INTRODUCTION TO SOFTWARE PRODUCT QUALITY

BY PANKAJ KAMTHAN

1. INTRODUCTION

Quality is never an accident; it is always the result of high intention, sincere effort, intelligent direction, and skillful execution; it represents the wise choice of many alternatives.
— William A. Foster

He who would run a company on visible figures alone, will in time have neither a company nor figures.
— William E. Deming

A refund for defective software might be nice, except it would bankrupt the entire software industry in the first year.
— Andrew S. Tanenbaum

There have been a number of studies on software quality, from technical perspective, from economics perspective, and from management perspective [Jones, 1998; Horch, 2003; Sikka, 2005; Jayaswal, Patton, 2006, Chapter 4; Kandt, 2006; Laird, Brennan, 2006, Chapter 8; **Naik, Tripathy, 2008, Chapter 1 and Chapter 17**; Bakota, Hegedüs, Kortvelyesi, Ferenc, Gyimóthy, 2011; Laporte, Berrhouma, Doucet, Palza-Vargas, 2012; Regan, 2014; Suryn, 2014; Wieczorek, Vos, Bons, 2014; Fenton, Bieman, 2015].

This document explores the notion of quality, in general, and as it pertains to software engineering, in particular. In doing so, theoretical as well as practical aspects of software product quality are considered.

CONVENTION/NOTATION

In this document, the terms **'quality model'** and **'software quality model'** are considered synonymous, and are used interchangeably. Furthermore, in this document, no distinction is made between **'product' and 'service'** [Bauer, Adams, 2014], unless otherwise specified.

2. DEFINITION OF QUALITY

You don't understand anything until you learn it more than one way.
— Marvin Minsky

There are various definitions of quality, including in the context of software engineering [Jones, Bonsignour, 2012, Chapter 1]. There are definitions of quality as given by national and international standards but there is **no 'ideal' definition of quality**.

Indeed, there can be **different viewpoints of quality**:

- Conformance Viewpoint
- Human Viewpoint
- Negative Viewpoint

CONFORMANCE VIEWPOINT

The next definition of quality is relatively abstract, independent of the subject, and from a conformance-viewpoint:

Definition [Quality] [ISO/IEC, 2001]. The totality of characteristics of an entity that bear on its ability to satisfy **stated and implied needs**.

For example, an 'entity' could be **project, process, product, people, or resources**.

HUMAN VIEWPOINT

The next definition of quality is relatively concrete, specific to a collection of subjects, and from a human-viewpoint:

Definition [Quality] [ISO/IEC/IEEE, 2010]. [The] ability of a product, service, system, component, or process to meet customer or user needs, expectations, or requirements.

NEGATIVE VIEWPOINT

The quality of a software system can be expressed in terms of **presence of defects** [Emam, 2005], specifically, in terms of **defect density** (**number of defects discovered per module size)** [Rico, 2004, Section 5.3]. In other words, the **lower the defect density**, the **higher the quality**.

3. MOTIVATION FOR QUALITY

There are several technical as well as non-technical reasons for interest in and attention to quality.

3.1. QUALITY IN THE DEFINITION OF SOFTWARE ENGINEERING

The significance of quality in software engineering is underscored by the mention of the term ‘quality’ **explicitly** in the **definition of software engineering**.

3.2. QUALITY IN SWEBOK

SWEBOK®

The **Guide to the Software Engineering Body of Knowledge (SWEBOK)** “describes the sum of knowledge within the profession of software engineering”. SWEBOK continues to **evolve**. The latest version of SWEBOK is SWEBOK3 [IEEE, 2014]. The purpose of the **SWEBOK3** is “to describe what portion of the Body of Knowledge is generally accepted, to organize that portion, and to provide a topical access to it”.

In SWEBOK, there are a number of Knowledge Areas (KAs). Software Quality is a KA of the SWEBOK3, as shown in Figure 1.

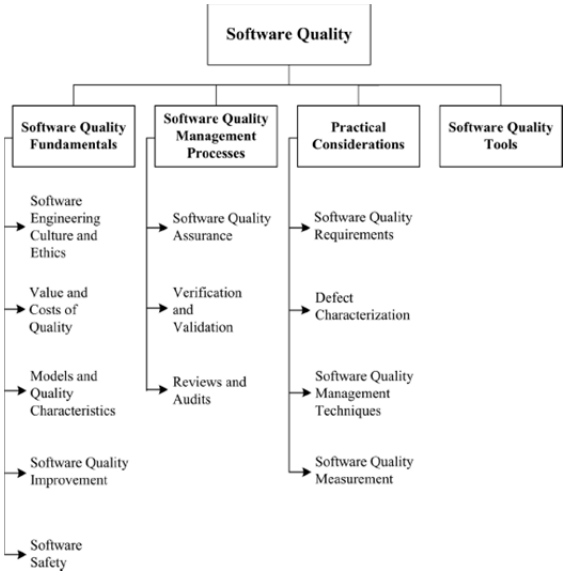


Figure 1. Software Quality is a Knowledge Area in the Guide to the Software Engineering Body of Knowledge (SWEBOK). (Source: SWEBOK [IEEE, 2014, Chapter 10].)

3.3. QUALITY AS PRINCIPLE

Make Quality Number 1 has been posited as one of the **principles of software engineering** [Davis, 1994].

3.4. QUALITY IN A MODEL OF SOFTWARE PROJECT MANAGEMENT

There are a number of **'equilibrium triangles' of software project management**, and they usually include quality as one of the elements.

4. QUALITY VERSUS QUANTITY

Not everything that can be counted counts, and not everything that counts can be counted.
— Albert Einstein

When a system's power is measured by the number of its features, quantity becomes more important than quality.
— Niklaus Wirth

The notions of 'quality' and 'quantity' are **not** substitutes of each other. However, at times, aiming for quantity can, in the long-term, help towards improving quality.

4.1. QUALITY EN ROUTE QUANTITY: A CASE OF CERAMIC POTTERY

In the classical debate over "quality versus quantity", the following quotation [Buxton, 2007, Page 141] highlights the significance of practice:

> The ceramics teacher announced on opening day that he was dividing the class into two groups.
>
> All those on the **left** side of the studio, he said, would be graded solely on the **quantity** of work they produced, all those on the **right** solely on its **quality**.
>
> [The] procedure was simple: on the final day of class he would bring in his bathroom scales and weigh the work of the quantity group: fifty pounds of pots rated an "A", forty pounds a "B", and so on. Those being graded on "quality", however, needed to produce only one pot — albeit a perfect one — to get an "A".
>
> Well, came grading time and a curious fact emerged: the works of highest quality were all produced by the group being graded for quantity. **It seems that while the "quantity" group was busily churning out piles of work — and learning from their mistakes — the "quality" group had sat theorizing about perfection, and in the end had little more to show for their efforts than grandiose theories and a pile of dead clay.**

5. PERSPECTIVES OF QUALITY

It has been realized that quality is a **multifaceted concept**. The facets of quality include the entity of interest, the viewpoint on that entity, and quality attributes of that entity [Kitchenham, Pfleeger, 1996; Kan, 2003].

There are different perspectives of quality [Wong, 2006].

5.1. PERSPECTIVES OF QUALITY: APPROACH 1

In one of the earliest approaches towards perceptions of quality [Garvin, 1984], five views of quality are given, as illustrated in Figure 2 and elaborated in Table 1.

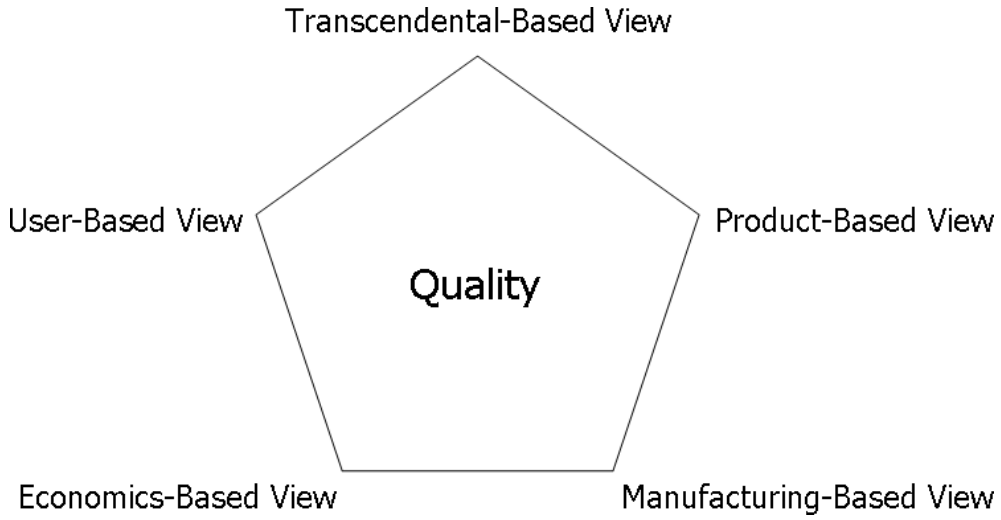


Figure 2. The five different views of quality.

	Type of View	Understanding of Quality
Quality	Transcendental-Based View	Quality is Perfective
	Product-Based View	Quality is Measurable
	Manufacturing-Based View	Quality is Conformance
	Economics-Based View	Quality is Benefit for Cost
	User-Based View	Quality is Satisfaction

Table 1. The five different views of quality.

REMARKS

- These views have been discussed in the context of **software engineering**, in general, in [Kitchenham, Pfleeger, 1996; Naik, Tripathy, 2008, Chapter 1 and Chapter 17], and sub-disciplines of software engineering, such as **software architecture**, in particular, in [Hansen, Jonasson, Neukirchen, 2011].

- It is likely that in certain initiatives, **multiple views** may need to be satisfied in addressing quality of a software product. Indeed, multiple views may be needed to make use of quality for practical purposes.

- These views are **not necessarily exclusive**. For example, an **economics-based view constrains the transcendental-based view** of quality. Therefore, any initiatives for quality assurance or evaluation need to end if **the cost exceeds the benefit**.

THE TRANSCENDENTAL-BASED VIEW OF QUALITY

Everything can be improved.
— Clarence W. Barron

That ... ideals cannot be attained is not a reason for disregarding them. Perfect cleanliness is also impossible, but it does not serve as a warrant for not washing, much less for rolling in a manure pile.
— Scott Gordon

In a transcendental-based view of quality, the notion of **quality is perfective**. In other words, it is possible to aim for quality, but never achieve it. For example, one group of students in an earlier quoted text had a **transcendental-based view** of quality.

The main issue in this view is that of **impracticality** [Bach, 1997]. In the extreme case, it can lead to a dichotomous view ("perfect" ≡ acceptable, "imperfect" ≡ unacceptable) [Gotterbarn, 2010]. If a software product is unacceptable, then, by definition, the corresponding software project can never be considered as 'concluded'. If a software project cannot be 'concluded', it is, by definition, unsuccessful.

