Best QMEs for measurement of Software quality for SMEs

Bishal Shrestha

Master's Thesis

ITÄ-SUOMEN YLIOPISTO

Faculty of Science and Forestry
School of Computing
March 2016

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu
School of Computing

Bishal Shrestha: Best QMEs for measurement of Software quality for SMEs
Master's Thesis , 51 p.
, Supervisor of the Master's Thesis : Professor Markku Tukiainen, PhD
Instructor of the Master's Thesis : Vesa Tenhunen, MA, PhD student
March 2016

Abstract

This thesis consists of two parts. The first part provides background on topics related to software quality, software quality models, SQuaRE, measurement, SMEs, metrics and QMEs. The second part emphasize on identifying best QMEs for software quality measurement in the context of SMEs.

SQuaRE is relatively new standard and there is almost no literature about QMEs other than ISO/IEC standards. Therefore, a questionnaire was used to identify software metrics that were used, purpose of measuring those metrics, data collected for measurement of the metric, method of data collection, stage of software development where measurement was performed, effort required and usefulness of the metric. The response to questionnaire from each company was analyzed to identify the software properties measured by them. Based on the software properties measured by each company and QME definitions in ISO/IEC 25021, a set of most useful QMEs were recommended to each company. After analysis was completed for all participating companies, a list of most useful QMEs, and best QMEs for software quality measurement for SMEs were selected using selection criteria defined in methodology section.

This thesis emphasizes on software quality, highlights the usefulness of software quality measurement in software development, and provides SMEs with the best QMEs for software quality measurement based on the selection criteria for the companies participating in the study. The selection criteria are explained in methodology section of this thesis.

Keywords: software, quality, quality model, measurement, metrics, QMEs

Preface

I am grateful to University of Eastern Finland for providing me this invaluable opportunity. I am extremely grateful to my Supervisor Prof. Markku Tukiainen (PhD) for providing ISO/IEC documents related with my research topic and for the help, support and supervision. I highly appreciate my instructor Vesa Tenhunen (MA, PhD student) for his help, support, suggestions, guidance and encouragements. My thesis writing would not have been possible without supervision of my supervisor and my instructor. I am thankful to the participating companies for supporting my studies by responding to my questionnaire.

I kindly appreciate all the positive words, feedbacks, love, support and care that I received from my friends and family. Also special thanks to all the people who directly or indirectly helped me during data collection.

Joensuu, March 2016

Bishal Shrestha

List of Abbreviations

ACM	Association for Computing Machinery
ISY	Itä-Suomen yliopisto
UEF	University of Eastern Finland
SQO-OSS	Software Quality Observatory for Open Source Software
QualOSS	Quality of Open Source Software
FURPS	Functionality Usability Reliability Performance Supportability
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
IS	International Standard
TR	Technical Report
SQuaRE	Software Product Quality Requirements and Evaluation
LOC	Lines of code
SME	Small and Medium Enterprise
QME	Quality Measure Element
QM	Quality Measure
I/O	Input and/or Output
AWU	Annual Work Unit
OECD	Organization for Economic Co-operation and Development

Contents

1 Introduction **1**

1.1 Small and Medium-sized Enterprises . . . 2

1.2 Purpose . . . 3

1.3 Research question . . . 3

1.4 Structure of thesis . . . 3

2 Software Quality **5**

2.1 Quality . . . 5

2.2 Software Quality . . . 6

2.3 Importance of Software Quality . . . 8

2.4 Cost of Software Quality . . . 8

2.5 Software Quality models . . . 9

2.5.1 McCall's Quality Model . . . 10

2.5.2 Boehm's Quality Model . . . 11

2.5.3 FURPS . . . 11

2.5.4 Dromey's Quality Model . . . 12

2.5.5 ISO/IEC 9126 . . . 12

2.5.6 ISO/IEC 25010 . . . 14

2.6 ISO/IEC 25000 . . . 15

3 Software Measurement and Metrics **17**

3.1 Software Measurement . . . 17

3.2 Importance of Measurement . . . 18

3.3 Software metrics . . . 19

3.4 Quality measure . . . 20

3.5 Quality measure element (QME) . . . 20

3.6 Quality measure elements categories . . . 22

3.6.1 Data size . . . 23

3.6.2 Number of Data items . . . 23

3.6.3 Number of failures . . . 24

3.6.4 Number of faults . . . 24

3.6.5 Number of functions . . . 24

3.6.6 Number of I/O . . . 25

3.6.7 Number of requirements . . . 26

3.6.8 Number of restarts 26
3.6.9 Number of system operations 26
3.6.10 Number of tasks 27
3.6.11 Number of test cases 27
3.6.12 Number of trials 27
3.6.13 Number of user operations 28
3.6.14 Product size 28
3.6.15 Time duration 28

4 Methodology 30
4.1 Response to questionnaire 30
4.2 Structure of questionnaire 31
4.2.1 Name, Email and Company name 31
4.2.2 Number of employees 31
4.2.3 Name of metric 31
4.2.4 Purpose of using metrics 31
4.2.5 Data collected 32
4.2.6 Method of data collection 32
4.2.7 Stage of software development 32
4.2.8 Effort [Effort needed to measure the metric] 32
4.2.9 Usefulness of metric 32

5 Results and conclusions 35
5.1 Data analysis method 35
5.2 Problems encountered in data analysis 36
5.3 Company A 36
5.4 Company B 40
5.5 Company C 44
5.6 Conclusion 47

6 Limitations and further studies 51

References

List of Figures

1 ISO/IEC 9126 External and Internal Quality Model 13

2 ISO/IEC 9126 Quality in use Model 13

3 Relationship between Internal, External and Quality in use attributes . 14

4 ISO/IEC 25010 Software product quality model 14

5 Quality model for quality in use 15

6 Relationship between property to quantify, measurement method, QME and QM. 22

7 Questionnaire as excel sheet 33

8 Page 1 of questionnaire in web form 34

9 Page 2 of questionnaire in web form 34

List of Tables

1 QME categories and QMEs for Company A 37
2 Sorted QMEs on the basis of their usefulness to Company A 38
3 QME categories and QMEs for Company B 41
4 Sorted QMEs on the basis of their usefulness to Company B 42
5 QME categories and QMEs for Company C 45
6 Sorted QMEs on the basis of their usefulness to Company C 46
7 Common QMEs in participating companies 48

1 Introduction

Software products are used in large variety of application areas. The growth in dependence on software for normal work to life critical systems is increasing everyday, and there is need for highly reliable, secure, efficient and user-friendly software. The rise in size and complexity of software resulted in more emphasis on software quality to minimize cost of production, maximize efficiency of software development process, and to specify and evaluate requirements for software. Software quality is one of the most important research field that has value for all the stakeholders involved in the software project.

All the software companies work independently or collaborate to fulfil the demands of the market. The main goal of software companies is to deliver functioning, highly reliable, defect free, secure and efficient software products that satisfy customers need. Large companies have enough resources and expertise for performing research on software process and product quality. Small and medium-sized companies are restricted with limited resources and expertise to research on software development process and software measurement. Most of the projects for small and medium-sized companies are of short to medium duration therefore there is always constraint of money and time, besides expertise needed for research. All the software companies are working for maximizing their revenue by minimizing resources used for software product development without compromising with the requirements of the customers and the quality standards specified by the customers. Therefore understanding, measuring, analyzing and managing software quality is of prime importance for software companies to have better understanding of software products and software development processes. Without evaluating software quality, a software company cannot provide quality assurance to the customers. Also understanding of software quality helps customers to clearly specify software requirements and evaluate the software quality.

There are different software quality models purposed to describe software quality in abstract level. ISO/IEC provided standard guidelines for defining and measuring software quality. This thesis studies about the software quality measurement performed in the participating small and medium-sized companies to identify the software properties measured by them. Then those software properties were associated with quality measure elements to identify what quality measure elements the company used. After analyzing all QMEs for each company, this thesis points out the most important

QMEs for small and medium-sized companies that were based on data collected from the participating companies. Using the selection criteria described in the data analysis method, the best QMEs for measurement of software quality for SMEs were selected.

1.1 Small and Medium-sized Enterprises

Small and medium-sized enterprises are companies with financial insecurity, limited resources and they lack expertise in software development, testing and quality assurance [1]. The European Commission defined enterprise as "Any entity engaged in economic activity, irrespective of its legal form". Economic activities include manufacture of goods, distribution and consumption of goods and services in the market. SMEs have an important role in world economy [2, 1]. Most countries have their own definition of SME according to their requirements [3]. In 1996, the European Commission adopted a common definition of SME and implemented it in the European Union [3, 4]. On 6th May 2003, the European Commission adopted a new definition of SME with some changes to the financial threshold [3, 4]. According to that definition, the categories of small and medium enterprises is given in table below [3, 4]:

Enterprise category	Headcount: Annual Work Unit (AWU)	Annual turnover	Total annual balance sheet
Micro	Less than 10	Less than 2 million euros	Less than 2 million euros
Small	between 10 and 49	Up to 10 million euros	Up to 10 million euros
Medium-sized	between 50 and 250	Up to 50 million euros	less than equal to 43 million euros

In the European Union, SMEs are the enterprises with less than 250 employees and annual turnover of less than 50 million euros or annual balance sheet of less than 43 million euros [3, 4]. Almost 98 percent of the enterprises in the European Union are SMEs [3]. In OECD countries, SMEs constitute over 95 percent of the enterprises and accounts for 60-70 percent of jobs in most countries and contribute in economy [**?**]. Most of the large enterprises were once SMEs [3].

Since the late 1970s, the availability of cheaper micro-computers have contributed to the rapid expansion of software industry [5]. This resulted in brisk increase in number of software companies, software size and number of software product [5]. The software market is huge and growing. The rapid expansion of software market resulted in rise of many micro, small and mid-sized software companies. Small and medium-sized enterprises are dominant in software industry [2]. In many countries like Finland, United States, Brazil, Canada, China, India, Ireland, Hungary, etc, small companies constitute up to 85 percent of the software industry and are responsible for majority of job creation [6]. Irrespective to the size of software companies, their main goal is to survive in the market and make profit by developing high quality software products and services within the constraints of available resources [1, 7]. The budget, resources and expertise required for measuring, improving and assuring software process and product quality are constrained by limited resources available to SMEs [6]. SMEs with limited expertise, budget and resources also have to assure high quality softwares and services to survive in competitive market [1, 7].

1.2 Purpose

The purpose of this thesis was to identify the best QMEs for measurement of software quality in context of SMEs by using ISO/IEC standards.

1.3 Research question

Following research questions are answered in this thesis:

1. What are the most important QMEs for software quality measurement in SMEs?
2. What are the best QMEs for software quality measurement in SMEs?

1.4 Structure of thesis

This thesis consists of six chapters. First chapter is introductory chapter about the research topic, purpose of thesis, research questions and structure of thesis. It gives overview of the thesis. Second chapter provides background for the thesis topic. In this chapter, topics such as quality in general, software quality, importance of software quality, cost of software quality, software quality models, SQuaRE series of standards and the structure of SQuaRE are discussed. Third chapter provides information about software measurement, importance of measurement, software metrics, quality measure and quality measure element and quality measure elements categories. Fourth chapter is the methodology section of the thesis. This chapter discusses about the criteria used for selecting method for analyzing data, structure of questionnaire and response from companies to the questionnaire. Fifth chapter provides the information about participating companies, data collected from them, data analysis technique used for each company, results of analysis for each company and final conclusion. Sixth chapter discusses about limitations of the study and possibilities of further studies.

2 Software Quality

Software does not have physical form, therefore it is difficult to understand, visualize, define and evaluate software quality by simpler means. To visualize software quality, quality in general should be understood.

2.1 Quality

Quality is a complex, multidimensional and dynamic concept which has different meaning in different context or in different perspective [8, 9, 10, 11, 12]. Greek philosophers including Socrates and Plato have defined quality as excellence [13]. This definition does not give quantitative definition of quality so measuring quality becomes subjective [13]. Also attaining excellence requires more resources which may exceed human and nonhuman resources allocated to the project [13]. Customers generally desire products with acceptable quality and performance at affordable price rather than absolute excellence [14]. W. Edwards Deming emphasizes customer satisfaction while defining quality [15, 9]. Joseph Juran [9, 11] defines quality as "fitness for use" and asserts planning, controlling and improving quality as the responsibility of management team. Juran suggests that a software product without any defect will reduce cost, satisfy customer's need and create customer enthusiasm about the product [16, 11].

Feigenbaum definition of quality [15]: "Quality is a customer determination, not an engineer's determination, not a marketing determination, nor a general management determination. It is based on upon the customer's actual experience with the product or service, measured against his or her requirements - stated or unstated, conscious or merely sensed, technically operational or entirely subjective - and always representing a moving target in a competitive market. Product and service quality can be defined as: The total composite product and service characteristics of marketing, engineering, manufacture and maintenance though witch the product and service in use will meet the expectations of the customer".

