



HAL
open science

Study of Qualimetry essentials applied to embedded software development

Yann Argotti

► **To cite this version:**

Yann Argotti. Study of Qualimetry essentials applied to embedded software development. Embedded Systems. INSA de Toulouse, 2021. English. NNT : 2021ISAT0034 . tel-03738148

HAL Id: tel-03738148

<https://theses.hal.science/tel-03738148>

Submitted on 25 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THÈSE

En vue de l'obtention du DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

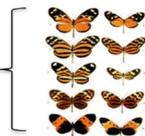
Délivré par l'Institut National des Sciences Appliquées
de Toulouse

Présentée et soutenue par

Yann ARGOTTI

Le 12 mai 2021

**Etude des caractères essentiels de la qualimétrie appliqués au
développement du logiciel embarqué**



Ecole doctorale : **SYSTEMES**

Spécialité : **Informatique et Systèmes Embarqués**

Unité de recherche :

LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes

Thèse dirigée par

Claude BARON et Philippe ESTEBAN

Président du jury

Rob VINGERHOEDS

Jury

M. Alain ABRAN, *Professor, ETS - Université du Québec, Canada*, Rapporteur

M. Alexander VERBRAECK, *Full Professor, Technical University Delft, Pays Bas*, Rapporteur

Mme Donna H. RHODES, *Principal Research Scientist, MIT, USA*, Rapporteuse

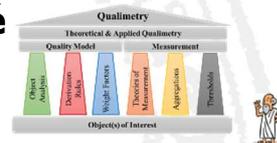
M. Rob VINGERHOEDS, *Professeur, ISAE-SUPAERO*, Examineur

Mme Karama KANOUN, *Directrice de Recherche, LAAS-CNRS*, Examinatrice

M. Silverio MARTINEZ-FERNANDEZ, *Assistant Professor, UPC, Espagne*, Examineur

Mme Claude BARON, *Professeur des Universités, INSA de Toulouse*, Directrice de thèse

M. Philippe ESTEBAN, *Maitre de Conférences, Université Paul Sabatier*, Co-directeur de thèse



“Dubium sapientiae initium” (Doubt is the origin of wisdom)

— René Descartes, Source: Meditations on First Philosophy

“One thing I have learned in a long life: that all our science, measured against reality, is primitive and childlike — and yet it is the most precious thing we have”

— Albert Einstein, Source: American Institute of Physics

*Je dédie cette thèse à ma femme, Estelle, à mes filles, Alwena, Loanne et Erell,
À mes parents, Joëlle et Gérard, à mon frère Xavier, à toute ma famille
Ainsi qu'àux gens que j'aime.*

Qu'à travers ce travail, ils y voient un reflet de ma pensée et de moi-même !

Acknowledgments

During the years of my thesis, I was able to meet and have the support of a number of people, from academia and industry. It would obviously be too long to list all these people, and even if I do not quote them here, I do not forget them.

First of all, I would like to thank Mrs Claude BARON, University Professor at INSA of Toulouse, responsible for the ISI team at LAAS-CRNS, and Mr Philippe ESTEBAN, Lecturer at the University Paul Sabatier of Toulouse. Thanks to their welcome, supervision, support and encouragement, these three years of thesis could be carried out under the best conditions in spite of many pitfalls encountered. I also thank them very much for their kindness and their precious advices which allowed me to grow both professionally and personally.

I would also like to thank Mr. Denis CHATON, Head of the DEA-LPC department at Renault SW Labs, and Mr. Thierry CAMMAL, General Manager of Renault SW Labs, who believed in this project and in me, supporting me in the setting up and the realization of this thesis.

My thanks also go to Mr. Rob VINGERHOEDS, Professor at ISAE-SUPAERO, and Mrs. Karama KANOUN, research director at LAAS-CRNS, who kindly gave me the honor of composing my follow-up committee. I thank them for their time, their listening, their judicious advices, and their kindness.

I would like to thank Mrs. Donna H. RHODES, Principal Research Scientist at MIT (*USA*), Mr. Alain ABRAN, Professor at ETS - Université du Québec (*Canada*), and Mr. Alexander VERBRAECK, Full Professor at Technical University Delft (*The Netherlands*) who have done me the honor and pleasure of being the reviewers of my thesis, as well as Mr. Silverio MARTINEZ-FERNANDEZ, Assistant Professor at UPC (*Spain*), for having honored me by being part of my thesis jury. Their academic and industrial experiences, their external and independent visions on my work, their feedbacks and advices for improvements comfort me in my research approach and the resulting proposals. They also allow me to appreciate even more the field of possibilities which results from it.

Finally, I would like to thank Mr. Alexander V. KOSTIN, Senior Researcher at Central Economics and Mathematics Institute of Russian Academy of Sciences (*Russia*), creator of the on-line library QUALIMETRY.RU, for his interest in my work and the exchanges we had around Qualimetry.

Preamble

Currently employee of the young Renault subsidiary, Renault Software Lab (formerly INTEL), located in Toulouse and focused on onboard software development, I seized a rare opportunity that is offered to me by this situation to achieve, with the downstream from my hierarchy, an important development in my career, namely to move from the managerial sector to the expertise one. Within this new entity, I must indeed evolve to become the specialist on Qualimetry. The objective is a scientific and technical challenge, and also constitutes a strategic industrial issue; it is to be able to quantitatively assess the quality of software development, from requirements, through models, code, to the maturity of the software product. Certainly, despite the existing models and quality measurement solutions, the technical optimization and correction activities linked to the integration and qualification, then acceptance and maintenance phases still represent 65% of the total cost of a project; non-quality costs 5% of turnover (source AFQP 2017 [1]). Therefore, it is essential for the industry to be able to invest in research on this subject in order to improve control, productivity, and quality not only for developers but also for the entire organization. The evaluations to be carried out thus relate to the product, here the embedded software, but also to the development processes of this software whose performance is to be estimated. Completing a doctoral thesis on these subjects will allow me to consolidate and develop my expertise, to establish my legitimacy as an expert and to promote this scientific field within the company. The ISI team at LAAS specializes in product-process-project modeling and is already conducting research in the field of software qualimetry, which is why I naturally built my research project with this team.

The Qualimetry science introduced by G. G. Azgaldov in 1968 [2], is an exciting multidisciplinary scientific field that combines technical and non-technical aspects, quantification through numerous metrics, analyzes and decisions thanks to statistics and data mining, methods, processes, etc. Scientific literature confirms that the subject is vast and indicates to us the need to develop an optimized solution for the embedded software, allowing to connect the two poles: adhesion and evaluation. In addition, this field links a variety of roles and responsibilities which are in line with my background and professional experience because I have been able to take on the role of developer, developer manager, program manager, test architect and validation team manager.

My early career, as a research and development engineer in the Computer Science and Applied Mathematics Department of the Institut Français du Pétrole, undoubtedly reinforced my deep interest in research. This is the reason why I completed my training with a DEA in the GRAVIR laboratory (Joint Research Unit between CRNS, the National Polytechnic Institute of Grenoble, INRIA and the Joseph Fourier University). Then I started a doctoral thesis in one of the laboratories of the "Center for Research and Education in Optics and Lasers" of the University of Central Florida -UCF- in the United States, accumulating a grant from the "National Institute of Health" and one from Lockheed-Martin, before making the choice to join an American start-up. Moreover, during this period, I co-authored several publications published in international conferences and journals, including four as first author. The rest of my career through various companies, including Intel, allowed me to develop numerous computer science skills. These skills were whether technological, around real-time systems, computer graphics, image processing, electronic payment systems, operating systems, or not technological, on the processes and methods of development and quality. This skillset, my scientific rigor and my taste for research give me all the keys to achieve this doctoral thesis in the best conditions.

Consequently, my objective during this thesis is to be able to scientifically contribute to advancing the state of the art through an optimized quality model, which connects and quantifies not only the embedded software development while combining the adhesion sides and evaluation, optimization of necessary and sufficient metrics - given the multitude of ways of measuring that exist.

Finally, I want to highlight that all this research work is performed in parallel to the current day to day work required by the company, even if the goal is to maximize the overlap between this additional workload and the research work.

Abstract

Today, when a company designs, develops and manufactures goods or services, it must not only target a high level of quality for the products to satisfy customers, but also comply with many standards and regulations. This is particularly true with transportation systems where we can name few reference standards and guidelines: the ISO 26262 [3] addresses the functional safety of road vehicles and covers the system, hardware and software, the ARP4754 [4] provides guidelines for the development of civil aircrafts, and the DO-178C addresses software safety [5] in aeronautics. Furthermore, these safety guidelines impose to the company to be at the state of the art for processes and methods, when designing and developing a new vehicle. Therefore, to ensure a high level of quality while complying with these standards and regulations, it is necessary that all quality requirements resulting from them are translated into a quality model. This model is the keystone for defining, evaluating, controlling, and even predicting the quality of the system.

Our research specifically addresses automotive systems (or software, depending on the choices made above), and in the context of its development, we therefore focus on the quality of embedded software in automotive vehicles.

Following an exploratory study of the literature in the field of quality models for embedded software, it is clear that there is an abundance of these models, but that there is no unified and operational solution that currently meets our needs. Our problematic of applying qualimetry essentials to the development of embedded software is therefore shifting towards reinforcement and unification of the activities to define, evaluate, control, and predict the quality of automotive embedded software.

To deal with it, we first explore the concepts of quality and qualimetry - the science of quality quantification - and establish a state of the art of quality modeling for software, including embedded software. The result of this study allows to synthesize the knowledge behind these complex concepts. It also makes it possible to conclude on the choice of qualimetry as the most relevant approach for this reinforcement and unification of the activities related to the quality modeling of embedded software. This conclusion is motivated not only by both theoretical and applied treatment of quality quantification, but also by the existence of a scientific community (e.g. civil engineers, economists, car manufacturers, architects) actively contributing to qualimetry.

Then, having noticed the existence of a similarity between the forms of evolutions, or adaptations, in quality models and in biology, during our exploration of quality modeling, we continue our study by rightly considering biology as a source of inspiration for solution guidance in our research.

We thus create a classified collection of more than 450 quality models for the software, which then shifts our problematic towards the choice of a quality model for embedded software in cars among this plethora of models. To make this model selection, which we will call the reference model, we analyze and take into account the constraints coming from standards, regulations, the automotive industry and stakeholders. Then, and after having introduced the concept of polymorphism in quality modeling as a built-in evolution and adaptation mechanism of quality models, the answer we bring to the problematic is to define and demonstrate concretely a methodology for adapting and operationalizing this reference model for embedded software in automotive vehicles.

Finally, as a consolidation of this answer, we determine a unified form of the reference quality model not, as suggested by Zouheyr Tamrabet et al [6], as a single quality model for software products, but as a meta-model serving as an aggregator of quality models for software products. The end result is the proposal of the genome of the software quality model that we model in the shape of 7 chromatids.

Résumé

Aujourd'hui, lorsqu'une entreprise conçoit, développe et fabrique des biens ou des services, elle doit non seulement viser un niveau de qualité élevé pour que ses produits satisfassent les clients, mais également se conformer à de nombreuses normes et réglementations. C'est particulièrement vrai pour les systèmes de transport pour lesquels nous pouvons citer quelques normes et directives de référence : la norme ISO 26262 [3] traite de la sécurité fonctionnelle des véhicules routiers et couvre le système, le matériel et le logiciel, l'ARP4754 [4] fournit des directives pour le développement des avions civils, et la DO-178C traite de la sécurité des logiciels [5] dans l'aéronautique. En outre, ces directives de sécurité imposent à l'entreprise d'être à la pointe de la technologie en matière de processus et de méthodes, lors de la conception et du développement d'un nouveau véhicule. Aussi, pour s'assurer d'un niveau de qualité élevé tout en étant en conformité avec ces normes et réglementations, il est nécessaire que toutes les exigences de qualité qui en découlent soient traduites au sein d'un modèle qualité. Ce modèle est la clef de voûte de la définition, de l'évaluation, du contrôle, et même de la prédiction de la qualité du système.

Nos recherches adressent spécifiquement les systèmes (ou logiciels, selon les choix faits plus haut) automobiles, et dans le contexte de leurs développements, nous nous concentrons donc sur la qualité des logiciels embarqués dans les véhicules automobiles.

À la suite d'une étude exploratoire de la littérature des modèles qualité pour le logiciel embarqué, force est de constater l'abondance de ces modèles, sans qu'il n'existe aucune solution unifiée et opérationnelle répondant actuellement à notre besoin. Notre problématique d'appliquer les caractères essentiels de la qualimétrie au développement du logiciel embarqué se déporte ainsi vers le renforcement et l'unification des activités de définition, d'évaluation, de contrôle et de prédiction de la qualité des logiciels embarqués dans les véhicules automobiles.

Pour la traiter, nous explorons en premier lieu les concepts de qualité et de qualimétrie - la science de la quantification de la qualité -, et établissons un état de l'art de la modélisation de la qualité pour le logiciel, incluant le logiciel embarqué. Le résultat de cette étude permet de synthétiser les connaissances qui se cachent derrière ces concepts complexes. Il permet aussi de conclure sur le choix de la qualimétrie comme l'approche la plus pertinente pour ce renforcement et unification des activités liées à la modélisation de la qualité des logiciels embarqués. Cette conclusion est motivée non seulement par un traitement à la fois théorique et appliquée de la quantification de la qualité, mais aussi par l'existence d'une communauté scientifique (par exemple, ingénieurs civiles, économistes, constructeurs automobiles, architectes) contribuant activement à la qualimétrie.

Puis, après avoir remarqué l'existence d'une similitude entre les formes d'évolutions, ou d'adaptations, dans les modèles qualité et dans la biologie, durant notre exploration de la modélisation de la qualité, nous poursuivons notre étude en considérant à juste titre la biologie comme source d'inspiration pour l'orientation de solutions dans notre recherche.

Nous créons ainsi une collection classifiée de plus de 450 modèles qualité pour le logiciel, ce qui déplace alors notre problématique vers le choix d'un modèle qualité pour le logiciel embarqué dans les véhicules automobiles parmi cette pléthore de modèles. Pour réaliser cette sélection de modèle, que nous dénommerons modèle de référence, nous analysons et prenons en compte les contraintes provenant des normes, des réglementations, de l'automobile et des parties prenantes. Ensuite, et après avoir introduit le concept de polymorphisme dans la modélisation de la qualité comme mécanisme d'évolution et d'adaptation intrinsèque des modèles qualité, la réponse que nous apportons à la problématique se résume à définir et démontrer concrètement une méthodologie d'adaptation et d'opérationnalisation de ce modèle de référence pour les logiciels embarqués dans les véhicules automobiles.

Enfin, en guise de consolidation de cette réponse, nous déterminons une forme unifiée de modèle qualité de référence non pas, comme le suggèrent Zouheyr Tamrabet et al. [6], en tant que modèle qualité unique pour les produits logiciels, mais en tant que méta-modèle servant d'agrégateur de modèles qualité pour les produits logiciels. Le résultat final est la proposition du génome du modèle qualité du logiciel que nous modélisons sur la forme de 7 chromatides.

Table of Contents

Acknowledgment	5
Preamble	7
Abstract	9
Résumé.....	11
Table of Contents	13
List of Figures.....	17
List of Tables.....	21
List of Acronyms	23
Chapter I. Introduction.....	25
Chapter II. Context and Problems.....	29
1. Introduction	29
2. A complex industrial research context with a need of quality.....	29
3. Quality modeling applied to embedded software development	31
a. Quality model creation or use studies.....	32
b. Quality characteristics or/and quality model identification studies.....	35
c. Reliability growth model creation or use studies	38
d. Miscellaneous work related to quality modeling in embedded software	39
e. Limits to existing quality modeling applied to embedded software development	40
4. Research questions	42
5. Threats to validity and discussions.....	47
Chapter III. Research Methodology	49
1. Introduction	49
2. Initial research methodology: qualimetry, classification and decision	49
3. Realigning our methodology: from theory to practice	51
4. Last step: construction of a meta-model	53
5. Threats to validity and discussions.....	54
Chapter IV. Quality, Quality Modeling and Qualimetry	55
1. Introduction	55
2. The essence of quality.....	55
a. Integral quality.....	56
b. Perceived quality	56
c. Quality perspectives and views	57
d. Quality dimensions and characteristics	57
e. Lagging and leading indicators.....	59
3. The essence of quality modeling.....	60
a. Quality model	60
b. Measurements.....	63

c.	Scale types	64
d.	Aggregation.....	66
e.	Threshold	68
4.	Key contributions to quality modeling of software.....	70
5.	Qualimetry: the science of quality quantification.....	74
6.	Contributions to Qualimetry	77
a.	The “House of Qualimetry”	77
b.	Polymorphism applied to quality model.....	80
c.	Quality model distance formula	83
d.	Measurement process	87
7-	Threats to validity and discussions.....	90
Chapter V.	Quality Model Classification and Selection	93
1.	Introduction	93
2.	Systematic literature review on software quality model.....	93
3.	Software quality model classification	100
4.	Contributions.....	105
a.	Cladistic as Classification Method of Software Quality Models.....	105
b.	The first list of 492 software quality models	107
c.	Software quality model landscape and the selection question.....	110
5.	Threats to validity and discussions.....	114
Chapter VI.	Quality Model Operationalization.....	117
1.	Introduction	117
2.	Operational challenges and issues with quality model.....	117
3.	Practical solutions to the operational issues.....	121
4.	Operational Contributions	126
a.	Quality model operational use: The “Quality Thermometer”	126
b.	Quality model operational development: a 6 stages process	129
5.	“Quality thermometer” and “6-stages” process comparison against current ISO/IEC standards	134
6.	Threats to validity and discussions.....	136
Chapter VII.	Put into Practice	137
1.	Introduction	137
2.	Quality modeling on a real-world use case: embedded software for the automotive industry.....	137
3.	Initiating quality model construction via the 6-stages process.....	139
4.	Survey result analysis.....	143
5.	Contributions.....	146
a.	Importance values and weight factors	146
b.	Three quality models for four sub-systems: IVI, IVC, ADAS & FOTA.....	149
c.	The polymorphic quality models	151

Table of Contents

6.	Threats to validity and discussions.....	153
Chapter VIII.	Meta-Model: Software Quality Model Genome.....	155
1.	Introduction	155
2.	Motivation and analogy with genetic.....	155
3.	Software quality model genome meta-model construction algorithm	158
a.	Construction methodology	158
b.	Construction algorithm.....	161
4.	Software quality model genome meta-model construction	165
a.	List of quality models	165
b.	List of genes with their variations.....	166
c.	Relationship links between genes and chromatid identification.....	171
5.	Contributions: The 7 Chromatids of SW Quality Model Genome	172
a.	“General Utility” Chromatid (A08).....	175
b.	“Product Operation” Chromatid (A16)	177
c.	“Product Revision” Chromatic (A17).....	179
d.	“Product Transition” Chromatid (A18)	181
e.	“Supportability” Chromatid (A24).....	182
f.	“Product in Use” Chromatid (B08).....	184
g.	“Software Product” Chromatid (B12)	185
6.	Threats to validity and discussions.....	188
Chapter IX.	General Synthesis and Research Perspectives	191
1.	General synthesis	191
2.	Research Perspectives.....	195
References.....		197
Annexes		237
Annex 1.	Catalog of the main existing aggregation operators (from Chapter IV.3.d).....	237
Annex 2.	Main sequence, string, or similarity distance formulas	241
Annex 3.	Example of distance calculation: diversity or polymorphism degree applied to ISO/IEC/IEEE 25010 and ISO/IEC 9126 quality models.....	243
Annex 4.	Measurement process key document: Evaluation plan template	247
Annex 5.	Systematic literature review results	253
Annex 6.	List of the 492 quality models, from 1968 to 2019	299
Annex 7.	Project scorecard description	337
Annex 8.	ISO/IEC/IEEE 25010 quality models [23]	343
Annex 9.	Survey used against our real-world use case	349
Annex 10.	Survey results.....	357
Annex 11.	Automotive real-world use case analysis results: importance values, inclusion / exclusion decisions and weight factors.....	363
Annex 12.	Basic set of measures to enable the real-world use case quality models	369

Annex 13.	Details about the eight selected software quality models for the software quality model genome meta-model construction	377
Annex 14.	Variations of quality characteristic genes.....	391
Chapter X.	Synthèse de la Thèse en Français.....	399
1.	Synthèse Générale	399
2.	Perspectives de recherche	402

List of Figures

Figure 1 - Dissertation chapters mapped against the 4 research questions	27
Figure 2 - Growing ECUs number per year for luxury cars (source Crolla <i>et al.</i> [15], chapter 14, figure 2)	30
Figure 3 - Embedded systems in contrast to other computing systems (source Lepistö [34])	32
Figure 4 – Ontology of quality model definition from IEEE 1061 [40]	33
Figure 5 - Use of type of information as a helper to find and associate metrics to quality requirements, based on Guessi <i>et al.</i> [68]	37
Figure 6 - Overview of our early research flow breakdown	51
Figure 7 - Overview of our research flow breakdown: in red the technological locks, in green our contributions	52
Figure 8 - Kano's model of perceived quality [58]	56
Figure 9 - Ontology of main quality keywords	59
Figure 10 - Leading indicator composition, Roedler <i>et al.</i> [106]	59
Figure 11 - The DAP classification introduced by Deissenboeck <i>et al.</i> 2009 [120]	61
Figure 12 – Example of a definition model: the ISO/IEC/IEEE 25010 System / Software product quality model [23]	62
Figure 13 – Example of an assessment model: <i>Maturity</i> sub-characteristics measurement extension of ISO/IEC/IEEE 25010 System / Software product quality model [23]	62
Figure 14 - Boehm's quality model (1976) [42] with an example of cycle in the graph structure	63
Figure 15 - Illustration of the concept of reliability and validity of measures [135]	64
Figure 16 – The Chrisman's scales [139], an extension of the Stevens' scales (in boldface)	66
Figure 17 – “ <i>Continuous Preference Logic</i> ” operators and “corresponding weighted power mean parameter used in the “ <i>Logic Scoring of Preference</i> ” of Dujmovic and Bayucan [141]	68
Figure 18 - Threshold for risk acceptance and opportunities: from ISO/IEC 25022 [140] theory to practice	69
Figure 19 - Timeline of the key contributions to quality modeling of software	71
Figure 20 - The “ <i>House of Qualimetry</i> ” and its 6 pillars	78
Figure 21 - The “ <i>House of Qualimetry</i> ” and its 6 pillars in a nutshell	79
Figure 22 - Example of a result from the application of the two of the three pillars of quality model architrave against “software product”: the ISO/IEC/IEEE 25010 quality models [23]	80
Figure 23 - Polymorphism mechanism showcased with butterfly analogy: example of a generic butterfly and its variety of butterflies with heredity links (source: [167] & [165])	81
Figure 24 - Polymorphism mechanism showcased with butterfly analogy: example of temporal evolution (source: [168])	82
Figure 25 - Example of some differences between ISO/IEC 9126 & ISO/IEC/IEEE 25010	87
Figure 26 - Software product quality evaluation process defined by ISO/IEC 25040 [144]	88
Figure 27 - Measurement process model of McGarry <i>et al.</i> [154]	88
Figure 28 - Our measurement process proposal articulated over three phases and cadenced with SDLC	89
Figure 29 - The distinct stages of the systematic literature review process for study selection, with selected document numbers and publication years	97
Figure 30 - Ratio of the two types (i.e., conference and journal) of selected studies	98
Figure 31 - Ratio of context (i.e., academic, industry, both) of the selected studies	98

Figure 32 - Selected conference and journal paper distribution over publication year.....	99
Figure 33 - Citation numbers of systematic literature review qualified papers per year; top bar graph, cumulative citation number, bottom bar graph, normalized citation number per number of papers.....	99
Figure 34 - Distributions of studies per type of study	101
Figure 35 - Basic vs. Tailored quality model categorization (source: Thapar et al. [11]).....	103
Figure 36 – Distributions of study classification or comparison criteria categories.....	103
Figure 37 - Comparison done by Snyder [207] on citation metrics difference between Google Scholar, Web of Science publisher and the reality	104
Figure 38 - The software quality model classification elements organized over five themes: id, bibliographic, definition, scope and structural.....	105
Figure 39 - Software Quality Model Clade based on a homology and five taxa.....	107
Figure 40 - The 492 created and published software quality models per formalism and year wise	108
Figure 41 - The normalized citation numbers of the 492 published software quality model papers year wise ..	108
Figure 42 - The normal distribution related to the number of quality models per study paper.....	109
Figure 43 - Cumulative number of quality model citations in the 136 study papers year wise	109
Figure 44 - The 35 most cited (i.e., cited more than 5 times) software quality models in the 136 study papers, order by chronologic order.....	109
Figure 45 - DAP type distribution of the 492 software quality model samples	111
Figure 46 - Formalism type distribution of the 492 software quality model samples	111
Figure 47 - Insight on the main prediction method distribution of the 492 software quality model samples	112
Figure 48 - Main quality perspectives, distribution of the 492 software quality model samples.....	112
Figure 49 - Nuance in quality perspectives, distribution of the 492 software quality model samples	113
Figure 50 - Scope distribution of the 492 software quality model samples.....	113
Figure 51 - Comparison between creation of new quality models and derived quality models over the 492 software quality model samples.....	113
Figure 52 - Frequencies of the 9 main issues identified by Thapar et al. [11], which impede the development and use of quality models.....	119
Figure 53 - The A-SPICE capability levels of process maturity [21]	123
Figure 54 - Mapping of practical solutions against the 16 issues preventing development and use of quality models	125
Figure 55 - The software project scorecard including indicators and metrics for project, process, and product	126
Figure 56 – Inheriting of the practical solutions to the operational issues, the “Quality Thermometer” process for project	128
Figure 57 - Visual display of quality level done via color coding associated to range decomposition coming from ISO/IEC 33020 [138].....	129
Figure 58 - Example of a 5 points Likert's scale.....	130
Figure 59 - Example of a 6 points Likert's scale.....	130
Figure 60 - 6-stages process for quality model development	132
Figure 61 - Algorithm of our analysis based on Fleiss and Cohen's kappa	133
Figure 62 – Over-The-Air adoption timeline of major automobile manufacturers (sources: [247], [248])	138
Figure 63 - Example of FOTA working principle with IVC, IVI and ADAS ECUs	139