THE PRODUCT-BASED VIEW OF QUALITY

The term product is a central concept in **marketing**. It designates anything that can be offered to a market for attention, acquisition, use, or consumption that might satisfy a want or need. The set of people or organizations the product is made available for is called a **market**. A **software product is a product** whose primary component is software [Maedche, Botzenhardt, Neer, 2012, Page 55].

In a product-based view of quality, the notion of **quality is measurable**. In other words, it should be possible to qualify or quantify quality in order to improve it. This view is based on the hypothesis that 'good' internal properties of a product result in 'good' external properties.

The main issue in this view is that of **quantifiability**. There are quality attributes, especially pertaining to human-machine interaction, which may not be quantifiable. For example, it is difficult to quantify comfortableness or satisfaction.

THE MANUFACTURING-BASED VIEW OF QUALITY

In a manufacturing-based view of quality, the notion of **quality is conformance**. In other words, it should be possible to set criteria to achieve quality. The **Capability Maturity Model Integration (CMMI)** and the **ISO 9001 Standard** are based on this view.

The main issue in this view is that of **satisfiability**. It is possible to have criteria that are not objective, and criteria that are not be quantifiable.

THE ECONOMICS-BASED VIEW OF QUALITY

In an economics-based view of quality, the notion of **quality is benefit for cost**. In other words, the reason for achieving quality and efforts towards improving quality is driven by profitability.

The main issue in this view is that of **complacency**. If sales of a software system are meeting an organization's target, then initiatives towards improving quality may subside.

Indeed, as pointed out in [Lahman, 2011]:

> A keynote speaker at a conference in 1995 asked everyone who used Microsoft Windows to raise their hand. Several hundred people did.
>
> He then asked all those whose system had crashed in the past week to keep their hands up. All but about a half dozen kept their hands up.
>
> The keynoter then asked everyone to keep their hands up who thought that Microsoft would go out of business as a result. All but a dozen hands went down.
>
> The keynoter used this to validate his assertion that **good quality in software was not necessarily a requirement that was important to a software company's survival**.

THE USER-BASED VIEW OF QUALITY

In a user-based view of quality, the notion of **quality is satisfaction**. In other words, the quality of a software system is equated to quality of that software system as perceived and experienced by its users, that is, quality of use.

The main issue in this view is that of **feasibility**. This is especially for software systems with a large number of users. It **may not be possible to satisfy all users**, or possible to check that every user is satisfied. Furthermore, it is known that users are not always right.

5.2. PERSPECTIVES OF QUALITY: APPROACH 2

The **Taguchi Approach**, originally developed for manufacturing processes, defines quality in terms of **loss imparted to society** by a product after delivery of the product to the end user.

The primary **advantage** of the Taguchi Approach to software engineering is the ability to quantify quality in **monetary terms** [Akingbehin, 2005; Jagannathan, Bhattacharya, Matawie, 2005].

SOFTWARE PRODUCT QUALITY AND 'LOSS IMPARTED TO SOCIETY' AS COST

The following are some examples of how software product quality can be related to (qualitative) loss to society via (quantitative) cost [Akingbehin, 2005]:

- **Cost of New Product.** A software product that is so unsatisfactory that it has to be replaced with a new product. The loss to society is the cost of replacing the software product; primarily the cost of the new product.

- **Cost of the Lost Time.** A software product that is so difficult to use that it results in loss of time and consequently loss of productivity. The loss to society is the labor cost of the lost time.

- **Cost of the Extra Learning Time.** A software product that has such a long learning curve that considerable time is lost in learning to use the product. The loss to society is the labor cost for the extra learning time.

- **Cost of Fixing the Software.** A software product that fails frequently and requires fixing at each failure. The loss to society is the cumulative cost of fixing the software.

- **Cost of Recovering and Continuing Work.** A software product that crashes frequently but does not require fixing at each failure. The loss to society is the labor cost that is equivalent to the cumulative time to recover from each crash and continue with the work that was interrupted.

- **Cost of Restoring Trust.** A software product that has such severe shortcomings that its use has brought serious harm. The loss to society is the cost of reestablishing reputation and restoring trust in the market.

6. DEFINITION OF SOFTWARE PRODUCT QUALITY

Definition [Software Quality] [ISO/IEC/IEEE, 2010]. [The] capability of a software product to satisfy **stated and implied** needs when used under specified conditions.

It could be noted that in this definition software quality is considered **synonymous** with software product quality, even though there are other aspects of a software system.

7. SOFTWARE PRODUCT QUALITY MODELS

The notion of 'quality' is rather **general** to be analyzed and assessed. Furthermore, over the years, it has been acknowledged that there are a number of aspects of quality that need to be made **explicit**. This can be done by **characterizing** quality in some manner [Ruhe, Wohlin, 2014, Chapter 6] and **expressing** quality through a quality model.

The purpose of a **quality model** is to provide (1) theoretical contribution (to **conceptualize quality**, thereby making the notion of quality understandable [Shekhovtsov, 2011]), and (2) practical contribution (to **operationalize quality**, thereby making the notion of quality practical [Ruhe, Wohlin, 2014, Chapter 6]).

It could be noted that a quality model can be **deterministic or probabilistic** [Bakota, Hegedüs, Kortvelyesi, Ferenc, Gyimóthy, 2011]. In this document, the attention is exclusively on deterministic quality models.

7.1. DEFINITION OF QUALITY MODEL

There are a number of definitions of quality model, including the following:

Definition [Quality Model] [ISO/IEC, 2007b]. A defined set of characteristics, and of relationships between them, which provides a framework for specifying **quality requirements** and **evaluating quality**.

Definition [Quality Model] [Deissenboeck, Juergens, Lochmann, Wagner, 2009]. A model with the objective to describe, assess and/or predict quality.

7.2. MOTIVATION FOR A QUALITY MODEL

There are a number of perceived reasons for formulating a quality model. The following is an **adaptation and extension** of the reasons given in [Heston, Phifer, 2011]:

- **Awareness.** A quality model that is published openly is a concrete contribution to the subject of quality. This, for example, could create awareness among those committed to quality.

- **Motivation.** A quality model provides structure to (otherwise abstract and perhaps even elusive) concept of quality. This, for example, could provide an incentive (and, hopefully, interest) for people to pursue quality-related endeavors.

- **Consistency.** A quality model serves as a reference point. This, for example, can help towards a consistent use of the meaning of quality attributes across different software project artifacts.

- **Repeatability.** A quality model can be used repeatedly across multiple software projects.

- **Communication.** A quality model provides a means that stakeholders can use to communicate about quality.

7.3. CLASSIFICATION OF SOFTWARE PRODUCT QUALITY MODELS

There are a number of ways of classifying software product quality models.

SCHEME 1

The software product quality models could be **generic or specific**, as shown, for example, in Figure 7.

SCHEME 2

The software product quality models mentioned previously could be placed within the **type of view of software product quality**, specifically **product-based view, manufacturing-based view, or user-based view**, of quality discussed earlier.

SCHEME 3

The software product quality models mentioned previously could be placed within the **type of commitment to software product quality**, specifically **definition, assessment, and prediction** of quality.

These dimensions can be used to classify quality models [Deissenboeck, Juergens, Lochmann, Wagner, 2009; Reddy, Ramasree, 2012]. For example, ISO/IEC 9126-1 Quality Model and ISO/IEC 25010 Quality Model are **definition-based software product quality models**.

SCHEME 4

The software product quality models mentioned previously could be placed within the **type of approach used for software product quality modeling**. This can be used to classify software product quality models as direct or indirect. For example, a **meta-model for software product quality** is an indirect approach [Wagner, 2013, Section 2.1.1].

The advantage of a meta-model is that it provides a means to derive **customized, context- and organization-specific, software product quality models**.

SCHEME 5

The software product quality models mentioned previously could be placed within the **type of approach for improving software product quality**, specifically **process-oriented, product-oriented, and hybrid** (of process-oriented and product-oriented) [Ferenc, Hegedűs, Gyimóthy, 2014, Page 66].

The basic idea of process-oriented (or hybrid) software quality models is that improvement of software process (quality) leads to improvement of software product (quality).

Figure 3 shows that a hybrid software quality model, structured as a **cause-effect diagram** (also known as a **fishbone diagram**).

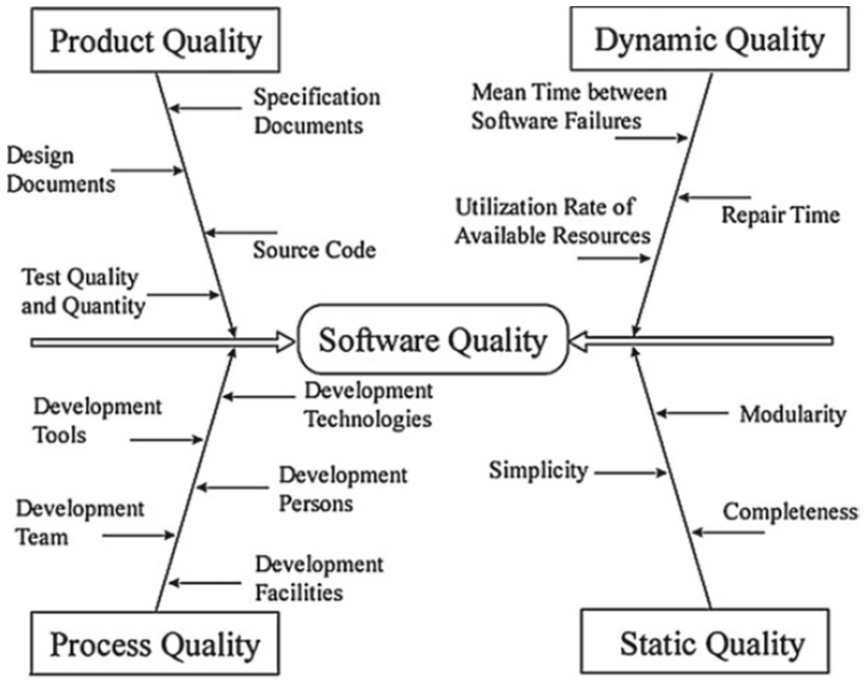


Figure 3. The software quality is a result of process quality and product quality. (Source: [Yamada, 2014].)

7.4. CHARACTERISTICS OF SOFTWARE PRODUCT QUALITY MODELS

There are certain **notable** characteristics of software product quality models, although the argument can be **generalized** to quality models for other aspects of software, such as process.

DECOMPOSITION

The notion of quality is decomposed into a number of **quality attributes**. These quality attributes may in some cases be decomposed further. For example, maintainability, security, or usability, can be such a quality attribute.

The quality attributes are a kind of **concerns** [Glinz, 2007].

