A FEATURE RANKING ALGORITHM IN PRAGMATIC QUALITY FACTOR MODEL FOR SOFTWARE QUALITY ASSESSMENT

RUZITA AHMAD

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)
UNIVERSITI UTARA MALAYSIA
2013

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUMCollege of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Kualiti perisian adalah satu bidang penyelidikan yang penting dan telah mendapat perhatian dikalangan komuniti kejuruteraan perisian terutama dalam mengenal pasti atribut penting dalam proses pembangunan perisian. Tesis ini menerangkan penyelidikan asli dalam bidang model kualiti perisian dengan memperkenalkan algoritma *Feature Ranking Algorithm* (FRA) untuk model *Pragmatic Quality Factor* (PQF). Algoritma yang dicadangkan mampu memperbaiki kelemahan model sedia ada dalam mengemaskini dan mempelajari kombinasi atribut untuk penaksiran kualiti perisian. Teknik penaksiran sedia ada kurang keupayaan untuk menyenaraikan atribut mengikut keutamaan dan keupayaan pembelajaran data yang boleh meningkatkan proses penaksiran kualiti. Tujuan kajian ini adalah untuk mengenal pasti dan mencadangkan penggunaan teknik dalam bidang Kepintaran Buatan ke arah meningkatkan proses penaksiran kualiti dalam model PQF. Oleh itu, algoritma FRA yang menggunakan *Feature Ranking Technique* (FRT) telah dibina dan prestasi algoritma FRA telah dinilai. Metodologi yang digunakan terdiri daripada kajian teori, reka bentuk rangka kerja formal untuk kualiti perisian pintar, mengenal pasti kesesuaian ciri-ciri FRT untuk penyenaraian atribut, pembangunan dan penilaian algoritma FRA. Penaksiran atribut telah bertambah baik dengan menggunakan algoritma FRA yang mengandungi formula untuk mengira keutamaan atribut dan diikuti oleh adaptasi pembelajaran melalui aplikasi *Java Library for Multi Label Learning* (MULAN). Hasil kajian menunjukkan bahawa prestasi algoritma FRA mempunyai kolerasi yang sangat kuat dengan model pakar iaitu model PQF. Ujian statistik menunjukkan bahawa FRA telah menghasilkan keputusan ketepatan yang lebih baik berbanding algoritma Kolmogorov-Smirnov Correlation Based Filter (KSCBF) iaitu 98% berbanding 83% masing-masing. Ujian statistik juga menghasilkan keputusan bagi algorithm FRA iaitu 0.052 adalah lebih baik berbanding dengan algoritma KSCBF iaitu 0.048. Ini menunjukkan bahawa keputusan FRA adalah lebih signifikan berbanding algoritma yang digunakan. Sumbangan utama kajian ini adalah dalam pelaksanaan teknik FRT yang memperkenalkan pengiraan *Most Priority of Features* (MPF) dalam algoritma FRA untuk teknik penaksiran tersebut. Kesimpulannya, penemuan kajian ini menyumbang kepada usaha penyelidikan baru dalam bidang pemilihan atribut dalam kualiti perisian.

Kata Kunci: Perisian kualiti, Algoritma FRA, Teknik Kepintaran Buatan, dan Mesin Pembelajaran

Abstract

Software quality is an important research area and has gain considerable attention from software engineering community in identification of priority quality attributes in software development process. This thesis describes original research in the field of software quality model by presenting a Feature Ranking Algorithm (FRA) for Pragmatic Quality Factor (PQF) model. The proposed algorithm is able to improve the weaknesses in PQF model in updating and learning the important attributes for software quality assessment. The existing assessment techniques lack of the capability to rank the quality attributes and data learning which can enhance the quality assessment process. The aim of the study is to identify and propose the application of Artificial Intelligence (AI) technique for improving quality assessment technique in PQF model. Therefore, FRA using FRT was constructed and the performance of the FRA was evaluated. The methodology used consists of theoretical study, design of formal framework on intelligent software quality, identification of Feature Ranking Technique (FRT), construction and evaluation of FRA algorithm. The assessment of quality attributes has been improved using FRA algorithm enriched with a formula to calculate the priority of attributes and followed by learning adaptation through Java Library for Multi Label Learning (MULAN) application. The result shows that the performance of FRA correlates strongly to PQF model with 98% correlation compared to the Kolmogorov-Smirnov Correlation Based Filter (KSCBF) algorithm with 83% correlation. Statistical significance test was also performed with score of 0.052 compared to the KSCBF algorithm with score of 0.048. The result shows that the FRA was more significant than KSCBF algorithm. The main contribution of this research is on the implementation of FRT with proposed Most Priority of Features (MPF) calculation in FRA for attributes assessment. Overall, the findings and contributions can be regarded as a novel effort in software quality for attributes selection.

Keywords: Software Quality, FRA Algorithm, Artificial Intelligence (AI) Technique, and Machine Learning

Acknowledgement

In the name of Allah, the Beneficent, the Merciful. Alhamdulillah, grateful to Almighty Allah (SWT) for all the blessing He has bestowed upon me, lastly this thesis is finally completed.

A lot of thanks to Universiti Utara Malaysia for the supports and the facilities provided. My deepest gratitude and appreciation goes to my respected supervisors, Associate Professor Dr. Jamaiah Hj Yahaya and Dr. Siti Sakira Kamaruddin for never ending support, courage and helps along to complete this thesis.

Last but not least, I wish to express my deepest appreciation goes to my supportive husband, my loving mother, and my dear children, for believing in me, their continuous prayers and unconditional support. Lastly I would like to dedicate this work to the memory of my father, Allahyarham Ahmad bin Sulaiman (1938-2010) whom I unexpected lost during this study. Thank you all.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

AHS	Automatic Hybrid Search
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ARFF	Attribute Relation File Format
AUC	Area Under the Curve
BNS	Bi-Normal Separation
CBFS	Correlation Based Feature Selection
CBFSS	Consistency Based Feature Subset Selection
CBR	Case-Based Reasoning
CS	Chi-Square
DF	Document Frequency
DFM	Default F-Measure
DGM	Default Geometric Mean
ESD	Airforce Electronic System Division
FAS	Filter Attribute Selection
FCBF	Fast Correlation Based Filter
FRA	Feature Ranking Algorithm
FRT	Feature Ranking Technique
FS	Feature Selection
FSST	Feature Subset Selection Technique
GA	Genetic Algorithm
GR	Gain Ratio
GRNN	Generalized Regression Neural Network
GUI	Graphical User Interface
HFS	Hybrid Feature Selection
IBL	Instance Based Learning
IEEE	International Symposium on Requirement Engineering
IG	Information Gain

ISO	International Organization Standard
JRE	Java Runtime Environment
KNN	K-Nearest Neighbour
KS	Kolmogorov Smirnov
KSCBF	Kolmogorov-Smirnov Correlation Based Filter
K-S TEST	Kolmogorov Smirnov Two Sample Test
LEET	Large Experiment and Evaluation Tool
LOC	Lines of Code
LR	Logistic Regression
MATLAB	Matrix Laboratory
MI	Mutual Information
MLKNN	Multi Label K-Nearest Neighbour
MLOSS	Machine Learning Open Source Software
MLP	Multi Layer Perceptron
MPF	Most Priority of Attribute
MULAN	Java Library for Multi Label Learning
NB	Naïve Bayes
NN	Neural Network
OA	Overal Accuracy
PQF	Pragmatic Quality Factor
PS	Probabilistic Search
QFD	Quality Function Deployment
RADC	Rome Air Development Centre
RAKEL	Random *k*-Labelstes
RS	Rough Sets
SPSS	Statistical Package for the Social Sciences
SQA	Software Quality Assurance
SQuaRE	Software Product Quality Requirement and Evaluation
STS	Spring Source Tool Suite
SU	Symmetrical Uncertainty
SVM	Support Vector Machine

WEKA	Waikato Environment Knowledge Analysis
WLLR	Weighted Log Likelihood Ratio
WWW	World Wide Web
XML	Extensible Markup Language

CHAPTER ONE
INTRODUCTION

1.1 Overview

Chapter One presents the overall study and briefly explains the aims of the research. Several sections have been defined to classify and identify the purpose of this study. These include research background, problem statement of the research, research motivation, research objectives, scope of study and methodology.

1.2 Research Background

Nowadays, rapid development and diffusion of software quality is related to technologies in several industries. Statistics shows on insufficiently understood requirements accounted to 50% of errors. This was followed by design incorrectly understood from requirements, which accounted to 30% of errors. Hence, programming errors of system design contributed to 20% of errors (Humphrey et al., 1989). In fact, the organization has outlined the exactly errors in perfectly before they starts to develop a software product. Thus, Software Quality Assurance (SQA) is a very important domain in software development and its purpose is to find ways to reduce the rate and associated cost of failure from poor product and services (Humphrey et al., 1989).

In order to reduce errors in systems design and to fulfill user needs and requirements, the quality of systems development should be highlighted as an important goal. Normally, the standard level of quality is recommended by the International Organization for Standardization (ISO) and IEEE as well. ISO defines quality as the total of features and characteristics of a product and services that bear on its ability to satisfy stated or implied needs (ISO/IEC9126, 1991). IEEE defines software quality as a software feature or characteristic used to assess the quality of a system or component (IEEE, 1993). Furthermore, software quality is also defined as the fitness for use of the software product and conformance to software requirements and to provide useful services (Tervonen, 1996).

Later, software quality is defined as conformance to explicitly states that functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected for all professionally developed software (Tervonen, 1996). In many organizations, software is considered as one of the main assets with which the organization can enhance its competitive global positioning in current economic era. In past literatures, software quality have been qualified and assessed by the current quality models such as McCall (McCall et al., 1976), Boehm (Boehm et al., 1978), FURPS (Grady & Caswell, 1987), ISO 9126 (ISO/IEC 9126, 1991), Software Product Quality Requirement and Evaluation (SQuaRE) (The next generation of ISO/IEC 9126, 1999), Dromey Quality Model (Dromey, 1996), Systemic Quality Model (Callaos & Callaos, 2003), and Pragmatic Quality Factor (PQF) model (Yahaya et al., 2006).
All of these quality models are based on theoretical approaches and are known as static model. Most of the assessment methods used in existing quality models are not able to show how factors are evaluated. In fact, the current existing models have limitations to fulfill the transformation in the environment. This scenario is discussed in Chapter Two.

In order to cater for future high software requirements, different assessment methods should be used to cope with the problem that occurred in the software quality model itself. This study presents an intelligent software quality model that was developed based on an Artificial Intelligence (AI) approach. The model consists of an algorithm and mechanism for assessing quality characteristics. Furthermore, the algorithm incorporates a formula that acts as a medium to evaluate and assess the quality characteristics according to the values given by users, developers and independent assessors. The model is able to identify and recommend to the environment on the priority of attributes in the software development process. Moreover, the proposed new intelligent model is able to solve the weaknesses that exist in current models. This is to ensure that the software quality meets the nation and organizations requirements and meets current and future standards.

1.3 Research Problem Statement

The existing literatures on software quality models have consistently highlighted those models as static models. A static model refers to the development of a software product on time, within budget and efficient in performing all specified functions of requirements (Boehm et al., 1978). Although, researchers on software quality have been developed their own models to evaluate and measure a software product. Most of them had developed based on their experiences and theoretical approaches that include basic components of software quality (McCall et al., 1976; Boehm et al., 1978; Grady & Caswell, 1987; and Dromey, 1996).

Nowadays, the software quality environment is fast changing in terms of user requirements and needs to fulfill future requirements. Hence, the existing models in the literature have limitations to meet the future requirements due to the assessment technique used. Since, the latest quality model, PQF has focused on user approach and human aspects it is still considered as a static model. Hence, this model is also has weaknesses in updating necessary information that is derived from possible new combination of attributes during assessment. However, the evaluation and measurement techniques provided by PQF model were not tailored by using an advance tool that incorporates an intelligent mechanism such as learning capabilities. This is one of the limitations of the model and in order to meet current requirements in software quality model, an assessment technique using AI approach is needed.

In previous studies, AI approach has been widely used in software quality assessment. Some classification techniques used for software quality estimation include optimal set reduction (Briand et al., 2000), logistic regression (Khoshgoftaar & Allen, 1999; Schneidewind, 2001), decision trees (Khoshgoftaar et al., 2000; Suarez & Lutsko, 1999; Takahashi et al., 1997), neural networks (Khoshgoftaar et al., 1997; Paul, 1992; Pizzi et al., 2002), and case-based reasoning (Ross et al., 2002). Most of the studies were established to assist quality improvement efforts during operations in the software quality testing and enhancing resources. The latest studies by Gao et al. (2009) used four Feature Selection (FS) techniques such as Automatic Hybrid Search (AHS), Rough Sets (RS), Kolmogorov-Smirnov (KS) and Probabilistic Search (PS) and conducted experiments using the algorithms on a very large telecommunications software system.

The FS technique was used to remove irrelevant and redundant features from the original data set using filter approach. The FS technique used in this research consists of both approaches in feature selection such as filter and wrapper approaches. The development of an algorithm is an assessment technique that can improve software quality assessment in PQF model. The algorithm is known as Feature Ranking Algorithm (FRA). This algorithm includes a measuring technique in evaluating the priority of quality attributes using a formula known as Most Priority of Features (MPF). The learning adaptation through Java Library for Multi Label Learning (MULAN) application using classifiers like Random *k*-Labelsets (RA*k*el) and Multi Label *k*-Nearest Neighbour (ML*k*NN) is highlighted in this research. Furthermore, the developed algorithm provides an alternative in assessment on the quality of software product among users, developers and independent assessors.

1.4 Research Motivation

Firstly, the study presents an intelligent software quality model using Feature Ranking Technique (FRT) which is a type of FS technique to select a subset of relevant features for building learning models. The selected technique is widely used in the classification model of software quality estimation to perform prediction. Thus, the proposed technique can improve the assessment technique made by PQF model by incorporating the learning concept to identifying the priority of attributes. In fact, the new quality model provided an intelligent assessment technique to the software quality industry in Malaysia.

1.5 Research Objectives

The objectives of the research are:

I. to identify Feature Ranking Techique (FRT) as to improve Pragmatic Quality Factor (PQF) model.

II. to develop and evaluate an assessment technique in PQF model with the proposed Feature Ranking Technique (FRT).

1.6 Research Scope

The main scope of this research is to utilize the data adopted from PQF model developed by Yahaya et al. (2007) which is captured since 2007 until 2011. The description of the data is discussed in Chapter Two. The data was used as the input and it is claimed as important because the data were gathered from the previous study of software quality model in the literatures. The data were known as Efficiency, Functionality, Maintainability, Portability, Reliability, Usability, Integrity, and User Conformity. Another scope in this study is to apply Feature Ranking Technique (FRT) for assessment method in PQF model. The technique is used to develop an algorithm in providing an assessment method for attribute selection and adaptation of learning concept is applied to train and learn the data.

1.6.1 Research Methodology

This section explains the methodology of the research with the aim to develop a new intelligent software quality model as to achieve the objectives in this research. The following sub sections discuss the steps involved in the development process.

1.6.2 Theoretical Study

The sections reviews on existing studies related to software assessment. The studies were taken from references from journals, books, proceedings and other academic materials. The aim of this phase is to investigate the existing mechanisms and problems related to software quality. Furthermore, the important features that are expected to contribute in this research are identified.

1.6.3 Design of formal framework on intelligent software quality

The second phase of this research is designed the formal framework on intelligence software quality model. It involved identifying the specific features of software quality used FRT.

1.6.4 Identify and proposed the Feature Ranking Technique (FRT) for an intelligence software quality model

The third phase of the research is to identify and propose FRT technique for software quality model. Several techniques have been studied and the appropriate technique has been selected in this model.

1.6.5 Construction of an Feature Ranking Algorithm (FRA) algorithm

The fourth phase of the research is to construct a Feature Ranking Algorithm (FRA) algorithm with the proposed Feature Ranking Technique (FRT). The proposed technique discovered in the previous phase was used and integrated to construct an intelligent software quality model.

1.6.6 Evaluation of study

The development of an algorithm used to test and valid the intelligence model. This phase is very important to show that the intelligence model of software quality achieved using AI technique. Furthermore, the proposed algorithm is compared to PQF model and Kolmogorov-Smirnov Correlation-Based Filter (KSCBF) algorithm in the literature.

1.6.7 Research Contribution

This research contributes an enhancement of assessment technique in PQF model which is known as Feature Ranking Algorithm (FRA). The main contribution of this research is on the embedding of AI approach in software quality model. The application FRT with proposed Most Priority of Features (MPF) calculation in FRA for attributes assessment technique is achieved. In addition, the implementation of learning concept is reached through Java Library for Multi Label Learning (MULAN) application using classifiers such as Random *k*-Labelsets (RA*k*el) and Multi Label *k*-Nearest Neighbour (ML*k*NN). The detail of the proposed algorithm discusses in Chapter Four. Thus, the ideas of the existing quality model i.e. PQF model is used in terms of its components. It concludes behavioral characteristics, impact characteristic, responsibility, and weight. The detail explanation of this existing quality model is discussed in Chapter Two.

1.7 Thesis Outline

This thesis consists of six chapters and is structured as follows:

Chapter One: Introduction – The chapter includes the preliminary study that has been conducted, background of the study, problem statement, research motivation, objectives, scope of the research and methodology.

Chapter Two: Literature Review – It gives an overview of the current quality models in the literature including the strength and weaknesses for each approach. Furthermore, this section was mentioned about the elements of AI pertaining to the proposed technique and method, which relates and involves in developing an algorithm.

Chapter Three: Research Methodology – This chapter presents the research methodology that was used to achieve the research objectives. It gives an explanation of five sequential phases in the model development process.

Chapter Four: Development of Feature Ranking Algorithm (FRA) Model - This chapter discusses the algorithm in detail. It presents the concepts involved in the selected technique and method to evaluate quality attributes. Furthermore, it gives explanations on the formula used to solve data redundancy.

Chapter Five: Experimental Result – The FRA algorithm is evaluated based on expert review and results obtained from using KSCBF algorithm.

Chapter Six: Discussion and Conclusion – The chapter concludes the research findings and contribution. It also presents recommendations for future research. Besides that, this chapter also highlights the problems encountered research values.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

This chapter reviews on previous works related to this study. The purpose of this chapter is to discuss the preliminary study that has been carried out. Thus, it focuses on identifying the limitations in current works and problems decribed in Chapter One. Besides, it is also aimed at generating new ideas to enhance and support the constraints in the existing works. The discussion covers the definition of software quality and the significance of the issues. An overview of software quality models in the existing works has been investigated and they are discussed with special focus on the characteristic performance. Then, it is followed by the assessment techniques. Next, a discussion of PQF model and the compared method are presented. This is followed by a discussion on the elements of Artificial Intelligence (AI) approach and related works in software quality model. In the end, the chapter continues with a discussion on issues and problems faced in this study, followed by a summary of limitations and recommendations for future enhancements.

2.2 Definition of Software Quality

This section discusses the definitions of software quality, in which the term is understood in different ways by different individuals and organizations.

The International Organization for Standardization (ISO) defines software quality as a set of quality characteristics with different significance to fulfill the future requirements of software development. It has fully clarified the definition of term “software” and “quality” separately. Generally, ISO defines software as “all or part of the program, procedures, rules and associated documentation of information processing system”. Meanwhile, the term of quality is defined as “the totally of features and characteristics of a product or services that bear on its ability to satisfy stated or implied needs” (ISO/IEC 9126, 1996).

Thus, “software quality” is referred to as the application in developing a software product. Universally, the software product is referred to as a set of computer program, procedures, documentation rules, and the intended data to be used by user (ISO/IEC 9126, 1996). Further, the software quality consists of various attributes to behave in the software product with various functions provided by each characteristic. Hence, each characteristic will perform with different capabilities in fulfilling users’ requirements.

On the other hand, the International Symposium on Requirement Engineering (IEEE) which is also known as the body of standard for software quality metrics and methodologies that is responsible in providing rules in measuring the quality attributes. They defined software quality as software features which consists of a set of characteristics to act in software development (IEEE, 1993). In accordance, this study understands that the software quality refers to the behavior of the characteristics or known as attributes that influence the quality of systems. Thus, the interactions between each attribute in fulfilling user requirements are quite important in global competition.

Nevertheless, Garvin (1984) defines software quality as a “complex concept” in accomplishing user expectations. It conveys that the quality of software development is quite hard to achieve and it can be viewed from different perspectives including transcendental, user, manufacturing and value base view. The stated perspectives indirectly affect on the software product in term of user interactions with the final product.

This has been supported by Denning (1992), who argues that software quality is important to be stressed in software development in supports of user satisfaction. Denning (1992) believes that software quality should not be based only on technical aspects but also on human, hence accomplishing user requirement is necessary.

2.3 Software Quality Model

This section discusses the chronological order of software quality models in the literatures. It also describes the characteristics of software quality model.

2.3.1 Overview

The literatures reveal several software quality models that could be classified as static quality model. The best known in chronological order of appearance are McCall (McCall et al., 1976), Boehm (Boehm et al., 1978), FURPS (Grady & Caswell, 1987), ISO 9126 (ISO/IEC 9126, 1991), Software Product Quality Requirement and Evaluation (SQuaRE) (The next generation of ISO/IEC 9126, 1999), Dromey Quality Model (Dromey, 1996), Systemic Quality Model (Callaos & Callaos, 2003), and Pragmatic Quality Factor (PQF) model (Yahaya et al., 2007).

The earliest model in software quality is McCall model build in 1977 by the US Airforce Electronic System Division (ESD), the Rome Air Development Centre (RADC) and General Electric (GE) (Pfleeger et al., 2001). This model has grouped the quality attributes into product operation, product revision, and product transition. Product operation is based on the product ability that can affect on user friendliness. Meanwhile, product revision is related to the capability in handling error correction and system adaptation. In contrast, product transition is related to the distribution process in the software development.

The Boehm model is similar to McCall model, in which it presents a hierarchy of characteristics and it contributes to overall the quality (Yahaya et al., 2008). In Pfleeger et al. (1997) and Khosravi et al. (2004), the Boehm model has been addressed based on the collection of characteristics including the user needs and the characteristics that are not encountered in the McCall model. In contrast, FURPS model (developed by Hewlett Packard) combines the main characteristics into an acronym that makes the model's name. Each noun in this model name is referred to one characteristic. It divides the main characteristics into five characteristics and each consists of sub characteristics to be measured (Rawashdeh et al., 2006).

The next model is ISO/IEC 9126, developed in 1991. ISO/IEC 9126 is the International standard which is known as information technology software product evaluation quality characteristics and provides guidelines to develop and measure product quality (Azuma et al., 2001). The model defines product quality as a set of product characteristics. Additionally, the model is also recommended to the environment the internal and external characteristics of software product.

Later, Software Product Quality Requirement and Evaluation (SQuaRE) has been introduced in 1999 and it was completely in Madrid in 2000 (Azuma, 2001). The SQuaRE model is known as the next generation of the ISO/IEC 9126. The model is focused on requirements of specification, measurement and evaluation (Suryn et al., 2003). The strategy measurement in this model is adopted from ISO/IEC 9126 including the new general reference model, detailed guidelines, standard for measurement primitives, quality requirement and so forth (Suryn et al., 2003). The aim of this model is focused on the product side, which translates the required quality into characteristics, sub characteristics, and defines the relationship for each characteristic (Qutaish, 2009).

Next, Tomar (2011) describes that Dromey model values a product based on a quality model that recognizes the quality evaluation for each product. It is a broad quality model that works in different systems (Khosravi et al., 2004). In this model, the software product is divided into four areas: Correctness, Internal, Contextual, and Description. These four areas are internal properties which are related to the internal quality and measured at the source code.

The model is followed by Systemic Quality Model, which is proposed by Calloas and Calloas in 2003. It concerns on product efficiency and effectiveness. It identifies the relationship between product-process, efficiency-effectiveness, and user-customer to obtain global systemic quality (Ortega et al., 2003). The product-process efficiency refers to the external attributes or known as product properties such as requirement, design, and implementation properties (Ortega et al., 2003). Meanwhile, product-process effectiveness is related to the user satisfaction and customer satisfaction which is focused on identification requirement and interfaces design (Ortega et al., 2003).

The latest static quality model is Pragmatic Quality Factor (PQF), created by Yahaya et al. (2007). It aims at making an assessment on software product for certification. The model portrays good impacts to the software quality environment in terms of the assessment techniques. This model describes the relationships between attributes and clarifies them from un-measureable attributes to the measureable attributes using the measurable metrics. The assessment technique consists of four components i.e. behavioral characteristics, impact characteristics, responsibility, and weight.

Among all the models described in the previous paragraphs, the PQF has been used as a benchmark in providing a guideline to measure the quality attributes in this research. Consequently, the assessment of components included in this model is applied in this research and indirectly enhances the quality assessment technique in PQF model using the proposed AI approach. The details of PQF model is discussed in the next section as an expert model in this research.

2.3.2 Software Quality Model Characteristics

Findings from the literatures show that consists of several attributes which are inherited from the previous models and contributes the new attribute related to the new requirement in respected era. In overall, the models contain various attributes covering various dimensions. Thus, the identification of quality attributes that is meets the user requirements and expectations extremely difficult. Most of software quality models were developed based on experiences, theoretical, and practical approach.

In short, the main quality attributes found in most of the models are Efficiency, Reliability, Usability, Portability, Functionality, and Maintainability. In fact, this is a set of attributes provided by ISO/IEC 9126 as a standard model. Most of the quality models in the literatures have used the standard quality attributes and also recommend to the environment the new required attributes to be highlighted. As an example it could be seen in SQuaRE model (Azuma, 1991) and Systemic model (Ortega et al., 2003). Also, Dromey model has contributed eight quality attributes in which six attributes are inherited from ISO/IEC 9126 and two additional attributes are Reusability and Process Maturity (Khosravi et al., 2004). Nevertheless, the latest model such as PQF model also includes the recommended attributes by the standard model and the additional attribute such as Integrity and User Conformity attribute (Yahaya et al., 2007).

In earlier development of McCall, it consisted of 55 attributes to be measured. Later, it has been reduced to 11 for easier analysis. In this model the attribute is called factor, which are Maintainability, Flexibility, Testability, Portability, Reusability, Interoperability, Correctness, Reliability, Usability, Integrity and Efficiency. According to the Pfleeger et al. (2001) and Khosravi et al. (2004), Boehm model also includes user needs as available in McCall. It adds with other attributes i.e. Understandability, Human engineering and Changeability. Also, Boehm model includes general utility characteristics which are broken down into Portability, Utility and Maintainability. Furthermore, Utility attribute is divided into Reliability, Efficiency and Human Engineering (Rawashdeh et al., 2006). In fact, the model classifies the characteristics from the views of end user in different locations and time (Rawashdeh et al., 2006).