According to Software Quality Theory and Management book, ISO 9001:2008 perspective on quality [14]: "The quality of something can be determined by comparing a set of inherent characteristics with a set of requirements. If those inherent characteristics meet all requirements, high or excellent quality is achieved. If those characteristics do not meet all requirements, a low or poor level of quality is achieved".

There are always constraints (budget, time and other human and non-human resources) to quality due to which there is trade-off between quality and the cost of quality [14]. The value that product or services provide to the customer should always be more than the cost of the product or services for customer's satisfaction [14, 13]. Therefore, from the business point of view, required quality should be specified by customers and managers should evaluate, control and manage the development or manufacturing process so that the product is of desired quality and of value to customers [14, 15]. Conformance to customer's early specifications cannot always guarantee high quality products and services [13].

2.2 Software Quality

A software is a multifaceted and intangible product that does not have a physical form. Each software project is unique with characteristics like unpredictability, low repeatability and intangibility [13]. Therefore it is difficult to visualize, define or evaluate the complexity of designing and developing software product that fulfils all the requirements of the user and guarantee desired software quality [10]. This makes software products different from industrial products. Better process quality guarantees better product quality in industrial products [10, 14]. Good process quality enhances software quality but it cannot completely guarantee superior product quality [10, 14, 17]. For achieving business goals, it is more important to understand, measure, assess and control software product quality for software quality assurance and customer's satisfaction [8, 17, 10]. The changing client requirements with time demand all software products to evolve and adapt over their life time so as to serve the purpose and need of client [18]. The longevity and usefulness of software to the users depend on controlling the complexity of design and development of software [18]. The complexity and size of software products are growing. The customer needs are also growing and there is constant pressure to deliver a product of required quality within specific time duration using less effort. Therefore, there is need for defining, measuring, understanding, analyzing and controlling software quality during software development for quality assurance, increasing productivity and customer's satisfaction.

Software quality has different definitions when seen from different perspective. Philip

Crosby advocates for zero defects as performance standard, and defines software quality as the compliance of software products and services to fully understood specifications presented by the customer [9, 10, 11]. Garvin defines quality from a set of different views, namely transcendental approach, product-based approach, user-based approach, manufacturing approach and value-based approach [12, 17, 19]. Transcendental approach is based on intuition, and it states quality as something that can be felt but cannot be defined absolutely [12, 17, 19]. It refers to quality as innate excellence and argues that quality should have uncompromising standards [12]. The product-based approach identifies quality as a explicit and measurable variable [12]. It emphasizes the use of proper metrics for measuring internal quality indicators and comparing the results with desired values of attributes to assure good quality [17, 19]. This approach views quality as the fundamental attribute of goods and states that high quality goods are expensive to produce [12]. User-based approach views quality from the perspective of customers and relates quality with customer's satisfaction [12]. It identifies software quality as the fitness of a product for their purpose of meeting or exceeding user's expectation [12, 17, 19]. This gives a highly subjective definition of quality that relies heavily on customer's personal preference [12]. Manufacturing approach views quality as compliance to customer's specifications and focuses on developing well-defined specifications and improving manufacturing process to produce good quality products at low cost [12, 17, 19]. Value-based approach defines quality as the value that customers gain from product at a tolerable cost [12]. It articulates the fact that views from different stakeholder may conflict, therefore it advocates evaluation of potential benefits by comparing product quality with the cost of quality to determine value of the product [12, 17, 19]. Garvin's approaches of quality should be applied to the most relevant stages of the software development life cycle for producing better quality softwares and services [12, 17, 19].

US Department of Defense [14] defines software quality as "The degree to which the attributes of the software enable it to perform its intended end use".

Hansen Pressman's definition of software quality [20]: "Conformance to explicitly defined functional and implicit characteristics that are expected of professionally developed software".

ISO/IEC 25010:2011 [21] defines software quality as "degree to which the software product satisfies stated and implied needs when used under specified conditions".

The above definitions can be summarized to define software quality as the degree of compliance of characteristics of error free software product with the functional, non-functional and implicit specifications of customers and satisfying expectations of customers under the constraints of resources allocated for the software [19, 20, 17, 10, 21]. Software quality depends on software development processes quality, product quality, service, information, people and system [19, 20, 17, 10, 21].

2.3 Importance of Software Quality

Today's world activities are supported and/or controlled by the reliable functioning of large and complex software-driven computer systems. Computer systems are progressively used in a large variety of application areas such as schools, banks, hospitals, military, business firms, airplanes, automobiles, air traffic control, factories, power plants, etc for performing daily activities related to keeping and updating records, performing calculations, analyzing and solving problems, decision-making, messaging, etc. Software systems have significant impact in our life and our dependence on software products and services are increasing day by day. Therefore, there is need of a safe, efficient and highly reliable softwares. Software quality is one of the decisive factor that has influence on success and competitiveness of a software product [22, 16, 20, 23]. Therefore it is necessary to develop or select high quality software products that provides value to the customer and satisfies customer's need. Measuring quality of a software product will allow all the stakeholders check if the product has met the quality standards set by them. A poor quality product is costly as it causes inefficiencies, low productivity, waste of resources such as time, money, effort, etc, reduce customer's allegiance, physical harm, erroneous analysis and bad decisions, security flaws and may result in product failure [16, 20, 23]. On the other hand, high quality products satisfy customers and will increase efficiency, profitability, productivity and decrease manufacturing cost in long run [12].

2.4 Cost of Software Quality

Software is imperceptible so the software quality requirements should be measurable and clearly defined [10]. Software quality cost is defined as the extra expenditure on the software product than the cost that would incur in absence of any imperfections or defects in the software product [12]. For achieving business goals, the quality goals should be achievable within the constraints of budget, time and available resources [22]. The cost of quality should be identified, measured, analyzed and controlled to meet the customer's requirements at the lowest cost [22]. Inadequacy in software quality results in extra cost for both users and the software company [24].

2.5 Software Quality models

There are several quality models and standards in the software engineering literature [24]. Software quality models refine required software quality into a set of characteristics and subcharacteristics, and clarify the relationships between them [25, 26, 27]. A software quality model provides a basis for specifying quality requirements and assessing quality of software [27]. They are developed with a purpose to understand, specify, assess and/or predict quantitative and qualitative qualities of different types of general and specific software to satisfy all the stakeholders of the software product [28, 14, 17, 24]. Quality models are helpful to set quality goals to support quality management for a software product [29, 14]. ISO/IEC 25010:2011 [21] states "A software quality model consists of measurable characteristics and subcharacteristics that provide consistent terminology for specifying, measuring, and evaluating system and software product quality". Quality model is used to assess the completeness of stated and implied quality specifications from perspective of various stakeholders [21]. Basic software quality models are mostly hierarchical consisting of a number of factors or criteria where each quality criteria have a set of measures or metrics associated with them [14, 24, 27].

The evolution of software quality models started from 1970's with McCall's quality model in 1977 [30, 14, 24, 27]. Since McCall's quality model presented in 1977, many other quality models have been purposed [27]. These models can be categorized into basic quality models and tailored quality models [27]. Basic quality models (1977 - 2001) are general quality model that provides total and comprehensive product evaluation [27]. Tailored quality models started to appear from 2001 onwards to respond to the need of software industries for developing quality models capable of evaluating individual components [27]. They are based on basic quality models with some modifications and they emphasize on the features that are more specific to specialized application or domain according to needs software organizations [27]. Some of the tailored quality models are: Bertoa model (2001), Georgiadou model (2003), Alvaro Model (2005), Rawashdesh Model (2006), Andreu Model (2007), SQO-OSS model (2008), QualOSS model (2009), Al-Badareen model (2012), Quamoco model (2012), Midas model (2013), etc [27].

The most important basic quality models are discussed in the following subsection.

2.5.1 McCall's Quality Model

McCall's quality model, presented in 1977 by Jim McCall et al., is one of the earliest quality models that defines software quality subjectively and quantitatively. This model considers both user's perspective and developer's preference while defining software quality characteristics and defines software quality from three perspectives: product revision, product transition and product operations. The product revision view can be defined as the ability of a software product to adapt to changes in the system and software requirements, correct error and function correctly as mentioned in the software specification. Maintainability, flexibility and testability are included in this category. The product transition perspective emphasizes the adaptability of software to changing operating environment of the software and changes in the computer hardware. Portability, reusability and interoperability fall under this category. The product operations category considers user's view of quality and focuses on operational characteristics of software. This group consists of correctness, reliability, efficiency, integrity and usability. This model describes software quality from three perspectives that consists of a hierarchy of 11 quality factors to describe the behaviour of software product from user's perspective, which are simplified into quality criteria to describe the internal view of the software and metrics to evaluate those quality criteria [30, 14, 15, 28, 31, 19, 32, 33].

McCall model considers general application systems but it has some shortcomings. This model defines software quality in terms of quality factors, but the correlation between those quality factors is not specified. The combination of these quality factors gives overall software product quality. There are quality factors like usability and efficiency which has inverse correlation so there must be a compromise [14]. The quality factors does not include analyzability but the criteria like simplicity and modularity are connected to analyzability. On the other hand, architectural integrity and domain-specific characteristics are ignored in this model [31, 33].

2.5.2 Boehm's Quality Model

Barry W. Boehm introduced a hierarchical quality model in 1978 for qualitative evaluation and analysis of software quality from utility perspective on quality characteristics. Top of the hierarchy has three fundamental high level software characteristics: as-is utility, maintainability and portability. The intermediate-level hierarchy consists of 7 quality factors, namely portability, reliability, efficiency, human engineering, testability, understandablity and modifiability. Portability is further classified into device independence and self-containedness. Reliability includes accuracy, completeness, robustness/integrity and consistency. Efficiency includes accountability, device efficiency and accessibility. Human engineering consists of robustness/integrity, accessibility and communicativeness. Testability includes accountability, accessibility, communicativeness, self-descriptiveness and structuredness. Understandability contains self-descriptiveness, consistency, structuredness, conciseness and legibility. Modifiability contains structuredness and augmentability [34, 28, 15, 32, 31].

Boehm mentions that the complex and invisible nature of software restricts the capability to automatically and quantitatively determine the software quality. Customers face difficulty for prioritizing and quantifying preferences in conflicting situations between individual characteristics of a software product. Therefore, this model presents a framework to qualitatively define, measure and evaluate software quality. Similar to McCall's model, architectural integrity and domain specific properties are excluded and understandability somehow includes analyzability [34, 28, 15, 32, 31].

2.5.3 FURPS

Robert Grady developed FURPS model in 1992 emphasizing functional and nonfunctional requirements. There are 5 characteristics, namely functionality, usability, reliability, performance and supportability. The FURPS model prioritizes user requirements over developer's preferences. This model also fails to incorporate architectural integrity, domain specific properties, portability and maintainability [28, 15, 32, 31].

2.5.4 Dromey's Quality Model

R. Geoff Dromey in 1995 extended ISO/IEC 9126:1991 and presented an empirical approach for defining quality by correlating tangible product attributes with less tangible product properties [35, 31]. This model provides a basis for determining requirements and assists in design and implementation phases [31]. It consists of four software product properties, namely correctness, internal, contextual and descriptive. Correctness addresses the functionality and reliability of software. Internal emphasizes design factors like maintainability, efficiency and reliability. Contextual considers the external impacts while using software and has maintainability, reusability, portability and reliability as quality attributes. Descriptive evaluates the documentation of software component and has quality attributes maintainability, reusability, portability and usability. Architectural design and analyzability are not prioritized whereas extensibility and domain-specific properties are ignored. Reliability and Maintainability cannot be evaluated before the product is actually developed [35, 31, 15, 32, 33].

2.5.5 ISO/IEC 9126

The increase in number of quality models made it difficult for specifying, measuring and evaluating software quality [32, 15, 33, 36]. Therefore, there was need for the standardization of quality model. In 1991, based on international consensus on the terminology for quality attributes for software product, ISO developed a standard software quality model called ISO/IEC 9126 for evaluating software product quality characteristics with guidelines for their use. This standard is comparable to the McCall and Boehm models. It provides a framework for specifying and evaluating a software product in terms of its internal and external software characteristics [32, 15, 33, 36]. The current version of ISO 9126 series includes one international standard and three technical reports [32, 36]. They are listed below:

1.	ISO/IEC IS 9126-1:	Quality Model[ISO, 2001]
2.	ISO/IEC TR 9126-2:	External Quality Metrics[ISO, 2003]
3.	ISO/IEC TR 9126-3:	Internal Quality Metrics[ISO, 2003]
4.	ISO/IEC TR 9126-4:	Quality in Use Metrics[ISO, 2004]

ISO/IEC 9126 quality model comprises of external and internal quality model and quality in use model [32, 36]. The internal and external quality model emphasizes on developer's viewpoint while the quality in use model focuses on customer's viewpoint of the software product. The characteristics and subcharacteristics of external and internal quality model are shown in figure 1 [32, 36].