Figure 64 - The ISO/IEC/IEEE 25010 quality models: "System / Software product quality model" and "Quality in use model"	140
Figure 65 - Example of a mapping between quality models, properties, measures and an automotive systems and software (source and inspiration from ISO/IEC/IEEE 25010 [23] & ISO/IEC 25030 [251])	141
Figure 66 - Survey extract: participant role and project	141
Figure 67 - Survey extract: the ranking choice of quality characteristics.....	142
Figure 68 - Survey extract: the ranking choice of quality sub-characteristics.....	142
Figure 69 - Survey extract: the final open question of the survey	142
Figure 70 - Response distribution per order of importance ranking.....	143
Figure 71 - The resulting IVI embedded software quality model; numbers in parenthesis are characteristic /sub-characteristic weight factors	149
Figure 72 - The resulting IVC embedded software quality model; numbers in parenthesis are characteristic /sub-characteristic weight factors	150
Figure 73 - The resulting ADAS & FOTA embedded software quality model; numbers in parenthesis are characteristic /sub-characteristic weight factors	150
Figure 74 - The common quality model from which the other quality models are derived	151
Figure 75 - The secondary common quality model from which IVI and ADAS & FOTA quality models are derived	151
Figure 76 - The polymorphism tree structure with the five polymorphic quality models	152
Figure 77 - <i>Genetic-Quality</i> analogy: global overview with chromosome, chromatids, and DNA sequence (genetic terminology is in green, quality terminology is in purple).....	156
Figure 78 - <i>Genetic-Quality</i> analogy: detail overview with locus, sites, genes and alleles (genetic terminology is in green, quality terminology is in purple)	156
Figure 79 - <i>Genetic-Quality</i> analogy: an example based on ISO/IEC/IEEE 25010 quality models	157
Figure 80 - <i>Genetic-Quality</i> analogy: Meta-model ontology	157
Figure 81 - The seven steps in the software quality model genome meta-model construction algorithm	158
Figure 82 - Example of "Supportability" chromatid computation with the main gene and some of its sub-genes (testability, adaptability, maintainability, changeability and reusability); sub-genes are site names with color background.....	160
Figure 83 - Example from the online Semantic Atlas: three synonym constellations for "efficiency" word as quality characteristic (source: http://www.atlas-semanticques.eu/).....	164
Figure 84 – Links between the 43 genes; chromatids are identified by X and dotted arrows indicate a relationship link that requires to take into account parent gene context	172
Figure 85 - From the 43 linked genes to a specific the chromatid linked genes and the corresponding site sequence.....	173
Figure 86 - The 27 genes of "General utility" chromatid, with their respective links	175
Figure 87 - "General utility" chromatid quality characteristic sequence with its 27 genes and the 114 sites, including sites likelihood	176
Figure 88 - The 17 genes of "Product operation" chromatid, with their respective links	177
Figure 89 - "Product operation" chromatid quality characteristic sequence with its 17 genes and the 85 sites, including sites likelihood	178
Figure 90 - The 15 genes of "Product revision" chromatid, with their respective links	179

Figure 91 - " <i>Product revision</i> " chromatid quality characteristic sequence with its 15 genes and the 96 sites, including sites likelihood	180
Figure 92 - The 10 genes of " <i>Product transition</i> " chromatid, with their respective links	181
Figure 93 - " <i>Product transition</i> " chromatid quality characteristic sequence with its 10 genes and the 43 sites, including sites likelihood	181
Figure 94 - The 15 genes of " <i>Supportability</i> " chromatid, with their respective links.....	182
Figure 95 - " <i>Supportability</i> " chromatid quality characteristic sequence with its 15 genes and the 72 sites, including sites likelihood	183
Figure 96 - The 11 genes of " <i>Product in use</i> " chromatid, with their respective links	184
Figure 97 - " <i>Product in use</i> " chromatid quality characteristic sequence with its 11 genes and the 34 sites, including sites likelihood	184
Figure 98 - The 32 genes of " <i>Software product</i> " chromatid, with their respective links	185
Figure 99 - " <i>Software product</i> " chromatid quality characteristic sequence with its 32 genes and the 195 sites, including sites likelihood (part 1 of 2)	186
Figure 100 - " <i>Software product</i> " chromatid quality characteristic sequence with its 32 genes and the 195 sites, including sites likelihood (part 2 of 2)	187
Figure 101 - Overview of the software quality model genome meta-model composed of 7 chromatids	188
Figure 102 - General synthesis of the thesis research work and achievements	194
Figure 103 - Synthèse générale des travaux de recherche et des réalisations de la thèse.....	403

List of Tables

TABLE 1 - DETAILED DISSERTATION PLAN	28
TABLE 2 – SUMMARY OF STUDY KEY POINTS	41
TABLE 3 - OUR FIRST THREE RESEARCH QUESTIONS.....	49
TABLE 4 - SUMMARY OF THE 4 SCALES OF MEASUREMENTS DEFINED BY S. S. STEVENS [139]	65
TABLE 5 – NOMINAL SCALE-BASED RATING, ACCORDING TO ISO/IEC 33020 [136]	65
TABLE 6 - ORDINAL SCALE-BASED RATING IN PERCENTAGE VALUES, ACCORDING TO ISO/IEC 33020	66
TABLE 7 - EXAMPLE OF KHADDAJ AND HORGAN [148] RELATIONSHIP CHART USED FOR POLARITY PROFILE.....	69
TABLE 8 - GENERAL RULES FOR QUALITY MODEL TREE DERIVATION	75
TABLE 9 - SPECIFIC RULES FOR QUALITY MODEL TREE DERIVATION	75
TABLE 10 - COMPARISON OF EIGHT MAIN DISTINCT APPROACHES SUPPORTING QUALITY MODEL DEVELOPMENT AND USE.....	76
TABLE 11 - EXAMPLE OF DISTINCT MEASURABLE OBJECTIVES USING FURPS QUALITY MODEL FOR EACH LIFE CYCLE PHASE (SOURCE: GRADY AND CASWELL [85], FIGURE 11-7, PAGE 161).....	80
TABLE 12 - EXAMPLE OF LEXICAL AND SEMANTIC COMPARISON RESULT BETWEEN TWO CHARACTERISTICS FROM ISO/IEC 9126 AND ISO/IEC/IEEE 25010	86
TABLE 13 - INTERMEDIATE CALCULATION LINKED TO PREVIOUS EXAMPLE.....	87
TABLE 14 - LIST OF ELECTRONIC DATABASE SOURCES	95
TABLE 15 - SEARCH QUERY ELEMENTS: KEYWORDS VS. OPERATORS VS. SEARCH FIELDS	95
TABLE 16 – THE FIVE DIGITAL LIBRARY SEARCH QUERIES	96
TABLE 17 - INCLUSION AND EXCLUSION CRITERIA RELATED TO OUR SEARCH STRATEGY	96
TABLE 18 - MAPPING OF THE 136 PAPER STUDIES AGAINST THE SEVEN TYPES OF STUDY	100
TABLE 19 - MAPPING OF THE 136 PAPERS STUDIES AGAINST THE EIGHT CATEGORIES OF CLASSIFICATION OR COMPARISON CRITERIA	102
TABLE 20 – ALPHABETICALLY SORTED LIST OF MAIN POTENTIAL ISSUES THAT CHALLENGE DEVELOPMENT AND USE OF QUALITY MODELS	121
TABLE 21 - KAPPA INTERPRETATION (SOURCE LANDIS AND KOCH [245])	131
TABLE 22 - COMPARISON BETWEEN THE "QUALITY THERMOMETER" PROCESS AND THE CURRENT RELEVANT STANDARDS: ISO/IEC 250NN SERIES AND ISO/IEC/IEEE 15939	135
TABLE 23 - COMPARISON BETWEEN THE "6-STAGES" PROCESS AND THE CURRENT RELEVANT STANDARD: ISO/IEC 250NN SERIES AND ISO/IEC/IEEE 15939	135
TABLE 24 - EXTRACTED NUMBER OF SURVEY RESPONSES PER QUALITY IN USE QUALITY CHARACTERISTICS AND IMPORTANCE FOR IVC AND PROJECT MANAGER ROLE	144
TABLE 25 – THE ASSOCIATED RANKING RESPONSE MATCHES BETWEEN ALL POSSIBLE RESPONSE COMBINATIONS; THE GREEN CELLS INDICATE PERFECT MATCH	144
TABLE 26 – EXTRACTED NUMBER OF SURVEY RESPONSES PER SYSTEM/SOFTWARE PRODUCT QUALITY CHARACTERISTICS AND IMPORTANCE FOR IVC AND ANY ROLE	145
TABLE 27 - SURVEY DATA ANALYSIS WITH COHEN K AND FLEISS K. COLORED CELLS HIGHLIGHT K BASED CHOICE FOR EACH ECU; GRAYED CELLS HIGHLIGHT AT LEAST MODERATE AGREEMENT	145
TABLE 28 - <i>GENETIC-QUALITY</i> ANALOGY: TERMINOLOGY SUMMARY	158
TABLE 29 - ALLELE AND SITE EXAMPLE FOR " <i>PORTABILITY</i> " QUALITY CHARACTERISTIC.....	159
TABLE 30 - GENE COMPUTATION EXAMPLE FOR " <i>PORTABILITY</i> " QUALITY CHARACTERISTIC	159

TABLE 31 - OVERVIEW OF THE EIGHT SELECTED QUALITY MODELS FOR SOFTWARE QUALITY MODEL GENOME META-MODEL CONSTRUCTION 165

TABLE 32 – THE 55 DISTINCT QUALITY CHARACTERISTIC GROUPS FROM THE EIGHT SELECTED SOFTWARE QUALITY MODELS..... 166

TABLE 33 - THE 10 MERGED QUALITY CHARACTERISTIC GROUPS (GREEN BACKGROUND CELL INDICATES A MERGED ENTRY) 167

TABLE 34 - LIST OF THE 27 SINGLE QUALITY MODEL GENES 168

TABLE 35 - LIST OF THE 16 MULTI-QUALITY MODELS GENES 169

TABLE 36 - THE RELATIONSHIP LINKS BETWEEN THE QUALITY CHARACTERISTIC GENES..... 171

TABLE 37 - EXAMPLE OF GENE 05 "MAINTAINABILITY" DETAILED WITH ALL ITS SUB-GENES AND SITES: ON THE LEFT SIDE, THE DIRECT DETAILED SITE SEQUENCE, ON THE RIGHT THE DETAILED AND OPTIMIZED SITE SEQUENCE WITH SAME QUALITY COVERAGE 174

TABLE 38 - DEGREE OF SUBJECTIVITY FOR EACH CHROMATID OF THE SOFTWARE QUALITY MODEL GENOME META-MODEL 188

TABLE 39 - CATALOG OF THE MAIN EXISTING AGGREGATION OPERATORS, BASED ON DETYNIECKI [143], WAGNER [27] AND DUJMOVIC & BAYUCAN [144] 237

TABLE 40 - EXAMPLES OF HAMMING'S DISTANCE COMPUTATION: RED CHARACTERS INDICATE DIFFERING CHARACTERS 241

TABLE 41 - EXAMPLES OF LEVENSHEIN'S DISTANCE COMPUTATION: RED CHARACTERS INDICATE DIFFERING CHARACTERS 241

TABLE 42 - EXAMPLES OF DAMERAU-LEVENSHEIN'S DISTANCE COMPUTATION: RED CHARACTERS INDICATE DIFFERING CHARACTERS 241

TABLE 43 - EXAMPLES OF JARO'S DISTANCE COMPUTATION: RED CHARACTERS INDICATE DIFFERING CHARACTERS 242

TABLE 44 - EXAMPLES OF JARO-WINKLER'S DISTANCE COMPUTATION WITH $L = 2$: RED CHARACTERS INDICATE DIFFERING CHARACTERS 242

TABLE 45 - LIST OF FILTERED PUBLISHED STUDIES RESULTING THE SYSTEMATIC LITERATURE REVIEW 253

TABLE 46 - MAIN RAW RESULTS FROM THE SYSTEMATIC LITERATURE REVIEW 262

TABLE 47 - MAPPING SYSTEMATIC LITERATURE REVIEW STUDIES WITH QUALITY MODELS 272

TABLE 48 - THE 492 DISTINCT QUALITY MODELS, FROM 1968 TO 2019 299

TABLE 49 - DEFINITIONS OF THE DIFFERENT ELEMENTS OF THE PROJECT SCORECARD 338

TABLE 50 - LIST OF THE 27 SINGLE-QUALITY MODEL GENES WITH THEIR POSSIBLE VARIATIONS 391

TABLE 51 - LIST OF THE 16 MULTI-QUALITY MODELS GENES WITH THEIR POSSIBLE VARIATIONS 393

List of Acronyms

ADAS	Advanced Driver-Assistance Systems
AHP	Analytic Hierarchy Process
ANP	Analytic Network Process
ATAM	Architecture Tradeoff Analysis Model
BCM	Body Control Module
CNRS	Centre National de la Recherche Scientifique, The French Center for Scientific Research
COTS	Component On The Shelves
DAP	Definition, Assessment, Prediction
DEA	Direction Engineering Alliance
ECU	Electronic Control Unit
FMEA	Failure Model and Effects Analysis
FOTA	Firmware Over The Air
GQM	Goal - Question - Metrics
HEVC	Hybrid and Electrical Vehicle Controller
HIL	Hardware In the Loop
ISI	Ingénierie Système et Intégration (i.e., Systems Engineering and Integration)
IVC	In-Vehicle Communication
IVI	In-Vehicle Infotainment
LAAS	Laboratory of Analysis and Architecture of Systems
MTBF	Mean Time Between Failure
NLP	Natural Language Processing
ODC	Orthogonal Defect Classification
QM	Quality Model
SFMEA	Software Failure Model and Effects Analysis
SFTA	Software Failure Tree Analysis
SIL	Software In the Loop
SOQUAL	Software Quality project
SQM	Software Quality Model
SQuaRE	System and software Quality Requirements and Evaluation (i.e., ISO/IEC 250nn serie)
SW	Software
SWFMEA	Software Failure Mode Effects Analysis
TRL	Technology Readiness Level

Chapter I. Introduction

This thesis took place at LAAS – CNRS laboratory, in the ISI team *via* an industrial partnership with Renault Software Labs, a subsidiary of Renault SAS company. This research project, founded by Renault Software Labs, had its prequel in Renault several years before the creation of this subsidiary in July 2017. However, since the industrial research problematic (i.e., define and evaluate automotive embedded software quality) remained still valid and open, and for the sake of consolidating the new company's expertise, it was decided to foster the training through research approach against that problematic. In the following paragraphs we are going to review successively the global context, the problematic, the thesis objective with the expected results and the dissertation plan.

Research, focused on systems and software, is performed in an international research laboratory jointly with strong relationships with the industry.

LAAS, Laboratory of Analysis and Architecture of Systems, is an own research unit of CNRS. Over its 52 years of history¹, it has settled close and strong relationships with regional, national, and international industries, sometimes resulting in the creation of common laboratories (e.g., the common laboratory between LAAS and Lacroix company created on end of 2016 [7]). The scientific research in LAAS is organized around 6 departments, composed themselves from 3 to 7 teams. And even if each of them has its own field of expertise, these teams are all working on a wide range of systems: “*integrated systems, embedded systems with real time and safety requirements, distributed systems, mobile systems, autonomous and robotics systems, micro and nano systems, biological systems*” [8].

Thus, as a member of the ISI research team (“*Ingénierie Système et Intégration*” in English “*Systems Engineering and Integration*”), part of the RISC group (“*Réseaux, Informatique, Systèmes de Confiance*” in English “*Trustworthy Computing Systems and Networks*”), my research is currently focused on systems engineering applied to embedded system and software, which are both aligned with the company domains.

An industrial context of a distributed company whose business is the development of embedded software in the automotive sector.

Today, when a company designs, develops and manufactures goods or services, it must not only target a high level of quality for the products to satisfy customers, but also comply with many standards and regulations. This is particularly true with transportation systems where we can name few famous standards and guidelines: the ISO 26262 [3] addresses the software functional safety in automotive, the ARP4754 [4] provides guidelines for the development of civil aircrafts, and the DO-178C addresses software safety [5] in aeronautics. Furthermore, these safety guidelines impose to the company to be at the state of the art for processes and methods, when designing and developing a new vehicle.

So, in order to take into account these obligations from standards while targeting at the same time a high level of quality for the products to satisfy customers, the international car manufacturer Renault SAS, initiated an internal project called *SOQUAL* (i.e., Software QUALity) in 2015. As the project name indicates, the point of attention is set on the software because the quality and customer satisfaction company department identified the software as one of the most problematic elements in a car since software is especially developed by suppliers and only a minority of the company employees is familiar with software concepts. It should be noted also that software becomes more and more important, functionally, in terms of volume and criticality for vehicles [9], which therefore reinforces the general need to pay attention to this element. Nevertheless, at the time Renault Software Labs joined Renault SAS, only 3 main indicator definitions (i.e., coverage, completeness, and consistency indicators), a delivery checklist and a document listing some software measurements were the results of this project.

An arrival that changes the rules in the company embedded software development model and quality, but a remaining problematic.

With the arrival of Renault Software Labs on July 2017, bringing on board software experts over all activities, from specification to validation, and from methods to tools, Renault SAS makes clear that software development is considered as one of the company strategic activities since then. In this context of software activity

¹ On May 2020

reinforcement, it becomes also crucial to strengthen and unify the definition, assessment, control, or prediction of the embedded software quality. To solve this problematic, we have first to **understand and accurately define what qualimetry**, the young science of quality quantification, **as well as quality**, are.

We can rephrase this statement under our first research question:

Research Question 1

Is Qualimetry, as the science of quality quantification, the right approach and what are quality and Qualimetry essentials?

The answers to this question not only aim to remove any ambiguity, popularize, and synthetize the knowledge behind these complex concepts but also, they should confirm or deny the choice of Qualimetry as the right approach to solve our problematic.

Then, strong with that knowledge and assuming that Qualimetry is the right approach, the next step consists in **modeling the quality** of embedded software within the context of automotive embedded systems.

Through some early investigation, we notice the existence of many software quality models in the literature (e.g., Kläs *et al.* identified 22 quality models [10], Thapar *et al.* conducted a comparison analysis over a reference set of 24 quality models [11], and Oriol *et al.* collected and used a set of 51 distinct quality models [12]). So rather than creating a quality model from scratch, we may select the most appropriate ones from this pool of existing quality models since embedded software is sub-set of software.

This leads us to our second research question:

Research Question 2

Considering the set of software quality models, how to identify and decide which quality model is the most suitable for embedded software?

Therefore, we expect to structure the knowledge of quality models for software *via* a taxonomy, for its classification, and a cartography, to depict accurately and precisely its landscape interpretation. Both classification and interpretation should serve as the decision basis to identify which quality model is the most suitable one to address our needs within our embedded software scope.

Nevertheless, selecting a quality model is not enough to use it either for quality definition, assessment, control, or prediction. This requires some further actions that be summarized and generalized under the action of **operationalizing a software quality model**.

This is our third research question:

Research Question 3

Considering a quality model for a software product, how to operationalize it?

This pivotal question addresses the transition from the theory space to the practice space. Indeed, identifying a quality model was the first major step to resolve our global problematic, however, to claim that we answered the problematic, we must be able to **use that quality model** for assessment, control, or prediction activities. Thus, with that third research question, we expect to build a process to guide the operationalization of quality models, avoiding all technological locks, and demonstrate it against a real-world use case.

At this stage we have answered to our problematic, considering the specific case of embedded software within the context of automotive embedded systems.

However, it would be relevant to conclude our research investigation about the generalization of our approach to a broader software scope. We note that the specificity of our answer for a specific scope occurs in the research question 2 with the search for the most suitable quality model. Thus, that generalization result is directly linked by confirming or refuting that a unique quality model for software product is appropriate.

This is our final research questions:

Research Question 4

Can we have a unique reference quality model for software product?

The expected result is either the unique reference quality model, if it is viable, or a meta-model as quality model aggregator for software product.

All these four research questions can be summarized under the thesis subject “*Study of Qualimetry essential applied to embedded software development*” and they cadence the chapter sequence of this dissertation (cf. Figure 1).

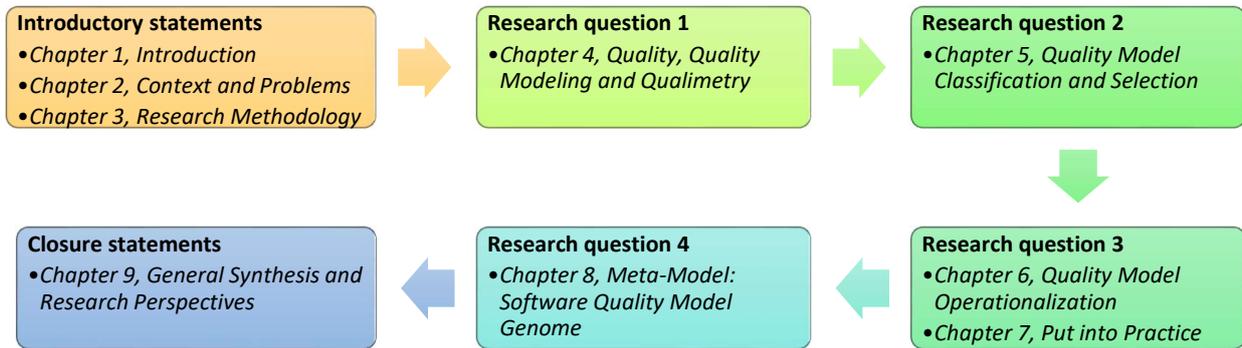


Figure 1 - Dissertation chapters mapped against the 4 research questions

Consequently, the detailed plan of this document, shown in TABLE 1, follows our logical reasoning, starting with introductory statements, detailing context, problems and our research approach, then addressing sequentially each research question, before ending with our closure statements, summarizing the research achievements, contributions and the opening research topics.

TABLE 1 - DETAILED DISSERTATION PLAN

Introductory statements

•Chapter 1, Introduction

•This is the general dissertation introduction, introducing the main research problematic, questions, and dissertation plan.

•Chapter 2, Context and Problems

•In this chapter deals about the current industrial context, the vehicle and embedded software, and the quality modeling for embedded software problems we aim to solve, aligned with company needs.

•Chapter 3, Research Methodology

•During our research we didn't follow a straight line, and we had to reconsider our research approach at a certain time. So here we explain the different steps of our research flow, the analysis technological locks and decision we made (e.g., we decide to reuse, adapt quality model rather than create a new one: in fact when we look for quality model, we can notice that most of cases are reusing, changing or being closed to what have been done rather than a complete disruption).

Research question 1

Is Qualimetry, as the science of quality quantification, the right approach and what are quality and Qualimetry essentials?

•Chapter 4, Quality, Quality Modeling and Qualimetry

•This chapter aims to understand what is behind quality, quality modeling (including characteristics/sub-characteristics, attributes, metrics; with few quality model example), qualimetry, what are the essential characters, introduce some new concepts such as polymorphism that extend quality modeling.

Research question 2

Considering the set of software quality models, how to identify and decide which quality model is the most suitable for embedded software?

•Chapter 5, Quality Model Classification and Selection

•Following previous section, we understand that quality model is a pivot concept, there is no reference list of software quality models. So, if possible, how to classify and select quality model. This is completed with a systematic literature review and creation of a quality model landscape cartography.

Research question 3

Considering a quality model for a software product, how to operationalize it?

•Chapter 6, Quality Model Operationalization

•Here, we investigate how we can move from theory to practice with quality model, what issues prevent the quality model operation (development and use of quality model), and then review practical resolution of these issues, with some processes for polymorphic quality model tailoring.

•Chapter 7, Put into Practice

•We apply now what we have seen above to solve the company needs against a concrete example of 3 ECUs and one transverse function. We use a survey based on a reference quality model (i.e., from standard), identify consensus, build polymorphic quality models, and connect with metrics,

Research question 4

Can we have a unique reference quality model for software product?