Nevertheless, the FURPS model is quite different from the previous models in the literatures in terms of the types of categories provided in the model. They are functional requirements (F) and non-functional requirements (URPS). The combination of words consists of five (5) quality attributes in system development. In detail, the Functional requirements (F) are related to the input and expected output, while Non-Functional requirements (URPS) refers to Usability, Reliability, Performance and Supportability (Rawashdeh et al., 2006). The detail on the characteristics and attributes of each model is summarized in Table 2.1 below (Source: Yahaya & Deraman, 2008).

Table 2.1: Quality Characteristics in Previous Software Quality Models (Source: Yahaya & Deraman, 2008)

Quality Characteristics/ Software Quality Models	McCall (1976)	Boehm (1978)	FURPS (1987)	ISO 9126 (1991)	Dromey (1996)	SQuaRE (1999)	Systemic Quality Model (2003)	PQF (2007)
Testability	*	*						
Correctness	*							
Efficiency	*	*	*	*	*	*	*	*
Understandability		*			*			
Reliability	*	*	*	*	*	*	*	*
Flexibility	*							
Functionality			*	*	*	*	*	*
Human Engineering		*						
Integrity	*						*	*
Interoperability	*							
Process Maturity					*			
Maintainability	*	*	*	*	*	*	*	*
Changeability		*						
Portability	*	*		*	*	*	*	*
Reusability	*							
Usability			*	*		*	*	*
Performance	*		*					
User Conformity								*

* *is refer to the selected atrributes*

2.3.3 Summary of Software Quality Model

Most of the software quality models in the literatures including McCall, Boehm, FURPS, ISO 9126, SQuaRE, Dromey, Systemic Quality Model, and PQF model have been developed by theoretical approach, developer's view, and the experiences with their own characteristics. The developer for each existing software quality model has contributed beneficial ideas in generating a software quality model. As an example, McCall has divided quality characteristics into the product operation, product revision and product transition. The main characteristics are Maintainability, Flexibility, Testability, Portability, Reusability, Interoperability, Correctness, Reliability, Usability, Integrity and Efficiency. Later, Boehm model improved the McCall model by incorporating user needs and suggesting general utility characteristics i.e. Portability, Utility and Maintainability.

FURPS model contributes five characteristics (Functionality, Usability, Reliability, Performance and Supportability) which are divided into functional requirement (F) and non-functional requirement (URPS). Also, ISO/IEC 9126 (standard model) contributes standard characteristics i.e. Efficiency, Reliability, Usability, Portability, Functionality, and Maintainability. On top of that, this standard model presents the guidelines to the developers in generating a software quality model.

Consequently, the SQuaRE and Systemic models used the standard characteristics similar to the standard model in measuring the quality attributes. Particularly, SQuaRE model improves the new evaluation process. Meanwhile, the Systemic model identifies the relationships between product-process, efficiency-effectiveness and user-customer to obtain global systemic quality. However, it is a little bit different in Dromey model that contributes eight characteristics inherited from the standard model. It introduces Reusability and Process Maturity characteristics. In addition, the PQF model presents the Integrity and User Conformity characteristics as the new characteristics in the model. It aims at clarifying the un-measureable attributes to the measureable attributes using the measureable metrics.

2.4 PQF Model

This section discusses the PQF model in detail. The model is still known as a static model due to the capabilities in fulfilling the requirements and expectations in the future. As a result, this model has been established and recommended for a better software quality standard and procedure to assess quality attributes. In fact, this model has been accepted for practice in the industry and has been used in several large organizations in Malaysia (Yahaya et al., 2010; 2008). Further, the following sub sections discusses on the PQF model pertaining to the component, assessment technique, strength, and the limitation.

2.4.1 Component of PQF Model

The model was created by Yahaya et al. (2007) with a specific goal that is to make an assessment on software product for certification. As reviewed in the previous section, it is an excellent quality model because it describes the relationships between attributes. This model shows the assessment technique on un-measurable attributes using the measurable metrics, which consists of four components (behavioral characteristics, impact characteristics, responsibility, and weight). The details of each component are presented in Figure 2.1.

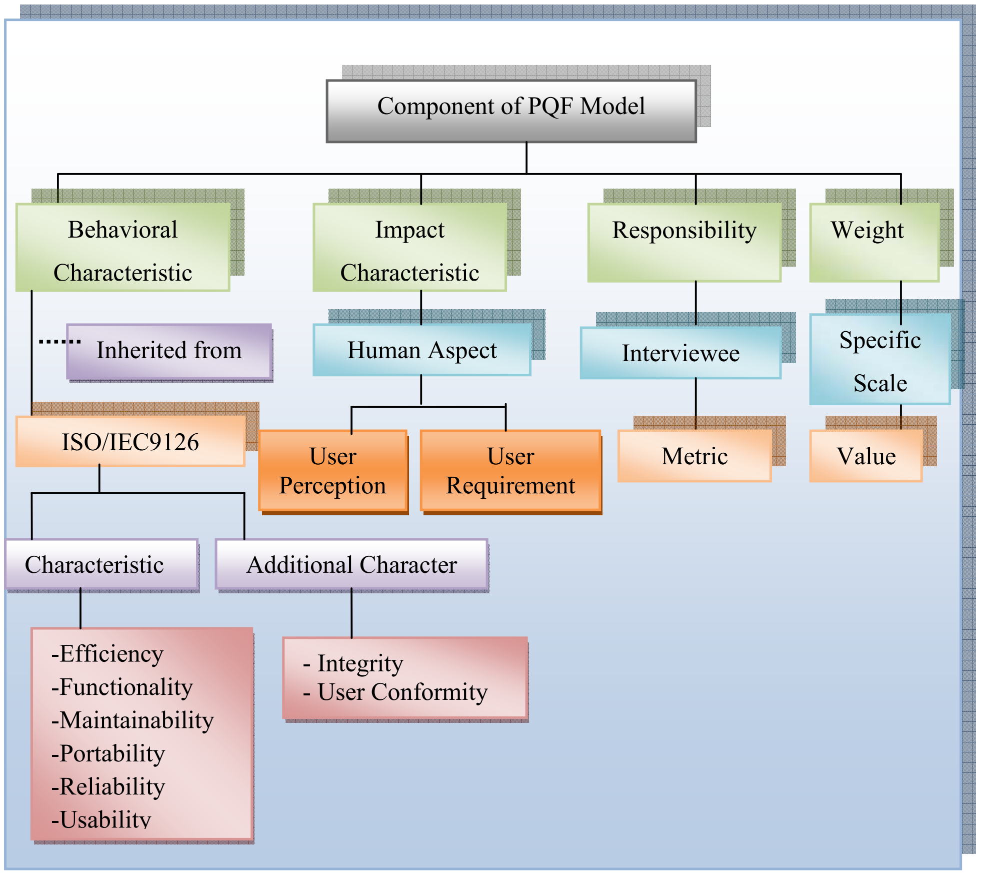


Figure 2.1: Component of Pragmatic Quality Factor (PQF) Model

Yahaya et al. (2007) describes the behavioral attribute as the internal quality characteristic of software development. The statement shows that the behavioral the attribute and human perspectives act as important elements to be stressed on the quality environment. In this model, the behavioral attribute is inherited from ISO/IEC9126 model including new additional attribute (Integrity and User Conformity). In particular, the Integrity attribute is pertaining to the security aspect and very critical to be pointed out as an important attribute in software development. Meanwhile, the User Conformity is related to the impact attribute such as user perception and user requirement. Based on the discussions in the previous section, this study believes that the attribute is extremely important to balance the quality model between technical measurement of software and human factor (Yahaya et al., 2008).

The second component refers to the impact characteristic, which is divided into two categories: user perceptions and user requirements. In particular, user perception measures the popularity, performance, trustworthiness, law and regulation, recommendation, environment, and adaptability, while user requirement measures the user acceptance and satisfaction. In this model, it decomposes all the attributes into the metric that enables the assessment on each attribute.

Further, the third component in PQF model is responsibility. This component acts as the main role that directly involves users who are the interviewees that are responsible in assigning the weight values by judging the priority of each attribute. The users consist of developers, managers, and assessors who are hired as experts for several years to assign the weight values. In this case, five point Likert scale is used which is based on collaborative perspective among assessment team members (Yahaya et al., 2008).

In the assessment, the scale is stressed on the satisfaction of the stakeholders in accepting the quality attributes in product. Subjects are expected to express their agreement or disagreement in numerical values (one to five: 1 = unacceptable, 2 = below average, 3= average, 4 = good and 5 = excellent). All the values are counted using a specific metric performed in the fourth components of PQF model.

The weight factor is the fourth component in PQF model. A specific metric has been performed in a formula that counts the priority of quality attribute. The information of the PQF weight calculation is presented in general. Based on original data collected from several large organizations in Kuala Lumpur, Malaysia, the following findings were gathered (Yahaya et al., 2008). In the study, the stakeholders were responsible in giving the weight value for each attribute depending on their experiences in the organization. Further, each of the attribute is sorted into the three classifications with respect to the calculated weight score as exhibited in Table 2.2.

Table 2.2: Classification of Attributes and Weight Factor (Source: Yahaya & Deraman, 2010)

Levels	Sub Attributes	Weight Factor
Low	Flexibility Intra-operability Inter-operability Portability	1-4
Moderate	Safety Efficiency Maintainability	5-7
High	Functionality Reliability	8-10

Table 2.2 divides the weight factors into three levels i.e. low, moderate, and high. It is seen that, Flexibility, Intra-operability, Inter-operability, Portability and Survivability attributes with scores of 1 to 4 are in the low level. This explains that this quality attributes have less priority in software development. In contrast, Safety,

Efficiency, Maintainability, and Usability attributes with scores of 5 to 7 are moderate, while Functionality, Reliability and Integrity with scores of 8 to 10 are high.

2.4.2 Assessment Technique in PQF Model

The assessment technique in PQF consists of technical aspect that stresses on two main components (the behavioral attributes and the impact attributes). The previous section explains that, the behavioral attributes deal with assessing software product to ensure the quality of the software and how it behaves in the environment (Yahaya et al., 2010; 2008), while the impact attributes deal with the reaction of the software product and its impact on the environment. These two components can generate stability between the technical requirement and the human factor.

On the other hand, Bevan et al. (1999) explains that the software measurement can be categorized into direct and indirect measurement. Particularly, direct measurement includes Lines of Code (LOC) which consists of execution speed, memory size, and faulty in period of time. Meanwhile, indirect measurement includes Functionality, Complexity, Efficiency, and Reliability (Khoshgoftaar et al., 2003). In this case, all attributes and sub attributes are known as un-measurable attributes in PQF model.

Bevan et al. (1999) is also defined metric as a quantitative measurement for attributes weight assigned by the assessors that can be used to estimate the quality attributes. In PQF model, the technique used in measuring the un-measurable attributes is the measurable metrics. According to the developer (Yahaya et al., 2010, 2008, 2007), the un-measurable attributes are decomposed into several sub attributes and metrics. The standard quality attributes are obtained from the ISO/IEC 9126, which consists of six attributes and includes new attributes in PQF model (Integrity and User Conformity attributes). Further, they are broken down into sub factors for estimation. In conjunction, the decomposition of the un-measurable attributes is shown in Table 2.3. It is adapted from Yahaya et al. (2008) on the comparison of quality score obtained by Cases X, Y, and Z industries in Malaysia through PQF model.

Table 2.3: Comparison of Quality Score Obtained by Case X, Y and Z (Source: Yahaya et al., 2008)

	Quality Attribute	Case X	Case Y	Case Z
		Score/5.00	Score/5.00	Score/5.00
1	Efficiency	3.73 (74.6%)	4.08 (81.6%)	4.70 (94.0%)
	Time Behavior	3.56	4.33	4.50
	Resource Utilization	4.00	3.70	5.00
2	Functionality	3.62 (72.4%)	3.69 (73.8%)	4.96 (99.3%)
	Suitability	3.83	3.65	4.88
	Accuracy	3.33	3.20	5.00
	Interoperability	3.63	4.50	5.00
3	Maintainability	3.34 (67.8%)	2.66 (53.2%)	3.58 (71.6%)
	Analysability	3.61	2.63	3.05
	Changeability	3.13	2.20	3.25
	Testability	2.83	3.06	2.00
4	Portability	3.20 (64.0%)	3.55 (71.0%)	3.50 (70.0%)
	Adaptability	3.56	5.00	4.75
	Installability	2.77	1.80	2.60
	Conformance	4.00	4.80	5.00
	Replacebility	3.33	4.40	5.00
5	Reliability	3.30 (66.0%)	3.36 (67.2%)	4.50 (90.0%)
	Maturity	3.83	3.80	4.75
	Fault Tolerance	3.00	3.20	4.38
	Recoverability	3.00	3.00	4.33
6	Integrity	3.67 (73.4%)	3.83 (76.6%)	4.33 (86.7%)
	Security	4.00	3.87	4.33
	Data Protection	3.33	3.06	3.00
7	Usability	3.20 (64.0%)	2.95 (59.0%)	3.41 (68.2%)
	Understandability	2.56	3.44	2.72
	Learnability	2.76	2.93	3.40
	Operability	3.70	3.01	4.61
8	User Conformity	3.53 (70.6%)	3.67 (73.4%)	4.73 (94.7%)
	User Perception	3.56	3.84	4.67
	User Requirement	3.50	3.40	4.83

Table 2.3 exhibits that in Case Z, the score for Functionality is 99.3%. It is the highest, followed with User Conformity and Efficiency (94.7% and 94.0% respectively). Meanwhile, the score for Integrity attribute is 86.7%. Case X has lower scores with Efficiency score the highest, followed by Integrity and User Conformity at 74.6%, 73.4% and 70.6% respectively. This shows that Case X and Z share a similarity, in which Usability attribute scores the least (64.0% and 68.2% respectively).

In addition, score in Case Y shows that Efficiency attribute is the highest of (81.6%). It is followed by Integrity and User Conformity (76.6% and 73.4% respectively). Meanwhile, Maintainability attribute is the least (53.2%).

Based on these actual results, using the assessment technique in PQF model, it is obvious that Integrity and User Conformity are recommended by the experts as very beneficial to be stressed on software development with the user perception views.

2.4.3 Step of Assessment Technique

The first step in PQF assessment technique is to convert the un-measurable attributes to the measurable attributes. The sub factors of the attributes are decomposed to the third level metrics which are named as M1, M2, M3, M4, M5, M6, M7, M8, and M9, in which 'M' represents the metric while the numbering order refers to the quality attributes. This could be illustrated in a tree diagram as seen in Figure 2.2 shows the decomposition of Functionality attribute which is sourced from Yahaya and Deraman (2010).

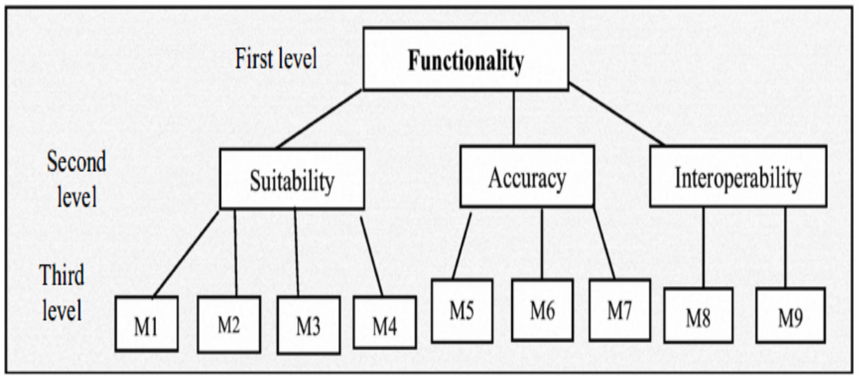


Figure 2.2: Decomposition of Functionality (Source: Yahaya & Deraman, 2010)

Referring to the diagram in Figure 2.2, the sub factors of Functionality are decomposed into the third level according to the weight given by the stakeholders. The metrics measurements are managed by the users or developers of the software product. They assigned weight for each attribute according to the quality of software product in timely basis. In this step, the assessment technique shows that the human judgment aspect is used on the behavioral characteristics in the software product.

The next step involves the impact attributes. According to Yahaya and Deraman (2010), the impact attributes illustrating the software impact to the users and the conformity of software to the user requirement is also evaluated. In fact, the impact attributes also decomposes to the sub attributes and metrics as to show the measurement of the attributes. It is known as user perceptions and user requirements. The user perception measures the elements such are popularity, performance, law and regulation, recommendation, trustworthiness, requirement and expectation, and environmental adaptability. In a complement, the user requirement measures the user acceptance and satisfaction. Five-point Likert Scale was used in collecting data based on perspective of assessment among team members. The management of the data and analysis was performed by Statistical Package for the Social Sciences (SPSS) analysis tool and the weight of each attributes is calculated using the in Equation 2.1.

$$\text{TotalVH} = \sum_{a=1}^{n} VH_a \qquad \text{(Eq. 2.1)}$$

Where n = number of attributes defined in the analysis and VH is the score for Very High Consideration. Further, the weight of the attributes is calculated using the following formula in Equation 2.2.

$$\text{Weight}_a = (VH_a / \text{TotalVH}), \qquad \text{(Eq. 2.2)}$$

Where a represents the selected attribute. The weight of the selected attributes is converted to the percentage value using the formula in Equation 2.3.

$$\%\ \text{Weight}_a = (VH_a / \text{TotalVH}) * 100 \qquad \text{(Eq. 2.3)}$$

Further, the next step is to use the function point approach to group and classify the attributes into three distinct classifications namely low, moderate, and high as discussed in section 2.4.1. The value of each attribute is sorted into these three classifications according to the calculated weight score according to Equations 2.2 and 2.3. Eventually, the model reveals the results pertaining to the quality attributes for the purpose of the assessment.

2.4.4 The Strength and Limitation of PQF Model

Referring to the literatures, it is found that PQF model has been applied in software certification model as the benchmark and standard model of the assessment. The new arrangement of the model as a software quality measurement has been shown and clearly defined including on the way to evaluate the quality attributes with human perspective approach. The aspects of quality as discussed before such as the behavioral and impact attributes are essential as to balance between the technical and non-technical aspects. Hence, the model provides flexibility by giving priorities and weight to the quality attributes. These two elements are necessary to reflect business requirement in the real business environment. Therefore, it is more practical and easy to be understood by the users, developers, and independent assessor pertaining to the way of assessment. In fact, this model shows how the un-measurable attributes can be measured indirectly by applying the measurement techniques and metrics approach.

Even though the PQF model seems like an expert model in the software quality community, it is also need to be improved in terms of its assessment technique by attaching the elements of Artificial Intelligent approach. In fact, the attachment of the dynamic elements can be more beneficial in the software quality environment with a self-learning capability with capturing knowledge from certification processes and experiences efficiently. Thus, the intelligent toolset should be capable of adapting and noticing the changes in environment and information needs. This can improve the limitation of PQF model in expanding its capability to fulfill the changes in the environment according to the assessment technique. Thus, the proposed technique in this study can enhance and overcome the limitation of PQF model.

2.4.5 Discussion of PQF Model

Recently, PQF model focuses on human aspect in order to give the priority value to each attributes in the software development. This opportunity provides more impact to the software product as to reflect the business requirement. While, the ISO/IEC 9126 as a standard quality model only addresses the external problem and excluded the human aspects in evaluating the quality attribute (Yahaya & Deraman, 2010). As mentioned in the previous section, the PQF model highlights the user requirements and user expectations clearly and has defined the way to evaluate quality attributes. It has been developed based on different perspectives, which highly focus on user's criterion perspective. Besides that, the quality of software product is claimed as good software quality when it achieves the user needs and expectations. Hence, the model also specifies the quality requirements in terms of high level quality attributes that meets the changes in the environment and meets the needs of the manufacturing view, which stipulates that quality as a conformance to a specification of requirements.

The benefits provided by the PQF model as the latest static model in the literature which presents the measurement metric to measure the un-measureable attributes is encouraging to be enhanced with dynamic element as proposed in this study. The PQF is also known as the excellent quality model and has been established and applied in large organizations in Malaysia. The assessment technique in PQF model is used as the benchmark and a baseline to be compared with the proposed assessment technique. Thus, the limitation faced in the PQF model can be solved by the proposed algorithm.

2.4.6 Summary of PQF Model

Element	**Details**
Component	Behavioral Impact Responsibility Weight
Assesment Technique	Focus on technical aspect such as behavioral attributes and impact attributes
Strength	1) The quality measurement is shown the way to evaluate the quality attributes with human perspective approach 2) The model provides flexibility by giving priorities and weight to the quality attributes 3) The model shows how the un-measurable attributes can be measured indirectly by applying the measurement techniques and metrics approach.
Limitation	Lack of capability in fullfilling the changes in the environment timely basis due to the assesmsent technique used is not tailored by element of intelligent in order to adapt and notice any other changes in the environment needs

2.5 Static Quality Model and Dynamic Quality Model

This section discusses on the dimension of software quality model, which is divided into static and dynamic. The static means fixed or permanent. It is unable to learn the changes occurred in in the context. In contrast, dynamic means it is capable and changeable in timely basis. The discussion on static and dynamic quality model is differentiated by the elements used to develop the model. It consists of assessment technique, assessment concept, and scope of assessment.

Most of the quality models in the literatures are static quality model. The latest established quality models including PQF incorporate the aspect of behavioral and human perspectives. They are still known as static model. The assessment technique used in most of the existing quality models is in structural forms (Yahaya et al., 2008). The structural forms are referred to the metrics in a form of checklist used to grade attributes of software development. Also, it defines the quality attributes via questionnaire. This technique defines the quality characteristics and clarifies the attributes, sub attributes, and examines the relationship among them.

According to Yahaya et al. (2008), the behavioral attributes is quite important to be stressed in the assessment. It will impact the users and conformity of software to fulfill user requirements. Previously, Denning et al. (1992) claimed that a quality model is not good quality if there is no user satisfaction aspect in software development. In regards to this, previous section has mentioned that PQF incorporates the aspect.

However, although this static quality model incorporates the user views in the assessment technique, it still has limitations and its components needs improvement to achieve the current and future requirements. The assessment concept used in the static model is theoretical approach which includes user, developer, and manager views. This refers to the theory, technical, and experiences as well. In fact, the views of stakeholders will define the different views and perceptions on quality. On top of that, Garvin et al. (1984) suggested that the quality of product during production is quite important to be highlighted. It shows that the scope of assessment in static quality model is focused on the quality of components and the functions in final products. Also, it shows the necessity to fulfill the transformation in the environment due to high specification of the software product itself. In fact, the static model supports building quality into products and processes.

The dynamic model is new advancement of software quality model which can be achieved using AI approach in the assessment technique. The elements involved in the dynamic model applied fully knowledge based approach with self-learning capabilities. This element is a toolset for adapting and noticing the changes in the future requirements and updated the changes respectively. The existing assessment technique such as structural forms is also used in the dynamic model to define the quality attributes through the user views. Then, to improve the assessment technique in static model, the adaptation of learning concept is highlighted in handling the classification of the data as to assess the priority of each attribute.

The concept used in the assessment technique in dynamic model is a practical approach. In this concept, the user views are considered as the main awareness to be highlighted in the software development. The dynamic model is also capable to interact in the environment if there are any attributes selections and modification occurs in future. Thus, the scope of assessment in dynamic model is highly focused on the priority of quality attributes that can be trained in timely basis. This is an additional purpose to improve the limitation in the existing quality model after the product and process quality is achieved.

With reference to the discussions in the previous paragraphs, Table 2.4 summarizes the element of static quality model and dynamic quality model in which the elements includes component assessment technique, assessment concept, and scope of assessment. In detail, the component assessment technique in static quality model is structural forms, which is used to define the quality attributes and clarify the relationship among them. In contrast, in dynamic quality model it involves AI approach to improve the component of assessment technique in the static quality model. Meanwhile, the assessment concept used in static quality model is theoretical approach which refers to theory, technical, and experiences of developer. Thus, in dynamic quality model uses practical approach which employed the user views in software development. On the other hand, the scope of assessment in static quality model is stressed on product and process quality, which supports the inclusion of quality into products and processes. On a contrary, the dynamic quality model highly focuses on the priority of quality attributes after implementation of product and process quality.

Table 2.4: Summary of Static Quality Model and Dynamic Quality Model

Element	Static Quality Model	Dynamic Quality Model
Component assessment technique	• Structural forms	• Structural forms • Artificial Intelligence (AI) approach
Assessment concept	• Theoretical Approach	• Practical Approach
Scope of assessment	• Product and process quality	• Product and process quality • Priority of quality attributes

2.6 Artificial Intelligence (AI) Approaches in Software Quality

Olivier (2001) defines AI as the application of computers, desiged to model the behavioral aspects of human reasoning and learning of the data. Later, Wenger

(2004) explains that AI involves an attempt to model the reasoning process in solving a problem either in natural language processing, algorithm, the proof of a theorem and so forth. Earlier, Pomerol (1997) argues that AI approach is able to sense and understand the conversations, human reasoning, and make a decision as would as human judgement.

Literatures show that there are several studies in software quality make use of AI techniques for several purposes. This includes studies by Khoshgoftaar, Szabo, and Guasti (1995), Lees, Hamza, and Irgens (1996), Khosgoftaar, Allen, Hudepohl, and Aud (1997), Khoshgoftaar, Chien, and Allen (1998), Khoshgoftaar, Nguyen, Gao, and Rajeevalochanam (2003), Khoshgoftaar, Gao, and Wang (2009) and Khoshgoftaar, Gao, and Napolitano (2009).

Khoshgoftaar et al. (1995) explored of the behavioral of neural network in software quality model. They compared two quality measures which are software complexity metrics and software quality metrics. The data were gathered from the components in software system and applied neural network as to train the data. The aim was to investigate the relationship between the two quality measures.

Later, Kumar et al. (1998) describes that Artificial Neural Network (ANN) is based on concepts of neuron or biological which consists of neuron connecting to the processing elements. The ANN are composed of two main structures namely the nodes and the links. The node is related to the neurons while the links related to the links between neurons. Further, the layer of nodes is referred to the hidden layer. Many ANNs contain multiple hidden layers and each feeds into the next layer. Before that, Khoshgoftaar et al. (1997) has also studied the ANN in which it is created from a network that the input data is already known. Meanwhile, the Multi

Layer Perceptron (MLP) is used as a classifier to learn the input data for the training data set.Additionally, in software quality model, data can be tested and trained using different classification algorithms such as Naïve Bayes (NB), K-Nearest Neighbour (KNN), Support Vector Machine (SVM) and Logistic Regression (LR) (Khoshgoftaar et al., 1998). The measurement on software quality models using NN involves a multiple regression quality model from the principal component of software. Particularly, the principal component of software refers to the data reduction technique that is used to reduce the dimensionality of multivariate data set. In this case, the dependent variable is a quality measure (Khoshgoftaar et al., 2003). On the other hand, the NN modeling identifies a list of dependent and independent variables. This is known as model selection and the selected variable is trained using the estimation technique in NN such as backward propagation algorithm, and forward propagation algorithm. Finally, the regression model will process the results to evaluate the quality models. However, the dimensionalities of the data are required to apply NN technique in evaluating the software quality models.