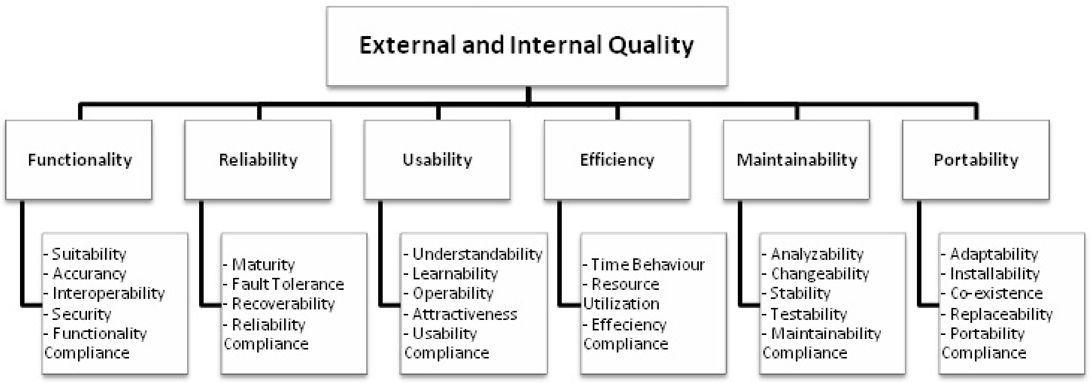


Figure 1: ISO/IEC 9126 External and Internal Quality Model

The quality in use model is shown in figure 2 [32, 36].

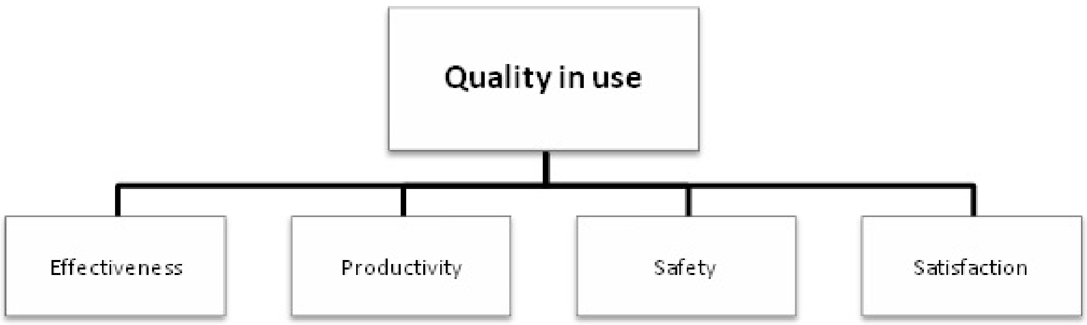


Figure 2: ISO/IEC 9126 Quality in use Model

The relationship between internal, external and quality in use attributes is shown in figure 3 [32, 36].

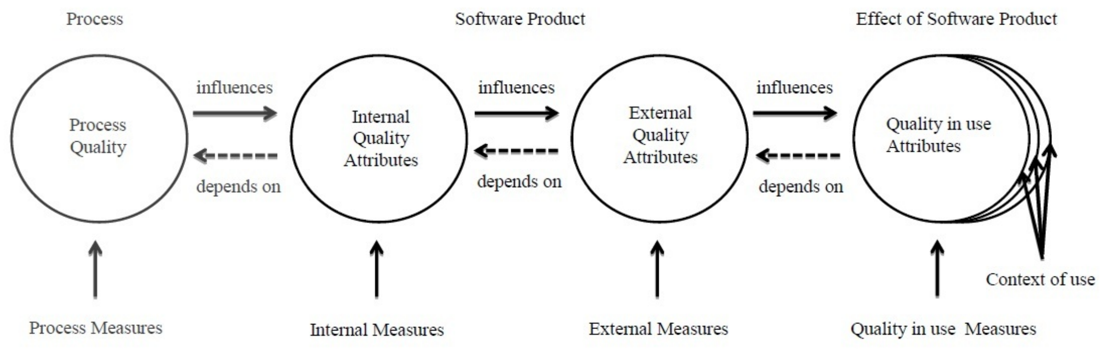


Figure 3: Relationship between Internal, External and Quality in use attributes

ISO/IEC 9126 is extensively promoted, referred and utilized. Analogous to other models, ISO/IEC 9126 lacks subcharacteristics of evolvability like architectural integrity and extensibility whereas domain specific characteristics are ignored [33, 25]. The quality requirement standard is excluded in ISO/IEC 9126 series, causing problems in quality evaluation [33, 25].

2.5.6 ISO/IEC 25010

ISO/IEC 25010 is the newest quality model that replaces ISO/IEC 9126 standard [24, 27]. It is an update to ISO/IEC 9126 standards [27]. It has 8 subcharacteristics and has minor changes with respect to ISO/IEC 9126 standard [27]. The software product quality as stated by ISO/IEC 25010 is shown in figure 4 [37], and the quality in use model is shown in figure 5 [37].

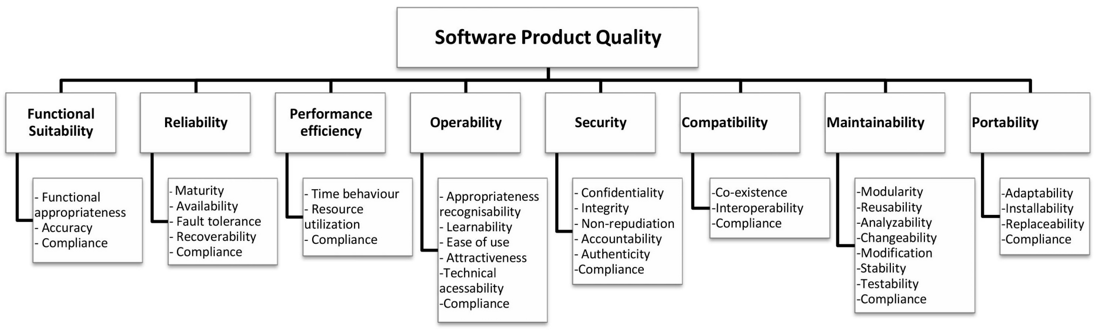


Figure 4: ISO/IEC 25010 Software product quality model

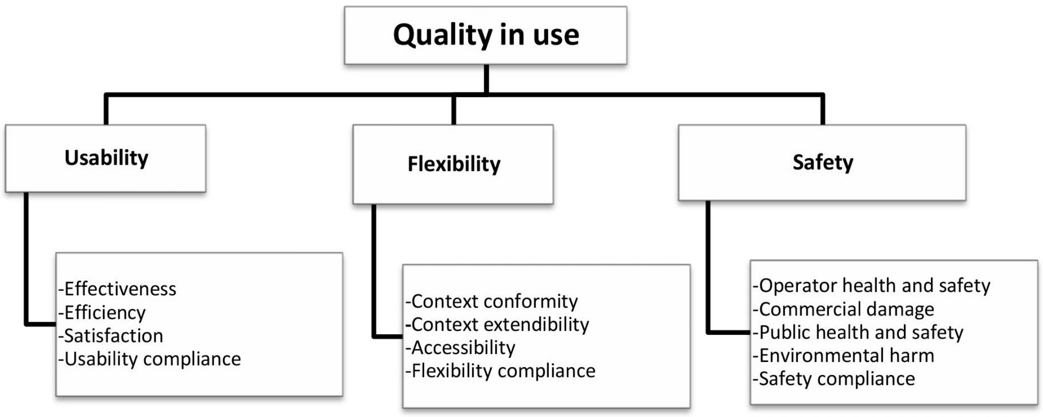


Figure 5: Quality model for quality in use

2.6 ISO/IEC 25000

ISO/IEC 25000, also known as Software Product Quality Requirements and Evaluation (SQuaRE), is a series of standards that emphasizes on the product perspective of software quality assurance [25, 26]. This series is the successor of ISO/IEC 9126 series and ISO/IEC 14598 standards [36, 14, 26]. They try to reconcile ISO/IEC 9126 and ISO/IEC 14598 (ISO standard for software process evaluation) [36, 14, 26]. SQuaRE is dedicated for helping software developers, acquirers and evaluators from perspective of software product [25, 26]. This series comprises five divisions [36, 14, 26]:

1. ISO/IEC 2500n on Quality Management.
2. ISO/IEC 2501n on Quality Model.
3. ISO/IEC 2502n on Quality Measurement.
4. ISO/IEC 2503n on Quality Requirements.
5. ISO/IEC 2504n on Quality Evaluation.

The quality management division has standards to define all SQuaRE models, terms and definitions that are specified by other standards from SQuaRE series [25, 26]. This division provides prominent guidance and practical suggestion for users through SQuaRE documents for managing technologies needed for using SQuaRE [25, 26].

Evaluation group that manages and evaluates requirements specification of software product quality requirements and product quality can refer to requirements and guidance from this division [25, 26].

The quality model division comprises standard for the quality model and guide for customizing and applying the model to individual product. The quality model consists of characteristics and subcharacteristics for internal and external quality model and quality in use model [25, 26].

The quality measurement division has standards for software product quality measurement using reference model, internal metrics, external metrics, and quality in use metrics [25, 26]. It provides guidance for choosing or developing, and using quality metrics [25, 26].

The quality requirements division includes standard for specifying software product quality specifications and a guide for using the model and metrics for defining requirements [25, 26].

The quality evaluation division consists of standards that present specifications, suggestions and guidelines for evaluating software product quality. This division also assists in software product measurement and evaluation [25, 26].

3 Software Measurement and Metrics

Software measurement and metrics are very important for quantifying the attributes of a software product.

3.1 Software Measurement

Measurement is the process of characterizing attributes of entities in the real world with numbers or symbols by using clearly defined measurement rules that are derived from a model or theory to make them more comprehensible, comparable and controllable [38, 39, 23]. It involves budget, effort, resources and technical expertise [38, 39]. The results of measurement should be analyzed to evaluate the accomplishment of measurement goals and necessary corrective actions should be taken for improvement [38]. Collecting necessary data to satisfy measurement goals, and evaluating well-defined data provides organization with valuable information which can be interpreted to make improvement decisions for better productivity and superior product and process quality [38]. Measurement is valueless, if it is limited to data collection only [38, 39]. The quality of collected data and the measurement process also affects the result of measurement. Collecting data for measuring all the attributes of an entity waste valuable time and resources [38]. Therefore, measurement should be goal driven and only necessary data should be collected and analyzed [38, 40]. More emphasis should be given to extract most information about properties of entities with less measurement effort [41].

Software measurement is a software engineering field that is related to quantification of software attributes related to the product, the process and allocated resources using clearly defined measurement process for better understanding of effectiveness of methods and tools used in software engineering and customizing the techniques and tools to achieve the goals of the software project [41, 23]. ISO/IEC 15939:2001 mentions that software measurement should support and facilitate software project management and quality management [42, 41]. In quality management, software measurement assists in developing better software products, processes and quality model by providing data required to evaluate software [41]. From project management perspective, software measurement provides a standard for clearly defining software requirements, collecting, analyzing and evaluating the quality of software development process and product throughout the software project to achieve the project objectives [42, 41, 23]. Software measurement is used to predict software cost, software size, effort for testing and maintenance early in the software life cycle, therefore it helps to manage and control software development within the constraints of allocated resources [38, 41, 23]. From developer's perspective, software measurement provides information about fault tolerance, testability of requirements, quality of software product and deviations from product or process goals [38, 41, 23].

ISO/IEC 15939 suggests that the purpose of software measurement is to provide a support for understanding, planning, controlling and improving the software development process and product by collecting, analyzing and reporting information needed for evaluating and managing activities of the software life cycle [41, 42, 23]. Software measurement also illustrates the quality of software product and provides a basis for negotiation between software organization and customer for software requirements and baseline for acceptance criteria [41, 42, 23].

3.2 Importance of Measurement

Software measurement has a fundamental role in software engineering [38, 23, 41]. Software development is a complex process and it requires human intelligence [38, 41]. The main goal of software companies is to supply customers with better quality softwares and services within the agreed delivery time by using least possible resources [38, 41, 40]. Therefore, there is scope for continuous improvement for software process and product quality. Software measurement facilitates companies by providing information needed to understand, evaluate, control and predict the performance of product or project to achieve project goals and business goals [40].

