Addis Ababa University

College of Natural Sciences

Maintainability Estimation Model for Object-Oriented Software Design

Melese Bitew Legese

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

July 2019

Addis Ababa University
College of Natural Sciences

Melese Bitew Legese

Advisor: *Dr. Dida Midekso (Asso. Professor)*

This is to certify that the thesis prepared by Melese Bitew Legese, titled: *Maintainability Estimation Model for Object-Oriented Software Design* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality. Signed by the Examining Committee:

	Name	Signature	Date
Advisor:	*Dr. Dida Midekso (Asso. Professor)*		
Examiner:	*Dadgmawi Lemma (PhD)*		
Examiner:	*Ayalew Belay (PhD)*		

Abstract

Object-oriented software design approach is one of the widely used software development approaches due to its ease of maintenance. The quality of the software is highly affected by the quality of the design. The aim of this research is to propose a model that will estimate the maintainability of the software design at early stage of development using multivariate linear regression approach. The maintainability of the design would be measured based on its sub-characteristics. For this work, seven maintainability sub-characteristics are taken to estimate the maintainability of the software design. These are modularity, reusability, analyzability, testability, modifiability, portability and traceability sub-characteristics. The maintainability estimation model is constructed from the class diagram model of the software design. In this work, the controlled experiment and data analysis are performed on class a diagram of XMI schema document. Metrics to estimate the class diagram are found from SDMetrics metric box. In this work, the previously used metrics and sub-characteristics are used. Moreover, additional metrics and sub-characteristics are also integrated to get a better, efficient and more inclusive estimation model. Hence, the experimental analysis has been done on a class diagram of having 125 classes. This makes the model to capture more behaviors of the class attributes. After analysis, for each seven maintainability sub-characteristics, seven prediction models are developed. The statistical significance and fitness of each model has also been analyzed on which showed encouraging evaluation result.

Keywords/phrases: Maintainability Sub-Characteristics, Object-Oriented Metrics, Prediction Model, Quality Attribute

Acknowledgment

Foremost, I would like to express my sincere gratitude to my advisor Dr. Dida Midekso (Asso Professor) for his continues support in my overall MSc study and research, for his patience, motivation, enthusiasm and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. My deep gratitude also goes to all my friends and classmates for helping me in every time when I need their help.

Table of Contents

List of Tables

List of Figures

Acronyms and Abbreviations

ANOVA – Analysis of Variance ISO/IEC FDIS – International Organization for Standardization/ International Electrotechnical Commission/Final Draft International Standard

MLR – Multilinear Regression

MVC – Model View Controller

OO – Object oriented

OOSD – Object Oriented Software Design

PC – Principal Component

PCA – Principal Component Analysis

QUES – Quality Evaluation Systems

SPSS – Statistical Package for Social Science

UML – Unified Modeling Language

UIMS – User Interface Management System

XMI – Xml Metadata Interchange

XML – eXtensible Mark-up Language

Chapter One : Introduction

1.1 Background

Object oriented software design is a software development paradigm that contains quality attributes/characteristics in which designers have to address during the design phase of software development life cycle. According to M. Kumar *et al*., [1] quality is defined as product features, which encountered the requirements of the customers and thus provide product satisfaction after development. These qualities can be either internal qualities, which are concerned with the detail implementation or external qualities which are visible to the users.

In recent year's software development, the productivity is growing in terms of size and complexity relatively in budget and time. In large systems, the software life cycle cost exceeds 80% - 90% of the total budget allocated to the software life cycle cost due to expenditure in software maintenances [2].

Maintainability is just one of the design goals for any engineering process however, that needs to be measured during the design phase of software development life cycle too. Software maintenance is defined as "Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment" [3]. Software maintainability is one of the key characteristics in which the designers could assess at the initial stage of the software system to ensure quality of object oriented software product.

A quality engineer can measure object-oriented quality attributes using a number of object oriented measurement metrics. The problem is not about deficiency of metrics rather selection of appropriate metrics that will fit with the correct quality attribute. Object oriented metrics provides measurement parameters to which one can estimate the complexities and quality related issues of any software design at their early stages of development [4]. Software metrics can be defined by measuring property or characteristic or quality of software objects related to any large and complex software project [4].

Motogna *et al.* [5], proposed maintainability measurement model based on ISO 25010 standard with its sub-characteristics of modularity, reusability, analyzability, modifiability and testability. Accordingly, they proposed an equivalent measurement metrics of Depth of Inheritance Tree (DIT), Weighted Methods per Class (WMC), Coupling Between Objects (CBO) and Tight Class

Cohesion (TCC). In this research, it is tried to enhance additional metrics from the existing identified metrics to measure the internal quality of a software design.

1.2 Motivation

Software maintenance has been under investigation for more than 40 years [6]. Due to the drastic growth of software and hardware technologies, maintainability is a serious issue for software companies and organizations before and after the software product is delivered. Even if maintainability is inevitable that can occur at any time, measuring and evaluating at the beginning of the design phase of the software development will reduce the occurrence of maintenance issues in the future.

In addition, the cost and effort required to maintain a software after development is much more than the development cost and effort. In order to reduce this maintenance cost and effort, it will be worthy to assess the maintainability of the software during the early phase of the software design development.

1.3 Statement of the Problem

Object oriented principals itself to asses and evaluate at an early stage of software development. Object oriented methodologies need a significant effort early in the development life cycle to identify classes, attributes, behaviors and relationship among objects.

Encapsulation, inheritance and polymorphism are also unique feature of object oriented software development in which designers structure carefully the design and their object interaction.

Even though there are a number of UML diagrams in object-oriented software design, only object-oriented UML class diagram is a concern in this research. UML class diagram is one of the software design artifacts that acts as a blue print for the software product. To get quality product, keeping the quality of the design is the major issue for software development [7]. Many of the metrics and quality models are available for assessing the quality of a software design [8, 9, 10]. Assessing the quality of the software product after development is too late to make changes if maintenance is required. Due to this, evaluating and assessing the object-oriented design at early stage makes maintenance simple in terms of effort as well as reduces maintenance cost. Software maintenance has several sub-characteristics, which influence the maintainability of the product. A number of maintainability estimation models and related measuring object oriented metrics have

been proposed by Rai *et al.* [11], Keshamoni and Harikrishna, [12] Punia, *et al.* [13]. Most of the authors have used limited sub-characteristic and most of them are related with little variance.

In ISO/IEC JTC1/SC7 N4098 [14] and Software Quality ISO Standards ISO 9126-1 [15], portability is assumed as a single software quality attribute with sub-characteristics of adaptability. Even if portability is preserved as a sole quality attribute, it is directly affected by the maintainability of software designs. According to IEEE standard for software maintenance [3], traceability is also identified as one sub-characteristic of maintainability. Although these portability, traceability are known as a sub-characteristics of maintainability they do not have a measurement variables. Defining the measurement variables for these sub-characteristics would be one artefact of this research.

In this research, adding these sub-characteristics to the existing Motogna *et al.* [5] assessment approach will enhance the model with aggregation of maintainability sub-characteristic to address the overall determinant factors. Finding a comparative metrics for these sub-characteristics is also part of the research to measure the influence of these sub-characteristics upon maintainability quality attribute.

1.4 Objectives

General Objective

The main objective of this research is to develop maintainability estimation model for object oriented software design.

Specific Objectives

The specific objectives of this research include:

- Reviewing literatures on sub-characteristics of maintainability quality attributes and object oriented metrics related to each quality attribute,
- Measuring the impact of metric upon each identified quality attribute
- Developing a maintainability measurement model and
- Evaluating the model against the existing maintainability models.

1.5 Methods

Data collection

In this process collecting information from all the relevant sources to find answers to the research problem, test the hypothesis and evaluate the outcomes. An important dataset for analysis, analysis tools and any relevant for the research will be collected.

Literature review

In this section, a thorough review of literatures and related research publications will be done. In this review, additional quality attributes that are sub-characteristics of maintainability and related metrics will be investigated.

Hypothesis testing

In this method, we will test an assumption regarding how the chosen maintainability sub-characteristics variables and metrics variables are related. The methodology depends on the nature of the data used and the reason for the analysis. Hypothesis testing is used to infer the result of a hypothesis performed on sample data from a larger dataset.

Model development

The model will be developed through data analysis using some data analysis tools and algorithms. In this experiment for each maintainability sub-characteristics, the appropriate equivalent measurement metrics will be examined.

Evaluation

Evaluating the model is the final stage to show whether the model fits. The evaluation will be conducted using the data analysis tool and expressed empirically.

1.6 Scope and Limitation

The scope of this research is limited only to develop a model for assessing the maintainability quality attribute in terms of its sub-characteristics for object-oriented software design. This research is concerned with extending Montogna's assessment approach [5] by adding additional metrics and additional sub-characters. Traceability and portability are the additional sub-characteristics for this research.

Although, there are different kinds of UML diagrams that can represent the design of object oriented software, due to time and effort required a UML class diagram is the only object-oriented design diagram that used for our model development analysis. Since, evaluating and assessing a software product at the design phase is in class diagram and the metrics are exist for class diagram. In addition, this research focuses only on finding the measurement parameters and their equivalent constants to measure the maintainability sub-characteristics.

1.7 Application of Results

The purpose of this research is to make the object oriented software development evaluation at early stage of software development life cycle with respect to maintainability quality attribute. Changes may come after software product has been produced and deployed due to technologies and business rule alterations. The designer should consider these changes by assessing and evaluating during the design phase of the software development. This reduces maintainability effort and cost of maintenance. Besides existing maintainability models, this research helps the software designers to make sure software design is maintainable by adding further related sub-characteristics and correlated metrics.

1.8 Organization of the Thesis

The remaining parts of the document is organized into five chapters. The second chapter reviews the literatures. Different quality models, model development approaches, metrics models, model evaluation mechanisms are discussed in detail. The third chapter discusses related research works to this research. The fourth chapter details the design of maintainability estimation model. It presents the general process model to construct the estimation model. The fifth chapter discusses the data analysis and results. In this chapter, environmental setups, analysis tools, data preparation, various data analysis approaches and model development techniques are discussed. The seventh chapter discusses the conclusion, contribution and future works.

Chapter Two : Literature Review

2.1 Introduction

The overall literature review is detailed in eight sections. The first section tries to answer the meaning of quality with respect to software development. Even if there exist many scholastic definitions about quality, only two related definitions were selected for this research. The second section demonstrates the different quality models developed previously. These quality models are defined based on quality of the software product. five quality models are described in this section. The third section describes the various kinds of object-oriented metrics that are used to measure the quality of software design. These metrics describe the structural behaviors of object oriented software design. The fourth section describes the various types of methods and techniques that are used in the previously conducted researches to develop quality measurement models. The fifth section describes the various tools used by other authors for experimentation. These tools are capable of analyzing the characteristics of object-oriented software design based on different object-oriented metrics. In the sixth section, the evaluation mechanisms where, how previously developed quality measurement models have been evaluated. The seventh section deals with principal component analysis (PCA). PCA is used to reduce the dimensions of the independent variables and to get the regression coefficients. The eighth section discusses univariate analysis. In this section, each individual independent variables are relevant to the dependent variable would be tested. The last section discusses multivariate regression analysis to construct maintainability estimation model once relevant metrics are recognized.

2.2 Quality Definitions

Before discussing assessment model development for object-oriented software design, it is rational to discuss the purpose why maintainability assessment is required on software design. The main goal of developing the OOSD before implementation is to maintain the quality of the software design. Hence, quality is the main target to be achieved after OOSD assessment model has been developed.

Even if there is nobuniversally agreed definition for software quality, the expression of different scholars according to their philosophical views leads to the same goal. In this paper, it is tried to

refer to prominent and correlated experts' definition of quality management community has provided.

Juran [16] provides two meanings to quality: The word quality has multiple definitions. Two of those meanings overlook the use of the word: Quality consists of those product features which meet the need of customers and thus provide product satisfaction. Quality consists of freedom from insufficiencies. It is the most convenient to standardize on a short definition of the word quality as "fitness for use".

In the same context, Shewhart [17] describes quality in two common aspects of quality. One of them has to do with the consideration of the quality of the material as an objective reality independent of the existence of customers. The other has to do with what we think, feel or sense as a result of the objective reality. In other words there is a subjective side of quality. The author talks about both an objective and subjective side of quality which nicely fits into both "conformance to specification" and "meeting customer needs" definitions. Both authors agree that a product is said to be with quality whenever customers are satisfied and feel comfortable with the product even if quality is subjective.

2.3 Quality Models

A number of quality models for software products were proposed by different scholars. From these quality models, five selected quality models are presented in this section.

2.3.1 McCall's Quality Model

McCall [18] identified some quality attributes and divided them into 3 groups, which are

1) Product revision in which maintainability, flexibility and testability were treated as quality attributes to measure the product revision.
2) Product transition in which portability, reusability and interoperability are the quality attributes to assess the product transitivity.
3) Product operations in which correctness, reliability, efficiency, integrity and usability are the quality attributes to measure the product operability.

Figure 2.1 shows McCall's quality model that grouped software quality attributes in three categories with relative sub-characteristics.

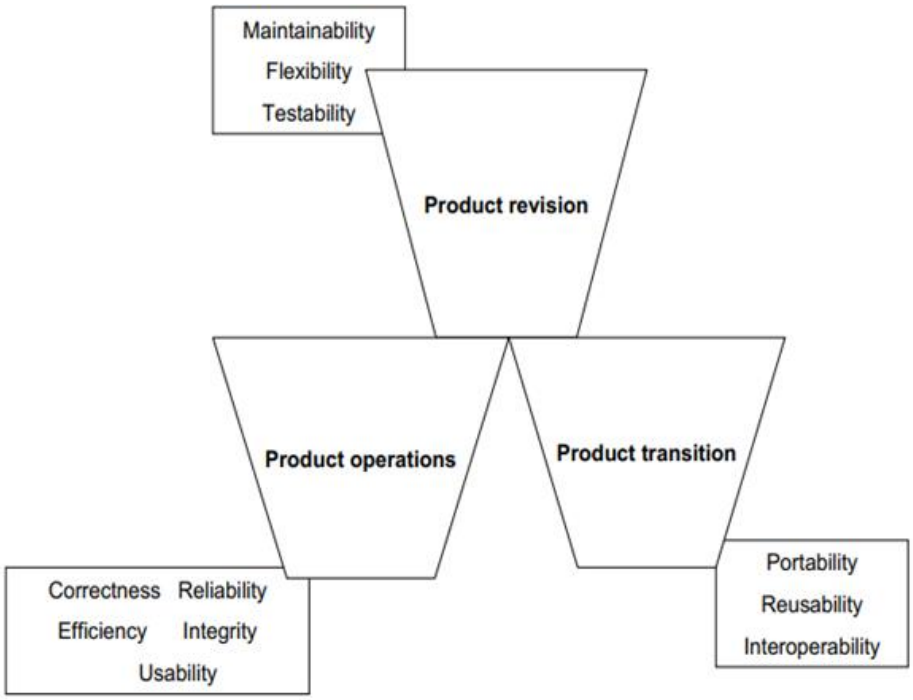


Figure 2.1: McCall's quality model [18]

2.3.2 ISO/IEC FDIS 9126-1 Quality Model

ISO/IEC FDIS 9126-1 [19] defines the internal and external quality model in to six categories.

1) Functionality: The capability of the software product to provide functions which encounter specified and expected needs during software usage under specified conditions.
2) Reliability: The capability of the software product to maintain a specified level when used under specified conditions.
3) Usability: The capability of the software product to be understandable, learnable, memorable, useable and attractive to the user, when used under specified conditions with relative to different users .
4) Efficiency: The capability of the software product to provide appropriate performance, relative to the amount of resources used under stated conditions.
5) Maintainability: The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.

6) Portability: the capability of the software product when transferred from one environment to another. This attribute determines the characteristics of a software product whether it is platform dependent or platform independent.

Figure 2.2 shows ISO/IEC FDIS 9126 quality model. Each quality factor has its own correlated sub-characteristics.

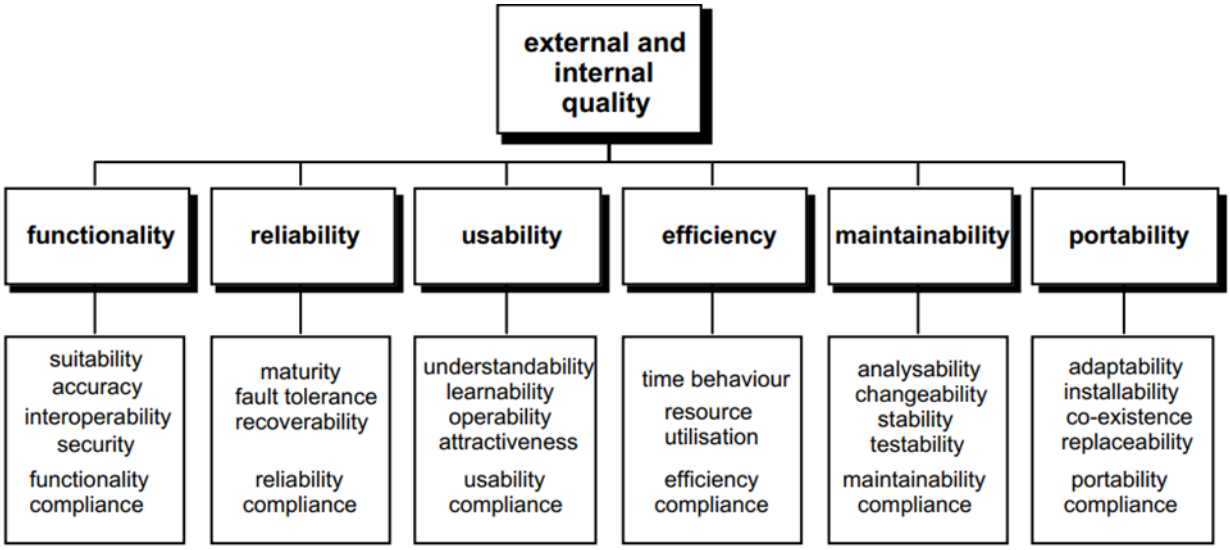


Figure 2.2: ISO/IEC FDIS 9126 Quality model for external and internal quality [19]

2.3.3 ISO/IEC 25010 Quality Model

ISO/IEC 25010 [20] defines quality in to two perspectives:

1) Product quality: In this category of product quality, eight quality attributes are stated. Functional suitability, performance efficiency, compatibility, operability, reliability, security, maintainability and transferability. Each quality attribute has its own sub-characteristics that has the direct impact by the change of one of its related sub-characteristics.