According to Kolodner (1992), Case-Based Reasoning (CBR) is a technique in adapting previous solutions stored in a library to solve new problems. In this technique, the CBR adapts the earlier problem to create a new solution in generating the new situation. The main purpose of CBR are to ensure the fitness for purpose of a software module, to identify an appropriate set of features which may be used and to describe the performance metrics and quality characteristics relating to each cases (Khoshgoftaar et al., 2003). Pertaining to the CBR, the quality attributes are measured by presenting a list of quality factors and determined the relationship among the quality factor. The establishment of quality factor uses metric performance like Quality Function Deployment (QFD), which quantifies the quality attributes and calculates the total quality measures for each attribute (Lees et al., 1996). CBR system has been very important in numerous fields including software cost estimation, software reuse and software quality estimation (Khoshgoftaar et al., 2003). Among the famous methods used in CBR in estimating the quality includes Kolmogorov-Smirnov (KS) method and the Kolmogorov-Smirnov Two Sample Test (K-S Test) (Khoshgoftaar et al., 2003).

Khoshgoftaar et al. (2003) applied the attribute selection method in CBR for software quality classification. The investigation on attribute selection was aimed to reduce the number of software metrics through CBR component in developing a software quality classification model. Particularly, the K-S Test was used in their study as a metric in determining the software metrics as an indicators of software quality. The research contributes that CBR technique was capable to develop software quality classification models in reducing the number of metrics in software quality for classification development.

Later, Khoshgoftaar et al. (2009) investigated for the same idea on software quality classification using Filter Attribute Selection (FAS) technique to improve the predictive accuracy of software quality models. The exploration on four different attribute selection techniques i.e. Automatic Hybrid Search (AHS), Probabilistic Search (PS), Kolmogorov-Smirnov (KS), and Rough set (RS) have been tested and the result found that the KS method performed better than the others in building classification models. Additionally, an extended research on exploration of software quality classification using wrapper approach has been carried out. They investigated several different performance metrics to influence the classification performance in software quality classification.

As a conclusion, the idea in this study is to enhance the static software quality model into a dynamic software quality model to improve and support the limitations in the static quality model. The incorporation of AI approach enables the process in noticing and adapting any changes occurs in the environment of software development. Therefore, the assessment technique used in dynamic quality model should bear the limitation of the existing quality model in measuring the priority of quality attributes. The related technique that highly focuses on attribute selection is Feature Selection (FS) technique. This technique is quite important and is able to process the selection quality attributes. The details of the FS technique are discussed in the following sub section.

2.6.1 Feature Selection Technique

Gao et al. (2009) defines FS as a process of selecting an attribute from relevant features in building a leraning model which is used to remove the less important features from training data set. In relation Khoshgoftaar et al. (2010; 2009) found that FS is an important activity in preprocessing data which is used in software quality modeling and data mining problems. They also describes that FS is divided into two categories namely Feature Ranking Technique (FRT) and Feature Subset Selection Technique (FSST).

Further, Gao et al. (2010) explains that FRT assesses the attributes individually and it will rank the attributes according to their individual predictive capability. The FRT evaluates the features individually and also will sort the attributes in term of the scores of each feature accordingly. It is required in a small sample size with rapid execution time to complete the function. Gao et al. (2010) applied the FRT technique with an aim to investigate either FS should be applied before or after data sampling. They found that the application of FS through FRT performed better and more stable in evaluating and ranking the attributes with the quality features.

The procedure of FRT is to score each feature in the data set using a particular method. The method counts the priority of features to be sorted and ranked in the list of features. In accordance, the FRT will use one of the methods in FS for allowing the selection of the best set of features. Several traditional methods which can be used in the FRT include document frequency (DF), Chi-Square (x^2 Statistic), Information Gain (IG), Gain Ratio (GR), and Symmetrical Uncertainty (SU). Additionally, alternative methods have been developed in several years by the researchers such as KS, AHS and Hybrid Feature Selection (HFS). All these methods are discussed in the next subsection.

In Wang et al. (2010) described that FSST is selects a subset of attributes in their predictive capability. Normally in FSST, the feature is evaluated using the classifiers which are contained in the black box as including the induction algorithm. This technique is suitable to be applied in the high dimensionality of data. In FSST, the input features are filtered independently using some classifiers such as SVM, NB, MLP, LR, Random k-LabelSets (RAkEL), and Multi-Label K-Nearest Neighbour (ML-kNN). Then, the results are prioritized. All of these classifiers are further discussed in the following section.

Referring back to FS technique, Tadeuchi et al. (2007) described that it contains two different approaches to subset selections, which are filter and wrapper approach.

The way of both approaches are applied is quite different to each other. The next following section is discusses on that matter.

2.6.1.1 Method in Feature Selection

This section discusses the methods in FS which are divided into two categories. There are traditional method and alternative method. The traditional method consists of DF, IG, SU, x^2 Statistic, BNS, WLLR, and Mutual Information (MI) (Yang et al., 1997). Meanwhile, the alternative method includes AHS, HFS and KS (Khoshgoftaar et al., 2009). In fact, DF, IG, x^2 Statistic, BNS and WLLR are related to the text categorization and they are not discussed in this thesis.

Literatures show that Friedman et al. (1997) has investigates that the MI using Bayesian Teorem as the baseline. The Bayes Teorem is also frequently referred to as Bayes' rule which is related to the probabilities theory. As an example it shows how a conditional probability B given by A can be inverted to yield the conditional probability A given by B. The teorem provides a way for considering two hypotheses and stresses on the probability of the data. It can be turned by a probability statement for a given data. Later, Yang and Pederson (1997) used MI in assessing two random variables by applying probability concept and the created formula was used to evaluate the score according to the data given.

On the other hand, SU is a very popular method in FS. It is used in preprocessing the irrelevant data and redundancy cases. It is stressed between features and the target concept which can be used to evaluate the goodness of features. Recently the SU method has been used for classifying data and it is relevant to be combined with Genetic algorithm (GA) in inductive learning strategy (Jiang et al., 2008). The SU is suitable to measure the correlation between features and the target concept by using the features corresponding weight. It is also used to guide the initialization of the population. The group of the irrelevant and redundant features is needed and the relationship between features is calculated using SU method.

According to the literature, many researchers are interested to apply the SU method in their proposed algorithm because SU is very easy to use and the results performed are unbiased as compared to other FS methods. In relation, Biesiada and Duch (2007) mentioned that the SU method has been used in filtering and sorting the irrelevant and redundancy of features in the Correlation-Based Feature Selection Algorithm (CBFS) and Fast Correlation-Based Filter Algorithm (FCBF) proposed by Yu and Liu (2003). Besides, a heuristic algorithm which is known as Relief algorithm proposed by Kira and Rendell (1992) has used this method to address the problem in averaging the relevance analysis of the candidate inputs in the class population. On top of that, the Kolmogorov-Smirnov Correlation Based Filter (KSCBF) algorithm also used the SU method as for averaging the weight of the features to find out the relevant features in their research (Biesiada & Duch, 2007).

On a contrary, the AHS alternative method is a modern FS method proposed by Wang et al. (2009). It processes the features with highest consistency rate and followed by the lowest consistency rate of the features. The selected feature is used to generate the superset. The process is repeated until the attribute subsets that have similar consistency rate value with selected feature is met. The method will involve a classifier such as C4.5 that appears in WEKA tool to learn the data. In regards to this, Khoshgoftaar et al. (2009) has used this method in their proposed algorithm for inducing the classification rules in the form of a decision tree.

Another new method is HFS that combined Filter-based Feature Ranking Technique (FRT) and AHS. It is also known as Consistency-Based Feature Subset Selection (CBFSS) algorithm. The HFS applies the combination of approach in FS such as filter and wrapper as discussed earlier. The proposed HFS method works in selecting the full feature set using FRT. The finding from the literature states that only thirty percent (30%) of the listing features are selected and the original data set is reduced (Khoshgoftaat et al., 2007). Also, the study contributes to the KS method to measure the maximum differences between the empirical distribution function of the probabilities of instances in each class. They used the KS score statistic to evaluate the attributes. Eventually, the attributes are ranked based on the KS scores obtained. In fact, the KS method performs better in evaluating the priority of the features than the other alternative methods proposed by the study.

2.6.1.2 Filter Approach

Filter approach selects the features independently without using any algorithm to execute the function (Tadeuchi et al., 2007). This approach is suitable to be used in a small sized data while the learning process is not presented in this approach. This approach has an advantage in which, the scoring and ranking function are immediately completed. However, it is unable to solve the redundancy of data because no learning algorithm is involved in the implementation. In fact, the filter approach is used in filtering and sorting quality attributes.

In a complement, Wang et al. (2009) investigated that three methods of Filter Attributes Selection (FAS) using filter approach such as RS, PS, and KS. They found that the FRT using KS method performs better than the others in filtering the features. On the other hand, Kohavi et al. (1996) found that some algorithms such as FOCUS and RELIEF algorithms used filter approach. Both of the algorithms examine all features in data set and filter the relevant features to the target concept. Thus, both algorithms do not support in handling data redundancy.

2.6.1.3 Wrapper Approach

Langley at al. (1994) found that the wrapper approach is more valuable in removing and solving irrelevant features in the data. It selects the features using an algorithm to train and learn the data. It also involves learning adaptation in training and learning the data. The concept of learning ranges in simplicity and complexity in various areas and now the adaptation of learning concept in software quality model becomes a novelty of research in the software quality community.

The wrapper approach has some limitations in the implementation such as long processing time and slow data execution because it involves many algorithms to perform. The wrapper involves a classifier to calculate the estimated accuracy of the learning algorithm. This function is important to remove the unimportant features in the data set. In regars to this, Wang et al. (2009) addressed that AHS uses the wrapper approach in developing the algorithm. It applies C4.5 classifier as learning algorithm in handling data redundancy.

2.6.1.4 Embedded Approach

Embedded approach is third class in FS technique. It combines filter and wrapper approaches. The preprocessing of data is built into the classifier construction after the filtering function in the search for an optimal subset of features (Saeys et al., 2007). Embedded approach is a specific approach, which uses wrapper approach directly to prove the performance. This approach also includes the interaction with the classification algorithm and the preprocessing of data is less computationally complex than wrapper approach. It also builds the model feature dependencies, in which the developer can choose the type of classifier in executing the classification task. According to Wolf et al. (2003) discovered that embedded approach has a drawback, in which the selection of classifier will affect the performance in the classification task. Besides, Duda et al. (2001) have used the embedded approach in their research pertaining to decision trees weighted using NB classifier. Later, Guyon et al. (2003) and Weston et al. (2003) followed by applying the approach in their study on feature selection using weight vector of SVM classifier.

2.6.1.5 Discussion on FS Techniques and Approaches

The FS technique can be combined with any other approaches in order to build the learning process. As an example, Gao et al. (2010, 2009) combined the FRT with the MLP using some of performance metrics to evaluate the classification of FS. Also, the literatures show that FRT is suitable to be combined with filter approach in ranking the attributes by evaluating the scores of the attributes using a method in FS. Besides, some studies applied the filter approach in developing algorithms for handling data redundancy. Both contexts could be observed in CBFS algorithm by Liu et al., (2002), FCBF algorithm by Yu & Liu (2003) and KSCBF algorithm by Biesiada and Duch (2007).

Literatures also reveal that only a few of studies have implemented the FSST using wrapper approach rather than filter approach. This supports the statement by Langley et al. (1994) that the wrapper approach has scales for large data set and needs some of classifiers to be performed.This takes a long time in completing the execution task. As a result, some of the studies used FRT combined with filter and wrapper as an embedded approach using a method in FS. As an example, Tadeuchi et al. (2007) combined filter and wrapper approaches in developing a quick online application for attribute selection method. They used the Generalized Regression Neural Network (GRNN) as the classifier in adapting the learning process in removing the irrelevant features in the data set. Similarly, Wang et al. (2009) applied both approaches in proposing the HFS algorithm.

2.6.2 Classification of Software Quality

Classification task refers to the arrangement of data item into the different groups according to their similarities and differences (Tsoumakas et al., 2010). The arrangement of the data item can make counting the probabilities of each feature in the data set easier. Thwin et al. (2005) mentioned that classification task assigns the data item into a collection of categories or classes. The goal of classification task is to predict the data item in the classes, which separates them into different categories. Thus, the data item is classified using a method for training data in classification (Tsoumakas et al., 2010 & 2011).

Tahir et al. (2010) illustrates that the classification task consists of two a step-process or known as method namely model construction and model usage. They further clarified that model construction is a processing of defining the class of data item in the class label attribute. As an example, the data item is divided into High, Moderate, and Low classes. All data items are defined according to their weight value of each attribute. Additionally, classification task can involve more than one classifier to train the data. In the end, the classification process is moved to the second process i.e. model usage. In model usage, the classifier will determine the results for each data as for estimation on the priorities of each feature.

On top of that, the classification task is also important in handling data cleaning in reduce the noise and handle the missing values. In addition, the classification task can support in handling the relevance analysis as to remove irrelevant or redundancies attributes. Besides, it is also used for data transformation to generalize and normalize the data. In fact, the classification task is performed using classifier as a method in training and learning the data set. In conjunction, the next sub section discuses on appropriate tools and classifiers.

2.6.3 Learning Tool and Classifier

Learning adaptation is the central to intelligence and it requires knowledge as an input for training process. The process of learning is supported by machine learning. Machine learning is refers to a system that is able to acquire and integrate the knowledge automatically (Tsoumakas et al., 2009). The system in machine learning is capable to learn from the experiences, training, analytical observation, and produces the results effectively. In conjuction, examples of machine learning tools include Waikato Environment for Knowledge Analysis (WEKA), Java Library for Multi Label Learning (MULAN) and Large Experiment and Evaluation Tool (LEET).

WEKA is an established tool in machine learning and data mining which proposed in year 1993. The software tool was programmed in Java and distributed under the

GNU Public License (Robu et al., 2010). The aim of WEKA is to build the facility for developing machine learning technique in solving the data mining problems. It consists of several standard data mining techniques such as data preprocessing classification, regression, clustering and association. In practice, it appears in four applications i.e. Explorer, Experimenter, Knowledge Flow and Simple CLI. These applications were performed in the interfaces as for user friendly used (Baumgartner & Serpen, 2009).

Meanwhile, MULAN is a Java library for learning tasks developed by Tsoumakas et al. (2010) with aim to provide a machine learning tool for classification tasks in open source software. The development team group has decided to support the benefits provided by Machine Learning Open Source Software (MLOSS) which is presented by Sonnenburg et al. (2007) in encouraging people to work with multi label data. It provides a multiple tasks such as classification, ranking, thresholding and dimensionality reduction algorithms. Besides, it works with multi label data which consists of training examples that are combined together with a subset of a finite set of labels (Tsoumakas et al., 2011). In addition, Tsoumakas et al. (2011) has described that multi label data is referred to a single set of data consisting of more than one feature which is called as label in data mining for easier referencing. Additionally, it is also a function to train and evaluate the data using more than one classifier (Tsoumakas et al., 2010). It inherits the functions available in WEKA but it does not have Graphical User Interface (GUI). All applications which covered in MULAN are imported from WEKA tool through command lines. This is one of the limitations in using MULAN that the command guideline is unavailable as for user referencing.

Baumgartner and Serpen (2009), has proposed the Large Experiment and Evaluation Tool (LEET) as a software workbench for data mining. It aims at simplifying the classification tasks provided by WEKA tool. It has been mentioned previously that WEKA incorporates a variety of tasks which are difficult to apply and most of the functions provided are not practical to be used. In contrast, LEET provides the experiments and evaluations with many algorithms and dataset through easy and user friendly to execute.

The features performed by LEET are classified into three tasks. There are executing the classification experiments using WEKA's built-in classifiers, evaluating the executed experiments to obtain performance measures, and evaluating datasets to calculate characteristics. User will choose the classifier which is provided by LEET through the interfaces. The execution results are provided in a single file, in which the results are stored and displayed in individual files for each simulation.

The learning algorithm is consists of some learner or well known as classifier. The classifier is used to evaluate and validate the performance result from the learning process. Khoshgoftaar et al. (2003) describes that the different classifier will impact to the difference performance result. This means that, the performance results are also depending on the capability of the classifier and the metric used in the classifier itself. Ideally, a good classifier will produce results closed to one (1). The famous classifiers include SVM, IBL, MLP, NB, RA*k*EL, and MLkNN.

2.6.4 Discussion on AI Approach in Software Quality Model

Literatures reveal that the FS technique is potential to be proposed in the construction of an attribute assessment algorithm in Feature Ranking Algorithm

(FRA). As discussed earlier, FS is divided into two categories namely FRT and FSST. The FRT has been selected as a dynamic technique to be used in this intelligent quality model. In conjunction, FRT is more relevant to be adapted into this study than the FSST due to its high performance in ranking the score of each attribute in a small sized sample. In fact, the FSST is relevant to work in high dimensionality of data. Therefore, this study plans to involve a small sample size. Besides, FRT is able to generate a function in the algorithm in term of ranking and sorting the new features in the data. This technique is embedded in the first phase of the algorithm which involves filter approach. As discussed earlier in the previous section, the filter approach is suitable to be applied because it does not involve any learning algorithm in the preprocessing data and the features are ranked and sorted independently. The flow in FRT technique is used in developing a formula to count the scores assigned by the assessors. In fact, the proposed formula acts as a new method in FS.

Based on the description in earlier section ANN and CBR are not relevant to be used in developing this proposed algorithm. This is partly because both of the techniques focus on the high dimensionality of data. The study from literatures also describe that ANN does not provide any function to calculate the weight of each attribute due because it presents networks with application of input from many traces.

The same reason also goes to another proposed technique like CBR. CBR is adapting the previous stored solutions in the CBR library to solve the new existing problems. It only focuses on relationship among attributes using QFD as a metric. In short, the functions provided by both techniques are unable to construct the assessment technique in intelligent software quality model. The performance metric used in both techniques do not concern on measuring the priority of quality attributes according to the weight given by the assessors. On top of that, both techniques are also lack of capability to fulfill the needs and future requirement to develop the learning process in the quality assessment technique.

In order to adapt the learning concept, the wrapper approach is also embedded in the second phase of the algorithm. In fact, Goodnow and Austin (1967) describe that the learning task is related to the human or machine learner in training and classifying the objects as referred to the related objects in the class labels. The application of wrapper approach is to build the learning algorithm that can support solving problem related to irrelevant and redundant data. As mentioned by Langley et al. (1994) and John et al. (1994), the wrapper approach will train all the features through the learning algorithm and proof the result through the capability of the classifier.

In addition, to implement the classification task of software quality, the MULAN tool is selected for executing the learning process. MULAN is the easier tool to use. In contrast, WEKA is not practical to be used and involves various tasks which are difficult to be understood by the user. Also, LEET is easy to be applied in presenting the interfaces and user friendly to execute. Based on that, WEKA and LEET are performed better in a large scale of data compared to MULAN (which is appropriate to small-scaled data). Since the data involved in this study are collected from a small sample size, then MULAN is appropriate to be selected in this study. Additionally, the easy classifiers i.e. RA*k*EL and MLkNN provided by MULAN are beneficial for handling data redundancy indirectly.

2.7 Feature Selection Algorithms

This section explains the FS algorithms in the literatures. It includes RELIEF algorithm, Correlation Based Feature Selection algorithm (CBFS), Fast Correlation Based Filter (FCBF) and Kolmogorov-Smirnov Correlation Based Filter (KSCBF).

2.7.1 Overview of FS Algorithm

The type of algorithm in the literature which is based on FS technique in preprocessing the data has been identified. They are RELIEF algorithm proposed by Kira and Rendell (1992), CBFS by Liu et al. (2002), FCBF by Yu and Liu (2003), and KSCBF algorithm by Biesiada and Duch (2007). All the algorithms are developed with the main goal, which is for handling the redundant features in data set using the techniques or methods recommended by FS approach.

Kira and Rendell (1992) proposed RELIEF algorithm using FS method i.e. Information Gain (IG) in estimating the quality attributes. The RELEIF algorithm measures the differences between the features with aim to differentiate the values among features that are close to each other. The RELEIF algorithm uses different probability via IG of FS method for filtering and ranking the quality of attributes. It assigns relevant weight to each feature to indicate the relevant features to the target concept. According to John et al. (1994), the RELEIF algorithm measures the features using two nearest neighbours search strategies such are *nearest hit* from the same class and *nearest miss* from different classes. The measurement in RELEIF algorithm focuses on high correlations of features and shows the weak relevant of features in the dataset. However, the RELEIF algorithm did not attend the redundancy problem in the data set.

Yu and Liu (2002) has proposed CBFS algorithm using Greedy hill climbing search strategies such as forward selection and backward elimination (Kittler, 1978) for training the samples of features. Then, the SU has been used for filtering and ranking the features. The CBFS algorithm is used to show the correlation between the features and the class of features. In regards to this, Hal et al. (1998) has mentioned that CBFS algorithm uses three selectors which are IBL, NB and C4.5 as the classifiers in the classification task. As the result, only C4.5 was found better than the other classifiers in showing the correlation among all features in the data set. However, the performance results are biased due to the failure of the algorithm to provide the validity of result especially in handling the data redundancy.

Later, Yu and Liu (2003), claimed that the development of FCBF algorithm is to enhance the capability and the performance of CBFS algorithm. The FCBF algorithm is based on predominant correlation, in which the correlation between features and classes is examined. It is used to solve the redundancy problem in high dimensionality of data. The algorithm consists of two stages. In stage one the SU is applied for filtering and ranking the features. It is important to show the relevancy of the features compared to the class. At the same time, the threshold value is selected to select the predominant features in the final ranking. Next, stage two is applied once the redundancy of features occurred. FCBF is a very fast algorithm to solve the redundancy of data. The algorithm compares the other features which are redundant to the predominant features. Consequently, features that are redundant to the highly relevant feature are automatically removed from the list without any measurement or validation.

Next, Biesiada and Duch (2007) proposed an algorithm to handle the redundancy of data by the strength of measurement technique in solving the data redundancy. The presented algorithm is named KSCBF is a successful algorithm in determining the validity of result. Thus, the KCBF algorithm is developed as to enhance the limitation of CBFS and FCBF algorithm. In relation, Blachnik et al. (2009) found that KSCBF has performed better than RELIEF, CBFS and FCBF algorithm in handling data redundancy. Thus, the advantages of KSCBF algorithm is encouraged this study to adopt it as the compared algorithm. Therefore, the component of KSCBF algorithm, strengths, and the limitation are further reviewed in the next section.

2.7.2 Component of KSCBF Algorithm

The KSCBF algorithm consists of three (3) components. They are Symmetrical Uncertainty (SU), Kolmogorov-Smirnov statistic (KS) and Kolmogorov-Smirnov Two Sample Test (K-S Test). SU is known as a traditional method in feature selection in preprocessing the list of attributes (discussed in section 2.6.1.1). In addition, SU is capable in averaging the weight of the features to find the relevant features in the ranking attributes (Biesiada & Duch, 2007). It has been widely used in handling data redundancy in several studies in the literatures. It operates using the formula below (Equation 2.4).

$$SU\ (X, C) = 2\ (MI\ (X,C\ /\ H(X) + H(C)) \qquad \text{(Eq. 2.4)}$$

Where, X is the selected features in the class attribute, C is class of the selected attribute, MI is the Mutual Information which is the basic quantity used for filtering method, H is uncertainty probabilities, H(X) is uncertainty of X features and H(C) is the uncertainty class of the selected attribute. The MI is basically contained in SU method formula (Li et al., 2009).

The second component is Kolmogorov-Smirnov statistic (KS). The KS statistics is used to compare two variables in the list of attributes. The formula of KS statistics is outlined in Equation 2.5.

$$KS_c\,(g,h) = \max_c\,(KS\,(g(c)\,,\,h\,(c)\,)) \qquad \text{(Eq. 2.5)}$$

Where, *c* is the class label, g(c) are samples of random variables *g* that belong to the class *c*, and h(c) are samples of random variables *h* that belong to the class *c*. The result from the statistics is then used to compare with the threshold to determine the existance of the data redundancy between the attributes. The threshold value is obtained by referring to the alpha value (δ) in KS statistic such as 1%, 5%, 10% and 20% (Blachnik et al., 2009).

The third component in KSCBF algorithm is Kolmogorov-Smirnov Two Sample Test (K-S Test). The K-S test is used to validate the priority of attributes in the final ranking result. The following formula is used for validation are as follows:

$$KS\,(g,h) = \sqrt{(ng)(nh)\,/\,(ng + nh)}\ \sup_k |\,G_k - H_k\,| \qquad \text{(Eq. 2.6)}$$

Where, *ng*, *nh* is the number of the samples for each attribute, *k* is the number of bins in discrete probability distribution, G and H are cumulative probability distributions of random variables *g* and *h* respectively. The random variables are referred to the pair of attributes in the ranking. The *sup* is refers to the highest differences between two random variables. All the components are associated in developing the measurement technique in KSCBF algorithm for evaluating the priority of attributes.

2.7.3 Assessment Technique in KSCBF algorithm

The assessment technique in KSCBF algorithm consists of two steps. In the first step, the algorithm trains the attributes in the data set using the SU method in feature FS for filtering and ranking the value of the attributes. The results are arranged in descending ordered. The second step is executed when there are data redundancies. In this stage, the algorithm will use the KS Statistic formula to obtain the threshold value by referring to the list of attribute's score from KS Statistic calculation. As an example, the last value of {33.4, 23.3, 18.3, 13.9, 12.2, 12.0, 9.3, 9.3} is 9.3, which is redundant. Hence, the alpha value (close to 9.3) is 10%. Consequently, the threshold value is equal to 10%, in which $\delta = 0.1$.

Then, the KS statistic is used to measure the highest differences between two variables attributes in the ranking through the K-S test formula. The K-S test will solve and validate the redundant attributes. In the loop features, the algorithm will initialize the first attribute in the ranking for comparison to the second attributes in the ranking. This function is repeated to the next attributes in the ranking until they find similar value of attributes from two variables. Eventually, the same value of two attributes is claimed as redundant.

In the KSCBF algorithm, the redundant attributes is removed directly from the ranking. As an example, if two attributes are redundant, then the algorithm will remove all the redundant features. However, the algorithm still places the redundant attributes in the ranking with different values after K-S test calculation. Usually, the KSCBF algorithm is used for analyzing data and will notice that only the most significant features are defined in the feature set. In addition, the KSCBF algorithm is also facing some strengths and limitationsas described in next sub section. In short, Table 2.5 summarizes the KSCBF algorithm.

Table 2.5: KSCBF Algorithm (Source: Blachnik et al., 2009)

Step	Algorithm
	Relevance analysis
1	Calculate the $SU(X,C)$ relevance indices and create an ordered list S of features according to the decreasing value of their relevance.
	Redundancy analysis
2	Take as the feature X the first feature from the S list
3	Find and remove all features for which X is approximately equivalent according to the K-S test
4	Set the next remaining feature in the list as X and repeat step 3 for all features that follow it in the S list.