Software projects are unique and have characteristics like low repeatability, dynamics, intangibility and conflicting interests [13]. It is essential for understanding, evaluating, managing and enhancing process and product quality [38, 41]. It provides necessary information for clearly identifying, specifying and evaluating software requirements, controlling and assuring software product or process quality, and it also provides information to facilitate resource management [38, 41]. It is the heart of improvement program in software engineering because without measurement the product or process quality cannot be compared with standard and the scope of improvement cannot be identified [38, 41, 40, 13].

From business perspective, software measurement is very important [40, 23]. Senior managers need information from software measurement for measuring performance, managing resources effectively, identifying and forecasting areas of improvement [40]. Project managers use software measurement data for reviewing project, predicting and managing quality, schedule and budget and checking follow-up of action points [40, 38, 23]. Engineers and developers needs software measurement for team planning, checking progress, identifying shortcomings in deliverables, evaluating and improving their performance [38, 40, 23].

3.3 Software metrics

Software metrics are one of the very important research area in software engineering [25]. A software metric is a scale with measurement rule and measurement method that is applied for a measurement process [25]. According to McCall, metrics are objective quantitative measures of software attributes that help to understand software quality [30]. Software metrics are the units of measurement for measuring and predicting the quality of the software products or processes [43, 44]. They help management to assess cost, effort, quality and complexity of software projects at various stages of the software development life cycle [43, 38]. They can also measure productivity and customer's satisfaction [43, 38]. Object-oriented metrics are used to measure object-oriented concepts such as cohesion, coupling, inheritance and polymorphism [43, 38]. Goodman [43] definition of software metrics can be stated as "The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products". IEEE [45] defined software metrics as "A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality".

Software metric provides value to the measured entity but it do not indicate anything about the result of measurement [18]. Therefore, a threshold should always be associated with metrics to provide meaningful information that supports evaluation or comparison of the metric [18]. Software metrics are classified as product metrics, process metrics and project metrics [8, 46]. Product metrics emphasize on product aspects of software quality such as size, complexity design features, performance etc [8, 46]. They help to assess and control the product quality [8, 46]. Process metrics emphasizes on software development process and they are used to assess, improve, predict and control the software development procedures [8, 46]. Examples of process metrics are the effectiveness of defect removal during software development, the pattern of defect arrival during testing etc. Project metrics emphasizes on project characteristics and execution [8, 46]. Examples of project metrics are cost, schedule, productivity, staffing pattern during software development etc [8, 46].

3.4 Quality measure

ISO/IEC 25000 defines quality measure as [26] "a measure of internal software quality, external software quality and quality in use that are described in IS0/IEC 25010". Here the term "measure" indicates a variable to which value is assigned as a result of the measurement [26]. It helps us to understand the characteristics and subcharacteristics of a quality model [47]. Quality measure is obtained when we apply a measurement function to one or more quality measure elements [47].

3.5 Quality measure element (QME)

ISO/IEC 25021 defines quality measure element as [47] "a measure defined in terms of a property and the measurement method for quantifying it, including optionally the transformation by a mathematical function". It may be a base measure or derived measure [47]. Quality measure elements are input for the measurement of the software quality measures that correspond to external quality, internal quality and quality in use as described in ISO/IEC 25010 quality model [47, 48].

ISO/IEC 25021 states that quality measure element is achieved when we apply a suitable measurement method to quantifiable property of the target entity [47, 48]. ISO/IEC 25021 defines target entity as [47] "fundamental thing of relevance to the user, about which information is kept, and need to be measured". Quality measure elements are used at any stage during entire software product life cycle to measure different attributes of a software product or process [47]. A quality measure element can be used independently or combined with another quality measure element to measure different quality measures [47]. ISO/IEC 25021 defines a set of rules and guidelines for companies to develop and implement their own QMEs in addition to an initial set of quality measure elements [47].

Quality measure elements are used during software product life cycle to measure the following attributes [49]:

1. Attributes of consumed resources, or activities related to the software product quality during software development, testing, and maintenance.
2. Attributes of the software product.
3. Attributes of the specific context of use of the software product.
4. Attributes of the software product while users are using the product in a specific context.

The relationship between property to quantify, measurement method, quality measure element and quality measure is given in figure 6 [47].

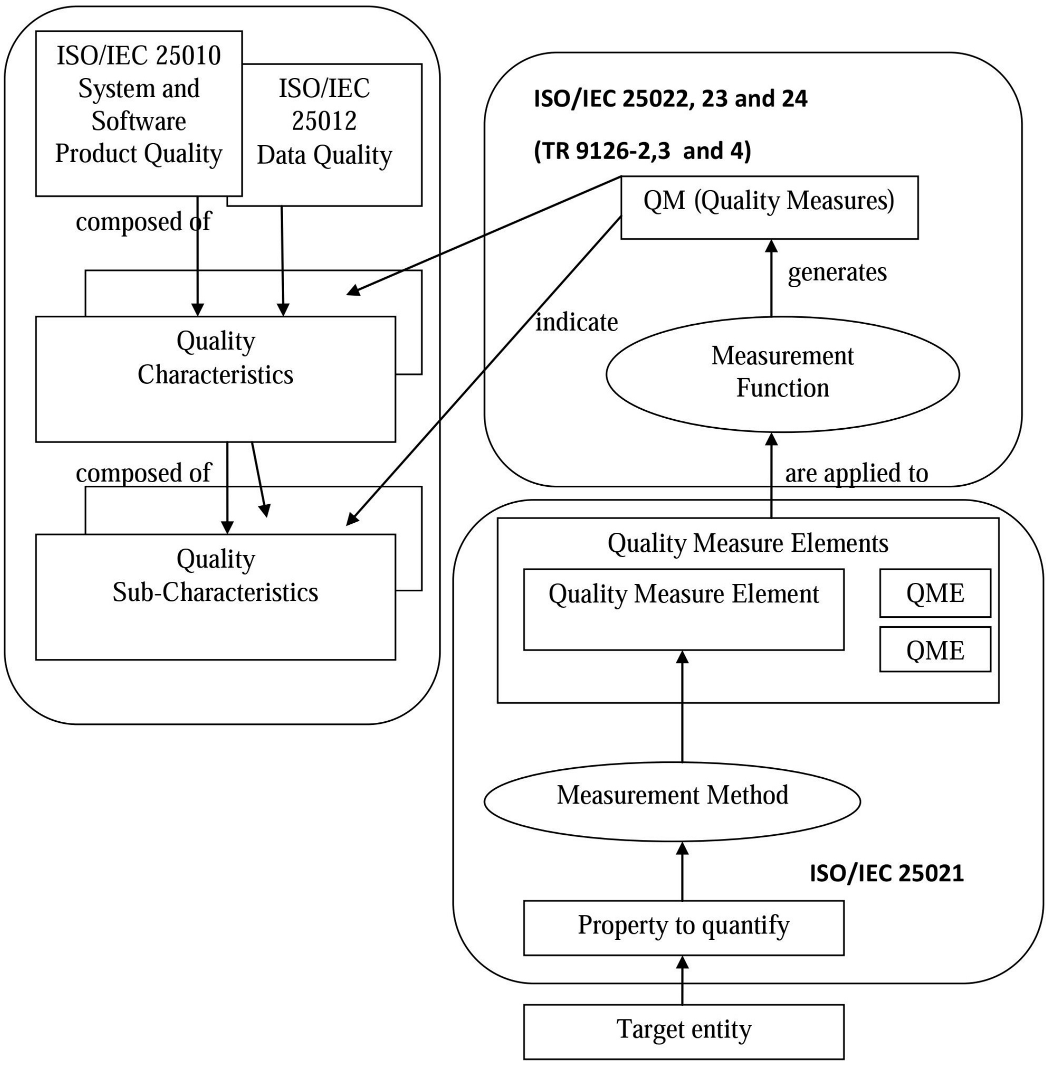


Figure 6: Relationship between property to quantify, measurement method, QME and QM.

3.6 Quality measure elements categories

Quality measure elements can be categorized into following categories [49]: data size, number of data items, number of failures, number of faults, number of functions, number of I/O, number of requirements, number of restarts, number of system operations, number of tasks, number of test cases, number of trials, number of user operations, product size and time duration.

3.6.1 Data size

This category consists of number of records of the same structure, class or format that satisfy the conditions given in the corresponding QME definitions [49]. The number of records can be count of records or size in bytes [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of change log data actually recorded.
2. Number of change log data planned to be recorded enough to trace software changes.
3. Number of data actually recorded during operation.
4. Number of data planned to be recorded enough to monitor status of software during operation.

3.6.2 Number of Data items

This category consists of the count of different structures, classes or formats of data that satisfy the conditions given in the corresponding QME definitions [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of data formats to be exchanged as in the specifications.
2. Number of data items implemented with specific levels of precision, confirmed in evaluation.
3. Number of data items that require specific levels of precision.
4. Number of data structures, which are operable and have no limitation after adaptation.
5. Number of input and output data items available from the interface.
6. Number of input and output data items which user successfully understands.
7. Number of interface data formats that have been implemented correctly as in the specifications.
8. Total number of data structures requiring adaptation capability.

3.6.3 Number of failures

This class consists of the QMEs that specify the count of all the expected or detected failures which occur in a given time duration [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of detected failures.
2. Number of resolved failures.
3. Number of transmission related error messages and failures.
4. Total number of actually detected failures.

3.6.4 Number of faults

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the count of software product faults detected or estimated in the given software product component [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of corrected faults in design/coding.
2. Number of detected faults.
3. Number of faults detected in review.

3.6.5 Number of functions

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the count of all the essential or optional functions that are related to requirement, implementation, testing, combination of them or more [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of functions implemented.
2. Number of functions (or types of functions) described in the product description.

3. Number of functions reviewed.
4. Number of functions described in requirement specifications.
5. Number of functions for which specific accuracy requirements need to be implemented.
6. Number of functions in which problems are detected in evaluation.
7. Number of functions in which specific accuracy requirements had been implemented, confirmed in evaluation.
8. Number of functions, which can be customized.
9. Number of implemented functions, which are capable of achieving, required results in specified multiple hardware environment as specified.
10. Number of implemented functions, which are capable of achieving, required results in specified multiple system software environment as specified.
11. Number of incorrectly implemented or missing functions detected.
12. Number of missing functions detected in evaluation.
13. Number of user interface functions.
14. Number of user interface functions whose purpose is understood by the user.
15. Total number of functions with hardware environment adaptation capability requirements.
16. Total number of functions with system software environment adaptation capability requirements.

3.6.6 Number of I/O

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the number of input and output events that may occur between observer and the system through direct interaction (e.g. dialogue) or automatic interactions that occur internally in the system to perform task given by observer [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of I/O messages during evaluation.
2. Required maximum number of I/O messages.
3. Required maximum number of transmission related error messages and failures.

3.6.7 Number of requirements

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the count of essential, optional, validated, or any type of requirement clauses [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of access controllability requirements implemented correctly as in the specifications.
2. Number of access controllability requirements in the specifications.

3.6.8 Number of restarts

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the number of attempts that are required for the system to recover to normal operation after a critical failure [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of restarts which met required time during testing or user operation support.
2. Total number of restarts during testing or user operation support.

3.6.9 Number of system operations

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the number of complete operations executed by the system irrespective of the number of individual steps required for each operation [49].

3.6.10 Number of tasks

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the number of the activities performed by the system or users for achieving a given goal [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of accesses to help until a user completes his/her task.
2. Number of completed tasks.
3. Number of tasks successfully completed after accessing online help and/or user documentation.
4. Throughput.
5. Total number of tasks tested.

3.6.11 Number of test cases

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the count of different test cases and scenarios that are designed, required and executed [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of passed test cases during testing or operation.
2. Number of performed test cases during testing or operation.
3. Number of test cases required.

3.6.12 Number of trials

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the count of attempts needed to perform the same operation with same input and same scenario as in stress testing or with different input and/or different scenarios [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Number of cases which a user succeeded to change install operation for his/her convenience.
2. Number of evaluations.
3. Total number of cases, which a user attempted to change install operation for his/her convenience.

3.6.13 Number of user operations

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specify the number of operations carried out by the user [49].

