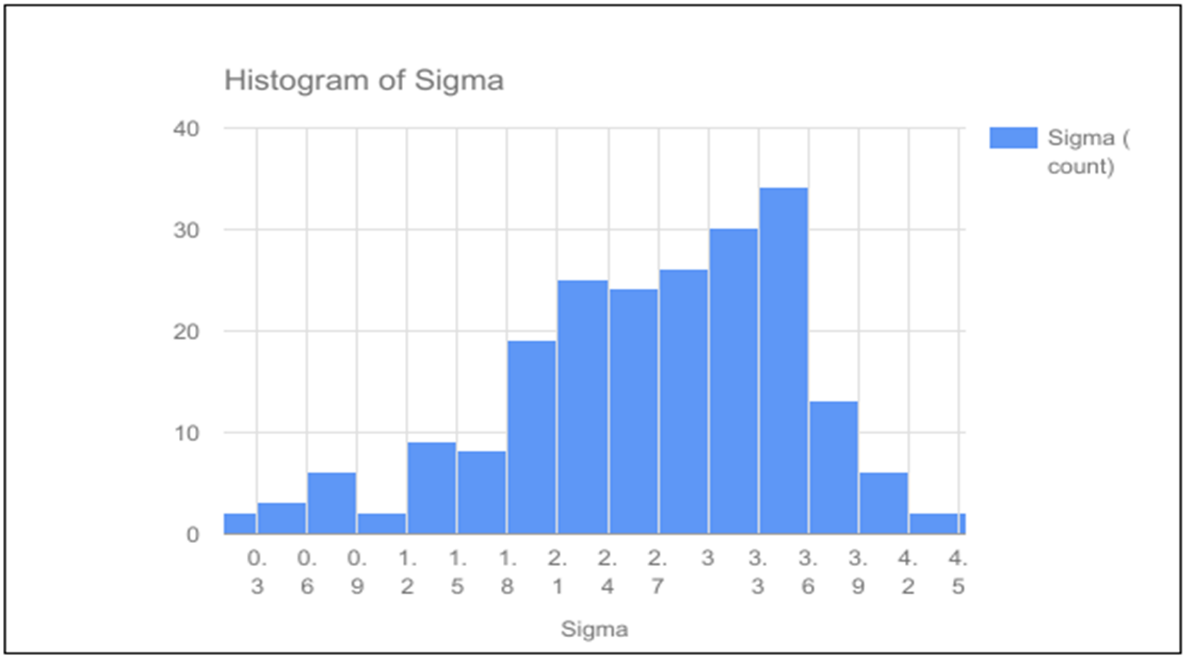


Figure 8.8 Sigma values of DFSS projects of SRI-imputed dataset N=211 projects

8.3 Classification of software projects based on Sigma levels of imputed SRI Dataset N=360 projects for defect estimation purposes

Based on the Sigma values of the imputed SRI Dataset of software projects (N=360 projects) the software projects are classified based on their Sigma values - see Tables 8.4 and 8.5. The purpose of this classification is to determine at which levels of Sigma; the software projects can be better used to build defect estimation models using independent variable ‘Functional Size’ (based on the imputed dataset using ‘stochastic regression imputation technique’ with dataset N=360 projects).

Table 8.4 Software projects classification based on Sigma levels N=360 projects

Sigma level	**Assigned Sigma Ranges**	**Number of projects**
δ	- $\delta < 2$	21 Projects
δ	- $2 \leq \delta < 2.5$	24 Projects
δ	- $2 .5 \leq \delta < 3$	83 Projects
δ	- $3 \leq \delta < 3.5$	186 Projects
δ	- $3.5 \leq \delta < 4$	35 Projects
δ	- $4 \leq \delta$	11 Projects
		360 Projects

From Table 8.4, it can be noted that:

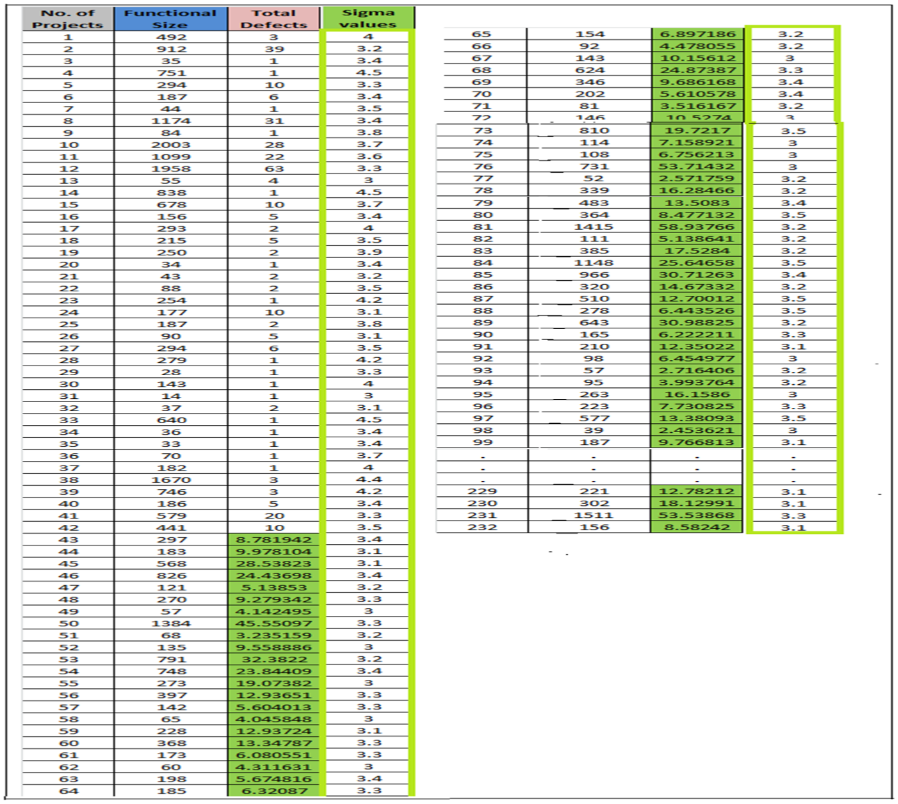
- 21 software projects operate at less than 2 Sigma.
- 24 software projects operate between 2 Sigma and 2.5 Sigma.
- 83 software projects operate between 2.5 Sigma and 3 Sigma.
- 186 software projects operate between 3 Sigma and 3.5 Sigma.
- 35 software projects operate between 3.5 Sigma and 4 Sigma.
- 11 software projects operate at more than 4 Sigma.

Table 8.5 Datasets defect density classification based on Sigma levels N=360 projects

Sigma Datasets	**Assigned Sigma-based Ranges**	**Total number of projects**
(1)	- From 2δ to 4.5δ.	339 Projects
(2)	- From 2.5δ to 4.5δ.	315 Projects
(3)	- From 3δ to 4.5δ.	232 Projects
(4)	- From 3.5δ to 4.5δ.	46 Projects
(5)	- From 4δ to 4.5δ.	11 Projects
		Out of 360 Projects

Based on Table 8.4, the Sigma-based datasets of software projects are classified based on their Sigma levels to determine at which Sigma ranges it can be better to build defect estimation models (using independent variable ‘Functional Size’), which can give better statistical analysis results (see Appendix X) in terms of the criteria used previously in Figure 3.11 such as: MMRE and the Coefficients of Determination (R^2).

For example, the Sigma-based dataset (from 3δ to 4.5δ or more) of N=232 Projects - see line (3) in Table 8.5, is used next, for the statistical analysis (from Figure 3.11) in order to build defect estimation models and analyze their results. (This dataset of N=232 projects - line (3) in Table 8.5 - has given the best statistical analysis results compared with other datasets in Table 8.5 - see Appendix X). Figure 8.9 presents a sample results of the Sigma-based dataset N=232 projects.