Figure 2.3 shows ISO/IEC 25010 quality model.

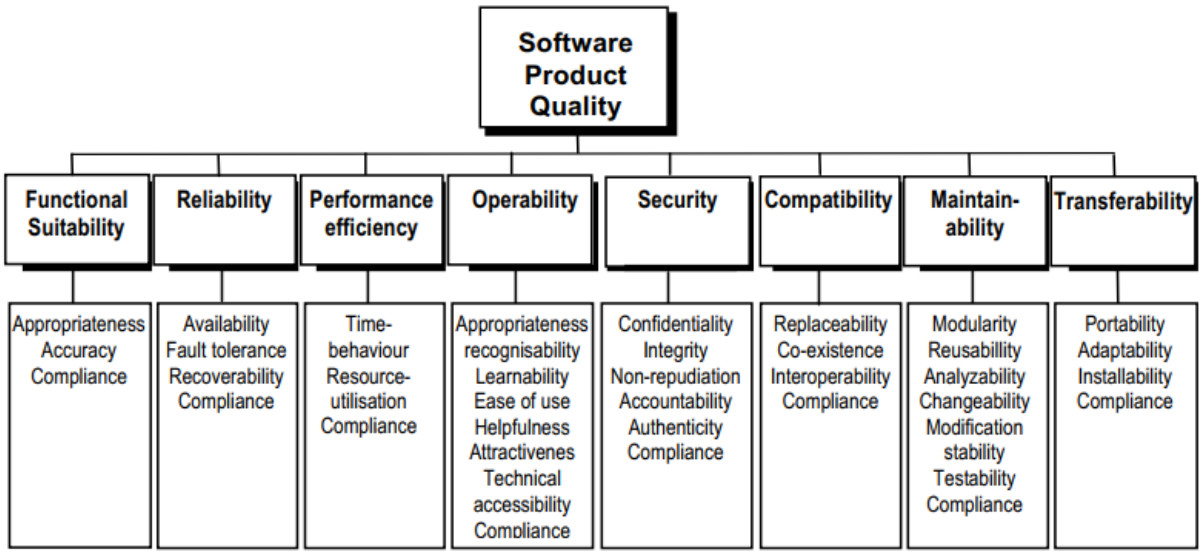


Figure 2.3: ISO/IEC 25010 Software product quality model [20]

2) Quality in use: is the degree to which a product used by specific users meets their needs to achieve specific goals. Quality in use is determined by the quality of the software, hardware, operating environment, and the characteristics of the users, tasks and social environment. All these factors contribute to quality in use. Quality in use can be used to assess the quality of software in a specific context of use.

Figure 2.4 shows ISO/IEC 25010 shows quality model.

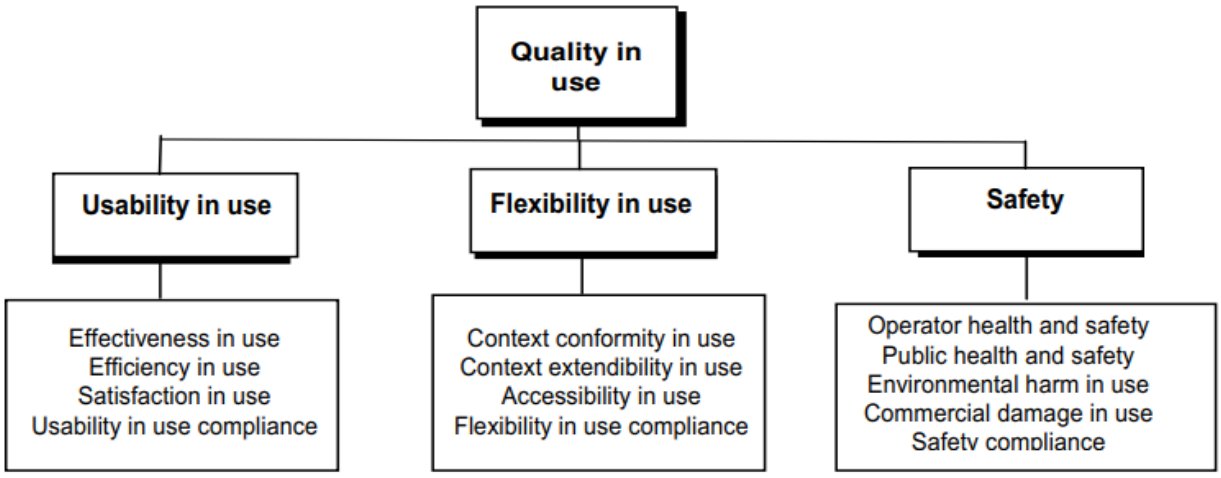


Figure 2.4: ISO/IEC 25010 Quality model for quality in use [20]

2.3.4 Boehm's Quality Model

Boehm [21] addresses the contemporary shortcomings of models that automatically and quantitatively evaluate the quality of software. His model attempts qualitatively to define software quality by a given set of attributes and metrics. The high-level characteristics address three main questions that a buyer of software has

- As-is utility: How well (easily, reliably, efficiently) can I use it as-is?
- Maintainability: How easy is it to understand, modify and retest?
- Portability: Can I still use it if I change my environment?

Figure 2.5 shows Boehm's software quality model.

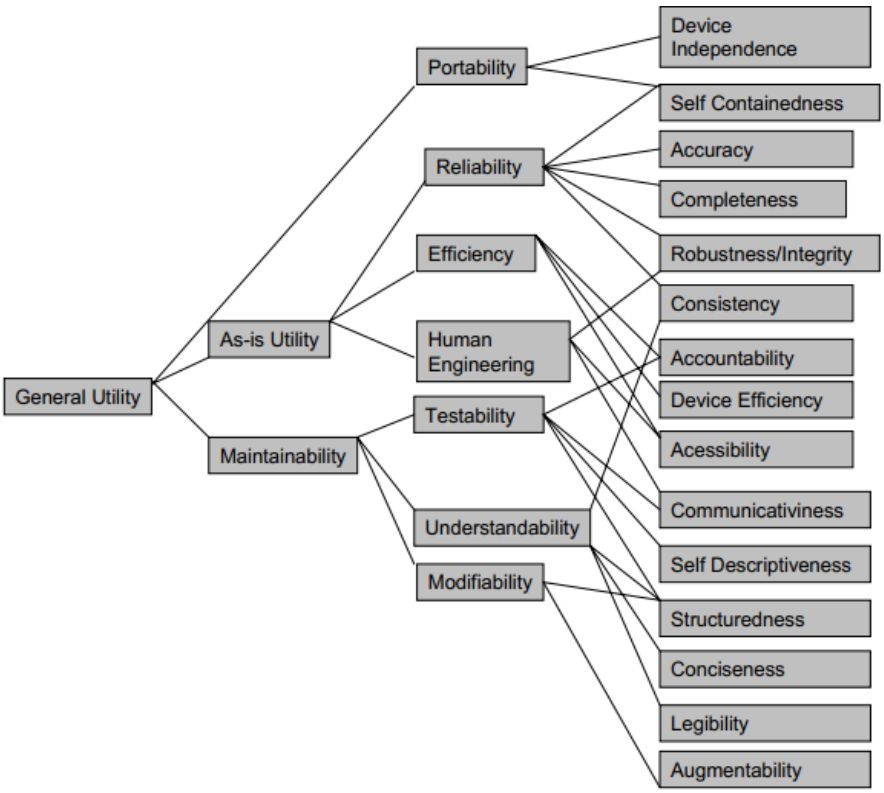


Figure 2.5: Boehm's Software Quality Model [21]

2.3.5 Dromey's Quality Model

Dromey [22, 23] proposed a product based quality model that recognizes that quality evaluation differs for each product and that a more dynamic idea for modeling the process is needed to be flexible to apply for different systems. The author is focuses on the relationship between the quality

attributes and the sub-attributes, as well as attempting to connect software product properties with software quality attributes. Figure 2.6 depicts Dromey's quality model with four quality factors, which are correctness, internal, contextual and descriptive quality factors.

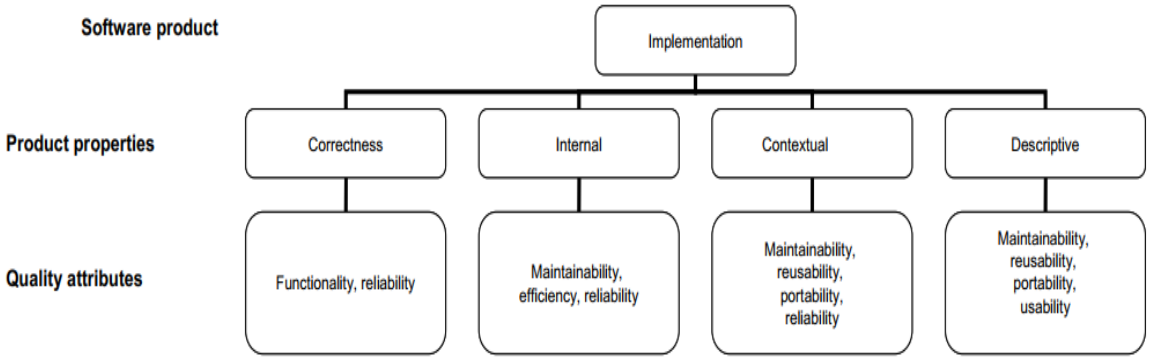


Figure 2.6: Dromey's Quality Model [22, 23]

2.4 Metric models

OO metrics are measurement parameters to achieve quality in software process and product [24]. Software measures are tools to measure the quality of software. The area of software measurement to assess the quality of the software is software metric. These metrics are capable of measuring the specified part of software quality attributes.

Chen Metrics: Mansi *et al.* [24] in analytical study of software metrics cited as, Chen Metrics which is defined as "What is the behavior of the metrics in object-oriented design?" They may describe all of the behaviors like, (i) CCM (Class Coupling Metric), (ii) OXM (Operating Complexity Metric), (iii) OACM (Operating Argument Complexity Metric), (iv) ACM (Attribute Complexity Metric), (v) OCM (Operating Coupling Metric), (vi) CM (Cohesion Metric), (vii) CHM (Class Hierarchy of Method) and (viii) RM (Reuse Metric). All of the terminologies in object-oriented language were considered, as the basic components of the paradigm. These terminologies are objects, classes, attributes, inheritance, method, and message passing.

Morris Metrics: Mansi *et al.* [24] referenced as Morris metric that are described as complexity of the object-oriented system in the form of a depth of a tree. Depth of a tree measures the number of

the sub nodes of the tree. The more number of sub nodes of the tree, the more complex the system is. Therefore, complexity of an object is equal to the depth of the tree or total number of sub nodes.

Lorenz and Kidd Metrics: Lorenz and Kidd metric as cited in [24, 25, 26] divide class-based metrics into four broad categories namely, size, inheritance, internal and external. The metrics are class size (CS)**,** number of operations overridden by a subclass (NOO), number of operations added by a subclass (NOA) and specialization index (SI), average operation size (OS), operation complexity (OC), average number of parameters per operation (NP).

Metrics for Object-Oriented Software Engineering (MOOSE): described Chidamber and Kemerer (CK) metrics *that* have proposed some metrics that have generated a significant amount of interest and are currently the well-known object-oriented suite of measurements for object-oriented software. The CK metrics suite consists of six metrics that assess different characteristics of the object-oriented design are; weighted methods per class (WMC), depth of inheritance tree (DIT), number of children (NOC), coupling between objects (CBO), response for class (RFC) and lack of cohesion in methods (LCOM) as cited in [24, 26, 27]..

Extended Metrics For Object-Oriented Software Engineering (EMOOSE): This model describes metrics of the EMOOSE model. It is an extended form of MOOSE. This model is described with metrics like a) message pass coupling (MPC): It means that the number of messages that could be sent by the class operations. b) Data abstraction coupling (DAC: It is used to count the number of classes, which can be aggregated to current class and defined the data abstraction coupling**.** c) Number of methods (NOM*):* It is used to count the number of operations that are local to the class i.e., only those class operation, which can give the number of methods to measure it. d) Size1*:* It is used to find the number of line of code. *e)* Size2*:* It is used to count the number of local attributes & the number of operation defined in the class as cited in [24, 26, 27].

Metrics for Object-Oriented Design (MOOD): this metric model defines MOOD (Metrics for Object-Oriented Design) metrics as referenced in [24, 26, 27]. In MOOD metrics model, there are two main features. These are methods and attributes. Attributes are used to represent the status of object in the system and methods are used to maintain or modify several kinds of status of an objects. The metrics include, a) Method hiding factor (MHF): It is defined as the ratio of the sum of the invisibilities of all methods defined in all classes to the total number of methods defined in the system under consideration. The invisibility of a method is the percentage of the total classes

from which this method is not visible. b) Attribute hiding factor (AHF): It is defined as the ratio of the sum of the invisibilities of all attributes defined in all classes to the total number of attributes defined in the system under consideration. c) Method inheritance factor (MIF): It is defined as the ratio of the sum of the inherited methods in all classes of the system under consideration to the total number of available methods (locally defined plus inherited) for all classes. d) Inheritance factor (AIF): AIF is defined as the ratio of the sum of inherited attributes in all classes of the system under consideration to the total number of available attributes (locally defined plus inherited) for all classes. e) Polymorphism factor (PF): It is defined as the ratio of the actual number of possible different polymorphic situation. MIF and AIF are used to measure the inheritance of the class and provide the similarity between the classes.

Goal Question Metrics (GQM): This approach was originally defined for evaluating defects for a set of projects in the NASA Goddard Space Flight Center environment. The goal of GQM is to express the meaning of the templates that covers the purpose, perspective and environment; a set of guidelines was also proposed for driving question and metrics. It provides a framework involving three steps [9, 11, 12]

i. List major goals of the development or maintenance project.
ii. Derive from each goal the questions that must be answered to determine if the goals are being met.
iii. Decide what must be measured in order to be able to answer the questions adequately.

Goal (Conceptual level): A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. Objects of measurement are products, processes and resources. Question (Operational level): A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model. Metric (Quantitative level): A set of data is associated with every question in order to answer it in a quantitative way. This data can be objectives and subjective, if they depend only on the objects that can be measured and not on the viewport from which they may be taken.

Quality Model for Object-Oriented Design (QMOOD): The whole description for QMOOD is found from the Bansiya's thesis as referenced in [24, 26, 27], QMOOD metrics can be further classified into two measures:

a) System Measures: System measures describe such metrics as DSC (Design Size in Metrics), NOH (Number of Hierarchies), NIC (Number of Independent classes), NSI (Number of Single Inheritance), NMI (Number of multiple Inheritance), NNC (Number of Internal Classes), NAC (Number of Abstract Classes), NLC (Number of Leaf Classes), ADI (Average Depth of Inheritance), AWI (Average Width of Inheritance), ANA (Average Number of Ancestors).

b) Class Measures: Class measure metrics are those metrics which can define some metrics: MFM (Measure of Functional Modularity), MFA (Measure of Functional Abstraction), MAA (Measure of Attribute Abstraction), MA (Measure of Abstraction), MOA (Measure of Aggregation), MOS (Measure of Association), MRM (Modeled Relationship Measure), DAM (Data Access Metrics), OAM (Operation Access Metrics), MAM (Member Access Metrics), DOI (Depth of Inheritance), NOC (Number of Children), NOA (Number of Ancestor), NOM (Number of Methods), CIS (Class Interface Size), NOI (Number of Inline Method), NOP (Number of Polymorphic Method), NOO (Number of Overloaded Operators), NPT (Number of Unique Parameter Types), NPM (Number of Parameter per Method), NOA (Number of Attributes), NAD (Number of Abstract Data Types), NRA (Number of Reference Attributes), NPA (Number of Public Attributes), CSB (Class Size in Bytes), CSM (Class Size in Metrics), CAM (Cohesion Among Methods of class), DCC (Direct Class Coupling), MCC (Maximum Class Coupling), DAC (Direct Attribute based Coupling), MAC (Maximum Attribute based Coupling), DPC (Directed Parameter based Coupling), MPC (Maximum Parameter based Coupling), VOM (Virtual ability Of Method), CEC (Class Entropy Complexity), CCN (Class Complexity based on Data), CCP (Class Complexity based on method Parameter), CCM (Class Complexity based on Members).

LI W. METRICS: this model proposed six metrics: number of ancestor classes (NAC), number of local methods (NLM): defined as the number of the local methods defined in a class that are accessible outside the class. It measures the attributes of a class that WMC metric intends to capture. Class method complexity (CMC): defined as the summation of the internal structural complexity of all local methods. Number of descendent classes (NDC): is an alternative to NOC is defined as the total number of descendent classes (subclass) of a class. Coupling through abstract data type (CTA): is defined as the total number of classes that are used as abstract data types in the data-attribute declaration of a class. Two classes are coupled when one class uses the other class as an abstract data type. Coupling through message passing (CTM): this element is defined

as the number of different messages sent out from a class to other classes excluding the messages sent to the objects created as local objects in the local methods of the class as cited in [24, 26, 27].