The **(software product) quality attributes form a basis for quality requirements**. The quality requirements, in turn, are a **subset** of **non-functional requirements (NFRs)**[1] [Chung, Nixon, Yu, Mylopoulos, 2000; Adams, 2015].

There is **no** 'standard', closed-form, list of relevant quality attributes applicable to **every** software system. The relevancy of a quality attribute varies with respect to the **type** of software system [Cesare, Xiang, 2012]. In general, the **understanding of relevant NFRs** is, for example, critical for many software project artifacts, including software architecture description.

For example [Suryn, 2014, Section 1.2.3], for **Network Management Systems**, the most relevant quality attributes are **fault tolerance, interoperability, and operability**; for **Telecommunication Systems**, the most relevant quality attributes are **functionality, reliability, usability, and efficiency**; and for **Decision Support Systems**, the most relevant quality attributes are **accuracy, analyzability, and suitability**.

The decomposition lends a **hierarchical structure** to the quality model, as for example, shown in Figure 4. (The structure in some cases is a **tree**, but, more generally, it should be a **graph** as, for example, in [Chung, Nixon, Yu, Mylopoulos, 2000; Ferenc, Hegedűs, Gyimóthy, 2014, Page 81].)

[1] The term non-functional requirement has been **controversial** in requirements engineering [Glinz, 2007].

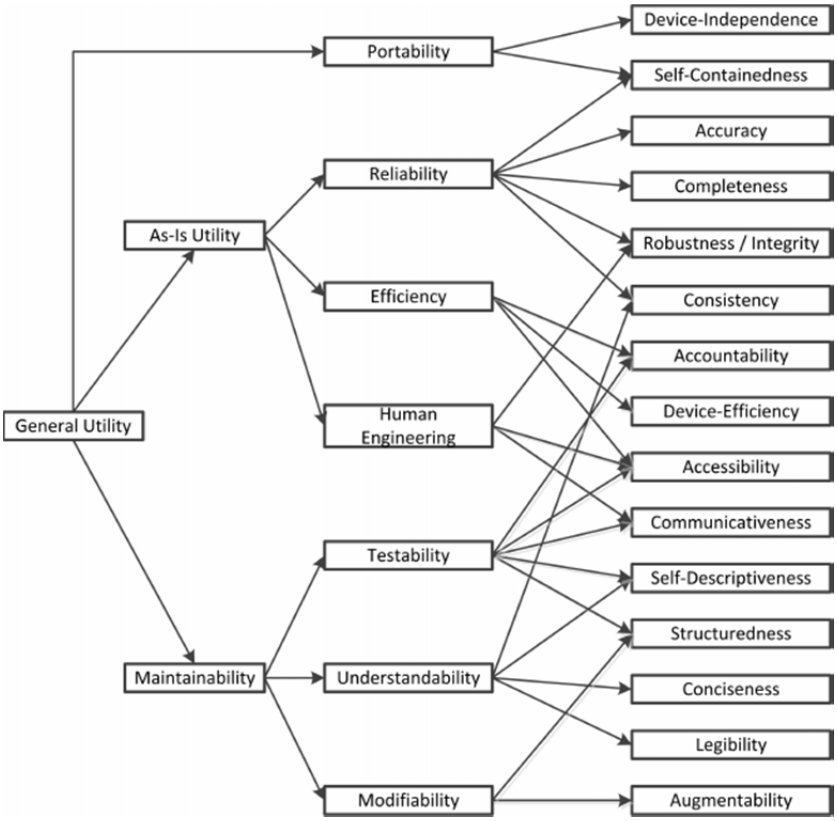


Figure 4. The shape of the Boehm-Brown-Lipow's Quality Model. (Source: [Boehm, Brown, Lipow, 1976].)

ORGANIZATION

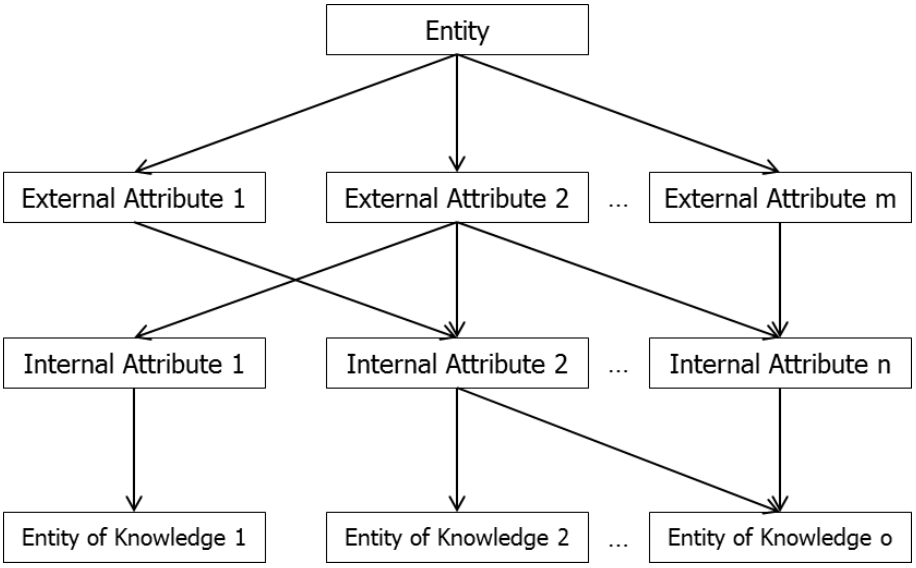
The collections of quality attributes may be **grouped** in some manner. For example, this is the case with the McCall-Richards-Walters's Quality Model.

ASSOCIATION

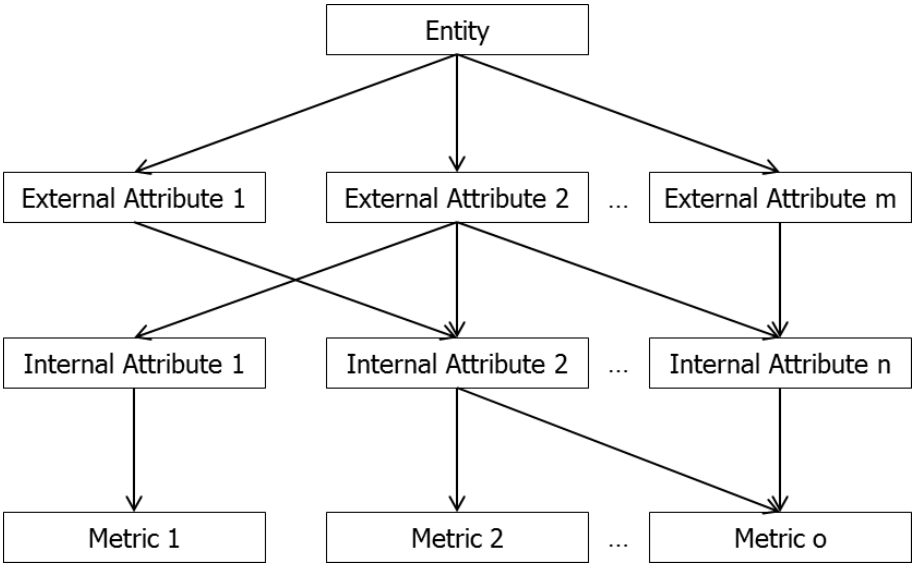
Figure 5 illustrates an abstract software product quality model.

The decomposition of a quality model leads to certain quality attributes being **'leaves'**. These **'leaves'** may be associated with some **entity of knowledge**.

The currently most common entity of knowledge is **metrics**. For example, this is the case with the Boehm-Brown-Lipow's Quality Model, the McCall-Richards-Walters's Quality Model, Boehm-Brown-Kaspar-Lipow-MacCleod's Quality Model, the SATC Quality Model, and the ISO/IEC 9126-1 Quality Model. It is also possible to associate other kinds of entities of knowledge, such as **patterns** [Kamthan, 2004].



(a)



(b)

Figure 5. An abstract software product quality model organized as a hierarchy.

In general, the mapping between the set of quality attributes and the set of entities of knowledge is **many-to-many**.

7.5. "UNDERSTANDING" VERSUS "X" OF A QUALITY ATTRIBUTE

The **understanding** of a quality attribute, and the **means** to achieve it or assess it, are **different concerns**, and should be **separated**.

For example, consider the case of security. It is important to **separate** the **purpose** of security and the **means** to achieve or assess it. For example, **Basic Authentication** (purpose) and **Username/Password** (means) are different things.

The means of achieving or assessing a particular quality attribute also depends on the underlying software development methodology being followed. For example, in **agile methodologies**, the common means of achieving or assessing a software product quality are **Refactoring**, **Continuous Integration**, and **Test-Driven Development (TDD)** [Hazzan, Dubinsky, 2014, Chapter 9; Mistrik, Bahsoon, Eeles, Roshandel, Stal, 2014, Chapter 1].

7.6. CHALLENGES IN FORMULATING A QUALITY MODEL

The **origin** of the challenges in formulating a quality model is significant variations in context.

DOMAIN

It is known that every software system has a **problem (application) domain**. The domain of a software system determines, in part, the kind of quality attributes that are relevant for that software system.

For example, quality attributes of a banking mobile application are different from a safety-critical cloud application.

It is therefore natural that the quality models for different domains can be different.

STAKEHOLDERS

The following is one of the goals of a software product quality model [Dromey, 1995]:

> "The first task in building a software product quality model is to identify what the **intended applications of the model** are and to address the needs of the **different interest groups** that will use the model in **different applications**."

This goal may be difficult to satisfy as **stakeholder interests vary broadly**.

ARTIFACTS

There are different types of software project artifacts (documents or models) being produced at different phases of a software process.

For example, realization of a software process may lead to the use case model, the software requirements specification (SRS), and the test suite.

The list of relevant quality attributes across different types of software project artifacts can be different. It is therefore natural that the quality models for different types of software project artifacts can be different.

7.7. A COLLECTION OF QUALITY MODELS

In the last few decades, a number of software product quality models have been proposed, as **surveyed** in [Kitchenham, Pfleeger, 1996; Albin, 2003, Chapter 8; Khosravi, Guéhéneuc, 2004; Lundberg, Mattsson, Wohlin, 2005, Chapter 1; Côté, Suryn, Georgiadou, 2007; **Naik, Tripathy, 2008, Chapter 17; Al-Qutaish, 2010;** Samadhiya, Wang, Chen, 2010; Reddy, Ramasree, 2012; Ferenc, Hegedűs, Gyimóthy, 2014].

The prominent software product quality models, listed chronologically, include the following:

- **Boehm**-Brown-Lipow's Quality Model [Boehm, Brown, Lipow, 1976].
- **McCall**-Richards-Walters's Quality Model [McCall, Richards, Walters, 1977].
- **Boehm**-Brown-Kaspar-Lipow-MacCleod's Quality Model [Boehm, Brown, Kaspar, Lipow, MacCleod, 1978].
- **Goal-Question-Metric (GQM)** Quality Model [Basili, Weiss, 1984].
- FURPS Quality Model[2] [Grady, 1992].