1. Number of user operations.

3.6.14 Product size

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the count of software product components in reference to a desired criterion such as lines of code (LOC), function points, modules, classes, or visual structures such as diagrams or their parts [49]. The set of quality measure elements recommended for this category is listed below [49]:

1. Product Size.

3.6.15 Time duration

This class consists of QMEs that satisfy the conditions in their corresponding QME definitions and specifies the time interval between starting time and finishing time of any activity [49]. The time interval may be internal or external and it can measure execution time, observation time or set time [49]. The set of quality measure elements recommended for this category are as below [49]:

1. Failure resolution time.

2. Operation time.
3. Response time.
4. Task time.
5. Turnaround time.

4 Methodology

There are much literature on software quality, software process quality models, software product quality models, hybrid software quality models, software development processes, software process improvement, software quality measurement, software metrics, etc. SQuaRE is relatively new ISO/IEC standard and there is almost no literature on software quality measure elements. Therefore, qualitative research using questionnaire was used for this thesis to know what kind of tools or methods are used by small and medium sized software companies for quality assurance of their software products. The companies were only provided with the questionnaire to know what metrics they were using for evaluating software quality, purpose of using those metrics, data collected for evaluating each metric, method of data collection, effort needed for metric evaluation and usefulness of the metric to the company. List of QMEs were not provided to any of the companies in order to focus purely on the software quality measurement processes in each company. Then the result of the questionnaire was analyzed, classified and standardized with ISO/IEC 25021 standards to recommend the best QMEs for software measurement.

The questionnaire was subjective in requiring explanation of the software quality evaluation procedure by the company. The name of columns for data collection were name of metrics, purpose of using metrics, data collected, method of data collection, stage of software development, effort [number of persons/time] and usefulness of metrics in Likert scale of 1 to 5 where 1 represented least useful metric and 5 represented highly useful metric. The questionnaire was sent to 31 small and medium-sized software companies for data collection. Out of them, only five responses were obtained. One company replied that they do not want to share such information even if that is for study purpose. One was rejected because they just gave reference to eclipse plugin and there was no information about measurement process in that company. Therefore, only three responses were considered eligible for data analysis.

4.1 Response to questionnaire

Most of the companies did not respond to the questionnaire. Some wished for web form of questionnaire with examples about the data fields to be filled. Therefore web form was also used for collecting data about software quality measurement in SMEs. All the respondents wanted confidentiality of data and prohibited the use of their company name or any kind of indicator to their company name.

4.2 Structure of questionnaire

The questionnaire in Ms Excel format is shown in figure 7. The questionnaire in web form had two pages as shown in figure 8 and figure 9. The data collected for each column or field is explained below:

4.2.1 Name, Email and Company name

These fields contained the responder's identity. The main purpose of these fields were to ensure authenticity of the responder.

4.2.2 Number of employees

This field indicated the number of employees in the company. This information was needed for classification of the company as small, medium or large companies. Some of the companies were diverse in nature, so we inquired about the number of software developers in the company for classifying them.

4.2.3 Name of metric

This data field was for naming the metric that the participating company was using. The respondent had full freedom to describe the metric being used in their own words.

4.2.4 Purpose of using metrics

This field allowed respondent to specify the purpose of using metrics. This was one of the most important data field that was considered for assigning QMEs for quality measurement.

4.2.5 Data collected

This data field contained information about the data collected for the measurement process of the metric.

4.2.6 Method of data collection

This data field contained information about the method used for collecting and/or evaluating the metric. The data collection method may be manual (needing human effort) or automatic. This field was expected to be crucial for selecting best QMEs but the obtained data was not sufficient for any meaningful conclusion. Therefore this field was also ignored.

4.2.7 Stage of software development

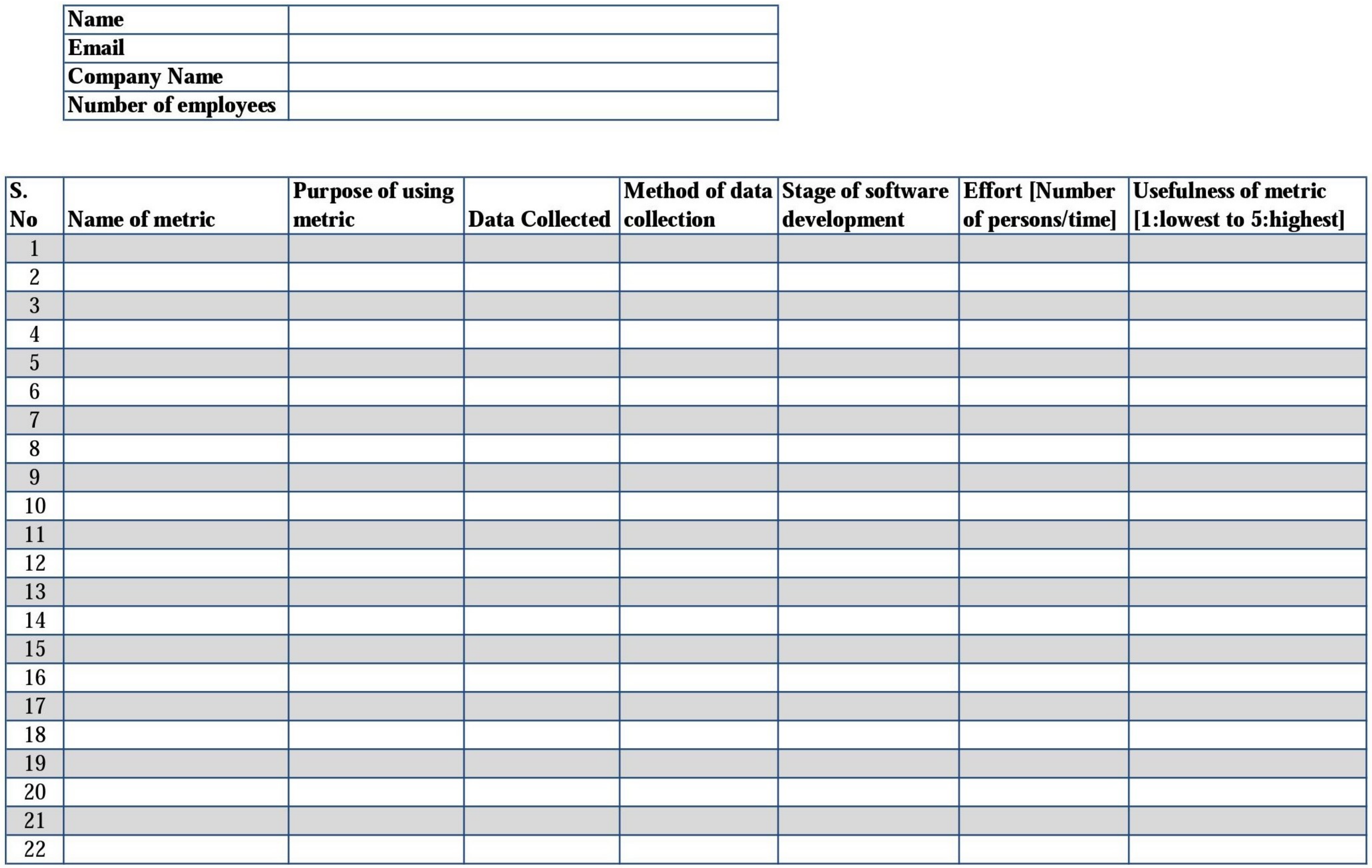
This data field contained information about the stage in the software development life cycle where the metric evaluation was performed. This field was helpful to assign QMEs.

4.2.8 Effort [Effort needed to measure the metric]

This data field contained information about the effort required for measuring the metric. The effort was expressed as the combination of number of persons and/or time required to perform the measurement process for the metric. The data obtained was not meaningful so data from this field was completely ignored.

4.2.9 Usefulness of metric

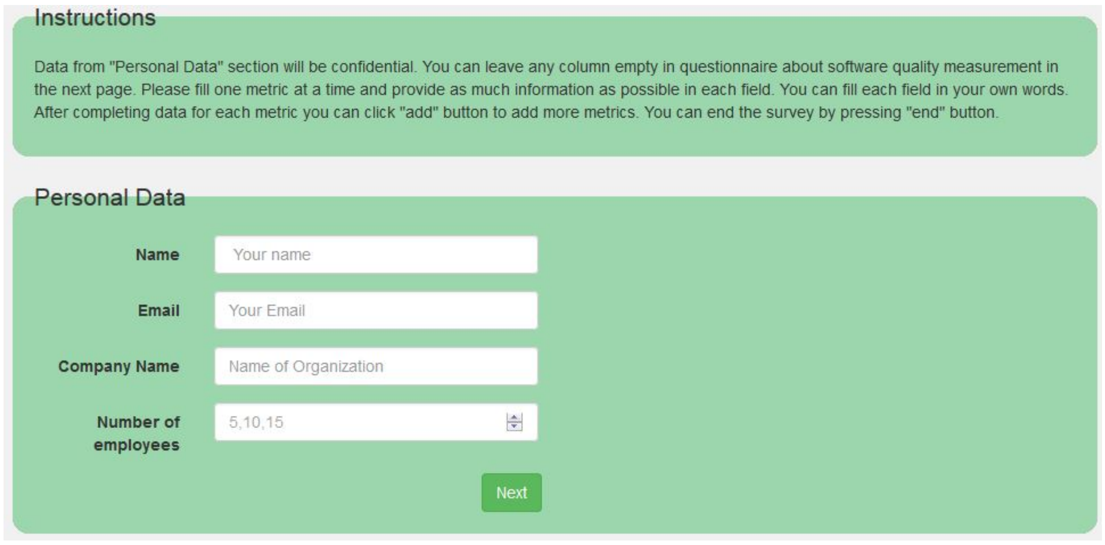
This data field contained information about the usefulness of metric from the metric evaluator's viewpoint and it was expressed in Likert scale of 1 to 5 where 1 indicated least useful metric and 5 indicated most useful metric. This field indicated the importance of the metric to the responding company.



Name	
Email	
Company Name	
Number of employees	

S. No	**Name of metric**	**Purpose of using metric**	**Data Collected**	**Method of data collection**	**Stage of software development**	**Effort [Number of persons/time]**	**Usefulness of metric [1:lowest to 5:highest]**
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							

Figure 7: Questionnaire as excel sheet

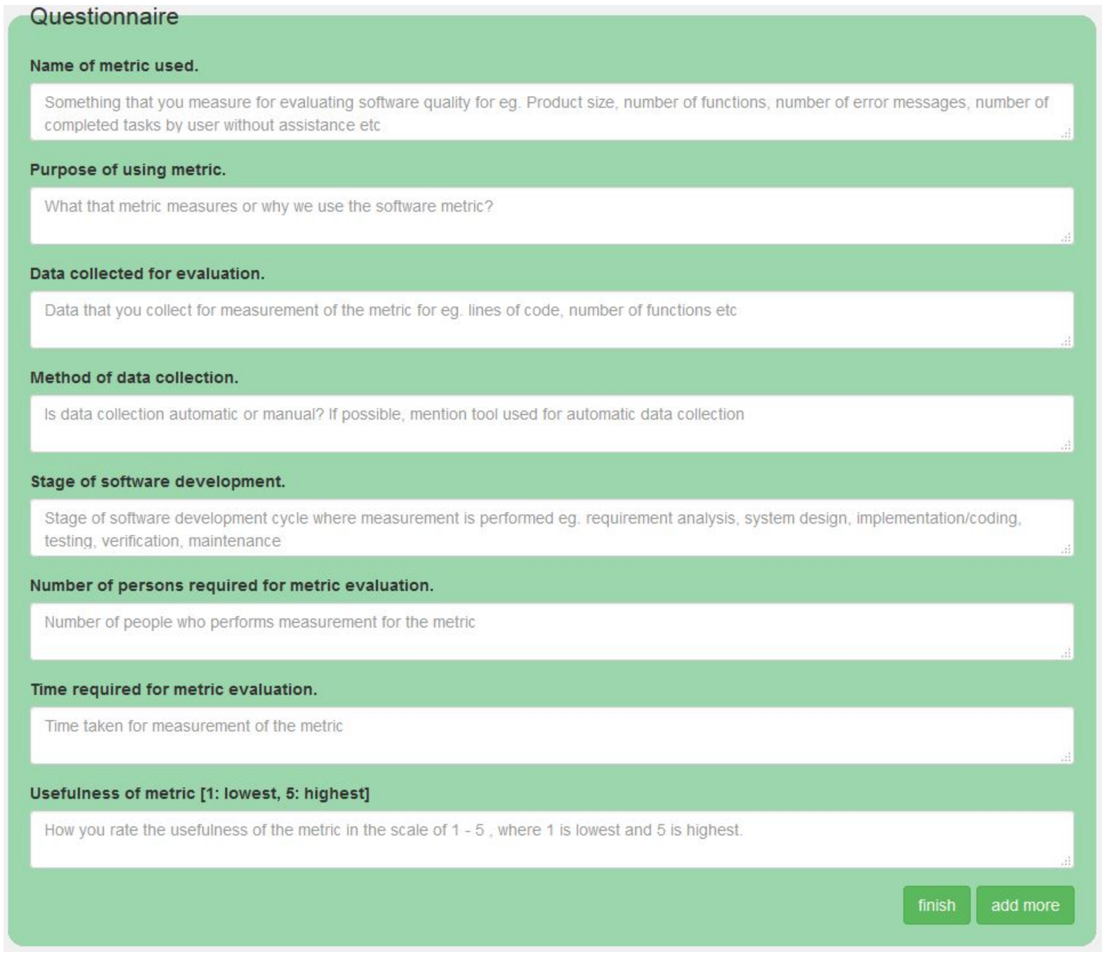

Instructions

Data from "Personal Data" section will be confidential. You can leave any column empty in questionnaire about software quality measurement in the next page. Please fill one metric at a time and provide as much information as possible in each field. You can fill each field in your own words. After completing data for each metric you can click "add" button to add more metrics. You can end the survey by pressing "end" button.