SATC's Metrics: This model has been proposed to select object oriented metrics that support the goal of measuring the code, quality, result. These metrics were used to evaluate the object-oriented concepts like methods, coupling and inheritance and mostly focuses on both of the internal and external efficiency measures of the psychological complexity factors that affect the ability of the programmer. It proposed two traditional metrics and five new metrics for the object-oriented system metrics [24, 26, 27]. The traditional metrics are: a) Cyclomatic Complexity (CC): It is used to measure the complexity of an algorithm in a method of class. b) Line of Code*:* It is a method used to evaluate the ease of understandability of the code by the developer and the maintainer. New Object Oriented Metrics are*:* a) Weight Method per Class (WMC): It is used to count the methods implemented within a class. b) Response for a Class (RFC): It is used to the combination of the complexity of a class through the number of methods and the communication of methods with other classes. c) Lack of Cohesion of Method (LCOM): Cohesion is a degree of methods through which all the methods of the class are inter-related with one another and provide a well bounded behavior. d) Depth of Inheritance Tree (DIT): Inheritance is a relationship between the class that enables the programmer to use previously defined object including the operators and variables. e) Number of Children (NOC): This is used to measure the subclass subordinate to a class in the hierarchy.

2.7 Principal Component Analysis

Principal component analysis (PCA) is a standard statistical tool used in analyzing multidimensional data [28]. It is widely used in almost all areas of research where manipulation of large numbers of attributes is necessary. It is a non-parametric method useful for obtaining relevant information from a complex data set. PCA is used to reduce the dimensionality of a data set, which consists of a large number of interrelated attributes, while retaining as much of the variation present in the original data set as possible [29, 30]. This process was done by linear transformation of the original set of attributes into a smaller set of attributes called principal components (PCs). Principal components are uncorrelated and ordered so that the first few retain most of the variation present in all of the original attributes.

According to Abdi and Williams [31], the goals of PCA are to:

1) Extract the most important information from the dataset table,

2) Reduce the size of the data set by keeping only the important information,

3) Simplify the description and interpretation of the dataset and

4) Analyze the structure of the observations and the variables.

2.8 Univariate Analysis

Univariate analysis is conducted for making data easier to interpret and to understand how data is distributed within a sample or population being studied. It is used to screen and evaluate the data whether it meets required assumptions for other complex statistical analyses. As such, they are often reported as preliminary analyses in research studies that address the complex interrelations and interactions between multiple variables that are commonly investigated in quality of life studies. These analyses provide us with descriptions of single variables we are interested in using in more advanced tests and help us narrow down exactly what types of bivariate and multivariate analyses we should carry out. According to Briand *et al.* [32], univariate regression analysis is conducted for these purposes:

1. To test the hypotheses;
2. To screen out the irrelevant measures which are not significantly related to each dependent variable. Only relevant measures that are significant at significance level a=0.05 are considered for the subsequent multivariate analysis and
3. To predict behavior and improving forecasting of likely outcomes using predictive tools.

2.9 Multivariate Regression Analysis

Multivariate multiple linear regression analysis (MMLR) is used to investigate the relations of object-oriented metrics and software maintainability quality attributes. Computationally, MMLR gives the same coefficients, standard errors, t-test and p-values and confidence intervals, as one would estimate with individual MLR computations for each of the dependent variables separately [33].

The effective use of such methods: (a) as an exploratory tool to investigate patterns in the data; (b) to identify natural groupings of the population for further analysis; and (c) to reduce dimensionality in the number of variables involved. We used these as preliminary steps that lead to the construction of maintainability dependent variables [34].

The multiple linear regression equation is as follows:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta 2X2 + \beta_3 X_3 + \ldots + \beta_c X_c \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Here, $\hat{Y}$ is the predicted or expected value of the dependent variable, X1 through Xc are distinct independent or predictor variables, β_0 is the constant value of Y when all of the independent variables (X_1 through Xc) are equal to zero, and β_1 through β_c are the estimated regression coefficients. Each regression coefficient represents the change in Y relative to a one-unit change in the respective independent variable.

Chapter Three : Related work

3.1 Introduction

In this Chapter, various maintainability estimation models are investigated in depth. The model are grouped into categories based on their development approach and technique.

3.2 Regression Based Models

Regression analysis is an important statistical technique to explain the relationships between the independent variables and dependent variables. In this technique, the increase in one variable changes the other variable. The coefficient of parameters are estimated with the help of mathematical technique. The technique is used for modeling and analysis of numerical data and it is an important technique for simplicity and fruitful interpretation [35].

Kiewkanya *et al.* [36] presented a maintainability model of class diagram and sequence diagram using two sub-characteristics of maintainability: understandability and modifiability to evaluate maintainability of a design. The main purpose of this research is to propose a methodology for constructing maintainability estimation model using three techniques of metrics-discriminant, weighted-score-level and weighted-predicted-level.

The experiment is conducted on 40 software design models with dissimilar domains. Twenty questions were prepared to examine the understandability of the design and 10 questions have been prepared to examine the modifiability of the given model. The questions are prepared based on the target to examine understandability and modifiability using the structure, behavior and some elements of the design. The questions and randomly selected models were given to each group. Subjects were asked to add new functionalities from the existing design and remove functionalities within the given time. Then, the data was collected and analyzed from this pilot test. Maintainability model was calculated from the summation of understandability and modifiability.

Metrics selected to assess a class diagram are related to classes, attributes, methods and relationships. Metrics selected to assess sequence diagram are related to scenarios, messages and conditions. Once metrics were identified, the experimental tools are used. Unisys Rose XML Tool, to export the diagram into runnable format of XML document. This XML document was used as an input to the automated tool for examination of the diagram. The tool was automated in java programming language.

The methodology to construct maintainability used different techniques. The first technique is constructing maintainability model using metrics-discriminant technique. In this technique, important metrics were selected by analyzing the pattern of correlation between maintainability levels and design complexity metrics by applying discriminant analysis. These metrics are analyzed using statistical techniques in order to determine which metrics can be indicators of maintainability. Linear regression with one independent variable is performed for each metric to determine the relevance. The second technique is constructing maintainability model using weighted-score-level technique. This technique was applied by combining understandability and modifiability levels, converted from understandability and modifiability scores, using a weighted sum method. The third technique is constructing maintainability model using weighted-predicted-level technique; this technique was applied by combining predicted understandability and modifiability levels, obtained from applying understandability and modifiability models, using a weighted sum method. The first step of this technique is constructing understandability and modifiability models in the same way. After constructing understandability and modifiability models, the next step is constructing maintainability model in a form of a weighted sum equation of both models. After the model is designed, the evaluation is done through comparison of actual values calculated by the model with examination from the questions.

Rizvi and Khan [37] developed a multivariate linear assessment model called maintainability estimation model for object-oriented (MEMOOD) software in design phase to estimate the maintainability of UML class diagram in terms of understandability and modifiability sub-characteristics. In this research, understandability and modifiability of class diagrams were quantified in terms of size and structural complexity metrics of class diagrams. All the three quality attribute models were developed using the technique of multiple linear regression.

Afterwards, eleven object-oriented metrics were selected from previous researches and validated empirically. These metrics are NC (Number of Classes), NA (Number of Attributes), NM (Number of Methods), NAssoc (Number of Associations), NAgg (Number of Aggregations), NDep (Number of Dependencies), NGen (Number of Generalizations), NAggH (Number of Aggregation Hierarchies) NGenH (Number of Generalizations Hierarchies), MaxDIT (Maximum DIT; DIT the value of a class is the longest path from the class to the root of the tree) and MaxHagg (Maximum Hagg, Hagg is the value of the longest path from the class to leave).

The relevant dataset was collected from other two related models developed previously. The study used this dataset for fitting two separate multivariate linear regression models for class diagram's understandability and modifiability individually. Taking object-oriented class diagram metrics that were identified for the model that are used as independent variables. Understandability and modifiability were used as dependent variables. The quantification of understandability and modifiability is the first task to get the value of maintainability estimation model. All the three models are quantified using multivariate linear regression technique.

The authors have validated the model in separate way. Understandability, modifiability and maintainability models were calculated independently by means of perspective model's equation. Therefore, Pearson's correlation coefficient is being calculated between the calculated (using model equation) and actual (already known) values of modifiability, understandability and maintainability.

Gautam and Kang [38] developed a multivariate linear model compound maintainability estimation model for object-oriented software in design phase (Compound MEMOOD) to estimate the maintainability of class diagram in terms of understandability, modifiability taken from previous research and including additional sub-characteristics, which are scalability and level of complexity**.** This model is an extended version of MEMOOD developed by Rizvi and Khan [30]. The model takes scalability, level of complexity, understandability and modifiability as independent variables while maintainability is taken as dependent variable.

The authors selected relevant metrics for the model using some criteria like, extensive recent literature survey as well as cross and within company data set which were chosen with the help of experts. Consequently, NGen (Number of Generalizations), NGenH (Number of Generalizations Hierarchies), NAggH (Number of Aggregation Hierarchies) are taken as modifiability model measurement metrics. NGen and NAggH were taken as understandability measurement metrics. Ca (Afferent Coupling), Ce (Efferent Coupling), percent coverage, distance from main sequence and nesting depth are taken as scalability measurement metrics. Coupling, cohesion, cyclomatic complexity and ILCC were also taken as complexity measurement metrics.

A dataset used for experimentation was derived from open source projects and from local software companies. The model is proposed using linear regression technique. In this model, metrics were taken as independent variables and sub-characteristics are taken as dependent variables. Finally, a

compound MEMOOD model was proposed which takes scalability, level of complexity, understandability and modifiability as independent variables and maintainability model is taken as dependent variable.

Hincheeranan and Rivepiboon, [39] presented quality factors that are flexibility and extendibility sub-characteristics of maintainability as measuring criteria to establish a maintainability estimation model. Multivariate linear regression was used to establish the maintainability estimation model and develop the maintainability estimation tool (MET) to estimate maintainability of a class diagram at the design phase of software development.

Metrics chosen for estimating maintainability in this model were based on eleven design properties. For each eleven design properties, eleven metrics (one for each) were chosen. These metrics were DSC (Design Size in Class), NOH (Number of Hierarchies), ANA (Average Number of Ancestors), DAM (Data Access Metric), DCC (Direct Class Coupling), CAM (Cohesion Among Methods in Class), MOA (Measure of Aggregation), MFA (Measure of Functional Abstraction), NOP (Number of Polymorphic Methods), CIS (Class Interface Size), NOM (Number of Methods).

The experiment was conducted using the tools StarUML tool to develop class diagram. This tool supports to export into XMI file format that acts as an input file for calculating a maintainability estimation of class diagram. XMI parser tool is used for this component to analyze class diagram by read element and attribute of XMI (Metadata Interchange (XMI) is an open standard file format that enables the interchange of model information between models and tools.

Kiranjit and Sami [40] tried to design a model of metrics for object oriented design (MOOD). In this research, a multivariate linear model was used to estimate the maintainability of a class diagram of reliability and portability are the sub-characteristics of maintainability. This model is an extended maintenance model previously designed in terms of modifiability and understandability sub-characteristics. For this research, metric dataset was taken from MOOD metrics for reliability and portability measurement.

3.3 Artificial Neural Network Based Models

Bhutani and Chug [41] tried to evaluate the prediction capability of three neural network based algorithms quantitatively. These models are, a) group method of data handling (GMDH): This is

a machine learning algorithm which builds a polynomial function. b) General regression neural network (GRNN): This is also a machine learning algorithm, which is a multi-layered feed forward network. c) Probabilistic neural network (PNN): This is a memory based network. This algorithm has a parallel structure and is used for solving regression based problems.

The metrics identified to evaluate the three models are: weighted methods per class (WMC), number of children (NOC), coupling between classes (CBO) and depth in inheritance (DIT) which were taken from Chidambar and Kemerer [42] suite of metrics as well as lines of code (LOC), McCabe's [43] cyclomatic complexity (CC) are taken from the traditional metrics.

The experiment was conducted within two versions of open source software containing 38 classes implemented in java programming language. The metrics are independent variables. The change was dependent variable upon the metrics. The six metrics are measured using CCCC (C and C++ Code Counter) tool.

The accuracy of the models was measured through magnitude of relative error (MRE). It is measured by taking the absolute value of the difference between the actual value and the predicted value, mean magnitude of relative error (MMRE) and Pred. that is measured by the predicted values whose MRE is less than or equal to a specified value. Finally, the authors concluded that, GRNN is the better model for prediction of software maintainability.

Dash *et al.* [44] proposed a maintainability prediction model using multi-layer perceptron (MLP) neural network approach to develop the model. This approach seems to be better than the models developed in artificial neural network. This approach encourages the network to train efficiently by transferring the raw data into principal components.

Metrics used for analysis in the first principal component were RFC, LCOM, WMC, NOM and SIZE2; DAC and SIZE1 were in the second principal component and DIT and NOC were in the third principal component. In order to estimate maintenance effort, UIMS dataset was used in this study. The evaluation was done by comparing with another model developed in neural network models. The goodness of fitting the model was measured by using the coefficient of correlation(r) and mean absolute error. The correlation between the predicted change and the observed change was represented by the coefficient of correlation (r).

3.4 Fuzzy Logic Based Models

Diwan [45] proposed based on one of the soft computing modeling mechanism of fuzzy logic technique. In this research, external characteristics such as analyzability, changeability, testability, stability, modularity, and self-descriptiveness were considered as sub-characteristics of maintainability quality factor. The maintainability model was developed from fuzzy models using fuzzy interference system (FIS).

In order to formulate rules, survey from 18 experts were conducted. Out of these experts, nine were working in software industries and the remaining nine were PhD candidates. On the basis of opinion from the experts, rules were formulated where fuzzy operator was used. A total of 729 rules were made. After formulating all the rules centroid implication method was used to calculate the aggregate of all the inputs.

Mishra and Sharm [46] studied the various soft computing techniques to implement software maintainability and implemented the result by adaptive neuro fuzzy technique, which uses QUES training data for predicting the value of change variable. The size of the dataset to be analyzed were 71. The dataset that was analyzed is normalized in order to get accurate result during maintainability prediction to the values within range. This normalization eliminates the problem of over-fitting the data while training the network. Once the dataset is normalized, network model is trained with the data of QUES for which prediction in the UIMS is to be done. After this, once the training of the network is done it is provided with the testing data. Using this, input data is used as training data that the network predicts the output of testing dataset. Comparing the trained value trends with the testing output trends to calculate the error in prediction. This process is done for all the data available. At the end, the average error is calculated to analyze and evaluate the efficiency of the network.

3.5 Rule and Algorithm Driven Models

Motogna *et.al.* [5] Proposed an approach to assess the maintainability quality attribute by its own sub-characteristics as defined by ISO 25010 [20]. These sub-characteristics are modularity, reusability, analyzability, modifiability and testability. For these sub-characteristics, four object-oriented metrics are chosen to assess the maintainability of the OOSD. The metrics are DIT (Depth of Inheritance Tree), WMC (Weighted Methods per Class) CBO (Coupling Between Objects) and

TCC (Tight Class Cohesion). The metrics are identified from previous research works. The impact of these metrics upon the sub-characteristics is controlled by some rules and algorithm that can capture these aspects. These metrics then estimate the change in maintainability of the UML class diagram.

The experiment was conducted through the study of two versions (v2.1 and v3.2.2) of the ASP.NET MVC Webstack project. These versions are different in some parts of the design. The metrics were computed using NDepend tool. The experimental code was written in visual studio environment. Still, it needs further to capture all sub-characteristics of maintainability and OO metrics. The status shows that some of the relations between OO metrics and quality characteristics have not been investigated as well an integrated maintainability estimation tool is not developed yet. In this paper, the measurement metrics are very limited. Adding an extra metrics will improve the efficiency of the model.

Bansiya and Devis [9] proposed an improved hierarchical model for the assessment of high-level object-oriented design attributes. This model evaluates the structural and behavioral design of classes, objects and relationships using a suite of object-oriented metrics. In addition, the model relates the design properties like, encapsulation, modularity, cohesion and coupling with the software quality attributes of reusability, flexibility, understandability, functionality, extendibility and effectiveness.

The authors anticipated a methodology to develop a hierarchical QMOOD (Quality Model for Object-Oriented Design) assessment model using four levels and three level mappings to link the four levels. In the first level, design quality attributes were identified. In this level reusability, flexibility, understandability, functionality, extendibility and effectiveness quality attributes were identified to develop the model. In the second level, object-oriented design properties were identified. In this level abstraction, encapsulation, coupling, cohesion, complexity and design size were identified as the design properties that should be considered at a time of the model's construction. In the third level, object-oriented metrics were identified. In this level, eleven object-oriented metrics were recognized to assess the quality of the software design. These metrics are DSC (Design Class in Size), NOC (Number of Hierarchies), ANA (Average Number of Ancestors), DAM (Data Access Metric), DCC (Data Class Coupling), CAM (Cohesion among Method of class), MOA (Measure of Aggregation), NOP (Number of Polymorphic Methods),

MFA (Measuring of Functional Abstraction), CIS (Class Interface Size) and NOM (Number of Methods). In the fourth level, object-oriented design components are identified. In this level, classes, objects and relationships are the main components that describe the architecture of object-oriented software design.

Once essential elements required to develop the model were identified, the next step is connecting the levels to get an aggregated high-level model. The first linkage is mapping quality carrying component properties with design properties. In this association, the fundamental quality carrying components like classes, attributes and methods were mapped with design properties. In the second linkage mapping design metrics to design properties. In this association, the relevant object-oriented metric to measure design properties based on the impact towards the design properties was identified and mapped. The third linkage is the linkage between design properties and quality attributes. This linkage helps to identify which design property expresses/affects which quality attributes. This linkage has been done by an extensive review of object-oriented books and publications.