[2] **FURPS** is an acronym for **Functionality, Usability, Reliability, Performance, and Supportability**. The FURPS quality model, now known as FURPS+ quality model [Eeles, 2005], was originally developed at Hewlett-Packard [Grady, 1992].

- DeGrace-Stahl Quality Model [DeGrace, Stahl, 1993; Albin, 2003, Chapter 8].
- Dromey's Quality Model [Dromey, 1995].
- The Software Assurance Technology Center (**SATC**) Quality Model [Hyatt, Rosenberg, 1996].
- **ISO/IEC 9126-1 Quality Model [ISO/IEC, 2001].**
- ISO/IEC 9000-3 Quality Model [ISO/IEC, 2004].
- SQUALE [Mordal-Manet, Balmas, Denier, Ducasse, Wertz, Laval, Bellingard, Vaillergues, 2009].
- **ISO/IEC 25010 Quality Model [ISO/IEC, 2011].**

REMARKS

There are comprehensive **historical accounts of software product quality models** in [Wagner, 2013, Section 2.1.1; Ferenc, Hegedűs, Gyimóthy, 2014].

GENEALOGY DIAGRAMS

Figure 6 presents a **genealogy diagram** of software product quality models. It also traces the history and evolution of program quality, software quality, and, to a certain extent, software metrics.

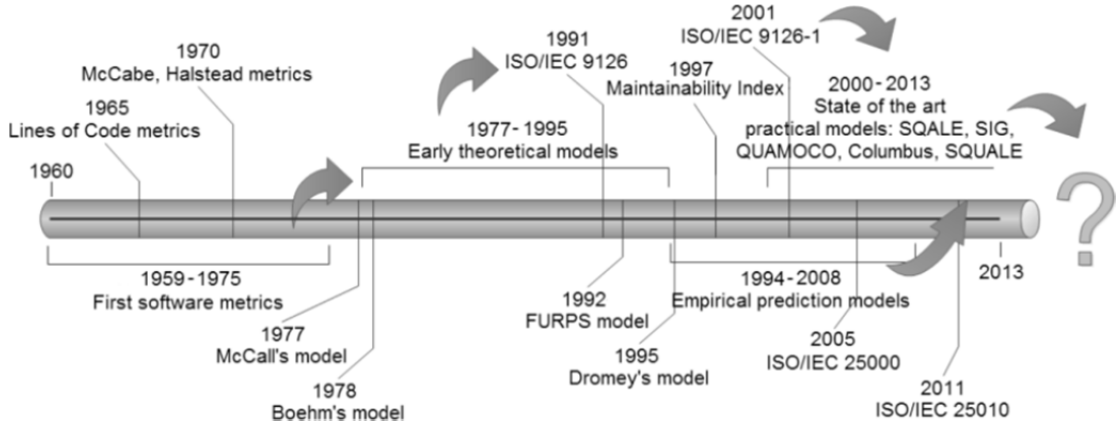


Figure 6. Evolution of Software Product Quality Models. (Source: [Ferenc, Hegedűs, Gyimóthy, 2014, Page 68].)

Figure 7 presents a **genealogy diagram** of software product quality models proposed relatively recently. It also provides a brief insight into the history and evolution of software product quality models, and a **classification** of software product quality models.

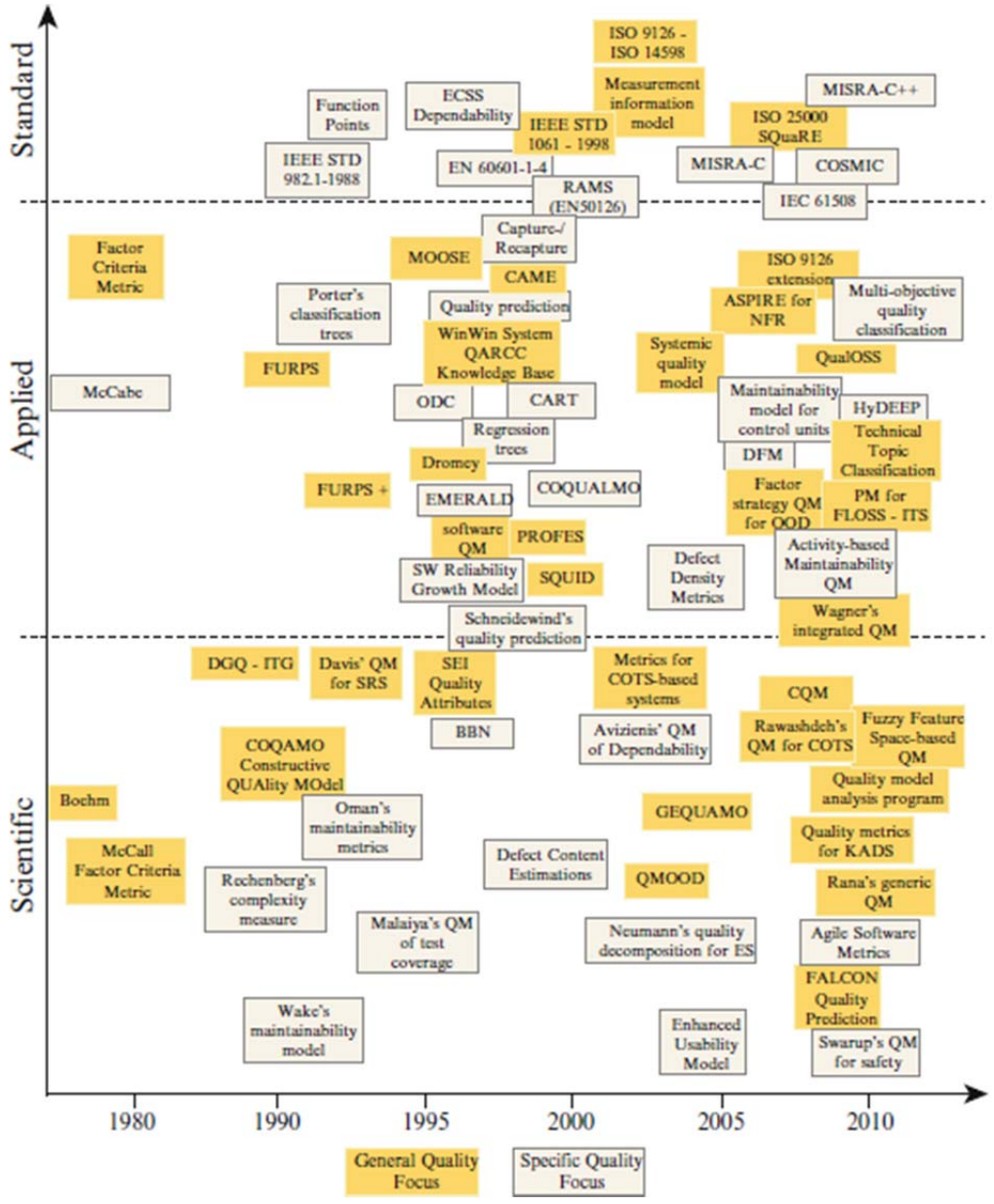


Figure 7. Evolution of Software Product Quality Models. (Source: [Ruhe, Wohlin, 2014, Chapter 6, Page 131].)

7.8. SELECTING A SOFTWARE PRODUCT QUALITY MODEL

There are multiple software product quality models available. It is important that the selection of a suitable software product quality model have a **proper basis** [Ruhe, Wohlin, 2014, Section 6.2]. This basis usually manifests in form of criteria, such as the following:

- Domain of the Software Project Artifact
- The Software Project Artifact to be Examined
- Purpose of Use of the Software Product Quality Model
- Relevant Quality Attributes of the Software Project Artifact
- Relevant Stakeholders for Software Product Quality
- **Goal of the Software Product Quality Model**
- Tool Support for the Quality Attributes in the Software Product Quality Model

7.9. CHALLENGES IN USING SOFTWARE PRODUCT QUALITY MODELS

In theory, many quality models, including those in current international standards, aim to be abstract and general enough to be applicable for all kinds of software systems. However, in practice, there are a number of challenges in applying these models sustainably [Moody, 2005; Ruhe, Wohlin, 2014, Chapter 6].

- **Challenge 1: Universality.** There is no single software product quality model that can be applied in every environment and is adopted universally (by all organizations) [Hazzan, Dubinsky, 2014, Chapter 9; Ruhe, Wohlin, 2014, Chapter 6].

- **Challenge 2: Need for Tailoring.** To be broadly applicable, the quality models in current international standards are expected to be independent of any organization. However, those standards also do not provide directions for integrating the quality models in the decision-making processes of any particular organization, and therefore it may not be possible to use the quality models as-is. To be used effectively and to avoid the consequences of a mismatch, the quality models may need to be tailored (that is, modified) to organization specifics and supported by corresponding tools.

- **Challenge 3: Cost-Effectiveness.** The use of a quality model entails cost. For example, there is cost of allocating resources, personnel, infrastructure, data collection, and so on. This cost may or may not be feasible to an organization, meaning the return on investment (ROI) for committing to a particular quality model may or may not be acceptable to an organization [Jagannathan, Bhattacharya, Matawie, 2005].

7.10. SOFTWARE PRODUCT QUALITY MODELS IN PRACTICE

In theory, quality models are expected to be useful. However, in practice, the **understanding and usefulness** of quality models can **vary** [Kläs, Lampasona, Nunnenmacher, Wagner, Herrmannsdörfer, Lochmann, 2010].

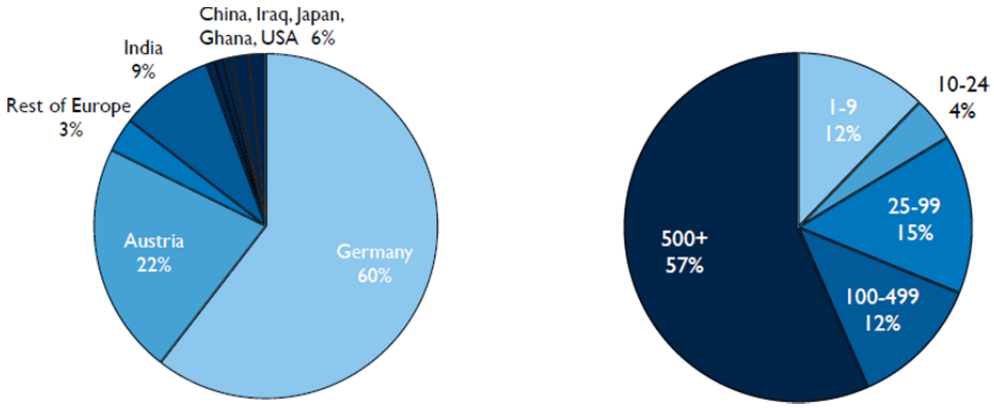
There are a number of approaches in **empirical software engineering**, one of which is a survey. A survey is a means to assess the current state-of-the-practice of something of interest.