Personal Data

Name	Your name
Email	Your Email
Company Name	Name of Organization
Number of employees	5,10,15

Figure 8: Page 1 of questionnaire in web form


Questionnaire

Name of metric used.

Something that you measure for evaluating software quality for eg. Product size, number of functions, number of error messages, number of completed tasks by user without assistance etc

Purpose of using metric.

What that metric measures or why we use the software metric?

Data collected for evaluation.

Data that you collect for measurement of the metric for eg. lines of code, number of functions etc

Method of data collection.

Is data collection automatic or manual? If possible, mention tool used for automatic data collection

Stage of software development.

Stage of software development cycle where measurement is performed eg. requirement analysis, system design, implementation/coding, testing, verification, maintenance

Number of persons required for metric evaluation.

Number of people who performs measurement for the metric

Time required for metric evaluation.

Time taken for measurement of the metric

Usefulness of metric [1: lowest, 5: highest]

How you rate the usefulness of the metric in the scale of 1 - 5 , where 1 is lowest and 5 is highest.

Figure 9: Page 2 of questionnaire in web form

5 Results and conclusions

This section describes method used for data analysis, problems encountered during data analysis and result of the data analysis for each participating company. The final conclusion section indicates best QMEs for measurement of software quality for SMEs.

5.1 Data analysis method

The companies that participated in the study were named Company A, Company B and Company C to conceal their identity. These companies were classified according to the number of software professionals working for them, irrespective to the actual number of employees. Under this criterion, we classified Company A and Company B as small company, and Company C as medium-sized company. Data collected from each company was analyzed, and each metric in the data field was categorized into one or more QME categories by comparing purpose of metric and/or data collected for metric measurement to QME definitions provided in ISO/IEC 25021. The obtained data was presented in a table containing QME category, QME name, QME frequency and QME usefulness. QME frequency indicated the count of the QME name that were used for measuring other metric. QME usefulness was obtained from response to the questionnaire, and it indicated the usefulness of the metric measurement to the responding companies in the Likert scale of 1 to 5 where 1 represented least useful metric and 5 indicated highly useful metric. The table was sorted according to QME usefulness for each company. The QMEs whose QME usefulness value was less than three were categorized as less useful QMEs. The QMEs having QME usefulness value of 3 or more were categorized as useful QMEs. Then a list of useful QMEs for each company was obtained by ignoring less useful QMEs.

For identifying best QMEs for SMEs, we created a table with QME category, QME name, Company names and Matches. Then the QME names in each QME category of each company were compared with QME names in same category of other companies and the QME names which were present in two or more companies were listed in the table. Company Names column contained a “x” symbol to indicate that the QME name was present in the table containing QME categories and QMEs for the company. Matches column was the count of “x” symbol for each QME in the table. The table was sorted according to the value of matches column to identify the QMEs that were useful for all the companies. The QMEs that were present in all the participating companies were recommended as the best QMEs for SMEs and all the QMEs in this table were recommended as the most important QMEs for SMEs.

5.2 Problems encountered in data analysis

The responses from companies reflected what they were measuring, what data they required/collected and how they rate the importance of measuring each software property they were measuring. The questionnaire was subjective, so the responses were entirely dependent on the responder's view and understanding of the questionnaire. At some places, there was inconsistencies in data, e.g. metric name did not correspond to purpose of measuring metric or data collected. At such places, either both were ignored or both were considered for assigning QMEs to the metric, but more emphasis was given to the purpose of using metric and data collected for measurement. Some metrics that were not related to software product quality evaluation were discarded.

5.3 Company A

Company A was categorized as a small company on the basis of number of employees involved in software development. The data collected from Company A was analyzed by discarding the data fields that contained insignificant data, such as effort and method of data collection. QME categories and QMEs were assigned to each metric based on the purpose of using metric and data collected for measurement. The QMEs associated with each metric are presented in table 1.

Name of metric	Purpose of using metric	Data Collected	QME category	QME name
Performance	Search bottlenecks in a code	Time of evaluation of different components\functions\parts of the code	Time duration	Failure resolution time, Operation time, Response time, Turnaround time
Others code review	Check the other developer code with "fresh eye". Probably identify some points for improvement, see defects, etc	Possible defects, notes on improvements, suggestions	Number of faults, Number of functions	Number of faults detected in review, Number of corrected faults in design/coding; Number of functions in which problems are detected in evaluation, Number of missing functions detected in evaluation, Number of incorrectly implemented or missing functions detected
Walkthrough	Present implemented components\modules to the rest of the dev team, so they can try to use it from "user perspective" and suggest some changes, give some feedback	Notes, feedback	Number of data items, Number of tasks, Number of functions,Number of test cases	Number of interface data formats that have been implemented correctly as in the specifications, Number of input and output data items which user successfully understands; Number of accesses to help until a user completes his/her task, Number of completed tasks ,Number of tasks successfully completed after accessing online help and/or user documentation, Throughput; Number of user interface functions whose purpose is understood by the user; Number of passed test cases during testing or operation, Number of performed test cases during testing or operation, Number of test cases required
Lines of code commited	Correlate commits statistics with tasks, developer was working on during some period of time. Low amount of lines of code commited at period of time and\or often refactorings of the same code can identify that overall approach to the problem solving was wrong – wrong API, architect decisions, non-clarified requirements	Statistics on code commits by developer	Data size, Product size	Number of change log data actually recorded, Number of change log data planned to be recorded enough to trace software changes; Product size
User feedback	Check how much problems does software bring to users. Unfortunately, users are pretty often are eager to define correct types of issues – bugs\improvements, and their severity. Sometimes some minor requested improvement (for example, change some label) can be reported as "blocking bug". However, after clarifying such cases amount of blocking and critical bugs can give some understanding of used software version quality	Amount, types and severity of reported issues.	Number of tasks, Number of trials	Number of accesses to help until a user completes his/her task, Number of completed tasks , Number of tasks successfully completed after accessing online help and/or user documentation, Throughput; Number of cases which a user succeeded to change install operation for his/her convenience, Number of evaluations, Total number of cases which a user attempted to change install operation for his/her convenience

Table 1: QME categories and QMEs for Company A

The sorted table on the basis of usefulness of metrics used in Company A based on the response of the questionnaire is shown in table 1.

QME S.No	QME category	QME name	QME frequency	QME usefulness
1	Time duration	Failure resolution time	1	4
2		Operation time	1	4
3		Response time	1	4
4		Turnaround time	1	4
5	Number of faults	Number faults detected in review	1	3
6		Number of corrected faults in design/coding	1	3
7	Number of functions	Number of functions in which problems are detected in evaluation	1	3
8		Number of missing functions detected in evaluation	1	3
9		Number of incorrectly implemented or missing functions detected	1	3
10		Number of user interface functions whose purpose is understood by the user	1	3
11	Number of data items	Number of interface data formats that have been implemented correctly as in the specifications	1	3
12		Number of input and output data items which user successfully understands	1	3
13	Number of tasks	Number of accesses to help until a user completes his/her task	2	3
14		Number of completed tasks	2	3
15		Number of tasks successfully completed after accessing online help and/or user documentation	2	3
16		Throughput	2	3
17	Number of test cases	Number of passed test cases during testing or operation	1	3
18		Number of performed test cases during testing or operation	1	3
19		Number of test cases required	1	3
20	Data size	Number of change log data actually recorded	1	2
21		Number of change log data planned to be recorded enough to trace software changes	1	2
22	Product size	Product size	1	2
23	Number of trials	Number of cases which a user succeeded to change install operation for his/her convenience	1	2
24		Number of evaluations	1	2
25		Total number of cases which a user attempted to change install operation for his/her convenience	1	2

Table 2: Sorted QMEs on the basis of their usefulness to Company A

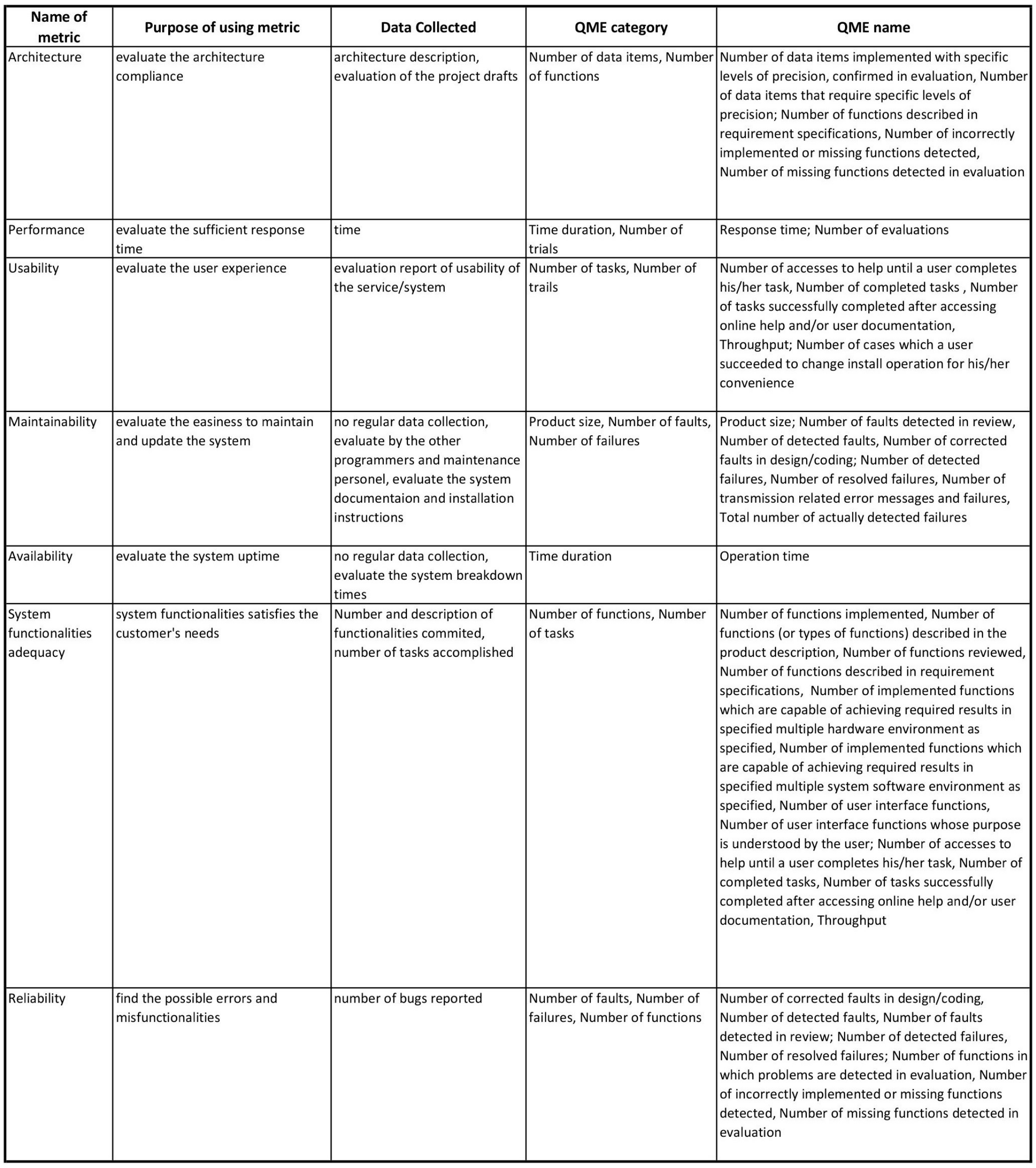
Data fields in the sorted table (table 2) indicated that QME category time duration was of highest priority with QME usefulness value of 4. QME categories such as number of faults, number of functions, number of data items, number of tasks and number of test cases had QME usefulness value of 3, therefore they were considered to be more useful to Company A than the categories like data size and number of trials. Therefore useful QMEs for Company A with QME usefulness value of 3 or higher are listed below:

1. Failure resolution time.
2. Operation time.
3. Response time.
4. Turnaround time.
5. Number of faults detected in review.
6. Number of corrected faults in design/coding.
7. Number of functions in which problems are detected in evaluation.
8. Number of missing functions detected in evaluation.
9. Number of incorrectly implemented or missing functions detected.
10. Number of user interface functions whose purpose is understood by the user.
11. Number of interface data formats that have been implemented correctly as in the specifications.
12. Number of input and output data items which user successfully understands.
13. Number of accesses to help until a user completes his/her task.
14. Number of completed tasks.
15. Number of tasks successfully completed after accessing online help and/or user documentation.
16. Throughput.
17. Number of passed test cases during testing or operation.
18. Number of performed test cases during testing or operation.
19. Number of test cases required.