No. of Projects	Functional Size	Total Defects	Sigma values
1	492	3	4
2	912	39	3.2
3	35	1	3.4
4	751	1	4.5
5	294	10	3.3
6	187	6	3.4
7	44	1	3.5
8	1174	31	3.4
9	84	1	3.8
10	2003	28	3.7
11	1099	22	3.6
12	1958	63	3.3
13	55	4	3
14	838	1	4.5
15	678	10	3.7
16	156	5	3.4
17	293	2	4
18	215	5	3.5
19	250	2	3.9
20	34	1	3.4
21	43	2	3.2
22	88	2	3.5
23	254	1	4.2
24	177	10	3.1
25	187	2	3.8
26	90	5	3.1
27	294	6	3.5
28	279	1	4.2
29	28	1	3.3
30	143	1	4
31	14	1	3
32	37	2	3.1
33	640	1	4.5
34	36	1	3.4
35	33	1	3.4
36	70	1	3.7
37	182	1	4
38	1670	3	4.4
39	746	3	4.2
40	186	5	3.4
41	579	20	3.3
42	441	10	3.5
43	297	8.781942	3.4
44	183	9.978104	3.1
45	568	28.53823	3.1
46	826	24.43698	3.4
47	121	5.13853	3.2
48	270	9.279342	3.3
49	57	4.142495	3
50	1384	45.55097	3.3
51	68	3.235159	3.2
52	135	9.558886	3
53	791	32.3822	3.2
54	748	23.84409	3.4
55	273	19.07382	3
56	397	12.93651	3.3
57	142	5.604013	3.3
58	65	4.045848	3
59	228	12.93724	3.1
60	368	13.34787	3.3
61	173	6.080551	3.3
62	60	4.311631	3
63	198	5.674816	3.4
64	185	6.32087	3.3
65	154	6.897186	3.2
66	92	4.478055	3.2
67	143	10.15612	3
68	624	24.87387	3.3
69	346	9.686168	3.4
70	202	5.610578	3.4
71	81	3.516167	3.2
72	146	10.5274	3
73	810	19.7217	3.5
74	114	7.158921	3
75	108	6.756213	3
76	731	53.71432	3
77	52	2.571759	3.2
78	339	16.28466	3.2
79	483	13.5083	3.4
80	364	8.477132	3.5
81	1415	58.93766	3.2
82	111	5.138641	3.2
83	385	17.5284	3.2
84	1148	25.64658	3.5
85	966	30.71263	3.4
86	320	14.67332	3.2
87	510	12.70012	3.5
88	278	6.443526	3.5
89	643	30.98825	3.2
90	165	6.222211	3.3
91	210	12.35022	3.1
92	98	6.454977	3
93	57	2.716406	3.2
94	95	3.993764	3.2
95	263	16.1586	3
96	223	7.730825	3.3
97	577	13.38093	3.5
98	39	2.453621	3
99	187	9.766813	3.1
.	.	.	.
.	.	.	.
.	.	.	.
229	221	12.78212	3.1
230	302	18.12991	3.1
231	1511	53.53868	3.3
232	156	8.58242	3.1

Figure 8.9 A sample results of the Sigma-based dataset
N=232 projects from SRI-imputed data set

A linear regression analysis is applied on variable ‘Total Number of Defects’ based on the independent variable ‘Functional Size’ in CFP - see Table 8.6.

Table 8.6 displays a 95% mean confidence interval and a t-test with the associated P-value and verifying the impact of ‘Software Size’ on TD parameter estimates: the inferences are based on the t-distribution and followed by a graphical representation of the relationship of TD based on Size in CFP - see Figure 8.10, which indicates that for every increase of 1000 CFP in ‘Functional Size’, ‘Total Defects’ increases by 26.1 TD. Based on the linear regression analysis results in Table 8.6, the p-value and the t-statistic are statistically significant (20.99 and 2.37277E-55 respectively).

Table 8.6 Regression analysis estimation model for ‘Total Number of Defects’ and ‘Functional Size’ using the Sigma-based dataset (N=232 projects)

Variables	**Intercept**	**Coefficient**	**R2**	**95% Confidence Limits**		**T-test**	**Standard Error**	**P-value**
Functional size	2.51	0.026	0.7	0.0237	0.0286	20.99	0.001245	2.37277E-55

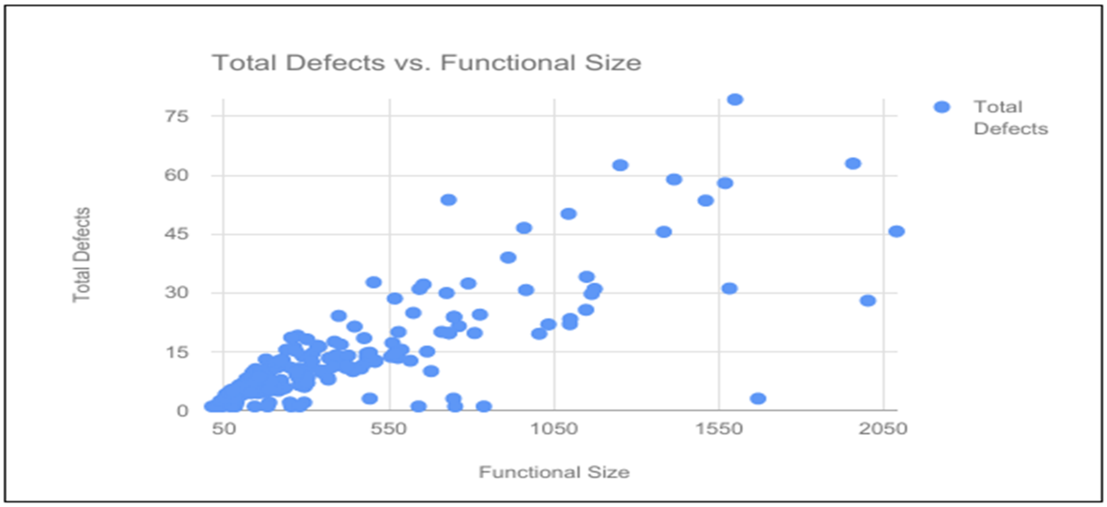


Figure 8.10 Software size (x-axis) and TD (y-axis) N=232 projects

Table 8.6 also shows the parameter estimates for the Total Defects model: (constant = 2.51 TD and 0.026 TD/CFP). Therefore, the TD estimation model based on Sigma-based dataset of N=232 projects of dependent ‘Total Number of Defect’ variable based on the independent variable ‘Functional Size’ is:

Total Number of Defects = 2.51 + 0.026*Functional Size

Table 8.6 also shows the coefficients of determination (R^2) for the TD estimation model (based on CFP) for the Sigma-based dataset N= 232 projects. The R^2 obtained is 0.7, which is the same R^2 value for the TD estimation model built from the original SRI imputed dataset of N=360 projects. The confidence interval is shaped upward (LL is 0.0237, and UL is 0.0286).

The MMRE is 87% and the Pred(25) is 50% for assessing the TD estimation model (based on CFP) derived from the Sigma-based dataset N=232 projects - see Table 8.7.

Table 8.7 MMRE and Pred(25) for ‘Total number of Defects’ and ‘Functional Size’ based on the Sigma-based dataset (N=232 projects)

MMRE	Pred(25)
87%	50%

Figure 8.11 shows the distribution of the software sizes of the Sigma-based dataset of N=232 software projects sized by COSMIC method, with a software size ranges from 14 CFP to 2090 CFP (COSMIC Function Points), with most values at the low end. The median is 213 CFP.

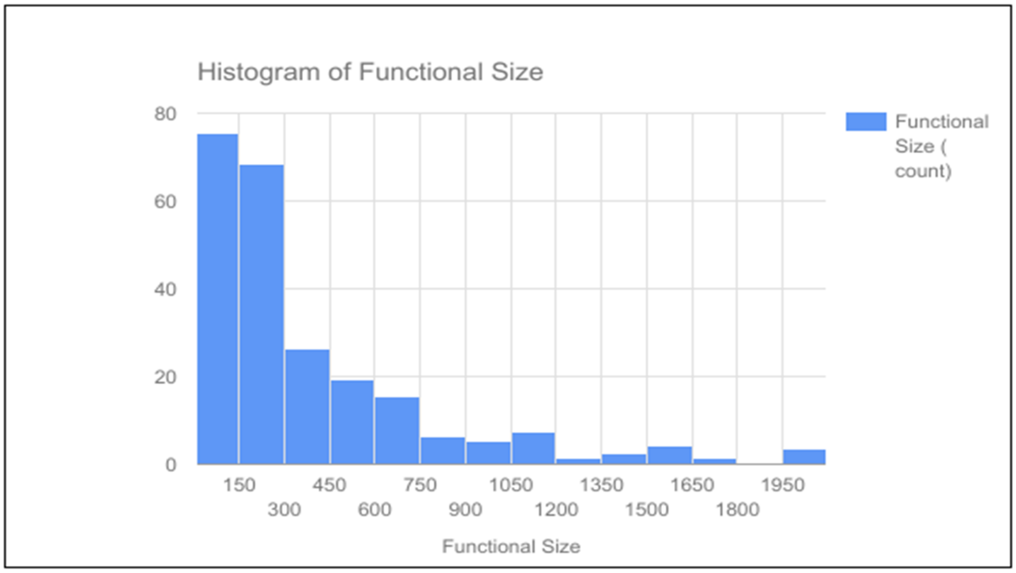


Figure 8.11 CFP software sizes of Sigma-based dataset, N=232 projects

Figure 8.12 shows the distribution of the total defects of the Sigma-based dataset of N=232 software projects sized by COSMIC method, with a range from 1 TD to 79 TD, with most values at the low end. The median is 8 TD.