2.7.4 The Strength and Limitation of KSCBF Algorithm

Each proposed algorithm has strength and limitation in processing the data. The researcher usually upgrades the proposed algorithm to enhance the performance presented by the algorithm. The KSCBF algorithm has used SU as the FS method in data preprocessing. The application of SU will equip the KSCBF algorithm with strong ability to train and learn the data. This is because the characteristic of SU is granted for stability in training and learning the data as well as filtering and ranking the data. As discussed in previous sections, SU uses probability distributions in estimationg the data. The ability of SU method can support the KSCBF algorithm in completing their relevance analysis on the data.

Additionally, the KSCBF algorithm also has several limitations. One of the limitations is that it only processes ordinal data which means the data can be counted and ordered directly. This explains that KSCBF is not able to train or learn the data if they merge between symbols and nominal features. Consequently, the determination of threshold selection is quite difficult to decide in order to conclude the validity of the hypothesis (Blachnik et al., 2009).

Also, the sensitivity of cumulative probability distribution to linear transformation is another disadvantage. This limitation occurs in analyzing the relationship between two attributes. In KSCBF algorithm, the application of K-S test is used to validate the hypothesis. Then, if occurs the redundancies of more than two attributes in a single execution, the KSCBF algorithm will reject all the redundant features in the data set and finally the algorithm will also reject the hypothesis in the redundant features. In fact, the KSCBF algorithm does not certify the full invariance to linear transformations (Biesiada et al., 2007).

Finally, the KSCBF algorithm faces big risks in handling data redundancy as mentioned in Chapter One. This affects the time required for handling the data redundancy because the algorithm is visited every attribute which is stated from initial until final attribute in the list.

2.7.5 Discussion on the Compared Algorithm

The aim of the KSCBF algorithm and the other existing algorithm such as CBFS and FCBF is to handle data redundancy effectively. The reviews on the literatures reveal that the KSCBF algorithm performs better than the other existing algorithms. Particularly, the assessment technique in KSCBF (using the K-S test) makes the algorithm strong in solving the redundant features.

Besides, KSCBF algorithm shows the way to evaluate the redundant features and the validation of the results is also available in this algorithm. In fact, the KSCBF algorithm highlights the FS method such as SU in attending the filtering and ranking task. This function can be compared to the method in the proposed algorithm in this study which is called Most Priority of Features (MPF), which handles the same cases to the KSCBF algorithm with embedded of FS approach.

On the other hand, CBFS and FCBF are unable to evaluate and handle the redundancy of features with a specific measurement technique. Finding from the literatures shows that the CBFS and FCBF directly remove the redundancy of features without any measurement and result validation. Also, the RELEIF algorithm is unable to provide an assessment technique in filtering the features and most significantly it is not attended in handling data redundant cases.

2.8 Discussion

The development of a Feature Ranking Algorithm (FRA) algorithm as a new enhancement of Pragmatic Quality Factor (PQF) model based on their assessment technique is necessary. As discussed in previous sections, the assessment technique provided by PQF model in evaluating the quality attributes is based on components such as behavioral characteristic, impact characteristic, responsibility, and weight. The measurement in the PQF model is focused on the weight calculation using a specific scale in classifying the quality attributes. The formula derived in this quality model is used to obtain the priority of quality attributes. In practice, the quality model still acts as a static model due to their limitations in handling the redundancy of data in prioritizing the quality of attributes. Thus, FRA algorithm is proposed for supporting the limitations in PQF model using AI approach as for commercializing the assessment technique embedded with an expert intelligence technique.

Literatures reveal that the FS technique has proposed a dynamic technique in the algorithm construction. Besides, the selection of FRT in enhancing the assessment technique in quality model act as the new improvement to upgrade the assessment technique in measuring the quality attributes. The FRT is relevant to be used in small sized data and performs better than other FS techniques. It has high capability to generate a function in filtering and ranking the quality attributes using filter approach. Based on that, this technique is used in in this study known as Most Priority of Features (MPF) method.

The evaluation of quality attributes is attended by two main steps. In the first step it is completed by MPF method for calculating the quality attributes. It is appropriate for filtering and ranking the priority of quality attributes. In this step, the filter approach in FS is presented. Then, in the second step the adaptation of learning classification is performed using MULAN. MULAN acts as a tool for providing multi label classifiers, including RA*k*EL and MLkNN for handling data redundancy. It is selected as a classification tool due to its high capability in determining good result of classifiers. Also, it is easy to apply in classifying the multi types of attributes such as nominal or ordinal of data type. In this step the wrapper approach is utilized to validate the performance result of data redundancy.

The results performed by FRA algorithm are validated by comparing with the expert model such as PQF and KSCBF. This could validate the proposed algorithm as a good trial in enhancement of assessment technique in the static quality model to the intelligent aspects. In accordance, Table 2.6 summarizes the selected elements in development of FRA algorithm.

Table 2.6: The Selected Elements in FRA Algorithm

Item	Selected element
Technique	Feature Ranking Technique (FRT)
Tool	Java Library for Multi Label Learning (MULAN)
Classifier	Random- *k* LabelSet (RA*k*EL) and MultiLabel – k Nearest Neighbour (MLkNN)
Expert Model	Pragmatic Quality Factor (PQF) model
Compared Model	Kolmogorov-Smirnov Correlation Based Filter algorithm (KSCBF)

2.9 Summary

This chapter revies related works in the literatures by emphasizing major contributions by common models in software quality such as McCall (McCall et al., 1976), Boehm (Boehm et al., 1978), FURPS (Grady & Caswell, 1987), ISO 9126 (ISO/IEC 9126, 1991), Software Product Quality Requirement and Evaluation (SQuaRE) (The next generation of ISO/IEC 9126, 1999), Dromey Quality Model (Dromey, 1996), Systemic Quality Model (Callaos & Callaos, 2003), and Pragmatic Quality Factor (PQF) model (Yahaya et al., 2007). The overview of software quality models covers the characteristic, assessment technique, strengths, and limitations of the static quality models in the literatures. This chapter also discusses on the components of PQF as the expert model and the compared algorithm in the literatures such as KSCBF for validating the proposed algorithm. Next, the reviews are followed by the detailed discussion on static quality model and dynamic quality model. The element of AI approach is also drawn in this chapter including the dynamic technique, FS methods, approach and learning adaptation. Finally, this chapter concludes the findings of reviews by discussing and summarizing the models. The next chapter discusses the methodology of this research.

CHAPTER THREE
RESEARCH METHODOLOGY

3.1 Research Methodology Phase

As stated earlier in the Chapter One, the research methodology has five phases with aim to enhance the assessment technique in Pragmatic Quality Factor (PQF) model developed by Yahaya et al. (2007) as an intelligent software quality model. Each of the phases is discussed in detail in the subsequent sections.

3.1.1 Theoretical Study

The theoretical study acts as the first phase in this research by investigating the literature review on the existing research related to the software quality assessment. This stage aimed to study the way on existing models in measuring the priority of software quality attributes in the software product. Furthermore, it is also focused on the existing sources as the references such as from journals, books, proceedings and other academic research in the current environment either by printed medium or electronic medium.

Hence, this phase investigates the dynamic requirements for the quality and assessment problem in the existing software product. The exploration is highlighted to the behavioral of characteristics in the existing model including their strengths and weaknesses in measures the attribute. The finding from the investigation can support the researcher to generate new ideas to be adopted, noticed and learned the changes in the environment and information needs. Furthermore, the characteristics from the software quality model is identified and analyzed in term of the significant of the attributes to support the changes in timely basis. The Artificial Intelligence (AI) approach is also reviewed as main element to be included in the assessment technique such as Feature Selection (FS), Artificial Neural Network (ANN), and Cased Based Reasoning (CBR). Thus, the proposed of Feature Selection (FS) technique such as Feature Ranking Technique (FRT) is fully highlighted in this research to enhance the assessment technique in PQF model. Figure 3.1 illustrates inputs, activities and deliverables of theoretical study in the phase one of the research.

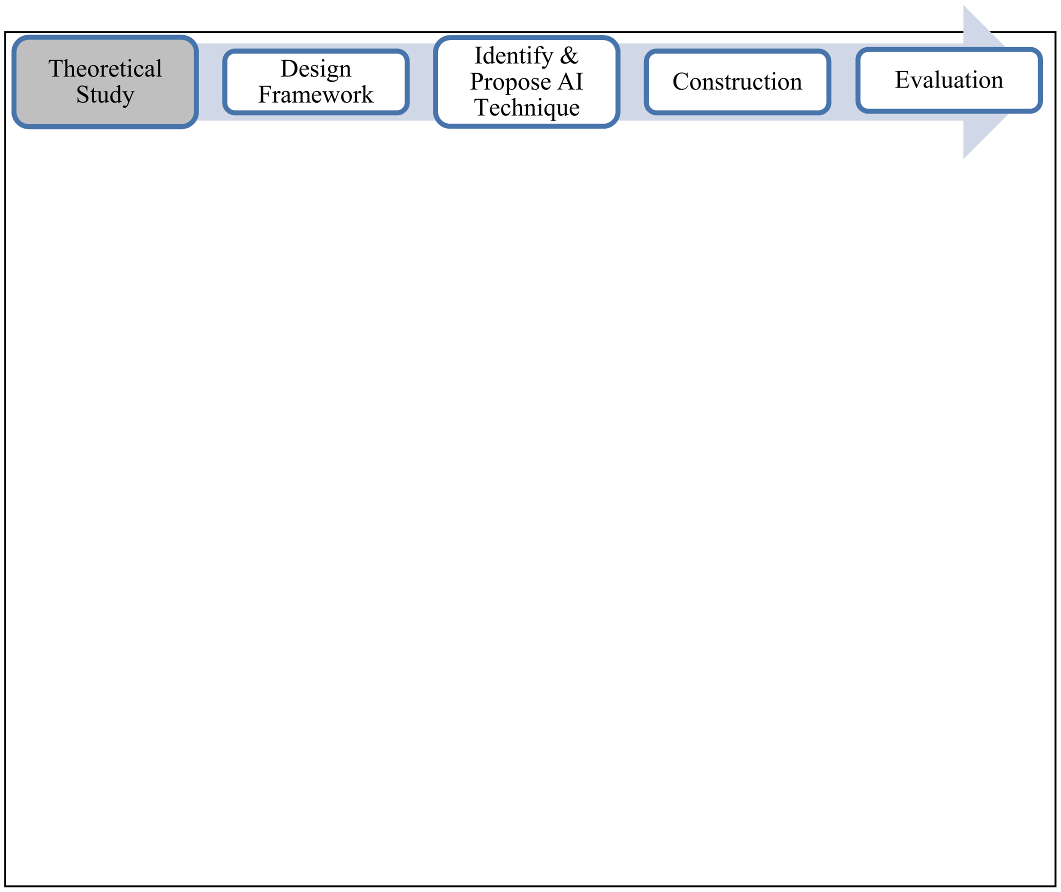


Figure 3.1 Inputs, Activities and Deliverables of Theoretical Study

3.1.2 Design of theoretical framework on intelligent software quality

The second phase of this research is designing the theoretical framework on intelligent software quality model. The theoretical framework identifies the specific features of Feature Ranking Algorithm (FRA) algorithm represented using AI approach. Furthermore, this theoretical framework can help the researcher to determine the problem areas and also consists of considerations, research questions that need to be addressed via the methodology. Hence, the researcher can illustrate the main focus of the research study and show the elements involved in the enhancement of assessment technique in PQF model. Figure 3.2 illustrates the Inputs, Activities and Deliverables of Phase two in the research and Figure 3.3 illustrates the Theoretical Framework of the research on intelligent software quality model.

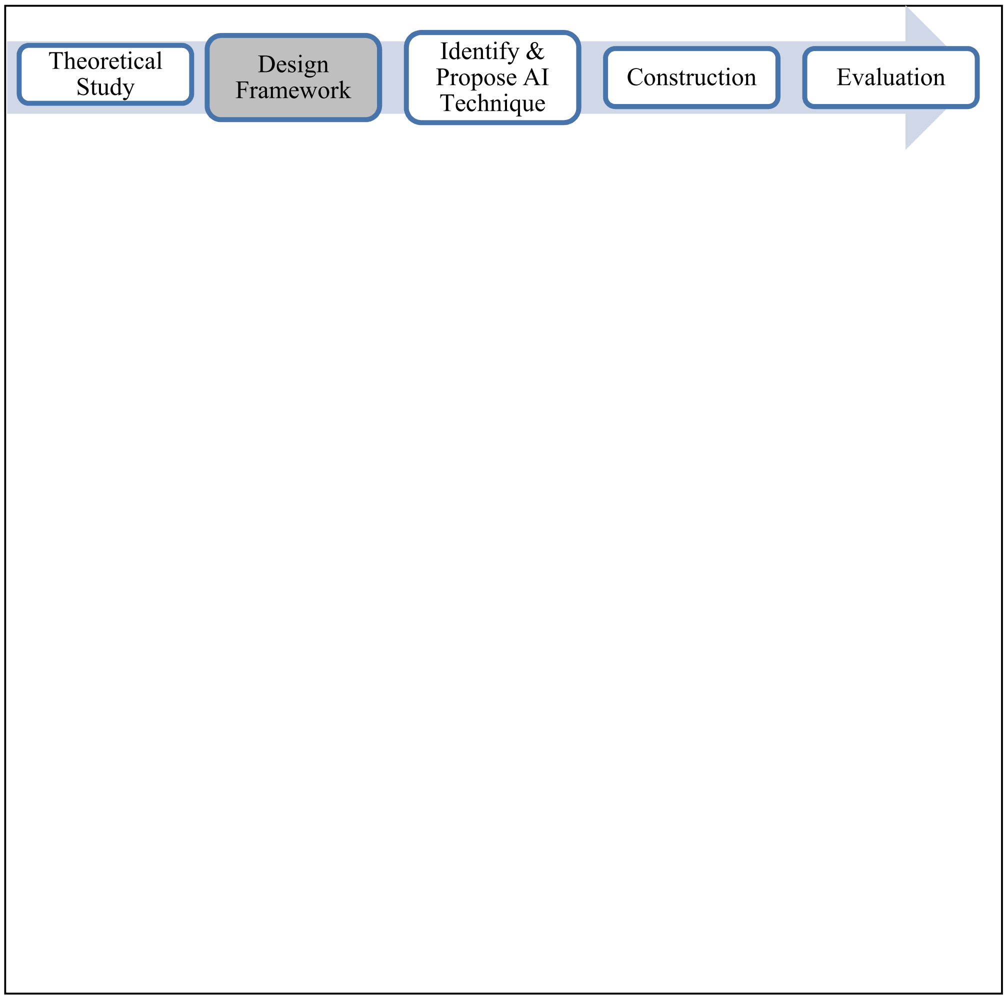


Figure 3.2: Inputs, Activities and Deliverables of Design Framework

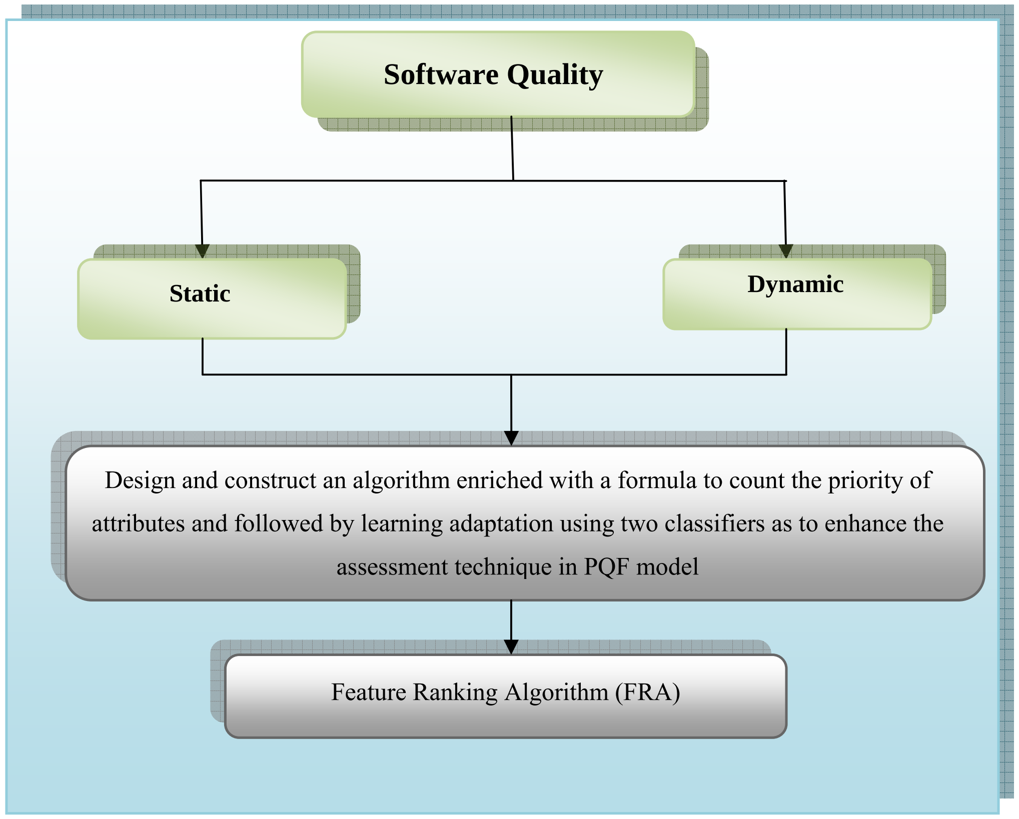

Software Quality
Static
Dynamic
Design and construct an algorithm enriched with a formula to count the priority of attributes and followed by learning adaptation using two classifiers as to enhance the assessment technique in PQF model
Feature Ranking Algorithm (FRA)

Figure 3.3: Theoretical Framework of Feature Ranking Algorithm (FRA)

3.1.3 Identify and proposes the Feature Ranking Technique (FRT) for intelligent software quality model

The third phase identifies and proposes the Feature Ranking Technique (FRT) for intelligent software quality model. The new intelligent algorithm is known as Feature Ranking Algorithm (FRA) model which is an enhancement of Pragmatic Quality Factor (PQF) model. From the literature review, several AI techniques have been found to enhance the assessment technique in software quality model such as Feature Selection (FS), Artificial Neural Network (ANN) and Cased-Based Reasoning (CBR). All of the techniques are investigated and discussed in Chapter Two.

In order to create an algorithm with the expert function as to measure and evaluate the attributes in software quality, the FRT as a type of FS is selected in this study. The proposed technique has been reviewed pertaining to the application in pre-processing of the data in order to sort and list the quality attributes. Furthermore, the designing of FRT involved both approaches in the feature ranking such as filter and wrapper approach which discusses in Chapter Two. Thus, the adaptation of learning concept is performed using the Java Library for Multi Label Learning (MULAN) application using classifiers such as Random *k*-Labelsets (RA*k*el) and Multi Label *k*-Nearest Neighbour (ML*k*NN). Figure 3.4 illustrates the Inputs, Activities and Deliverables of Identify and proposes the FRT for intelligent software quality model.

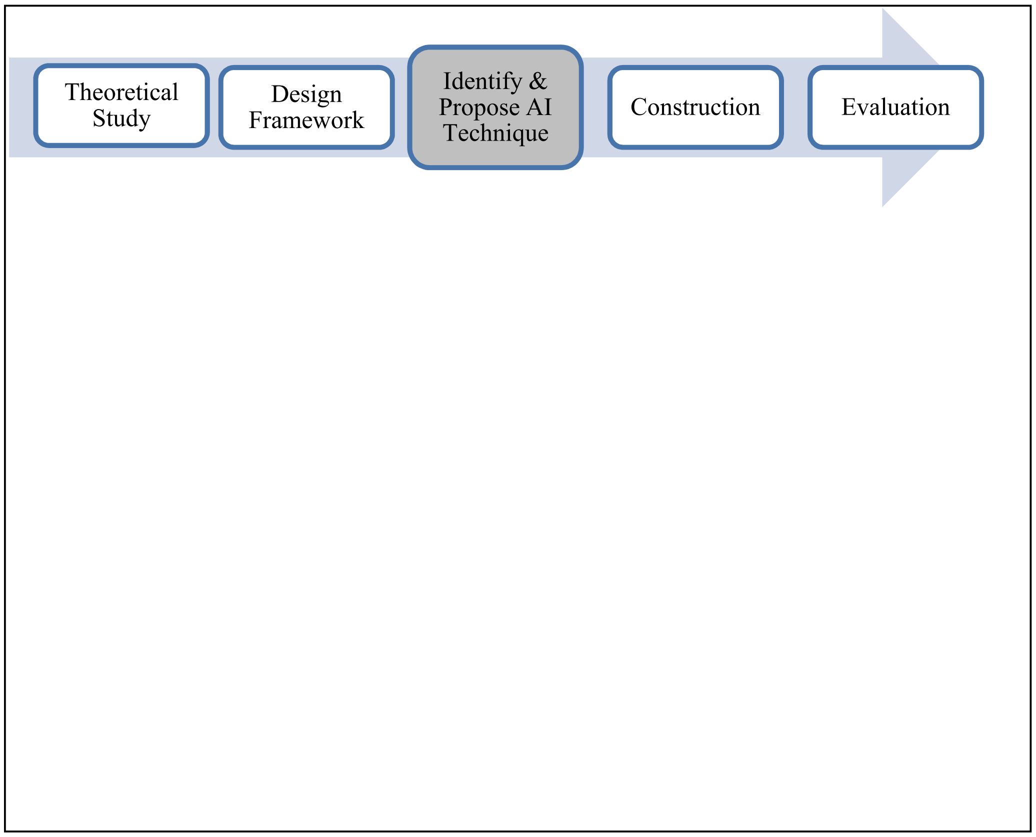


Figure 3.4: Inputs, Activities and Deliverables of Phase Three

3.1.4 Construction of an Feature Ranking Algorithm (FRA) Algorithm

The fourth phase of the research is construction of a Feature Ranking Algorithm (FRA) algorithm using FRT. In this selected technique involved both approaches in FS such as filter and wrapper approach. In construction phase, the algorithm enclosed by the formula known as Most Priority of Features (MPF) to count the scoring weight value given by the stakeholders for all attributes. The behavioural attributes such as Efficiency, Maintainability, Functionality, Portability, Reliability, Usability, User Conformity and Integrity were gathered from the research studies captured from the previous project certification ended by Yahaya from year 2007 until 2011. The filter approach is used to create MPF formula as to rank and sort the counted attributes accordingly and stored into PQF database respectively once the results performed in different value for all attributes. The elements involved in the formula are the standards deviation, weight assigned by the assessors and the frequency of the maximum weight assigned for all attributes.

The next stage is the learning concept in the algorithm through the Java Library for Multi Label Learning (MULAN) as a tool by importing two classifiers namely Random *k*-Labelsets (RA*k*el) and Multi Label *k*-Nearest Neighbour (ML*k*NN). According to Wang et al. (2011), the selection of classifier may affect the classification accuracy. As stated earlier, this research has applied two classifiers in handling the redundancy of the data to avoid biasness. The implementation of the wrapper approach is capable to train and test the data for classification task using the score counted by MPF formula. The performance metric such as the Area Under the Curve (AUC) is used to calculate the classification accuracy. The detail of the construction is entirely discussed in Chapter Four. Figure 3.5 illustrates the Inputs, Activities and Deliverables of Construction of Feature Ranking Algorithm (FRA) and Figure 3.6 illustrates the steps includes in the experimental design of this phase.

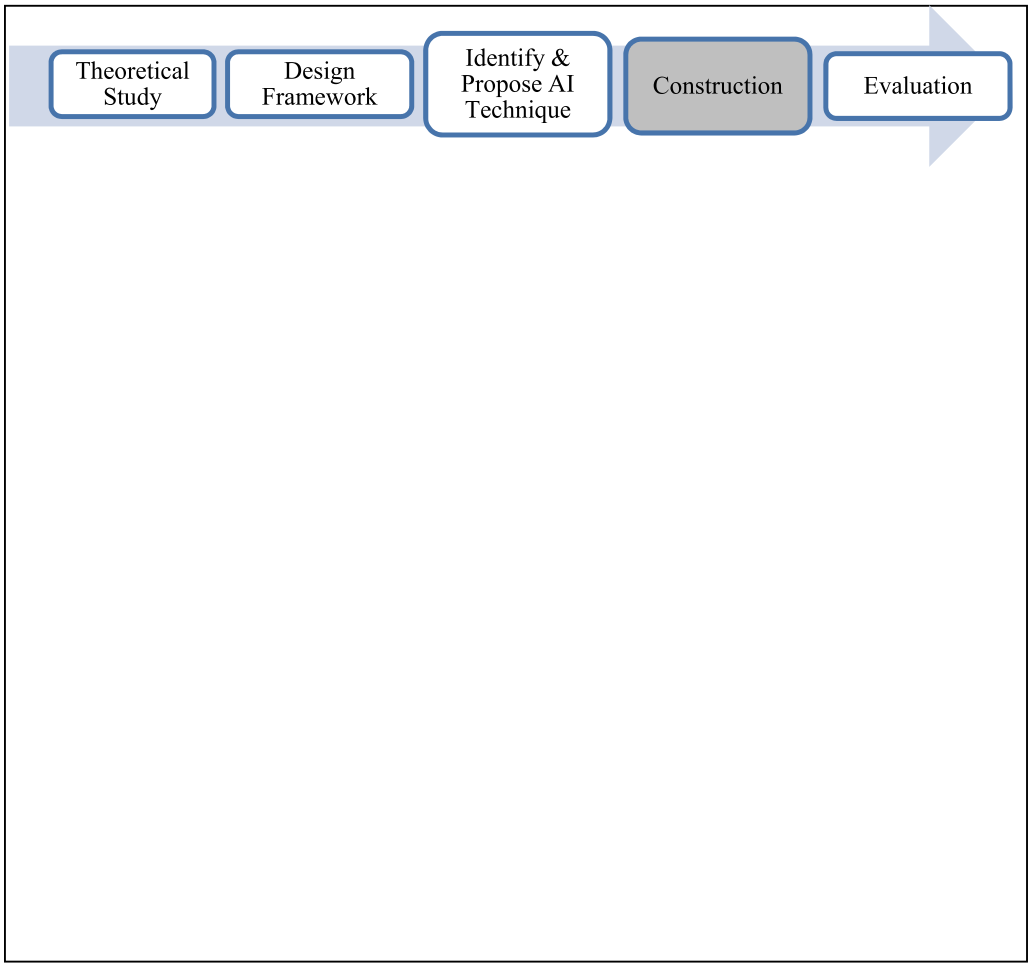


Figure 3.5: Inputs, Activities and Deliverables of Construction of Feature Ranking Algorithm (FRA)

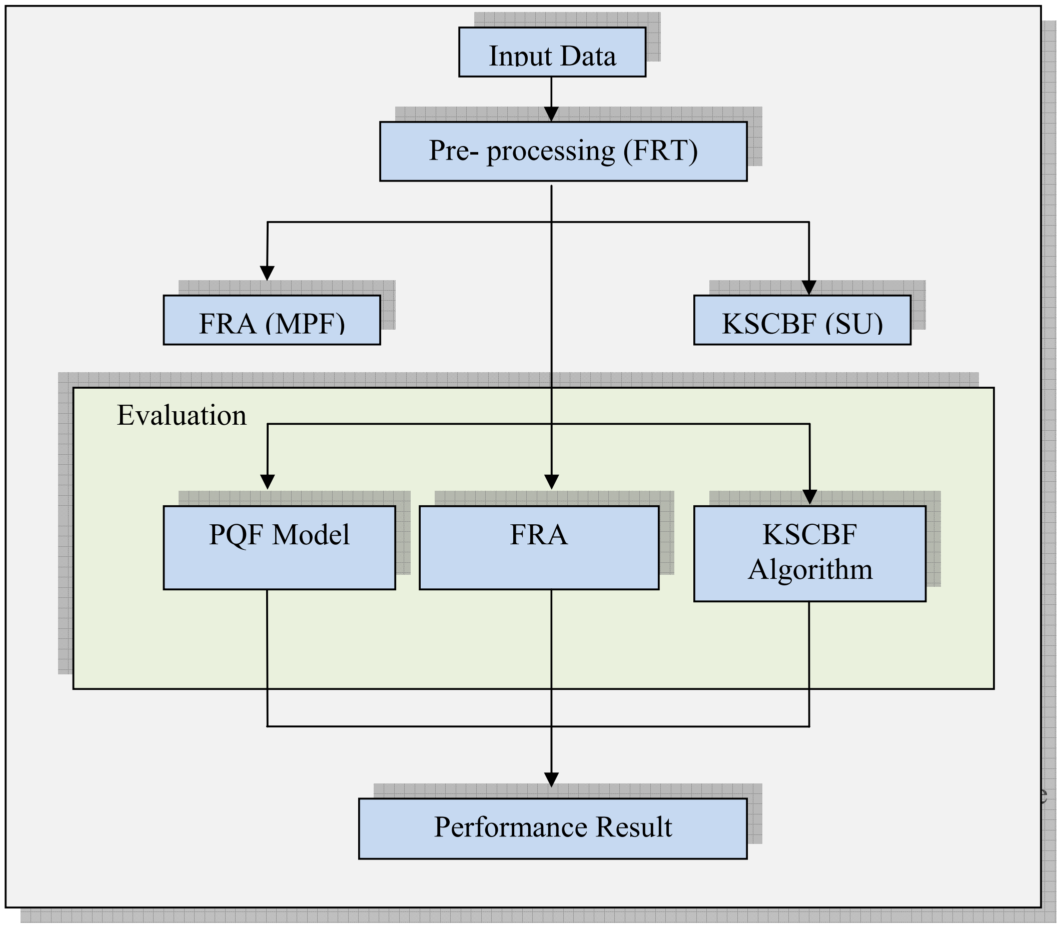


Figure 3.6: Experimental Design

The experimental design was conducted using a data set gathered from the previous cases of certification in PQF model. It contains over 1000 cases of software quality assessment data. The data is implemented by the algorithm which constructed using FRT as pre-processing process. The data is weighted by the assessors in the specific value were made on a five point Likert scale (1 = strongly disagree to 5 = strongly agree). The assessors consisted of the user or individual, developer, and manager who are concerned to apply this model for attributes assessment.