•Chapter 8, Meta-Model: Software Quality Model Genome

•We analyze this key question which lead us to rather have a meta-model as a quality model aggregator for software product. Furthermore, we benefit from genetic knowledge to create a meta-model that can be used as the beginning quality model to construct polymorphic quality model, considering that there are variations of same concept over multiple quality characteristics, for example ; we are avoiding to discard contribution on quality models

Closure statements

•Chapter 9, General Synthesis and Research Perspectives

•This last chapter wraps-up this dissertation and discuss on some openings.

Chapter II. Context and Problems

1. Introduction

Precisely understanding our current context and our object of investigation are fundamental, otherwise we have no guarantee to look for the right answer to our problematic related to quality definition, assessment, and control.

The purpose of this chapter is thus to first review the industrial context, the automotive industry and more particularly the automotive embedded software, understanding the specificities of vehicle platform development and the standards that must be considered. For instance, a vehicle system follows a development life cycle of five years which is much longer than smartphone or computer system where the development life cycle is on a six months cadence.

As a second step, an enquiry on the current state of the art on quality modeling for embedded software is performed, identifying 33 distinct contributions in embedded system and software quality domains, covering a period going from 1999 to 2020. This literature review aims to confirm not only the existing solutions but also the actual gaps, or road-blockers, that prevent to solve our problematic.

Last, a discussion and critic analysis on this chapter ensure a proper overhaul and boundary identification of the research content and findings.

2. A complex industrial research context with a need of quality

As we already highlighted, the context of this research is an industrial one, and more specifically the automotive industry one. Indeed, our mother company -Renault SAS- designs, develops, and produces automotive vehicles. To achieve such development and productization, the company is structured into multiple engineering departments called Direction Engineering Alliance (DEA) (e.g., DEA-S for systems engineering department, DEA-L for software engineering department, DEA-M for mechanical engineering department, DQ-SC for quality and customer satisfaction department) where there exist many different job types with their specific vocabulary, process and tools. Therefore, this situation often makes it complex to interconnect people, processes, and tools.

Regarding the product, from a systems engineering point of view, a vehicle platform is a complex system [13], itself part of a system of systems (e.g., one use case that the Architecture Reference for Cooperative and Intelligent Transportation [14] addresses is where multiple cars are connected within a road system and its infrastructures). Furthermore, a vehicle platform can be considered as one generic complex system from which several vehicle variants such as mini-compacts, crossovers, super cars, vans, sport utility vehicles, convertibles, etc. are derived. As Fairley [9] described, this system is made of sub-systems classified under domains (e.g., chassis, body, infotainment, X-by wire, powertrain) connected together through a gateway and internal car networks. These subsystems are themselves composed of many distinct parts (e.g., electronic control units (ECU), electrical wires, mechanical elements) whose number and complexity depend on the type and version of a car. For instance, a premium car may have more than 60 distinct ECUs while a low-cost car has around 30 ones, and we see in Crolla *et al.* [15] (cf. Figure 2) that the overall number of ECUs is constantly growing over time. So, to produce high volumes of those parts, the company must rely on suppliers. However, the part specification and assembly remain of the responsibility of Renault SAS, and the company must be extremely cautious in their specification to ensure that during multi-system assembly, all separately developed parts fit and work well together. Unfortunately, suppliers are usually working at their own cadence and sometimes release ECUs (both hardware and software) that works fine alone but not all together.

Like any transportation systems (e.g., car, airplane, train, trucks, construction equipment vehicle), the overall product life cycle is also longer compared to other industries such as consumer electronics (e.g., computer, smartphone, tablet) where the design, development and delivery cadence is about 6 months. For a car, it usually takes five years since the initialization of the project to the serial production in the company factories. Moreover, the overall car operation lifetime is about 20 years, which means that during 20 years since the first produced car, the company must keep all materials required for any car repair, corrective maintenance, or inquisition (in case of accident with injury or death).

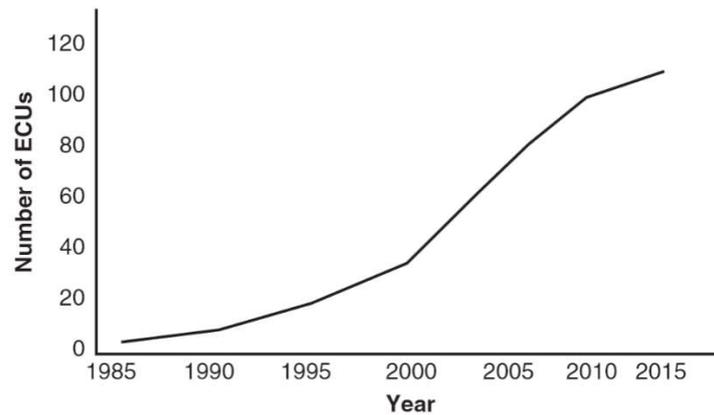


Figure 2 - Growing ECUs number per year for luxury cars (source Crolla *et al.* [15], chapter 14, figure 2)

This also means that quality of the produced car is fundamental, and has some strong economic impact, particularly in the case where quality level is not sufficient to handle such long lifetime period. In addition to this requirement on quality, the company has legal responsibility for any produced vehicle which are reflected in standards (e.g., security and safety [3], safety in autonomous vehicle context [16], cybersecurity [17]) and regulations (e.g., *European control on homologation* [18] linked to the tighten rules for safer and cleaner cars, the European Union *General Data Protection Regulation* [19] for personal data protection). All those requirement assessments are centralized and tracked by the DQ-SC department which uses customer satisfaction as main quality driver in addition to the compliance to these regulations and standards.

Coming back to our current problematic, the focus is set on the software embedded in the embedded systems that are ECUs.

Prior to Renault Software Labs integration, that embedded software was considered only as a subpart of the systems engineering deliveries. Thus, embedded software, developed by suppliers, was delivered as part of ECUs. Furthermore, embedded software quality was previously tracked as part of system quality through SIL (simulation in the loop) or HIL (hardware in the loop) intersystem tests. Since then, the company priorities evolved with the current age of industrial revolution (i.e., the age of software and digital [20]). These priorities are then put on car electrification, connected car and services, and justify the need of recognizing the importance of software in front of systems. Indeed, with connected services for example (e.g., *Firmware Over The Air* (FOTA)), software is also offboard (i.e., not in an ECU), and by consequence the software becomes highly configurable or changeable, at the opposite to the hardware. In addition, company internal teams develop either models or embedded software (e.g., FOTA source code) that are delivered to suppliers for integration in their own embedded software. The responsibility and process of development is consequently more complex and previous company approach doesn't fit anymore.

Fortunately there is a standard for automotive software development process: Automotive -SPICE [21] (A-SPICE). A-SPICE focuses on automotive software development², management and support processes, and improvement and assessment of the process capability level. It is neither a product quality assessment process, nor a product quality control process, but its guidelines recommend the use of ISO/IEC/IEEE 25010 [21]³ software product quality model (i.e., product view) and quality-in-use model (i.e., user view) to support quality assessment and control activities. Consequently A-SPICE guidelines must be followed for the development of all ECU embedded software. Note, ISO/IEC/IEEE 25010 is part of the ISO/IEC 250nn series [25], named System and software Quality Requirements and Evaluation (*SQuaRE*) which aim to cover quality requirement definition and evaluation.

To summarize, the automotive industrial context jointly with the vehicle as complex system, the development model with suppliers, and the current standard and regulation requirements are raising the overall complexity of our problematic to define, assess, control, or predict embedded software quality.

² The software development process is based on V-model [22].

³ In the previous Automotive-SPICE version, up to 2.5, the recommendation was to use ISO/IEC 9126 [24].

Moreover, compare to pure academic research, the research performed within an industrial context requires to address a concrete problem that a company has, and consequently the company expects to be able to use and apply these research findings. In other words, in this type of context, one unavoidable aspect of the solution under investigation is the operational aspect.

In our research work we must thereby address any issues that prevent practice of theory, which is not at all a straightforward matter when dealing with quality because quality is an elusive target as rightly highlighted Kitchenham and Pfleeger [26].

Therefore, in this context it is important to have a unified, operational, and appropriate way to define, assess, control, or predict quality of embedded software.

3. Quality modeling applied to embedded software development

In section 2, we learnt about the complexity of the industrial context of this research work with a clear need of quality for the developed embedded software. The goal of this section is to assess whether we already have a viable solution to our current problematic. In the negative case, we should be able to identify the gaps that we have then to address. So, after a clarification on our problematic about quality modeling applied embedded software development, we perform a state-of-the-art analysis on quality modeling during the development of the embedded system or software. This focus can be on process, specific development stages and/or product.

Today, there exists a myriad of embedded systems, more or less complex, from smart watch to printer, from internet of thing to car, rocket, or airplane. They are not only part of our everyday life but also are the depiction of the current technological revolution: *“the age of software and digital”* [20]. To succeed in this revolution, it is of first importance to define, assess, control, or predict the quality level of the software embedded in such systems.

This is achieved by applying properly quality modeling in embedded software development. Thus, quality modeling consists in the build, reuse, or adaptation of a quality model composed of quality characteristics / sub-characteristics, sometimes named attributes, together with metrics with the aim to define, assess, control, or predict quality [27] of a specific object of interest. Quality characteristics / sub-characteristics represent the qualitative side of the model while metrics represent the quantitative one. However, we remark that metrics are often missing at least partially from published quality models, preventing the quality models to be operational. Fortunately, one way to operationalize such a model is to use the Goal-Question-Metrics (GQM) paradigm [28]. Indeed, to measure quality characteristics / sub-characteristics, GQM starts by assigning goal to each characteristic/ sub-characteristic. Then the goals are derived into sets of questions, themselves completed with the proper metrics to answer to these questions. Unfortunately, the problem of defining quality characteristics / sub-characteristics is not solved with GQM.

As opposed to traditional computing systems shown in Figure 3, embedded systems have specific runtime constraints (e.g., safety, security, limited resources, real-time) and a specific application domain [29]. Both must be taken into account in the quality modeling applied to the embedded system life cycle stages [13], and which include software development. These specificities are expressed via quality requirements, characteristics / sub-characteristics, attributes, or again quality aspects, and synthesized into quality model.

Thus, to investigate the research question associated to our problematic, *“how quality modeling is applied to embedded software”*, we performed a review of the current state of the art in this field. To be more precise, we performed an exploratory literature review [30], a method close a systematic mapping study [31], to seek what already exist on *“quality model”* and *“embedded software”*, or *“embedded systems”* concepts over the software engineering online digital libraries⁴ and the scholarly literature search engine, Google Scholar.

We took 1968 as starting year for our investigation because this is the publication year of the two first published quality models in software engineering, term which emerged only few years before (1965) [32]: the quality assessment model of Rubey and Hartwick [33] and the reliability prediction model of Shooman [34].

⁴ Ieeexplore, ACM digital library, Springer, Scopus, and Web of Science

We found 33 main relevant research works published over the period from 1999 to 2020, and organized around four research focuses:

- quality model creation or use – for **42.4%** of the studies,
- quality characteristics or/and quality model identification - for **30.3%** of the studies,
- reliability growth model creation or use – for **15.2%** of the studies,
- miscellaneous work related to quality modeling in embedded software – for **12.1%** of the studies.

The following sections deep dive over each research focuses.

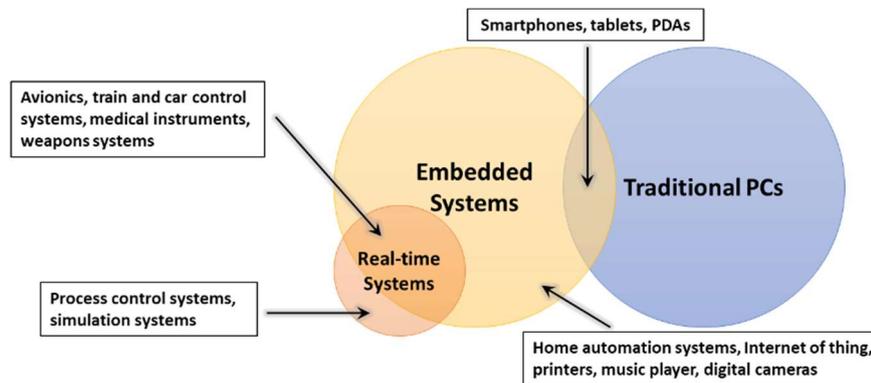


Figure 3 - Embedded systems in contrast to other computing systems (source Lepistö [35])

a. Quality model creation or use studies

Interestingly the principal research focus of these 33 studies, “*Quality model creation or use*”, is close to our research question. Indeed, it deals with the analyze of the embedded system and software of interest, the build of an appropriate quality model, and optionally with how to operate it.

As early as 2006, **Alvaro et al.** [36] designed a software Component Quality Model (CQM) focused on component-based software development for embedded systems. That study was performed under the perspective of reducing development costs and life cycles which can be achieved thanks to the reuse capability of software component, especially the component developed by third parties. CQM was based on ISO / IEC 9126 [24] and was composed of 7 quality characteristics, 29 sub-characteristics and 46 metrics. The authors decomposed sub-characteristics into the ones observable on “runtime” (e.g., stability) and the ones observable during “life-cycle” (e.g., testability). They also added marketability quality characteristic because they juggled about its importance for a certification process and the credibility it could bring to the component customers.

Completing this work linked to embedded software component, **Carvalho and Meira** [37] refactored CQM to came up with Embedded Component Quality Model (ECQM) and built a quality verification framework within the scope of certification. The authors aligned their work on ECQM with **ISO / IEC 25000 SQuaRE** [25], keeping the same 7 quality characteristics than CQM but with only 26 sub-characteristics, and used GQM to identify the corresponding set of metrics. Regarding the new quality verification framework, Carvalho and Meira introduced the Embedded software component Maturity Model, a maturity model similar to Capability Maturity Model for software [38]. Thus, depending on environment, safety; security, economic and domain potential damage and risk, the right maturity level for each component is selected, and then the evaluation technique(s) for each quality characteristics is applied accordingly. We note that the overall framework process lacked maturity even if the authors highlighted that it was used against Brazilian industry without any further detail.

In parallel, **Choi et al.** published [39] (2008) the Samsung software Component Quality evaluation Model (SCQM) to evaluate embedded software components for Digital TV systems. This in-house quality model was based on quality model definition from **IEEE 1061** [37] (cf. Figure 4 showing the corresponding ontology where quality is defined as one or more quality factors, themselves refined into quality sub-factors and finally into metrics), and several well-known quality models such as **ISO / IEC 9126**, **McCall** [41] & **Boehm** [42]. The authors identified a list of 8 quality characteristics jointly with 22 sub-characteristics but provided only few metrics as example to illustrate their work. During the creation of SCQM, the authors used expert Delphi method to identify strongness

relationships (i.e., strong versus weak) between characteristics and sub-characteristics. Consequently, they organized the sub-characteristics into two categories, common (i.e., invariant) and variance (i.e., depending on component quality requirements), where “variance” finally represents the candidate sub-characteristics used to tailor the SCQM depending on the quality requirements of the targeted Digital TV Systems.

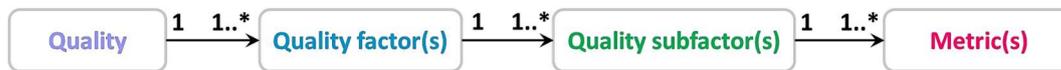


Figure 4 – Ontology of quality model definition from IEEE 1061 [40]

On another type of embedded system, **Peper and Schneider** were investigating the matching of quality of ambient intelligence systems from a quality of service point of view [43] (2009). Indeed, this embedded system, composed of software components, is an ad-hoc computer system (i.e., adaptive, distributed systems, acting like ad-hoc network) and therefore it provides not only services but also uses services from other ad-hoc systems. Those services must be negotiated between service provider and the client in term of provided versus requested functionality and quality. So, the authors suggested a service quality reference model for the matching quality problem between service provider and client components: they modeled quality (e.g., with a video system: video size, video rate, audio rate) as discretized into small number of intervals or values, defined per feature which can be similar, or shared, over other features.

Over a series of studies where the quality modeling apex was the **DeLone & McLean** success model [44], **Jeong et al.** aimed to adapt this model from its original scope, information system, to the software of embedded systems. They chose this model because of its frequent use by large audience, and the fact that it shows factors that influence organization. In the first study [29] (2012), the authors noticed that embedded systems, as opposed to desktop systems, deliver functionalities for specific domain, and also that their software can be considered as a lightweight component software. So, the authors replaced the three original quality input factors (information, system and service quality) of DeLone & McLean success model by a 5 quality factors (information, system, function, efficiency and maintenance quality) with the corresponding sub-factors, linked to lightweight component software. They continued in the next study [45] (2012) by regrouping the success model factors under three categories: system design, system delivery and system outcomes. By system design, they considered requirements of embedded system that a system designer must take into account such as interface, data flow, or quality of services, and redefined three quality input factors, with their own sub-factors: system quality, information quality and operation quality. The authors moved *use* and *user satisfaction* factors under system delivery, and *individual* and *organizational impact* under system outcomes. With the next study [46] (2012), Jeong et al. used ISO / IEC 9126 quality model to strengthen the quality input factors of the success model. They defined four criteria and sub-criteria. We retrieved *system quality* and *information quality* from DeLone & McLean success model but also two new ones: *software process quality*, representing a large part of ISO / IEC 9126 sub-characteristics, and *security quality*. Finally, with the last study [47] (2013) of this series, Jeong took into consideration an additional aspect of embedded software: software as a service (SaaS) with the cloud computing environment. Therefore, he identified and completed sub-criteria of system quality, information quality, software quality and security quality based on attributes for SaaS in cloud computing. Unfortunately, in none of these studies, the authors neither gave details on their construction choice, provided metrics, nor explained how to operate the model.

From another perspective **Mayr et al.** [48] (2012) investigated the creation of a code based quality model for embedded systems (i.e., ESQM). They noticed that existing solutions didn't provide any quality model that can be easily operated for assessment or control activities. So, in order to create such model with enough operationalization, they defined a four stages process jointly with the use of 336 metrics. They chose C and C++ code metrics evaluated with static analysis tools because their assumption is that embedded software is mainly written in these two languages. The first stage of this process is the quality requirement elicitation. They start from the embedded system and software characteristics to identify and group requirements linked to quality. In the second stage, these requirements are organized hierarchically, and code metrics are identified and assigned to each of them. With the third stage, they aim to associate these requirements and metrics to a higher level of quality characteristics, which they called “*quality aspect*”, targeting high level management. The quality aspect is achieved thanks to a mapping between these requirements and metrics with ISO / IEC 25010 quality view. To

consolidate the model, they introduced “*factor*” concept, and more particularly “*product factor*”, which is similar to **Dromey’s** quality carrying properties [49], used as the central element connecting requirements, quality aspects and measures. The purpose of the last stage is to finalize the quality model operationalization with threshold definition and weighted aggregation, based on **Trendowicz et al.** [50] software quality assessment method named SQUAD, a 5 steps approach using multi-criteria decision analysis. We noticed also that each stage last step consists in a quality gate validation where few “*experts*”, with a focus on completeness and validity quality factors and model, review the stage result. Finally based on their feedback, some adjustments are performed before moving to next stage. So, even if this approach is interesting with the operationalization assessment, similar to Trendowicz et al. method, the generalization to any embedded system is not straight forward and obvious since it started from the assumption of software is written in C or C++ and then favored code metrics. Moreover, the overall process is quite heavy to deploy, dependent of the set of selected “*experts*”.

Another attractive contribution is the creation, by **Ahrens et al.** [51] (2013), of a quality model, including its metrics, for objective evaluation of automotive software architecture. To define such quality model, the authors relied on general reference models such as McCall, Boehm, and ISO / IEC 9126 (see Annex 13 for details about these quality models), as well as on some architecture quality characteristics. The result is a quality model dealing with operating conditions (i.e., conformity, functionality, reliability), production costs (i.e., efficiency) and development costs (i.e., portability, mutability, reusability). Moreover, the authors proposed to close the gap between informal, qualitative evaluations methods, and objective quantitative measurement by comparing quality model result against a previous, or reference architecture result. Therefore, unlikely most of published contributions which have either quality model or metrics, Ahrens et al. ‘s contribution is an operational quality model that must be used to compare software architecture against a previous, or reference one to improve it, but unfortunately it cannot be used to assess automotive architecture quality.

On 2015, **Al-Sarayreh** [52], introduced a generic safety requirement model for any embedded and real-time software product where safety aspect is mandatory. This requirement model is based on 8 entity types which can be used as a basis for quality measurement, especially functional size. We note that the author didn’t refer to any software related safety standard like **ISO 26262** where some specific software functional metrics depending on safety level (e.g., ASIL for automotive), for instance, can be found. Moreover, that generic safety requirement model is not a quality model even if it can be considered as input for a quality model.

One year later, **Silva and Vieira** [53] introduced an innovative approach for quality modeling of satellite systems. They extended Orthogonal Defect Classification (ODC) [54] to cover embedded system, and more particularly critical ones with safety consideration. ODC is an efficient root cause analysis (i.e., it is 10 time faster compared to traditional root cause analysis) which allows to classify defects over three opener section attributes (i.e., defect removal activities, triggers, impact) and five closer section attributes (i.e., target, defect type, qualifier, age, source), and then get measures of quality linked to these attributes. Thus, the authors extend these attributes to include embedded system support (e.g., with safety), and their corresponding quality assessment was based on defect classification attribute measures and not from a quality model other than defect based.

Next, on 2017, **Garces et al.** [55] achieved a systematic mapping study on software of Ambient Assisted Living (AAL) for disabled or elder people. Due to its technical nature, an AAL is like an embedded system and its purpose falls under one the following scopes:

- independent living,
- health and care,
- occupation in life,
- recreation in life.

So, the goal of that literature review was to identify quality attributes and quality models defined, used, or assessed for AAL software. Over the 27 studies they found, Garces et al. retrieved only one quality model, OptimAAL based on ISO / IEC 9126 and ISO / IEC 25010, composed of reliability, availability, safety, integrity and maintainability, but this model was neither suitable for quality assessment, nor quality prediction of AAL software. In parallel, they collected 97 quality attributes where the most important ones were security, freedom from risk, usability, reliability, adaptivity, availability, fault tolerance and performance efficiency. To structure

these quality attributes into a quality model, the authors defined a mapping algorithm to map these attributes against first ISO / IEC 25010 quality models and then against ISO / IEC 9126 quality models and metrics, adding in passing adaptivity quality characteristic to ISO / IEC 25010 existing quality characteristics. Their reinforced this result by defining a taxonomy based on the four AAL scopes. The resulting quality model was the preliminary version of the authors' Quality Model for Ambient Assisted Living Systems (QM4AAL) [56]. Nevertheless, the authors highlighted the need to get industry involved to increase the maturity of the quality attributes and models, especially that out of all these 27 studies, only one was evaluated in industrial context.

The last relevant contribution with regard to “*quality model creation or use*” theme is the definition of a Software User Review Defect Corrective Model (SURDCM) from **Kasiviswanathan and Ramalingam** [57] on 2020, exercised against a 3D-drawing application. In this paper, the authors are considering that a high-quality product means that that product very few bugs, and consequently one of the major areas which requires strong attention to produce quality product is requirement analysis: good quality requirements are essential. So, SURDCM is done *via* a process starting from “customer requirement” up to “deployment and maintenance”. The idea is to perform first an analyze with Kano model [58] (i.e., satisfaction versus unsatisfaction based on observer, or customer perception) to understand the impact of each requirement against customer. Then, based on this analysis, completed with a correlation coefficient of customer satisfaction (i.e., extent of satisfaction and extent of dissatisfaction) and Saaty's Analytic Hierarchy Process (AHP) [59], the requirement prioritization is finalized. Indeed, AHP is an expert analysis process which evaluates pairs of alternatives (in this case the requirements) based on criteria (in this case they are performance, design, security, and usability) to determine the most important alternatives thanks to eigenmatrices, values, vectors and consistency index. The requirements are then designed and implemented relatively to their priority. The last SURDCM step is a Software Failure Mode Effects Analysis (SWFMEA)[60] which is triggered once the code is generated and entered the testing phase. SWFMEA is a failure mode analysis of the impact of both functional and non-functional software failures which helps to find any required corrective or evolutive action on requirements. So, in this contribution, the requirement quality evaluation is not achieved via a quality model with metrics but rather through a complex process whose purpose is to enhance requirements depending on failures.

We learnt with these studies related to “*Quality model creation or use*” that there is no obvious nor universal solution for modeling quality of embedded software. Some studies focus on requirements while some others on architecture or implemented code, but there is also no quality model usable during all development stages. Moreover, only few contributions investigate the quality model reuse, or the operationalization by providing metrics and a process, often complex to deploy, for objective, or subjective, quality evaluation.

b. Quality characteristics or/and quality model identification studies

“*Quality characteristics or/and quality model identification*” is the second principal research focus of the studies we identified. This corresponds implicitly to a subset of our main research question because we are identifying what model or quality characteristics is belonging to embedded software.