5.4 Company B

Company B was categorized as small company on the basis of the number of employees involved in software development. The data collected from Company B was analyzed by discarding the data fields that contained insignificant data, such as effort and method of data collection. All the irrelevant data which were not related to software product quality were ignored. Data for metrics "architecture" and "reliability" did not provide sufficient information so Company B was requested to choose related QMEs from the set of QMEs that could be related to them.

The QMEs associated with the purpose of using metric and/or data collected for measurement of the metric are listed with their categories in table 3.



Name of metric	Purpose of using metric	Data Collected	QME category	QME name
Architecture	evaluate the architecture compliance	architecture description, evaluation of the project drafts	Number of data items, Number of functions	Number of data items implemented with specific levels of precision, confirmed in evaluation, Number of data items that require specific levels of precision; Number of functions described in requirement specifications, Number of incorrectly implemented or missing functions detected, Number of missing functions detected in evaluation
Performance	evaluate the sufficient response time	time	Time duration, Number of trials	Response time; Number of evaluations
Usability	evaluate the user experience	evaluation report of usability of the service/system	Number of tasks, Number of trails	Number of accesses to help until a user completes his/her task, Number of completed tasks , Number of tasks successfully completed after accessing online help and/or user documentation, Throughput; Number of cases which a user succeeded to change install operation for his/her convenience
Maintainability	evaluate the easiness to maintain and update the system	no regular data collection, evaluate by the other programmers and maintenance personel, evaluate the system documentaion and installation instructions	Product size, Number of faults, Number of failures	Product size; Number of faults detected in review, Number of detected faults, Number of corrected faults in design/coding; Number of detected failures, Number of resolved failures, Number of transmission related error messages and failures, Total number of actually detected failures
Availability	evaluate the system uptime	no regular data collection, evaluate the system breakdown times	Time duration	Operation time
System functionalities adequacy	system functionalities satisfies the customer's needs	Number and description of functionalities commited, number of tasks accomplished	Number of functions, Number of tasks	Number of functions implemented, Number of functions (or types of functions) described in the product description, Number of functions reviewed, Number of functions described in requirement specifications, Number of implemented functions which are capable of achieving required results in specified multiple hardware environment as specified, Number of implemented functions which are capable of achieving required results in specified multiple system software environment as specified, Number of user interface functions, Number of user interface functions whose purpose is understood by the user; Number of accesses to help until a user completes his/her task, Number of completed tasks, Number of tasks successfully completed after accessing online help and/or user documentation, Throughput
Reliability	find the possible errors and misfunctionalities	number of bugs reported	Number of faults, Number of failures, Number of functions	Number of corrected faults in design/coding, Number of detected faults, Number of faults detected in review; Number of detected failures, Number of resolved failures; Number of functions in which problems are detected in evaluation, Number of incorrectly implemented or missing functions detected, Number of missing functions detected in evaluation

Table 3: QME categories and QMEs for Company B

The sorted table on the basis of usefulness of metrics used in Company B based on the response of the questionnaire is shown in table 4.

QME S.No	QME category	QME name	QME frequency	QME usefulness
1	Number of functions	Number of functions described in requirement specifications	2	5
2		Number of functions implemented	1	5
3		Number of functions (or types of functions) described in the product description	1	5
4		Number of functions reviewed	1	5
5		Number of implemented functions which are capable of achieving required results in specified multiple hardware environment as specified	1	5
6		Number of implemented functions which are capable of achieving required results in specified multiple system software environment as specified	1	5
7		Number of user interface functions	1	5
8		Number of user interface functions whose purpose is understood by the user	1	5
9	Number of tasks	Number of accesses to help until a user completes his/her task	2	5
10		Number of completed tasks	2	5
11		Number of tasks successfully completed after accessing online help and/or user documentation	2	5
12		Throughput	2	5
13	Number of faults	Number of detected faults	2	4
14		Number of faults detected in review	2	4
15		Number of corrected faults in design/coding	2	4
16	Number of functions	Number of incorrectly implemented or missing functions detected	2	4
17		Number of missing functions detected in evaluation	2	4
18		Number of functions in which problems are detected in evaluation	1	4
19	Number of data items	Number of data items implemented with specific levels of precision, confirmed in evaluation	1	4
20		Number of data items that require specific levels of precision	1	4
21	Number of failures	Number of detected failures	2	4
22		Number of resolved failures	2	4
23	Number of trials	Number of cases which a user succeeded to change install operation for his/her convenience	1	4
24		Number of evaluations	1	2
25	Product size	product size	1	2
26	Number of failures	Number of transmission related error messages and failures	1	2
27		Total number of actually detected failures	1	2
28	Time duration	Operation time	1	2
29		Response time	1	2

Table 4: Sorted QMEs on the basis of their usefulness to Company B

Table 4 showed that the QME categories namely number of functions, number of tasks, number of faults, number of data items, number of failures and number of trials had more than average value for QME usefulness. The QMEs associated with these categories that had QME usefulness value more than 3 were considered as more useful QMEs to company B. They are listed below:

1. Number of functions described in requirement specifications.
2. Number of functions implemented.
3. Number of functions (or types of functions) described in the product description.
4. Number of functions reviewed.
5. Number of implemented functions which are capable of achieving required results in specified multiple hardware environment as specified.
6. Number of implemented functions which are capable of achieving required results in specified multiple system software environment as specified.
7. Number of user interface functions.
8. Number of user interface functions whose purpose is understood by the user.
9. Number of incorrectly implemented or missing functions detected.
10. Number of missing functions detected in evaluation.
11. Number of functions in which problems are detected in evaluation.
12. Number of accesses to help until a user completes his/her task.
13. Number of completed tasks.
14. Number of tasks successfully completed after accessing online help and/or user documentation.
15. Throughput.
16. Number of detected faults.
17. Number of faults detected in review.
18. Number of corrected faults in design/coding.

19. Number of data items implemented with specific levels of precision, confirmed in evaluation.
20. Number of data items that require specific levels of precision.
21. Number of detected failures.
22. Number of resolved failures.
23. Number of cases which a user succeeded to change install operation for his/her convenience.

5.5 Company C

Company C was categorized as medium-sized company on the basis of the number of employees involved in software development. The data collected from Company C was analyzed by discarding the data fields that contained insignificant data, such as effort and method of data collection. The metric and/or purpose of using metric that could not be associated with any of the QME categories or QMEs recommended in ISO/IEC 25021 were not considered.

The QMEs associated with the purpose of using metric and/or data collected for measurement of the metric are listed with their categories is shown in table 5.

Name of metric	Purpose of using metric	Data Collected	QME category	QME name
Product size	Measure size	Lines of code, Number of Files, Number of Functions, Number of Classes, statements, accessors	Product size, Number of functions	Product size; Number of functions implemented, Number of functions described in requirement specifications
Issues/Errors	Finding out the errors/issues and to find out how critical is the issue	Issues in the code	Number of faults, Number of functions	Number of detected faults, Number of faults detected in review, Number of corrected faults in design/coding; Number of functions in which problems are detected in evaluation, Number of incorrectly implemented or missing functions detected, Number of missing functions detected in evaluation
Unit Tests	Check the Test cases	Lines Coverage, Condition Coverage, Testing use cases	Number of tasks, Number of test cases, Number of trials, Number of failures	Number of accesses to help until a user completes his/her task, Number of completed tasks, Number of tasks successfully completed after accessing online help and/or user documentation, Throughput, Total number of tasks tested; Number of passed test cases during testing or operation, Number of performed test cases during testing or operation, Number of test cases required; Number of cases which a user succeeded to change install operation for his/her convenience, Number of evaluations, Total number of cases which a user attempted to change install operation for his/her convenience; Number of detected failures, Total number of actually detected failures

Table 5: QME categories and QMEs for Company C

The sorted table on the basis of usefulness of metrics used in Company C that is based on the response of the questionnaire is shown in table 6.

QME S.No	QME category	QME name	QME frequency	QME usefulness
1	Product size	Product size	1	4
2	Number of functions	Number of functions implemented	1	4
3		Number of functions described in requirement specifications	1	4
4		Number of functions in which problems are detected in evaluation	1	4
5		Number of incorrectly implemented or missing functions detected	1	4
6		Number of missing functions detected in evaluation	1	4
7	Number of faults	Number of detected faults	1	4
8		Number of faults detected in review	1	4
9		Number of corrected faults in design/coding	1	4
10	Number of failures	Number of detected failures	1	3
11		Total number of actually detected failures	1	3
12	Number of tasks	Number of accesses to help until a user completes his/her task	1	3
13		Number of completed tasks	1	3
14		Number of tasks successfully completed after accessing online help and/or user documentation	1	3
15		Throughput	1	3
16		Total number of tasks tested	1	3
17	Number of test cases	Number of test cases required	1	3
18		Number of performed test cases during testing or operation	1	3
19		Number of passed test cases during testing or operation	1	3
20	Number of trials	Number of cases which a user succeeded to change install operation for his/her convenience	1	3
21		Number of evaluations	1	3
22		Total number of cases which a user attempted to change install operation for his/her convenience	1	3

Table 6: Sorted QMEs on the basis of their usefulness to Company C

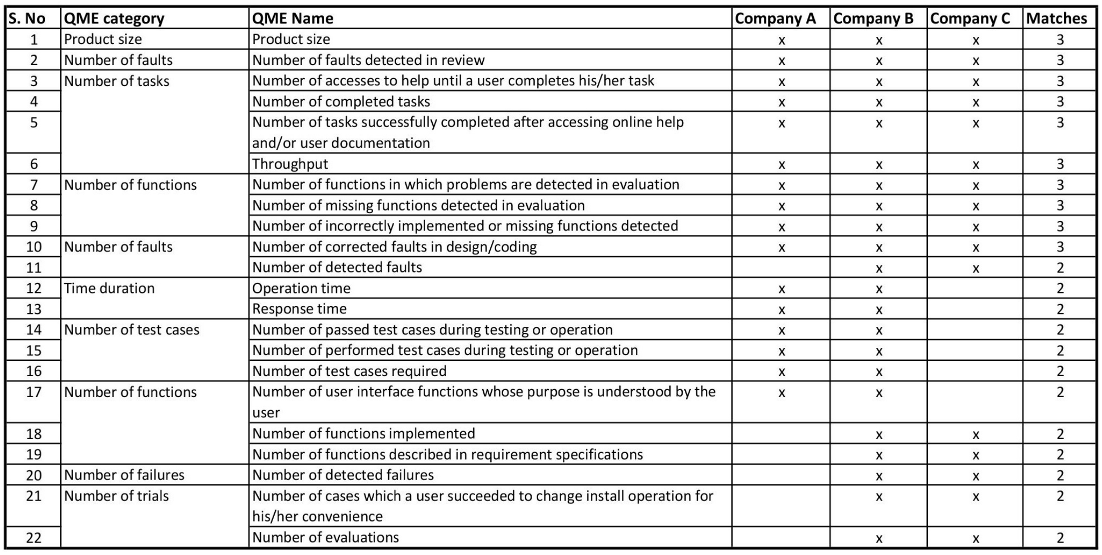
The QMEs that had QME usefulness value of 3 or more were considered as useful QMEs to Company C. They are listed below:

1. Product size.
2. Number of functions implemented.
3. Number of functions described in requirement specifications.
4. Number of functions in which problems are detected in evaluation.
5. Number of incorrectly implemented or missing functions detected.
6. Number of missing functions detected in evaluation.

7. Number of detected faults.
8. Number of faults detected in review.
9. Number of corrected faults in design/coding.
10. Number of detected failures.
11. Total number of actually detected failures.
12. Number of accesses to help until a user completes his/her task.
13. Number of completed tasks.
14. Number of tasks successfully completed after accessing online help and/or user documentation.
15. Throughput.
16. Total number of tasks tested.
17. Number of test cases required.
18. Number of performed test cases during testing or operation.
19. Number of passed test cases during testing or operation.
20. Number of cases which a user succeeded to change install operation for his/her convenience.
21. Number of evaluations.
22. Total number of cases which a user attempted to change install operation for his/her convenience.