The data collection is processed using MPF method in FRA algorithm and the comparison is also tested for the compared algorithm such as KSCBF algorithm with using their own method in pre-processing such data as SU. Both of the methods are executed in the same time in order to gain the accuracy of the methods. As for comparison, the MPF method is performed better than SU in ranking the priority of data.

As mentioned in the previous section, the proposed algorithm will generate the results for each attribute and acts as an alternative solution to solve the redundancy cases occurred in the databases. In fact, the final ranking result for each attribute is performed as to show the priority of quality attributes.

Hence, the evaluation is verified the proposed algorithm by the human judgement and statistical measurement on the final ranking result to the expert model and the existing algorithm in the literature such as Kolmogorov-Smirnov Correlation Based Filter (KSCBF) algorithm.

3.1.5 Evaluation of Study

The fifth phase of the research is the evaluation measurement of the proposed algorithm using the human judgement and statistical measurement methods such as correlation coefficient and statistical significant test. As stated earlier, the human judgement method is the comparison on the final ranking result of proposed algorithm to the expert model known as PQF model and the existing algorithm in the literature such as KSCBF. The correlation coefficient is extremely important as to show the relationship between the proposed algorithm to the PQF model and KSCBF algorithm.

Hence, the statistical significant test is used to test and evaluate the differences in scores of the results obtained by matching pairs of the expert model and the compared algorithm. Moreover, this phase is extremely important to prove that the new assessment technique of intelligence software quality model is achieved by using FRT. Figure 3.7 illustrates the fifth phases of the research.

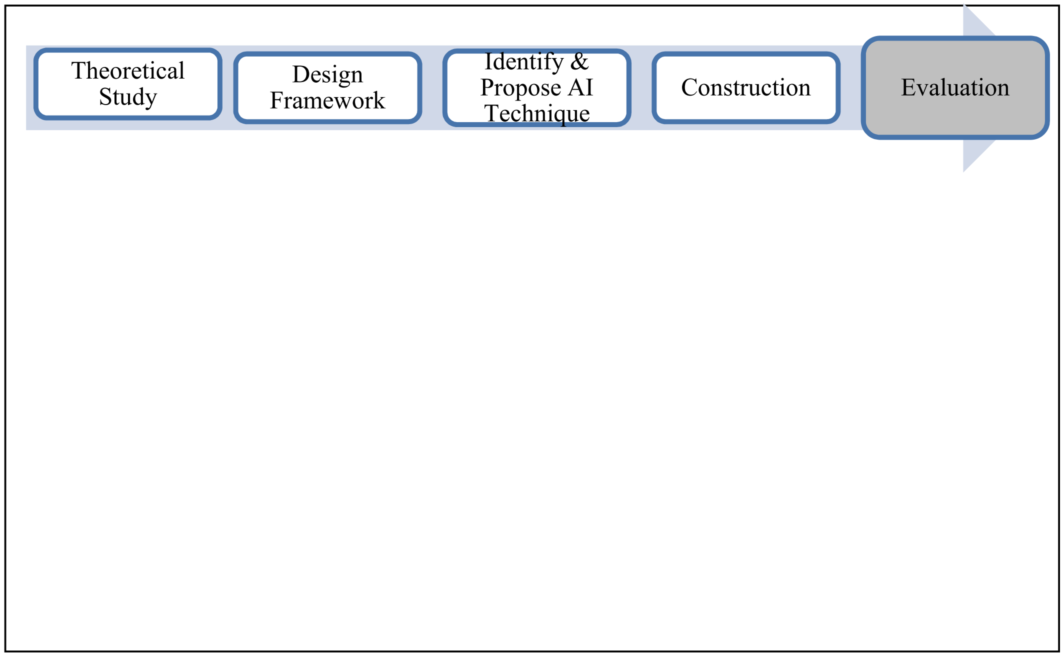


Figure 3.7: Inputs, Activities and Deliverables of Evaluation

3.2 Summary of Research Methodology

Five phases applied in this research as the process flow in developing the intelligent of software quality model which are: Theoretical Study, Design of Formal Framework on intelligence software quality, Identify and proposed the Feature Ranking Technique (FRT) for an intelligence software quality model, Construction of a Feature Ranking Algorithm (FRA) and the Evaluation of study. Each phase has key inputs, activities and deliverables to achieve the research objectives and solve the research problem in order to develop a new intelligent software quality model for attributes assessment.

CHAPTER FOUR
PROPOSED FEATURE RANKING ALGORITHM (FRA) ALGORITHM

4.1 Introduction

This chapter describes the proposed Feature Ranking Algorithm (FRA) algorithm using Feature Ranking Technique (FRT). Previous chapters addressed that the development of FRA algorithm acts as a new assessment technique on quality attributes to enhance the assessment technique in the existing model known as PQF model. The enhancement of PQF model in software quality assessment indirectly appears as a dynamic model in software quality community. The major components in FRA algorithm include generation of MPF formula and implementation of learning application. Both components appear as additional features of assessment technique in PQF model.

Further, following sections elaborate about the background issues of this study, the proposed algorithm itself and the way it is implemented.

4.2 Background Issues

Quality can mean different things to different people and situations (ISO/IEC9126, 1991). The development of PQF model incorporates behavioral and human aspects making it a bit different from other models. Even though, the criteria of assessment in PQF model can fulfill user requirement and expectation in future, it still has a limitation in adapting and noticing changes in attribute selection that might occur in future. As discussed in Chapter Two, the assessment technique used in tailoring the engine of PQF model is the elements of software engineering development and it is not capable to train and learn the knowledge in data. Thus, the proposed FRA algorithm in the assessment technique of attribute selection can enhance the limitation of PQF model in noticing and adapting the knowledge of data in future. In accordance, the list below contains issues solved by FRA algorithm;

1) The assessment technique is supported using AI approach such as FRT.
2) Data redundancy is solved using the classification task through MULAN.

4.3 Proposed FRA Algorithm

The development of FRA algorithm is based on the component of PQF model including behavioral characteristic, impact characteristics, responsibility, and weight value. These components are discussed details in Chapter Two. Also, AI approach is used as an additional element to generate a dynamic quality model in software quality. The main features of FRA algorithm are summarized as follows:

1) MPF formula as to count the scores of the priority attributes from the software quality data. This helps in ranking the software quality attributes according to the most prominent attribute.
2) Adaptation of learning concept through MULAN as to train the knowledge of data and handling data redundancy. These features can support the limitations of PQF model which are discussed in Chapter Two.

The FRA algorithm facilitates the interaction between user and the system by enabling the user to select attributes based on the collection of results in software quality attributes. The user can be a group of assessment team to ensure unbiased assessment and have capability to give weight value for each attribute in the assessment. The PQF database contains all possible software quality factors or attributes collected from the previous quality model in literatures including Functionality, Maintainability, Efficiency, Portability, Reliability, Usability, User Conformity, and Integrity. The PQF database is known as knowledge based, in which the collection of the attributes are trained and learned from the literature. Furthermore, the PQF model contains the selection of the attributes collected through research documents and experiences from developer. Eventually, the attributes in PQF model are trained and learned by FRA algorithm with the proposed methods involved.

In detail, the proposed method consists of two phases. Phase one calculates the MPF scores for all attributes from the database. The results of attributes are ranked and sorted according to the scores obtained. In this exercise, the highest score value of attributes is selected as the important attribute and directly selected as high priority of attributes and updated to the database. Consequently, Phase two remove redundant data that contains more than one highest value of the scores. This means that Phase two is executed if there are two or more redundant attributes that produce similar MPF scores.

In addition, the data from PQF database corresponding to the attributes which are redundant are obtained to be used for training the classifiers. The results are performed by classifiers as discussed in previous chapter is averaged to avoid biasness. Then, the attributes that produces the highest classification accuracy is then selected as the most priority of features and ranked first in the list. Consequently, the final ranking of the quality attributes are stored in the PQF database for future software quality assessment. In conjuction, a diagram showing assessment technique in FRA algorithm is shown in Figure 4.1. It is followed by the steps FRA algorithm in the next sub section.

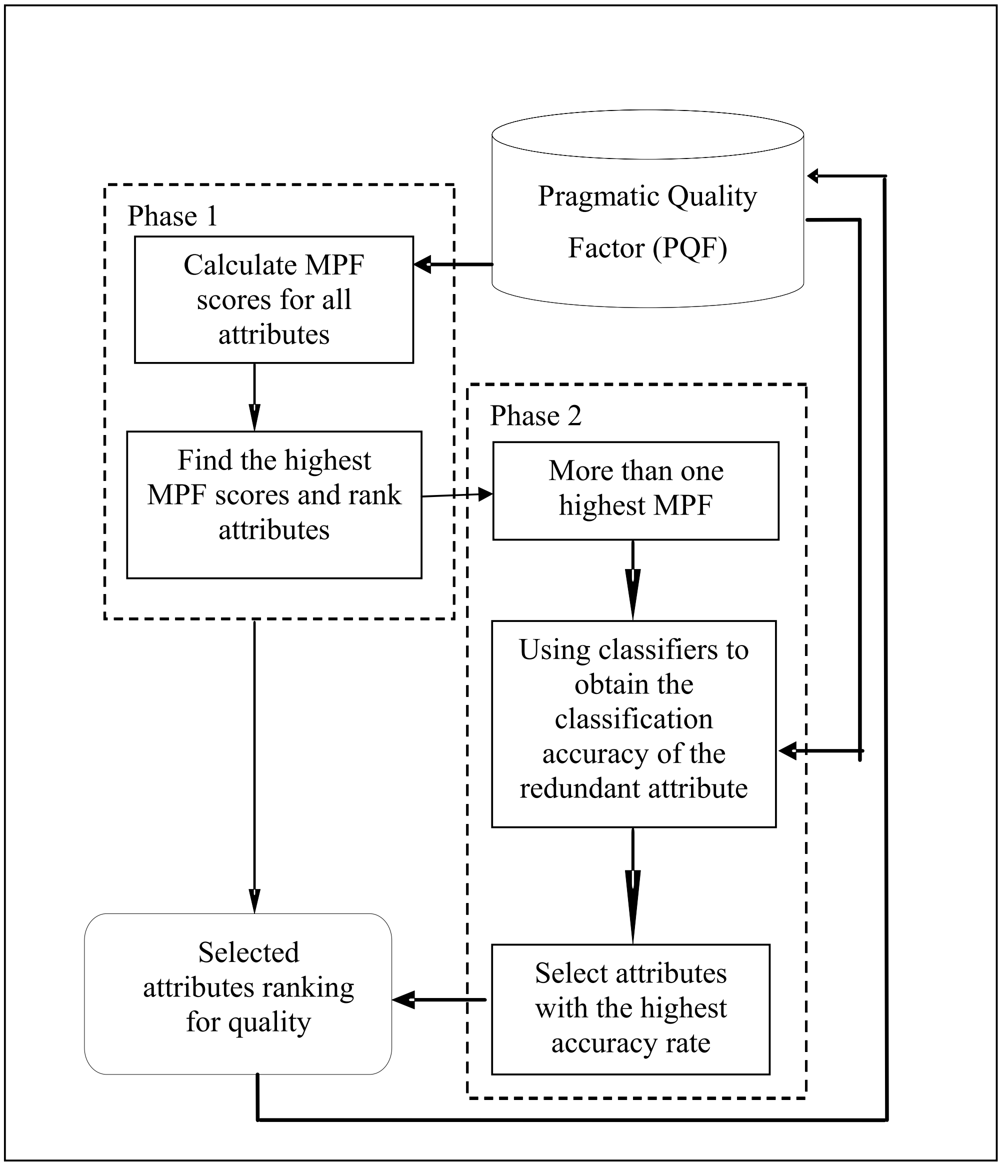

Pragmatic Quality Factor (PQF)
Phase 1
Calculate MPF scores for all attributes
Find the highest MPF scores and rank attributes
Phase 2
More than one highest MPF
Using classifiers to obtain the classification accuracy of the redundant attribute
Select attributes with the highest accuracy rate
Selected attributes ranking for quality

Figure 4.1: Assessment Technique in FRA Algorithm

4.3.1 The Development of Most Priority of Features (MPF) Formula

The main element involved in generating the MPF formula consisted in FRA algorithm is FRT by using filter approach. As discussed in Chapter Two, FRT is a part of feature selection which is the pre-processing data in reducing the high dimensionality of data (Khoshgoftaar et al., 2009). The FS is an important technique to speed up the learning process and is capable to improve the assessment technique in the software quality model. In the literatures, FS is widely used in assessing the performance of the classification models by using performance metrics such as Overall Accuracy (OA), Default F-Measure (DFM), and Default Geometric Mean (DGM) (Khoshgoftaar et al., 2009). On top of that, detailed descriptions about the FS in classification model are outlined in Chapter Two.

Meanwhile, the filter approach selects data as a pre-processing independently without involving any learning application (Guyon et al., 2003). It provides a task to rank and sort the selected attributes based on the MPF scores. Besides, other elements involved in the MPF formula include probability concept, standard deviation, mod frequency, and arithmetic mean. In detail, the probability concept is the likelihood of something happening in future which is expressed as a number between zero (0) and one (1). Further, the expressed number refers to something that can never happen and something that will always happen (Durret, 2010). This concept is used to make expectations on the priority of each attribute selected by the users. On a contrary, standard deviation is a formula for the average distances from the average, which refers to the dispersion of a set of data from its mean. It is computed by the mean for the data set and the deviation by subtracting the mean from each value. It also known as square root of the varians for their mean arithmetic in the data set. The generation of the formula is discussed further in the next sub-section.

The formula in FRA algorithm is capable to train the value of attributes and compare the result for each attribute as to find the MPF scores of the attributes. In accordance, the formula is exhibited in Equation 4.1.

$$\text{Most Priority of Feature (MPF)} = \delta_{yj} \sum (\max_{xi} \bullet f\max_{xi}) \qquad \text{(Eq. 4.1)}$$

As noted earlier, the formula created in FRA algorithm is a solution to remove irrelevant features in a list of data. Therefore, the formula has to train the data for each attribute in knowledge based. In regards to this, there are steps in generating the MPF formula as to count the priority of attributes, which is selected by the software quality assessors.Particularly, there are four steps involved and are explained in the following paragraph.

Step 1: The arithmetic mean of variables is calculated using the formula in Equation 4.2, which is adopted from book Introductory Statistics (2008).

$$\text{Mean of variable: } \mu = \frac{\sum xi}{N} \qquad \text{(Eq.4.2)}$$

Where, μ is the population mean, $\sum xi$ is the total of data collection in the database and N is the population size in the database. The mean of variable is obtained by calculating the mean value of each attribute and ranks the mean values subjected to the priority of the attributes.

Step 2: The next step is calculated the standard deviation of the attributes in the database. It is accomplished using the formula in Equation 4.3, which is adopted from book Introductory Statistics (2008).

$$\delta yj = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(xi - \mu)\,2} \qquad \text{(Eq.4.3)}$$

Where, δyj is the standard deviation for the selected quality attributes, N is the population size in the attribute's database, i is the value of vector in standard deviation value (standard value) and $(xi - \mu)2$ is the attribute's value in the database and deficiency to the mean value of the data collection in the attribute's database.

The correspondences of two random attributes are evaluated using the formula in Equation 4.3 to measure the scattered values in the collection value of attributes. In this case, the maximum value of the selected attributes in the database ($\max xi$) is used to be multiplied by the mod frequency of maximum value of the selected attribute in the same database ($f\max xi$).

Step 3: The value of the previous application is used to find the MPF by multiplying with the value of standard deviation of attribute's database.

Thus, step 2 is repeated to the other values of mean population of each attribute. Then, the MPF value is compared to the value of each attribute as to select the high priority of selected attributes.

Step 4: Finally, the formula in Equation 4.4 is created to sort and rank the features in the attribute listing.

$$\sum (\max xi \bullet f\max xi) \qquad \text{(Eq. 4.4)}$$

Where, maxxi is referred to the maximum of value in the database and $f$max$xi$ is referred to the mod of frequency for maximum value in the database. Having outlined the steps involved, the following section demonstrates an example of MPF calculation.

4.3.2 The Example of MPF calculation

The Efficiency attribute is selected for demonstrating the calculation. It involves data collected from assessors as shown in Table 4.1.

Table 4.1: Efficiency of Database

User	Value (*xi*)
User 1	4
User 2	5
User 3	3
User 4	5

Step 1: The assumption values contain four cases, which are {4,5,3,5}. This makes the mean quality score is 4.25 and standard deviation is 0.96. The maximum value ($\max_{xi}$) is 5 and the frequency of the maximum value in the efficiency database is 2. Therefore, the MPF score using the formula in Equation 4.1 is 0.96 (5*2) = 9.6. Then, the step of calculation is repeated to the other attributes in the database.

Step 2: The MPF scores of the data are checked and sorted according to scores obtained from the calculation. Then, in the sequential order, the small values are treated weak and ready to be sorted lowest in the database as irrelevant features.

As been discussed earlier, it has been noted that if the results contain redundancy, then the classifier is used as the solution to solve the redundancy problem using the proposed learning tool. In accordance, it is discussed in the next section.

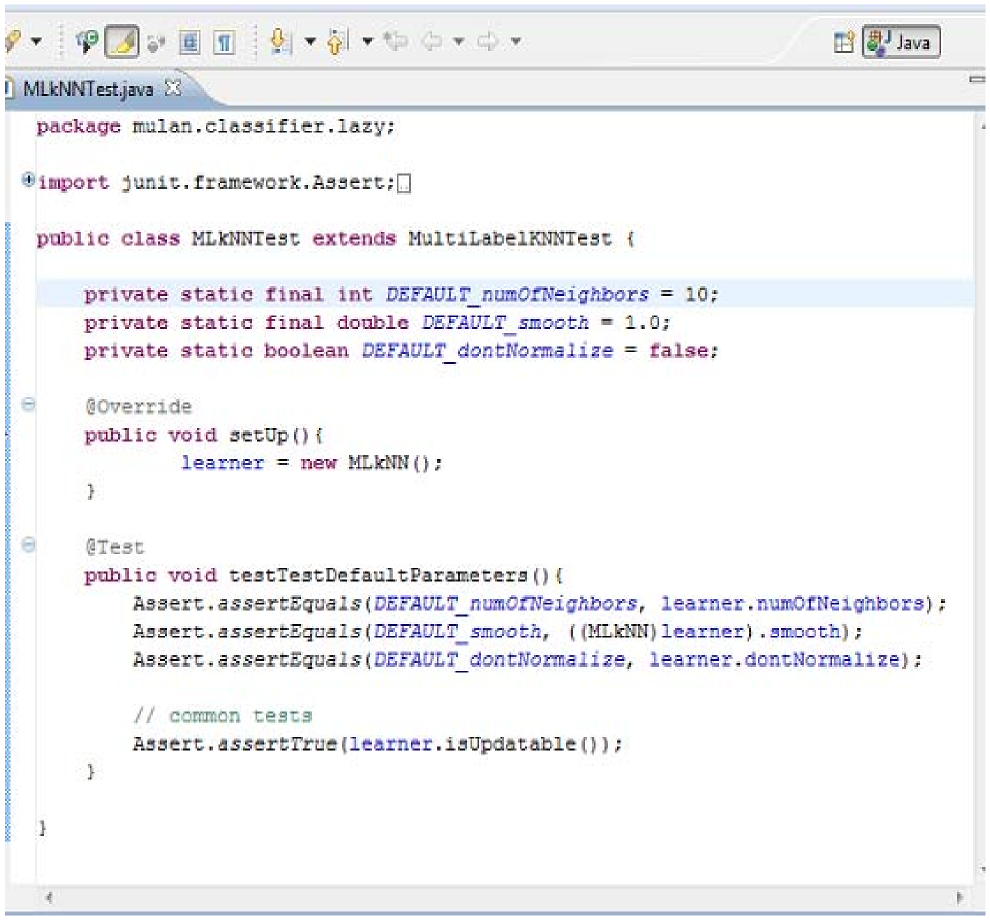
4.3.3 The Application of Classifiers

Chapter Two notes that the application of classifier can be found in the learning tools such as WEKA, MULAN, and LEET. In this study, the learning adaptation is performed using MULAN which is known as a Java library for learning from multi label data. It offers a variety of classification, ranking, thresholding and dimensionality reduction algorithm for learning from hierarchically structured labels (Tsoumakas et al., 2011) (as discussed in Chapter Two). However, it only offers the Application Programming Interface (API) to the library users without GUI.

The MULAN application can be started by downloading WEKA version 3.7.6, Java Runtime Environment (JRE) version 30 from Oracle[1] website. The next step is to execute the unit tests to import the classifiers that can exist in MULAN application. Referring to the literatures, only several classifiers can be adapted in MULAN and it depends on the capability of the classifiers to act in the MULAN environment as elaboratively described in Chapter Two.

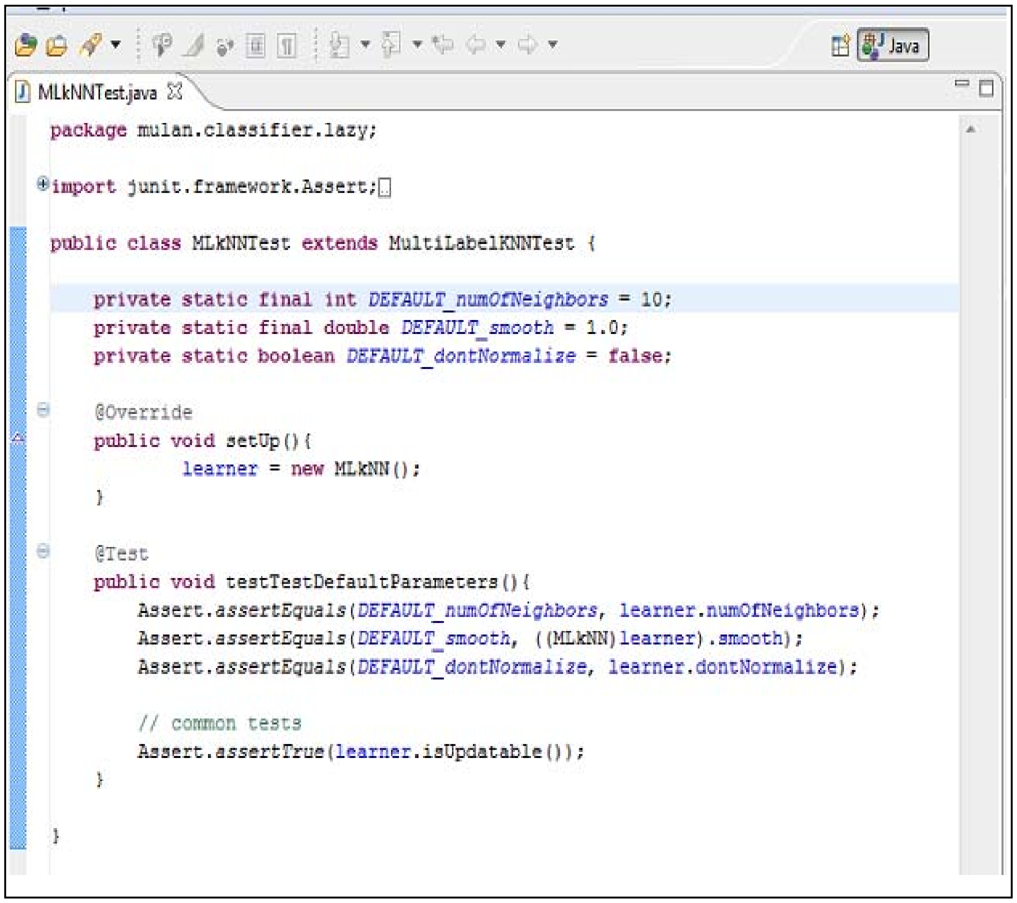
[1] http://www.oracle.com/technetwork/java/javase/downloads/jre-6u30-download-1377142.html.