Finally, the model was evaluated in two levels: evaluating individual quality attributes evaluation and aggregate model evaluation. Evaluating individual quality attributes is done by taking the continuous release of software versions of the designs, analyze the change and compare with each framework releases alongside QMOOD model. QMOOD evaluation is an aggregate evaluation as an overall quality of the model. Validation of QMOOD of predictability was based on comparison of several object-oriented designs that had been developed previously for the same requirements. This model is more inclusive model capable of measuring the different quality attributes of OOSD. Still, it needs to include additional sub-characteristics to get an efficient estimation model.

3.6 Summary

In many of the maintainability models, the development has been done using limited number of sub-characteristics of maintainability quality factors. The existence of multiple number of models but limited number of sub-characteristics over maintainability estimation becomes difficult for developers to come up with multi-platform model to measure maintainability of the OOSD. Modifiability and understandability are the most frequently used sub-characteristics to measure the maintainability of the software design. Identifying the remaining sub-characteristics of maintainability quality factors and adding them to the existing models will help to come up with

single and more inclusive model. In addition, identifying equivalent metrics for these new sub-characteristics and adding new metrics to the existing model will improve the efficiency of the model to measure and estimate the maintainability of the OOSD.

According to Motogna, *et.al.* [5], the assessment approach is done based on ISO 25010 maintainability sub-characteristics. ISO 25010 defines maintainability of a software is attributable to modularity, reusability, analyzability, changeability, modification, stability, testability and compliance. Among these sub-characteristics, the authors used five of the sub-characteristics modularity, reusability, analyzability, modification, testability. For this model, only four object-oriented metrics were used to construct the model. However, additional metrics are required to construct highly efficient and more inclusive model. In addition, portability and traceability of the software design have a great impact during software maintenance. These sub-characteristics were not considered in the models developed earlier. Therefore, adding portability and traceability sub-characteristics to Motogna's model with their relative metrics is the main focus of the current research we will also explore additional object-oriented metrics for the existing model.

Chapter Four : Maintainability Variables

4.1 Introduction

In this Chapter, the detail of the proposed solution for maintainability assessment of a software design is presented. The chapter is organized into three sections. The first section discusses the quality factors that are identified as sub-characteristics of maintainability quality attribute. These sub-characteristics have an impact over the software design during software maintenance and need to be considered. In the second section, object-oriented software design metrics are identified and discussed in detail. These metrics are used as measurement parameters for each sub-characteristics. These measurement metrics are the independent variables of the model being developed. The third section discusses the proposed methodology of analysis and procedures to develop maintainability prediction model.

4.2 Maintainability sub-characteristics (Dependent variables)

Except Montogna's model which contains five sub-characteristics most of the reviewed models were developed by the combination of two or three sub-characteristics. Moreover no model has described portability and traceability sub-characteristics even if traceability was considered as a thread from the requirements, design and analysis to the implementation [15, 3], it can be applied as part of design assessment and need to be integrated as design assessment factor. Here, the thread of the software design should be clear and understandable for developers to review the design in case needed when modification, validation, testing and reuse of the software product. On the other hand, portability of a software product needs to be considered during design assessment. The design may contain a number of interfaces for either extending with additional features or interacting with other systems. In order to handle designs with such kind of properties, measuring the level of portability is inevitable. Adding this sub-characteristic to the existing model will enhance the productivity of maintainability prediction model being developed.

Even though there exists a number of maintainability measurement models that are capable of predicting the maintainability of the design with in limited number of quality factors, collecting these different models together will help to create a single more inclusive and efficient measurement model. Figure 4.1 shows quality factors, that are directly related with maintainability quality attribute. The proposed maintainability prediction model will be designed based on these factors.

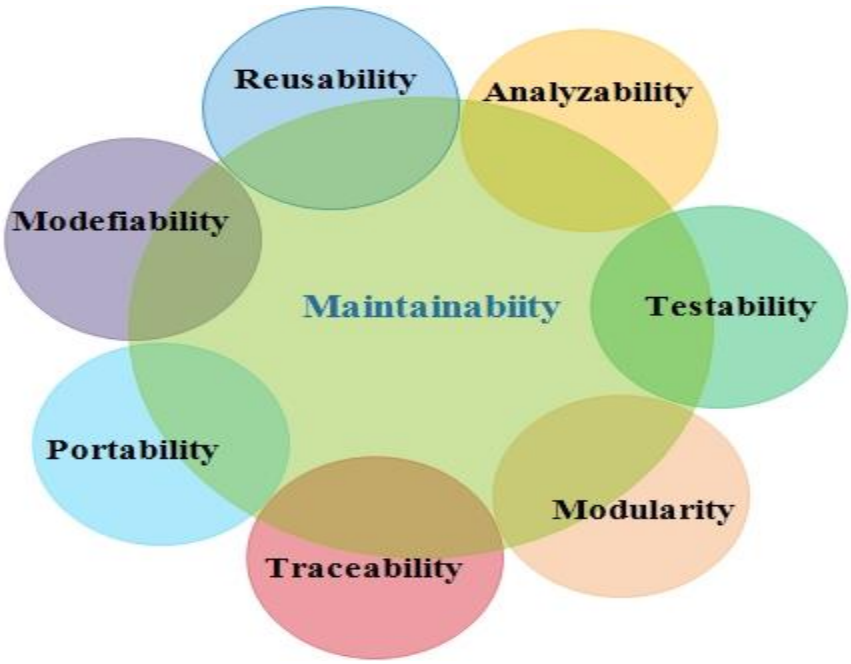


Figure 4.1: Maintainability sub-characteristics

4.3 Object-oriented Metrics (Independent Variables)

Design size and complexity, inheritance and coupling are some of the object-oriented design properties. The metrics chosen to develop the prediction model are based on these design properties. The relevant metrics required to develop the model have been gathered through systematic reviews from previous researches and object-oriented metrics models. The previously used metrics that are found from the SDMetrics model [47] are taken as they are and new metrics that were considered to measure the sub-characteristics are included. Table 4.1 shows the overall metrics selected for maintainability prediction model. Table 4.2 lists previously used metrics while Table 4.3 depicts additional metrics to be used in the current research.

Table 4.1: Overall metrics selected for maintainability prediction model

Category	Domain	Metrics
Inheritance	Class/ Abstraction	IFImpl - The number of interfaces the class implements OpsInh: Number of Operator Inherited AttrInh: Number of Attributes Inherited NOC - The number of children of the class

Category	Domain	Metrics
	Class/ Generalization	CLD: Class to leaf depth. The longest path from the class to a leaf node in the inheritance hierarchy below the class.
		NumDesc -The number of descendents of the interface
	Class/ hierarchy	DIT- The depth of the class in the inheritance
		NumAnc - The number of ancestors of the class
		Ca- Afferent coupling
		Ce- Efferent coupling.
Coupling	Package/Class	Dep_Out: The number of elements on which this class depends.
		Dep_In: The number of elements that depend on this class
		MsgSent: The number of messages sent.
		MsgRecv: The number of messages received.
	Class	NumAssEl_ssc: The number of associated elements in the same scope as the class.
		NumAssEl_sb: The number of associated elements in the same scope branch as the class.
		NumAssEl_nsb: The number of associated elements not in the same scope branch as the class.
		EC_Attr: The number of times the class is externally used as attribute type.
		IC_Attr: The number of attributes in the class having another class or interface as their type.
		EC_Par: The number of times the class is externally used as parameter type.
		IC_Par: The number of parameters in the class having another class or interface as their type.
	Class	NumAttr- The number of attributes in the class
	Class	NumOps- The number of operations in a class.
Size	Class	NumPubOps - The number of public operations in a class.
	Packages	NumCls - The number of classes in the package.
	Packages	NumInterf- The number of interfaces in the package

Category	Domain	Metrics
	Package	NumCls_tc- The number of classes in the package, its subpackages, and so on.
	Package	NumOpsCls - The number of operations in the classes of the package.
	Class	Setters- The number of operations with a name starting with 'set'.
	Class	Getters-The number of operations with a name starting with 'get', 'is', or 'has'.
Nesting	Class/Package	The nesting level of the class (for inner classes).

Table 4.2: Previously used metrics for maintainability prediction model

Metrics	Description
NOC	The number of children of the class
DIT	The depth of the class in the inheritance
NumAnc	The number of ancestors of the class
Ca	Afferent coupling
Ce	Efferent coupling
NumAttr	The number of attributes in the class
NnumOps	The number of operations in a class
NumCls	The number of classes in the package
CLD	Class to leaf depth
NumDesc	The number of descendants' of the interface

Additional metrics, that were not used previously by other authors during maintainability prediction model development, are listed in Table 4.3. These metrics are again found in the SDMetrics model [47]. The metrics are stated as characteristics for an object-oriented class diagram. Hence, these metrics should have to be integrated in the maintainability estimation model of a software product at class level design to increase the effectiveness of the model.

Table 4.3: Additional metrics for maintainability estimation model

Metrics	Description
NumPubOps	The number of public operations in a class
IFImpl	The number of interfaces the class implements
OpsInh	Number of Operator Inherited
attrInh	Number of Attributes Inherited
NumAssEl_ssc	The number of associated elements in the same scope as the class
NumAssEl_sb	The number of associated elements in the same scope branch as the class
NumAssEl_nsb	The number of associated elements not in the same scope branch as the class
EC_Attr	The number of times the class is externally used as attribute type
IC_Attr	The number of attributes in the class having another class or interface as their type
EC_Par	The number of times the class is externally used as parameter type
IC_Par	The number of parameters in the class having another class or interface as their type
NumInterf	The number of interfaces in the package
NumCls_tc	The number of classes in the package, its subpackages, and so on.
NumOpsCls	The number of operations in the classes of the package.
Setters	The number of operations with a name starting with 'set'.
Getters	The number of operations with a name starting with 'get', 'is', or 'has'.
Dep_Out:	The number of elements on which this class depends.
Dep_In:	The number of elements that depend on this class
MsgSent:	The number of messages sent.
MsgRecv:	The number of messages received.
Nesting	The nesting level of the class (for inner classes).

4.5 Analysis and Procedures

In this section, the research analysis and procedures would be presented in detail. Variety of components that will be applied in order to address the research output are presented.

4.5.1 Process model construction

The general proposed process model depicted in the Figure 4.2 has five main components inorder to develop our maintainability prediction model.

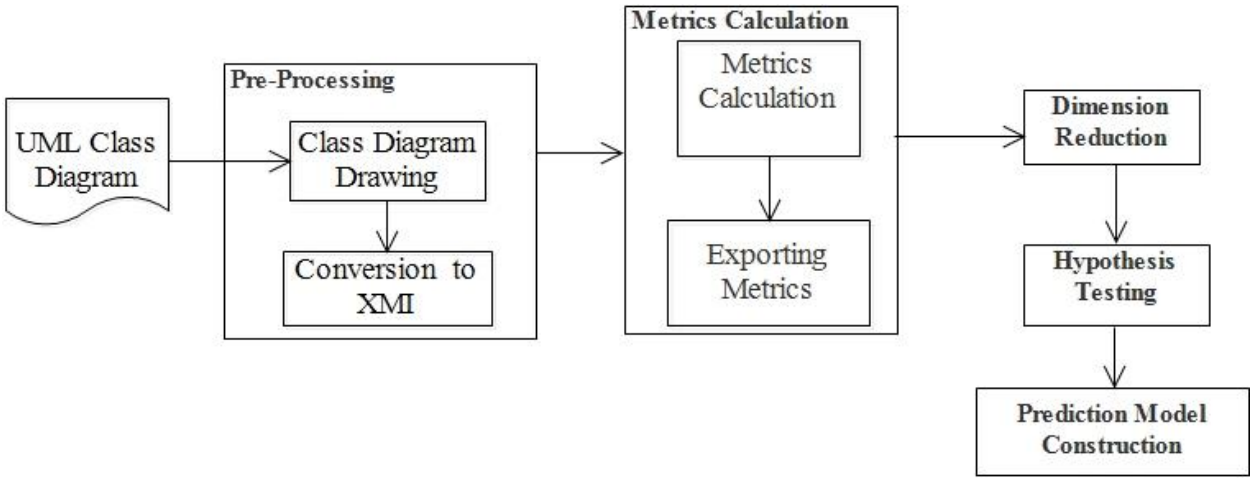


Figure 4.2: Process model for the model construction

4.5.1.1 Pre-processing Component: In this component, the data required for analysis is organized and prepared in the desired format for experimentation. The class diagram is taken from open source project documents either in image file or in XMI file. If the project class diagram is found as image file, it has to be transformed to another file format called XMI document.

4.5.1.2 Metrics Calculator Component: This component takes the preprocessed XMI file, calculates the metrics and generates discrete values for each metrics. In this component, the metrics measurement model only calculates the values of target metrics required to know the values. Once metrics are calculated, the values will be exported to the required format type for further processing.

4.5.1.3 Dimension Reduction Component: in is component only the most observed independent variables would be captured. The less significant variables in the dataset will be removed from the model analysis. This component enables to reduce the less observed variables from the model and keeps more observed variables for further analysis.

4.5.1.4 Hypothesis Testing Component: In this component, the significance of each individual hypothetical metrics will be tested against maintainability sub-characteristics. Only the relevant dimensions, which are passed from dimension reduction component will be entered for testing the hypothesis.

4.5.1.5 Prediction Model Component: In this component, collection of metrics will be measured against maintainability sub-characteristics. Metric, which passed from hypothesis testing in wrong way, will be captured at this component. At this component, the significant variables will be part of the model and the less significant variables will be left out of the model.

Chapter Five : Maintainability Prediction Model

This chapter presents document tools, experimental procedures and prediction model construction in detail.

5.1 Development tools

This Section discusses the experimental tools used in this research. Most tools were open source tools and they were not purchased. The tools used in this research were:

*IBM SPSS***:** used to analyze data:

Microsoft Excel 2016: used to present data for inputting.

*Matlab 2016b**:*** This tool used for data analysis.

*SD*Metric Tool: used to parse the XMI file and calculate metrics values.

*StarUML**:*** is a design modeling tool used to draw a class diagram and export to XMI file.

5.2 Hardware Environment

A personal computer used for this research has the following specifications.
Processor: Intel(R) Core(TM) i3-5005U CPU
Processor speed: 2.00 GHz, 2000 MHz,
Number of Processors: 2 Core(s), 4 Logical Processor(s)
Operating System: Microsoft Windows 10 Pro
System Type: x64-based PC
Installed Physical Memory (RAM): 4.00 GB

5.3 Experimental procedures

5.3.1 Data preparation

The required data for this research is obtained from SDMetrics project [47] which contains 125 classes. SDMetrics project is an open-source XMI parser tool that calculate a metrics value. In this research, we had identified 31 metrics as detailed in Chapter Four. These metrics are selected for only class diagram. Even if there are more number of metrics, which describe the behavior of the class diagram more frequently known metrics are chosen.

5.3.2 Descriptive statistics

From the identified 31 metrics, six metrics were manually rejected during manual inspection of the dataset. Because of less number of occurrence in the sample dataset: these are Dep_Out, Dep_In, MsgSent, MsgRecv, EC_Attr and IC_Attr.

The 25 metrics were entered for analysis and their descriptive statistical values of standard deviation and mean for each metrics are obtained as depicted in Table 5.1.

Table 5.1: Descriptive statistics result

Descriptive Statistics				
S/N	**Metrics**	**Mean**	**Std. Deviation**	**Analysis N**
1	NumAttr	1.79	3.552	125
2	NumOps	5.08	5.382	125
3	NumPubOps	2.64	3.410	125
4	Setters	.32	.907	125
5	Getters	1.97	3.172	125
6	Nesting	0.17	0.380	125
7	IFImpl	.05	.213	125
8	NOC	.31	1.373	125
9	NumDesc	.49	2.387	125
10	NumAnc	.49	.775	125
11	DIT	.49	.775	125
12	CLD	.11	.383	125
13	OpsInh	4.52	7.486	125
14	AttrInh	.49	.983	125
15	NumAssEl_ssc	.70	1.382	125
16	NumAssEl_sb	1.02	1.657	125
17	NumAssEl_nsb	.71	1.574	125
18	EC_Par	2.76	10.099	125
19	IC_Par	2.77	4.755	125
20	NumCls	.72	4.837	125
21	NumCls_tc	2.86	14.589	125
22	NumOpsCls	4.54	24.593	125
23	NumInterf	.01	.089	125
24	Ca	.69	4.846	125
25	Ce	.30	1.635	125

5.3.3 Dimension Reduction

In order to reduce the number of dimensions PCA is used. Table 5.2 shows the result of PCA analysis. Initial eigenvalues column shows the variance explained by each metrics.

Total column gives the eigenvalue of the correlation matrix or amount of variance in the original variables accounted for by each component. The sum of the eigenvalues is equal to the number of independent variables. Eigenvalues near zero indicate a multicollinearity problem in the data.

% of Variance column gives the ratio the variance accounted for by each component to the total variance in all of the variables. It is the percent of the total eigenvalue. In an ideal situation, these percentages would be equal. Percent near zero indicate a multicollinearity problem in the data.

Cumulative % column gives the percentage of variance accounted for by the first n components. For example, the cumulative percentage for the second component is the sum of the percentage of variance for the first and second components. This is the running total of the Incremental Percent. For the initial solution, there are as many components as variables, and in a correlations analysis, the sum of the eigenvalues equals the number of components. Cumulative percentage is 74.384% in which variance is explained by each of the components.

Rotation is eigenvectors (factors) are rotated in an attempt to achieve simple structure in order to obtain simple and interpretable factors.