In [Wagner, Lochmann, Winter, Goeb, Klaes, Nunnenmacher, 2012], the results of a **survey to examine the current practice of software quality models** is presented.

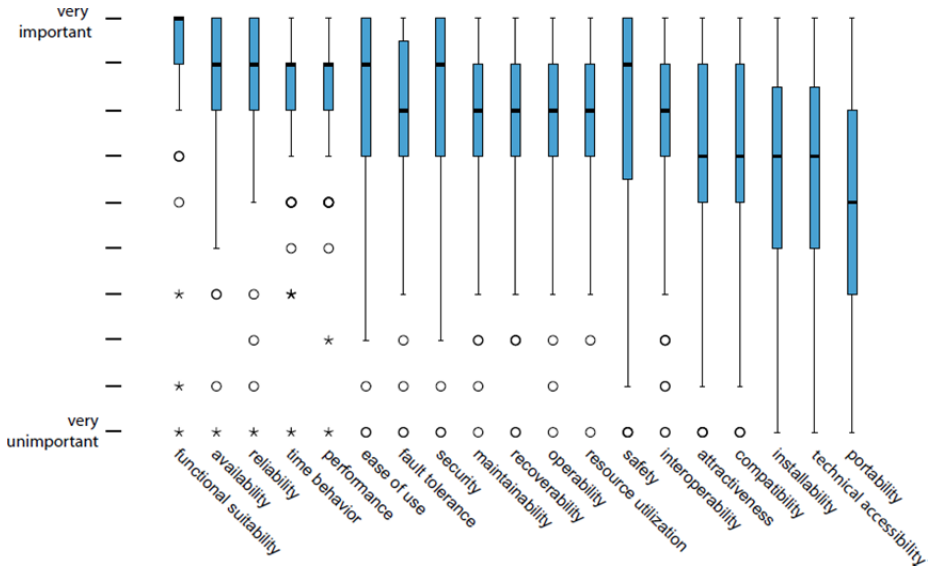
The purpose of the survey was:

1. To identify the main classes of quality models that are currently used.
2. To identify the quality attributes that are considered important.
3. To identify the quality assurance approaches that are applied with these quality models.
4. **To identify problems that are related to these quality models** (and, in doing so, suggest potential directions of improvement).

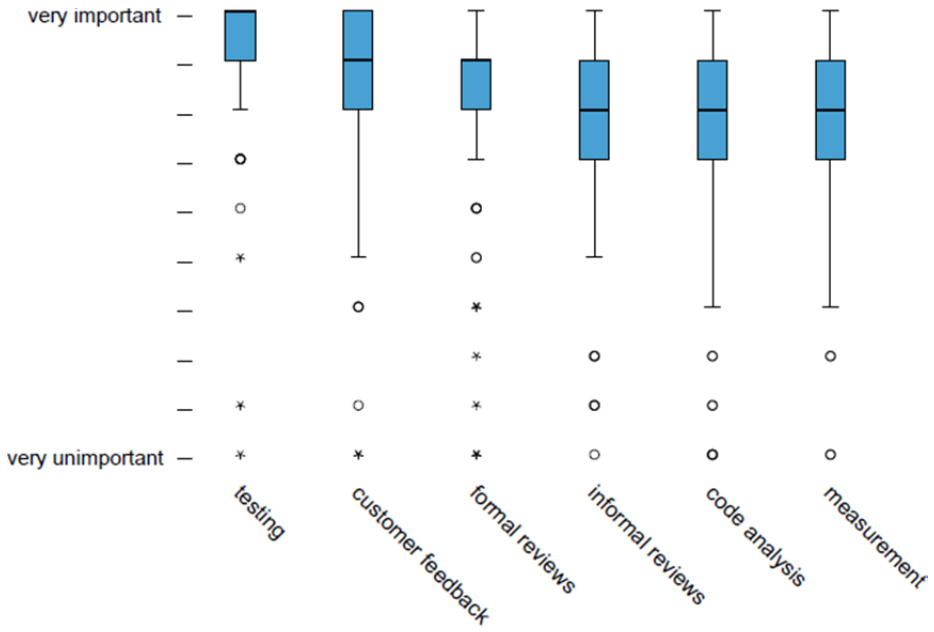
The salient results of the survey are given in Figure 8.



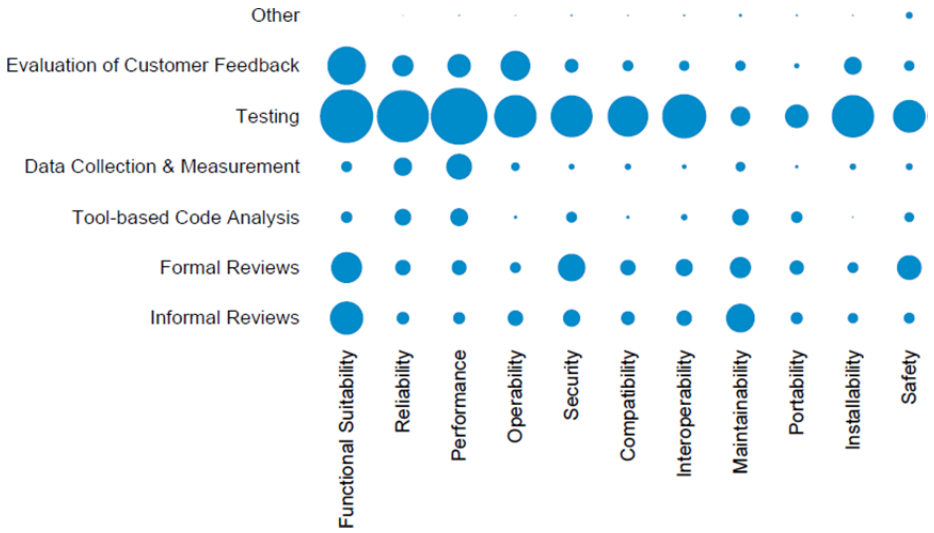
(a)


very important
very unimportant
functional suitability
availability
reliability
time behavior
performance
ease of use
fault tolerance
security
maintainability
recoverability
operability
resource utilization
safety
interoperability
attractiveness
compatibility
installability
technical accessibility
portability

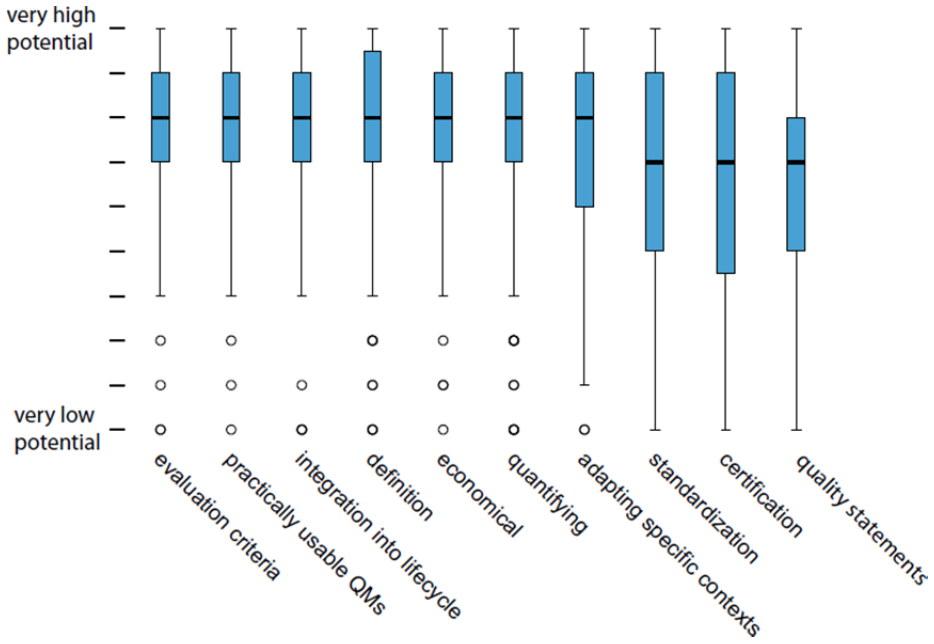
(b)


very important
very unimportant
testing
customer feedback
formal reviews
informal reviews
code analysis
measurement

(c)



(d)



(e)

Figure 8. (a) The distribution of survey respondents, by country and by the number of employees. (b) The perceived significance of quality attributes. (c) The perceived significance of approaches for assuring quality. (d) The perceived significance of approaches per quality attribute. (e) **The suggested directions of improvement for quality models.** (Source: [Wagner, Lochmann, Winter, Goeb, Klaes, Nunnenmacher, 2012].)

8. SOFTWARE PRODUCT QUALITY META-MODEL

It is **unlikely** that a **single** quality model will ever be sufficient for **all** software systems. This sentiment is echoed[3] by the **Quamoco Consortium**[4]:

> "The diversity of software, ranging from embedded systems to mainframe applications, from entertainment systems to highly safety-critical control systems, **prevents** the establishment of a quality standard that covers **all** areas without exception."

However, there are efforts in the direction of a quality **meta-model** [Albin, 2003, Chapter 8; Kläs, Lampasona, Nunnenmacher, Wagner, Herrmannsdörfer, Lochmann, 2010; Lochmann, Wagner, Goeb, Kirchler, 2010; Lochmann, Goeb, 2011], that is, basic concepts that should entail **any** quality model.

Definition [Quality Modeling Framework] [Wagner, 2013]. A framework to define, evaluate and improve quality. This usually includes a **quality meta-model** as well as a methodology that describes how to instantiate the meta-model and use the model instances for defining, assessing, predicting and improving quality.

Definition [Meta-Model]. A model for defining models.

Definition [Quality Meta-Model] [Deissenboeck, Juergens, Lochmann, Wagner, 2009]. A model of the constructs and rules needed to build specific quality models.

REMARKS

A meta-model is an **intensional definition** of a modeling language. It specifies the **abstract syntax of the language** [Ferenc, Hegedűs, Gyimóthy, 2014].

The **Quamoco Consortium** consisting of research institutions and companies has developed a quality standard [for] software products made in Germany.

[3] URL: https://quamoco.in.tum.de/wordpress/?lang=en .
[4] URL: https://quamoco.in.tum.de/wordpress/?lang=en .

The aim of the project is "to develop a quality standard applicable in practice that can be used to objectively define, review, and permanently maintain software quality." It is responsible for the development of the **Quamoco: Open Quality Model and Tool Support for Quality Modelling and Evaluation**[5].

9. RELATIONSHIPS AMONG SOFTWARE PRODUCT QUALITY ATTRIBUTES: 'A BALANCING ACT'

If you **produce a software system that has terrible quality, you lose** because no one will want to buy it. If on the other hand **you spend infinite time, extremely large effort, and huge sums of money to build the absolutely perfect piece of software**, then it's going to take so long to complete and it will be so expensive to produce that **you'll be out of business** anyway. Either you missed the market window, or you simply exhausted all your resources.