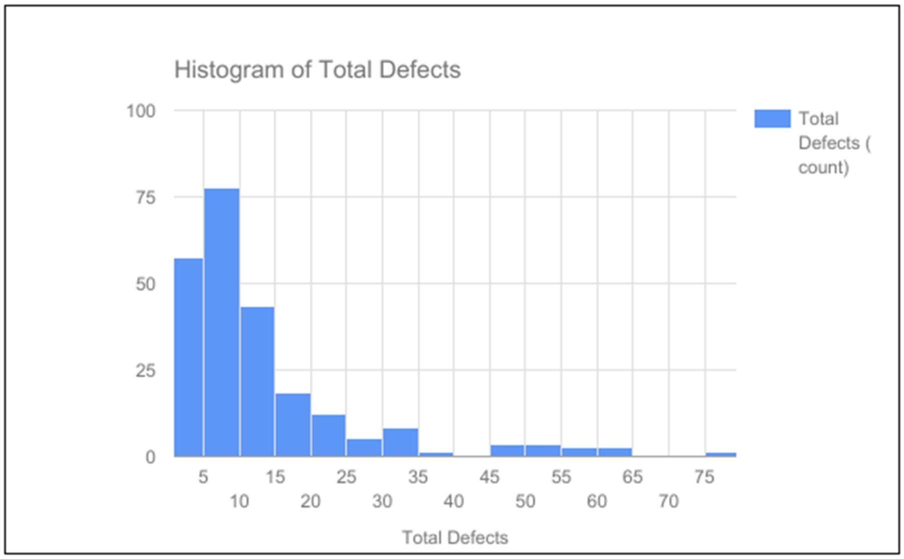


Figure 8.12 Total Defects of Sigma-based dataset, N=232 projects

Figure 8.13 shows the Defect Density of Sigma-based dataset of N=232 software projects, with a range from 0.001193 TD/CFP to 0.073523 TD/CFP, and most of the values are around 0.0397 TD/CFP, plus or minus 0.0153 TD/CFP.

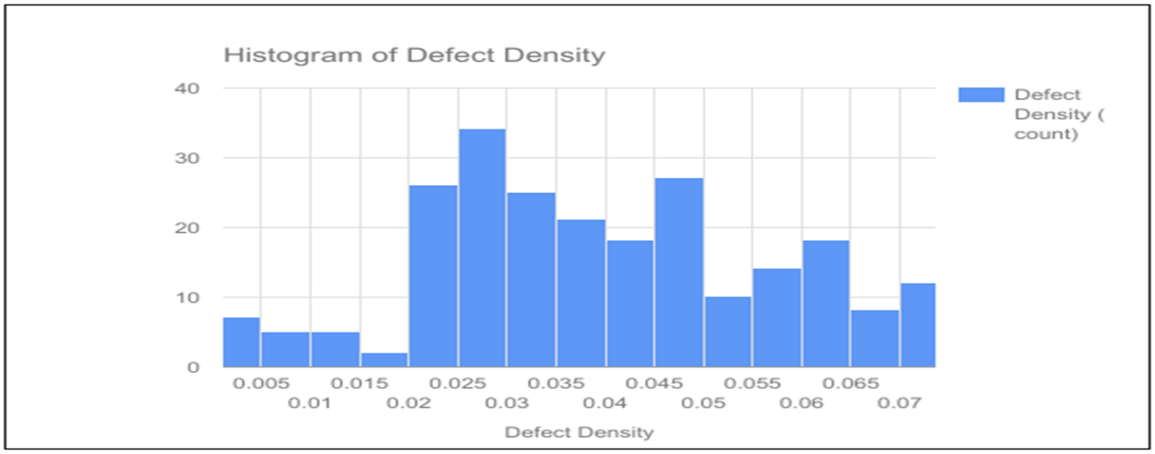


Figure 8.13 Defect density of Sigma-based dataset, N=232 projects

8.4 Summary

It can be noted from Figures 8.7 and 8.8: that the Sigma values of the DFSS projects were higher than the ones at DMAIC projects, which indicated that using DFSS for creating new processes and preventing defects before they occur at the early stages of software life cycle; increases the rate of Sigma values of software projects, compared to the Sigma values of DMAIC projects that aims to improve the existed processes.

The Sigma analysis results showed that:

- Sigma values for imputed software projects by ‘Single imputation technique’, with a range from 0.105827 Sigma to 4.537356 Sigma, the average was 2.52 Sigma.
- Sigma values for imputed software projects by ‘Regression imputation technique’, with a range from 0.531782 Sigma to 4.537356 Sigma, with most values at the high end. The median was 3.39 Sigma.

- Sigma values for imputed software projects by ‘Stochastic regression imputation technique’, with a range from 0.032724 to 4.537356, with most values at the high end. The median was 3.1 Sigma.
- Sigma values of DMAIC projects N=211, with a range from 0.27936 Sigma to 4.455167 Sigma, the average was 2.31 Sigma for imputed software projects by stochastic regression imputation.
- Sigma values of DFSS projects N=149, with a range from 0.105827 Sigma to 4.537356 Sigma, the average was 2.67 Sigma for imputed software projects by stochastic regression imputation.

The Sigma values of software projects of the imputed SRI Dataset N=360 projects were used for a Sigma-based classification for defect estimation purposes. The software projects were classified based on their Sigma levels in Table 8.4 and then classified to represent Sigma-based datasets using their Sigma levels in Table 8.5. Therefore, these Sigma-based datasets were then used to build defect estimation models. The Sigma-based dataset (From 3δ to 4.5δ) of N=232 Projects - see line (3) - in Table 8.5 was used to build defect estimation model based on the independent variable ‘Functional Size’ in CFP, executing the statistical analysis strategy in Figure 3.11 and its statistical criteria, such as: P-value and MMRE. (Based on the statistical analysis results; it represented the better Sigma-based dataset to be used for defect estimation, compared with the other Sigma-based dataset from Table 8.5 - see Appendix X).

The linear regression analysis results in Table 8.6 showed that the p-value and the t-statistic were statistically significant (20.99 and 2.37277E-55 respectively). It also showed that the coefficients of determination (R^2) for the TD estimation model (based on CFP) for the Sigma-based dataset N= 232 projects was 0.7, which is the same as R^2 value for the TD estimation model built from the original SRI imputed dataset of N=360 projects. The TD estimation model based on Sigma-based dataset of N=232 projects of dependent ‘Total Number of Defect’ variable based on the independent variable ‘Functional Size’ in CFP was:

$$\text{Total Number of Defects} = 2.51 + 0.026*\text{Functional Size}$$

The MMRE was 87% and the Pred(25) was 50% for assessing the TD estimation model (based on CFP) derived from the Sigma-based dataset N=232 projects.

CONCLUSION

Six Sigma has achieved recognizable success over the past 20 years in the industry in general. But a limited number of research studies have been conducted within the software industry and there is a lack of empirical studies based on large data repositories of software projects. Thus, such empirical studies would help and clarify up to which point Six Sigma can be used in the software industry, Six Sigma focuses on measuring defects and uses number of defect measures such as a Sigma level. The Sigma defect measures can be helpful for building defect estimation models based on collected data in software engineering such as the ISBSG data repository.

Six sigma has three perspectives: as Sigma measurement perspective such as a Sigma level, as improvement methodologies perspective such as DMAIC and DFSS, and as a management perspective. This research work has focused on only two perspectives: as a sigma level and as improvement methodologies (DMAIC and DFSS) related measurement steps.

In this research work, the ISBSG data repository has been selected because its quality-related data span the entire software life cycle from initiation to completion, while it is burdened by a large number of missing data.