5.6 Conclusion

During data analysis, a list of more useful QMEs to each company were identified. It was evident from the table containing QME categories and QMEs, that each company had different usefulness value for the same QME. It was noticed that the most important QME for one company was even not considered to be measured or was considered less useful in the other company. These inconsistencies made the data field "QME usefulness" to be less useful and no information could be extracted from the data field to recommend useful QMEs to all the participating companies. Therefore, we ignored the usefulness of QME to each company and compared QMEs in each category of a company to QMEs of same category of other companies to identify the common QMEs to the participating companies. As described in data analysis method, a table was created to summarize the study.



S. No	QME category	QME Name	Company A	Company B	Company C	Matches
1	Product size	Product size	x	x	x	3
2	Number of faults	Number of faults detected in review	x	x	x	3
3	Number of tasks	Number of accesses to help until a user completes his/her task	x	x	x	3
4		Number of completed tasks	x	x	x	3
5		Number of tasks successfully completed after accessing online help and/or user documentation	x	x	x	3
6		Throughput	x	x	x	3
7	Number of functions	Number of functions in which problems are detected in evaluation	x	x	x	3
8		Number of missing functions detected in evaluation	x	x	x	3
9		Number of incorrectly implemented or missing functions detected	x	x	x	3
10	Number of faults	Number of corrected faults in design/coding	x	x	x	3
11		Number of detected faults		x	x	2
12	Time duration	Operation time	x	x		2
13		Response time	x	x		2
14	Number of test cases	Number of passed test cases during testing or operation	x	x		2
15		Number of performed test cases during testing or operation	x	x		2
16		Number of test cases required	x	x		2
17	Number of functions	Number of user interface functions whose purpose is understood by the user	x	x		2
18		Number of functions implemented		x	x	2
19		Number of functions described in requirement specifications		x	x	2
20	Number of failures	Number of detected failures		x	x	2
21	Number of trials	Number of cases which a user succeeded to change install operation for his/her convenience		x	x	2
22		Number of evaluations		x	x	2

Table 7: Common QMEs in participating companies

Table 7 shows the QMEs that were present in two or more responding companies. As we had only three companies that participated in our study, therefore these QMEs were considered as the most important QMEs for those companies which were evaluating them. These QMEs were recommended as the most important QMEs to SMEs. They are listed below:

1. Product size.
2. Number of faults detected in review.
3. Number of accesses to help until a user completes his/her task.
4. Number of completed tasks.
5. Number of tasks successfully completed after accessing online help and/or user documentation.

6. Throughput.
7. Number of functions in which problems are detected in evaluation.
8. Number of missing functions detected in evaluation.
9. Number of incorrectly implemented or missing functions detected.
10. Number of corrected faults in design/coding.
11. Number of detected faults.
12. Operation time.
13. Response time.
14. Number of passed test cases during testing or operation.
15. Number of performed test cases during testing or operation.
16. Number of test cases required.
17. Number of user interface functions whose purpose is understood by the user.
18. Number of functions implemented.
19. Number of functions described in requirement specifications.
20. Number of detected failures.
21. Number of cases which a user succeeded to change install operation for his/her convenience.
22. Number of evaluations.

Table 7 clearly indicated that the following QMEs were present in all the three participating companies in our study. They were considered to be the best QMEs for the participating companies and they were recommended as the best QMEs for measurement of software quality in SMEs. They are listed below:

1. Product size.
2. Number of faults detected in review.

3. Number of corrected faults in design/coding.
4. Number of accesses to help until a user completes his/her task.
5. Number of completed tasks.
6. Number of tasks successfully completed after accessing online help and/or user documentation.
7. Throughput.
8. Number of functions in which problems are detected in evaluation.
9. Number of missing functions detected in evaluation.
10. Number of incorrectly implemented or missing functions detected.

6 Limitations and further studies

The main goal of the study was to recommend the best QMEs for the measurement of software quality for SMEs. There were some limitations to the study. They are listed below:

1. The low response rate restricted the study to three SMEs. The low response rate from SMEs may be due to various reasons like maintaining confidentiality of their software quality measurement process, lack of proper software quality measurement in their company, and busy schedule of employees involved in software quality control.
2. Some data fields of the collected data were not consistent with the associated data fields. For example, at some places, the metric name did not represent the actual purpose of the metric, data collected for metric measurement did not satisfy the purpose of using metric, etc. For such cases, the QMEs related to the purpose of using metric and/or data collection were associated with those metrics.

For better understanding of actual software quality measurement process and practices in SMEs, we recommend that the researcher should be part of software development team in order to closely observe and understand the software quality measurement process. This will provide an opportunity for the researcher to interview software testers and/or quality assurance managers, and know their viewpoint about the quality measure elements they consider to be useful for them, their reasons for choosing a particular set of metrics or tools, and the most common quality requirements that are specified by clients. This can help to improve the selection criteria for identifying and selecting the best QMEs software quality measurement for SMEs.

We would recommend for further studies to select software quality measure elements for SMEs on the basis of constraints such as time, cost, ease of evaluation, and client's software quality requirements.

References

[1] Mario Gleirscher, Dmitriy Golubitskiy, Maximilian Irlbeck, and Stefan Wagner. Introduction of static quality analysis in small-and medium-sized software enterprises: experiences from technology transfer. *Software Quality Journal*, pages 1–44.

[2] Francisco J Pino, Félix García, and Mario Piattini. Software process improvement in small and medium software enterprises: a systematic review. *Software Quality Journal*, 16(2):237–261, 2008.

[3] Edit Lukács. The economic role of smes in world economy, especially in europe. *European Integration Studies*, (1 (4):3–12, 2005.

[4] Europäische Kommission. *The new SME definition: User guide and model declaration*. European Comm., Publication Office, 2005.

[5] Mohamed E Fayad, Mauri Laitinen, and Robert P Ward. Thinking objectively: software engineering in the small. *Communications of the ACM*, 43(3):115–118, 2000.

[6] Ita Richardson and Chrisiane Gresse von Wangenheim. Why are small software organizations different. *IEEE software*, 24(1):18–22, 2007.

[7] Romana Vajde Horvat, Ivan Rozman, and József Györkös. Managing the complexity of spi in small companies. *Software Process: Improvement and Practice*, 5(1):45–54, 2000.

[8] Stephen H Kan. *Metrics and models in software quality engineering*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[9] Gerard O'Regan. *A practical approach to software quality*. Springer, 2002.

[10] Daniel Galin. *Software quality assurance: from theory to implementation*. Pearson education, 2004.

[11] Barrie G Dale, Ton Van Der Wiele, and Jos Van Iwaarden. *Managing quality*. John Wiley & Sons, 2013.

[12] David A Garvin. What does "product quality" really mean. *Sloan management review*, 1, 1984.

[13] Joana G Geraldi, Elmar Kutsch, and Neil Turner. Towards a conceptualisation of quality in information technology projects. *International Journal of Project Management*, 29(5):557–567, 2011.

[14] Alan Gillies. *Software quality: theory and management.* Lulu. com, 2011.

[15] Patrik Berander, Lars-Ola Damm, Jeanette Eriksson, Tony Gorschek, Kennet Henningsson, Per Jönsson, Simon Kågström, Drazen Milicic, Frans Mårtensson, Kari Rönkkö, et al. Software quality attributes and trade-offs. *Blekinge Institute of Technology*, 2005.

[16] A Blanton Godfrey. *Juran's quality handbook.* McGraw Hill, 1999.

[17] Stefan Wagner and Florian Deißenböck. *Software product quality control.* Springer, 2013.

[18] Michele Lanza and Radu Marinescu. *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems.* Springer Science & Business Media, 2007.

[19] Barbara Kitchenham and Shari Lawrence Pfleeger. Software quality: The elusive target. *IEEE software*, 13(1):12–21, 1996.

[20] Mrs Manisha L Waghmode and Pallavi P Jamsandekar. Software quality models: A comparative study.

[21] ISO ISO. Iec25010: 2011 systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models. *International Organization for Standardization*, page 34, 2011.

[22] Andrea Schiffauerova and Vince Thomson. A review of research on cost of quality models and best practices. *International Journal of Quality & Reliability Management*, 23(6):647–669, 2006.

[23] Norman Fenton and James Bieman. *Software metrics: a rigorous and practical approach.* CRC Press, 2014.

[24] Bruce R Maxim and Marouane Kessentini. An introduction to modern software quality assurance. *Software Quality Assurance: In Large Scale and Complex Software-intensive Systems*, page 19, 2015.

[25] Motoei Azuma. Square: the next generation of the iso/iec 9126 and 14598 international standards series on software product quality. In *ESCOM (European Software Control and Metrics conference)*, pages 337–346, 2001.

[26] I Standard. Software engineering – software product quality requirements and evaluation (square) – guide to square. *ISO Standard*, 25000:2005, 2005.

[27] José P Miguel, David Mauricio, and Glen Rodríguez. A review of software quality models for the evaluation of software products. *arXiv preprint arXiv:1412.2977*, 2014.

[28] Anas Bassam Al-Badareen, Mohd Hasan Selamat, Marzanah A Jabar, Jamilah Din, and Sherzod Turaev. Software quality models: A comparative study. In *Software Engineering and Computer Systems*, pages 46–55. Springer, 2011.

[29] Benjamin Zeiss, Diana Vega, Ina Schieferdecker, Helmut Neukirchen, and Jens Grabowski. Applying the iso 9126 quality model to test specifications. *Software Engineering*, 15(6):231–242, 2007.

[30] Joseph P Cavano and James A McCall. A framework for the measurement of software quality. In *ACM SIGMETRICS Performance Evaluation Review*, volume 7, pages 133–139. ACM, 1978.

[31] Deepshikha Jamwal. Analysis of software quality models for organizations. *International Journal of Latest Trends in Computing*, 1(2), 2010.

[32] Rafa E Al-Qutaish. Quality models in software engineering literature: an analytical and comparative study. *Journal of American Science*, 6(3):166–175, 2010.

[33] Hongyu Pei Breivold and Ivica Crnkovic. Analysis of software evolvability in quality models. In *Software Engineering and Advanced Applications, 2009. SEAA'09. 35th Euromicro Conference on*, pages 279–282. IEEE, 2009.

[34] Barry W Boehm, John R Brown, and Mlity Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering*, pages 592–605. IEEE Computer Society Press, 1976.

[35] R. Geoff Dromey. A model for software product quality. *Software Engineering, IEEE Transactions on*, 21(2):146–162, 1995.

[36] Rafa E Al-Qutaish. An investigation of the weaknesses of the iso 9126 international standard. In *Computer and Electrical Engineering, 2009. ICCEE'09. Second International Conference on*, volume 1, pages 275–279. IEEE, 2009.

[37] David Zubrow Yukio Tanitsu Markku Tukiainen Nigel BEVAN, Vipula Godamunne. Iso/iec cd 25010.3: Systems and software engineering – software product quality requirements and evaluation(square) – quality models for software product quality and system quality in use. Unpublished paper, 2009.

[38] Norman E Fenton and Shari Lawrence Pfleeger. *Software metrics: a rigorous and practical approach*. PWS Publishing Co., 1998.

[39] Cem Kaner and Walter P Bond. Software engineering metrics: What do they measure and how do we know? *methodology*, 8:6, 2004.

[40] Christof Ebert, Manfred Bundschuh, Reiner Dumke, and Andreas Schmietendorf. *Best practices in software measurement*. Springer, 2005.

[41] Horst Zuse. *A framework of software measurement*. Walter de Gruyter, 1998.

[42] Nigel BEVAN Danilo SCALET, Witold SURYN. Fcd 25000–software engineering – software product quality requirements and evaluation (square) – guide to square. Unpublished paper, 2005.

[43] Ruchika Malhotra. *Empirical Research in Software Engineering: Concepts, Analysis, and Applications*. CRC Press, 2015.

[44] Jitender Kumar Chhabra and Varun Gupta. A survey of dynamic software metrics. *Journal of computer science and technology*, 25(5):1016–1029, 2010.

[45] IEEE Computer Society. Software Engineering Technical Committee. *IEEE Standard for a Software Quality Metrics Methodology*. Institute of Electrical and Electronics Engineering, 1993.

[46] Tu Honglei, Sun Wei, and Zhang Yanan. The research on software metrics and software complexity metrics. In *Computer Science-Technology and Applications, 2009. IFCSTA'09. International Forum on*, volume 1, pages 131–136. IEEE, 2009.

[47] ISO. Systems and software engineering – systems and software quality requirements and evaluation (square) – quality measure elements. Unpublished paper, 2012.

[48] Alain Abran, R Al Qutaish, J Desharnais, and Naji Habra. Iso-based models to measure software product quality. *Institute of Chartered Financial Analysts of India (ICFAI)-ICFAI Books*, 2007.

[49] Prof. Lee Keum-Suk Vaníček Jiří Dr. Renate Sitte Motoei AZUMA, Dr. Ota Novotny. Software engineering – software product quality requirements and evaluation (square) – quality measure elements. Unpublished paper, 2007.