So people in industry try to get to that **magical middle ground** where the **product is good enough not to be rejected** right away, such as during evaluation, but also **not the object of so much perfectionism** and so much work that it would take **too long or cost too much to complete**.
— Bertrand Meyer

Teoriya Resheniya Izobretatelskikh Zadatch (TRIZ) is a "Theory of Inventive Problem Solving" [Altshuller, 1994]. It follows from **TRIZ Technical Contradiction** that more of something desirable also brings more of something less desirable, or less of something else also desirable.

The (software product) **quality attributes are not necessarily mutually exclusive** [Wiegers, 2003; Berander, Damm, Eriksson, Gorschek, Henningsson, Jönsson, Kågström, Milicic, Mårtensson, Rönkkö, Tomaszewski, 2005; Wiegers, Beatty, 2013; Sangwan, 2015, Section 2.6]. Indeed, they can affect or impact each other in one of the following manner: **positive (+), negative (–), or neutral ()**.

- **Positive.** A positive (+) relationship means that changing one attribute affects the other positively.

- **Negative.** A negative relationship (–) means that changing one attribute affects the other one negatively.

- **Neutral.** An empty () relationship means that the attributes are independent of each other.

[5] URL: http://www.quamoco.de/ .

REMARKS

The aforementioned scheme can be **granularized** even further [Chung, Nixon, Yu, Mylopoulos, 2000]. For example, one could have **very positive (++)** and **very negative (– –)**.

9.1. A REPRESENTATION OF RELATIONSHIPS AMONG SOFTWARE PRODUCT QUALITY ATTRIBUTES

THE ABSTRACT CASE

If the collection of (software product) quality attributes is placed on a **grid (matrix)** for the sake of **pairwise comparison**, then the relationships between them appear as in Figure 9.

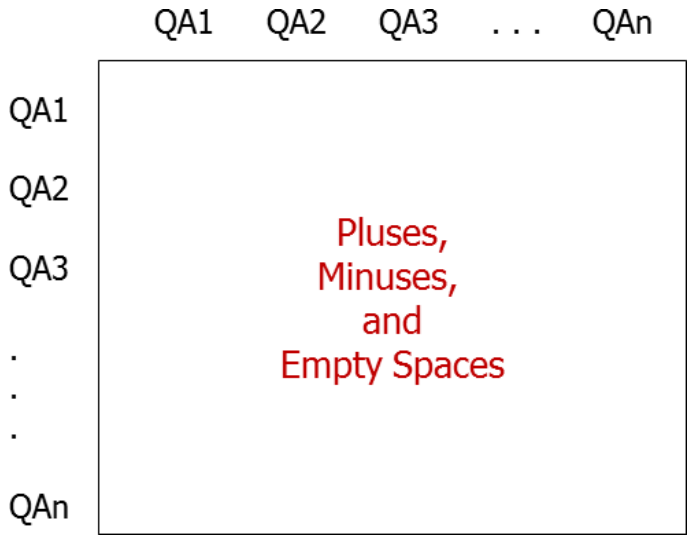
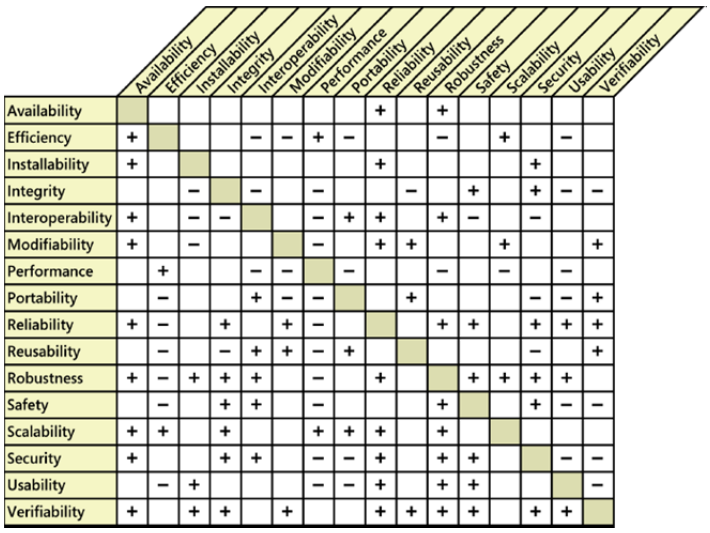


Figure 9. An abstract matrix of relationships between (software product) quality attributes.

A CONCRETE CASE

The positive and negative relationships among selected quality attributes are shown in Figure 10.



	Availability	Efficiency	Installability	Integrity	Interoperability	Modifiability	Performance	Portability	Reliability	Reusability	Robustness	Safety	Scalability	Security	Usability	Verifiability
Availability								+		+						
Efficiency	+				−	−	+	−			−		+		−	
Installability	+								+					+		
Integrity			−		−		−			−		+		+	−	−
Interoperability	+		−	−			−	+	+		+	−		−		
Modifiability	+		−				−		+	+			+			+
Performance		+			−	−		−			−		−		−	
Portability		−			+	−	−			+				−	−	+
Reliability	+	−		+		+	−				+	+		+	+	+
Reusability		−		−	+	+	−	+						−		+
Robustness	+	−	+	+	+		−		+			+	+	+	+	
Safety		−		+	+		−				+			+	−	−
Scalability	+	+		+			+	+	+		+					
Security	+			+	+		−	−	+		+	+			−	−
Usability		−	+				−	−	+		+	+				−
Verifiability	+		+	+		+			+	+	+	+		+	+	

Figure 10. A concrete matrix of relationships between (software product) quality attributes. (Source: [Wiegers, Beatty, 2013, Chapter 14].)

OBSERVATIONS

- It is evident that the **main diagonal** of the matrix in Figure 10 would contain all **empty spaces**.

- The quality attribute matrix is **not symmetric**. For example, efforts towards **increasing security** may come at the cost of **decreasing usability**. (However, converse is **not** necessarily the case.)

REMARKS

There are matrices of relationships between quality attributes for **specific** quality models [Lundberg, Mattsson, Wohlin, 2005, Chapter 5; Wiegers, Beatty, 2013, Chapter 14].

9.1. CONSEQUENCES OF THE RELATIONSHIPS BETWEEN SOFTWARE PRODUCT QUALITY ATTRIBUTES

There are a number of consequences of the dependencies between (software product) quality attributes [Henningsson, Wohlin, 2002].

- **Prioritization of Quality Requirements.** The stakeholder perceptions of (software product) quality vary. Therefore, the significance associated with each of the (software product) quality attributes also vary. Thus, the (software product) quality attributes are prioritized differently by different stakeholders.

- **Assessment of Software Project.** It is not possible to optimize all (software product) quality attributes simultaneously. This means some stakeholders are likely to be **disappointed**. If these stakeholders are 'High' on the importance/influence scheme, then this could mean that the software project could be assessed as 'Challenged' or even 'Failed'.

10. QUALITY AND BUSINESS

10.1 BUSINESS GOALS AND SOFTWARE PRODUCT QUALITY

In simplest terms, product improvement is often reflected in the slogan **'better, faster, cheaper'**.

(It could be noted that 'faster' refers to the speed of developing the product, not the performance of the product itself. In practice, it is possible to achieve only one out of three, but that has not stopped people from advertising and claiming otherwise.)

These three things are often cited because they represent the **lowest common denominators** of most product improvement programs [Heston, Phifer, 2011] and represent variables that are easy to communicate to non-technical stakeholders.

However, successful product improvement programs are those that **tie improvement strategy to business objectives**. In other words, **business goals come first and software product quality comes second**.

For example, consider a company that is satisfied with product or service quality, speed, and cost, and is now looking to expand into **new geographical locations**. Then, instead of focusing on improving the product or service, the business objective to expand might lead the company to focus on **improving sales effectiveness or distribution network** [Heston, Phifer, 2011].

10.2. SOFTWARE PRODUCT QUALITY TRADE-OFF: YIN AND YANG

> **Yin and yang** can be thought of as **complementary** (instead of opposing) forces interacting to form a **holistic dynamic system** (that is, the whole is greater than the sum of parts). In Taoist metaphysics, **good-bad distinctions** and other **dichotomous moral judgments** are **perceptual**, not real; so, yin-yang is an indivisible whole.
>
> Everything has both yin and yang aspects. (For example, shadow cannot exist without light.)
>
> Either of the two major aspects may manifest **more strongly** in a particular object, depending on the criterion of the observation. (Source: Wikipedia.)

Figure 11 shows the **Quality Triangle**. It implies that it is necessary to make **compromises** among 'high quality', 'low cost' and 'short time to market[6]', and that it is usually only possible to achieve **one out of three**.

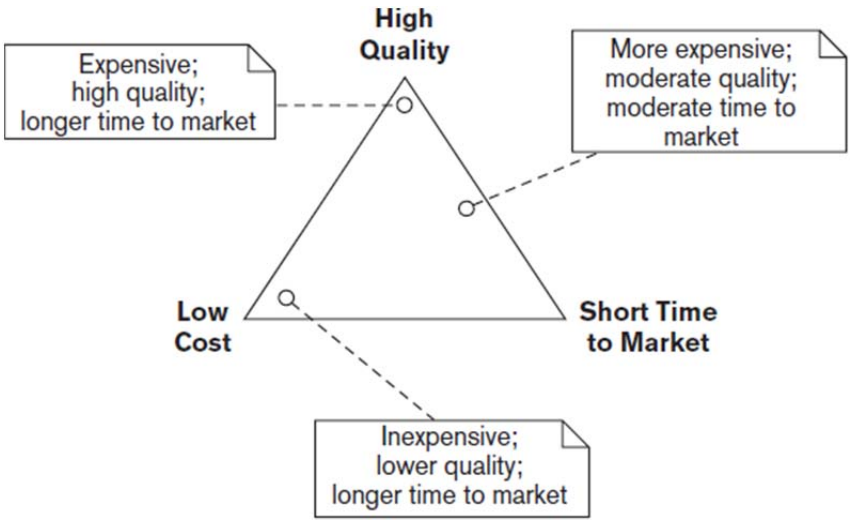


Figure 11. The Quality Triangle. (Source: [Rozanski, Woods, 2005, Figure 2-4].)

[6] In commerce, the **time to market** is the length of time it takes from a product being conceived until its being available for sale [Wikipedia].

For example, a ‘high quality’ system tends to take longer to develop and costs more; conversely, it is possible to reduce the initial development time, but that comes at the expense of reducing the quality of the delivered software, assuming cost is kept almost constant.

It is likely that one or more of these dimensions are important to different stakeholders. A **software architect** needs to understand which of these dimensions is important to whom, and to reach an acceptable **compromise** when necessary.