The research goal was to improve software defect estimation (in terms of the independent variable 'Functional Size') with Six Sigma defect measures (Sigma level) using the ISBSG data repository (ISBSG software development and enhancement Repository, Release 12, 2013), handling the high ratio of missing data of variable 'Total Number of Defects' with imputation techniques.

This research work is placed in the context of using the Sigma defect measure of the Sigma level, with software projects defect estimation (in terms of 'Functional Size') on the ISBSG data repository that have a high ratio of missing data in the 'Total Number of Defects' data fields. Thus, the selected imputation techniques were evaluated for dealing with missing data in context of defect estimation modeling.

A few studies tackled the missing data problem in software engineering and some techniques have been proposed trying to handle such problem. Although many of these techniques have been widely used outside of the software engineering field, they also have their own limitations. The literature review has described some of the common techniques used in the literature to deal with the missing data, and also summarized common limitations of their usage and impact when to decide how to handle the missing data.

This research work has focused on three imputation techniques:

- Single Imputation;
- Regression Imputation; and
- Stochastic Regression Imputation.

These four research objectives have been achieved through the studies presented in chapters 3 to 8.

- **OBJECTIVE #1:** To investigate the use of imputation techniques (Single imputation, Regression imputation, and, Stochastic Regression imputation) with the ISBSG data repository R12 for dealing with missing data within the 'Total Number of Defects' variable.

To achieve this objective: chapter 3 has discussed the quality-related information in the ISBSG questionnaire. Chapter 3 also presented the data set preparation which consists of two levels of data preparations based on (Déry et Abran, 2005), and next analyzed the quality-related data fields in the ISBSG MS-Excel data extract (Release 12, 2013). It presented an analysis for the extracted software projects of ISBSG dataset N=360 projects based on the development type and Sigma project type; and finally it identified the steps of the strategy used to implement the imputation techniques - see Figures 3.10 - step (1).

Chapters 4, 5, and 6 have described respectively the single imputation, regression imputation, stochastic regression imputation techniques, and presented how single imputation is implemented to impute the missing data fields of the ‘Total Number of Defects’ in order to produce a complete data set with a sample size N = 360 of software projects.

- On single imputation, the missing values from the ISBSG R12 were imputed as follows: random numbers were generated to provide the values that are missing from the selected data field, that is: Total Number of Defects. The seed values selected for the full sample of 360 projects were set to the minimum and maximum values within the Total Defects data fields (TD) that did not have missing value in R12. The minimum is 1 TD and the maximum is 63 TD.
- On regression imputation, the missing values from the ISBSG R12 were imputed as follows: predicted values were generated from defect estimation model of ‘Total Number of Defects’ based on ‘Functional Size’. This TD estimation was built from the complete values within the ‘Total Number of Defects’ data fields. These complete values are the observations reported within in the Total Number of Defects’ data fields.
  - The estimation model based on the complete subset X, N=49 software projects:

$$\text{Total Number of Defects} = 1.63 + 0.017 * \text{Functional Size}$$

- On stochastic regression imputation, the missing values were imputed as follows: predicted values were generated from the defect estimation model of ‘Total Number of Defects’ based on ‘Functional Size’. This TD estimation was built from the complete values within the ‘Total Number of Defects’ data fields. The estimation step of those values was accomplished previously on standard regression imputation. Then a residual term was added to the predicted values, where this procedure has restored the lost data variability that was resulted to the use of standard regression imputation technique where it has underestimated the standard errors.
  - The estimation model based on the complete subset X, N=49 software projects:

$$\text{Total Number of Defects} = 1.63 + 0.017 * \text{Functional Size} + z_i$$

- **OBJECTIVE #2:** To demonstrate the impact and evaluate the performance of the imputation techniques (Single imputation, Regression imputation, and, Stochastic Regression imputation) on the ISBSG data repository R12, dealing with missing data within 'Total Number of Defects' data fields, for defect estimation purposes based on the independent variable 'Functional Size'.

To achieve this objective: chapters 4, 5, and 6 have used the statistical analysis procedures and their statistical criteria that were identified in chapter 3 - see Figures 3.10 - step (2) and Figure 3.11.

This has achieved by: modeling through a linear regression of the relationship of dependent variable 'Total Number of Defects' based on an independent variable 'Functional Size' in Function Points is used the imputed dataset to obtain the TD estimates and standard errors (build TD estimation models).

The statistical analysis includes:

- Estimate TD (dependent variable) based on Functional Size (independent variable).
- Analysis TD with R^2 and P-value of the estimation results of TD using FP as the dependent variable.
- Outliers' detection: using Grubbs test to investigate whether the outliers affects the rest of data points on TD after filling out its missing data by the three selected imputation techniques.
- Observe the values of Defect Density (DD) for each software project within dataset N=360 projects based on the formula of the Defect Density which measures the quality of software in terms of defects delivered in unit size of software. It is expressed as Defects per Function Points (TD⁄CFP).

The following criteria for analyzing the results of TD estimation models:

- R^2 should be large, R^2: close to 1;
- Standard Errors (STD-E): low Standard Errors;
- Mean Magnitude Relative Error (MMRE): low values of Mean Magnitude Relative Error.

- P-value: Statistical Significance (P-value < 0.05).
- T-test: Statistical Significance (t-test > 2).
- Predictive quality of the TD estimation model: Pred(0.25) = 0.75.

The statistical analysis results show that:

- Using 'single imputation technique' on the dataset N=360 projects to impute 'Total Number of Defects' with COSMIC 'Functional Size' has resulted in: not statistically significant t-test and P-values with outliers for the 'Total Number of Defects' estimates; the coefficient of determination (R^2) was very low at 0.062 with outliers and 0.06346 without outliers, the standard error was 0.00293 and 0.0051 with and without outliers respectively. The outliers were investigated in order to verify whether these results are influenced by the outliers: it can be observed that the R^2 from 0.062 (with outliers on size) to 0.064 (without outliers on size) has increased very little for the dataset without outliers, indicating that the outliers did not influence the TD estimation models. The MMRE was very high at 290%. Thus, these results indicated to the need for exploring more appropriate imputation techniques. (See Appendix II)
  - The TD estimation model for the dependent Total Number of Defects variable based on the independent software 'Functional Size' variable is:

    Total Number of Defects = 26.89 + 0.0033 * Functional Size.

  - Using 'regression imputation technique' on the dataset N=360 projects to impute 'Total Number of Defects' with COSMIC 'Functional Size' has resulted to: statistically significant at t-test and P-values of 'Total Number of Defects' estimates, the standard error has significantly decreased to 0.00053 and the coefficients of determination (R^2) was significantly increased to 0.8 compared to the R^2 when 'single imputation technique' has used. The MMRE was also significantly decreased to 31%. Thus, these results indicated that 'regression imputation technique' had performed better than 'single imputation technique'. However, based on the standard error and the MMRE values; it seemed that the 'regression imputation technique might underestimate the standard error and lacks the variability of data which may lead to bias in the analysis results, which has indicated to the need of exploring more appropriate imputation techniques. (See Appendix III)

- The TD estimation model based on the imputed data set of N=360 projects of dependent 'Total Number of Defect' variable based on the independent variable 'Functional Size' is:

  Total Number of Defects = 1.4 + 0.017 * Functional Size

- Using 'stochastic regression imputation technique' on the dataset N=360 projects to impute 'Total Number of Defects' with COSMIC 'Functional Size' has resulted to: statistically significant at t-test and P-values of 'Total Number of Defects' estimates, the standard error has a little increased to 0.00096 and the coefficient of determination (R^2) was 0.7 which is still good compared to the value of R^2: 0.8 when 'regression imputation technique' was used. The MMRE was also significantly increased to 77% and indicated that the use of 'stochastic regression imputation technique' has restored the lost data variability with statistically significant results. Thus, these results have indicated that 'stochastic regression imputation technique' has performed better than both 'single imputation technique' and 'regression imputation technique'. (See Appendix IV)
  - The TD estimation model based on the imputed data set of N=360 projects of dependent 'Total Number of Defect' variable based on the independent variable 'Functional Size' is:

    Total Number of Defects = 3.8 + 0.025 * Functional Size

To verify the contribution of the three previously implemented imputation techniques on defect estimation: Chapter 7 has measured the predictive accuracy of the defect estimation models (based on the independent variable 'Functional Size' in CFP) obtained from complete dataset N=49 projects and imputed datasets N=49 projects. This has involved developing a verification strategy for analyzing the defect estimation models results, that it verifies the impact of the independent variable

‘Functional Size’ on the parameter estimates of the dependent variable ‘Total Number of Defects’.