Eigenvalue: The sum of the eigenvalues is equal to the number of independent variables. Eigenvalues near zero indicate multicollinearity in the data. The eigenvalues represent the spread (variance) in the direction defined by the new axis. As eigenvalue is a reduction point for PCA variables whose eigenvalue greater than 1 are included in PC. Small eigenvalues indicate directions in which there is no spread. Hence, variables whose eigenvalue less than 1 are excluded from the PC. Since regression analysis seeks to find trends across values when there is no spread, the trends were not computed accurately. Annex D shows the correlation matrix between the 25 independent variables which measures and expresses the quantitative relationship between variables. The correlation coefficient is always between -1 and 1.

Table 5.2: Total variance explained

Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total
1	*5.406*	*21.624*	*21.624*	*5.406*	*21.624*	*21.624*	*3.696*
2	*3.814*	*15.255*	*36.879*	*3.814*	*15.255*	*36.879*	*3.637*
3	*2.651*	*10.605*	*47.484*	*2.651*	*10.605*	*47.484*	*3.394*
4	*2.483*	*9.932*	*57.416*	*2.483*	*9.932*	*57.416*	*2.686*
5	*1.658*	*6.630*	*64.046*	*1.658*	*6.630*	*64.046*	*2.850*
6	*1.465*	*5.860*	*69.906*	*1.465*	*5.860*	*69.906*	*3.482*
7	*1.119*	*4.477*	*74.384*	*1.119*	*4.477*	*74.384*	*1.127*
8	.949	3.795	78.179				
9	.828	3.312	81.490				
10	.785	3.142	84.632				
11	.678	2.713	87.346				
12	.610	2.442	89.788				
13	.499	1.997	91.785				
14	.448	1.792	93.577				
15	.395	1.581	95.158				
16	.302	1.209	96.367				
17	.216	.866	97.232				
18	.191	.764	97.997				
19	.164	.657	98.653				
20	.127	.507	99.160				
21	.085	.338	99.498				
22	.061	.245	99.743				
23	.043	.170	99.913				
24	.022	.087	100.000				
25	-1.249E-16	-4.996E-16	100.000				

Component factors Extraction

The eigenvalues of Table 5.1 is plotted using the scree plot graph as indicated in Figure 5.1. as shown seven factors are extracted whose eigenvalues are greater than 1.

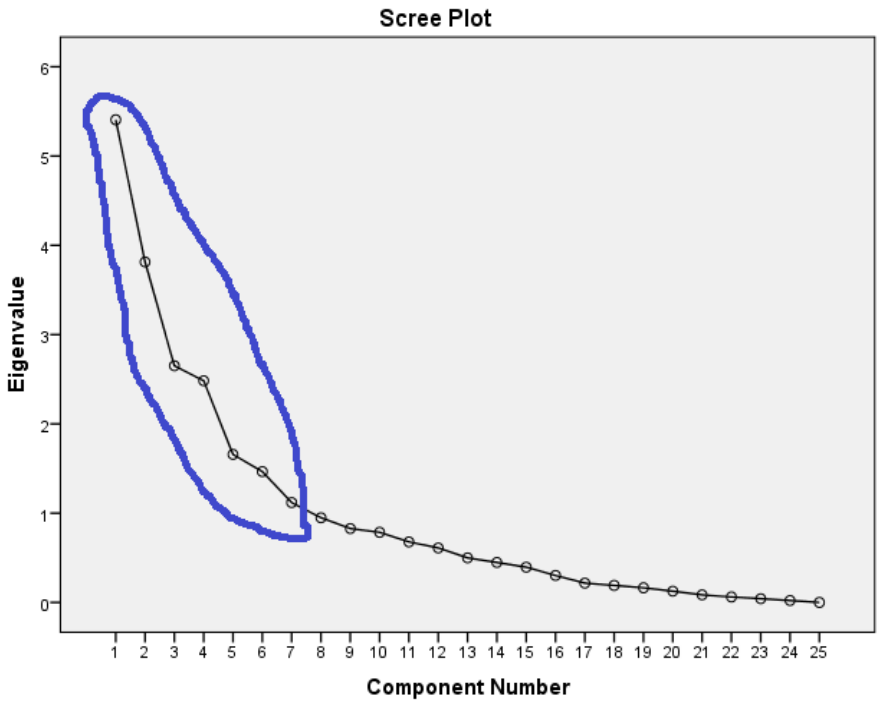


Figure 5.1: Scree plot diagram

Component / Factor Loadings

The factor-loading Table 5.3 shows the component matrix with various different loadings for the different variables within each component. Once the number of components are extracted that is less than the total number of variables, then, we rotate this again in the remaining dimensions and gain an even simpler interpretation. Under oblique rotation, Oblimin with Kaiser Normalization [48] is used and only the higher loading scores which are greater than 0.7 have been observed in the Table 5.3. Table 5.3 shows PCA factor loadings filtered with loading score greater than 0.7.

Table 5.3: PCA factor loading score

	Component						
	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Eigenvalue	***5.406***	***3.814***	***2.651***	***2.483***	***1.658***	***1.465***	***1.119***
% Variance	***21.624***	***15.255***	***10.605***	***9.932***	***6.630***	***5.860***	***4.477***
% Cumulative Var.	***21.624***	***36.879***	***47.484***	***57.416***	***64.046***	***69.906***	***74.384***
NumAttr							
NumOps							
NumPubOps					.741		

	Component						
	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Setters							
Getters							
Nesting							
IFImpl					.882		
NOC				-.873			
NumDesc				-.912			
NumAnc		.939					
DIT		.939					
CLD				-.899			
OpsInh		.935					
AttrInh		.784					
NumAssEl_ssc	.954						
NumAssEl_sb	.884						
NumAssEl_nsb						.727	
EC_Par						.905	
IC_Par	.867						
NumCls			.878				
NumCls_tc							
NumOpsCls			.947				
NumInterf							.780
Ca							
Ce			.915				

From Table 5.3 seven PCs are extracted:

PC1 has three metrics: NumAssEl_ssc, NumAssEl_sb and IC_Par. These metrics are associated together to measure the degree of coupling in a class diagram. The degree of coupling is highly influential over the quality attributes of modularity, modifiability, traceability and analyzability. The higher degree of coupling the higher effort on modularity, modifiability, traceability and analyzability quality attribute assessment.

PC2 has four metrics: DIT, NumAnc, OpsInh and AttrInh. These metrics are correlated together to measure the unit of a class diagram based on the inheritance property. The inheritance property is the vital characteristics of an object-oriented software design. As the degree of the inheritance increases, the modularity, modifiability, traceability, reusability and portability of an object-oriented software quality attributes assessment would be simpler.

PC3 has three metrics: NumOpsCls, Ce and NumCls. NumOpsCls, and NumCls metrics would measure the size of a software design at package level. As well, the Ce metric measures the coupling of classes at package level. These component factors quantify both size and coupling properties of a class diagram at package domain to measure the quality attributes of reusability and portability.

PC4 has three metrics: NumDesc, CLD and NOC: these metrics were observed as a measure of a software design based on the inheritance property at class level. These object-oriented software design characteristics are extremely influential over the quality attributes of reusability, analyzability, traceability and modifiability. As the degree of inheritance increases, reusability, analyzability, traceability decrease and modifiability of the design will be simpler.

PC5 has two metrics: IFImpl and NumPubOps: IFImpl measures the unit of inheritance in a class diagram. NumPubOps measures the size of a class diagram. In this component, IFImpl would measure portability, modifiability, traceability and analyzability quality attributes. As the IFImpl increases, portability, modifiability, traceability and analyzability would be difficult, NumPubOps also measures modularity, analyzability, testability and traceability quality attributes. As, NumPubOps increases modularity, analyzability, testability and traceability quality attributes assessment effort decrease.

PC6 – EC_Par and NumAssEl_nsb line up together to measure the unit of coupling in a class level. In this component, the two parameters would measure the quality attributes of modularity, reusability, testability, traceability and modifiability proceeding the design characteristic of inheritance property. As EC_Par and NumAssEl_nsb increase reusability and modifiability difficult. While testability and traceability would be complex.

PC7 – NumInterf would measure the size of a class diagram. As NumInterf increases analyzability, traceability and portability quality attributes assessment effort would be increased.

5.3.4 Hypothesis Testing

The p-value approach to hypothesis testing uses the calculated probability to determine whether there is evidence to reject the null hypothesis. The null hypothesis, also known as an assumption, which is the initial claim about a dataset of statistics. The p-value is used as an alternative to rejection points to provide the smallest level of significance at which the null hypothesis would be

rejected. A smaller p-value means that there is stronger evidence in favor of the alternative hypothesis.

A. Modularity hypothesis testing

An object-oriented software is composed of different modules based on their affiliation and each module can interact among one another. The proper modularization and their interaction is affected by the size and complexity of the module contents and their multiple interaction.

Table 5.4 shows independent variable estimate and significance over the dependent variable of modularity. The significance values of the independent variables NumAssEl_sb, NumInterf and Ce over the dependent variable modularity are greater than 0.05. In this case, the null hypothesis is rejected and not included in the multivariate regression model. The p-value of these variables is greater than the significance level and are not statistically significant to the model to be developed.

Table 5.4 Modularity model independent variable estimate and significance table

Parameter	Coefficient	Std. Error	p-value	95% Confidence Interval	
				Lower Bound	Upper Bound
Constant	9.842	.278	**.000**	9.292	10.393
NumPubOps	2.865	.328	**.000**	2.215	3.516
NumAssEl_ssc	4.526	.747	**.000**	3.047	6.006
NumAssEl_sb	.910	.751	.228	-.578	2.397
NumAssEl_nsb	1.370	.441	**.002**	.498	2.243
EC_Par	8.209	.406	**.000**	7.405	9.013
NumCls	4.272	.589	**.000**	3.106	5.437
NumInterf	.134	.512	.795	-.880	1.147
Ce	1.250	.764	.104	-.262	2.762

B. Reusability hypothesis testing

Reusability of an object-oriented software is highly influenced by the number of inheritance of the class diagram and with the existing of high coupling properties. Table 5.5 indicates the significance and variable estimates of the independent variables over reusability dependent variable. All the hypothetical metrics are significantly relevant to the model and these are the true assumption of the research at the beginning. From these hypothetical variables, the coefficients are positive and

the p-values are less than the significance level of 0.05. All these metrics will be included in the next analysis of prediction model development.

Table 5.5: Independent variable estimate and significance for reusability model

Parameter	Coefficient	Std. Error	p-value	95% Confidence Interval	
				Lower Bound	Upper Bound
Constant	13.597	3.611E-016	.000	13.597	13.597
NOC	1.198	5.748E-016	.000	1.198	1.198
NumDesc	2.177	6.967E-016	.000	2.177	2.177
DIT	1.456	8.688E-016	.000	1.456	1.456
CLD	.344	5.714E-016	.000	.344	.344
OpsInh	7.000	9.373E-016	.000	7.000	7.000
AttrInh	.771	5.076E-016	.000	.771	.771
NumAssEl_ssc	1.318	1.079E-015	.000	1.318	1.318
NumAssEl_sb	1.465	1.011E-015	.000	1.465	1.465
NumAssEl_nsb	1.144	5.683E-016	.000	1.144	1.144
EC_Par	9.140	5.524E-016	.000	9.140	9.140
IC_Par	4.122	5.846E-016	.000	4.122	4.122
Ce	1.496	3.717E-016	.000	1.496	1.496

C. Analyzability hypothesis testing

An object-oriented software analysis during maintenance is highly influenced by its number of inheritance of the design, coupling between classes, the size of the software and its complexity of the software design. Table 5.6 shows the metrics that are assumed to measure the level of analyzability of the software during the design phase.

Table 5.6: Independent variable estimate and significance for analyzability model

Parameter	Coefficient	Std. Error	p-value	95% Confidence Interval	
				Lower Bound	Upper Bound
Constant	8.333	.020	**.000**	8.293	8.373
NumPubOps	2.560	.024	**.000**	2.513	2.606
IFImpl	.163	.023	**.000**	.118	.209
NOC	1.280	.029	**.000**	1.222	1.338
NumDesc	2.370	.029	**.000**	2.312	2.428
NumAnc	-.058	.047	**.**220	-.151	.035
DIT	0[a]	.	.	.	.
OpsInh	7.087	.051	**.000**	6.986	7.188
AttrInh	.736	.028	**.000**	.680	.792
NumCls	4.247	.043	**.000**	4.163	4.332

NumInterf	.071	.037	**.**059	-.003	.144
Ce	1.490	.055	**.000**	1.380	1.600

In Table 5.6, coefficient value of NumAnc significance level is greater than the p-value of 0.05. DIT is not significant to the model. These two metrics were ignored for developing the prediction model. NumInterf is also not required for the model.

D. Testability hypothesis testing

With the extraordinary size and complexity, high coupling in the design of object-oriented software development decreases the level of testability for the software. Table 5.7 shows testing the level of significance of independent variables over the dependent variable of testability.

Table 5.7: Independent variable estimate and significance for testability model

Parameter	**Coefficient**	**Std. Error**	**p-value**	**95% Confidence Interval**	
				Lower Bound	**Upper Bound**
Constant	9.569	.152	**.000**	9.268	9.869
NumOps	1.744	.195	**.000**	1.358	2.130
NumAssEl_ssc	1.824	.438	**.000**	.956	2.693
NumAssEl_sb	.936	.413	**.**025	.117	1.754
NumAssEl_nsb	1.623	.237	**.000**	1.154	2.092
EC_Par	9.237	.226	**.000**	8.788	9.685
IC_Par	3.762	.218	**.000**	3.330	4.193
NumCls	4.198	.155	**.000**	3.891	4.505
NumInterf	.044	.154	**.**775	-.261	.349

In the table 5.7, the significance values for variables NumAssEl_sb is greater than the p-value while the significance value of NumInterf is greater than its p-value. These variables are not significant to the model construction. Hence, NumInterf variables is false assumption of the initial hypothesis of this research; it is rejected in the course of prediction model development.

E. Portability hypothesis testing

An object-oriented software design has the form of a collection of interacting modules that communicate through interfaces. It has to be clear to trace the size and connection of interface design is a crucial element of good software design. Table 5.8 shows the test result for the significance level of the independent variables over dependent variable of portability quality attribute.

Table 5.8: Independent variable estimate and significance for portability model

Parameter	Coefficient	p-value	95% Confidence Interval	
			Lower Bound	Upper Bound
Constant	6.194	.000	6.194	6.194
IFImpl	.188	.000	.188	.188
NumDesc	-2.177	.000	-2.177	-2.177
CLD	-.344	.000	-.344	-.344
OpsInh	7.000	.000	7.000	7.000
AttrInh	.771	.000	.771	.771
NumAssEl_ssc	1.318	.000	1.318	1.318
NumAssEl_sb	1.465	.000	1.465	1.465
NumAssEl_nsb	1.144	.000	1.144	1.144
NumInterf	.069	.000	.069	.069

In the Table 5.8, the coefficients of the variables NumDesc and CLD are negative values. The significance levels are less than the p-value of 0.05. Consequently, their relationship to portability is in the opposite direction. In a class diagram, classes deeper down in the inheritance hierarchy would likely to be more difficulty for portability case.

F. Traceability hypothesis testing

Even if this quality attribute is widely applied in requirement phase, it is clear that software design should be clear and understandable to trace for different purposes such as modification, extension, maintenance etc. This quality attribute was measured with the size and complexity, coupling, inheritance properties. Table 5.9 shows the significance level and coefficient estimates test result for the independent variables over the traceability dependent variable.

Table 5.9: Independent variable estimate and significance for traceability model

Parameter	**Coefficient**	**Std. Error**	**p-value**	**95% Confidence Interval**	
				Lower Bound	**Upper Bound**
Constant	12.653	.498	**.000**	11.667	13.639
NumPubOps	-.697	.706	.325	-2.096	.701
IFImpl	.211	.618	.733	-1.014	1.436
NOC	-1.230	.794	.124	-2.804	.344
NumDesc	2.108	.979	.033	.168	4.048
NumAnc	.910	1.201	.450	-1.470	3.290
DIT	0[a]	.	.	.	.
CLD	.115	.798	.885	-1.466	1.697
OpsInh	5.530	1.293	**.000**	2.967	8.092

Parameter	Coefficient	Std. Error	p-value	95% Confidence Interval	
				Lower Bound	Upper Bound
AttrInh	-.046	.704	.948	-1.442	1.349
NumAssEl_ssc	2.325	1.498	.123	-.642	5.292
NumAssEl_sb	.031	1.387	.982	-2.717	2.779
NumAssEl_nsb	.842	.810	.301	-.762	2.446
EC_Par	9.487	.792	**.000**	7.917	11.056
IC_Par	4.014	.821	**.000**	2.388	5.639

For the case of traceability hypothesis in Table 5.9, OpsInh, EC_Par and IC_Par are the only significant variables whose p-values are less than 0.05. Hence, these three variables will be included in the construction of traceability model.

G. Modifiability hypothesis testing

Object-oriented properties, inheritance, coupling, size and complexity of a software design are some determinant factors to modify the software product after development. These properties are measured with in these object-oriented design metrics. Table 5.10 shows hypothesis test result for the level of significance and parameter estimates of the independent variables over modifiability dependent variable.