The first and earliest contribution on this research focus was done by **Wijnstra** [61] on 2001. He looked for identifying the quality attributes and aspects of medical systems which are composed of devices made of hardware and embedded software. He remarked that quality attributes can be derived into system, hardware, mechanical and software aspects, from an architecture point of view, and by consequence, quality attributes and aspects are both complementary. Thus, Wijnstra identified a list of seven quality attributes for embedded system in medical domain (reliability, safety, functionality, portability, modifiability, testability, serviceability) with three specificities belonging to modifiability for medical (configurability, extensibility and evolvability) and completed with various aspects that must be took into account: for instance, operational, start-up, shutdown, error handling, graceful degradation. Finally, even if there is a clear quality attributes for embedded systems with relevant explanations, neither quality model, nor metrics are given in this contribution which foster its replication or operationalization.

One year later, **Purhonen** [62] sought to identify quality attributes for digital signal processing embedded software in wireless system. The author successfully identified three main quality attributes (performance, cost and variability) which he derived from quality goals coming from either standards, customers or development

source. Furthermore, he used **Barbacci et al.**'s taxonomy [63] to classify these quality attributes. This taxonomy is organized over 3 factors which are stimulus (i.e., events causing architecture response or change), response (i.e., measurable or observable quantities to assess architecture) and architectural (i.e., parameters that define architecture decision). Then, despite that software development for digital signal processing for wireless systems is quite similar to some other embedded software developments, the resulting quality model in this contribution covers only three quality attributes, without any metrics, and unfortunately the use of a taxonomy for its refinement was not sufficient to generalize it to any embedded software.

Another relevant contribution in the field is the investigation done by **Akerholm et al.** [64] (2004) to spot the most important quality attributes for component-based software embedded into vehicular systems (i.e., cars, construction equipment vehicles, heavy trucks, trains). The authors' methodology can be summarized over three main steps. The first one was to build a list of relevant, to vehicular systems, quality attributes collected from existing literature. These quality attributes are mainly associated to non-functional properties, also called extra-functional properties, or -ilities. The second step consisted in interviewing experts from several companies to sort that list, creating groups from the most important ones to less important ones:

- safety, reliability, predictability
- usability,
- extendibility, maintainability,
- efficiency, testability,
- security and flexibility.

In the last step, the authors used **Larsson's classification** [65] against these 10 quality attributes to complete their description. This classification is based on the categorization of on how a quality attribute could be facilitated by component technology, and where support of a quality attribute should be implemented. Thus, the classification-based description can be one or a combination of: directly composable, architecture related, derived attribute, usage dependent and system environment context. This list of quality attributes cannot serve as it is for embedded software quality evaluation, but rather as a starting point for quality model definition of vehicular systems.

On 2008, **Paulitsch et al.** [66] aimed to analyze the aerospace domain to build a list of non-functional requirements for avionic embedded systems. These non-functional requirements are assimilated to embedded system quality characteristics, or attributes, and therefore must be reflected in both hardware and software architecture of such systems. Thus, the authors came up with a list of 10 requirements related to dependability, performance, development, and operation (i.e., availability, diagnosis, integrity, maintenance, obsolescence, temporal performance, testing, safety, schedulability, and security). Paulitsch *et al.* indicated also that this list is not exhaustive but together with their analysis and explanations, they have provided a basement to be used and integrated in further work related to non-functional requirements in avionic context.

At the same time, **Sherman** [67] shared a list of 30 quality attributes for embedded systems, where few of them were specific physical attributes (e.g., weight, physical size). What makes this study interesting is the approach that the author used to establish this list. Indeed, he identified quality attributes based on the evaluation of 11 embedded system architecture trade studies made with Architecture Tradeoff Analysis Model (ATAM). These ATAM studies were performed with the main system, hardware, and software stakeholders to review architecture solutions and then decide which architecture design tradeoff to choose. So, Sherman used those architecture decisions to find which quality attributes were cited, and consolidate their knowledge with their citation number, the number of studies where they were referred, and whether or not their definition changed between studies (this happened for two third of the attributes).

To continue on embedded system architecture, **Guessi et al.** [68] (2012) led a systematic literature review [31] to identify quality requirements and constraints for reference, and software, architectural description of embedded systems. The authors found a set of 12 architecture concerns, declined into quality requirements: adaptability, correctness, dependability, fault-tolerance, interoperability, knowledge reuse, maintainability, performance, power consumption, reliability, safety, and timing. In addition, they remarked the existence of architecture information types (components, interactions, interfaces, synchronization, timing, configuration,

constraints, dataflow, processes, threads, connectors, concurrency, abstraction layers, concepts, design decision, events, states, behavior) that they decided to associate with these quality requirements. They completed their survey with an analyze of the most frequent architectural language description (e.g., the two main ones are the Unified Modeling Language (UML) and Architecture Analysis and Design Language (AADL)).

We note that based on Guessi *et al.*'s contribution, we have an easier way to find and associate metrics to quality requirements through these types of information. We depict our proposal in Figure 5 where types of information act as the central joint between quality requirements and metrics. Indeed, one or more types of information (e.g. interfaces, timing, dataflow, events, states) can be associated to a quality requirement, and each type of information can be measured thanks to one or several metrics.



Figure 5 - Use of type of information as a helper to find and associate metrics to quality requirements, based on Guessi *et al.* [68]

Another relevant systematic literature review is the one from **Oliveira *et al.*** [69] (2013). Their research objective was to get a panorama about quality models and quality attributes defined, assessed, or used for embedded system. The authors identified 11 studies where 27.3% of them proposed a quality model while the other 72.7% concentrated only on quality attributes. Among all the information they extracted from their analysis, we learn that 54.5% of studies were developed from document analysis (e.g., systems requirements), 36% from personal experience and 27.3% from literature reviews. Also, 45.4% of the study contributions were evaluated through academic or expert opinion, 27.3% via their application in embedded system, and the remaining 27.3% gathered no clue on an evaluation method. In term of quality attributes, the authors identified 18 attributes (% of presence in the studies):

- 91%: maintainability, reliability,
- 64%: security, safety, functionality, efficiency, portability, testability,
- 45%: performance, usability,
- 36%: availability, extensibility, reusability, cost,
- 27%: fault tolerance, recoverability / repairability, interoperability, flexibility.

The authors found also that defining quality model and attribute is complex task since it requires multiple sources of information and there was still need for a quality model to be widely accepted or adopted. In fact, there was no strong proposal from any of those studies, and only two studies showed some industrial evidence of the use of their quality model and attributes.

In the continuity of the study series, we saw in the section a, where the assumption was completion of DeLone & McLean success model, **Jeong *et al.*** [70] (2014) search identified what they considered as the most important quality characteristics, as well as their weights, for secure embedded system. They used the Analytic Network Process (ANP) method to compute the value of the degree of influence for each quality characteristics. AHP and ANP are two closed methods, both based on the evaluation of pairwise importance relation between each characteristic, and then on eigenmatrix, values and vectors to determine importance of quality characteristics. However, AHP assumes the independence of the factors used for decision making which is the opposite behavior the authors looked for. We note that surprisingly the authors didn't include any security related characteristic for such secured embedded system, but rather they tried to see which characteristics may be the most important in this context.

Bianchi *et al.* [71] (2015) contributed as well to this topic but from a slightly distinct system perspective: they focused on system of systems. And because a system of systems often includes embedded systems (e.g., internet of objects / things, automotive, avionic, medical), we decided to include their contribution. So, over a systematic literature review, the authors aimed to determine what quality attributes are associated to a system of systems mainly due to its specific architecture, and to the interdependencies or interoperability relationship between the different systems that compose it: for instance, safety or reliability are complex to determine since any systems can impact in different ways and multiple workarounds can exist also. In their survey, Bianchi *et al.* retrieved from the literature a list of 56 quality attributes where the five most cited are security, interoperability, performance,

reliability, and safety. Regarding their application domains, the-most frequent ones are military, IT systems, smart grid, and automotive. The authors consolidated their survey by determining the amount of these quality attributes which are present in ISO / IEC 25010, actual standard reference for system and software quality model. They figured out that 48% of them are not present in this standard, nevertheless these missing attributes are not the major cited ones, and it appeared that complex interdependencies of attributes are hardly achievable with a hierarchical structure such as ISO/ IEC 25010 quality models, but rather with a general graph structure, for instance.

To complete the quality characteristics or/and quality model identification studies, **Zouheyr Tamrabet et al.** [6] (2018) did a survey on quality attributes and quality models for embedded software. The authors started with a description of what an embedded system is, and then clarified software quality concept, including quality attribute and model. Next, they established three groups of the most represented quality attributes in literature:

- standard characteristics: reliability, usability, maintainability, portability, performance, functionality,
- sub-characteristics: availability, fair tolerance, interoperability, adaptability, recoverability, reusability, testability,
- specific attributes: efficiency, safety, flexibility.

In their conclusion, they stated that there were neither consensus on a list of quality attributes, nor on quality model for embedded systems, and therefore in their future work, they aim to propose a generic quality model *“that encompasses relevant quality attributes in order to define the quality of embedded software”*.

Over these 10 studies related to quality characteristics or/and quality model identification, we remark that the authors use either quality attribute or quality characteristic for the same concept but, in all cases, these characteristics are non-functional quality characteristics, and they are mainly associated to requirements, architecture design or constraints.

Moreover, none of these surveys introduce any metrics for quality characteristic, which are essential from an operational aspect.

Finally, like Zouheyr Tamrabet *et al.*, we can state that we didn't notice any consensus on quality attributes and quality model for embedded system and software through those contributions.

Furthermore, from what we saw, we may find a subset of quality attributes (e.g., maintainability, reliability, safety, security, and testability) which are common to all embedded system or software.

However, each embedded system or software case is different, and that variety, or variant, must consequently be reflected in the quality model and characteristic solution. Therefore, we may have some doubts on the feasibility to build a generic quality model or come-up with a generic list of quality characteristics.

c. Reliability growth model creation or use studies

Our third group of studies is on *“Reliability growth model creation or use”* and thus consist in the stream of work about the creation or use of reliability growth model. The objective of this type of model is to determine an implicit or statistical quality model to evaluate and predict quality of embedded software prior to its operation, for example. It is a complementary approach to the previous group of studies.

With Khoshgoftaar and Allen [72] (1999) and Khoshgoftaar *et al.* [73] (2002) studies, we have an illustration of some of the research performed **Khoshgoftaar et al.** They explored the construction and use of reliability growth models to predict which software modules for telecommunication embedded system were the most fault prone. The developed quality models were statistical models built from that telecommunication system historical data, and their inputs were software process, product, and execution metrics. Unfortunately, the models cannot be applied to other embedded systems because of the specific historical data set used to determine the statistics models.

Regarding reliability evaluation for another type of telecommunication system, **He and Li** [74] (2012) developed an alternative approach to determine the software reliability of voice over internet phone. Their inspiration came

from safety analysis. Indeed, they achieved software reliability analysis and improvement based on a sequential combination of Software Failure Tree Analysis (SFTA) and then Software Failure Model and Effects Analysis (SFMEA). SFTA is used to retrieve the top events that cause failures as well as their related basic events through a failure tree. SFMEA is the application of Failure Model and Effect Analysis (FMEA), originally developed by US Military, to software domain since 1979. FMEA and therefore SFMEA are an analysis of the cause of failure mode for top and basic events. So even if He and Li solution was not a reliability growth model, the overall behavior of their approach is quite identical to evaluate and then increase the reliability of embedded system.

In 2016, **J. Liu et al.** [75] introduced a reliability growth model combining reverse engineering approach. That reliability model aimed to evaluate embedded software quality of electric smart meter. The author innovation was to reverse engineer binaries (i.e., compiled and lined code, as opposed to high-level language source code) in order to reconstruct the software program into a high-level language. Then they evaluated the reliability of that reconstructed software, based on utilization-oriented fault model and the reconstruction of the software control flow graph. This contribution is similar to Goel-Okumoto software reliability model (i.e., G-O model) [76] (1979) but the authors highlighted that their model was simpler to use. The obvious drawbacks are that the deployment of that solution is complex, it requires to have binaries artifact and the reverse engineering software code result is subject to interpretations which may differ from original software code.

The contribution of **S. Juneja et al.** [77] (2019) is different from the above research works. The authors proposed a basic random model as reliability growth model for general embedded system. They considered both hardware and software failure occurrences to elaborate their model and once completed, they compared this model against five widely used software reliability growth models. The comparisons were done through simulations of model failure rate with Matlab tool. As expected, their proposed model gave the best result when predicting embedded system reliability. The best result meant that model had the lowest failure rate because the authors characterized embedded systems as mission critical ones and implicitly, assumed that such systems were developed to have the highest reliability than other software systems (e.g., desktop computer, server). However, this mission critical assumption is accurate for a subset of embedded systems but not for all embedded systems (e.g., smartphone, smartwatch, printers, camera, tablet).

To sum up about reliability growth model, these studies are important because they show a complementary view on quality modeling for embedded software. The reliability evaluation and prediction are achieved in most of the cases with implicit or statistical models.

Unfortunately, these types of model prevent the direct model utilization to other embedded software because the model was elaborated from a specific historical data set.

Finally, concerning the other minor approach which involves reliability analysis process, even if the process can be generalized to any embedded software, that solution does not correspond to neither quality model, nor quality characteristics.

d. Miscellaneous work related to quality modeling in embedded software

Our intent within this last group of research works is to collect some further studies that may of an interest in quality modeling for embedded software, despite the fact that they don't propose any quality model or quality characteristic.

The first research study, driven by **Rohloff et al.** [78] (2006), introduced a practical solution to get real-time feedback to control system quality and produce evidences for certification. That solution made of real-time utility quality measures, was elaborated specifically for distributed real-time embedded systems, with a focus on adaptive reflective middleware systems. The authors considered two aspects here, real-time and quality of produced data. That last aspect is key for certification evidences and an enhancement of that study could be to attach these quality measures to a "utility" quality model.

With this other survey of **Oliveira et al.** [79] (2008) on space system design, the authors investigated the potential existence of relationship between metrics, more precisely between software quality metrics (e.g., lines of code, McCabe complexity, nested block depth, efferent coupling) and embedded software or system metrics (e.g.,

memory usage, performance and energy consumption metrics measured in a worst case scenario and also in a best case scenario). The leading idea behind this comparison is to find correlations between these two sets of metrics in order to support architecture trade-off decision between performance, maintainability, and component reusability, to shrink time to market.

The last two studies of this group are concerning the same embedded system, automotive.

In 2012, **Stürmer and Pohlheim** [80] created an approach to measure and assess model-based design software. The authors highlighted that embedded software development evolved to include frequently model-based design. We remark that this statement is also aligned with ISO 26262 [5] and DO-178-C [7], for instance. So, to evaluate quality, the authors defined a quality model to aggregate measures from model analysis, issue tracking, test management and requirement management. They also included assessment of quality operations (e.g., testing, reviews) for all the generated development artifacts, and not only from model or code. There was no reference to quality characteristic likes in legacy quality models such as McCall, Boehm, ISO / IEC 9126 but rather the quality model in this study was a reference to the artifact that were analyzed and assessed. In other words, the author's quality model reflected the degree of success of the quality operation and not quality characteristic.

From another research perspective, **Bouquet et al.** [81] (2018) introduced model quality objective concept. The model quality objective was an adaptation of software quality objective concept for model-based design, frequently used in embedded system development. The authors described 16 model quality objectives that were declined into requirements to ensure a high level of quality of embedded software which are developed through model-based design.

These four contributions taught us few additional considerations that we must consider for quality modeling applied to embedded software.

At first, we have to pay attention to quality of data, especially when generating the mandatory evidences for certification.

We must also investigate potential correlations between metrics and quality characteristics to support trade-off decisions (e.g., on architecture design).

And finally, an embedded software is not only a matter of code implementation, but as well from model-based design.

e. Limits to existing quality modeling applied to embedded software development

Over the 33 studies of this exploratory literature review, we saw a diversity of approaches and research works, summarized in Table 2. We observed that this diversity is the reflection of the myriad of existing or possible embedded systems and software, each of them having its own specificities and requirements. We remarked also that often, quality characteristics for embedded software are non-functional quality characteristics which are mainly expressed or associated to quality requirements, architecture design or constraints.

Nevertheless, among all embedded software, and thanks to systematic literature review studies, it is possible to identify a limited set of shared properties, or quality characteristics likes maintainability, reliability, safety, security, and testability.

Moreover, as Zouheyr Tamrabet *et al.* stated, there is no consensus on the adoption of quality attributes and quality model for embedded system and software, despite the existence of standard such as ISO / IEC 25010 which is by construction the result of an international work and collegial decisions. But this is not a surprising finding because the embedded software variety, or variant, must be retrieved also in the development of quality model and quality characteristics.

So, there is no right and unique solution yet to our question "how quality modeling is applied to embedded software".

Context and Problems

TABLE 2 – SUMMARY OF STUDY KEY POINTS

Id	Reference	Year	Type of work stream	Embedded System Domain	Embedded Software Domain	Study relies specifically on
S01	[72]	1999	Reliability growth model creation or use	Telecommunication systems	Software module	Classification And Regression Trees algorithm
S02	[61]	2001	Quality characteristics or/and quality model identification	Medical	Component-Based Software Development	-
S03	[73]	2002	Reliability growth model creation or use	Telecommunication systems	Software module	Case-based reasoning model
S04	[62]	2002	Quality characteristics or/and quality model identification	Wireless system	Digital Signal Processing software, Commercial Off-The-Shelf	-
S05	[64]	2004	Quality characteristics or/and quality model identification	Vehicular (i.e., cars, construction equipment vehicles, heavy trucks, trains)	Component-Based Software Development	ISO/IEC 9126
S06	[78]	2006	Miscellaneous related work: real-time utility measure to control system and use them as evidence for certification	Distributed real-time embedded systems	Adaptive reflective middleware software	-
S07	[36]	2006	Quality model creation or use	Generic	Embedded software component, Component-Off-The-Self; Component-Based Software Development	CQM model, ISO/IEC 9126 model
S08	[66]	2008	Quality characteristics or/and quality model identification	Aerospace, Integrated modular avionics	Commercial Off-The-Shelf	Multiple Independent Levels of Security/Safety, ARP4754, ARP4761, DO178B, model-based analysis
S09	[39]	2008	Quality model creation or use	Digital TV	Software component, Commercial Off-The-Shelf, open source	IEEE 1061, McCall model, Boehm model, ISO/IEC 9126 model
S10	[67]	2008	Quality characteristics or/and quality model identification	Generic	Architecture	Architecture Tradeoff Analysis Model (ATAM)
S11	[79]	2008	Miscellaneous related work: investigate the existence of relationship between metrics	Design of space system	Generic	-
S12	[37]	2009	Quality model creation or use	Generic	Embedded software component	CQM (S07), ISO /IEC 25010 model, CMM
S13	[43]	2009	Quality model creation or use	Ambient intelligence systems	Ad-hoc computer systems	-
S14	[29]	2011	Quality model creation or use	Generic	Lightweight embedded software component	DeLone & McLean success model
S15	[45]	2012	Quality model creation or use	Generic	Software component	S14, DeLone & McLean success model
S16	[46]	2012	Quality model creation or use	Generic	Software component	DeLone & McLean success model, ISO/IEC 9126 model
S17	[80]	2012	Miscellaneous related work: Creation of an approach to measure and assessment of model-based design software	Automotive	Generic	ISO 26262
S18	[74]	2012	Reliability growth model creation or use	Voice over Internet Phone	Generic	Software Failure Tree Analysis, Software Failure Model and Effects Analysis
S19	[48]	2012	Quality model creation or use	Generic	Generic	Systematic literature review, S02, S05, McCall model, Boehm model, ISO/IEC 9126 model, ISO/IEC 25010 model, SQUAD

S20	[68]	2012	Quality characteristics or/and quality model identification	Generic	Architecture	Systematic literature review
S21	[47]	2013	Quality model creation or use	Generic	Software as a Service	S16, DeLone & McLean success model, ISO/IEC 9126 model
S22	[51]	2013	Quality model creation or use	Automotive	Architecture	McCall model, Boehm model, ISO/IEC 9126 model
S23	[69]	2013	Quality characteristics or/and quality model identification	Generic	Generic	Systematic literature review, S02, S04, S05, S08, S09, S10, S12, S13, S14, S20, S23, McCall model, Boehm model, ISO/IEC 25010
S24	[70]	2014	Quality characteristics or/and quality model identification	Secure embedded system in sensor network	Networking software, Software As A Service	DeLone & McLean success model, Analytic Network Process
S25	[52]	2015	Quality model creation or use	Generic	Embedded, real-time software product with safety	ISO 25021
S26	[71]	2015	Quality characteristics or/and quality model identification	Systems of Systems	Generic	Systematic literature review, ISO/IEC 25010 model
S27	[53]	2016	Quality model creation or use	Satellite systems	Generic	Orthogonal Defect Classification
S28	[75]	2016	Reliability growth model creation or use	Smart meter	Reverse engineering	Goel-Okumoto software reliability model
S29	[55]	2017	Quality model creation or use	Ambient assisted living	Generic	Systematic mapping study, OptimAAL model, ISO/IEC 9126 model, ISO/IEC 25010 model
S30	[81]	2018	Miscellaneous related work: introduction of model quality objective for model-based design	Automotive	Model based design	Software quality objective
S31	[6]	2018	Quality characteristics or/and quality model identification	Generic	Generic	S02, S09, S10, 12, S14, S20, S21, S22, S23, S26, S28, ISO/IEC 9126 model, ISO/IEC 25010 model
S32	[77]	2019	Reliability growth model creation or use	Generic	Generic	Goel Okumoto Model, Jelinski Moranda Model, Littlewood Verral Model, Generalized Goel Model, NHPP Model, Basic Random Model
S33	[57]	2020	Quality model creation or use	3D Drawing embedded software	Generic	Kano model, Analytic Hierarchy Process, Software Failure Mode Effects Analysis

Another limit that was not resolved with these research studies is the operationalization of quality model but, nonetheless, we found some elements of answer in few of these studies. For example, Arhens *et al.* [51] highlighted this operation aspect by emphasizing the importance of metrics, and aggregation of metrics together with quality characteristic and quality model. We note also that most of these surveys didn't introduce any metrics for quality characteristics. A second operational blocking aspect, happening in quality assessment, control, or prediction activities, is the objectivity versus subjectivity of the quality evaluation. Indeed, the specification of threshold values used in decision making is often not obvious, and consequently decision in those activities may be subjective. An alternate solution is to use relative threshold or evaluate quality against a reference, or previous embedded software; for example, when improving the quality of a software over multiple consecutive releases. In that case we are able to make objective quality evaluation and decision, but its usage cannot be generalized to any decision making.

Next to that exploratory literature review and analysis of the main limits, we have enough elements to detail our research questions.

4. Research questions

In the previous section 3, through an exploratory literature review, we analyzed 33 research contributions related to quality model and quality characteristics for embedded software, or systems, and conclude that there does

not exist a generally adopted solution to model quality of embedded software. However, to support our company critical needs, we have to define, assess, control, and even predict embedded software quality as well as the quality of its development. In addition, the resulting quality indicators must be robust to support embedded software variants, which are the usual case in automotive domain, constraints coming from standards and regulations, and mandatory evolution accordingly to the development life cycle stages.

We should also manage an overall complexity due to:

- An automotive system is a complex system to build with multiple elements that require alignments and interoperability,
- Systems and software interface require alignment between process, technology, protocols, teams, and tools,
- Alignment over the complete vehicle, architectures, many distinct jobs, and specific vocabulary,
- Safety compliance leads to complex and costly tasks (e.g., ASIL D recommend strongly to perform Modified Conditions / Decision Coverage assessment which is costly and complex analysis),
- Mandatory continuous software evolution required to avoid obsolescence (first law of Lehman [82]),
- Evolving standard and regulations to support,
- Internal development process execution is complex due to sub-systems split over many teams and suppliers,
- Difficulty to have objective quality assessment; for instance, management of suppliers with offshore tracking where quality reporters tend to escape strong engagement by reporting an average quality level.

Therefore, in this context, it is important to have a unified and right way to define, assess, control, or predict embedded software quality.

If we take Automotive-Spice, the guidance is to use ISO/IEC 25010 for quality model. Nevertheless, this standard lack of metrics, for example, and require a customization of quality model per ECU or transverse software to be operationalized. This statement is aligned with Wagner *et al.* survey [83]. Indeed, in this survey Wagner *et al.* found that only 28% of the survey participant companies use quality model from standards and over them 71% customize them.

So, obviously a quality model coming from standard should not be used as it is. We note that such quality models are often too generic because they aim to cover a wide range of cases. These models are regularly ambiguous in their quality characteristic definitions, with a risk to be misunderstood and then hard to use, and they usually require customization. From another perspective, if we look for quality models for software, we can find more than 450 more or less distinct quality models.

Thus, it is not trivial to select one knowing that usually they are very case specific, hard to reuse or adapt. We remark that creating a new quality model for automotive embedded software can drive us to that same situation: a new specific quality model that could be hard to reuse or adapt. Thus, there seems to be a discrepancy between to be at the state of the art, with a collection of more than 450 models, and the fact that we must consider standard like ISO / IEC 25010 despite its issues.

Finally, the solution should consist in a trade-off between these two states and consequently, we should consider an alternate approach like qualimetry.

This guides us to the following research questions:

Research Question 1

Is Qualimetry, as the science of quality quantification, the right approach and what are quality and Qualimetry essentials?