This strategy for analyzing the performance of the three imputation techniques used in the empirical studies involved to work with a dataset of complete data set (e.g., it does not contain any missing value: here N=49 software projects) through creating artificially a subset by deleting the data values within the intended variable, and next, impute these artificially missing data by the selected imputation techniques, and next to generate estimation models from the original complete data set and the other imputed data sets, in order to compare and assess the estimates derived from these estimation models through evaluation criteria of such statistical models, such as: Magnitude of Relative Error (MRE) - see Figure 3.11.
This phase used the identified strategy (see Figure 3.10) for implementing the imputation techniques and identified the activities to build defect estimation models (using ‘Functional Size’) with the associated statistical criteria.

The verification strategy was developed as follows:

- Given the complete data set N sample size of projects, randomly split the data set into two subsets X and Y.
- From subset Y, delete the data values for the data field ‘Total Number of Defects’,
- Use *Single Imputation* (SI) technique: based on absolute seeds (min, max) for the missing values of ‘Total Number of Defects’ (TD) within subset Y.
- Use *Regression Imputation* (RI) technique based on replacing each missing value with a predicted value based on estimation model built using complete observations of Total Number of Defects (TD).
- Use *Stochastic Regression Imputation* (SRI) technique based on replacing each missing value with a predicted value based on estimation model built using complete observations of Total Number of Defects (TD).
- Defect estimation models (based on independent variable ‘Functional Size’) will be built with both the initial complete dataset N of software projects and all the imputed dataset N of software projects (N: represents number of software projects in the dataset).

- Compare the TD estimate by assessing and comparing the predictability with MMRE and Pred(25) to assess the predictability of these estimation models based on the following criteria (Conte, Dunsmore et Shen, 1986) and (Abran, 2010):
    - Magnitude of Relative Error (MRE) = | Estimated value – Actual value | / Actual
    - Mean Magnitude of Relative Error for N projects (MMRE) = 1/n*Σ(MREi)
    - Measure of Prediction Quality = Pred(x/100)

The statistical analysis results show that:

- The complete TD dataset N=49 projects resulted to: statistically significant at t-test and P-values of 'Total Number of Defects' estimates; the coefficients of determination (R^2) for the TD estimation model based on 'Functional Size in CFP' was 0.5. The MMRE was 167% and the Pred(25) was 20%. (See Appendix XI)
    - The TD estimation model based on the complete of 'Functional Size' in CFP, N=49 software projects is:

Total Number of Defects = 1.63 + 0.017 * Functional Size

- Using 'single imputation technique' on the dataset N=49 projects to impute 'Total Number of Defects' with COSMIC 'Functional Size' has resulted to: statistically significant t-test and P-values with outliers for the 'Total Number of Defects' estimates; coefficient of determination (R^2) is very low at 0.1033. The MMRE was very high at 291% and the Pred(25) was 18%. (See Appendix XII)
    - The TD estimation model form imputed N=49 projects for the dependent Total Number of Defects variable based on the independent software 'Functional Size' variable is:

Total number of Defects = 13.91+ 0.0119 * Functional Size

- Using 'regression imputation technique' on the dataset N=49 projects to impute 'Total Number of Defects' with COSMIC 'Functional Size' has resulted to: statistically significant t-test and P-values with outliers for the 'Total Number of Defects' estimates; and the coefficients of determination (R^2) was significantly increased to 0.74 compared to the R^2 when 'single imputation technique' has used. The MMRE was also significantly decreased to 124% and the Pred(25) was 30%. Thus, these results indicated that 'regression imputation technique' was performed better than 'single imputation technique'. (See Appendix XIII)

- The TD estimation model form imputed N=49 projects for the dependent Total Number of Defects variable based on the independent software 'Functional Size' variable is:

  Total Number of Defects = 0.13 + 0.022 * Functional Size

- Using 'stochastic regression imputation technique' on the dataset N=49 projects to impute 'Total Number of Defects' with COSMIC 'Functional Size' has resulted to: statistically significant t-test and P-values with outliers for the 'Total Number of Defects' estimates; and the coefficient of determination (R^2) was 0.7 which is still good compared to the value of R^2: 0.74 when 'regression imputation technique' was used. The MMRE was also increased to 173% and the Pred(25) was 33%. Thus, the results indicated that the use of 'stochastic regression imputation technique' has restored the lost data variability with statistically significant results. (See Appendix XIV)
  - The TD estimation model form imputed N=49 projects for the dependent Total Number of Defects variable based on the independent software 'Functional Size' variable is:

  Total Number of Defects = 3.62 + 0.02* Functional Size

- Comparing the performance of the estimation models built with each other indicated that:
  - The performance of the dataset with single imputed-values against the complete dataset represented an increase in the MMRE of 123%, and represented a decrease in the Pred(25) of 3% – see line 1 in Table 7.15.
  - The performance of the dataset with estimated imputed-values against the complete dataset represented a decrease in the MMRE of 44%, and represented an increase in the Pred(25) of 9% – see line 2 in Table 7.15.

- The performance of the dataset with stochastic estimated imputed-values against the complete dataset represented an increase in the MMRE of 6%, and represented an increase in the Pred(25) of 12% – see line 3 in Table 7.15.

- **OBJECTIVE #3:** To investigate and analyze the use of the related Six Sigma aspects of defect measures and its improvement methodologies (DMAIC and DFSS) with the ISBSG data repository R12, after dealing with the missing data in the variable 'Total Number of Defects'.

To achieve this objective: chapter 3 mapped the ISBSG questionnaire to Six Sigma (DMAIC and DFSS) methodologies.

The analysis has shown that (see Appendix VI):

- Number of software projects by type and their percentage:
  - Number of software enhancement projects is 149 projects, which represents 41% of projects number,
  - Number of software re-development projects is 11 projects, which represents 3% of projects number, and
  - Number of new software development projects is 200 projects, which represents the highest percentage of 56% of projects number.

- Number of Sigma projects by type and their percentage, where:
  - Number of the DMAIC projects is 149 projects, which represents 41.4% of projects number, and
  - Number of the DFSS projects is 211 projects, which represents higher percentage of 58.6% than DMAIC projects.

- Software sizes of DMAIC projects, with a range from 2 to 2003 CFP, with most values at the low end. The median size is 95 CFP.
- Software sizes of DFSS projects, with a range from 8 to 2090 CFP, with most values at the low end. The median size is 175 CFP.

Chapter 8 also presented the sigma values analysis of each imputed software datasets N=360 projects imputed with the 3 selected imputation techniques. Software projects' sigma values were calculated through the NORMSINV Excel function, taking in consideration the 1.5 sigma shift.

The Sigma analysis results (see Appendices VII, VIII, and IX) showed that:

- Sigma values for imputed software projects by 'Single imputation technique', with a range from 0.105827 Sigma to 4.537356 Sigma, the average was 2.52 Sigma.
- Sigma values for imputed software projects by 'Regression imputation technique', with a range from 0.531782 Sigma to 4.537356 Sigma, with most values at the high end. The median was 3.39 Sigma.
- Sigma values for imputed software projects by 'Stochastic regression imputation technique', with a range from 0.032724 to 4.537356, with most values at the high end. The median was 3.1 Sigma.
- Sigma values of DMAIC projects N=211, with a range from 0.27936 Sigma to 4.455167 Sigma, the average was 2.31 Sigma for imputed software projects by stochastic regression imputation.
- Sigma values of DFSS projects N=149, with a range from 0.105827 Sigma to 4.537356 Sigma, the average was 2.67 Sigma for imputed software projects by stochastic regression imputation.

- **OBJECTIVE #4:** To build defect estimation models (based on the independent variable 'Functional Size') along with the Six Sigma defect measures from the imputed dataset of the better imputation technique performance among: Single imputation, Regression imputation, and, Stochastic Regression imputation.

To achieve this objective: chapter 8 presented how the Sigma values of software projects of the SRI imputed dataset N=360 projects were used for a Sigma-based classification for defect estimation purposes. This procedure allowed producing Sigma-based datasets with software projects ranges based on Sigma levels (e.g., Sigma-based dataset with a range of software projects from 3δ to 4.5δ or more). (See Appendix X)

Based on Table 8.4, the classified datasets of software projects based on their Sigma level are used for statistical analysis based on Figure 3.11 - see Appendix X. A linear regression analysis was applied on variable 'Total Number of Defects' based on the independent variable 'Functional Size' in CFP using the Sigma-based dataset N=232 projects with a range of from 3δ to 4.5δ of software projects - see Table 8.5.