Table 5.10: Independent variable estimate and significance for modifiability model

Parameter	Coefficient	Std. Error	p-value	95% Confidence Interval	
				Lower Bound	Upper Bound
Intercept	12.000	1.036E-016	**.000**	12.000	12.000
IFImpl	.188	1.100E-016	**.000**	.188	.188
NOC	-1.198	1.651E-016	**.000**	-1.198	-1.198
NumDesc	-2.177	2.002E-016	**.000**	-2.177	-2.177
NumAnc	1.456	2.503E-016	**.000**	1.456	1.456
DIT	0[a]	.	.	.	.
CLD	-.344	1.641E-016	**.000**	-.344	-.344
OpsInh	7.000	2.690E-016	**.000**	7.000	7.000
AttrInh	.771	1.457E-016	**.000**	.771	.771
NumAssEl_ssc	1.318	3.117E-016	**.000**	1.318	1.318
NumAssEl_sb	1.465	2.902E-016	**.000**	1.465	1.465
NumAssEl_nsb	1.144	1.669E-016	**.000**	1.144	1.144
EC_Par	9.140	1.607E-016	**.000**	9.140	9.140

Parameter	Coefficient	Std. Error	p-value	95% Confidence Interval	
				Lower Bound	Upper Bound
IC_Par	4.122	1.690E-016	**.000**	4.122	4.122
Ce	1.496	1.069E-016	**.000**	1.496	1.496

In the Table 5.10, variables coefficient values of the variables NOC, NumDesc and CLD are negative. Hence, the relationship with modifiability is in reverse direction. As inheritance level increases to the deeper, the modifiability level of the system will decrease. DIT variable is totally irrelevant to the model and rejected from the model. The other remaining variables are significant as the p-values are less are than 0.05. Hence, these variables have direct relationship with modifiability quality attribute.

5.4 Prediction Model Construction

In this section, the multivariate regression technique is applied as metrics variable selection tool. Multivariate regression equation along with investigation in to specific predictor variables. The aim of selection is to reduce the set of predictor variables to those that are necessary and account for nearly as much of the variance as is accounted for by the total set. This determines the level of importance for each variable. P-value is the determinant point to determine the significance of the independent variables with 95% confidence.

5.4.1 Modularity Prediction Model

For modularity prediction model, the significant metrics used in this model are NumAssEl_ssc, NumAssEl_nsb, NumPubOps, NumCls and EC_Par as indicated in Table 5.11.

To derive the modularity prediction model, we first draw statistical significance of the independent variables. as follows. Table 5.11 shows the statistical significance of the independent variables. In Table 5.11, NumAssEl_nsb is the inverse relationship with the modularity quality attribute.

Table 5.11: Parameter significance and estimate for modularity model

Parameters	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
(Constant)	**9.842**	**.441**		**.000**
NumAssEl_ssc	**6.317**	**.497**	**.454**	**.000**
NumAssEl_nsb	-.083	.698	-.006	.906

Parameters	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
NumPubOps	**2.227**	**.519**	**.160**	**.000**
NumCls	**1.551**	**.446**	**.111**	**.001**
EC_Par	**8.554**	**.633**	**.614**	**.000**

Thus, the model is given as using equation 1.

Modularity *= 9.842 + 6.317*NumAssEl_ssc+ -0.083* NumAssEl_nsb + 2.227*NumPubOps + 8.554*EC_Par*

We next show the statistical significance of the model and its fitness

a) statistical significance of the model

The *F*-ratio in the ANOVA Table 5.12 tests whether the overall regression model is a good fit for the data. Table 5.12 shows that the independent variables are statistically significant to predict the dependent variable, F (5, 121) = 173.591, $p < .0005$ (i.e., the regression model is a good fit of the data).

Table 5.12: ANOVA for modularity prediction model

Model	Sum of Squares	df	Mean Square	F	Sig.
Regression	21437.083	5	4287.417	173.591	.000
Residual	2988.501	121	24.698		
Total	24425.584	126			

b) Fitness of the model

Table 5.13 shows the summary of the modularity model. R is one measure of the quality of the prediction of the dependent variable for modularity of a class diagram. A value of 0.937 indicates that the model is in a good level of prediction.

R^2 is coefficient of determination. This is the proportion of variance in the dependent variable that can be explained by the independent variables. The value of R^2 is 0.878 where the independent variables explain 87.3% of the variability of modularity dependent variable.

Table 5.13: Modularity Model Summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate
1	.937	.878	.873	4.969744

5.4.2 Reusability Prediction Model

In reusability prediction model, the significant metrics NOC, NumDesc, DIT, OpsInh, AttrInh, NumAssEl_ssc, NumAssEl_sb, NumAssEl_nsb, EC_Par, IC_Par and Ce are entered to the multivariate regression model as indicated in Table 5.14

To derive the reusability prediction model, we first draw statistical significance of the independent variable. Table 5.14, shows the statistical significance of the independent variables. Only one non-significant metric CLD is removed from the model. All the other independent variables are significant to the model.

Table 5.14: Parameter significance and estimate for reusability model

Parameters	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
(Constant)	**22.735**	**.000**		**.000**
NOC	**1.199**	**.000**	**.109**	**.000**
NumDesc	**2.177**	**.000**	**.199**	**.000**
DIT	**1.457**	**.000**	**.133**	**.000**
CLD	.000	.000	.000	.130
OpsInh	**7.000**	**.000**	**.639**	**.000**
AttrInh	**.772**	**.000**	**.070**	**.000**
NumAssEl_ssc	**1.318**	**.000**	**.120**	**.000**
NumAssEl_sb	**1.465**	**.000**	**.134**	**.000**
NumAssEl_nsb	**1.144**	**.000**	**.104**	**.000**
EC_Par	**1.000**	**.000**	**.091**	**.000**
IC_Par	**4.122**	**.000**	**.376**	**.000**
Ce	**1.496**	**.000**	**.137**	**.000**

Thus the formula given by:

Reusability *= 22.735+ 1.199*NOC + 2.177*NumDesc + 1.457*DIT + 7.0*OpsInh + 0.772*AttrInh + 1.318*NumAssEl_ssc + 1.465*NumAssEl_sb + 1.144*NumAssEl_nsb + 1.0*EC_Par + 4.122*IC_Par + 1.496*Ce*

We next is to show the statistical significance of the model and its fitness.

a) Statistical significance of the model

Table 5.15 shows the ANOVA table for reusability model. All the independent variables are statistically significant to predict the dependent variable reusability quality attribute with p-value of $p<0.0005$. Hence, the regression model is a good fit of the data.

Table 5.15: ANOVA for reusability prediction model

Model	Sum of Squares	df	Mean Square	F	Significance
Regression	15111.314	12	1259.276	868279172.219	.000
Residual	.000	114	.000		
Total	15111.314	126			

b) Fitness of the model

Table 5.16 shows the summary of the reusability model. R in the second column to be one measure of the quality of the prediction of the dependent variable for reusability of a class diagram. A value of 1 indicates that the model is in a perfect level of prediction. R^2 is coefficient of determination. This is the proportion of variance in the dependent variable that is explained by the independent variables. The value of R^2 is one where the independent variables explain 100% of the variability of reusability dependent variable. The value of R^2 equal to one and has less error that shows that the model developed is extended with great fitness level of one.

Table 5.16: Reusability model summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate
1	1.000	1.000	1.000	.001204

5.4.3 Analyzability Prediction Model

In an analyzability prediction model, the metrics NumPubOps, NumDesc and OpsInh were identified as relevant to the analyzability model in multivariate linear regression model test as indicated in Table 5.17. To derive the analyzability prediction model, we first draw statistical significance of the independent variables over analyzability dependent variable as follows.

Table 5.17 shows the significance level and parameter estimate of each independent variable over analyzability dependent variable. Only NumPubOps, NumDesc and OpsInh independent variables are statistically significant to analyzability prediction model. The independent variables whose p-values are < 0.05 would be the measure of analyzability model.

Table 5.17: Parameter significance and estimate table for analyzability model

Parameters	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
(Constant)	**8.331**	**.364**		**.000**
NumPubOps	**1.908**	**.426**	**.218**	**.000**
IFImpl	.237	.415	.027	.569
NOC	1.264	.530	.145	.019
NumDesc	**2.205**	**.529**	**.252**	**.000**
OpsInh	**6.536**	**.517**	**.747**	**.000**
AttrInh	.729	.507	.083	.154
NumCls	1.635	.770	.187	.036
NumInterf	.210	.670	.024	.755
Ce	.920	1.001	.105	.360

Thus the model given as equation 1.

Analyzability *= 8.333+ 1.908* NumPubOps + 2.205* NumDesc + 6.536* OpsInh*

We next show the statistical significance of the model and its fitness.

a) Statistical significance of the model

From the Table 5.18, the significant level of the independent variables over the independent variable of analyzability was described. In this table the significance level p-value is < 0.0005 at confidence interval level of 95%.

Table 5.18: ANOVA for analyzability prediction model

Model	Sum of Squares	df	Mean Square	F	Significance
Regression	7664.088	9	851.565	50.583	.000
Residual	1969.690	117	16.835		
Total	9633.778	126			

b) Fitness of the model

Table 5.19 shows the summary of analyzability model. The value of R is 0.892 which refers the prediction model is in good level of fit.

The value of adjusted R^2 is 0.780 which refers 78% of the independent variable are explained in analyzability dependent variable.

Table 5.19: Analyzability model summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate
1	.892	.796	.780	4.103042

5.4.4 Testability Prediction Model

In a testability prediction model, the metrics used to construct the model were NumPubOps, NumAssEl_ssc, NumAssEl_nsb, EC_Par, IC_Par and NumCls as indicated in Table 5.20In order to construct testability prediction model we first draw statistical significance of the independent variables.

Table 5.20 indicates parameter significance and estimate for testability model. All independent variables are statistically significant to predict testability quality attribute. The significant values of each independent variable value is < 0.0005.

Table 5.20: Parameter significance and estimate for testability model

Parameters	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
(Constant)	9.569	.000		.000
NumPubOps	2.514	.000	.176	.000
NumAssEl_ssc	2.673	.000	.187	.000
NumAssEl_nsb	1.128	.000	.079	.000
EC_Par	9.275	.000	.650	.000
IC_Par	4.071	.000	.285	.000
NumCls	4.222	.000	.296	.000

Thus the model given by:

Testability *= 9.569+ 2.514*NumPubOps + 2.673*NumAssEl_ssc + 1.128*NumAssEl_nsb + 9.274*EC_Par + 4.071*IC_Par + 4.222*NumCls.*

Then we next show the statistical significance of the model and its fitness.

a) **Statistical significance of the model**

Table 5.21 shows ANOVA result for testability prediction model. From this table the significant level of the independent variables over the independent variable of testability was described as total. In this table, the significance level p-value is < 0.0005 at confidence interval level of 95%.

Table 5.21: ANOVA for testability prediction model

Model	Sum of Squares	df	Mean Square	F	Significance
Regression	25662.653	6	4277.109	1971008339.684	.000
Residual	.000	120	.000		
Total	25662.653	126			

a) Fitness of the model

Table 5.22 shows the summary of testability model. The value of R is 1 which is the prediction model is perfect and the value R^2 is also 1 hence the independent variables explained 100% of the dependent variable testability quality attribute.

Table 5.22: Testability model summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate
1	1.000	1.000	1.000	.001473

5.4.5 Portability Prediction Model

For portability prediction model, OpsInh, AttrInh, NumAssEl_sb and NumAssEl_nsb were used to construct portability prediction model is indicated in Table 5.23. In order to construct portability prediction model we first draw statistical significance of the independent variables as follows.

Table 5.23 shows parameter significance and coefficient estimate for portability model of independent variables. Independent variables IFImpl and NumInterf both are less significant. The regression coefficient is also very much less. In addition, the regression coefficient of NumAssEl_ssc is zero which indicates that it does not have an impact over the dependent variable. The remaining independent variables are relevant to portability model and all independent variables are directly related to the dependent variable of portability quality attribute.

Table 5.23: Parameter significance and estimate table for portability model

Parameters	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
(Constant)	6.192	.000		.000
IFImpl	**-1.609E-005**	**.000**	**.000**	**.853**
OpsInh	6.904	.000	.943	.000

Parameters	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
AttrInh	.962	.000	.131	.000
NumAssEl_ssc	**.000**	**.000**	**.000**	**.426**
NumAssEl_sb	1.968	.000	.269	.000
NumAssEl_nsb	1.359	.000	.186	.000
NumInterf	**1.577E-005**	**.000**	**.000**	**.853**

Thus the given model.

*Portability: 6.194+6.904*OpsInh+0.961*AttrInh+ 1.968*NumAssEl_sb+1.359* NumAssEl_nsb*

Then we next show the statistical significance of the model and its fitness.

a) Statistical significance of the model

Table 5.24 shows ANOVA result for portability prediction model. In ANOVA table, the significance level of the independent variables over portability dependent variable is labelled as determinants. The significance value of the model is $p < 0.000$. This shows that the metrics are real determinants for portability quality attribute.

Table 5.24: ANOVA for portability prediction model

Model	Sum of Squares	df	Mean Square	F	Significance
Regression	6752.878	7	964.697	1069123699.908	.000
Residual	.000	119	.000		
Total	6752.878	126			

b) Fitness of the model fits

Table 5.25 shows the summary of portability prediction model. The portability model summary was described as the model is in a perfect fit. The R-value for this model is one which tells us the model is a perfect fit and R^2 is equal to one which tells us the independent variables explain 100% for the dependent variable of portability. The standard error is also very small in this model.

Table 5.25: Portability model summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate
1	1.000[a]	1.000	1.000	.000950

5.4.6 Traceability Prediction Model

In traceability prediction model metrics used to construct are EC_Par and IC_Par as indicated in Table 5.26. In order to construct traceability prediction model we first draw th statistical significance of each individual independent variables as follows.

Table 5.26 shows parameter significance and coefficient estimates for traceability model. The significance of the independent variables EC_Par and IC_Par in the Table 5.26 shows that they are just a determinant for traceability quality attribute. The p-value of both metrics are < 0.000. In addition, these metrics have a positive coefficient and they have direct relation with the dependent variable of traceability quality attribute.

Table 5.26: Parameter significance and estimate table for traceability model

Parameters	**Unstandardized Coefficients**		**Standardized Coefficients**	**Significance**
	B	**Std. Error**	**Beta**	
(Constant)	12.652	.000		.000
EC_Par	9.789	.000	.850	.000
IC_Par	5.864	.000	.509	.000

Thus the model given by:

Traceability: 12.653+ 9.788*EC_Par + 5.864*IC_Par

Then we next show the statistical significance of the model and its fitness.

a) Statistical significance of the model

Table 5.27 shows the ANOVA result for traceability prediction model. In this table, the significance of the model within each independent variables to be measured in a harmonized way. This indicates the parameters are significant and determinant to the dependent variable of traceability. The value of p is < 0.000.

Table 5.27: ANOVA for traceability prediction model

Model	**Sum of Squares**	**Df**	**Mean Square**	**F**	**Significance**
Regression	16708.896	2	8354.448	3737299228.134	.000
Residual	.000	124	.000		
Total	16708.896	126			

b) Fitness of the model

Table 5.28 shows the summary of traceability model. As overall, traceability model summary is in a good state. The R-value is exactly one which indicates that the model is in a perfect fit. The value of R^2 is also 1 that specifies the all independent variables explained in 100% for the dependent variable of traceability.

Table 5.28: Traceability model summary

Model	R	R^2	Adjusted R^2	Std. Error of the Estimate
1	1.000	1.000	1.000	.001495

5.4.7 Modifiability Prediction Model

For modifiability prediction model, metrics used to construct were NOC, NumDesc, NumAnc, OpsInh, AttrInh, NumAssEl_ssc, NumAssEl_sb, NumAssEl_nsb, EC_Par, IC_Par and Ce as indicated in Table 5.29. In order to construct modifiability prediction model we first draw the statistical significance of each individual independent variables as follows.

Table 5.29 shows the significance level of and coefficient estimates of parameters. In this table the two metrics IFImpl and CLD have been found irrelevant to the model. The parameter value of IFImpl is zero, which does not have any relation with the dependent variable. The Coefficients of CLD is also has an inverse relation with the dependent variable. However, the value is very small hence p-value is greater than 0.005.

Table 5.29: Parameter significance and estimate table for modifiability model

Parametrs	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
(Constant)	**11.998**	**.000**		**.000**
IFImpl	.000	.000	.000	.280
NOC	**-1.197**	**.000**	**-.083**	**.000**
NumDesc	**-2.177**	**.000**	**-.151**	**.000**
NumAnc	**1.457**	**.000**	**.101**	**.000**
CLD	-3.619E-005	.000	.000	.853
OpsInh	**7.000**	**.000**	**.487**	**.000**
AttrInh	**.771**	**.000**	**.054**	**.000**
NumAssEl_ssc	**1.318**	**.000**	**.092**	**.000**
NumAssEl_sb	**1.465**	**.000**	**.102**	**.000**
NumAssEl_nsb	**1.144**	**.000**	**.080**	**.000**

Parametrs	Unstandardized Coefficients		Standardized Coefficients	Significance
	B	Std. Error	Beta	
EC_Par	**9.141**	**.000**	**.636**	**.000**
IC_Par	**4.122**	**.000**	**.287**	**.000**
Ce	**1.496**	**.000**	**.104**	**.000**

Thus the model given by:

***Modifiability**: 12.000 + -1.198*NOC + -2.177*NumDesc + 1.456*NumAnc + 7*OpsInh + 0.771* AttrInh + 1.318*NumAssEl_ssc + 1.465*NumAssEl_sb + 1.144*NumAssEl_nsb + 9.140*EC_Par + 4.122*IC_Par + 1.496*Ce*

Then we next show the statistical significance of the model and its fitness.

a) Statistical significance of the model

Table 5.30 shows ANOVA result for modifiability prediction model, which indicates the significance of independent variable as overall in the model. The significance value of the model is $p < 0.000$ which implies that all the metrics are relevant to the model.

Table 5.30: ANOVA for modifiability prediction model

Model	Sum of Squares	df	Mean Square	F	Significance
Regression	26047.591	13	2003.661	1039873593.505	.000
Residual	.000	113	.000		
Total	26047.591	126			

b) Fitness of the model

Table 5.31 demonstrates the summary of modifiability model how the model fits well. For the model fit measures, the value of R is one however; the model is in a perfect fit. In addition the value of R^2 is one hence the independent variables in a model explain 100% for the dependent variable.