In order to select the classifiers to be allocated in MULAN, the execution of unit tests is applied and is extremely important to examine the availability of the classifiers and performed in the MULAN. Thus, the selected classifiers such as RA*k*el and ML*k*NN should be accepted in execution process. In regards to this, Figure 4.2 shows the execution process of unit tests in Eclipse platform. It is followed by the execution of unit tests for MLkNN classifier in Figure 4.3.



```
package mulan.classifier.lazy;

import junit.framework.Assert;

public class MLkNNTest extends MultiLabelKNNTest {

    private static final int DEFAULT_numOfNeighbors = 10;
    private static final double DEFAULT_smooth = 1.0;
    private static boolean DEFAULT_dontNormalize = false;

    @Override
    public void setUp(){
            learner = new MLkNN();
    }

    @Test
    public void testTestDefaultParameters(){
        Assert.assertEquals(DEFAULT_numOfNeighbors, learner.numOfNeighbors);
        Assert.assertEquals(DEFAULT_smooth, ((MLkNN)learner).smooth);
        Assert.assertEquals(DEFAULT_dontNormalize, learner.dontNormalize);

        // common tests
        Assert.assertTrue(learner.isUpdatable());
    }

}
```

Figure 4.2: Execution Process of Unit Tests for MLkNN Classifier



```
package mulan.classifier.lazy;

import junit.framework.Assert;

public class MLkNNTest extends MultiLabelKNNTest {

    private static final int DEFAULT_numOfNeighbors = 10;
    private static final double DEFAULT_smooth = 1.0;
    private static boolean DEFAULT_dontNormalize = false;

    @Override
    public void setUp(){
            learner = new MLkNN();
    }

    @Test
    public void testTestDefaultParameters(){
        Assert.assertEquals(DEFAULT_numOfNeighbors, learner.numOfNeighbors);
        Assert.assertEquals(DEFAULT_smooth, ((MLkNN)learner).smooth);
        Assert.assertEquals(DEFAULT_dontNormalize, learner.dontNormalize);

        // common tests
        Assert.assertTrue(learner.isUpdatable());
    }

}
```

Figure 4.3: Executed of Unit Tests for MLkNN Classifier

In this test, the classifiers (MLkNN and RAkEL) are certified by executing the unit test and ready to be used in MULAN. Both of the classifiers are accepted in all tests. As an example, Figure 4.3 shows the executed of unit test for MLkNN classifier. In this unit testing, the classifiers are tested according to the JUnit testing in meeting the required formats such as testing for default parameter, technical information, making copy test, building null dataset, and missing values. All the tests are available in WEKA and adaptation into MULAN is great as WEKA and it is beneficial to the users as they can choose several classifiers as needed.

Chapter Two elaborates that MULAN is only suitable for small data. Thus, the MULAN is unable to execute on typical programming language platform in the environment with large data. Hence, in this study, the Spring Source Tool Suite (STS) is used. It is a development tool available in Wide World Web (WWW), the most advanced tool for all the latest enterprise Java based technologies. It is available in Unix version and windows version. In this study, the window version is used which is downloaded from STS[2].

In MULAN, the set of data are kept in the required two files which are Attribute Relation File Format (ARFF) and Extensible Markup Language (XML) formats. Then, the data are loaded up using the function below:

```
MultiLabelInstances dataset = new MultiLabelInstances (arffIPQF, xmlIPQF);
```

These format also required by WEKA as to train and learn the data in the required format. Hence, the creation of an instance from each classifier is very important to evaluate the learning result. In regards to this, the following codes are utilized:

```
RAkEL learner1 = new RAkEL (new LabelPowerset (new J48()));

MLkNN learner2 = new MlKNN();
```

Next, these two imported classifiers by MULAN are trained and learned on the redundancies of data to evaluate their predictive capability in determining the final

[2] http://download.springsource.com/release/STS/2.8.1/dist/e3.7/springsource-tool-suite-2.8.1.RELEASE-e3.7.1-win32-installer.exe

result. Eventually, the final results are ranked and sorted subjected to the highest value among the redundancies of data.

Nevertheless, the wrapper approach is attached indirectly in this task for handling the redundancies cases among the data. As discussed in Chapter Two, the wrapper approach selects relevant attributes based on the performances of the selected classifiers (Tadeuchi et al., 2007). In conjunction, the assessment methods involved are elaborated in the next sub sections.

4.3.4 The Step of FRA Algorithm

The FRA algorithm acts as an intelligent assessment technique in software quality model. Besides, the proposed technique creates a dynamic software quality model in attributes assessment as well. The steps involved in FRA algorithm are outlined in Table 4.2.

Table 4.2: Algorithm of FRA

Step	Algorithm
1	Get the software quality attributes and weights from the PQF database
2	Use weight value to calculate the MPF scores for all attributes
3	Sort and rank the attributes according to the highest MPF scores
4	If there are more than one highest MPF score For each of the attribute with same MPF score Begin a. Get the corresponding data and weights from the PQF database b. Input the data into two classifiers c. Calculate the average classification accuracy of the two classifiers d. Output the average classification accuracy End
5	Select the attribute with the highest classification accuracy
6	Output the ranked software quality attributes

The FRA algorithm consists of two steps. In the first step, the quality attributes achieved from PQF database is calculated using MPF formula. The MPF functions as a method to sort and rank the quality attributes according to the counted scores in descending order using the outlined in Section 4.3.1. Then, the highest score value is selected as the prominent attribute and is directly updated in the database. In case, if there are more than one highest MPF scores (redundant) the algorithm proceeds to use classifier in classification task. Then, the second step commences.

The performance of classifiers in handling the redundancy of data is established and the output is averaged in order to avoid biasness. Consequently, the results are tested for accuracy and validated. Then, the algorithm selects the attribute with the highest classification accuracy. Eventually, the final ranking result is updated in the database.

4.4 The Development of Kolmogorov-Smirnov Correlation Based Filter (KSCBF) Algorithm

Having reviewed the existing algorithms in the literatures, the Kolmogorov-Smirnov Correlation Based Filter (KSCBF) algorithm has been decided to be used for comparison (discussed in Chapter Two). The KSCBF algorithm has been implemented on Java following the steps provided in the algorithm as outlined in Table 4.3. The following section discusses the development process.

Table 4.3: Kolmogorov-Smirnov Correlation Based Filter (KSCBF)

Step	Algorithm
	Relevance analysis
1	Calculate the *SU*(*X,C*) relevance indices and create an ordered list *S* of features according to the decreasing value of their relevance.
	Redundancy analysis
2	Take as the feature *X* the first feature from the *S* list
3	Find and remove all features for which *X* is approximately equivalent according to the K-S test
4	Set the next remaining feature in the list as *X* and repeat step 3 for all features that follow it in the *S* list.

KSCBF algorithm consists of two stages. In the first stage, the relevance analysis, in which the algorithm is ranked all attributes uses a traditional method in FS such as SU for the ranking coefficients. The first step is ranked the attributes in descending order. In regards to this, the algorithm is discussed elaboratively in Chapter Two, in which SU is computed using formula in Equaltion 4.5.

$$\mathrm{SU}\,(X,C) = 2\left(\frac{\mathrm{MI}\,(X,C)}{H\,(X) + H\,(C)}\right) \qquad \text{(Eq. 4.5)}$$

Where, X is the selected features in the class attribute, C is class of the selected attribute, MI is the Mutual Information which is the basic quantity used for filtering method, H is uncertainty probabilities, H(X) is uncertainty of X features and H(C) is the uncertainty class of the selected attribute.

In short, the presented formula is used to count the quality attributes to be sorted and ranked according to the quality score value. Further, the second stage is to remove the redundancies of data in the list using K-S test. This is accomplished using formula in Equation 4.6.

$$KS\,(g;\,h) = \sqrt{\frac{ngnh}{ng + nh}}\ \sup_{k} |Gk - Hk| \qquad \text{(Eq. 4.6)}$$

Where, *ng* and *nh* are the number of samples for each random variables, *k* is the number of bins in discrete probability distribution, G and H are cumulative probability distribution of random variables *g* and *h* respectively.

According to the algorithm, the random variables are counted in pair. Two of features are counted and the highest differences between the cumulative distributions from the features are selected. The second phase is repeated for the others attributes ranking in the list. Every feature in the list is counted using K-S test to remove the redundancy. Then, the values of the features are sorted and ranked in the new list after K-S test calculation. Eventually, the algorithm will remove all redundant features in the new attributes ranking.

However, for the purpose of showing the priority of quality attributes in this study, the redundant features are remained in the list for easier comparison.

4.5 Summary

This chapter describes the development of FRA algorithm in detail. The main issues to be solved by the proposed model are clearly defined followed by a discussion on the elements involved with the main features of the proposed algorithm as to attend the claimed issues. This includes an explanation on the elements used in generating the FRA algorithm and MPF formula as to count the score of attributes and followed by sorting and ranking the results. Besides, this chapter also describes the learning adaptation in handling the redundancies through MULAN using two classifiers (RA*k*el and ML*k*NN). In this regards, the capability of the classifiers has been determined by their predictive competency in the final result. On top of that, this chapter also discusses on the method of FRA algorithm and the steps involved. Particularly, the development process involved for the proposed algorithm is outlined. Also, the development of KSCBF algorithm as comparison method is addressed. Eventually, this chapter exhibits and discusses the results obtained from both models detail. As a consequent, the explanations on the analysis of the performance are discussed in detail in the next chapter.

CHAPTER FIVE
DATA ANALYSIS AND RESULT

5.1 Introduction

This chapter elaborates the data analysis and results of FRA for a dynamic software quality model. The assessment technique has been developed using FRT to enhance the assessment technique in static quality model namely PQF model. All the data were organized, analyzed, and interpreted systematically in an attempt to answer the question of how the FRT can be used to develop an intelligent assessment technique which is known as a dynamic model in software quality. Various statistical techniques were used for the analysis including correlation coefficient statistic and the statistical significance test.

Section 5.2 presents the result of quality attributes, which is described according to the score obtained from FRA and KSCBF algorithms. Furthermore, the results obtained from both methods are compared with expert judgment. Next, Section 5.3 analyzes the performance evaluation for this study. Finally, a summary of the findings is outlined in Section 5.4 at the end of the chapter.

5.2 Performance Results

The performance results obtained from FRA algorithm are compared to the existing algorithm in the literature (KSCBF algorithm) and the expert judgment i.e. PQF model. The results are analyzed respectively in subsequent sections.

5.2.1 Result of Experiment: Feature Ranking Algorithm (FRA) Algorithm

The experiment results of FRA algorithm gives a list of attributes according to the ranking scores obtained from MPF calculation (as discussed in Chapter Four). Data were gathered from one thousand quality assessment data collected from the PQF model. In conjunction, Table 5.1 shows an example of the software quality attributes and the weights assigned by the assessors adapted from the PQF's database. As noticed earlier in Chapter Two, the weight assessment were made on a five point Likert scale (1= strongly disagree to 5 = strongly agree).

Table 5.1: Example of Software Quality Attributes with Assigned Weight

Attribute		Weight
1	Efficiency	4.08
2	Functionality	3.69
3	Maintainability	2.66
4	Portability	3.55
5	Reliability	3.36
6	Integrity	3.83
7	Usability	2.95
8	User Factor	3.67

According to the weight assigned by the assessors, the proposed algorithm counts the weight value using the MPF formula in phase one. The averaged score for each attribute is required to get the value of standard deviation in the data collection in order to attain the MPF score for each attribute. Normally, the standard deviation for one thousand data is dissimilar in term of the scores given by the assessors and the probability of similarity rarely occurs in the analysis.

As referred to the results in Table 5.2, the averaged score for attributes portability and User Conformity is 3.9, with standard deviations 1.58 and 1.38 respectively. This does not agree with the principle of redundancy, that standard deviations should also be similar. As been referred to the Table 5.2, it is seen that the averaged score for Maintainability and Usability is 3.4 with standard deviations is 1.8. In this case, they are meets the principle of redundancy. Hence, the MPF formula is used to count the priority of attributes. As referred to the full results of FRA algorithm in Table 5.3, the MPF score value for Maintainability and Usability is 45.0. Hence, they are proven redundant.

Table 5.2: Result in Averaged Score and Standard Deviation

Attribute Id	Attribute Name	Averaged Score	Std.Deviation
P006	Reliability	4.1	1.04
P005	Portability	3.9	1.38
P008	User Conformity	3.9	1.58
P001	Efficiency	3.7	1.19
P002	Functionality	3.6	1.63
P004	Maintainability	3.4	1.80
P007	Usability	3.4	1.80
P003	Integrity	3.1	1.51

Table 5.3: Result of FRA Algorithm

Attribute Id	**Attribute Name**	**MPF Score**
P008	User Conformity	47.34
P004	Maintainability	45.00
P007	Usability	45.00
P002	Functionality	40.63
P005	Portability	34.38
P006	Reliability	26.10
P003	Integrity	22.70
P001	Efficiency	17.81

As a consequent, to resolve this issue, the classification task with the weight as the main target is applied. The previous chapter explains that the classification task operates in phase two of FRA algorithm. This means that, the wrapper approach is conducted where the data related to this attributes are trained and tested for classification task. In this study, RA*k*el and ML*k*NN are employed in this task (explained in previous chapter). In the conjuction, The Area Under the Curve (AUC) performance metric is used to count classification accuracy. In relation, Table 5.4 lists the results of classification accuracy for handling the redundancy of data between the Maintainability and Usability.

Table 5.4: Result of Classification Accuracy for Two Redundant Attributes

Attribute Id	**Attribute Name**	**RA*k*el Classifier**	**ML*k*NN Classifier**	**Averaged**
P004	Maintainability	0.8121	0.9171	0.8646
P007	Usability	0.8131	0.7616	0.7874

As can be seen in Table 5.4, the classification accuracy of RA*k*el is almost identical for both Maintainability and Usability attributes. However, the ML*k*NN classifies the Maintainability attribute with 92% accuracy compared to Usability (79%). This finding supports the statement posted by Wang et al. (2011), that the accuracy of the classification result performed is influenced by the selected classifier. Hence, the result of classification accuracy is averaged in order to avoid biasness towards a single classifier. In this case, based on the detail in the table, the classification accuracy of the Maintainability attribute is higher than the Usability attribute (with a difference of 8%). Therefore, the Maintainability attribute is selected to be ranked higher than Usability. Consequently, the redundancy of attributes is ranked accordingly between the redundant data in the final ranking result. Also, the result of the others quality attributes are ready to be ranked respectively with the MPF scores of each attribute in final ranking result. Eventually, the final ranking result of the software quality attributes are displayed in tabular form and shown in Table 5.5.

Table 5.5: Final Ranking of FRA Algorithm

Attribute Id	Attribute Name
P008	User Conformity
P004	Maintainability
P007	Usability
P002	Functionality
P005	Portability
P006	Reliability
P003	Integrity
P001	Efficiency

Based on the details in Table 5.5, the User Conformity is ranked as the highest priority among the attributes in the software development. In contrast, the Efficiency attribute is found as the least priority attribute. Additionally, in order to access the effectiveness of FRA algorithm, the comparison between the final results with the ranking result produced by the KSCBF algorithm is carried out. For the purpose of making a baseline for the comparison, the result performed by expert judgement such as PQF model has been used, as explains in detail in the following subsection.

5.2.2 Result of Experiment: Kolmogorov-Smirnov Correlation Based Filter (KSCBF) Algorithm

To further evaluate the proposed algorithm, the result of FRA algorithm is compared to the KSCBF algorithm pertaining to the final scores obtained. Table 5.6 presents the score for attributes obtained from SU for filtering in KSCBF algorithm. It is followed by Table 5.7 that lists the final ranking result of scores for attributes in KSCBF algorithm.

Table 5.6: Result of Symmetrical Uncertainty (SU) Value

Attribute ID	Attribute Name	SU Value
P004	Maintainability	33.44
P007	Usability	23.33
P008	User Conformity	18.34
P001	Efficiency	13.92
P005	Portability	12.12
P006	Reliability	12.03
P002	Functionality	9.33
P003	Integrity	9.33

Table 5.7: Final Ranking Result of Attributes Scores in KSCBF Algorithm

Attribute ID	Attribute Name	KS- Test Score
P004	Maintainability	1.00
P007	Usability	0.83
P008	User Conformity	0.72
P001	Efficiency	0.63
P005	Portability	0.57
P006	Reliability	0.55
P002	Functionality	0.51
P003	Integrity	0.48

In KSCBF algorithm, the attributes are filtered and ranked according to the score obtained from the SU calculation. As be seen in the Table 5.6, the Maintainability is ranked the highest score (33.44). While, the Functionality and Integrity attributes are ranked the least (9.33). Further, the results of SU calculation are used in the KS-Test calculation to solve the redundancy.

In this experiment, the final scores obtained from the KSCBF that Maintainability is a very important feature to be highlighted and ranked highest with score 1. Grunwald et al. (2008), states that the better quality measures is closer to 1 as the higher degree. As can be seen from the Table 5.7, Integrity is ranked lowest (0.48). It means

that the attribute is less priority than the other attributes. In this algorithm, the step is completed once the K-S test is performed and the data are claimed as valid. Eventually, the final ranking results are outlined in the same ranking attributes although there are still redundancies. This is one of the limitations faced in KSCBF algorithm in handling the redundancies cases as mentioned earlier in Chapter Two.
This is supported the statement reported by Grunwald et al. (2004), who argues that this method will not be able to change the position of each attribute in the ranking list and the final results are outlined as the same ranking attributes with different values after K-S test calculation. In conjunction, the comparison on the results is elaborated in the following section.

5.3 Evaluation Measurement

The performance evaluation is very important in this study to validate the proposed algorithm. The purpose of this experimental evaluation is to present the accuracy of the proposed FRA algorithm. This section shows the evaluation measurement techniques that were used in the evaluation process. Particularly, the measurements used in this evaluation process are correlation coefficients and statistical significance test (t-test). Furthermore, the experimental evaluation results of FRA algorithm are compared to the KSCBF algorithm and PQF model. In detail, all testing procedures and results are described in the following section.

5.3.1 Human Expert Evaluation

In order to evaluate the performance of the proposed model, the result produced by the expert were used for the comparison and act as the benchmark of the assessment in this research. The ranking list by expert analysis quality by individual attribute based on previous and current assessment (adapted from Yahaya et al., 2011) is outlined in Appendix B. The results are averaged and used as the baseline to compare the performance of the proposed algorithm and KSCBF algorithm. In conjunction, a comparative graph is plotted in the next section to show the results.

5.3.2 Normalization of Data Performance

In order to compare the results, human expert ranking result and FRA is normalized to the standard scale between 1 and 0 for analysis. Thus, the result of KSCBF algorithm is determined between 0 and 1. Therefore, the result does not require any normalization process. Accordingly, Table 5.8 shows the final results performed by PQF model, FRA and KSCBF algorithm before normalization. In addition, Table 5.9 shows the normalization of the data and is used as data comparison between PQF model, FRA and KSCBF algorithm.

Table 5.8: Final Result of PQF model, FRA and KSCBF Algorithm

	Model / Algorithm		
Attribute Name	PQF	FRA	KSCBF
User Conformity	4.56	47.34	0.72
Maintainability	4.46	45.00	1.00
Usability	4.31	45.00	0.83
Functionality	4.30	40.63	0.51
Portability	4.12	34.38	0.57
Reliability	3.52	26.10	0.55
Efficiency	3.33	17.81	0.63
Integrity	3.25	22.70	0.48

According to the final result in Table 5.8, the data is transformed using the concept of normalization of data. The process of transformation is involved the formula in Equation 5.1 below:

$$\frac{a + (x - A) \text{ X } (b-a)}{(B-A)} \quad \text{(Eq. 5.1)}$$

Where, a is the smallest scale of data in the ranking, b is the biggest scale of the data in the ranking. In this study, the scale used was between 0 and 1. A is the smallest data in the ranking, B is the biggest data in the rankimg, and *x* is the final ranking result. After calculation for the normalization process, the result obtained is shown in Table 5.9 below.

Table 5.9: Normalization of Data

Attribute Name	Model / Algorithm		
	PQF	FRA	KSCBF
User Conformity	1.00	1.00	0.72
Maintainability	0.92	0.92	1.00
Usability	0.81	0.92	0.83
Functionality	0.80	0.77	0.51
Portability	0.66	0.56	0.57
Reliability	0.21	0.28	0.55
Efficiency	0.06	0.00	0.63
Integrity	0.00	0.17	0.48
Percentage of Similarity to PQF Model (%)		Score = 6/8 * 100 75%	Score = 3/8 * 100 37.5%

Table 5.9 reveals that User Conformity scores the highest in FRA (score = 1), similar to PQF model. In contrast, the KSCBF algorithm shows that Maintainability scores the highest (score = 1). On the other hand, FRA records that Efficiency attribute is the least important attributes to be highlighted in software development. This

contrast the PQF model and KSCBF algorithm, in which Integrity attributes is the lowest priority.

The findings explain that, the scores produced by FRA associates better with the expert model compared to KSCBF algorithm. Overall, the quality attributes performed by the proposed algorithm are strongly similar to the ranking by the expert model (with 75% of scores). In contrast, the quality attributes ranked by KSCBF algorithm shows that the model correlates inferior with the expert model (with 37.5% of scores). The percentage of the similarity is obtained by comparing the equality of attributes in final ranking to the expert model's attribute ranking. Otherwise, the dissimilarity judgement is referred to the comparison on inequality of attribute ranking compared to the expert model's attribute ranking.

In calculating the equality, the similarity of attributes to PQF model is divided by the total quality of quality attributes. The result is multiplied by 100% (Similarity of Attributes to PQF model / Total of Quality attributes * 100%). Hence, with reference to Table 5.9, the attributes are User Conformity, Maintainability, Usability, Functionality, Portability and Reliability (six attributes). This makes up 6/8 * 100%, which equal to 75%.

Meanwhile, in calculating the dissimilarity of attributes to PQF model is also divided by the total quality of quality attributes. The result is also multiplied by 100% (Dissimilarity of Attributes to PQF model / Total of Quality attribute * 100%). With reference to the Table 5.9, the attributes are Efficiency and Integrity (two attributes). This makes up 2/8 * 100%, which equal to 25%. The dissimilarity of results is compared to KSCBF algorithm, which counted to 62.5%. As a consequent, the line graph in Figure 5.1 are plotted as to show the ranking attributes of FRA, KSCBF algorithm, and PQF model.

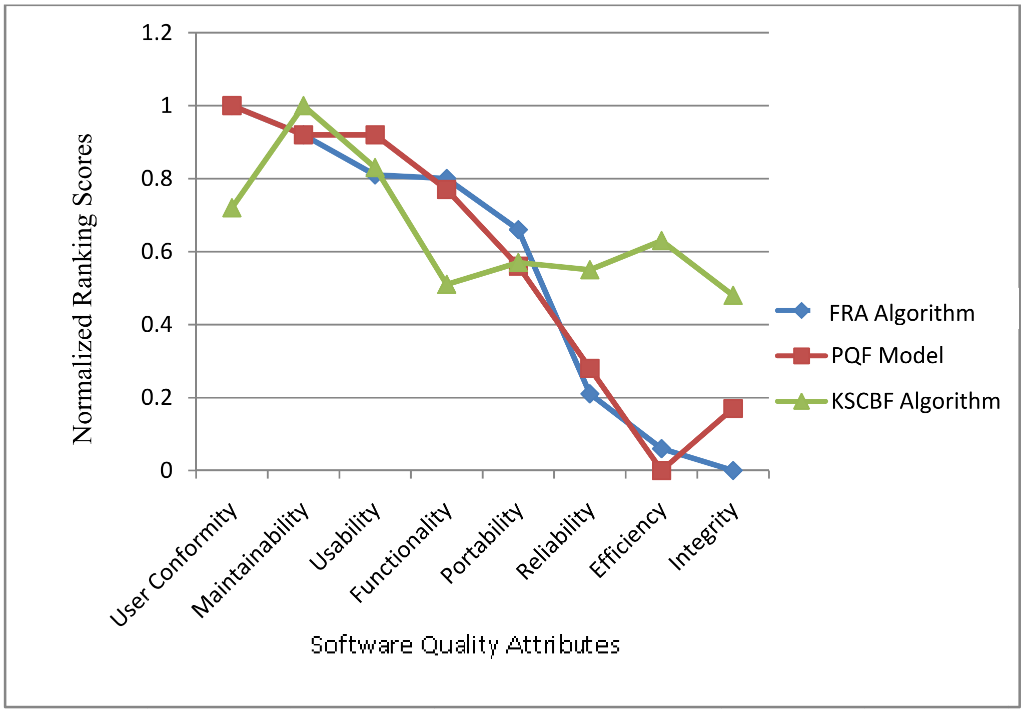


Figure 5.1: The Quality Attributes Ranking of FRA, KSCBF and PQF Model

As can be seen in Figure 5.1, the line graph of FRA line is similar to the expert model ranking as opposed to line graph of KSCBF algorithm. The attributes ranking in FRA as remained the same to the PQF model such as User Conformity, Maintainability, Usability, Functionality, Portability and Reliability. As for comparison to KSCBF algorithm only three attributes are similar to PQF model. There are Portability, Reliability and Integrity. This is proof that the FRA ranking is performed better than KSCBF algorithm and indirectly solving the feature ranking attributes problems.

In addition, the statistical measurement is also calculated using the correlation coefficient and statistical significant test of FRA and KSCBF algorithm, compared to the expert's ranking. The next sub sections discusses on this.

5.3.3 Correlation Coefficient

The performance of the proposed algorithm is analyzed by calculating the correlation coefficient to show the relationship between FRA and KSCBF algorithm with PQF model. The formula of correlation coefficient is exhibited in Equation 5.2.

$$\text{Correlation (r)} = \frac{N\Sigma XY - (\Sigma X)(\Sigma Y)}{\sqrt{([N\Sigma X^2 - (\Sigma X)^2][N\Sigma Y^2 - (\Sigma Y)^2])}} \qquad \text{(Eq. 5.2)}$$

Where, N = Number of values or elements, X = First Scores, Y = Second Scores, ΣXY = Sum of the product of first and Second Scores, ΣX = Sum of First Scores, ΣY = Sum of Second Scores, ΣX^2 = Sum of square First Scores and ΣY^2 = Sum of square Second Scores.

Furthermore, this analysis can be measured the strength of FRA and KSCBF algorithm related to the PQF model. Table 5.10 shows the analyzed result of correlation coefficient of FRA to the PQF model and followed by Table 5.11 shows tha analyzed result of correlation coefficient of KSCBF algorithm to the PQF model.