Our first step is undeniably to understand and synthetize with clarity and unambiguity the concepts of quality, quality modeling and qualimetry.

Once their main characteristics described, we should be in a position where we will be able to confirm or deny that qualimetry is the right approach to use for quality quantification of automotive embedded software with regards to other approaches.

This research question 1 can then be refined into four research sub-questions:

Research Sub-question 1a What is the essence of quality?

Based on an exploration and analysis related to quality and their directly linked key concepts, we should end with a clarification and synthesis of quality concept.

Research Sub-question 1b What is the essence of quality modeling and particularly in software field?

Like for quality, we aim to analyze and describe quality modeling concept. An investigation of the key contributions in that domain will certainly foster its understanding and mastery.

Research Sub-question 1c What is qualimetry, and is this approach the right one for our needs?

Qualimetry is often misunderstood and therefore we should analyze this science of quality quantification to synthesize and popularize its essential characteristics. Moreover, we have to compare qualimetry to existing alternate approaches in order to confirm that qualimetry is the right approach to use for our needs or deny and therefore realign our research by selecting the right one. Thus, the answer to this sub-question is going to lead our research work.

Research Sub-question 1d How to unify diversity and time evolution in quality modeling?

In the automotive domain, we have to manage a wide variety, constraints and complexity. The variety comes not only from the car variants to be supported, but also from the embedded software of ECUs. The constraints are mainly linked to the requirement to be compliant with regulations and standards (e.g., A-SPICE, ISO/IEC 25010, ISO 26262) but also specific requirements from stakeholders. About complexity, it is due to automotive product development requirements, environment, and evolution with respect to the life cycle stages.

Hence, we must find a solution, or a “*mechanism*”, that includes all those elements into a unified quality modeling approach.

Research Question 2 Considering the set of software quality models, how to identify and decide which quality model is the most suitable for embedded software?

Now that we have explored the key concepts about quality, quality modeling, and acknowledged which quality quantification approach is the most suitable for our needs, the next step is to apply that approach to model quality of embedded software.

In Chapter I, we highlighted the existence of many quality models for software, with evidences in Kläs *et al.* [10], Thapar *et al.* [11], or and Oriol *et al.* [12], and even for embedded software in Chapter II.3. So instead of creating a quality model from scratch, with a non-negligible risk to reinvent the wheel, our research strategy is to take benefit of this pool of existing quality models and discover which one, if any, is the most suitable for automotive embedded software.

Therefore, replying to this second research question requires to collect and list quality model within the scope of software, englobing embedded software, classify them and identify decision criteria to decide which quality model(s) answers best to our needs.

We rephrase these tasks into the following research sub-questions:

Research Sub-question 2a Considering software scope, what is the set of existing quality models?

The goal is to collect and consolidate an exhaustive list of quality models for software by proceeding in a systematic literature review as detailed by Kitchenham and Charter [31].

We are not limiting the research time range, but from our early exploratory literature review, the oldest publications of software quality models we found were the quality assessment model of Rubey and Hartwick [33], and the reliability prediction model of Shooman [34].

So, we suppose that our oldest finding won't be older than 1968, or at least than 1965, year during software engineering concept emerged.

Research Sub-question 2b

Considering a set of quality models, how to classify these quality models, what are the methodology, the criteria, and the characteristics to use?

Then, to analyze, get some fruitful benefits and make decisions based on that exhaustive list of quality models, we must structure its knowledge.

Therefore, we are going to apply the right taxonomy, defining an appropriate set of criteria, or taxons, to index and classify these models.

The expected result from this research sub-question, in addition to that classification methodology and its criteria, is a cartography depicting the current panoramas of quality models for software.

Research Sub-question 2c

Considering at least two quality models, how to compare together quality models, and can we define a reliable distance formula between quality models?

Through this sub-question, we aim to clarify the relevant characteristics and criteria to compare quality models together, keeping in mind that we are looking for the most appropriate quality model for our needs, from a classified list of quality models.

The choice or definition of a formula to evaluate the distance between two quality models can also serve for decision purpose, for example.

Research Sub-question 2d

What is the most appropriate quality model is for embedded software in automotive?

This is the conclusion step of our research question 2.

From the classified list of software quality models we build, the key information we extracted, the comparison criteria we identified, and our knowledge about automotive embedded software, we are now able to conclude on which quality model is the most appropriate one for embedded software in automotive.

Research Question 3

Considering a quality model for a software product, how to operationalize it?

The quality model identification, or creation, corresponds to the first phase in quality modeling: the quality definition.

In the next phases, we operate the quality model. This operationalization enables its quality definition completion, quality evaluation activities (e.g., assessment, control, or prediction), and quality model evolution or adaptation management. However, there exist challenges and issues that may prevent the development or use of quality models as Thapar *et al.* demonstrated [11].

So, through this third research question, we set our focus on identifying potential issues that may prevent quality model operations, on finding practical solution(s) to these potential issues and then, on applying our findings against our needs for automotive embedded software. The application success to this real-world use case is the proof that we have answered to our problematic. We note that the

operationalization aspect is also fundamental for our research work to fulfill our company expectations (see Chapter II.2).

The following research sub-questions address that problem decomposition:

Research Sub-question 3a

What are the main challenges and the issues that prevent operationalization of quality model?

As we could expect, this first research sub-question explores the main operational challenges and issues that prevent quality model development and use. Their synthesis is going to be reflected in a list of challenges and issues with their own descriptions.

Research Sub-question 3b

What are the practical solutions to those challenges and issues?

For each of the listed operational challenges and issues, we target to find practical solutions to all of them. Practical solution means that the solution not only solve the linked challenge or issue, but also it must be usable and deployable. Therefore, general statement or theoretical solution alone should be prohibited for those solutions.

Research Sub-question 3c

What is the process to ensure quality model operationalization?

Over the two previous research sub-questions, the main operational challenges, and issues, which prevent development and use of quality model, are listed, and associated at least to one practical solution.

In order to finalize a proposal to solve our third research question, a process should be elaborated, integrating the practical solutions to those operational challenges and issues, to ensure quality model operationalization.

Research Sub-question 3d

What is the practical answer to our needs on automotive embedded software case?

The last step of our research work is to exercise our findings against the real-use case coming from our industrial needs. The success of this application is key because it should assess the correctness and relevance of our research work.

Therefore, through this sub-question, we apply our practical solution to automotive embedded software development, taking into account any necessary automotive specific requirements (i.e., related to variants, complexity or constraints from standard, regulation, and stakeholders).

The result should be the development and use of the right operational quality model.

Research Question 4

Can we have a unique reference quality model for software product?

The successive responses to these first three research questions, including their respective research sub-questions, allow us to answer to our original needs to have a unified, operational, and appropriate way to define, assess, control, or predict quality of embedded software.

However, with this final research question, we would like to go one step further. Indeed, Zouheyr Tamrabet *et al.* [6] concluded their study by willing to propose a generic quality model “*that encompasses relevant quality attributes in order to define the quality of embedded software*”.

So, we expect to assess whether a unique reference quality model for software product is more appropriate than a meta-model as quality model aggregator for software product., and afterward define how to build and initiate the construction of either this unique reference quality model or this meta-model.

Research Sub-question 4a

Is it possible to have a unique reference quality model for software product, or instead should we have a meta-model?

Based on our research work, our findings, and a relevant analysis, we target to find and demonstrate the proper answer to that research sub-question. Then, depending on the answer result, we select the correct research sub-question set.

Case 1: Answer is a “unique reference quality model” for software product is possible

Research Sub-question 4b

What is the construction algorithm for such unique reference quality model?

To elaborate and build such quality model, we design the unique reference quality model construction algorithm.

Research Sub-question 4c

What is the first result of the unique reference quality model construction?

We initiate the build of the unique reference quality model. Hence, we execute the construction algorithm, provide the intermediary construction step results to facilitate its comprehension, and deliver the first quality model result.

Case 2: Answer is a “unique reference quality model” for software product is not possible and therefore a “meta-model” is the preferable solution

Research Sub-question 4b

What is the construction algorithm for such meta-model?

To elaborate and build such meta-model, we design the meta-model construction algorithm.

Research Sub-question 4c

What is the first result of the meta-model construction?

We initiate the build of the meta-model. Hence, we execute the construction algorithm, provide the intermediary construction step results to facilitate its comprehension, and deliver the first meta-model result.

5. Threats to validity and discussions

The purpose of this last section is to discuss, as needed, about what has been covered in the previous parts and, above all, review threats to validity of this chapter content.

Regarding our analysis and synthesis of our complex industrial research context with a need of quality, we chose to provide a high-level overview. In fact, a deeper analysis and description bring certainly much more details, but the risk is to lose from our mind the most important aspects that characterize our research context and therefore diverge from our original intention. However, at a later stage of this dissertation, we are going to push further some analysis and then detail more precisely, for instance, automotive embedded systems.

On the methodology for the appraisal of the state of the art on quality modeling for embedded software, we decided to perform an exploratory literature mapping instead of a systematic literature review, described in Kitchenham and Charter’s guidelines [31]. The reason is directly linked to their definitions. Indeed, on the one hand, the exploratory literature mapping aims to support the elaboration of unanswered research questions by collecting theory, method, or empirical evidence on a specific topic. On the other hand, a systematic literature review follows a clearly defined protocol (e.g., acceptance or rejection criteria to consider a literature contribution) and relies on a strict analysis and synthesis of the identified research contributions, to answer to a properly defined research question. By consequence, since we were looking for the current state of the art to set and refined our research questions, the exploratory literature mapping appears to be the obvious methodology to apply in this chapter.

Furthermore, with the literature review, we enlarged our research scope. First of all, we extended our scope from automotive embedded software to any embedded software, with the goal to avoid narrowing too much the literature findings and allowing us to learn from other embedded software cases. Then we encompassed embedded system to embedded software research scope because there are both closely related: systems requirements are derived to hardware requirements and to embedded software requirements. Thus, the inclusion of quality modeling for embedded system in our literature review is relevant. In addition, we decided to accept reliability growth model as a valid result. This particular aspect of quality modeling is complementary. Indeed, it covers quality prediction aspect, is often based on statistical or implicit model using historical data -as opposed to hierarchical quality models (e.g., ISO /IEC 25010 quality models)-, relies on various sources of metrics even other than software metrics (e.g., process metrics), and it depicts objective quality evaluation (e.g., comparison of the quality model result over successive software release).

At last, when we elaborated our research questions, we took some assumptions. Qualimetry is our privileged approach for quality modeling, mainly because it is the science of quality quantification, without being restricted to a specific domain, and consequently covers theory and applied aspect of quality quantification. We avoid limiting quality characteristics, or requirements, to non-functional quality, despite the existing overlaps between “quality requirements” and “non-functional requirements” that we noticed in the literature survey related to embedded software. Thus, we decided to not use quality attributes found in our survey to build a new quality model, but rather to construct on top of existing research contributions and quality models, fostering the reuse of many valuable achievements.

Chapter III. Research Methodology

1. Introduction

Over Chapter II, we explored our industrial context and the problematic related to quality modeling within the context of automotive embedded software development. We also performed an exploratory literature review about this problematic, analyzing many valuable contributions but also highlighting different gaps. That analysis allowed us to define a series of four main research questions that we refined whenever it was necessary.

The purpose of this chapter is to pursue our study by explaining the research methodology we applied to address these four research questions. It is therefore a pivot chapter since it structures the rest of the dissertation. Indeed, our research work didn't follow a straight line, and we had to change the direction of our research approach at a certain moment.

So, this chapter give the overall picture of our research flow, explaining the different steps, the technological locks, the decisions we made and gives hints about our contributions. For example, when we looked for quality models, we noticed that most of the contributions reused and changed slightly what was previously done rather than creating a complete disruption in the research. Thus, we decided to reuse and adapt quality model rather than creating a new one but with some disruption created from biology knowledge and analogy. Figure 7 depicts the summary of our research flow.

2. Initial research methodology: qualimetry, classification and decision

During our initial analysis of the problematic, we identified three research questions (cf. Table 3). We thought about the fourth one (see Chapter II.4) only during our second analysis, at a later stage of the research.

TABLE 3 - OUR FIRST THREE RESEARCH QUESTIONS

Research Question 1	Is Qualimetry, as the science of quality quantification, the right approach and what are quality and Qualimetry essentials?
Research Question 2	Considering the set of software quality models, how to identify and decide which quality model is the most suitable for embedded software?
Research Question 3	Considering a quality model for a software product, how to operationalize it?

Thus, starting with the first research question (i.e., *“Is Qualimetry, as the science of quality quantification, the right approach and what are quality and Qualimetry essentials?”*) as our primary step, we seek to understand and learn accurately the complex concepts of quality, quality modeling, and qualimetry. We also seek to assess that qualimetry is the right path to solve our problematic, considering that it fosters both quality quantification theory and practice, for example. And since qualimetry is often misunderstood, we build a synthetic vision depicted by the *“House of Qualimetry”* (cf. Chapter IV.6.a).

Moreover, during our investigation on quality modeling, we notice a recurrent fact. Effectively, quality models are often more or less similar, and the variants for objects of interest (from a quality modeling point of view) is handled like the customization with Horgans' essential views of quality characteristics [84], the commonality *versus* variation in quality characteristics of Choi et al. [39], or variation in metrics with FURPS [85], to cite few.

So, the global observation is that there is no generalized method to manage those variants on quality modeling. Nevertheless, in object-oriented programming and genetic there is a simple way to manage variety *via* polymorphism which represents multiple variations or evolutions of a same object, or gene.

We consequently introduce the polymorphism concept applied to quality modeling (see Chapter IV.6.b) to enable built-in adaptation and evolution of quality modeling objects (e.g., quality model, quality characteristic, measurement).

We pursue our quest of knowledge then on quality model for software and rapidly, our findings tend to demonstrate that many quality models exist. For example in Oriol *et al.* study [12], the authors indexed up to 51 distinct quality models, building a genealogical tree of quality models highlighting parent links between them. Unfortunately, we fail to find an exhaustive reference list of software quality models.

Then, we naturally move to the second research question (i.e., “*Considering the set of software quality models, how to identify and decide which quality model is the most suitable for embedded software?*”) that we split into two complementary fields (see Figure 6): theoretical field and practical field.

Our focus in the theoretical field encompasses the research from a theory perspective while in the practical field, the research embraces a practical perspective.

Theoretical field

Thus, in theory field, we start to identify the most appropriate methodology, categories, and criteria to classify or index quality models. We find few studies about classification with some categories (e.g., Thapar et al. [11] classification criteria relied on operational issue number ; Kläs et al. [10] defined a classification scheme based on quality model characteristics) but none of them are unified. So, recalling the “*genealogical tree of quality models*” shown by Oriol *et al.*, we push that reasoning further and consequently arrive at the conclusion that we should use cladistic to classify the variety of quality models (see Chapter V.4.a). Cladistic is usually used to classify species, but it can also be used to organize and simply classify a large list of quality models, highlighting some specific connections between those models.

Next to the classification is the decision methodology. Indeed, the purpose of this second stage of the theoretical field is to identify criteria, characteristics and / or formula to compare together quality models and finally decide on which quality model(s) fits best the requested needs.

About the comparison, a distance formula is a relevant approach to measure how closed quality models are together. A quality model is generally composed of different level of quality characteristics. So, we took the assumption that the distance formula in our case should focus on these characteristics, and since they are expressed in natural language, we may rely on string metrics.

However, we prefer the Nei and Li formula from genetic [86] (see Chapter IV.6.c) rather than the frequently used string metrics such as Hamming’s distance [87] (i.e., general sequences of symbols difference distance), Levenshtein’s distance [88] (i.e., string edition distance ; Hamming’s distance generalization), Damerau–Levenshtein’s distance [89] (i.e., string transformation distance ; extension of Levenshtein’s distance and distance used also in biology as measure of variations on DNA sequence), Jaro’s distance [90] (i.e., string similarity distance), Jaro-Winkler’s distance [91] (i.e., string similarity distance ; extension of Jaro’s distance), Gordieiev *et al.* ‘s cumulative characteristics matching based distance [92], [93] (i.e., string semantic distance), Natural Language Processing (NLP) based distance [94] (i.e., string lexical distance), or even other similarity distances like Jaccard’s distance [95] (i.e., general sample similarity distance based on sample attributes). The main reason of this choice is that compared to the other formulas, the genetic approach introduces a statistical notion that we can use to represent the statistical presence of quality characteristics.

Practical field

Parallely to the theoretical field, the practical field has also two stages synchronized with their corresponding theoretical stage. The first one is related to the application of the classification methodology on a case study. Our original case study is the embedded software but as we noticed above (cf. Chapter II.3), the set of embedded software quality model is quite small (i.e., 17 models) and therefore, we decide to extend the scope to software quality models. So, we perform a systematic literature review (see Chapter V.2) on software quality models, and during this study, we remark that we get a collection of more than 450 quality models. Many of them are justified variations of some previous quality models. The second stage takes as input the output of this first stage (i.e., the classified set of software quality models) and applies the decision methodology in order to identify the most appropriate quality models to apply to our company use cases.

Finally, the result of the second stage of practical field must be deployed and operationalized at our company level in order to answer to third research question (i.e., “*Considering a quality model for a software product, how to operationalize it?*”) and also to our problematic. The complete breakdown of this research flow is summarized over the Figure 6.

Nevertheless, the decision task is tricky because quality model differences are sometimes subtle, and there exists many criteria that we could use with no guaranty to make a right choice, especially from a qualitative point of view. And in case of decision mistake, the risk impact may be reduced, if those decision criteria are properly

chosen. However, decision action means that we discard many valuable contributions that could be very beneficial for the solution at the end.

Thus, if we want to minimize proactively the impact of a wrong, not totally accurate decision, or even of a partial solution, we must bring some nuances in the decision result and, integrate some of the potentially discardable contributions to our quality model solution prior to its deployment.

In short, we must reconsider our original strategy about research approach, and update accordingly our heuristic.

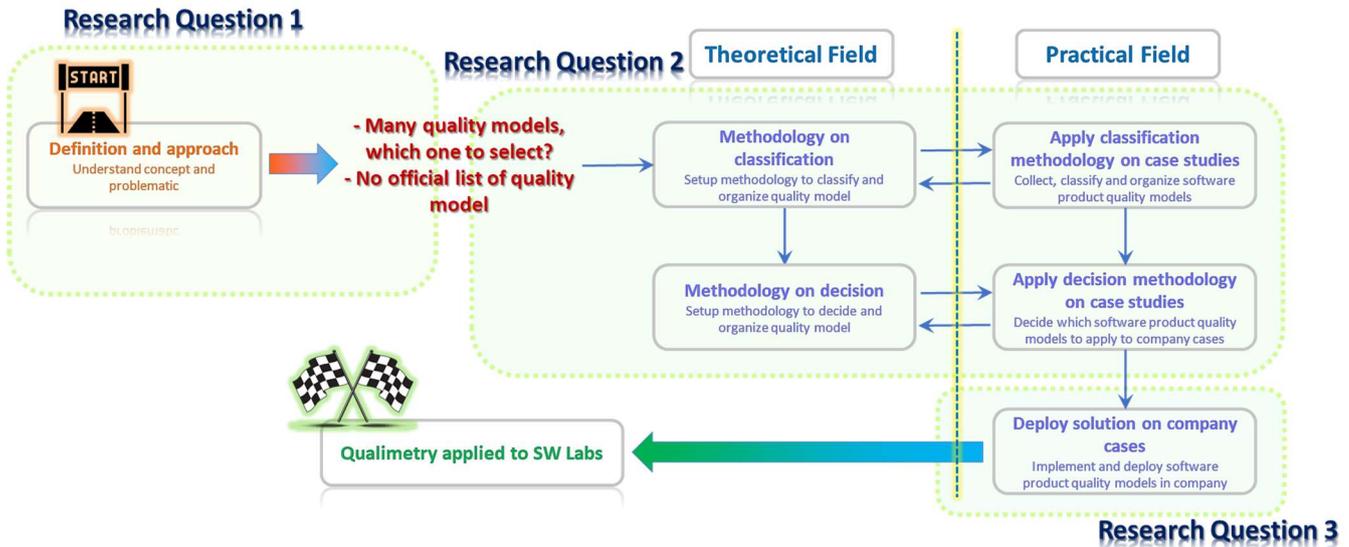


Figure 6 - Overview of our early research flow breakdown

3. Realigning our methodology: from theory to practice

So, instead of directly deploying the selected quality model to the company, we realign our research strategy to address operationalization aspect of quality model and proactive integration of some quality model valuable contributions, that a decision result could discard but should not afford to ignore.

To do so, we investigate how to go from the theory to the practice in quality modeling, reviewing the key theory concepts and the practical contributions in that domain. Thus, to answer to our third research questions (i.e., "Considering a quality model for a software product, how to operationalize it?"), we must work on these two aspects.

First, we identify the potential issues that may prevents quality model operations (i.e., development and use). We also find practical solution(s) to these potential issues, identifying the most important contributions in quality modeling, and assess their respective similarities or alignment.

Second, we integrate these practical solutions through a consolidated process for the development and use of quality models.

So, with further details, we begin by looking for papers dealing with quality model operationalization, or issues preventing their use or development. For instance, Thapar *et al.* [11] identified 9 issues that prevent development and use of quality models, Abran *et al.* [96] highlighted three main harmonization issues with ISO-based quality models, Kläs *et al.* [97] investigated software quality model adaptation, and Wagner *et al.* [98] were proceeding on a survey of quality models in practice among four software companies in Germany.

Based on the review and consolidation of the findings from these retrieved papers, we build a first list of issues.

Next, we proceed on a further analysis and raise our list to a total of 16 issues that prevent development and use of quality models (see Chapter VI.2).

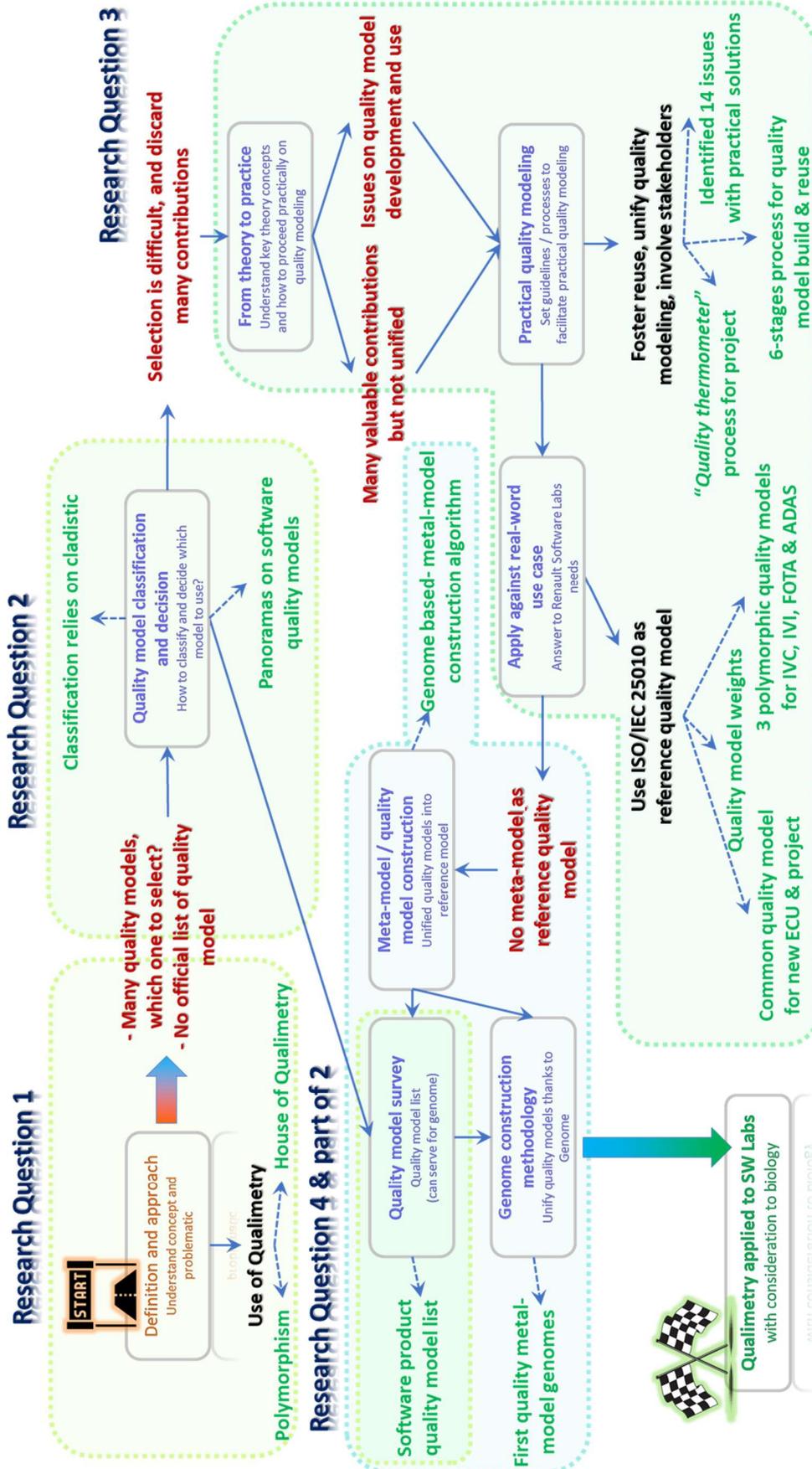


Figure 7 - Overview of our research flow breakdown: in red the technological locks, in green our contributions

To continue, we address one by one these issues to find some practical solutions over the 50 years of research contributions in quality modeling (see Chapter VI.3). Once each issue has been associated to at least one solution, we design two processes to take implicitly benefit of these key contributions and to foster their usage.