Table 5.31: Modifiability model summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	1.000	1.000	1.000	.001388

Chapter Six : Conclusion and Future Works

6.1 Conclusion

In this research, we developed a maintainability prediction model to assess the effect of a software design during maintenance. In order to assess the design of a software, we have used some object-oriented metrics as means of measurement variables. Object-oriented software design approach is one of the widely used software development approach due to its ease of maintenance cost. Most frequently known metrics are selected from SDMetrics model to construct the estimation model.

In this work, is has been tried to propose a model that will estimate the maintainability of the software design at early stage of development using multivariate linear regression approach. The maintainability of the design would be measured with its sub-characteristics. For this work, seven maintainability sub-characteristics are taken to estimate the maintainability of the software design. These sub-characteristics are modularity, reusability, analyzability, testability, modifiability, portability and traceability sub-characteristics. Its class diagram of the software design would measure the maintainability of the software design. The estimation model is constructed from the behaviors of the class diagram.

The controlled experiment and data analysis is performed on class diagram of XMI schema document. Initially 31 class diagram level metrics were selected to estimate the maintainability of the software design. The class level metrics are found from SDMetrics metric box. At the start of the experimentation, only 25 metrics are captured from the dataset and six metrics are removed. To construct the model previously used metrics and sub-characteristics as well as additional metrics and sub-characteristics are integrated together to get a better, efficient and more inclusive estimation model. Hence, the experimental analysis has been done on a class diagram of having 125 classes. The increasing number of classes makes the model to capture more behaviors of the class attributes. In order to get the model several steps have been performed. Descriptive analysis is done. Then the hypothetical metrics are tested individually against the independent variable. Then, the significant metrics are entered to the regression model to construct the maintainability estimation model. After analysis, for each seven maintainability sub-characteristics, seven prediction models are proposed. Finally, the proposed models are evaluated statistically based on their **R**, $\mathbf{R^2}$ values and the models are in a good fit.

6.2 Contributions

The contributions of this research is summarized as follows.

- Enhancing maintainability measurement models by adding further maintainability sub-characteristics, which are portability and traceability. Adding additional sub-characteristics will help to capture more factors during maintenance case.
- Portability and traceability sub-characteristics did not have measurement metrics. In this research, we presented the corresponding metrics for these maintainability sub-characteristics.
- Moreover, we have identified additional metrics for the previously used maintainability sub-characteristics. Adding more measuring metrics to the sub-characteristics would enhance the efficiency and capability of the model for measuring the software design.

6.3 Future Works

The current work can be further extended and enhanced by adding the following feature:

- Automating maintainability estimation tool: Once the dependent and independent variables are identified, it will be worthy to develop standalone maintainability predictor applications based on this research output.

References

[1] M. Kumar, R. Thakur, M. Mann, "A Tool for Quality Measurement of Software based on Object Oriented Design", *International Journal of Advance Research in Computer Science and Management Studies*, Volume 2, Issue 5, May 2014.

[2] B. Katoch, L. K. Shah., "A Systematic Analysis on MOOD and QMOOD Metrics", *International Journal of Current Engineering and Technology*, Vol.4, No.2 April 2014.

[3] IEEE Standard for Software Maintenance, Approved 25 June 1998.

[4] N. Jayalakshmi, N. Satheesh, "Software Quality Assessment in Object Based Architecture", *International Journal of Computer Science and Mobile Computing (IJCSMC)*, Vol. 3, Issue. 3, March 2014, pg.941 – 946

[5] S Motogna., A.Vescan, C.Serban, P. Tirban, "An Approach to Assess Maintainability Change", *IEEE*, 2016.

[6] V. Lenarduzzi, A. Sillitti, D. Taibi**, "**Analyzing Forty Years of Software Maintenance Models", *IEEE/ACM 39th IEEE International Conference on Software Engineering Companion*, 2017.

[7] J. Bansiya and C. Devis, "A Hierarchical Model for Object-Oriented Design Quality Assessment," *IEEE Transaction on Software Engineering*, vol. 28, no. 1, pp. 4 - 17, 2002.

[8] C. G. DESAI, "Object Oriented Design Metrics, Models And Quality Models",

Information Science and Technology, Volume 3, Issue 2, 2014

[9] C. Neelamegam, M. Punithavalli, "A Survey - Object Oriented Quality Metrics",

Global Journal of Computer Science and Technology

[10] S. C. Chidamber, C. F. Kemerer "Ametric Suite for Object Oriented Design", *IEEE Transaction On Software Engineering*, vol 20, No. 6, June 1994

[11] V. Rai, A. M. Srivastava, H. Pandey, V. K Singh, "Estimation of Maintainability in Object Oriented Design Phase: State of the art", *International Journal of Scientific & Engineering Research*, Volume 6, Issue 9, September-2015

[12] K. Keshamoni, M. Harikrishna "Improved Visual Cryptography Scheme for Data Security", *Information Science and Technology*, Volume 3, Issue 2, pp.-066-070. 2014

[13] S. K. Punia, P. Kumar, A. Gupta, “A Review of Software Quality Metrics for Object-Oriented Design”, *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 6, Issue 8, August 2016

[14] ISO/IEC JTC1/SC7 N4098, 2008-07-17, *http://sa.inceptum.eu/sites/sa.inceptum.eu/files/Content/ISO_25010.pdf*

[15] Software Quality ISO Standards, *www.arisa.se/compendium/node6.html*

[16] Juran, J. M., Juran's Quality Control Handbook, McGraw-Hill, 1988.

[17] Shewhart, W. A., Economic control of quality of manufactured product, Van Nostrand, 1931.

[18] McCall, J. A., Richards, P. K., and Walters, G. F., Concepts and definitions of software quality, Factors in Software Quality, NTIS, Vol. 1, 1977

[19] ISO/IEC FDIS 9126-1, International Standard, 2000

[20] ISO/IEC CD 25010: Software engineering – Software product Quality Requirements and Evaluation (SQuaRE), 2008

[21] Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M., Characteristics of Software Quality, North Holland, 1978.

[22] Dromey, R. G., "Concerning the Chimera [software quality]", *IEEE Software*, no. 1, pp. 33-43, 1996.

[23] Dromey, R. G., "A model for software product quality", *IEEE Transactions on Software Engineering*, no. 2,pp. 146-163, 1995.

[24] Mansi A., Verma K., Mishra V. H., “An Analytical Study of Object-Oriented Metrics”, *International Journal of Engineering Trends and Technology (IJETT)* – Volume 6 number 2- Month 2013

[25] Punia K. S., Kumar P., Gupta A., “A Review of Software Quality Metrics for Object-Oriented Design”, *International Journal of Advanced Research in Computer Science and Software Engineering*. Volume 6, Issue 8, August 2016

[26] Sharma A., Kumar S. D., “Comparison of Software Quality Metrics for Object-Oriented System”, *IJCSMS International Journal of Computer Science & Management Studies*, Special

Issue of Vol. 12, June 2012

[27] Shaik A., Reddy C. R. K., Manda B., Prakashini. C, Deepthi. K, “Metrics for Object Oriented Design Software Systems: A Survey”, *Journal of Emerging Trends in Engineering and Applied Sciences (JETEAS)*, 1 (2): 190-198 (ISSN: 2141-7016), 2010

[28] K. Pearson "On Lines and Planes of Closest Fit to Systems of Points in Space," Philosophical Magazine. 2 (11), 559–572, 1901.

[29] H. Hotelling “Analysis of a complex of statistical attributes into principal components”. Journal of Educational Psychology, 24, 417–441, and 498–520, 1933.

[30] J.V. Gulmine and L. Akcelrud, "Correlation between structure and accelerated artificial ageing of XLPE", Eur. Polym. J, Vol.42, pp. 553-562, Mar 2006.

[31] H. Abdi and L. J. Williams , “Principal component analysis”, WIREs Computational Statistics, Volume 2, July/August 2010

[32] L. C. Briand, J. Wüst, H. Lounis, “Replicated Case Studies for Investigating Quality Factors in Object-Oriented Design”,

[33] Mehmet M. “Multivariate Multiple Regression Analysis Based on Principal Component Scores to Study Relationships between Some Pre- and Post-slaughter Traits of Broilers”, *Journal of Agricultural Sciences*, 30 March 2011

[34] Sahn, D.E. and D. Stifel, “Assets as a measure of household welfare in developing countries. Center for Social Development”, Working Paper 00-11, Washington University, 2000

[35] A. S. Singh and M. B. Masuku, “Applications of Modeling and Statistical Regression Techniques in Research”, *Research Journal of Mathematical and Statistical Sciences*, Vol. 1(6), 14-20, July 2013

[36] Kiewkanya M., Jindasawat N., Muenchaisri P., “A methodology for constructing maintainability model of object-oriented design,” *Quality Software QSIC 2004 Proceedings Fourth International*, 2004, pp. 206 213

[37] S. W. A. Rizvi and R. A. Khan, "Maintainability Estimation Model for Object- Oriented Software in Design Phase (MEMOOD)," *COMPUTING*, vol. 2, no. 4, pp. 26-32. APRIL

2010

[38] C. Gautam and S. S. Kang, “Comparison and implementation of compound memood model and memood model," *International Journal of Computer Science and Information Technologies* vol. 2, no. 5, pp. 2394-2398, 2011.

[39] A. Hincheeranan, W. Rivepiboon, “A Maintainability Estimation Model and Tool”, *International Journal of Computer and Communication Engineering*, Vol. 1, No. 2, July 2012

[40] Kiranjit K., Sami A. “A Maintainability Estimation Model and Metrics for Object-Oriented Design (MOOD)”, *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, Volume 2, No 5, May 2013.

[41] S. Bhutani, A. Chug, “Prediction of Software Maintainability using Neural Networks”, *International Journal of Computer Science & Communication Networks*, Vol 5(2), May 2015

[42] S. Chidamber, D. Darcy, C. Kemerer, “Managerial use of metrics for object-oriented software: An Exploratory Analysis”, *IEEE Transactions on Software Engineering*, vol. 24, no. 8, August 1998.

[43] A. H. Watson, T.J. McCabe, “A Testing Methodology Using the Cyclomatic Complexity Metric”, *National Institute of Standards and Technology*, Special Publication 500-235, September 1996

[44] Y. Dash, S. K. Dubey, A. Rana, “Maintainability Prediction of Object Oriented Software System by Using Artificial Neural Network Approach”, *International Journal of Soft Computing and Engineering (IJSCE),* Volume-2, May 2012

[45] Diwan P., “Predicting Maintainability of Object-Oriented System using Fuzzy Logic”, *International Journal of Scientific and Research Publications,* Volume 6, Issue 3, March 2016

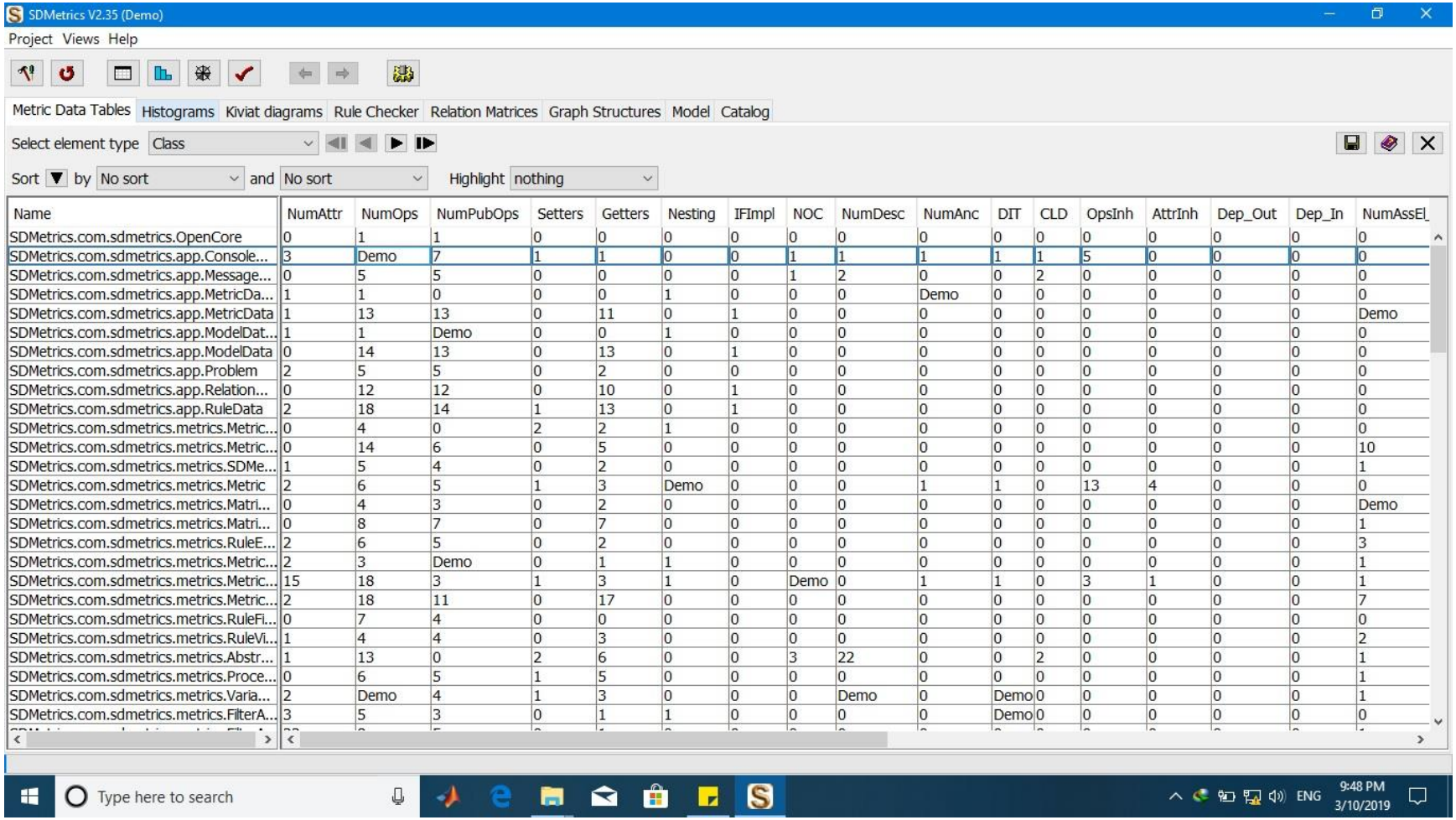
[46] S. Mishra, A. Sharm, “Maintainability Prediction of Object Oriented Software by using Adaptive Network based Fuzzy System Technique”, *International Journal of Computer Applications,* Volume 119 – No.9, June 2015

[47] J. Wüst, SDMetrics user manual, http://www.sdmetrics.com, Last Accessed June 28, 201

[48] J. D. Brown. “Choosing the Right Type of Rotation in PCA and EFA”, *JALT Testing & Evaluation SIG Newsletter.* November 2009