Table 5.10: Analyzed Results of Correlation Coefficient of FRA to PQF

Iteration (10 folds Cross validation)	**Correlation Coefficient**
1	0.96
2	0.97
3	0.98
4	0.97
5	0.97
6	0.98
7	0.98
8	0.99
9	0.98
10	0.99
Average of the result	0.977

Table 5.11: Analyzed Results of Correlation Coefficient of KSCBF to PQF

Iteration (10 folds Cross validation)	**Correlation Coefficient**
1	0.82
2	0.83
3	0.82
4	0.83
5	0.81
6	0.82
7	0.83
8	0.83
9	0.83
10	0.83
Average of the result	0.825

As can be seen in the Table 5.10 and Table 5.11, the results performed by iteration of 10 fold cross validation in order to gain the accuracy of the results by averaged them to avoid biasness. The correlation coefficient in Table 5.10 shows that FRA is strongly correlates to the expert judgement with scores of 0.98 or 98%. On a contrary, in the Table 5.11 shows the KSCBF algorithm correlates 83% with scores of 0.83 to expert judgement. The result shows that the compared method has a limitation in solving the data redundancy case and the reason for this difference is because the KSCBF does not use the learning concept in the presented algorithm.

5.3.4 Statistical Significance Test (t-test)

Finally, the statistical significant test is used to test and validate the differences in scores of the results obtained by matching pairs of FRA or KSCBF to the expert model. The statistical significance test aims to measure the probability and determine whether there is significant differences between the two models by chance of errors if the compared results remaine the same. Hence, the statistical technique measures the confidence level of the results obtained from the compared methods to determine either it is significantly different or not in each other.

The testing process is accomplished by using a statistical significance test formula which correlates the samples t-test to consider the correlation exists among two compared models or methods. For each dissimilarity score produced by method *A, Ai, (I =1,2,…….d)* it test for correlated samples, *t* to a compared method *Bj, (j=1, 2, …..d)* is given in Equation 5.3.

$$t = \frac{D}{\sqrt{\frac{Sd^2}{n}}} \qquad \text{(Eq. 5.3)}$$

Where, D = (Ai – Bj), is the difference between compared method A to an expert dissimilarity score, D = is the mean of the difference, $S^2d = \sum \frac{(Di-D)}{n-1}$, is the standard error of difference, and n is number of pairs.

Table 5.12 below shows the results of statistical significance test of FRA and KSCBF to the expert model by iteration of 10 folds cross validation as to proof the correctness. The specified alpha value used in this research is 0.05. The final result shows that the similarity of FRA algorithm to the expert model is 0.052, occurred by chance of errors as compared to the KSCBF algorithm (alpha is 0.048, which is lesser than the specified alpha value). This explains that on t-value proofs that FRA algorithm is validated and more significant than KSCBF algorithm related to the expert model. Consequently, Table 5.13 shows the final result of measurement methods for FRA and KSCBF algorithm related to PQF model.

Table 5.12: Analyzed Results of Statistical Significance Test of FRA and KSCBF

Iteration (10 folds Cross validation)	**FRA**	**KSCBF**
1	0.0052	0.0047
2	0.0053	0.0047
3	0.0053	0.0048
4	0.0053	0.0047
5	0.0052	0.0048
6	0.0052	0.0048
7	0.0053	0.0048
8	0.0052	0.0047
9	0.0052	0.0047
10	0.0052	0.0048
Average of the result	0.0524	0.0475

Table 5.13: Final Results of Correlation Coefficient and Statistical Significance Test for FRA and KSCBF to PQF Model

Item	**Correlation Coefficient**	**t-test**
FRA	0.98	0.052
KSCBF	0.82	0.048

5.4 Summary

This chapter reveals the results of both methods of evaluation involving human judgement and statistical measurement. It describes the results from the expert judgement compared to the proposed algorithm and KSCBF algorithm in the literature. Based on the performance of the results, this study concludes that the performance of FRA is better than the KSCBF algorithm. Meanwhile, the result of the measurements performance shows that FRA is correlates strongly to the expert model rather than KSCBF algorithm. Additionally, the results have also been validated by the statistical significant test (t-test), which performs excellent related to the expert model rather than KSCBF algorithm. In fact, the proposed algorithm can also support the limitation of the current model in software quality in terms of the assessment technique provided in this study. Nevertheless, the adaptation of intelligence tool set used in this study makes the model as good as human approach, which is capable to learn and notice the future requirements and expectations.

CHAPTER SIX
DISCUSSION AND CONCLUSION

6.1 Overview

This chapter concludes the research of study by emphasizing major research contributions, the value of the research to the software quality community, the problem faced in this research and the suggestions and recommendations for future work.

6.2 Research Summary

This study focused on the construction Feature Ranking Algorithm (FRA) using Feature Ranking Technique (FRT) for quality attributes assessment in software quality model. It consists of an algorithm for assessment technique including a set of formula for attribute selection that is the most priority of features in the quality attributes. The application of the classifiers i.e. RA*k*el and ML*k*NN has contributed in handling the redundancy of data and the result of the quality attributes ranking was validated.

The experiments were conducted to illustrate the capability of the proposed algorithm to achieve the goal and objectives of this research. The proposed technique produces in this research development performed better than the compared method in terms of solving the data redundancies in the process of ranking software quality attributes. Experimental results also show that the proposed algorithm produced ranking results which are comparable with the experts ranking. Furthermore, the statistical measurements such as correlation coefficient and statistical significant testing have proved that the proposed algorithm highly correlates to the expert's judgements and validated as statistically significant.

6.3 Research Contribution

In this thesis a new software quality model has been developed by enhancing of the PQF's assessment technique and this new model is called an Feature Ranking Algorithm (FRA). This section summarizes the main contributions of the thesis by referring to the research objectives as stated in Chapter One:

a. **To identify Feature Ranking Technique (FRT) as to improve Pragmatic Quality Factor (PQF) model**

This objective has been achieved with the FRT proposed in this thesis. This is a type of FS Technique that can enhance the assessment technique in the PQF model and induce dynamic element in the existing software quality model. The implementation of learning concept as mentioned earlier, provides more impact to the achievement of this objective. The proposed technique solves the problem faced in the static quality model in the literature by introducing dynamic elements.

b. **To develop and evaluate an assessment technique in PQF model with the proposed Feature Ranking Technique (FRT)**

The second objective is achieved through the development of a Feature Ranking Algorithm (FRA) using Feature Ranking Technique (FRT). The technique proposed in this research has accomplished this objective with the following contributions:

i. Provides an algorithm which contains a formula to measure and evaluate the quality attributes

Generation of an algorithm which incorporates a new assessment formula is capable to take into consideration the future requirements and expectations. The proposed assessment technique has performed well in the software quality attributes by contributing to the selection of attributes according to the most priority of features score value.

ii. Implementation of learning concept using classifiers such as Random *k*-Labelsets (RA*k*el) and Multi Label *k*-Nearest Neighbour (ML*k*NN)

The implementation of classifiers in the proposed assessment technique is another contribution in this research. The learning concept in the proposed FRA algorithm enables handling the redundancy of data. This application involved the classifiers such as RA*k*el and ML*k*NN as discussed earlier. Each of the classifier was implemented with improved method to ensure the accuracy of the ranking attributes in solving the redundancies of data.

iii. Human Expert Evaluation proof the performance of FRA algorithm

The ranking results produced by the expert model such as PQF model is used as a baseline for the comparison. The ranking attributes by expert model is also used as a benchmark to compare the performance of FRA algorithm and the KSCBF algorithm presented in the literature. This evaluation technique confirms that the developed intelligent algorithm performed better than the compared method. In fact, the result produced by the proposed algorithm strongly correlates to the expert judgment in term of accuracy of quality attributes ranking. Furthermore, the FRA ranking result is better compared to the KSCBF algorithm. The comparison results act as the contribution in this research in terms of the performance of the proposed algorithm.

iv. Statistical Measurement

Another contribution is achieved in this research by performing correlation coefficient calculation and statistical significant testing to validate and prove the results. This statistical evaluation method has confirmed that the performance provided by FRA model is reliable. The FRA algorithm has achieved the objective in this research by producing the better results compared to the KSCBF algorithm.

Overall, this study has achieved its intended objectives outlined in Chapter One. The proposed algorithm presented in this study is considered extremely suitable for dynamic software quality model in evaluating the quality attributes.

6.4 Limitation of the Research

Every research study will face difficulties in the research process due to various constraints. As discussed earlier, the approaches used in FS technique in this research are filter and wrapper approaches. The limitation faced in this research is on the application of the approaches where both of this approaches were used separately in the proposed algorithm. The filter approach is enclosed in the MPF formula for ranking of the quality attributes. This function was implemented in the first phase of the proposed algorithm. While, the wrapper approach was used in the second phase for handling the redundancies of attribute ranking. The quality attributes is ranked accordingly and stored in knowledge base respectively if the redundancies of attributes do not exist in the first phase. Besides that, the second phase will not be implemented if the data redundancies do not occur. Consequently, the machine learning adaptation is unaccomplished if the second phase does not executed. However, this limitation is rarely possible to happen due to the data applied in this research is large enough and the chances of data repetition are high.

The next section discusses the future work to expand the contribution and to support the limitation of this research.

6.5 Future Work

For future development and expansion of this research, the following are suggested:

1. Improved FRA algorithm as hybrid algorithm consisting a combination of approaches in FS i.e. filter and wrapper used inclusively. In this research, major part of the algorithm focused on filter approach and wrapper has used exclusively for solving the redundancy of data. The wrapper approach is more suited for high dimensional data especially to evaluate the usefulness of features.
2. Enhance the proposed algorithm using more than two classifiers such as SVM, NB, LR, C4.5, and MLP. In this research, the classifiers provided by MULAN application are used to learn the knowledge from the software quality data. Although, the function in the MULAN Java library has limited functionality and capability to import classifiers indirectly from WEKA. The

application of WEKA as a learning tool can provide more classifiers to be used in solving the data redundancy in high dimensionality of data.

3. Finally, the development of FRA algorithm was based on FS technique as the main technique in the software quality assessment engine. Besides that, other techniques in AI approach can be explored such as NN and GA.

6.6 Summary

As a conclusion for this research, this chapter has discussed and concluded the overall research, the contribution of the research and followed by the limitation of the research. The discussion and recommendations for further development and extension were outlined. The embedding of AI approach contributes to the assessment technique as good as human approach and reduced uncertainty of quality attributes. Hence, the developed model can be a good alternative model to support the software quality model community in evaluating the quality attributes with intelligence technique. Hopefully, the findings of this study are able to provide positive inputs to the future researches and the evaluation efforts can encourage individuals and organizations in software quality community to utilize the proposed dynamic model as well as using an intelligent approach in the assessment engine.

REFERENCES

Aamodt, A. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, *7*, 39–59.

Abran, A., Khelifi, A. & Suryn, W. (2003). Usability meanings and interpretations in ISO standards. *Software Quality Journal, 11,* 323-336.

Aguero, M., Madou, F., Esperon, G. & Lopez, D. L. (2010). Artificial intelligence for software quality improvement. *World Academy of Science and Technology, 63*.

Anna, S. C., Garcia, A. P., Chavez, C. V. & Lucena, C. J. (2003). On the reuse and maintenance of aspect-oriented software: An assessment framework. *Computer Science Department,* Retrieved September 27, 2010 from IEEE Computer Society.

Allen, E. B. (2001). Controlling Overfitting in Classification Tree Models of Software Quality. *2001 International Symposium on Empirical Software Engineering, 2001.*, 59–79.

Ashrafi, N. (2003). The impact of software process improvement on quality in theory and practice. *Information & Management*, *40*(7), 677–690.

Azuma, M. (1991). SQuaRE the next generation of the ISO/IEC 9126 and 14598 international standards series on software product quality. *Technical Report, ISO/IEC* JTC1/SC7/WG6.

Bansiya, J., & Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering*, *28*(1), 4–17.

Baumgartner, D., & Serpen, G. (2009). Large Experiment and Evaluation Tool for WEKA Classifiers. *5th International Conference on Data Mining*, 340–346.

Bevan, N. (1984). Quality in use: Incorporating Human Factors into the Software Engineering Lifecycle. *Software Engineering Standards Symposium and Forum, 1997.'Emerging International Standards'. ISESS 97., Third IEEE International*, 169–179.

Bevan, N. (1997). Quality and usability: A new framework. *National Physical Laboratory*, United Kingdom.

Bevan, N. (1999). Quality in use: Meeting user needs for quality. *The Journal of Systems and Software, 49, March 19.* 89-96.

Bevan, N. (1999). Quality in use: Incorporating human factors into the software engineering lifecycle. *National Physical Laboratory*, Retrieved September 27, 2010 from IEEE Computer Society.

Bhatti, S. N. (2005). Why Quality? ISO 9126 Software Quality Metrics (Functionality) Support by UML Suite. *Advances in Engineering Software*, *30*(2), 1–5.

Biesiada, J., & Duch, W. (2005). Feature Selection for High-Dimensional Data: A Kolmogorov-Smirnov Correlation-Based Filter. *Advances in Soft Computing*, *30*, 95–103.

Biesiada, J., & Duch, W. (2007). Feature Selection for High-Dimensional Data: A Pearson Redundancy Based Filter. *Advances in Soft Computing*, *45*, 242–249.

Blachnik, M., Duch, W., Kachel, A., & Biesiada, J. (2009). Feature Selection for Supervised Classification: A Kolmogorov-Smirnov Class Correlation-Based Filter. In *AIMeth, Symposium On Methods Of Artificial Intelligence. Gliwice, Poland (10-19 November 2009)*.

Blum, A. L., & Langley, P. (1997). Artificial Intelligence Selection of relevant features and examples in machine. *Artificial Intelligence*, *97*, 245–271.

Briand, L. et al., (2000). Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of Systems and Software 51*, 245–273.

Buglione, L. & Abran, A. (1999). A quality factor for software. Proceedings from: *QUALITA99, 3rd International Conference on Quality and Reliability,* 335-344.

Burgess, C. J. (2000). Using Artificial Intelligence to solve problems in software quality management. Proceedings from: *The 8th International Conference on Software Quality Management (SQM2000), Software Quality Management VIII*. ISBN 1-902505-25-5, 77–89.

Cheikhi, L., Abran, A. & Suryn, W. (2006). Harmonization of usability measurements in ISO9126 software engineering standards. Proceedings from: *IEEE International Conference on Software Engineering, July 9-12, 2006, Montreal Quebec, Canada,* ISBN: 1-4244-0497-5, 3246-3251.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, *1*(1-4), 131–156.

Dash, M., & Liu, H. (2003). Consistency-based search in feature selection. *Artificial Intelligence*, *151*(1-2), 155–176.

Denning, P. J. (1992). What is Software Quality? *A Commentary from Communications of ACM* (January).

Deraman, A. & Yahaya, J. H. (2010). Measuring the unmeasurable characteristics of software quality using pragmatic quality factor. Proceedings from: *2010 3rd IEEE International Conference on Computer Science and Information Technology, July 7-10, 2010, Chengdu, China,* ISBN:978-1-4244-5539-3, 197-202.

Dromey, R. G., & Popper, K. (1994). A model for software product quality. *Software Quality Institute*, (October), 1–35.

Dromey, G. R. (1995). A model for software product quality. *IEEE Transaction on Software Engineeering, February, 21(2),* 146-162.

Dromey, G. R. (1998). Software product quality: Theory, model and practice. *Software Quality Institute. Griffith University, Brisbane, Technical Report*. Retrieved 23 August, 2010, from http://www.sqi.gu.edu.au.

Dromey, G. R. (1999). Cornering the chimera, *IEEE Software, January*, 33-43.

Duch, W., Winiarski, T., Biesiada, J., & Kachel, A. (2003, June). Feature selection and ranking filters. *International Conference on Artificial Neural Networks (ICANN) and International Conference on Neural Information Processing (ICONIP)*. 251-254.

Durrett, R. (2010). *Probability: theory and examples*. Cambridge University Press.

Engineers, E. (1993). *IEEE Standard for a Software Quality Metrics Methodology* (pp. 1–73).

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research, 3,*1289–1305.

Fitzpatrick, R. (1996). Software quality: Definitions and strategies issues. Retrieved Sept 13, 2010, from http://ieeexplore.ieee.org/xpl/standards.jsp.

Fitzpatrick, R. & Higgins, C. (1998). Usable software and its attributes synthesis of software quality: European community law and human-computer. Retrieved Sept 13, 2010, from http://ieeexplore.ieee.org/xpl/standards.jsp.

Forman, G. (2003). An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Machine Learning Research*, *3*, 1289–1305.

Frank, E., Hall, M., Trigg, L., Holmes, G., & Witten, I. H. (2004). Data mining in bioinformatics using Weka. *Bioinformatics (Oxford, England)*, *20*(15), 2479–81.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, *29*(2), 131-163.

Gao, K., Khoshgoftaar, T. M., & Napolitano, A. (2009). Exploring Software Quality Classification with a Wrapper-Based Feature Ranking Technique. *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, 67–74.

Gao, K., Raton, B., & Wang, H. (2009). An Empirical Investigation of Filter Attribute Selection Techniques for Software Quality Classification. *Information Reuse & Integration, 2009. IRI'09. IEEE International Conference*, 272–277.

Gao, K. (2010). An Evaluation of Sampling on Filter-Based Feature Selection Methods. *Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference (FLAIRS 2010)*, 416–421.

Garcia, A. F. (2003). On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework. *Proceedings of Brazilian Symposium on Software Engineering*, 19-34.

Goulao, M. & Abreu, F. B. (2007). Towards a components quality model. *Information Systems Group (INESC)*, Retrieved Oct 26, 2010, from http://www.msc.com.my/xtras/fact_figures/msc.asp.

Grunwald, P., & Vitanyi, P. (2008). Shannon Information and Kolmogorov Complexity, 1–54. Retrieved Nov 29, 2012, from http://arxiv.org/abs/cs/0410002

Guyon, I. (2003). An Introduction to Variable and Feature Selection 1 Introduction. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hall, M. A., & Smith, L. A. (1997). Feature Subset Selection: A Correlation Based Filter Approach. Retrieved Aug 8, 2011, from http://scholar.google.com.my/scholar?q=feature+subset+selection%3A+A+correlation+based+filter+approach&hl=en&as_sdt=0%2C5

Hall, M. A., & Smith, L. A. (1998). Practical Feature Subset Selection for Machine Learning. Retrieved Aug 8, 2011, from http://researchcommons.waikato.ac.nz/bitstream/handle/10289/1512/Practical%20feature%20subset?sequence=1

Hamann, D., Jarvinen, J., & Birk, A. (1998). A Product-Process Dependency Definition Method. *IEEE Transactions on Software Engineering*, *15504*(23239), 898–904.

Hall, M. (1999). Correlation-based feature selection for machine learning. Retrieved Aug 8, 2011, from http://www.lri.fr/~pierres/donn%E9es/save/these/articles/lpr-queue/hall99correlationbased.pdf

Hall, M. A. (2000). Benchmarking Attribute Selection Techniques for Data Mining. Retrieved Aug 8, 2011, from http://researchcommons.waikato.ac.nz/bitstream/handle/10289/1026/uow-cs-wp-2000-10.pdf?sequence=1

Hall, M. A., & Holmes, G. (2003). Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, *15*(6), 1437–1447.

Hall, M., National, H., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10-18.

Hall, M. A., & Witten, I. H. (2010). WEKA Experiences with a Java Open-Source Project. *Journal of Machine Learning Research*, *11*, 2533–2541.

Humphrey, W. S., Kitson, D., Olson, T. G., Humphrey, W. S., & Kitson, D. (1989). *Conducting SEI-Assisted Software Process Assessments Conducting SEI-Assisted. Software Engineering Institute.*

Ioannou, M., Sakkas, G., Tsoumakas, G., & Vlahavas, I. (2010). Obtaining Bipartitions from Score Vectors for Multi-Label Classification. *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, 409-416.

IEEE. (1993). IEEE standard for a software quality metrics methodology. Retrieved August 20, 2010, from http://ieeexplore.ieee.org/xpl/standards.jsp.

ISO/IEC 9126. (1996). Software quality characteristics and metrics-Part2: External metrics. *Technical Report, ISO/IEC* JTC1/SC7/WG6.

Jain, A., & Zongker, D. (1997). Feature Selection: Evaluation, Application, and Small Sample Performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *19*(2), 153-158.

Jenner, M. G. (1995). *Software Quality Management and ISO 9001*. New York: A Wiley/QED publication.

Jiang, B., Ma, L., & Xie, W. (2008). A Hybrid Feature Selection Algorithm: Combination of Symmetrical Uncertainty and Genetic Algorithms. *The Second Internatiional Symposium on Optimization and System Biology (OSB'08)*, 152–157.

John, G. H., Kohavi, R., & Karl, P. (1994). Irrelevant Features and the Subset Selection Problem. *Proceedings of the Eleventh International Conference* (pp. 121–129).

Jorgensen, M. (1999). Software quality measurement. *Advances in Engineering Software*, *30*(12), 907–912.

Jung, H., & Kim, S. (2004). Product Quality: A Survey. *IEEE Computer Society*, 88–92.

Khoshgoftaar, T. M., Munson, J. C., Bhattacharya, B. B., & Richardson, G. D. (1995). Predictive modeling techniques of software quality from software measures. *IEEE Transactions on Software Engineering*, *18*(11), 979-987.

Khoshgoftaar, T. M., Allen, E. B., Hudepohl, J. P., & Aud, S. J. (1997). Application of neural networks to software quality modeling of a very large telecommunications system. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, *8*(4), 902–9.

Khoshgoftar, T., Chien, P. D., & Allen, E. (1998). GP-based software quality prediction. *Proceedings of the Third Annual Conference Genetic Programming, volume* (pp. 60-65).

Khoshgoftaar, T. M., Allen, E. B., Halstead, R., Trio, G. P., & Flass, R. M. (1999). Using process history to predict software quality. *Computer* (Vol. 31, pp. 66-72).

Khoshgoftaar, T. M., & Allen, E. B. (2000). A practical classification rule for software quality models. *Reliability, IEEE Transactions on*, *49*(2), 209-216.

Khoshgoftaar, T. M., Yuan, X., & Allen, E. B. (2000). Balancing misclassification rates in classification tree models of software quality. *Empirical Software Engineering*, *5*(4), 313-330.

Khoshgoftaar, T. M., Nguyen, L., Gao, K., & Rajeevalochanam, J. (2003). Application of an attribute selection method to CBR-based software quality classification. *Proceedings of 15th IEEE International Conference on Tools with Artificial Intelligence*, 47–52.

Khoshgoftaar, T. M., & Su, X. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, *2009*, Retrieved Aug 8, 2011, from http://dl.acm.org/citation.cfm?id=1722966

Khoshgoftaar, T. M., Wang, H., & Van Hulse, J. (2010). A comparative study of threshold-based feature selection techniques. In *Granular Computing (GrC), 2010 IEEE International Conference.* 499-504.

Khosravi, K. (2004). A Quality Model for Design Patterns. Retrieved Aug 8, 2011, from http://www-etud.iro.umontreal.ca/~ptidej/yann-gael/Work/Publications/Documents/041021+Kashayar+Khosravi+Technical+Report.doc.pdf

Kilidar, H., Cox, K., & Kitchenham, B. (2005). The use and usefulness of the ISO/IEC 9126 quality standard. *2005 International Symposium on Empirical Software Engineering, 2005.*, 122–128.

Kitchenham, B., & Pfleeger, S. L. (1996). Introduction Software quality: The Elusive Target. *IEEE Software*, *13*(1), 12–21.

Kira, K., & Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the National Conference on Artificial Intelligence*. 129-129.

Kohavi, R., & John, G. H. (1996). Wrappers for Feature Subset Selection. *Artificial Intelligence*, *97*(1), 273–324.

Kolodner, J. L. (1992). An Introduction to Case-Based Reasoning. *Artificial Intelligence Review*, *6*(1), 3–34.

Kolodner, J. (1993). *Case-Based Reasoning, Morgan Kaufmann.*

Kolodner, J. L., Simpson, R. L, & Cyrans, K. S. (1993). A process model of cased-based reasoning in problem solving. *Georgia Institute of Technology, Atlanta, Technical Report.* Retrieved 27 August 2010, from http://www.sqi.gu.edu.au.

Kononenko, I. (1994). Estimating Attributes: Analysis and Extensions of RELIEF. *Machine Learning: ECML-94*, 171–182.

Kumar, R., Rai, S. & Trahen, J. L. (1998). Neural network techniques for software quality evaluation. *Proceedings of the Annual Reliability and Maintainability Symposium*, 155-161.

Laboratorio, M. O. (2002). A Systemic Quality Model for Evaluating Software Products. Retrieved Sept10, 2011, from http://www.lisi.usb.ve/publicaciones/02 calidad sistemica/calidad_24.pdf

Langley, P., & Flamingo, L. (1994). Selection of Relevant Features in Machine Learning. *AAAI Technical Report FS-94-02*, 127–131.

Langley, P., & Blum, A. L. (1997). Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, *97*, 245–271.

Lee, Y. W., Strong, D. M., Kahn, B. K., & Wang, R. Y. (2002). AIMQ: A Methodology for Information Quality Assessment. *Information & Management*, *40*(2), 133–146.

Lees, B., Hamza, M., & Irgens, C. (1996). Applying Case-Based Reasoning to Software Quality Management. *Burkhard & Lenz (1996)*, 162-169.

Li, S., Xia, R., Zong, C., & Huang, C. R. (2009). A Framework of Feature Selection Methods for Text Categorization. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, *2*(August), 692–700.

Liu, H., Li, J., & Wong, L. (2002). A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome informatics. International Conference on Genome Informatics*, *13*, 51–60.

Moreira, A., Araújo, J., & Brito, I. (2002). Crosscutting quality attributes for requirements engineering. *Proceedings of the 14th international conference on Software engineering and knowledge engineering - SEKE '02*, 167-174.

Norman, F. (2002). Body of Knowledge for Software Quality. *IEEE Computer Society*, (February), 77-82.

Olivier, P. (2001). Diagrammatic reasoning: An artificial intelligence perspective. *Artificial Intelligence Review*, *15*(1), 63-78.

Ortega, M., Pérez, M., & Rojas, T. (2000). A model for software Product Quality with a Systemic Focus. *4th World Multiconference on Systemics, Cybernetics and Informatics SCI 2000 and The 6th International Conference on Information Systems, Analysis and Synthesis ISAS 2000* (pp. 395-401).

Ortega, M., & Rojas, T. (2003). Construction of a systemic quality model for evaluating a software product. *Software Quality Jurnal*, *11*(July), 219–242.

Ortega, M., Perez, M. & Rojas, T. (2003). A model for software product quality with a systemic focus. Retrieved October 28, 2010 from IEEE Computer Society.

Pfleeger, S. L. (2001). *Software Engineering: Theory and Practice", 2nd ed*. Upper Saddle River, N.J: Prentice Hall.

Pomerol, J. C. (1997). Artificial intelligence and human decision making. *European Journal of Operational Research*, *99*(1), 3-25.

Punch, W. F., Goodman, E. D., Pei, M., Chia-shun, L., Hovland, P., & Enbody, R. (1993). Further Research on Feature Selection and Classification Using Genetic Algorithms. *Proceedings of the 5th International Conference on Genetic Algorithms*, (Jun), 557–564.