The linear regression analysis results in Table 8.6 showed that the p-value and the t-statistic were statistically significant (20.99 and 2.37277E-55 respectively). It also showed that the coefficients of determination (R^2) for the TD estimation model (based on CFP) for the Sigma-based dataset N= 232 projects is 0.7, which is the same as R^2 value for the TD estimation model built from the original SRI imputed dataset of N=360 projects. The TD estimation model based on Sigma-based dataset of N=232 projects of dependent 'Total Number of Defect' variable based on the independent variable 'Functional Size' in CFP is:

$$\text{Total Number of Defects} = 2.51 + 0.026*\text{Functional Size}$$

The MMRE was 87% and the Pred(25) was 50% for assessing the TD estimation model (based on CFP) derived from the Sigma-based dataset N=232 projects.

This research analysis work reported on a set of empirical studies tackling the research issues of improving software defect estimation models with Sigma defect measures - such as: Sigma level - using of ISBSG data repository with a high ratio of missing data - more than 50% of data. Several imputation techniques were discussed, which resulted in three imputation techniques selected for this research work: single imputation, regression imputation, and stochastic regression imputation. The selected imputation techniques were used to impute the missing data within the variable 'Total Number of Defects', and then compared with each other using the common verification criteria in the scientific research. A further verification strategy was developed in order to compare and assess the performance of the selected imputation techniques through verifying the predictive accuracy of the obtained software defect estimation models form the imputed datasets. A Sigma-based classification was carried out on the imputed dataset of the better performance imputation technique on software defect estimation. The Sigma-based classification has resulted in Sigma-based datasets based on the imputed dataset. Finally, software defect estimation models were built on the Sigma-based datasets.

This empirical study found that 'stochastic regression imputation technique' performed better than the other two selected imputation techniques ('single imputation technique' and 'regression imputation technique').

We encourage researchers to conduct similar methodological assessments to find the most suitable method of imputation for their specific datasets and measures, and to conduct more empirical studies for Six Sigma defect measures based on other known data repositories in software engineering, such as the PROMISE data repository.

FUTURE WORK AND RECOMMENDATIONS

The following have been identified as future works:

1) Perform a set of empirical studies tackling the research issues of defect estimation models using the software projects sized by the IFPUG functional size method in the ISBSG data repository, with Six Sigma analysis, which include:
   - Investigate the missing values and whether these missing data requires to be handled.
   - Perform Sigma defect-based analysis on these software projects sized by the IFPUG functional size method with defect estimation.
2) Investigate the applicability of using other types of imputation techniques such as the Machine Learning (ML) imputation techniques, especially the K-Nearest Neighbor Algorithm technique, to deal with the missing TD data.
3) Request a special data extract of the software development and enhancement repository form the ISBSG organization, and perform the previously accomplished of set of empirical studies on that special data extract. That includes:
   - Information regarding the 'Total Number of Defects' reported and the measured 'Functional Size' for the software projects of the ISBSG dataset, that were collected during the phases of software life cycle, such as:
     - 'Number of Defects' recorded at the Implementation phase and the measured 'Functional Size', which were collected by the ISBSG data questionnaire at Questions 49 and 51.
     - 'Number of Defects' recorded at the Specification phase and the measured 'Functional Size', which were collected by the ISBSG data questionnaire at Questions 27 and 28.
     - 'Number of Defects' recorded at the Design phase and the measured 'Functional Size', which were collected by the ISBSG data questionnaire at Questions 32 and 34.
     - Investigate the use of other Sigma defect measures with building defect estimation models, such as: process capability indices.

  - And build defect estimation models at these phases of software life cycle, and also to determine the Sigma values (or levels) at these phases.

4) Investigate using the Design for Six Sigma (DFSS) along with the ISBSG Software Maintenance and Support repository, including its data collection questionnaire.
5) Conduct empirical studies using the PROMISE data repository with the 54 defect estimation datasets, which involves:
   - Investigating the missing data and whether it needs to be handled.
   - Investigating the applicability of performing Sigma defect-based analysis on their datasets.

ANNEX I

LIST OF APPENDICES ON CD-ROM

The following is the list of appendices referenced within this thesis and on the attached CD-ROM:

Appendix #	File name	Description
Folder name: Imputation Datasets		
I	Dataset_N-360_TD_Size.xls	The data set of 360 software projects of independent variables 'Total Number of Defects', and 'Software Functional Size', with missing data.
II	Dataset_N-360_TD_Size_SI.xls	The data set of 360 software projects of independent variables 'Total Number of Defects', and 'Software Functional Size', that imputed by Single Imputation. (With and without outliers).
III	Dataset_N-360_TD_Size_RI.xls	The data set of 360 software projects of independent variables 'Total Number of Defects', and 'Software Functional Size', that imputed by Regression Imputation. (With and without outliers).
IV	Dataset_N-360_TD_Size_SRI.xls	The data set of 360 software projects of independent variables 'Total Number of Defects', and 'Software Functional Size', that imputed by

		Stochastic Regression Imputation. (With and without outliers).
Folder name: Defect Density after Imputation Datasets		
V	Dataset_N-360_TD_Size_SI-RI-SRI_Defect-Density.xls	The observed 'Defect Density' values after the imputation procedures of Single, Regression, and Stochastic Regression Imputation techniques.
Folder name: Six Sigma Analysis Datasets		
VI	Dataset_N-360_Size_software-projects-development-type_Sigma projects-type.xls	The data set of 360 software projects with regards to software projects' development type and sigma projects' type.
VII	Dataset_N-360_TD_Size_SI_Sigma-Values.xls	The data set of 360 software projects that imputed by Single Imputation, with regards to their Sigma values.
VIII	Dataset_N-360_TD_Size_RI_Sigma-Values.xls	The data set of 360 software projects that imputed by Regression Imputation, with regards to with regards to their Sigma values.
IX	Dataset_N-360_TD_Size_SRI_Sigma-Values.xls	The data set of 360 software projects that imputed by Stochastic Regression Imputation, with regards to their Sigma values.
X	Sigma-based_Datasets_Analysis.xls	Statistical analysis on Sigma-based datasets that are classified from the data set of 360 software projects that imputed by Regression Imputation.
Folder name: Verification of Imputation Performance		
XI	Dataset_N-49_TD_Size.xls	The data set of 49 software projects of independent variables 'Total

		Number of Defects’, and ‘Software Functional Size’, with missing data.
XII	Dataset_N-49_TD_Size_SI.xls	The data set of 49 software projects of independent variables ‘Total Number of Defects’, and ‘Software Functional Size’, that imputed by Single Imputation.
XIII	Dataset_N-49_TD_Size_RI.xls	The data set of 49 software projects of independent variables ‘Total Number of Defects’, and ‘Software Functional Size’, that imputed by Regression Imputation.
XIV	Dataset_N-49_TD_Size_SRI.xls	The data set of 49 software projects of independent variables ‘Total Number of Defects’, and ‘Software Functional Size’, that imputed by Stochastic Regression Imputation.
Folder name: For More Details		
XV	CSF_Implment_Six_Sigma_software.docx	Critical success factors for implementing Six Sigma in software organizations.
XVI	PROMISE_Investigation.docx	An investigation for the use of PROMISE data repository.
XVII	COSMIC_VS_IFPUG.docx	Comparison between COSMIC and IFPUG sizing methods.

BIBLIOGRAPHY

Abran, Alain. 2010. *Software metrics and software metrology*. John Wiley & Sons.

Abran, Alain. 2015. *Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers*. Wiley-IEEE Computer Society Pr, 288 p.

Abran, Alain, Iphigénie Ndiaye et Pierre Bourque. 2007. « Evaluation of a black-box estimation tool: A case study ». *Software Process: Improvement and Practice,* vol. 12, n° 2, p. 199-218.

Al-Qutaish, Rafa E, et Khalid T Al-Sarayreh. 2008. « Applying six-sigma concepts to the software engineering: myths and facts ». In *Proceedings of the 7th International Conference on Software Engineering Parallel and Distributed Systems SEPADS08*. p. 178-183. Citeseer.

Al Qutaish, Rafa. 2007. « SPQMM: A software product quality maturity model using ISO/IEEE standards, metrology, and sigma concepts ». École de technologie supérieure.