The first process (see Chapter VI.4.a) focuses on the development or adaptation of quality models, emphasizing their reuse. Indeed, even if we select a quality model for a subjective reason (e.g., use of standard quality model for compliance to standard), this process allows to integrate contributions from other quality models, and to be aligned with key stakeholders and any constraints. The second process (see Chapter VI.4.b) is dedicated to quality model use and call the first process in its early stage.

Finally, to close the loop from the theory to the practice, we must successfully apply our findings against a real use-case.

This real use case is naturally linked to the embedded software from our automotive industrial context: three ECU embedded software and one transversal embedded software. In addition, the quality model selection criteria have to be aligned also with our industrial context, and therefore to the guidelines from A-SPICE standard which focuses on automotive development processes.

Thus, we select the two recommended quality models of ISO / IEC / IEEE 25010 (i.e., system and software product quality model, and quality-in-use model), and apply our development process for quality model to them. The result (see Chapter VII.5) is three polymorphic quality models with their characteristics and sub-characteristics weights, and their “*mother*”, or common, quality model that could be used by default for a new ECU or architecture, for example. Regarding their associated metrics, they are linked to a scorecard, where each polymorphic quality model is used to compute quality product indicator.

So, we demonstrate that our new approach allows us to start from a subjective selection, arrives to an objective and operational solution, and consequently, assesses the correctness of our answer to our company problematic.

4. Last step: construction of a meta-model

As we highlighted previously, at this stage of our research we have answered to our original problematic. However, during our studies we noticed that there existed many quality models but no unique reference quality model. Moreover, we subjectively selected two quality models from ISO standard as our reference models, like the 71% of the 28% of the companies that Wagner *et al.* studied [83] and which use and customize quality models from standard. So, to go one step further, we aim to address that last research question “*Can we have a unique reference quality model for software product?*”.

Zouheyr Tamrabet *et al.* [6] finished their study with the opening to propose a generic quality model “*that encompasses relevant quality attributes in order to define the quality of embedded software*”. Unfortunately, this quest is hardly reachable, and even if we succeed on getting such unique and generic quality model, that solution may become either less satisfactory, because of the constant evolution of our real-world resulting in constant evolution of most of systems and software (e.g., Lehman’s laws [82], [99], [100] are depicting that evolution aspect), or unsuitable, because of the wide diversity of both systems and software we have nowadays. So, the preferable approach for this research question is to build a meta-model as quality model aggregator.

This meta-model can serve as an advice quality model where the main quality perspectives (e.g., Garvin quality perspectives [101]) are present and where we have the notion of likelihood to have a quality characteristic or sub-characteristic as well as consideration to their nuances which are present in existing quality models (e.g., resource utilization, resource behavior and utilization of resource ; instablability and deployability). That description is comparable to what we can find in genetic, more particularly with the genome.

Therefore, strong from this analogy, we decide to construct a meta-model which is the genome of software quality model. After defining the meta-model ontology, we initiate its construction based on eight quality models. To select this set of quality models, we take the quality models from Thapar *et al.* study [11] with the least operational issues that we complete with quality models from standard or widely used. We also pay attention to the fact that those selected models must have definitions for all their quality characteristics and sub-characteristics. At last, we limit the selection results to eight quality models since the purpose is to initiate the meta-model construction. The selected quality models are ISO/IEC 25010 [23], ISO/IEC 9126 [24], Boehm [42],

McCall [41], Alvaro [36], FURPS [85], Kalaimagal Q'Facto 12 [102] and Bawane [103]. The genome builds finally result in six distinct quality perspectives, with one of them (i.e., supportability) that can be interpreted as a polymorphic variation of one of the other five quality perspectives (i.e., product revision).

Those quality perspectives are:

- General Utility,
- Product in Use,
- Product Operation,
- Product Revision,
- Product Transition,
- Supportability,
- Software Product.

5. Threats to validity and discussions

Like in previous chapter, this section goal is to review the chapter content, and particularly the threats to validity.

The main treat is with regards to our decision to change the research approach. Undoubtedly, we could proceed further on the creation of an oracle to predict the best quality model for quality definition and assessment, continuing our investigation of a sharpener list of quality model selection criteria, and then use experimentation to validate the oracle prediction results. However, proceeding on such selection is only one side of the problematic and certainly results in discarding many valuable and complementary contributions in quality modeling. Indeed, during our research investigation on quality model, we noticed that there were many valuable contributions that it would be interesting to take advantage of. And moreover, we noticed that many published quality models are not operationalized yet. Thus, operational criteria should be included also as decision criteria. Nevertheless, such inclusion may consequently reduce even more the pool of candidate quality models to only few, but unfortunately with no guaranty that, from a qualitative point of view, the decision result is satisfying. So, the real problem is not to find the best quality model but rather to make a subjective selected quality model (i.e., selected based for some reasons such as criteria matching, use of standards, company forcing a solution, to cite few) to become the right and optimum objective quality model for our needs. This is the main motivation of our research methodology realignment.

Regarding our source of inspiration to elaborate solutions, we can obviously find many clues of research direction in software engineering since its research domain is very active. However, sometimes we have to think out of the box to tempt to bring some disruption. Like Oriol *et al.* [12] who relied on genealogical tree to represent kinship line of quality models for web-service, our choice is to refer to biology, making analogy between our research cases, technical locks and what we can find in biology.

Therefore, we borrow some solution from biology like cladistic to classify quality models, but also from a sub-branch of biology: the genetic. We start with polymorphism, concept present also in object-oriented programming, and the degree of polymorphism, or variety, formula that includes likelihood on the elements considered for the measurement, as opposed to regular software engineering distance formulas where that notion is missing. Then, the comparison of a quality characteristic with a specific DNA sequence (i.e., a gene), and thus a quality model with a chromatid (i.e., a chromosome is composed of two identical chromatids), allowed us to confirm the use of the degree of polymorphism in our case, and to construct a meta-model: the *Software Quality Model Genome*.

All those research results affirm our choice of considering biology as one of our research drivers.

In conclusion, the doubt we had with regards to our initial research methodology, and which caused the methodology realignment, together with our inspiration from biology were finally a salvation to solve successfully our problematic while bringing some disruption.

Chapter IV. Quality, Quality Modeling and Qualimetry

1. Introduction

The purpose of this chapter is to answer to the first research question, and therefore explore and understand accurately the complex concepts of quality, quality modeling and qualimetry.

Research Question 1

Is Qualimetry, as the science of quality quantification, the right approach and what are quality and Qualimetry essentials?

Indeed, quality definition evolved since the ancient philosophers on the 5th century B.C., and today a standardized definition is hopefully available. Nevertheless, there are still some further subtilities to capture with quality like, for example, the notions of quality perception with Kano's model [58], Garvin's quality perspectives [101] and quality dimensions [104], Horgan *et al.*'s essential views [84], lagging or leading indicators [105], [106].

Regarding quality modeling concept resulting in quality model, it is pivotal for quality definition and evaluation activities. We remark that, whatever the type of quality models is, those models can be perceived as composed of dependent (i.e., computed from observation) and independent (i.e., observed) variables, themselves consequently associated to either indicators, quality characteristics, sub-characteristics or measures.

So, we consolidate its knowledge thanks to a review of the key contributions in quality modeling. About the last concept, we have to get to know qualimetry which is in fact the science of quality quantification, and more particularly what the essential characteristics of that science are.

Finally, with this tripartite knowledge (i.e., quality, quality modeling and qualimetry), we are able to affirm which quality modeling or quantification solution is the right one for our problematic case, parallely contributing to the resolution of the technical locks we encountered.

2. The essence of quality

Research Sub-question 1a

What is the essence of quality?

Since the 5th century B.C. with the ancient Greek philosophers that are Socrates, Aristotle, Protagoras, Heraclitus and Plato in the quest of "*what is knowledge?*" [107], and summarized by Aristotle through "*By 'quality' I mean that in virtue of which people are said to be such and such [...] The body is called white because it contains whiteness.*" [108], the definition of quality evolved and nowadays, it is hopefully possible to find converged and standardized definitions. This is, for instance, the case with the international standard for quality management, ISO 9000 [109] and the ISO/IEC/IEEE 24765 International Standard Systems and software engineering—Vocabulary [110] where quality definitions are similar despite their distinct scope and a more elaborated definition for ISO/IEC/IEEE 24765:

- **ISO 9000 quality definition:** "*The quality of an organization's products and services is determined by the ability to satisfy customers and the intended and unintended impact on relevant interested parties*".
- **ISO/IEC/IEEE 24765 quality definition:** "*1. the degree to which a system, component, or process meets specified requirements. (IEEE Standard for Software and System Test Documentation.3.1.25). 2. ability of a product, service, system, component, or process to meet customer or user needs, expectations, or requirements. 3. the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs. (ISO/IEC 9126-1:2001, Software engineering — Product quality — Part 1: Quality model.B.21). 4. conformity to user expectations, conformity to user requirements, customer satisfaction, reliability, and level of defects present. (ISO/IEC 20926:2003, Software engineering — IFPUG 4.1 Unadjusted functional size measurement method — Counting practices manual) 5. the degree to which a set of inherent characteristics fulfils requirements. (A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fourth Edition). 6. the degree to which a system, component, or process meets customer or user needs or expectations. (IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation.3.1.25)*".

We remark that the International Software Testing Qualification Board glossary (ISTQB Glossary 3.1, [111]) refers to the IEEE Standard glossary definition [112] which is as well similar to part 1 and 2 definition of ISO/IEC/IEEE

24765. Additionally, to this quality definition, there are some further nuances and concepts around quality to understand in order to capture the very essence of quality. We cover them in the next sub-sections.

a. Integral quality

The first further nuance about quality is the concept of “integral quality”. This concept mainly comes from the socioeconomic field [113], [114]. There are debates about integral quality [114], nevertheless, this concept is important as its joints quality with cost. This last one may vary for same type of object of interest, depending on project model, for example. Thus, the integral quality corresponds to the sum of quality with cost effectiveness (cf. equation (1)). The cost effectiveness is described by the object properties that are linked to the input capital for production and consumption of that object. This nuance with quality is important because selecting one or the other concept can lead to different results in quality related activities.

$$\text{Integral Quality} = \text{Quality} + \text{Cost Effectiveness} \quad (1)$$

b. Perceived quality

Recalling the knowledge quest of the ancient Greek philosophers, particularly the summary from Aristotle (i.e., “*The body is called white because it contains whiteness*”), and the second point of IEEE Standard glossary definition of quality (i.e., “*Customer or user needs or expectations*”), we discern that quality is also a matter of perception or interpretation. On 1984, Kano *et al.* [58] investigated how customers perceive quality, and remarked that customer satisfaction and dissatisfaction are dissymmetrical. They noticed, for instance, that if the existence of a feature can create a customer satisfaction, its absence doesn’t automatically create a dissatisfaction, and *vice-versa*. Kano *et al.* summarized their findings over the Kano’s model depicted in Figure 8. The two orthogonal axes of this model are the level of customer expectation, or requirements implementation, versus the level of customer satisfaction or dissatisfaction. The main elements of the Kano’s model are:

- **Must-be / basic:** these are the basic mandatory customer expectations or requirements. Their absence⁵ creates dissatisfaction while their presence won’t create any satisfaction. Moreover, it is not required or expected that they are asked by customer (e.g., this is the case of product legacy features).
- **Attractive / exciting:** these are not expected by customer. Their presence creates satisfaction while their absence won’t create any dissatisfaction (e.g., this is the case with product innovation or disruptive features).
- **One-dimensional / required:** These are the new customer expectation or requirements. Their presence creates satisfaction while their absence creates dissatisfaction. Satisfaction and dissatisfaction are proportional to the performance of implemented requirements (e.g., the new planned product features).
- **Indifference area:** In this area neither absence, nor presence of customer expectations or requirements don’t significantly influence customer satisfaction or dissatisfaction.

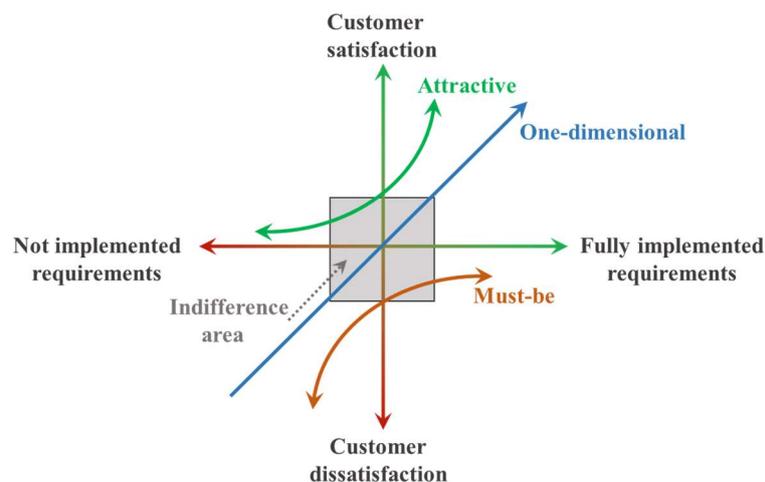


Figure 8 - Kano's model of perceived quality [58]

⁵ When we indicate presence, requirement must be implemented and working as expected, with absence it is when feature is missing or doesn't work as expected.

Moreover, the quality perception evolves with time because of the evolution of customer expectations: an “attractive” product feature may become a “required” product feature after a certain number of product releases, and even later, may become a “basic” feature.

As example to illustrate this evolution, we consider the creation of the first mobile-phone and then a new generation of it. When creating the first mobile phone, the basic feature was “to emit and receive phone calls”, the required feature was “to be mobile” and the attractive feature was “to have call log”. In the next generation, moving one step to our today’ smartphone, the basic features became “to emit and receive phone calls” and “to be mobile”, the required features were “to have call log” and “to send and receive SMS/MMS”, and finally the attractive feature was “to take photos to attach to MMS”. We note that this behavior is what Lehman characterized previously with his laws of software evolution [82]. We have the same behavior with quality, and therefore, when dealing with quality, we must take into account these perception impacts and time evolution like in von Dran *et al.* [115] or Liang *et al.* [116]. However, through his study [117], Han highlighted that these considerations on perceived quality are not obvious in quality evaluation activities because of the existence of a gap between perceived and measured data quality.

c. Quality perspectives and views

Through the description of the perceived quality, we could notice that quality is also a matter of point of view. Indeed, in 1999, Horgan *et al.* [84] introduced the notions of quality essential views through two concepts: **Key Quality factors** (KDF) and **Locally Defined Factors** (LDF). According to Horgan, quality, characterized by a set of quality characteristics or factors (e.g., usability, correctness, reliability, maintainability), can be viewed from two viewpoints. The KDF encompass the quality factors which are relatively invariant (i.e., they are key) over project, product and/or stakeholders, while the LDF include the quality factors that change (i.e., they are locally defined), depending on the project, product, or stakeholders.

In 1984 Garvin brought some further subtleties in the discernment of what quality is, by splitting quality aspect into five quality perspectives or views [101]:

- **Transcendental view:** quality as an ideal that cannot be defined but can be recognized,
- **User view:** quality as a user's expected needs,
- **Manufacturer view:** quality as conformity to specifications, regulations, standards, involving optimization during production and operation,
- **Product view:** quality as the internal characteristics of a product that influence its external ones,
- **Value-based view:** quality as how much a customer is willing to pay for the quality outcomes.

These five perspectives of quality are distinct and can be used alone or combined. As an example, a company which wants to optimize the maintenance cost should focus on the manufacturer quality perspective and may use a quality model accordingly to this perspective to predict its product quality during the development stages.

d. Quality dimensions and characteristics

Then in 1987, deepening his previous analysis, Garvin proposed eight critical dimensions of product quality [104]. These are performance, features, reliability, conformance, durability, serviceability, aesthetics, and perceived quality. These quality dimensions must be aligned with the targeted quality perspective(s) to be used accurately. With this proposal, he illustrated the elicitation of quality requirements [24] from stakeholders.

Moreover, we noticed that Garvin indifferently used quality dimension or quality category to express the same concept. Over our literature studies, we also noticed the same behavior of using different terms for describing the same concept. The consequence is a mix of vocabulary, sometimes slightly antinomic between contributions. For example in Boehm *et al.* [42], ISO/IEC 9126 [24] or ISO/IEC 25010 [23], the authors deal with multiple levels of quality characteristics and metrics, on their side, McCall *et al.* [41] consider perspectives, quality factors, quality criteria, and metrics, Dromey [49] use quality attributes.

For all these sample cases, the targeted contribution is the creation of a quality model.

So, in order to avoid any confusion with regards to the terminology, here are the definition of the main keywords:

- **Attribute:** “Inherent property or characteristic of an entity that can be distinguished quantitatively or qualitatively by human or automated means” (from ISO/IEC 25000 [25]),
- **Quality attribute:** “1. characteristic of software, or a generic term applying to quality factors, quality subfactors, or metric values. (IEEE Std 1061-1998 (R2004) IEEE Standard for Software Quality Metrics Methodology.2.17). 2. feature or characteristic that affects an item's quality. 3. requirement that specifies the degree of an attribute that affects the quality that the system or software must possess” (from ISO/IEC/IEEE 24765 [110]),
- **Quality factor:**” 1. a management-oriented attribute of software that contributes to its quality. (IEEE Std 1061-1998 (R2004) IEEE Standard for Software Quality Metrics Methodology.2.18). 2. higher-level quality attribute” (from ISO/IEC/IEEE 24765 [110])
- **Quality characteristic:** “inherent distinguishing feature of an object related to a requirement” (from ISO/IEC 9000 [107]),
- **Software quality characteristic:** “category of software quality attributes that bears on software quality; NOTE: Software quality characteristics can be refined into multiple levels of sub-characteristics and finally into software quality attributes” (from ISO/IEC 25000 [25]),
- **Quality criteria (or quality standards):** “the parameters established or adopted by an organization to
- measure the compliance of its products, services and processes to a certain defined standard” (from IAEA glossary terms [118]),
- **Quality model:** “defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality” (from ISO/IEC 25000 [25]),
- **Quality objective:** “result to be achieved related to quality” (from ISO/IEC 9000 [107]),
- **Quality requirement:** “need or expectation related to quality, that is stated, generally implied or obligatory” (from ISO/IEC 9000 [107]),
- **Indicator:** “measure that provides an estimate or evaluation of specified attributes derived from a model with respect to defined information needs” (from ISO/IEC/IEEE 15939 [115])
- **Metric:** “measurement scale and the method used for measurement” (from ISO/IEC 14598-1 [115] and ISTQB Glossary [111]).
- **Quality measure:** “derived measure (i.e., measure that is defined as a function of two or more values of base measures (i.e., measure defined in terms of an attribute and the method for quantifying it), from ISO/IEC/IEEE 15939 [119]) that is defined as a measurement function of two or more values of quality measure elements” (from ISO/IEC 25021 [121])
- **Quality measure element:** “measure defined in term of a property and the measurement method for quantifying it, including optionally the transformation by a mathematical function” (from ISO/IEC 25021 [121])
- **Property to quantify:** “property of a target entity that is related to a quality measure element and which can be quantified by a measurement method” (from ISO/IEC 25021 [121])

Finally, we build an ontology depicted by Figure 9, taking profit of these definitions. That ontology summarizes in one drawing the links between all these concepts.

Thus, to define, evaluate or predict quality, first we must enumerate and define accurately the corresponding quality objectives. These objectives are then refined into quality requirements and quality criteria, or standards. They also serve to identify the proper quality perspectives or views we aim our quality to focus on. Next, the criteria guide the determination of the necessary indicators set. That set is used jointly with the quality perspectives to construct the right quality models.

Quality models are directly elaborated from the set of quality characteristics compliant, not only to the quality model properties (e.g., type, perspective, lagging or leading indicator) but mostly to the quality requirements. At last, these quality characteristics are decomposed into quality sub-characteristics, attributes, sub-attributes, and metrics, as needed.

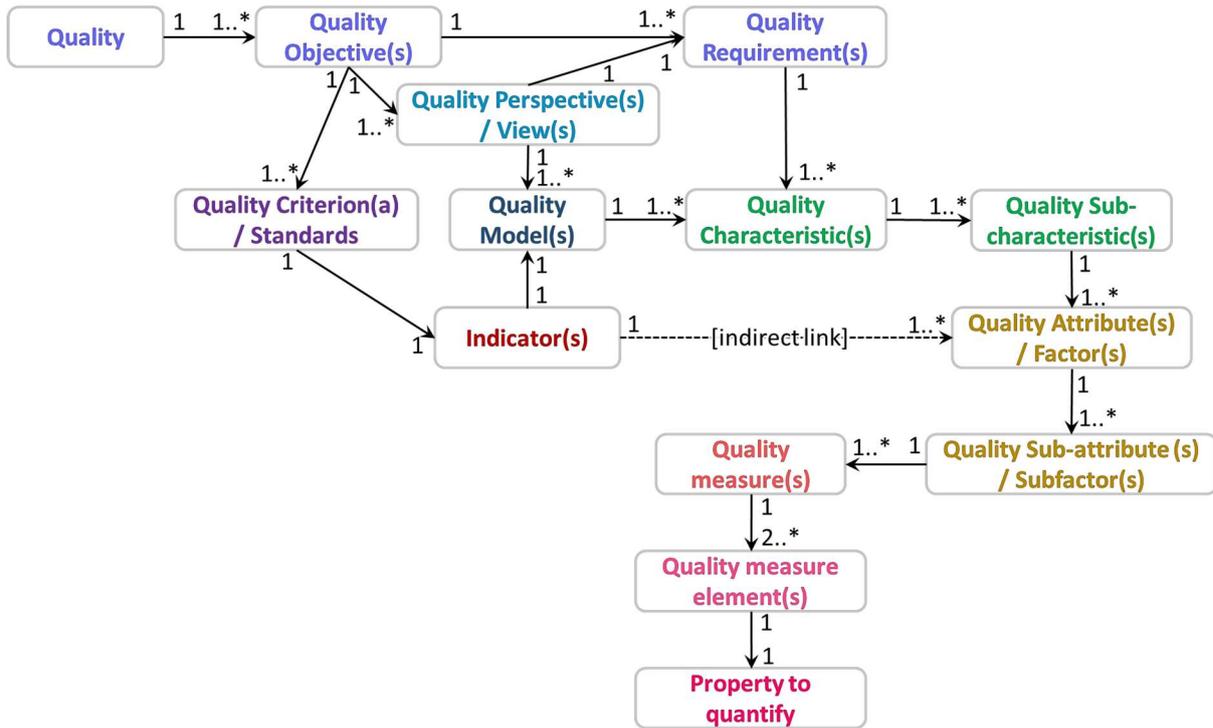


Figure 9 - Ontology of main quality keywords

e. Lagging and leading indicators

The last complementary aspect to consider about quality is whether we deal with lagging or leading indicators. The lagging indicators focus on the past performance, what already happened, and therefore measure the consequence (e.g., test results). They are easy to measure but hard to improve. The leading indicators focus on the current or future performance, prediction and then measure the causes. They are composed of characteristics, a condition which are both analyzed regularly or on-demand to predict a behavior with a certain level of confidence and a lag range (see Figure 10). They are hard to measure but relatively easy to improve (e.g., monitor test development skillset of a team). Nevertheless, that difficulty can be skirted thanks to the work lead by Roedler and Rhodes [105], completed by Roedler *et al.* [106], where a set of 18 leading indicators (e.g., requirements trends, interface trends, technical measurement trends, systems engineering staffing & skills trends, schedule and cost pressure) is fully described and explained.

Knowing whether quality indicators happen in lagging and/or leading space is crucial in activities depending on quality, including quality modeling and evaluation activities. This certainly influences the resulting quality model since it may be sub-optimal or inefficient.

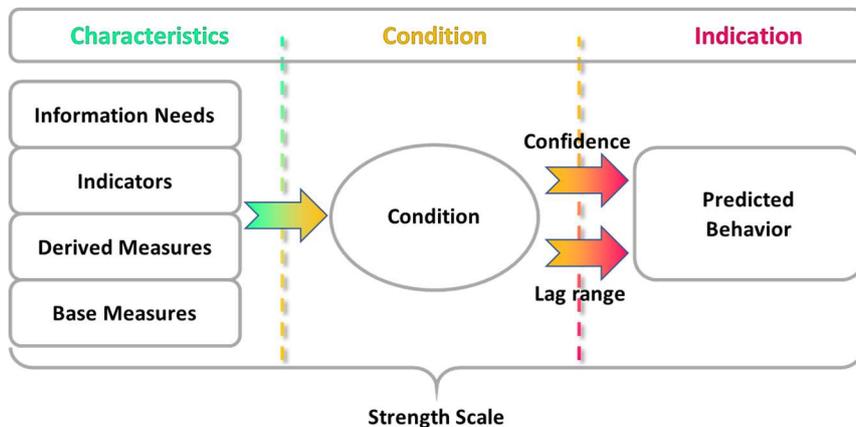


Figure 10 - Leading indicator composition, Roedler *et al.* [106]

All these further knowledges and nuances are keys to capture the essence of quality and for succeeding in the modeling of quality, perceived or not.