Qutaish, R. E. (2009). Measuring the software product quality during the software development life-cycle: An international organization for standardization standard perspective. *Journal of Computer Science, 5 (5),* 392-397.

Raton, B. (2003). Fault Prediction Modeling for Software Quality Estimation: Comparing Commonly Used Techniques. *Empirical Software Engineering*, *8*, 255–283.

Rawashdeh, A., & Matalkah, B. (2006). A new software quality model for evaluating COTS components. *The Journal of Computer Science 2 (4).* 373-381.

Reformat, M., Pedrycz, W., & Pizzi, N. J. (2003). Software Quality Analysis with the Use of Computational Intelligence. *Information and Software Technology*, *45*(7), 405–417.

Rieser, V., & Lemon, O. (2006). Using Machine Learning to Explore Human Multimodal Clarification Strategies. *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, (July), 659–666.

Robu, R., Stoicu., & Tivadar, V. (2010). Arff Convertor Tool for WEKA Data Mining Software. *2010 International Joint Conference on Computational Cybernetics and Technical Informatics*, 247–251.

Ross, S., Fang, L., & Hipel, K. W. (2002). A case-based reasoning system for conflict resolution: design and implementation. *Engineering Applications of Artificial Intelligence*, *15*(3), 369-383.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A Review of Feature Selection Techniques in Bioinformatics. *Bioinformatics (Oxford, England)*, *23*(19), 2507-2517.

Sajnani, H., Javanmardi, S., Mcdonald, D. W., & Lopes, C. V. (2010). Multi-Label Classification of Short Text: A Study on Wikipedia Barnstars. *The AAAI-11 Workshop on Analyzing Microtext.* Retrieved Nov 23, 2011, from http://pensivepuffin.com/dwmcphd/papers/Sajani.et.al-MultiLabelBarnstars-AAAIShortTextWorkshop.pdf

Science, F., & Box, P. O. (2006). A New Software Quality Model for Evaluating COTS Components Adnan Rawashdeh and Bassem Matalkah. *Journal of Computer Science*, *2*(4), 373–381.

Software, E., & Raton, B. (2004). Comparative Assessment of Software Quality Classification Techniques: An Empirical Case Study. *Empirical Software Engineering*, *9*, 229–257.

Sonnenburg, S., Braun, M. L., Ong, C. S., Bengio, S., Bottou, L., Holmes, G., & Williamson, R. C. (2007). The need for open source software in machine learning. Retrieved Nov 26, 2012, from http://researchcommons.waikato.ac.nz/handle/10289/3928

Stefani, A., & Xenos, M. (2008). E-Commerce System Quality Assessment using a Model based on ISO 9126 and Belief Networks. *Software Quality Journal*, *16*(March), 107–129.

Suryn, W., Abran, A. & April, A. (2003). ISO/IEC SQuaRE: The second generation of standards for software product quality. Retrieved December 20, 2010, from http://www.lrgl.uqam.ca/publications/pdf/799.pdf.

Swiniarski, R. W., & Skowron, A. (2003). Rough Set Methods in Feature Selection and Recognition. *Pattern Recognition Letters*, *24*, 833–849.

Szabo, R. M., & Guasti, P. J. (1995). Exploring the Behaviour of Neural Network Sofware Quality Models. *Software Engineering Journal*, (May), 89–96.

Tadeuchi, Y., Oshima, R., Nishida, K., Yamauchi, K., & Omori, T. (2007). Quick Online Feature Selection Method for Regression-A Feature Selection Method Inspired by Human Behavior. *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference.* pp. 1895-1900.

Tadeuchi, Y., Oshima, R., Nishida, K. & Yamauchi, K. (2007). A feature selection method inspired by human behavior. *University Research Institute, Tamagawa.* 1895-1900.

Tahir, M. A., Kittler, J., Mikolajczyk, K., & Yan, F. (2010). Improving Multilabel Classification Performance by Using Ensemble of Multi-label Classifiers. *Multiple Classifier Systems*, 11–21.

Tervonen, I. (1996). Support for quality-based design and inspection. *IEEE Software (January),* 44-54.

Thwin, M. M. T., & Quah, T. S. (2005). Application of Neural Networks for Software Quality Prediction using Object-Oriented Metrics. *Journal of Systems and Software*, *76*(2), 147–156.

Tomar, A. B. (2011). A Systematic Study of Software. *International Journal of Software Engineering & Application (IJSEA)*, *2*(4), 61–70.

Trohidis, K., & Kalliris, G. (2008). Multi Label Classification of Music into Emotion. *ISMIR 2008: Proceedings of the 9th International Conference of Music Information Retrieval*, 325–330.

Tsoumakas, G., & Vlahavas, I. (2010). Random k-labelsets: An ensemble method for multilabel classification. *Machine Learning: ECML 2007*, 406-417.

Tsoumakas, G., & Vilcek, J. (2011). MULAN: A Java Library for Multi-Label Learning. *Journal of Machine Learning Research*, *12*, 2411–2414.

Vivanco, R. (2007). Improving Predictive Models of Software Quality Using an Evolutionary Computational Approach. *2007 IEEE International Conference on Software Maintenance*, 503–504.

Wang, Q. (2009). Feature Selection and Clustering in Software Quality Prediction. *Evaluation and Assessment in Software Engineering 2007*, 1–12.

Wang, H., Khoshgoftaar, T. M., Gao, K., & Seliya, N. (2009). High-Dimensional Software Engineering Data and Feature Selection. *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, 83–90.

Wang, H., Khoshgoftaar, T. M., & Seliya, N. (2011). How many software metrics should be selected for defect prediction? In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference* (pp. 69-74).

Weiss, N.A. (2008). *Introductory Statistics*. Pearson International Edition.

Wenger, E. (2004). Artificial intelligence and tutoring systems. *International Journal of Artificial Intelligence in Education*, *14*, 39-65.

Whittaker, J. A., & Voas, J. M. (2002). 50 years of software: key principles for quality. *IT professional*, *4*(6), 28-35.

Witten, I. H., Frank, E., Trigg, L., Hall, M., Holmes, G., & Cunningham, S. J. (1999). Weka: Practical Machine Learning Tools and Techniques with Java Implementations. Retrieved January 10, 2012, from http://scholar.google.com.my/scholar?q=Weka+%3A+Practical+Machine+Learning+Tools+and+Techniques+with+Java+Implementations&hl=en&as_sdt=0%2C5

Wolf, L., & Shashua, A. (2003). Feature selection for unsupervised and supervised inference: the emergence of sparsity in a weighted-based approach. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference,* 378-384.

Xenos, M., & Christodoulakis, D. (1997). Measuring perceived software quality. *Information and Software Technology*, *39*(6), 417–424.

Yahaya, J. H., Deraman, A. & Hamdan, A. R. (2007). A case study in applying software certification model by product quality approach. *The International*

Conference on electrical Engineering and Informatics, June 17-19, Bandung, Indonesia, 706-709.

Yahaya, J. H, Deraman, A. & Hamdan, A. R. (2008). Software quality from behavioural and human perspectives. *IJCSNS International Journal of Computer Science and Network Security, 8(8), August 30*, 53-63.

Yahaya, J. H, Deraman, A. & Hamdan, A. R. (2008). Software certification implementation: Case study analysis and findings. *The 3rd International Symposium on Information Technology 2008 (ITSIM 2008), August 26-29, Kuala Lumpur,* 1541-1548.

Yahaya, J. H., Deraman, A., Hamdan, A. R. & Baharom, F. (2008). Software product certification: A collaborative perspective approach. *The 9th Asia Pacific Industrial Engineering & Management Systems Conference, December 3-5, Bali, Indonesia,* 760-768.

Yahaya, J. H., Deraman, A. & Hamdan, A. R. (2010). Continuously ensuring quality through software certification: A case study. *The International Conference on Information Society (i-Society 2010), June 28-30, London, UK.*

Yang, J., & Honavar, V. (1997). Feature Subset Selection Using A Genetic Algorithm Feature Subset Selection Using 1 Introduction. *Intelligent Systems and Their Applications*, *13*(2), 44–49.

Yu, L., & Liu, H. (2003). Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, *2*(2), 856–864.

Yu, L., & Liu, H. (2004). Efficient Feature Selection via Analysis of Relevance and Redundancy. *Journal of Machine Learning Research*, *5*, 1205–1224.

Appendix A
Sample Data

Attributes Score Assigned by the Assessors adapted from Yahaya et al. (2011).

Efficiency	Functionality	Maintainability	Portability	Reliability	Integrity	Usability	UserComfomity
3	2	1	3	1	2	2	2
5	3	4	2	3	3	5	5
3	4	2	4	1	4	3	2
4	1	4	1	2	1	4	3
5	3	4	1	3	3	5	5
2	4	5	5	1	4	3	2
4	2	1	3	2	2	2	3
3	3	4	4	3	3	5	4
5	1	2	1	3	1	3	2
2	4	4	3	2	4	4	5
4	3	1	1	4	3	4	3
3	2	1	3	1	2	3	4
1	3	5	3	3	3	2	2
4	4	4	1	2	4	5	3
2	1	5	4	1	1	4	5
1	3	1	3	3	3	4	4
3	2	5	1	2	2	3	3
4	1	4	1	3	1	4	2
2	3	1	2	2	3	2	4
4	5	5	3	2	5	5	5
1	2	5	3	3	2	4	3
4	3	4	5	1	3	3	2
2	1	1	1	3	1	5	4
3	5	5	4	3	5	4	4
4	3	4	3	4	3	5	4
1	2	1	3	2	2	3	5
5	3	5	5	3	3	5	3
5	1	4	1	1	1	2	4
4	4	4	1	2	4	5	3
2	1	5	4	1	1	4	5
1	3	1	3	3	3	4	4
3	2	5	1	2	2	3	3

Appendix B

Result of Experiment

P001: Efficiency Attribute

```
RaKEL-------------------------------

Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.2179±0.0267
Subset Accuracy: 0.2488±0.0386
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6173±0.0501
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5098±0.0501
Micro-averaged Precision: 0.6592±0.0617
Micro-averaged Recall: 0.6260±0.0549
Micro-averaged F-Measure: 0.6407±0.0401
Macro-averaged Precision: 0.6509±0.0600
Macro-averaged Recall: 0.6144±0.0556
Macro-averaged F-Measure: 0.6230±0.0575
Average Precision: 0.7762±0.0405
Coverage: 1.9414±0.1693
OneError: 0.3120±0.0722
IsError: 0.5297±0.0689
ErrorSetSize: 1.3575±0.2418
Ranking Loss: 0.1891±0.0359
Mean Average Precision: 0.6919±0.0506
Micro-averaged AUC: 0.8269±0.0261
Macro-averaged AUC: 0.8104±0.0272

---------MLkNN-------------------------------------------------
Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.1991±0.0293
Subset Accuracy: 0.2831±0.0532
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6050±0.0568
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5346±0.0615
Micro-averaged Precision: 0.7241±0.0571
Micro-averaged Recall: 0.6087±0.0506
Micro-averaged F-Measure: 0.6598±0.0723
Macro-averaged Precision: 0.7230±0.0692
Macro-averaged Recall: 0.5922±0.0425
Macro-averaged F-Measure: 0.6243±0.0413
Average Precision: 0.7965±0.0406
Coverage: 1.7884±0.1634
OneError: 0.2835±0.0740
IsError: 0.5028±0.0815
ErrorSetSize: 1.1443±0.2128
Ranking Loss: 0.1633±0.0320
Mean Average Precision: 0.7271±0.0426
Micro-averaged AUC: 0.8500±0.0235
Macro-averaged AUC: 0.8346±0.0501
```

P002: Functionality Attribute

```
RaKEL------------------------------

Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.2178±0.0255
Subset Accuracy: 0.2478±0.0486
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6273±0.0601
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5088±0.0581
Micro-averaged Precision: 0.6572±0.0557
Micro-averaged Recall: 0.6260±0.0588
Micro-averaged F-Measure: 0.6417±0.0601
```

```
Macro-averaged Precision: 0.5509±0.0500
Macro-averaged Recall: 0.6144±0.0539
Macro-averaged F-Measure: 0.6230±0.0465
Average Precision: 0.7762±0.0405
Coverage: 1.9414±0.1698
OneError: 0.3120±0.0724
IsError: 0.5297±0.0689
ErrorSetSize: 1.3575±0.2418
Ranking Loss: 0.1891±0.0359
Mean Average Precision: 0.6918±0.0505
Micro-averaged AUC: 0.8272±0.0261
Macro-averaged AUC: 0.8104±0.0272

---------MLkNN--------------------------------------------------
Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.1951±0.0243
Subset Accuracy: 0.2831±0.0538
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6050±0.0578
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5326±0.0515
Micro-averaged Precision: 0.7242±0.0571
Micro-averaged Recall: 0.6087±0.0505
Micro-averaged F-Measure: 0.6598±0.0423
Macro-averaged Precision: 0.7330±0.0692
Macro-averaged Recall: 0.5922±0.0425
Macro-averaged F-Measure: 0.6243±0.0413
Average Precision: 0.7965±0.0406
Coverage: 1.7884±0.1634
OneError: 0.2835±0.0740
IsError: 0.5028±0.0815
ErrorSetSize: 1.1443±0.2128
Ranking Loss: 0.1633±0.0320
Mean Average Precision: 0.7271±0.0426
Micro-averaged AUC: 0.8590±0.0235
Macro-averaged AUC: 0.8766±0.0311
```

P003: Integrity Attributes

```
RaKEL-------------------------------

Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.2100±0.0455
Subset Accuracy: 0.2498±0.0496
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6263±0.0401
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5198±0.0531
Micro-averaged Precision: 0.6572±0.0527
Micro-averaged Recall: 0.6260±0.0548
Micro-averaged F-Measure: 0.6407±0.0501
Macro-averaged Precision: 0.6509±0.0600
Macro-averaged Recall: 0.6144±0.0538
Macro-averaged F-Measure: 0.6230±0.0565
Average Precision: 0.7762±0.0405
Coverage: 1.9414±0.1698
OneError: 0.3120±0.0722
IsError: 0.5297±0.0654
ErrorSetSize: 1.3575±0.2418
Ranking Loss: 0.1891±0.0375
Mean Average Precision: 0.6928±0.0515
Micro-averaged AUC: 0.8269±0.0233
Macro-averaged AUC: 0.8264±0.0265

---------MLkNN--------------------------------------------------
Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.1951±0.0243
Subset Accuracy: 0.2831±0.0538
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6050±0.0578
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5326±0.0515
Micro-averaged Precision: 0.7242±0.0571
Micro-averaged Recall: 0.6087±0.0505
Micro-averaged F-Measure: 0.6598±0.0423
Macro-averaged Precision: 0.7330±0.0692
Macro-averaged Recall: 0.5922±0.0425
Macro-averaged F-Measure: 0.6243±0.0413
Average Precision: 0.7965±0.0406
Coverage: 1.7884±0.1634
OneError: 0.2835±0.0740
IsError: 0.5028±0.0815
ErrorSetSize: 1.1443±0.2128
Ranking Loss: 0.1633±0.0320
Mean Average Precision: 0.7271±0.0426
Micro-averaged AUC: 0.8590±0.0235
Macro-averaged AUC: 0.7568±0.0301
```

P004: Maintainability Atrribute

```
RaKEL-------------------------------

Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.2178±0.0255
Subset Accuracy: 0.2478±0.0486
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6273±0.0601
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5098±0.0501
Micro-averaged Precision: 0.6572±0.0517
Micro-averaged Recall: 0.6260±0.0548
Micro-averaged F-Measure: 0.6407±0.0501
Macro-averaged Precision: 0.6509±0.0600
```

```
Macro-averaged Recall: 0.6144±0.0538
Macro-averaged F-Measure: 0.6230±0.0565
Average Precision: 0.7762±0.0406
Coverage: 1.9414±0.1698
OneError: 0.3120±0.0724
IsError: 0.5297±0.0689
ErrorSetSize: 1.3575±0.2418
Ranking Loss: 0.1891±0.0359
Mean Average Precision: 0.6918±0.0505
Micro-averaged AUC: 0.8269±0.0261
Macro-averaged AUC: 0.8121±0.0272

---------MLkNN--------------------------------------------------
Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.1951±0.0243
Subset Accuracy: 0.2831±0.0538
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6050±0.0578
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5326±0.0515
Micro-averaged Precision: 0.7242±0.0571
Micro-averaged Recall: 0.6087±0.0505
Micro-averaged F-Measure: 0.6598±0.0423
Macro-averaged Precision: 0.7330±0.0692
Macro-averaged Recall: 0.5922±0.0425
Macro-averaged F-Measure: 0.6243±0.0413
Average Precision: 0.7965±0.0406
Coverage: 1.7884±0.1634
OneError: 0.2835±0.0740
IsError: 0.5028±0.0815
ErrorSetSize: 1.1443±0.2128
Ranking Loss: 0.1633±0.0320
Mean Average Precision: 0.7271±0.0426
Micro-averaged AUC: 0.8590±0.0235
Macro-averaged AUC: 0.9171±0.0301
```

P005: Portability Attributes

```
RaKEL------------------------------

Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.2197±0.0235
Subset Accuracy: 0.2455±0.0476
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6243±0.0401
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5086±0.0451
Micro-averaged Precision: 0.6545±0.0563
Micro-averaged Recall: 0.6261±0.0538
Micro-averaged F-Measure: 0.5407±0.0341
Macro-averaged Precision: 0.6519±0.0610
Macro-averaged Recall: 0.6144±0.0554
Macro-averaged F-Measure: 0.6221±0.0515
Average Precision: 0.7762±0.0477
Coverage: 1.9414±0.1698
OneError: 0.3120±0.0724
IsError: 0.5297±0.0689
ErrorSetSize: 1.3575±0.2418
Ranking Loss: 0.1891±0.0359
Mean Average Precision: 0.6918±0.0505
Micro-averaged AUC: 0.8451±0.0361
Macro-averaged AUC: 0.8564±0.0372

---------MLkNN--------------------------------------------------
Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.1851±0.0245
```

```
Subset Accuracy: 0.2731±0.0638
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6151±0.0599
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5336±0.0532
Micro-averaged Precision: 0.7244±0.0522
Micro-averaged Recall: 0.6057±0.0515
Micro-averaged F-Measure: 0.6598±0.0423
Macro-averaged Precision: 0.7330±0.0692
Macro-averaged Recall: 0.5922±0.0425
Macro-averaged F-Measure: 0.6243±0.0413
Average Precision: 0.7965±0.0416
Coverage: 1.7984±0.1694
OneError: 0.2830±0.0700
IsError: 0.5028±0.0805
ErrorSetSize: 1.1403±0.2108
Ranking Loss: 0.1600±0.0300
Mean Average Precision: 0.7271±0.0426
Micro-averaged AUC: 0.8675±0.0325
Macro-averaged AUC: 0.8453±0.0321
```

P006: Reliability Attribute

```
RaKEL-------------------------------

Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.2238±0.0267
Subset Accuracy: 0.2408±0.0496
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6263±0.0541
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5098±0.0501
Micro-averaged Precision: 0.6562±0.0547
Micro-averaged Recall: 0.6250±0.0558
Micro-averaged F-Measure: 0.6427±0.0201
Macro-averaged Precision: 0.6535±0.0530
Macro-averaged Recall: 0.6874±0.0638
Macro-averaged F-Measure: 0.6440±0.0545
```

```
Average Precision: 0.7766±0.0516
Coverage: 1.9417±0.1568
OneError: 0.3121±0.0721
IsError: 0.5292±0.0611
ErrorSetSize: 1.3565±0.2428
Ranking Loss: 0.1791±0.0349
Mean Average Precision: 0.6928±0.0615
Micro-averaged AUC: 0.8266±0.0263
Macro-averaged AUC: 0.8365±0.0232

---------MLkNN---------------------------------------------------
Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.1751±0.0253
Subset Accuracy: 0.2837±0.0598
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6150±0.0478
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5356±0.0415
Micro-averaged Precision: 0.7246±0.0578
Micro-averaged Recall: 0.6187±0.0532
Micro-averaged F-Measure: 0.6548±0.0453
Macro-averaged Precision: 0.7430±0.0672
Macro-averaged Recall: 0.5322±0.0415
Macro-averaged F-Measure: 0.6253±0.0414
Average Precision: 0.7985±0.0436
Coverage: 1.7884±0.1624
OneError: 0.2935±0.0760
IsError: 0.5028±0.0825
ErrorSetSize: 1.1463±0.2328
Ranking Loss: 0.1663±0.0330
Mean Average Precision: 0.7671±0.0486
Micro-averaged AUC: 0.8270±0.0243
Macro-averaged AUC: 0.7646±0.0501
```

P007: Usability Attribute

```
RaKEL------------------------------

Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.2178±0.0255
Subset Accuracy: 0.2478±0.0486
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6273±0.0601
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5098±0.0501
Micro-averaged Precision: 0.6572±0.0517
Micro-averaged Recall: 0.6260±0.0548
Micro-averaged F-Measure: 0.6407±0.0501
Macro-averaged Precision: 0.6509±0.0600
Macro-averaged Recall: 0.6144±0.0538
Macro-averaged F-Measure: 0.6230±0.0565
Average Precision: 0.7762±0.0406
Coverage: 1.9414±0.1698
OneError: 0.3120±0.0724
IsError: 0.5297±0.0689
ErrorSetSize: 1.3575±0.2418
Ranking Loss: 0.1871±0.0356
Mean Average Precision: 0.6918±0.0505
Micro-averaged AUC: 0.8267±0.0262
Macro-averaged AUC: 0.8131±0.0272

---------MLkNN--------------------------------------------------
Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.1951±0.0243
```

```
Subset Accuracy: 0.2831±0.0538
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6050±0.0578
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5326±0.0515
Micro-averaged Precision: 0.7242±0.0571
Micro-averaged Recall: 0.6087±0.0505
Micro-averaged F-Measure: 0.6598±0.0423
Macro-averaged Precision: 0.7330±0.0692
Macro-averaged Recall: 0.5922±0.0425
Macro-averaged F-Measure: 0.6243±0.0413
Average Precision: 0.7965±0.0406
Coverage: 1.7884±0.1634
OneError: 0.2835±0.0740
IsError: 0.5028±0.0815
ErrorSetSize: 1.1444±0.2127
Ranking Loss: 0.1633±0.0320
Mean Average Precision: 0.7271±0.0426
Micro-averaged AUC: 0.8580±0.0225
Macro-averaged AUC: 0.7616±0.0301
```

P008: User Conformity Attribute

```
RaKEL-------------------------------

Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.2188±0.0265
Subset Accuracy: 0.2448±0.0436
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6263±0.0631
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5168±0.0401
Micro-averaged Precision: 0.6772±0.0417
Micro-averaged Recall: 0.6261±0.0538
Micro-averaged F-Measure: 0.6407±0.0521
Macro-averaged Precision: 0.6589±0.0601
Macro-averaged Recall: 0.6165±0.0543
Macro-averaged F-Measure: 0.62120±0.0515
```

```
Average Precision: 0.7762±0.0406
Coverage: 1.9414±0.1698
OneError: 0.3120±0.0724
IsError: 0.5297±0.0689
ErrorSetSize: 1.3655±0.2318
Ranking Loss: 0.1891±0.0359
Mean Average Precision: 0.6918±0.0505
Micro-averaged AUC: 0.8619±0.0271
Macro-averaged AUC: 0.8174±0.0292

---------MLkNN--------------------------------------------------
Fold 1/10
Fold 2/10
Fold 3/10
Fold 4/10
Fold 5/10
Fold 6/10
Fold 7/10
Fold 8/10
Fold 9/10
Fold 10/10
Hamming Loss: 0.1551±0.0273
Subset Accuracy: 0.2821±0.0528
Example-Based Precision: NaN±NaN
Example-Based Recall: 0.6150±0.0518
Example-Based F Measure: NaN±NaN
Example-Based Accuracy: 0.5336±0.0535
Micro-averaged Precision: 0.7342±0.0561
Micro-averaged Recall: 0.6037±0.0525
Micro-averaged F-Measure: 0.6698±0.0413
Macro-averaged Precision: 0.7230±0.0592
Macro-averaged Recall: 0.5912±0.0415
Macro-averaged F-Measure: 0.6143±0.0411
Average Precision: 0.7925±0.0401
Coverage: 1.7784±0.1644
OneError: 0.2825±0.0710
IsError: 0.5026±0.0812
ErrorSetSize: 1.1353±0.2426
Ranking Loss: 0.1723±0.0540
Mean Average Precision: 0.6371±0.0486
Micro-averaged AUC: 0.8270±0.0335
Macro-averaged AUC: 0.8259±0.0221
```

Appendix C

List of Publication

1. Jamaiah Yahaya**,** Siti Sakira Kamaruddin, Ruzita Ahmad, Aziz Deraman. 2010. Artificial Intelligence Tecniques in Software Quality: A Review. *Proceedings of Social Economic and Information Technology 2010 (SeiT 2010), Hatyai, Thailand, 23-25 Nov 2010, pp.379-387.*

2. Jamaiah Haji Yahaya, Aziz Deraman, Siti Sakira Kamaruddin, Ruzita Ahmad. 2010. Development of an Intelligent Software Quality Model based on Software Behavioural and Human Factor Approach. *The Proceedings of the UK-Malaysian-Ireland Engineering Science Conference 2010 (UMIES 2010), Queen's University, Belfast, June 23-25, 2010.*

3. Jamaiah Haji Yahaya, Aziz Deraman, Siti Sakira Kamaruddin, Ruzita Ahmad. 2010. Intelligent Software Quality Model by Product Quality Approach: A Review. *The Social Economic & Information Technology Seminar (SEiT 2010), Universiti Utara Malaysia, Sintok, November 23-25, 2010.*

4. Ruzita Ahmad, Jamaiah Yahaya, Aziz Deraman, Siti Sakira Kamaruddin. 2011.Intelligent Software Quality Model: The Theoretical Framework. *The Proceedings of the 3rd International Conference on Computing and Informatics, ICOCI 2011, 8-9 June 2011, Bandung Indonesia, pp. 160-166.*

5. Jamaiah Yahaya, Aziz Deraman, Siti Sakira Kamaruddin, Ruzita Ahmad. 2011. Development of a Dynamic and Intelligent Software Quality Model. *The International Conference on Digital Information and Communication Technology, ICIEIS2011, 21-23 June 2011, Berlin Heidelberg, pp. 537-550.*

6. Jamaiah Yahaya, Aziz Deraman, Siti Sakira Kamaruddin, Ruzita Ahmad. 2011. Feature Subset Selection Method For Dynamic Software Quality Assessment. *The5th Malaysian Conference in Software Engineering, MYSEC2011, 13-14 Dec 2011, Johor Malaysia, pp. 304-306.*

7. Jamaiah Yahaya, Aziz Deraman, Siti Sakira Kamaruddin, Ruzita Ahmad. 2012. Filter-Wrapper based Feature Ranking Technique for Dynamic Software Quality Attributes. *TheKnowledge Management International Conference, KMICE2012, 4-6 July 2012, Johor Malaysia, pp. 604-608.*