Antony, Jiju. 2007. « What is the role of academic institutions for the future development of Six Sigma? ». *International journal of productivity and performance management,* vol. 57, n° 1, p. 107-110.

Antony, Jiju, et Ricardo Banuelas. 2002. « Key ingredients for the effective implementation of Six Sigma program ». *Measuring business excellence,* vol. 6, n° 4, p. 20-27.

Antony, Jiju, et Craig Fergusson. 2004. « Six Sigma in the software industry: results from a pilot study ». *Managerial Auditing Journal,* vol. 19, n° 8, p. 1025-1032.

Azen, S, et M Van Guilder. 1981. « Conclusions regarding algorithms for handling incomplete data ». In *Proceedings of the Statistical Computing Section, American Statistical Association*. Vol. 4, p. 53-56.

Bala, Abdalla. 2013. « Impact analysis of a multiple imputation technique for handling missing value in the ISBSG repository of software projects ». École de technologie supérieure.

Baraldi, Amanda N, et Craig K Enders. 2010. « An introduction to modern missing data analyses ». *Journal of school psychology,* vol. 48, n° 1, p. 5-37.

Bart, Massey. 2005. « Longitudinal analysis of long timescale open source repository data ». *ACM SIGSOFT Software Engineering Notes,* vol. 30, n° 4, p. 11-16.

Bendell, T. 2004. « Managing engineering improvement by six sigma ». In *Engineering Management Conference, 2004. Proceedings. 2004 IEEE International*. Vol. 3, p. 1114-1116. IEEE.

Biehl, Richard E. 2004. « Six Sigma for software ». *IEEE Software,* vol. 21, n° 2, p. 68-70.

Binder, Robert V. 1997. « Can a manufacturing quality model work for software? ». *IEEE Software,* vol. 14, n° 5, p. 101-102.

Chapman, Alan. 2005. « Six Sigma training, history, definitions - Six Sigma and quality management glossary ». < http://www.businessballs.com/sixsigma.htm >.

Cheikhi, Laila. 2008. « Études empiriques des relations entre les modèles de qualité du logiciel d'ISO 9126 en utilisant le référentiel de données d'ISBSG et la méthode Taguchi ». École de technologie supérieure.

Cheikhi, Laila, et Alain Abran. 2013. « Promise and isbsg software engineering data repositories: A survey ». In *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*. p. 17-24. IEEE.

Cheikhi, Laila, Alain Abran et Luigi Buglione. 2006. « ISBSG Software Project Repository & ISO 9126: An Opportunity for Quality Benchmarking ». *European Journal for the Informatics Professional,* vol. 7, n° 1, p. 46-52.

Cheikhi, Laila, Alain Abran et Luigi Buglione. 2007. « The ISBSG software project repository: an analysis from the ISO 9126 quality perspective ». *Software Quality Professional,* vol. 9, n° 2, p. 4-24.

Clark, Brad, et Dave Zubrow. 2001. « How good is the software: a review of defect prediction techniques ». In *Software Engineering Symposium, Carreige Mellon University*.

Colledge, MJ, JH Johnson, R Pare et IG Sande. 1978. « Large scale imputation of survey data ». *Survey Methodology,* vol. 4, p. 203-224.

Conte, Samuel Daniel, Hubert E Dunsmore et Vincent Y Shen. 1986. *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc.

Coronado, Ricardo Banuelas, et Jiju Antony. 2002. « Critical success factors for the successful implementation of six sigma projects in organisations ». *The TQM magazine,* vol. 14, n° 2, p. 92-99.

Cukic, Bojan. 2005. « Guest editor's introduction: The promise of public software engineering data repositories ». *IEEE software,* vol. 22, n° 6, p. 20-22.

Davies, Laurie, et Ursula Gather. 1993. « The identification of multiple outliers ». *Journal of the American Statistical Association,* vol. 88, n° 423, p. 782-792.

Déry, David, et Alain Abran. 2005. « Investigation of the effort data consistency in the ISBSG repository ». In *15th International Workshop on Software Measurement-IWSM*. p. 123-136.

Donders, A Rogier T, Geert JMG van der Heijden, Theo Stijnen et Karel GM Moons. 2006. « Review: a gentle introduction to imputation of missing values ». *Journal of clinical epidemiology,* vol. 59, n° 10, p. 1087-1091.

Electronic, General. 2005. « GE Six Sigma ». <http://www.ge.com/en/company/companyinfo/quality/whatis.htm >.

Enders, Craig K. 2010. *Applied missing data analysis*. Guilford Press.

Eurostat. 2007. « practical guide to data validation in eurostat eurostat ». 2007 Edition. p. 44. <http://ec.europa.eu/eurostat/ramon/statmanuals/files/PRACTICAL_GUIDE_TO_DATA_VALIDATION.pdf >. Consulté le 2016.

Fehlmann, Thomas. 2004. « Six sigma for software ». In *Proceedings of the 1st SMEF Conference, Rome*.

Feng, Qianmei. 2008. « Six sigma: Continuous improvement toward excellence ». In *Collaborative Engineering*. p. 43-60. Springer.

Fox-Wasylyshyn, Susan M, et Maher M El-Masri. 2005. « Handling missing data in self-report measures ». *Research in nursing & health,* vol. 28, n° 6, p. 488-495.

García-Laencina, Pedro J, José-Luis Sancho-Gómez et Aníbal R Figueiras-Vidal. 2010. « Pattern classification with missing data: a review ». *Neural Computing and Applications,* vol. 19, n° 2, p. 263-282.

Gay, Gregory, Tim Menzies, Misty Davies et Karen Gundy-Burlet. 2010. « Automatically finding the control variables for complex system behavior ». *Automated Software Engineering,* vol. 17, n° 4, p. 439-468.

Graham, John W. 2012. « Missing data theory ». In *Missing Data*. p. 3-46. Springer.

Graham, John W, Scott M Hofer, Stewart I Donaldson, David P MacKinnon et Joseph L Schafer. 1997. « Analysis with missing data in prevention research ». *The science of prevention: Methodological advances from alcohol and substance abuse research,* vol. 1, p. 325-366.

Graham, John W, et Joseph L Schafer. 1999. « On the performance of multiple imputation for multivariate data with small sample size ». *Statistical strategies for small sample research,* vol. 50, p. 1-27.

Haitovsky, Yoel. 1968. « Missing data in regression analysis ». *Journal of the Royal Statistical Society. Series B (Methodological)*, p. 67-82.

Hawkins, Douglas M. 1980. *Identification of outliers*, 11. Springer.

Heckl, Diana, Jürgen Moormann et Michael Rosemann. 2010. « Uptake and success factors of Six Sigma in the financial services industry ». *Business Process Management Journal,* vol. 16, n° 3, p. 436-472.

Hong, GY, et TN Goh. 2003. « Six Sigma in software quality ». *The TQM Magazine,* vol. 15, n° 6, p. 364-373.

ISBSG. 2013. « ISBSG Development and Enhancement Repository R12 ». <http://isbsg.org/project-data/ >.

isixsigma. 2011. « Six Sigma ». < http://www.isixsigma.com >.

isixsigma. 2014. « 1.5 Sigma process shift ». < http://www.isixsigma.com/new-to-six-sigma/dmaic/15-sigma-process-shift15-sigma-process-shift >.

Jacowski, Tony. 2006. « Six Sigma In The Software Industry ». <http://ezinearticles.com/?Six-Sigma-In-The-Software-Industry&id=198268 >.

Janiszewski, Steve, et Ellen George. 2004. « Integrating PSP, TSP, and Six Sigma ». *Software Quality Professional,* vol. 6, n° 4, p. 4-13.

Jiang, Zhizhong, Peter Naudé et Binghua Jiang. 2007. « The effects of software size on development effort and software quality ». *International Journal of Computer and Information Science and Engineering,* vol. 1, n° 4, p. 230-234.

Johnson, Albert, et Beth Swisher. 2003. « Managers at Work: How Six Sigma Improves R&D ». *Research-Technology Management,* vol. 46, n° 2, p. 12-15.

Kuhnt, Sonja, et Jörg Pawlitschko. 2005. « Outlier identification rules for generalized linear models ». *Innovations in Classification, Data Science, and Information Systems*, p. 165-172.

Kwak, Young Hoon, et Frank T Anbari. 2006. « Benefits, obstacles, and future of six sigma approach ». *Technovation,* vol. 26, n° 5, p. 708-715.