3. The essence of quality modeling

Research Sub-question 1b

What is the essence of quality modeling and particularly in software field?

The premises of quality modeling can be found through plethora of examples often associated to quality quantification. For instance, back to 1910's, the mechanic and shipbuilder A. N. Krylov defined quality measurement in a scientific manner to find the best warship design [113].

From the previous Chapter IV.2, we understand what quality is, and what are the elements to use in order to describe an object of interest (e.g., in our case embedded software product) qualitatively and quantitatively. Although quality is an elusive target, as rightly highlighted by Kitchenham and Pfleeger in their article [26] on 1996, a quality model is a powerful tool, such as a bow and arrow, for hitting this target.

A quality model is the delivered result of quality modeling activity which enables quality definition, evaluation, and prediction.

So, in the following paragraph, we review the distinct and fundamental elements characterizing the essence of quality modeling starting with quality model, and then measurement, scale, aggregation, and threshold.

a. Quality model

As we have seen above in Chapter IV.2.d, the current international ISO standard for software product quality requirements and evaluation (SQuaRE), ISO/IEC 25000 [25], defines a quality model as a *“defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality”*.

Based on this definition, we notice that a quality model is used for multiple purposes such as quality requirement specification or quality evaluation. In 2009, Deissenboeck [122] classified quality models into three main categories: definition, assessment and prediction. This is the concentric DAP (i.e., Definition, Assessment, Prediction) classification (see Figure 11) where the further we move from the center, the more advanced our model is.

At the center of the DAP classification, we found the first type of usage for a quality model. This is the set of definition models which corresponds to *“specifying quality requirements”* through a set of quality characteristics and sub-characteristics linked together. The goal is then to identify, refine and organize all quality characteristics required for the quality⁶ objectives, and then quality requirements, related to the focused object of the quality modeling activity. Figure 12 shows an example of such definition model ; this is the system / software product quality model from ISO/IEC/IEEE 25010 [23]. We note, for instance, that *“Functional suitability”* quality characteristic is refined into three quality sub-characteristics: functional completeness, functional correctness, and functional appropriateness.

The second type of usage is the assessment, or control. Control activity is close to assessment one, except that control is an assessment performed regularly. So, in order to assess quality, the definition model must be extended with measurement information. The current SQuaRE standard ISO/IEC 25020 [119] recommends to associate at least one software quality measure to each quality sub-characteristic. These software quality measures are then structured under a method to perform the measurement and the elements to measure with their properties, or attributes (cf. Chapter IV.2.d). For example, to measure the *“Maturity”* sub-characteristic of *“Reliability”* characteristic, we use the mean time between failure (i.e., MTBF) as the measurement method. The corresponding formula is defined as the ratio of the number of failures per duration period. Consequently, the measure elements are *“number of failures”* and *“duration period”*, and the associated properties to quantify are then *“failures”* and *“duration”*. This example is depicted in Figure 13. We remark that similarly than in

⁶ Perceived and not perceived quality.

mathematics, physics or statistics, the properties to quantify, or to observe, can be seen as independent variables – if the independency with the other independent variables is verified – and the measurement methods as dependent variables, outcomes of some combination of those independent variables.

This observation leads us to the last level in the DAP classification: the prediction models. As the model name indicates, the purpose of this type of model is to predict quality and its construction requires to have previously defined the quality (i.e., identify the quality characteristics, sub-characteristics and their relationship) and also the way to evaluate them. In general, this type of model cannot be used as a definition or assessment model because of this model construction often results in statistics or implicit model (e.g., regression analysis over historical data [124]). Equation (2) illustrates a prediction model for software fault from Khoshgoftaar and Szabo [125]. The authors elaborated a Poisson regression (i.e., named *preg* by the authors) model, based on the principal components derived from a set of observed software measures.

$$Fault_{preg} = e^{-0.1939 + 1.1248 \cdot PC_1 - 0.2277 \cdot PC_2} \quad (2)$$

$$\text{where } PC_1 = w_{\eta_1} \cdot \eta_1 + w_{\eta_2} \cdot \eta_2 + w_{N_1} \cdot N_1 + w_{N_2} \cdot N_2 + w_{LOC} \cdot LOC + w_{XQT} \cdot XQT$$

$$\text{and } PC_2 = w_{V_1(G)} \cdot V_1(G) + w_{V_2(G)} \cdot V_2(G)$$

PC_1 is the principal component 1 and corresponds to a weighted w sum of η_1 (i.e., number of unique operators), η_2 (i.e., number of unique operands), N_1 (i.e., total number of operators), N_2 (i.e., total number of operands), LOC (i.e., lines of code), and XQT (i.e., number of executable statements). PC_2 is the principal component 2 set as a weighted sum of $V_1(G)$ (i.e., McCabe’s cyclomatic number) and $V_2(G)$ (i.e., extended cyclomatic number, $V_1(G) +$ the number of logical operators (see paper [125] regarding weights and further details).

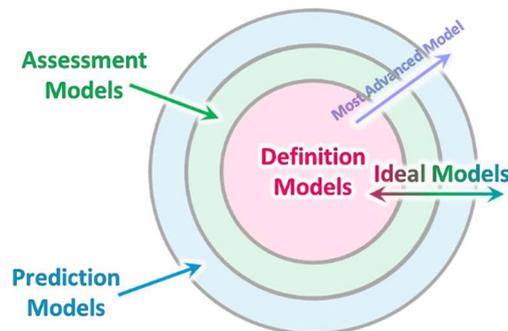


Figure 11 - The DAP classification introduced by Deissenboeck et al. 2009 [122]

Additionally, to these three types of model, there is one last model to consider in the DAP classification. This is the ideal model, or multi-purpose model. Its particularity is to cover the three levels of DAP. Based on Wagner experience [27], this type of model is rare and one example we can cite is Kitchenham’s COQUAMO [126]. COQUAMO is a constructive quality model and therefore, this model has the transversal capability to address definition, assessment, and prediction purpose. Nevertheless, the main difficulty with this quality model is around the unclear relationship with measurements. And as Wagner [27] pointed out, in 1997 Kitchenman et al. finally concluded that “there were no software product metrics that were, in general, likely to be good predictors of final product qualities” [127] which calls into question COQUAMO as prediction model.

In parallel to the quality model purpose, with DAP classification, there exist several types of model representation: hierarchical, statistic/implicit, and meta-model.

Figure 12 illustrates the hierarchical model representation. In fact, instead of hierarchical, or tree, graph, we should rather speak about oriented graph (e.g., Boehm’s quality model [42], [128] presents some cycles like “As-is Utility”, “Reliability”, “Robustness /integrity” and “Human engineering” and therefore this model is not by definition a tree or hierarchical graph ; cf. Figure 14) which is a more generic graph representation, even if in most of the cases, the quality model is an hierarchical graph (e.g., ISO/IEC 9126 [24] or ISO/IEC/IEEE 25010 [23]). Nevertheless, it is possible to remove the cycles from the graphs by splitting and repeating some graph elements, resulting finally in a hierarchical graph. For instance in Figure 14, we can break the highlighted cycle by splitting and repeating the lowest level elements “Robustness /integrity”, having one “Robustness /integrity” for “Reliability” and another for “Human engineering”. The main advantage of this type of representation is its

human comprehension, readability. Indeed, not only the information organized hierarchically relies on natural language, but also the hierarchical graph reflects the relationship between its elements in an obvious manner. However, quality evaluation requires further completion of the model, including measurement and aggregation method.

The next type of model representation is the statistics or implicit model. Those models are defined through mathematical model or function (e.g., Khoshgoftaar and Szabo’s model [125] shown in equation (2)). The principal interest of this type of model is the readiness to be used either for assessment or prediction purpose. This is due to the mathematical description which uses measurements as inputs. The main drawback is the complexity to understand or interpret the model, and the difficulty to retrieve the quality characteristics and sub-characteristics behind the model.

The last type is the meta-model. A metal-model describes the general concepts, depicting a view that shows the different these key concepts, their properties, their relationships (sometimes including description of relationship type) and their cardinality. This is similar to a UML class diagram or to an ontology (e.g., Figure 9). Thus, by conception a meta-model is a good candidate to become a multi-purpose quality model, but it requires to be applied against the object of interest in order to be used either for definition, assessment or prediction.

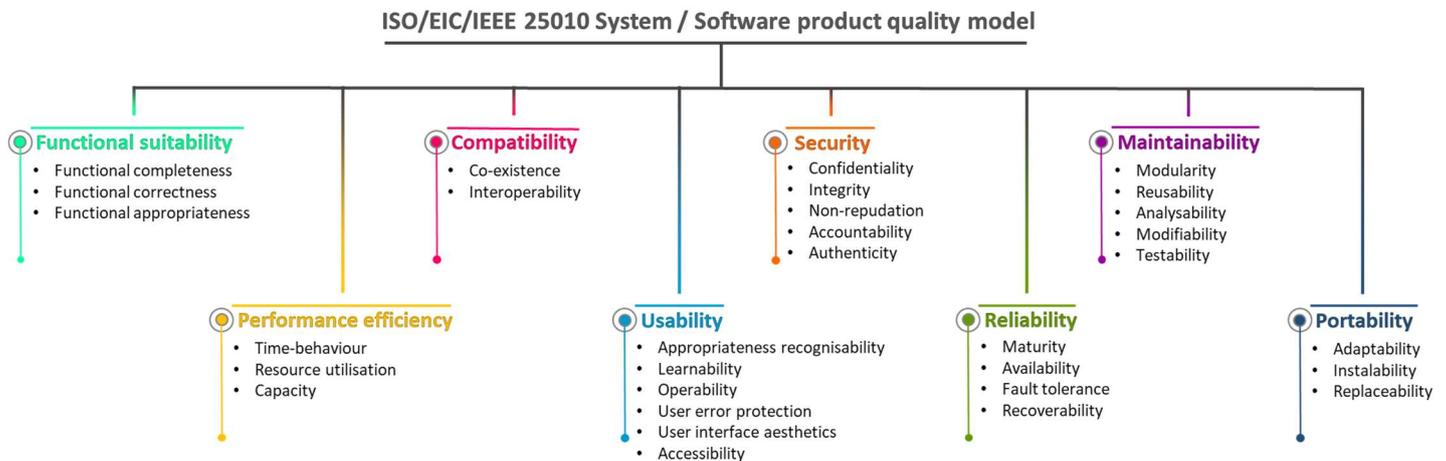


Figure 12 – Example of a definition model: the ISO/IEC/IEEE 25010 System / Software product quality model [23]

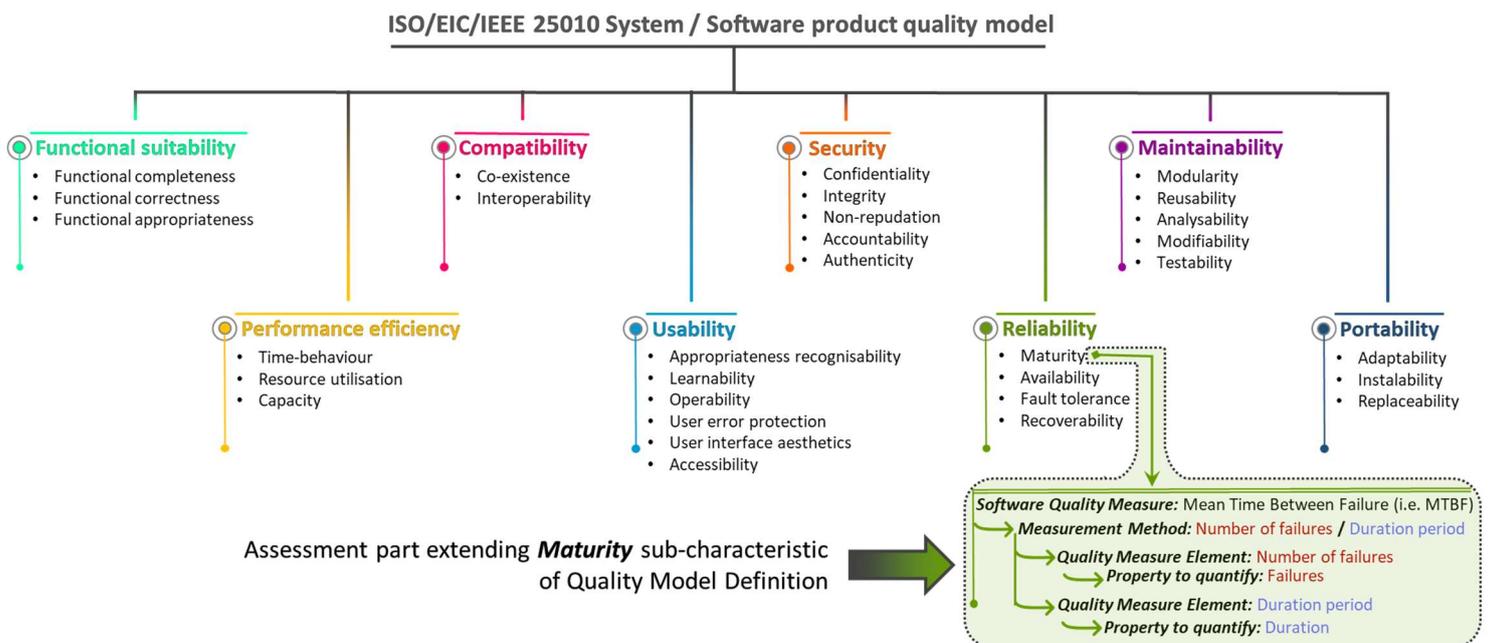


Figure 13 – Example of an assessment model: *Maturity* sub-characteristics measurement extension of ISO/IEC/IEEE 25010 System / Software product quality model [23]

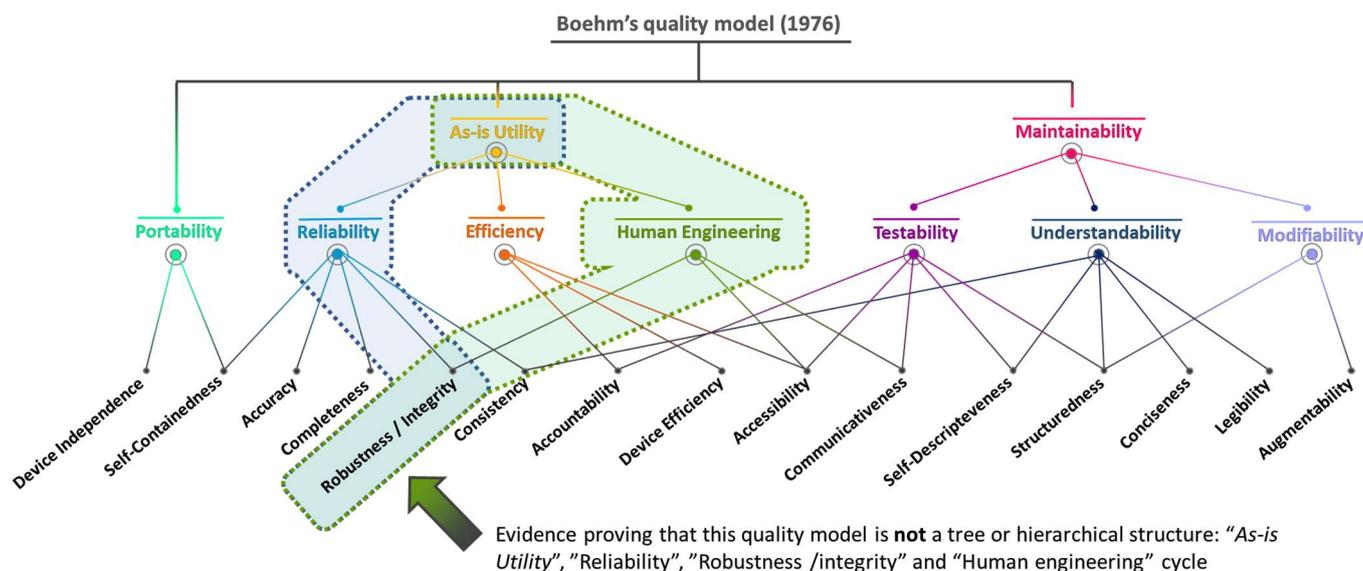


Figure 14 - Boehm's quality model (1976) [42] with an example of cycle in the graph structure

To summarize, aligned to its purpose and representation, a quality model corresponds to the abstraction of quality of interest via the identification of independent and dependent variables related to quality, and completed with their mathematical or statistical relationships.

The independent variables are the measurements of quality characteristics (e.g., the eight critical dimensions of Garvin [104]) while the dependent variables are the aggregated or combined results for some specific quality perspective (e.g., Garvin's five quality perspectives [101]).

The next fundamental element to proceed from a quality model definition to quantification, assessment, and quality control, deals with those independent variables. This is the measurement.

b. Measurements

Historically, there are three may streams of measurement theory [129]. The "*representational measurement*" theory, initiated with the work of von Helmholtz in 1887 [130], focuses on setting a relationship between objects and number systems via equivalence classes. The second theory, the "*operational measurement*", was introduced by Bridgman in 1927 [131]; its focus is put only on operations used to proceed to measure, neglecting relationships between equivalent measured objects. The last stream is not a single theory but rather a sort of "*melting pot*" of various other theories [132], [133], minor in front of representational and operational theories. In most of these theories, major or minor, we can find a common underlying factor: the scale. We cover "*scale*" concept in the Chapter IV.3.c.

In parallel to these stream of measurement theories, the scientific study of measurement, called metrology [134], establishes convergences related to measurement in human activities through three overlapping active topics: definition, practical realization and traceability. The definition activity of units of measurement drives especially the definition unifications of unit with standard (e.g., international system of units (SI)) and vocabulary (e.g., the international vocabulary of metrology [134] which is written in both English and French). Concerning the practical realization of these units of measurement, this activity aims to address calibration and uncertainty aspect of measurements, including with measuring devices. At last, the traceability activity focuses on bridging the practical realized measurements to the reference standards.

Another aspect linked to measure is the measurement properties. For instance, in Schneidewind's methodology to validate software metrics [135], the author proposed six validity criteria applicable to software metrics for assessment, control or prediction: association, consistency, discriminative power, tracking, predictability, and repeatability. In his analysis of software measurements thanks to metrology [136], Abran focused on the design validation of software measures, including measurement methods, and highlighted key measurement properties such as maturity, precision, uncertainty, calibration and traceability. In health related field [137], it is frequent to use measurement reliability, measurement validity, construct validity, responsiveness for measurement tools or

interpretability property, for example. Among all these properties, there are in fact two main critical properties to qualify a measure, encompassing the instrument and method used to perform it. These properties are reliability and validity of the measure. The reliability means how repeatable over time the measurement is providing consistent results (i.e., with same exact condition and same exact way to measure, measurement values must be equal). Part of reliability is measurement error and internal consistency. Concerning validity, this property means whether or not measured value is measuring what we are expecting to measure; it includes content validity, construct validity and criterion validity. The Figure 15 gives a good visual explanation of impact of these two measurement properties, based on the analogy of “target shooting”. Here, our “sniper” is trying to hit the target shooting several times in the same conditions. If all shootings are in the target, result is valid, and if shootings are close together, we can conclude that the shootings are repeatable over time and therefore result is reliable.

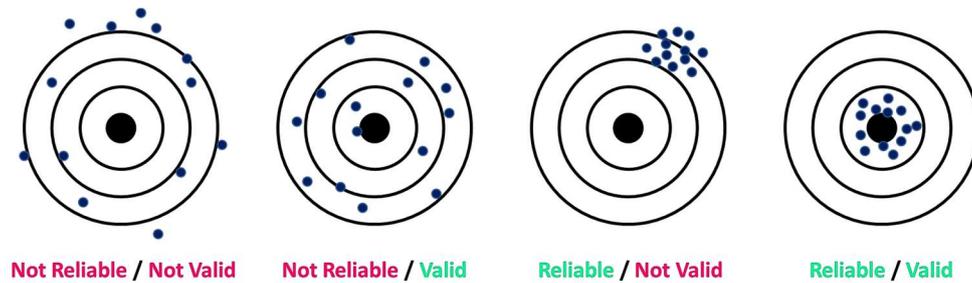


Figure 15 - Illustration of the concept of reliability and validity of measures [138]

Starting from left to right of Figure 15, on the most left target, we notice that the shootings are both spread (i.e., not quite repeatable), inside and outside the target (i.e., they are not valid shootings). On the second target, the shootings are still spread but now all of them are inside the target; we can conclude that the shootings are valid but not repeatable, or reliable. With the third target, all shootings are grouped but some of them are outside; we understand that shootings are repeatable over time but not valid. In the last case not only, shootings are close together and they are all inside the target; the result is therefore valid and reliable. Therefore, we must be in that last case when proceeding with measurement.

Finally, we note that despite which definition is used in any of those theories or activities, “size, amount, or degree” and “standard units” are direct references to the scale theory of measurement, or scale types.

c. Scale types

Scale is essential when we measure quality attributes or properties, with the goal of either assessing that the measurements are matching some criteria (e.g., being above a certain threshold) or controlling quality by comparing their differences over a certain period, or even predicting what will be quality measures. It enables measurement interpretation, standardization, and comparability by describing mathematically types of element with their corresponding operators, properties, functions, permissible statistics, and distributions.

In 1946, S. S. Stevens⁷ introduced and published a theory of scales of measurements [139] which is part of the representational measurement theory. This theory is still widely used by scientist nowadays, despite criticism from statisticians [140]. Stevens categorized data into four typologies or scales: nominal, ordinal, interval, and ratio. The ‘*nominal*’ scale represents the group of data related to labels or type of numbers and words or letters (e.g., {“Red”, “Green”, “Blue”} or {“True”, “False”}). It can be used for classification and membership assessment. The ‘*ordinal*’ scale corresponds to the data set characterized by order, rank, and therefore allows comparison and sort of elements: for instance, it is possible to tell that a quality result is better or worse to another one. The third scale, ‘*interval*’, is extending the field of possible operations one step further by adding difference and affinity operations between values to be meaningful. Thereby, distances between measures are used to take action and decision. For example, man can tell that a quality result is not only better or worse than previous one but also about how much it progresses in term of difference. And last scale is ‘*ratio*’. At that level, and as its name is indicating, the ratio - or magnitude - between measurement values is relevant and useful. It has also the important property of having a unique and non-arbitrary zero value (i.e., Kelvin temperature scale is a good

⁷ An American psychologist who was the founder and director of psycho acoustic laboratory at Harvard University

illustration of a ratio scale since its zero absolute corresponds to -273.15° Celsius, while Celsius temperature scale is an interval scale).

Table 4 summarizes for each scale its associated measure properties, the permissible statistics, the mathematical operators, and group. It is then possible so see that some operations or manipulations are permitted for a specific scale but not for another scale. As illustration of this fact can be found with ISO/IEC 33020 [136].

Indeed, this standard describes a process measurement framework for assessment of process capability, and for example, there are two process capability rating tables, one based on 'nominal' scale (see Table 5) and the second one based on 'ordinal' scale (see Table 6).

It obviously appears that it is possible to compare together measurements in 'ordinal', knowing which one is greater than the other. In 'nominal', this type of operation is not possible, nevertheless we can classify or verify the group membership. Consequently, an accurate scale selection is primordial to operate, process and manipulate appropriately the measures.

In addition to the Stevens' four scales of measurement, there exist several other alternate scales such like Chrisman [142] who proposed an extension of the original Stevens' scale list in 1998, shown in Figure 16. That extended typology is interesting since it is not diverging from Stevens' one. It includes therefore six new scale types that are useful in some particular cases (e.g., gradation of membership is key in fuzzy set theory, log interval is widely use in stock market graphics, cyclical ratio is suitable for times and angles).

TABLE 4 - SUMMARY OF THE 4 SCALES OF MEASUREMENTS DEFINED BY S. S. STEVENS [139]

Scale	Measure properties	Mathematical operators	Mathematical group structure	Permissible Statistics
Nominal	Classification, membership	=, ≠	<i>Permutation group</i> $x' = f(x)$ <i>f(x) means any one-to-one substitution</i>	Grouping (unordered) Number of cases Mode Contingency correlation χ^2
Ordinal	Comparison, level	>, <	<i>Isotonic group</i> $x' = f(x)$ <i>f(x) means any monotonic increasing function</i>	Sorting Median Percentiles
Interval	Difference, affinity	+, -	<i>General linear group</i> $x' = ax + b$	Step, affine line Mean Standard deviation Rank-order correlation Product-moment correlation Regression Variance analysis
Ratio	Magnitude, amount	×, ÷	<i>Similarity group</i> $x' = ax$	Ratio All statistics permitted for interval scales, plus: Geometric mean Coefficient of variation Harmonic mean Logarithms

TABLE 5 – NOMINAL SCALE-BASED RATING, ACCORDING TO ISO/IEC 33020 [136]

Rating	Meaning	Description
N	Not achieved	There is little or no evidence of achievement of the defined process attribute in the assessed process.
P	Partially achieved	There is some evidence of an approach to, and some achievement of, the defined process attribute in the assessed process. Some aspects of achievement of the process attribute may be unpredictable.