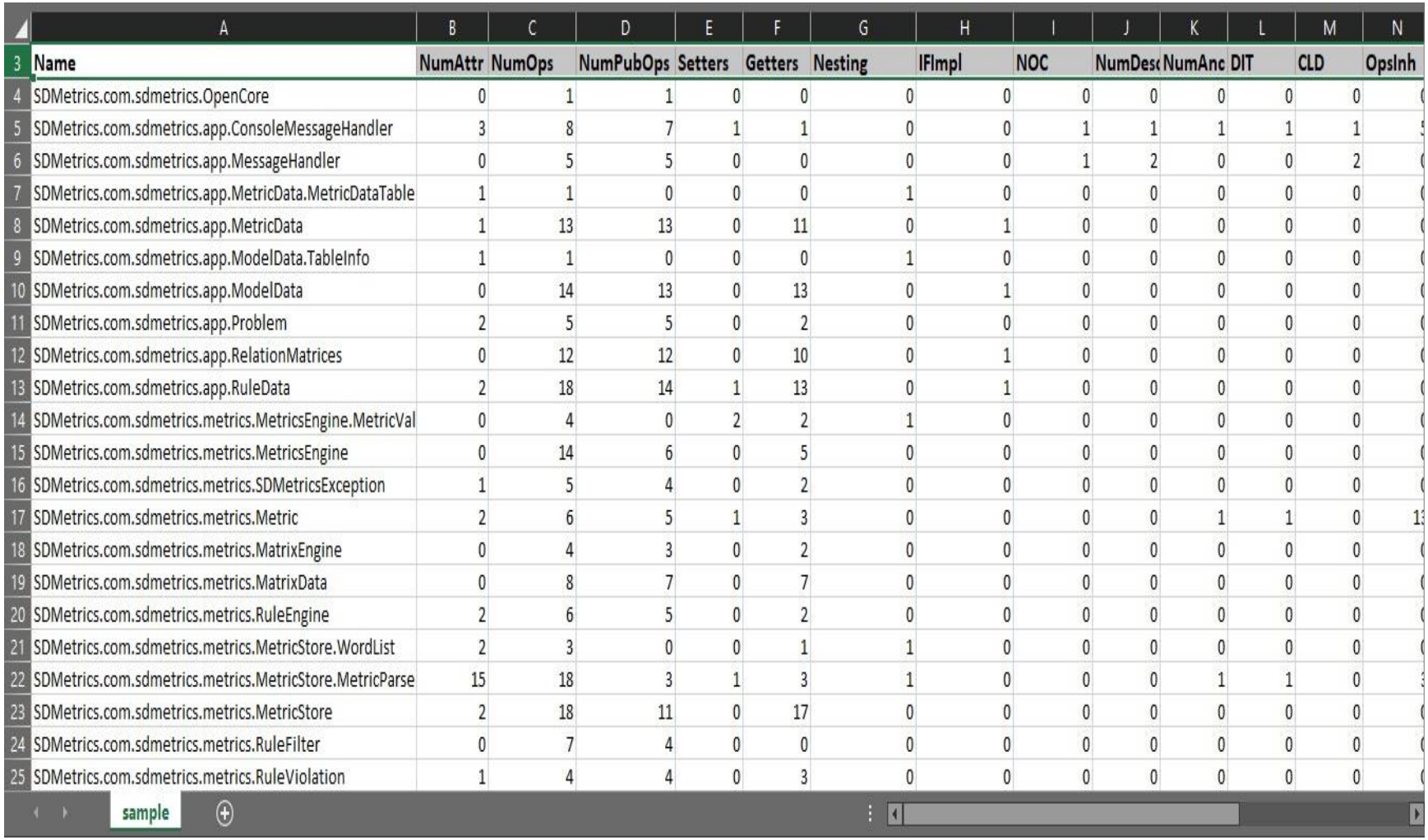
Annexes

Annex A: SDMetrics Home page



Name	NumAttr	NumOps	NumPubOps	Setters	Getters	Nesting	IFImpl	NOC	NumDesc	NumAnc	DIT	CLD	OpsInh	AttrInh	Dep_Out	Dep_In	NumAssEl
SDMetrics.com.sdmetrics.OpenCore	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.app.Console...	3	Demo	7	1	1	0	0	1	1	1	1	1	5	0	0	0	0
SDMetrics.com.sdmetrics.app.Message...	0	5	5	0	0	0	0	1	2	0	0	2	0	0	0	0	0
SDMetrics.com.sdmetrics.app.MetricDa...	1	1	0	0	0	1	0	0	0	Demo	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.app.MetricData	1	13	13	0	11	0	1	0	0	0	0	0	0	0	0	0	Demo
SDMetrics.com.sdmetrics.app.ModelDat...	1	1	Demo	0	0	1	0	0	0	0	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.app.ModelData	0	14	13	0	13	0	1	0	0	0	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.app.Problem	2	5	5	0	2	0	0	0	0	0	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.app.Relation...	0	12	12	0	10	0	1	0	0	0	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.app.RuleData	2	18	14	1	13	0	1	0	0	0	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.metrics.Metric...	0	4	0	2	2	1	0	0	0	0	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.metrics.Metric...	0	14	6	0	5	0	0	0	0	0	0	0	0	0	0	0	10
SDMetrics.com.sdmetrics.metrics.SDMe...	1	5	4	0	2	0	0	0	0	0	0	0	0	0	0	0	1
SDMetrics.com.sdmetrics.metrics.Metric	2	6	5	1	3	Demo	0	0	0	1	1	0	13	4	0	0	0
SDMetrics.com.sdmetrics.metrics.Matri...	0	4	3	0	2	0	0	0	0	0	0	0	0	0	0	0	Demo
SDMetrics.com.sdmetrics.metrics.Matri...	0	8	7	0	7	0	0	0	0	0	0	0	0	0	0	0	1
SDMetrics.com.sdmetrics.metrics.RuleE...	2	6	5	0	2	0	0	0	0	0	0	0	0	0	0	0	3
SDMetrics.com.sdmetrics.metrics.Metric...	2	3	Demo	0	1	1	0	0	0	0	0	0	0	0	0	0	1
SDMetrics.com.sdmetrics.metrics.Metric...	15	18	3	1	3	1	0	Demo	0	1	1	0	3	1	0	0	1
SDMetrics.com.sdmetrics.metrics.Metric...	2	18	11	0	17	0	0	0	0	0	0	0	0	0	0	0	7
SDMetrics.com.sdmetrics.metrics.RuleFi...	0	7	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SDMetrics.com.sdmetrics.metrics.RuleVi...	1	4	4	0	3	0	0	0	0	0	0	0	0	0	0	0	2
SDMetrics.com.sdmetrics.metrics.Abstr...	1	13	0	2	6	0	0	3	22	0	0	2	0	0	0	0	1
SDMetrics.com.sdmetrics.metrics.Proce...	0	6	5	1	5	0	0	0	0	0	0	0	0	0	0	0	1
SDMetrics.com.sdmetrics.metrics.Varia...	2	Demo	4	1	3	0	0	0	Demo	0	Demo	0	0	0	0	0	1
SDMetrics.com.sdmetrics.metrics.FilterA...	3	5	3	0	1	1	0	0	0	0	Demo	0	0	0	0	0	0

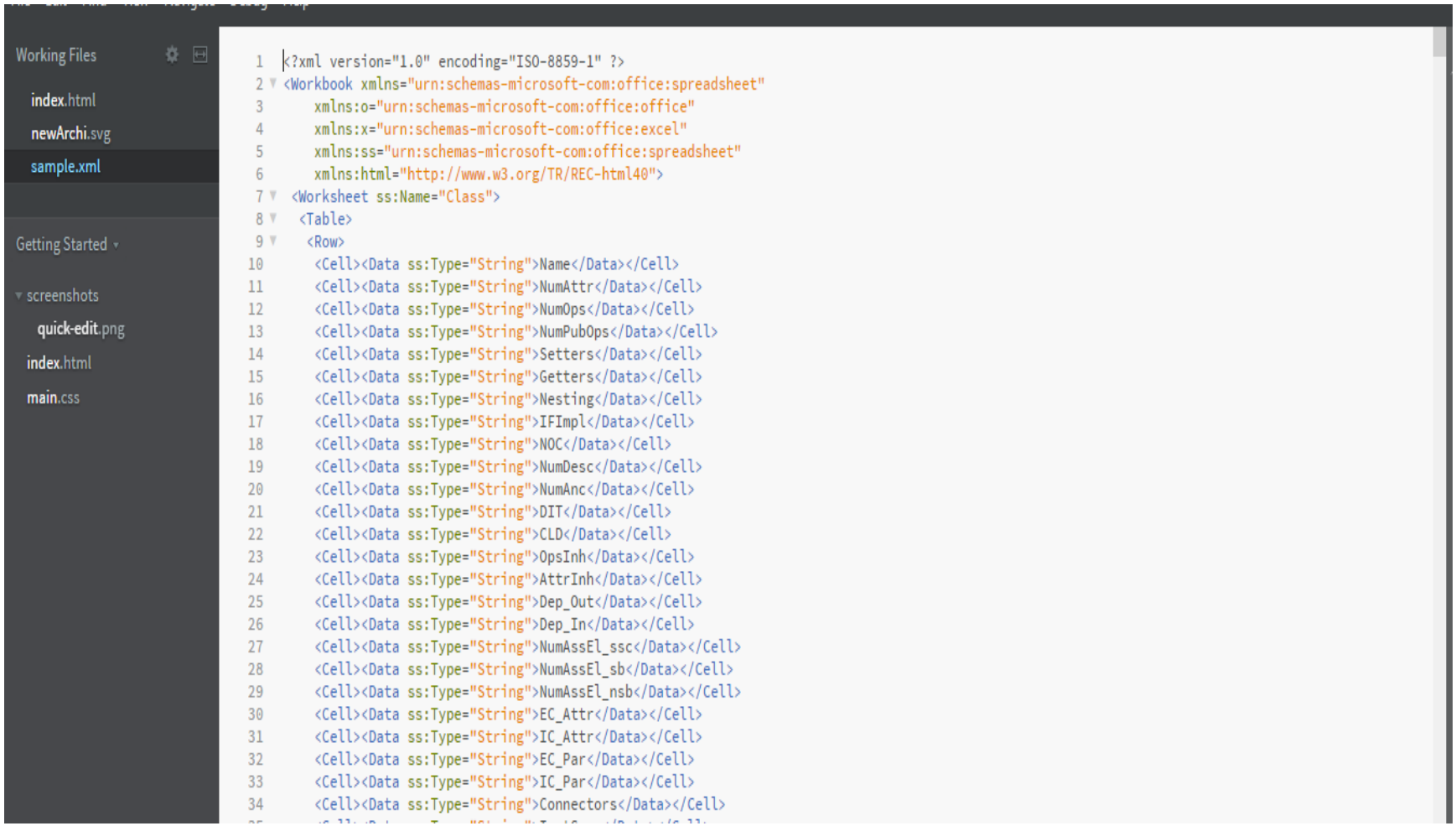
Annex B: Excel metrics sheet sample



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
3	Name	NumAttr	NumOps	NumPubOps	Setters	Getters	Nesting	IFImpl	NOC	NumDesc	NumAnc	DIT	CLD	OpsInh
4	SDMetrics.com.sdmetrics.OpenCore	0	1	1	0	0	0	0	0	0	0	0	0	0
5	SDMetrics.com.sdmetrics.app.ConsoleMessageHandler	3	8	7	1	1	0	0	1	1	1	1	1	5
6	SDMetrics.com.sdmetrics.app.MessageHandler	0	5	5	0	0	0	0	1	2	0	0	2	0
7	SDMetrics.com.sdmetrics.app.MetricData.MetricDataTable	1	1	0	0	0	1	0	0	0	0	0	0	0
8	SDMetrics.com.sdmetrics.app.MetricData	1	13	13	0	11	0	1	0	0	0	0	0	0
9	SDMetrics.com.sdmetrics.app.ModelData.TableInfo	1	1	0	0	0	1	0	0	0	0	0	0	0
10	SDMetrics.com.sdmetrics.app.ModelData	0	14	13	0	13	0	1	0	0	0	0	0	0
11	SDMetrics.com.sdmetrics.app.Problem	2	5	5	0	2	0	0	0	0	0	0	0	0
12	SDMetrics.com.sdmetrics.app.RelationMatrices	0	12	12	0	10	0	1	0	0	0	0	0	0
13	SDMetrics.com.sdmetrics.app.RuleData	2	18	14	1	13	0	1	0	0	0	0	0	0
14	SDMetrics.com.sdmetrics.metrics.MetricsEngine.MetricVal	0	4	0	2	2	1	0	0	0	0	0	0	0
15	SDMetrics.com.sdmetrics.metrics.MetricsEngine	0	14	6	0	5	0	0	0	0	0	0	0	0
16	SDMetrics.com.sdmetrics.metrics.SDMetricsException	1	5	4	0	2	0	0	0	0	0	0	0	0
17	SDMetrics.com.sdmetrics.metrics.Metric	2	6	5	1	3	0	0	0	0	1	1	0	13
18	SDMetrics.com.sdmetrics.metrics.MatrixEngine	0	4	3	0	2	0	0	0	0	0	0	0	0
19	SDMetrics.com.sdmetrics.metrics.MatrixData	0	8	7	0	7	0	0	0	0	0	0	0	0
20	SDMetrics.com.sdmetrics.metrics.RuleEngine	2	6	5	0	2	0	0	0	0	0	0	0	0
21	SDMetrics.com.sdmetrics.metrics.MetricStore.WordList	2	3	0	0	1	1	0	0	0	0	0	0	0
22	SDMetrics.com.sdmetrics.metrics.MetricStore.MetricParse	15	18	3	1	3	1	0	0	0	1	1	0	3
23	SDMetrics.com.sdmetrics.metrics.MetricStore	2	18	11	0	17	0	0	0	0	0	0	0	0
24	SDMetrics.com.sdmetrics.metrics.RuleFilter	0	7	4	0	0	0	0	0	0	0	0	0	0
25	SDMetrics.com.sdmetrics.metrics.RuleViolation	1	4	4	0	3	0	0	0	0	0	0	0	0

sample

Annex C: XMI schema document sample



Annex D: Correlation Matrix Table

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Correlation	1	**1.000**	.349	.254	.182	.204	.080	-.060	-.050	-.046	-.175	-.175	-.059	-.210	-.104	.136	.197	.440	.321	.123	-.075	-.099	-.094	-.045	-.073	-.093
	2	.349	**1.000**	.718	.490	.741	-.220	.225	.087	.160	-.180	-.180	.235	-.118	-.057	.468	.480	.456	.339	.479	-.141	-.186	-.176	-.084	-.136	-.174
	3	.254	.718	**1.000**	.346	.768	-.245	.461	-.042	-.068	-.176	-.176	.007	-.161	-.020	.332	.325	.496	.390	.243	-.115	-.153	-.144	-.069	-.111	-.143
	4	.182	.490	.346	**1.000**	.376	-.094	-.038	.211	.249	-.090	-.090	.239	-.104	.018	.305	.361	.266	.463	.174	-.053	-.070	-.066	-.032	-.051	-.066
	5	.204	.741	.768	.376	**1.000**	-.219	.460	.019	.089	-.245	-.245	.042	-.178	-.089	.447	.468	.567	.426	.402	-.093	-.123	-.116	-.056	-.089	-.114
	6	.080	-.220	-.245	-.094	-.219	**1.000**	-.102	-.105	-.094	-.155	-.155	-.132	-.216	-.101	-.203	-.030	-.087	-.121	-.140	-.068	-.090	-.085	-.041	-.066	-.084
	7	-.060	.225	.461	-.038	.460	-.102	**1.000**	-.051	-.046	-.141	-.141	-.064	-.135	-.111	-.059	-.047	.112	-.061	.034	-.033	-.044	-.041	-.020	-.032	-.041
	8	-.050	.087	-.042	.211	.019	-.105	-.051	**1.000**	.718	.018	.018	.659	.084	.009	.029	.001	-.082	-.032	.061	-.034	-.045	-.043	-.021	-.033	-.042
	9	-.046	.160	-.068	.249	.089	-.094	-.046	.718	**1.000**	-.035	-.035	.740	.004	-.031	.035	.002	-.072	-.035	.301	-.031	-.040	-.038	-.018	-.029	-.038
	10	-.175	-.180	-.176	-.090	-.245	-.155	-.141	.018	-.035	**1.000**	1.000	-.022	.899	.612	-.226	-.266	-.221	-.116	.000	-.094	-.124	-.117	-.056	-.091	-.116
	11	-.175	-.180	-.176	-.090	-.245	-.155	-.141	.018	-.035	1.000	**1.000**	-.022	.899	.612	-.226	-.266	-.221	-.116	.000	-.094	-.124	-.117	-.056	-.091	-.116
	12	-.059	.235	.007	.239	.042	-.132	-.064	.659	.740	-.022	-.022	**1.000**	.052	-.039	.018	-.028	-.052	-.030	.136	-.043	-.057	-.054	-.026	-.042	-.053

	13	-.210	-.118	-.161	-.104	-.178	-.216	-.135	.084	.004	.899	.899	.052	**1.000**	.682	-.217	-.279	-.208	-.090	-.046	-.090	-.119	-.112	-.054	-.087	-.111
	14	-.104	-.057	-.020	.018	-.089	-.101	-.111	.009	-.031	.612	.612	-.039	.682	**1.000**	-.073	-.117	-.087	.042	-.020	-.074	-.098	-.092	-.044	-.072	-.092
	15	.136	.468	.332	.305	.447	-.203	-.059	.029	.035	-.226	-.226	.018	-.217	-.073	**1.000**	.924	.430	.282	.645	-.076	-.100	-.094	-.045	-.073	-.094
	16	.197	.480	.325	.361	.468	-.030	-.047	.001	.002	-.266	-.266	-.028	-.279	-.117	.924	**1.000**	.443	.342	.564	-.092	-.121	-.114	-.055	-.088	-.113
	17	.440	.456	.496	.266	.567	-.087	.112	-.082	-.072	-.221	-.221	-.052	-.208	-.087	.430	.443	**1.000**	.712	.214	-.067	-.089	-.084	-.040	-.065	-.083
	18	.321	.339	.390	.463	.426	-.121	-.061	-.032	-.035	-.116	-.116	-.030	-.090	.042	.282	.342	.712	**1.000**	.021	-.041	-.054	-.051	-.024	-.039	-.050
	19	.123	.479	.243	.174	.402	-.140	.034	.061	.301	.000	.000	.136	-.046	-.020	.645	.564	.214	.021	**1.000**	-.087	-.115	-.109	-.052	-.084	-.108
	20	-.075	-.141	-.115	-.053	-.093	-.068	-.033	-.034	-.031	-.094	-.094	-.043	-.090	-.074	-.076	-.092	-.067	-.041	-.087	**1.000**	.320	.945	.116	.345	.745
	21	-.099	-.186	-.153	-.070	-.123	-.090	-.044	-.045	-.040	-.124	-.124	-.057	-.119	-.098	-.100	-.121	-.089	-.054	-.115	.320	**1.000**	.288	.025	.093	.220
	22	-.094	-.176	-.144	-.066	-.116	-.085	-.041	-.043	-.038	-.117	-.117	-.054	-.112	-.092	-.094	-.114	-.084	-.051	-.109	.945	.288	**1.000**	.256	.530	.834
	23	-.045	-.084	-.069	-.032	-.056	-.041	-.020	-.021	-.018	-.056	-.056	-.026	-.054	-.044	-.045	-.055	-.040	-.024	-.052	.116	.025	.256	**1.000**	.116	.640
	24	-.073	-.136	-.111	-.051	-.089	-.066	-.032	-.033	-.029	-.091	-.091	-.042	-.087	-.072	-.073	-.088	-.065	-.039	-.084	.345	.093	.530	.116	**1.000**	.507
	25	-.093	-.174	-.143	-.066	-.114	-.084	-.041	-.042	-.038	-.116	-.116	-.053	-.111	-.092	-.094	-.113	-.083	-.050	-.108	.745	.220	.834	.640	.507	**1.000**

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: *Melese Bitew Legese*_____

Signature: _____

Date: __*Monday, July 1, 2019*_____

Confirmed by advisor:

Name: __*Dr. Dida Midekso (Asso. Professor)*

Signature: _ _____

Date: _____*Monday, July 1, 2019*_____