REMARKS

It is possible to have similar ‘triangles’, including the **Iron Triangle**[7], for **trade-offs** among different elements of a software project [Lundberg, Mattsson, Wohlin, 2005, Section 3.2].

11. CHALLENGES TOWARDS IMPROVING SOFTWARE PRODUCT QUALITY

In theory, improvement of software product quality is a necessity. In fact, doing so may even be contractual. However, in practice, there are a number of challenges.

11.1. CHALLENGES ORIGINATING FROM THE NATURE OF SOFTWARE

- **Lack of Tangibility.** It is evident that software is intangible. Unlike physical products, including hardware, there is **no sensory feedback** of ‘broken software’ [Wagner, 2013, Section 1.1].

[7] URL: http://www.ambysoft.com/essays/brokenTriangle.html .

- **Non-Trivial Isolation of Root Cause.** It is not always straightforward to pinpoint the root cause of software-related problem. This is because a software system usually needs several other entities to function properly, including hardware, operating system, network protocols, compiler, and so on. For example, it is sometimes not obvious to identify the origin of even the HTTP Status Code 404 (Not Found).

11.2. CHALLENGES ORIGINATING FROM THE NATURE OF HUMANS

It is **not automatic** that initiatives towards improving quality will be adopted and, if adopted, will be successful [Albin, 2003, Chapter 8].

Indeed, based on an empirical study[8], it has been pointed out [Streit, Pizka, 2011] that quality improvement often **fails** as a result of **one or more** of the following **reasons**:

- **Negligence.** The quality-related tasks are neglected due to the needs of daily business or simple laziness.

- **Prohibition.** The improvement is boycotted openly or in secrecy.

- **Hypocrisy.** The quality management processes are applied only formally (the real problems are not reported, all indicators remain green in the management overview).

- **Economics.** The economic value of quality improvements is not visible, and projects are stopped by management or not even approved in the first place.

- **Transience.** The effects last only a **short time**, and both the way of working and the resulting quality slowly drop back to their previous levels.

Figure 12 points out the **current challenges towards quality improvement**, and **means to overcome them**.

[8] The study was carried out by the employees of **itestra GmbH**, and the results are based on more than 50 quality analysis and quality improvement projects in mission-critical software systems of European companies.

Challenge / Best Practice	Provide a Business Case	Obtain Management Support	Set Common Goals	Respect the Context	Respect Employees' Privacy	A Benefit for Everyone	Be the Developer's Friend	Look Forward, Not Back	Present Intermediate Results	Adapt the Argumentation to the Audience	Provide Different Levels of Detail	Use Both Metrics and Code Samples	Provide Benchmarks	Prepare, Filter and Highlight Data	Explain the Effects	Do Not Be Smart	Provide Action Hooks	Prepare Answers for Frequent Questions	Schedule Action Early	Demonstrate Improvement	Start Small	Provide Enough Budget	Provide Independent QA and Support	Provide Fast Feedback
Fear of Being Rated			x	x	x	x	x	x	x	x				x				x						
Feared Loss of Reputation				x		x		x								x		x						
The "Fortress"				x		x		x										x						
Fear of Increased Workload	x	x	x	x		x	x			x					x		x	x		x	x	x	x	
Idleness				x		x	x					x	x		x		x			x	x		x	x
Lack of Trust in Assessment			x	x					x	x	x	x						x		x	x			x
Lack of Understanding				x					x	x	x	x		x	x		x	x		x				x
Inability to Correct or Do Better				x			x		x	x		x					x	x		x				
Focus on Unimportant Quality Aspects			x	x					x	x			x	x	x			x		x				
Silent Boycott[1]		x		x					x										x	x		x	x	
The Inerrable Expert	x	x		x					x	x		x	x		x			x	x	x				x
Putting Forward Mock Reasons	x	x	x	x					x	x		x	x		x	x		x	x	x				x
Adaption to a Metric				x													x						x	
Denying an Assessment's Value	x	x	x	x								x	x		x			x						x
Missing Business Case	x														x				x		x	x		x
Too Many False Positives														x							x	x	x	
Too Much Data											x			x							x			
Self-Confidence of Assessors		x		x														x	x	x				

Figure 12. The challenges and 'best' practices in the efforts towards the improvement of software product quality. (Source: [Streit, Pizka, 2011].)

12. QUALITY AND STANDARDS

There are a number of standards for quality, in general, and software product quality, in particular [IEEE, 2014]. In general, standards **aim** for better quality, and it is expected that they would do so.

However, the relationship between standards and quality is **equivocal** [Schneidewind, Fenton, 1996]. This is because, like other forms of knowledge, commitment to standards is contingent on their **appropriate use**.

Indeed, as part of the disclaimer of the **ISO/IEC 15939 Standard** [ISO/IEC, 2007a]:

> This standard is not intended to assure safety, security, health, or environmental protection in all circumstances.

ACKNOWLEDGEMENT

The inclusion of images from external sources is only for non-commercial educational purposes, and their use is hereby acknowledged. In particular, the author is grateful to Richard J. Kinch for the use of the ‘dangerous bend’ symbol.

REFERENCES

[Adams, 2015] Non-functional Requirements in Systems Analysis and Design. By K. M. Adams. Springer International Publishing. 2015.

[Akingbehin, 2005] Taguchi-Based Metrics for Software Quality. By K. Akingbehin. The Fourth Annual ACIS International Conference on Computer and Information Science (ICIS 2005). Jeju Island, South Korea. July 14-16, 2005.

[Albin, 2003] The Art of Software Architecture: Design Methods and Techniques. By S. T. Albin. John Wiley and Sons. 2003.

[Al-Kilidar, Cox, Kitchenham, 2005] The Use and Usefulness of the ISO-IEC 9126 Quality Standard. By H. Al-Kilidar, K. Cox, B. Kitchenham. The Fourth International Symposium on Empirical Software Engineering (ISESE 2005). Noosa Heads, Australia. November 17-18, 2005.

[Al-Qutaish, 2010] Quality Models in Software Engineering Literature: An Analytical and Comparative Study. By R. E. Al-Qutaish. The Journal of American Science. Volume 6. Number 3. 2010. Pages 166-175.

[Altshuller, 1994] And Suddenly the Inventor Appeared. By G. Altshuller. Technical Innovation Center. 1994.

[Bach, 1997] Good Enough Quality: Beyond the Buzzword. By J. Bach. Computer. Volume 30. Issue 8. 1997. Page 96-98.

[Bakota, Hegedüs, Kortvelyesi, Ferenc, Gyimóthy, 2011] A Probabilistic Software Quality Model. By T. Bakota, P. Hegedüs, P. Kortvelyesi, R. Ferenc, T. Gyimóthy. The Twenty Seventh IEEE International Conference on Software Maintenance (ICSM 2011). Williamsburg, U.S.A. September 25-30, 2011.

[Basili, Weiss, 1984] A Methodology for Collecting Valid Software Engineering Data. By V. R. Basili, D. M. Weiss. IEEE Transactions on Software Engineering. Volume SE-10. Number 6. 1984. Pages 728-738.

[Bauer, Adams, 2014] Service Quality of Cloud-Based Applications. By E. Bauer, R. Adams. John Wiley and Sons. 2014.

[Berander, Damm, Eriksson, Gorschek, Henningsson, Jönsson, Kågström, Milicic, Mårtensson, Rönkkö, Tomaszewski, 2005] Software Quality Attributes and Trade-Offs. By P. Berander, L.-O. Damm, J. Eriksson, T. Gorschek, K. Henningsson, P. Jönsson, S. Kågström, D. Milicic, F. Mårtensson, K. Rönkkö, P. Tomaszewski. Blekinge Institute of Technology. Ronneby, Sweden. June 2005.

[Boehm, Brown, Kaspar, Lipow, MacCleod, 1978] Characteristics of Software Quality. By B. W. Boehm, J. R. Brown, J. R. Kaspar, M. L. Lipow, G. MacCleod. Elsevier. 1978.

[Boehm, Brown, Lipow, 1976] Quantitative Evaluation of Software Quality. By B. W. Boehm, J. R. Brown, M. Lipow. The Second International Conference on Software Engineering (ICSE 1976). San Francisco, U.S.A. October, 13-15 1976.

[Buxton, 2007] Sketching User Experiences: Getting the Design Right and the Right Design. By B. Buxton. Morgan Kaufmann. 2007.

[Cesare, Xiang, 2012] Software Similarity and Classification. By S. Cesare, Y. Xiang. Springer Science+Business Media. 2012.

[Chung, Nixon, Yu, Mylopoulos, 2000] Non-Functional Requirements in Software Engineering. By L. Chung, B. A. Nixon, E. Yu, J. Mylopoulos. Kluwer. 2000.

[Côté, Suryn, Georgiadou, 2007] In Search for a Widely Applicable and Accepted Software Quality Model for Software Quality Engineering. By M.-A. Côté, W. Suryn, E. Georgiadou. Software Quality Journal. Volume 15. Number 4. 2007. Pages 401-416.

[Davis, 1994] Fifteen Principles of Software Engineering. By A. M. Davis. IEEE Software. Volume 11. Issue 6. 1994. Pages 94-96.

[DeGrace, Stahl, 1993] The Olduvai Imperative: CASE and the State of Software Engineering Practice. By P. DeGrace, L. Stahl. Yourdon Press. 1993.

[Deissenboeck, Juergens, Lochmann, Wagner, 2009] Software Quality Models: Purposes, Usage Scenarios and Requirements. By F. Deissenboeck, E. Juergens, K. Lochmann, S. Wagner. The Seventh Workshop on Software Quality (WOSQ 2009). Vancouver, Canada. May 16, 2009.

[Dromey, 1995] A Model for Software Product Quality. By R. G. Dromey. IEEE Transactions on Software Engineering. Volume 21. Number 2. 1995. Pages 146-162.

[Eeles, 2005] Capturing Architectural Requirements. By P. Eeles. IBM developerWorks. November 15, 2005.

[Emam, 2005] The ROI from Software Quality. By K. El Emam. Auerbach Publications. 2005.

[Fenton, Bieman, 2015] Software Metrics: A Rigorous and Practical Approach. By N. Fenton, J. Bieman. Third Edition. CRC Press. 2015.

[Ferenc, Hegedűs, Gyimóthy, 2014] Software Product Quality Models. By R. Ferenc, P. Hegedűs, T. Gyimóthy. In Evolving Software Systems. T. Mens, A. Serebrenik, A. Cleve (Editors). Springer-Verlag. 2014. Pages 65-100.

[Garvin, 1984] What does Product Quality Really Mean? By D. A. Garvin. MIT Sloan Management Review. Volume 26. Number 1. 1984. Pages 25-43.

[Glinz, 2007] On Non-Functional Requirements. By M. Glinz. The Fifteenth International Requirements Engineering Conference (RE 2007). New Delhi, India. October 15-19, 2007.