Linderman, Kevin, Roger G Schroeder, Srilata Zaheer et Adrian S Choo. 2003. « Six Sigma: a goal-theoretic perspective ». *Journal of Operations management,* vol. 21, nº 2, p. 193-203.

Little, Roderick JA. 1988. « Missing-data adjustments in large surveys ». *Journal of Business & Economic Statistics,* vol. 6, nº 3, p. 287-296.

Little, Roderick JA. 1992. « Regression with missing X's: a review ». *Journal of the American Statistical Association,* vol. 87, nº 420, p. 1227-1237.

Little, Roderick JA, et Donald B Rubin. 2014. *Statistical analysis with missing data*. John Wiley & Sons.

Mahanti, Rupa. 2011. « Software Six Sigma and Cultural Change: The Key Ingredients ». *Software Quality Professional Magazine,* vol. 13, nº 2.

Mahanti, Rupa, et Jiju Antony. 2005. « Confluence of six sigma, simulation and software development ». *Managerial Auditing Journal,* vol. 20, nº 7, p. 739-762.

Mahanti, Rupa, et Jiju Antony. 2006. « Six Sigma in software industries: some case studies and observations ». *International Journal of Six Sigma and Competitive Advantage,* vol. 2, nº 3, p. 263-290.

Mahanti, Rupa, et Jiju Antony. 2009. « Six Sigma in the Indian software industry: some observations and results from a pilot survey ». *The TQM Journal,* vol. 21, nº 6, p. 549-564.

McKnight, Patrick E, Katherine M McKnight, Souraya Sidani et Aurelio Jose Figueredo. 2007. *Missing data: A gentle introduction*. Guilford Press.

Menzies, Tim. 2008. « Improving iv&v techniques through the analysis of project anomalies: Text mining pits issue reports-final report ».

Menzies, Tim, Bora Caglayan, Ekrem Kocaguneli, Joe Krall, Fayola Peters et Burak Turhan. 2012. « The promise repository of empirical software engineering data ». *West Virginia University, Department of Computer Science*.

Mockus, Audris. 2008. « Missing data in software engineering ». In *Guide to advanced empirical software engineering*. p. 185-200. Springer.

Motorola. 2005. « Free Six Sigma Lessons ». <http://web.archive.org/web/20051107013618/http://www.motorola.com/content/0,,3069-5787,00.html# >.

Motorola. 2011. « What is Six Sigma? ». <http://www.motorola.com/web/Business/_Moto_University/_Documents/_Static_Files/What_is_SixSigma.pdf >.

Murugappan, Mala, et Gargi Keeni. 2000. « Quality improvement-the six sigma way ». In *Quality Software, 2000. Proceedings. First Asia-Pacific Conference on*. p. 248-257. IEEE.

Myrtveit, Ingunn, Erik Stensrud et Ulf H. Olsson. 2001. « Analyzing data sets with missing data: An empirical evaluation of imputation methods and likelihood-based methods ». *IEEE Transactions on Software Engineering,* vol. 27, n° 11, p. 999-1013.

Nam, Jaechang. 2014. « Survey on software defect prediction ». *Department of Compter Science and Engineerning, The Hong Kong University of Science and Technology, Tech. Rep.*

Nanda, V., et J. Robinson. 2011. *Six Sigma Software Quality Improvement*. McGraw-Hill Education.

Pan, Zhedan, Hyuncheol Park, Jongmoon Baik et Hojin Choi. 2007. « A Six Sigma framework for software process improvements and its implementation ». In *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*. p. 446-453. IEEE.

Pendharkar, Parag C, James A Rodger et Girish H Subramanian. 2008. « An empirical study of the Cobb–Douglas production function properties of software development effort ». *Information and Software Technology,* vol. 50, n° 12, p. 1181-1188.

Peugh, James L, et Craig K Enders. 2004. « Missing data in educational research: A review of reporting practices and suggestions for improvement ». *Review of educational research,* vol. 74, n° 4, p. 525-556.

Raymond, Mark R, et Dennis M Roberts. 1987. « A comparison of methods for treating incomplete data in selection research ». *Educational and Psychological Measurement,* vol. 47, n° 1, p. 13-26.

Redzic, Cvetan, et Jongmoon Baik. 2006. « Six sigma approach in software quality improvement ». In *Software Engineering Research, Management and Applications, 2006. Fourth International Conference on*. p. 396-406. IEEE.

Roth, Philip L. 1994. « Missing data: A conceptual review for applied psychologists ». *Personnel psychology,* vol. 47, n° 3, p. 537-560.

Rubin, Donald B. 2004. *Multiple imputation for nonresponse in surveys*, 81. John Wiley & Sons.

Saini, Dinesh Kumar, Lingaraj A Hadiman, Poonam V Vaidya et Sanad Al Maskari. 2011. « Software Quality Model Six Sigma Initiatives ». In *Proceedings of the World Congress on Engineering*. Vol. 2.

Saunders, Jeanne A, Nancy Morrow-Howell, Edward Spitznagel, Peter Doré, Enola K Proctor et Richard Pescarino. 2006. « Imputing missing data: A comparison of methods for social work researchers ». *Social work research,* vol. 30, n^{o} 1, p. 19-31.

Schafer, Joseph L. 1997. *Analysis of incomplete multivariate data*. CRC press.

Seow, Christopher, et Jiju Antony. 2004. « Some pros and cons of six sigma: an academic perspective ». *The TQM Magazine,* vol. 16, n^{o} 4, p. 303-306.

Shaout, Dr Adnan, et Dr B El-Haik. 2008. *Software Design for Six Sigma: A roadmap for excellence*. John Wiley Press.

Shenvi, Ajit Ashok. 2008. « Design for six sigma: software product quality ». In *Proceedings of the 1st India software engineering conference*. p. 97-106. ACM.

Siviy, Jeannine M, et Eileen C Forrester. 2004. « Using Six Sigma to Accelerate the Adoption of CMMI for Optimal Results ». In *Six Sigma for Software Development Conference*.

Song, Qinbao, et Martin Shepperd. 2007. « A new imputation method for small software project data sets ». *Journal of Systems and Software,* vol. 80, n^{o} 1, p. 51-62.

Switzer, Fred S, Philip L Roth et Deborah M Switzer. 1998. « Systematic data loss in HRM settings: A Monte Carlo analysis ». *Journal of Management,* vol. 24, n^{o} 6, p. 763-779.

Symons, CR, et A Lesterhuis. 2014. « Introduction to the COSMIC method of measuring software ».

Tayntor, Christine B. 2007. *Six Sigma software development*. Crc Press.

Teng, SJ Jerome. 2008. « The Pros and Cons of Six Sigma Quality Management ». In *International Conference on Advanced Information Technologies (AIT)* p. 10.

Tennant, Geoff. 2001. *Six Sigma: SPC and TQM in manufacturing and services*. Gower Publishing, Ltd.

Tonini, Antonio Carlos, Mauro De Mesquita Spinola et Fernando Jose Barbin Laurindo. 2006. « Six Sigma and software development process: DMAIC improvements ». In *Technology Management for the Global Future, 2006. PICMET 2006*. Vol. 6, p. 2815-2823. IEEE.

Van Hulse, Jason, et Taghi M Khoshgoftaar. 2008. « A comprehensive empirical evaluation of missing value imputation in noisy software measurement data ». *Journal of Systems and Software,* vol. 81, n° 5, p. 691-708.

VanHilst, Michael, Pankaj K Garg et Christopher Lo. 2005. « Repository mining and Six Sigma for process improvement ». In *ACM SIGSOFT Software Engineering Notes*. Vol. 30, p. 1-4. ACM.

Wang, Hongbo. 2008. « A review of six sigma approach: methodology, implementation and future research ». In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*. p. 1-4. IEEE.

Wood, Angela M, Ian R White et Simon G Thompson. 2004. « Are missing outcome data adequately handled? A review of published randomized controlled trials in major medical journals ». *Clinical trials,* vol. 1, n° 4, p. 368-376.

Xia, Wei, Danny Ho et Luiz Fernando Capretz. 2015. « Calibrating function points using neuro-fuzzy technique ». *arXiv preprint arXiv:1508.00028*.

Zhao, Xiaosong, Zhen He, Fangfang Gui et Shenqing Zhang. 2008. « Research on the application of six sigma in software process improvement ». In *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP'08 International Conference on*. p. 937-940. IEEE.