[Gotterbarn, 2010] Perfection is Not 'Good Enough': Beyond Software Development. By D. Gotterbarn. ACM Inroads. Volume 1. Number 4. 2010. Pages 8-9.

[Grady, 1992] Practical Software Metrics for Project Management and Process Improvement. By R. Grady. Prentice-Hall. 1992.

[Hazzan, Dubinsky, 2014] Agile Anywhere: Essays on Agile Projects and Beyond. By O. Hazzan, Y. Dubinsky. Springer. 2014.

[Henningsson, Wohlin, 2002] Understanding the Relations between Software Quality Attributes - A Survey Approach. By K. Henningsson, C. Wohlin. The Twelfth International Conference for Software Quality (ICSQ 2002). Ottawa, Canada. October 28-30, 2002.

[Heston, Phifer, 2011] The Multiple Quality Models Paradox: How Much 'Best Practice' is Just Enough? By K. M. Heston, W. Phifer. Journal of Software Maintenance and Evolution: Research and Practice. Volume 23. Number 8. 2011. Pages 517-531.

[Horch, 2003] Practical Guide to Software Quality Management. By J. W. Horch. Second Edition. Artech House. 2003.

[Hyatt, Rosenberg, 1996] A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality. By L. E. Hyatt, L. Rosenberg. The Eighth Annual Software Technology Conference. Salt Lake City, U.S.A. April 21-26, 1996.

[IEEE, 2014] Guide to the Software Engineering Body of Knowledge (SWEBOK) Version 3.0. IEEE Computer Society. 2014.

[ISO/IEC, 2001] ISO/IEC 9126-1:2001. Software Engineering -- Product Quality -- Part 1: Quality Model. The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC). 2001.

[ISO/IEC, 2004] ISO/IEC 90003:2004. Software Engineering -- Guidelines for the Application of ISO 9001:2000 to Computer Software. The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC). 2004.

[ISO/IEC, 2007a] ISO/IEC 15939:2007. Software Engineering -- Software Measurement Process. The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC). 2007.

[ISO/IEC, 2007b] ISO/IEC 25030:2007. Software Engineering -- Software Product Quality Requirements and Evaluation (SQuaRE) -- Quality Requirements. The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC). 2007.

[ISO/IEC, 2011] ISO/IEC 25010:2011. Systems and Software Engineering -- Systems and Software Quality Requirements and Evaluation (SQUARE) -- System and Software Quality Models. The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC). 2011.

[ISO/IEC/IEEE, 2010] ISO/IEC/IEEE 24765:2010. Systems and Software Engineering -- Vocabulary. The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC)/IEEE Computer Society. 2010.

[Jagannathan, Bhattacharya, Matawie, 2005] Value Based Quality Engineering. By S. R. Jagannathan, S. Bhattacharya, K. Matawie. TickIT International. 1Q05. 2005. Pages 3-9.

[Jayaswal, Patton, 2006] Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software. By B. K. Jayaswal, P. C. Patton. Prentice-Hall. 2006.

[Jones, 1998] Sizing Up Software. Scientific American. By C. Jones. Volume 279. Number 6. 1998. Pages 104-111.

[Jones, Bonsignour, 2012] The Economics of Software Quality. By C. Jones, O. Bonsignour. Addison-Wesley. 2012.

[Kamthan, 2004] A Framework for Addressing the Quality of UML Artifacts. By P. Kamthan. Studies in Communication Sciences. Volume 4. Number 2. 2004. Pages 85-114.

[Kan, 2003] Metrics and Models in Software Quality Engineering. By S. H. Kan. Second Edition. Addison-Wesley. 2003.

[Kandt, 2006] Software Engineering Quality Practices. By R. K. Kandt. Auerbach Publications. 2006.

[Khosravi, Guéhéneuc, 2004] A Quality Model for Design Patterns. By K. Khosravi, Y. -G. Guéhéneuc. Technical Report 1249. Department of Computer Science and Operations Research. University of Montreal. Montreal, Canada. 2004.

[Kitchenham, Pfleeger, 1996] Software Quality: The Elusive Target. By B. Kitchenham, S. L. Pfleeger. IEEE Software. Volume 13. Issue 1. 1996. Pages 12-21.

[Kläs, Lampasona, Nunnenmacher, Wagner, Herrmannsdörfer, Lochmann, 2010] How to Evaluate Meta-Models for Software Quality? By M. Kläs, C. Lampasona, S. Nunnenmacher, S. Wagner, M. Herrmannsdörfer, K. Lochmann. The Twentieth International Workshop on Software Measurement (IWSM 2010). Stuttgart, Germany. November 10-12, 2010.

[Lahman, 2011] Model-Based Development: Applications. By H. S. Lahman. Addison-Wesley. 2011.

[Laird, Brennan, 2006] Software Measurement and Estimation: A Practical Approach. By L. M. Laird, M. C. Brennan. John Wiley and Sons. 2006.

[Laporte, Berrhouma, Doucet, Palza-Vargas, 2012] Measuring the Cost of Software Quality of a Large Software Project at Bombardier Transportation: A Case Study. By C. Y. Laporte, N. Berrhouma, M. Doucet, E. Palza-Vargas. Software Quality Professional. Volume 14. Number 3. 2012. Pages 14-31.

[Lochmann, Goeb, 2011] A Unifying Model for Software Quality. By K. Lochmann, A. Goeb. The Eighth International Workshop on Software Quality (WoSQ 2011). Szeged, Hungary. September 4, 2011.

[Lochmann, Wagner, Goeb, Kirchler, 2010] Categorization of Software Quality Patterns. By K. Lochmann, S. Wagner, A. Goeb, D. Kirchler. Workshop zur Software-Qualitätsmodellierung und -bewertung (SQMB 2010). Paderborn, Germany. February 22, 2010.

[Lundberg, Mattsson, Wohlin, 2005] Software Quality Attributes and Trade-Offs. By L. Lundberg, M. Mattsson, C. Wohlin (Editors). Blekinge Institute of Technology. Blekinge, Sweden. June 2005.

[Maedche, Botzenhardt, Neer, 2012] Software for People: Fundamentals, Trends and Best Practices. By A. Maedche, A. Botzenhardt, L. Neer (Editors). Springer-Verlag. 2012.

[McCall, Richards, Walters, 1977] Factors in Software Quality. By J. A. McCall, P. K. Richards, G. F. Walters. National Technology Information Service. Volumes 1, 2, and 3. 1977.

[Mistrik, Bahsoon, Eeles, Roshandel, Stal, 2014] Relating System Quality and Software Architecture. By I. Mistrik, R. Bahsoon, P. Eeles, R. Roshandel, M. Stal (Editors). Morgan Kaufmann. 2014.

[Moody, 2005] Theoretical and Practical Issues in Evaluating the Quality of Conceptual Models: Current State and Future Directions. By D. L. Moody. Data and Knowledge Engineering. Volume 55. Issue 3. 2005. Pages 243-276.

[Mordal-Manet, Balmas, Denier, Ducasse, Wertz, Laval, Bellingard, Vaillergues, 2009] The SQUALE Model - A Practice-Based Industrial Quality Model. By K. Mordal-Manet, F. Balmas, S. Denier, S. Ducasse, H. Wertz, J. Laval, F. Bellingard, P. Vaillergues. The Twenty Fifth IEEE International Conference on Software Maintenance (ICSM 2009). Edmonton, Canada. September 20-26, 2009.

[Naik, Tripathy, 2008] Software Testing and Quality Assurance: Theory and Practice. By K. Naik, P. Tripathy. John Wiley and Sons. 2008.

[Reddy, Ramasree, 2012] Software Quality Modeling and Current State of the Art. By N. R. Reddy, R. J. Ramasree. International Journal of Soft Computing and Engineering. Volume 2. Issue 1. 2012. Pages 502-511.

[Regan, 2014] Introduction to Software Quality. By G. O'Regan. Springer-Verlag. 2014.

[Rico, 2004] ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers. By D. F. Rico. J. Ross Publishing. 2004.

[Rozanski, Woods, 2005] Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. By N. Rozanski, E. Woods. Addison-Wesley. 2005.

[Ruhe, Wohlin, 2014] Software Project Management in a Changing World. By G. Ruhe, C. Wohlin (Editors). Springer-Verlag. 2014.

[Samadhiya, Wang, Chen, 2010] Quality Models: Role and Value in Software Engineering. By D. Samadhiya, S.-H. Wang, D. Chen. The Second International Conference on Software Technology and Engineering (ICSTE 2010). Puerto Rico, U.S.A. October 3-5, 2010.

[Sangwan, 2015] Software and Systems Architecture in Action. By R. S. Sangwan. CRC Press. 2015.

[Schneidewind, Fenton, 1996] Do Standards Improve Quality? By N. F. Schneidewind, N. Fenton. IEEE Software. Volume 13. Issue 1. 1996. Pages 22-24.

[Shekhovtsov, 2011] On the Evolution of Quality Conceptualization Techniques. By V. A. Shekhovtsov. In: The Evolution of Conceptual Modeling: From a Historical Perspective towards the Future of Conceptual Modeling. R. Kaschek, L. Delcambre (Editors). Springer-Verlag. 2011. Pages 117-136.

[Sikka, 2005] Maximizing ROI on Software Development. By V. Sikka. Auerbach Publications. 2005.

[Streit, Pizka, 2011] Why Software Quality Improvement Fails (and How to Succeed Nevertheless). By J. Streit, M. Pizka. The Thirty Third International Conference on Software Engineering (ICSE 2011). Honolulu, U.S.A. May 21-28, 2011.

[Suryn, 2014] Software Quality Engineering: A Practitioner's Approach. By W. Suryn. John Wiley and Sons. 2014.

[Wagner, 2013] Software Product Quality Control. By S. Wagner. Springer-Verlag. 2013.

[Wagner, Lochmann, Winter, Goeb, Klaes, Nunnenmacher, 2012] Software Quality Models in Practice: Survey Results. By S. Wagner, K. Lochmann, S. Winter, A. Goeb, M. Klaes, S. Nunnenmacher. Technical Report TUM-I129. Technical University of Munich. Paderborn, Germany. 2012.

[Wieczorek, Vos, Bons, 2014] Systems and Software Quality: The Next Step for Industrialisation. By M. Wieczorek, D. Vos, H. Bons. Springer-Verlag. 2014.

[Wiegers, 2003] Software Requirements. By K. E. Wiegers. Second Edition. Microsoft Press. 2003.

[Wiegers, Beatty, 2013] Software Requirements. By K. Wiegers, J. Beatty. Third Edition. Microsoft Press. 2013.

[Wong, 2006] Different Views of Software Quality. By B. Wong. In: Measuring Information Systems Delivery Quality. By E. Duggan, J. Reichgelt (Editors). Idea Group. 2006. Pages 55-88.

[Yamada, 2014] Software Reliability Modeling: Fundamentals and Applications. By S. Yamada. Springer. 2014.