L	Largely achieved	There is evidence of a systematic approach to, and significant achievement of, the defined process attribute in the assessed process. Some weaknesses related to this process attribute may exist in the assessed process.
F	Fully achieved	There is evidence of a complete and systematic approach to, and full achievement of, the defined process attribute in the assessed process. No significant weaknesses related to this process attribute exist in the assessed process.

TABLE 6 - ORDINAL SCALE-BASED RATING IN PERCENTAGE VALUES, ACCORDING TO ISO/IEC 33020

Rating	Meaning	Range values
N	Not achieved	0 to ≤ 15% achievement
P	Partially achieved	> 15% to ≤ 50% achievement
L	Largely achieved	> 50% to ≤ 85% achievement
F	Fully achieved	> 85% to ≤ 100% achievement

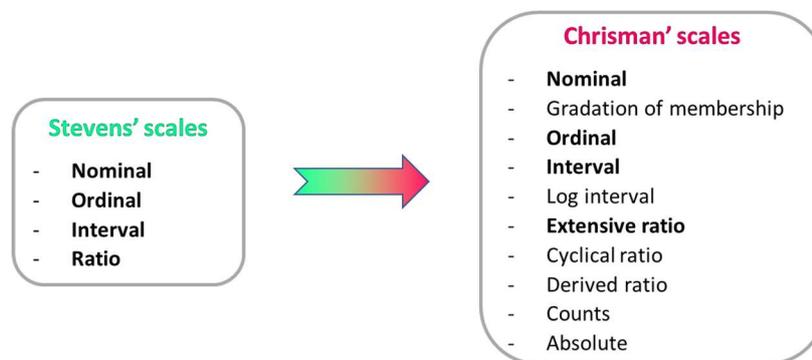


Figure 16 – The Chrisman's scales [142], an extension of the Stevens' scales (in boldface)

To conclude on this part, it is important to select, apply and use the right measurement scale for each property, attribute, and object where we want to apply metrics. Scale brings mathematical and statistical “tools” on measures, rating (e.g., ISO / IEC 33020 [141]) allowing discretization, sampling (i.e., scale change) and validity and reliability of measure such as in [138]. Knowing limits and recommendations from statisticians [140], but also the fact that it is still widely adopted by scientists, we can safely consider Stevens’ typology as the main scale typology to use. And, when it is required to bring more details, accuracy and control on measurement levels, Chrisman’s extended types are offering a good practicable complement.

d. Aggregation

Aggregation consists in operations with the objective to combine elements together, going from a large set of elements to smaller set of elements. In quality modeling, the aggregation is implicit with statistical quality model (e.g., equation (2)), but for the other types of quality model, this is an important operation because it allows to join properly quality measures, attributes, and characteristics for different purposes, thanks to the correct aggregation operators.

According to Wagner [27], there are five major aggregation purposes to consider: *assessment*, *prediction*, *hot spot identification*, *comparison* and *trend analysis*. For *assessment*, the objective is to assess the aggregation results likes for a product quality assessment, where the measures of its quality attributes are aggregated to define the product quality measure. *Prediction* purpose is identical to assessment but with a scope of prediction, and frequently use of statistic model. About *Hot spot identification*, the intent is to focus on specific area of interest, or properties instead of all system or software. It is used for improvement. *Comparison* aims to compare sets of aggregated elements (e.g., systems, software components). Regarding the last purpose, *trend analysis*, the goal is to focus on the evolution over time of some specific, or all, set of elements of interest.

Knowing the aggregation purpose is important but not sufficient to clarify the objectives behind the data aggregation. In fact, we must also identify which aggregation operator category we need before selecting the correct aggregation operator(s). Wagner enumerated seven categories: *grouping*, *rescaling*, *cluster analysis*, *central tendency*, *dispersion*, *concentration*, *ratio and indices* [27]. The goal of *grouping* operators is to regroup elements together into sets, and therefore create groups. Concerning *rescaling*, this is close to the *grouping* category. Indeed, the first step for this kind of operators is to group elements together following a certain scale, and then change that scale. For example, we start with ordinal (e.g., Table 6) and then move to nominal (e.g., Table 5). *Cluster analysis* operators are closed to rescaled group, but the clusters are identified with a more thin and refined analysis (e.g., K-means algorithm). The *central tendency* category corresponds to determine an “average” of elements. Such operators can be mode⁸, median, arithmetic mean, geometric mean, or harmonic mean. The *dispersion* operators qualify the scattering of inputs. In this category we have variation ratio (e.g., for hotspot identification), maximum, minimum, range, median absolute deviation, variance, standard deviation. From an opposite perspective, *concentration* operators (e.g., Lorenz curve – gives graphical hint of the concentration-, Gini coefficient) focus on finding the strongest data cluster (i.e., closest data together). Finally, the *ratio and indices* operators are binary operators because they are defined as the ratio of two measures, where the ratio divider can be also a reference index value instead of a measure). Thus, this category is composed of fraction measures, relation measures or indices.

In addition to the aggregation operators indicated in the above category descriptions, it is possible to find many other aggregation operators. We are not detailing them here but rather initiate their exploration, first by citing the mathematical and behavioral properties that defined them and second, by providing a catalog of the main existing operators. Definition and applicability of these properties and operators are detailed in Detyniecki [143] or Wagner [27]. So, the properties that defined, or characterized aggregation operators are:

- **Mathematical properties:** boundary conditions, Monotonicity (non-decreasing), continuity, associativity, symmetry, bisymmetry, absorbent element, neutral element, idempotence, compensation, counterbalance, reinforcement, stability for a linear function, invariance.
- **Behavioral properties:** decisional behavior, interpretability of the parameters, weighs on the arguments.

The catalog of the main existing aggregation operators is described in Table 39 in Annex 1. We remarked that some operators are weighted (e.g., weighted mean). These weight factors depict the fact that characteristics are of relative important to the other characteristics. Moreover these weight factors are often determined through surveys [27], [51], or expert-nonexpert-hybrid methods from qualimetry [113]. Finally, despite this list of the main aggregation operators, respecting the mathematical and behavioral properties, there is an alternate aggregation approach to include to complete that list. Indeed, in 1997 Dujmovic and Bayucan introduced the “*Logic Scoring of Preference*” method and the “*Continuous Preference Logic*” operators [144] to evaluate quality software products - in their particular case, it was windowed environment software products. The main idea, shown in Figure 17 and by the equations (3), is to specify the relationships (i.e., the “*Continuous Preference Logic*” operators) between the entries E_i and the output $E(r)$, relying on some trade-off between conjunction (i.e., low entry values influence output the most), neutral and disjunction (i.e., high entry values influence output the most). Then the result of the “*Continuous Preference Logic*” operators gives the parameter r which is then injected in the weighted power mean of weighted entries (i.e., E_i) to determine the output result $E(r)$.

$$E(r) = \begin{cases} (\sum_{i=1}^n w_i \cdot E_i)^{\frac{1}{r}} & \text{for } r \neq +\infty \text{ and } r \neq -\infty \\ \max(E_1, \dots, E_n) & \text{for } r = +\infty \\ \min(E_1, \dots, E_n) & \text{for } r = -\infty \end{cases} \quad (3)$$

where $0 \leq E_i \leq 1$, $0 < w_i < 1$ for $i \in [1; n]$ and $\sum_{i=1}^n w_i = 1$

⁸ Mode determines the value which occurs the most often; It is also the only way to determine an average with nominal scale.

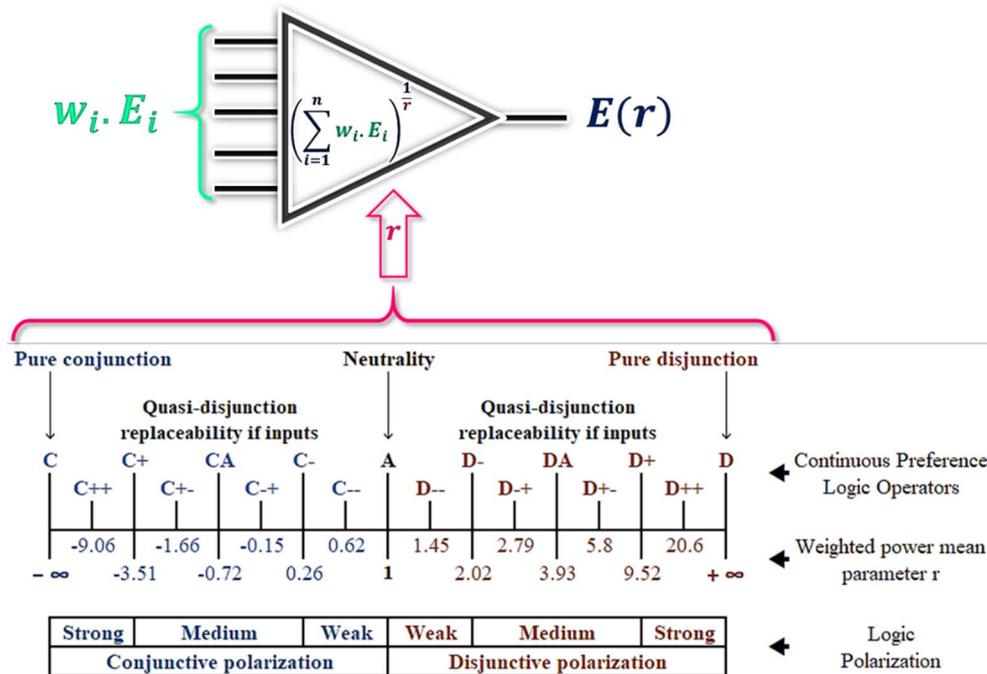


Figure 17 – “Continuous Preference Logic” operators and corresponding weighted power mean parameter used in the “Logic Scoring of Preference” of Dujmovic and Bayucan [144]

e. Threshold

To complete our overview of the fundamental elements that characterized the essence of quality modeling, we end with threshold concept. This concept is closely related to decision making and quality criteria, or quality standards where compliance criteria rely on the comparison between measurements and threshold value (e.g., whether measurements are below, on, or above threshold value). Thus, threshold is mandatory to assess, control or even predict the compliance of products, services, projects, and processes, but it is optional when the quality modeling purpose is only to create, or manage, definition quality model.

Even if a threshold can be perceived as a level that may trigger decisions or actions, we found six distinct categories of threshold from academic and industrial domains.

- **Acceptance, or acceptable:** this is the worst but still acceptable value of a measurement ; this category corresponds to the most frequent use case of threshold and can be linked to risk management (i.e., this is the minimum value from which risk is manageable) (e.g., Azgaldov *et al.* [113], ISO/IEC 25022 [145]),
- **Forecasted:** this is the estimated or predicted value of a measurement ; this category is used within a safety or a process maturity context, for instance (e.g., ISO 26262 [3], ISO/IEC 9126-4 [141], ISO/IEC 25040 [141]),
- **Opportunity:** this is the minimum value of measurement from which opportunities (e.g., in economics, health, environmental) can occurs; it is often associated to management of risk and opportunity (e.g., ISO/IEC 25022 [145]),
- **Rejection:** this is the least worst but still rejected value of a measurement ; it is an alternate category to the acceptable one (e.g., Azgaldov *et al.* [113]),
- **Reference:** this is the best value of the measurement currently achieved during evaluation period ; the reference is usually the best world-wide achievement and may be used as target (e.g., Azgaldov *et al.* [113]),
- **Target:** this is aimed value of the measurement; this category is between, or identical to, the acceptance and the reference ones (e.g., Wagner [27], Azgaldov *et al.* [113], ISO/IEC 25040 [141]).

Another interesting aspect of threshold is the simultaneous use of several thresholds from same or different categories. For instance, the use of two acceptance thresholds defining a range of acceptable values, or the use of acceptance, target, and reference, or again acceptance with opportunity (cf. Figure 18).



Figure 18 - Threshold for risk acceptance and opportunities: from ISO/IEC 25022 [140] theory to practice

Our final highlights on threshold are regarding two specific difficulties that may occur during threshold elaboration. Firstly, it is sometimes hard to specify objective threshold but, as Arhens *et al.* [51] shown, it is possible to get around this obstacle with the use of relative threshold (e.g., reference against a previous release, results to see progress against it). Secondly, the relationship of quality characteristics to each other may influence threshold values. Khaddaj and Horgan [148] established quality characteristics relationship to each other by means of a factor polarity profile (see Table 7).

This polarity profile identifies the dependency links between characteristics or sub-characteristics. There are three types of polarity profile relationship: direct (i.e., if a characteristic A is enhanced, the related characteristic B is likely to be enhanced), neutral (i.e., if a characteristic A is enhanced, the related characteristic B is unlikely to be enhanced) and inverse (i.e., if a characteristic A is enhanced, the related characteristic B is likely to be degraded). For instance, an increase in the security quality characteristic may improve, or degrade the overall performance quality characteristics. This example illustrates the inverse relationship. Further-more, the relationship behaviors of polarity profiles are applicable as well to the thresholds or levels of quality assigned to each quality characteristic.

In conclusion, to define the essence of quality modeling: namely, the abstraction of quality of interest via the identification of independent and dependent variables related to quality (e.g., quality characteristics and sub-characteristics), completed with their mathematical and statistical relationships.

TABLE 7 - EXAMPLE OF KHADDAJ AND HORGAN [148] RELATIONSHIP CHART USED FOR POLARITY PROFILE

		Criterion A										
		Key Quality Factor (KQF)							Locally Defined Factor (LDF)			
		U	S	CB	T	C	M	R	LDF1	...	LDFn	
Criterion B	KQF	Usability (U)	∅	↻	↕	=	=	=	=	=	=	=
		Security (S)	↻	∅	↕	=	=	↻	=	=	=	=
		Cost/Benefit (CB)	↕	↕	∅	↕	↕	↕	↕	=	=	=
		Timeliness (T)	=	=	↕	∅	=	=	=	=	=	=
		Correctness (C)	=	=	↕	=	∅	=	=	=	=	=
		Maintainability (M)	=	↻	↕	=	=	∅	=	=	=	=
		Reliability (R)	=	=	↕	=	=	=	∅	=	=	=
LDF	LDF1	=	=	=	=	=	=	=	∅	=	=	
	...									∅	=	
	LDFn	=	=	=	=	=	=	=	=	=	∅	

Legend:		None (same characteristic)
		Direct (if criterion A is enhanced, then criterion B is likely to be enhanced)
		Neutral (if criterion A is enhanced, then criterion B is unlikely to be enhanced)
		Inverse (if criterion A is enhanced, then criterion B is likely to be degraded)

4. Key contributions to quality modeling of software

To consolidate quality modeling theory, a narrow set of the most well-known, significant, or influential quality models or contributions to quality modeling of software, is frequently cited in papers or used as a basis. In the following content, after providing a timeline with the most significant contributions that we found in that field (see Figure 19), we give details about each of them.

- **1965 – Software Engineering concept** [32]: even if Alan Turing proposed the first theory about software, and more precisely about algorithm, on 1936 [149], the "software engineering" concept only emerged in the 1960s. The first publication that used this term, was "*The Computer Directory and Buyer's Guide*" of June 1965 [32]. This is the first key contribution of the timeline.
- **1968 - Rubey & Hartwick model** [33]: Following their study on quality for spaceborne software, Rubey and Hartwick contribution was the first published hierarchical quality model for software product. Their model, organized around a set of 7 quality attributes from mathematical and logical correctness to usability, is associated to 57 metrics composed of checklists, logical, mathematical or structural metrics. The quality perspectives of this model are both product and user.
- **1968 - Shooman model** [34]: Parallely to Rubey and Hartwick, Shooman's model was the first published quality model for software reliability. This model estimates the error detection rate based on the number of errors per instruction and the average instruction processing rate. It is therefore the first published software reliability model and by consequence reflects the manufacturer view.
- **1968 - Azgaldov et al. Qualimetry genesis** [2]: Azgaldov *et al.* laid the foundations of "Qualimetry", the science of quality quantification. Thus, qualimetry is a scientific discipline which uses methods for quantifying quality. It covers both the theoretical and applied aspects of quality quantification with a range of activities such as quality definition, assessment, control, or prediction. The main original contributor to this science is Azgaldov whose PhD [2] was to build, study and develop qualimetry.
- **1976 - Boehm's quality model** [42]: Boehm's quality model is one of the most famous hierarchical quality models built by Boehm *et al.*. In 1973, a first model was published [150]. It consisted of 7 primitive characteristics linked to 12 intermediate characteristics, themselves associated with about 151 metrics. Subsequently, the model underwent successive evolutions during the 1970s. The first iteration was completed in 1976 [42], mainly by grouping primitive characteristics under three types of actual use of software, namely: general utility, as-is utility and maintainability. The last release of this quality model in 1978 [128] was consolidated with a few more primitives and intermediate characteristics. The quality perspective of this model is a product view.
- **1978 - McCall's quality model** [41]: McCall's quality model is also one of the most well-known hierarchical quality models, known also as FCM (i.e., Factor-Criteria-Metrics). It was published in 1977 by Mc Call *et al.* Their model is structured around a hierarchy of 3 factors, 11 criteria and a minimum of 23 to 35 metrics. It is sometimes known as the FCM model, the acronym for Factor-Criteria-Metrics. The main difference with Boehm's model is that the factor decompositions are associated with three specific software product activities: product revision, product transition and product operation. This quality model reflects a product quality perspective.
- **1984 - Garvin's quality perspectives** [101]: In his study about quality Garvin remarked that previously philosophy, economics, marketing and operations management disciplines considered quality from their own vantage point of view, each view competing with the others. By summarizing the views on quality over five distinct quality perspectives, Garvin offered a way to reunify these divergences and use "*quality as a competitive weapon*"[101].

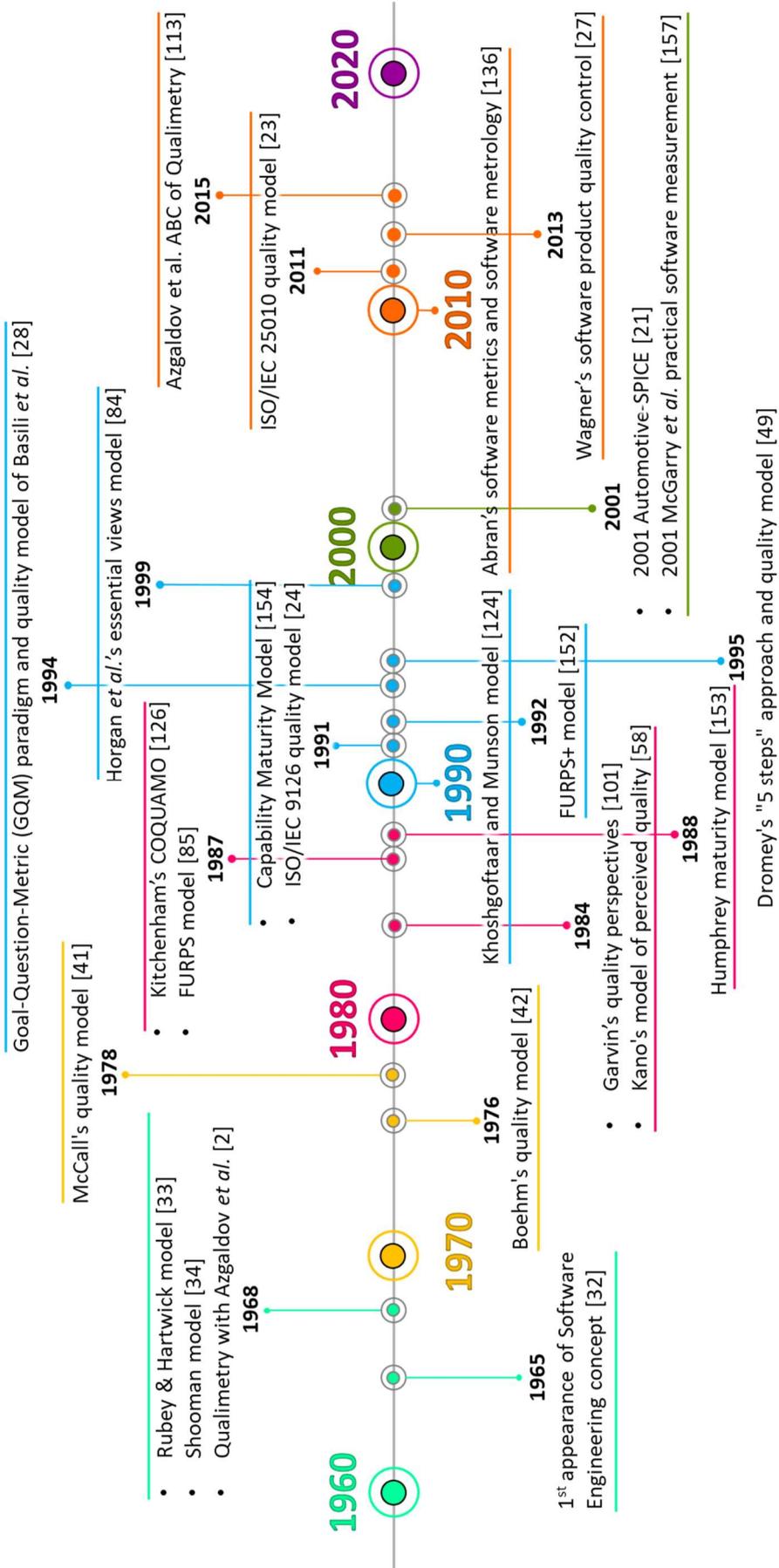


Figure 19 - Timeline of the key contributions to quality modeling of software

- **1984 - Kano's model of perceived quality** [58]: Kano's model is a particular quality model since its purpose is to integrate customer preferences and satisfaction through the concept of perceived quality. In addition, Kano *et al.* found that the perception of quality evolves with time, similarly to Lehman's laws of software evolution [82], and therefore, quality definition and evaluation must be adapted accordingly. This model can also be used to identify the most important product attributes from a customer perspective. This model covers product, user and value-based quality perspectives.
- **1987 – Kitchenham's COQUAMO** [126]: This research work was performed within ESPRIT⁹ project for reliability and quality of European software. Thus, in this context Kitchenham defined the Constructive QUALity MOdel (i.e., COQUAMO) approach, a 'do your own' quality model quite similar to the Boehm's constructive cost model, also known as COCOMO, approach [151]. She then used it to create a quality model based on Boehm [42] and McCall models. So, she identified the main quality drivers, or factors, from Boehm and McCall [41] models. She finally limited to five main factors for user perspective and three for manufacturer one. Therefore, the quality perspective addressed by COQUAMO are the user and manufacturer ones.
- **1987 the FURPS [85] & 1992 FURPS+ [152] quality models**: The FURPS [85] quality model was built by Grady and Caswell in 1987. This model is interesting in that it is adapted to the different stages of a software life cycle without modifying its 5 characteristics (Functionality, Usability, Reliability, Performance, Supportability). This is achieved by fitting the measures used for each characteristic to the current stage. In 1992 a new model based on FURPS was created: FURPS+ [152]. This new model is an extension of the original model, which specifies constraints with respect to design, implementation, interface and physical properties and conditions. Both models cover a user and a product quality perspective.
- **1988 - Humphrey's maturity model** [153]: Humphrey's maturity model is the first published maturity model, and it corresponds to a framework for process maturity. This model is key because it is the original foundation of many other maturity models like the Capability Maturity Model (CMM) of the Software Engineering Institute (SEI) [154] in 1991, or Automotive-SPIICE [21] in 2001, for example. This model represents the manufacturer quality view.
- **1990 - Khoshgoftaar and Munson model** [124]: Khoshgoftaar and Munson model represents one of the earliest starting points of a long series of research studies and fruitful contributions led by Khoshgoftaar on prediction of error-prone quality models for telecommunication systems software. During their researches, they explored a wide range of likelihood methods to elaborate their quality models. To cite a few of them, they investigated regression analysis, discriminant analysis, neural network, logistic regression, classification tree, genetic algorithm, analogy-based classification, and Poisson regression. It appears that the resulting quality models were specific to their study case and therefore couldn't be reused as it for other types of systems or software. Obviously, they are focusing on manufacturer quality perspective.
- **1991- the ISO/IEC 9126 quality models** [24]: The ISO/IEC 9126 quality model was first published in 1991, with a final release in 2001. This hierarchical quality model is the first standardized quality model. It got its inspiration from the Boehm and McCall models. ISO/IEC 9126 is a hierarchical quality model composed of two quality views (internal & external quality, quality-in-use), 10 quality characteristics and 27 quality sub-characteristics. The suggested metrics are provided by ISO/IEC TR 9126-2 (external metrics) [155], ISO/IEC TR 9126-3 (internal metrics) [156] and ISO/IEC TR 9126-4 (quality-in-use metrics) [146]. Its status as an official standard quality model made it a reference that is widely used as a basis for many quality models. Obviously, this quality model is based on a user and a product quality perspective.
- **1994 - the Goal-Question-Metric (GQM) paradigm and quality model of Basili et al.** [28]: The Goal-Question-Metric (GQM) paradigm and quality model of Basili *et al.* was first cited in 1984. It is frequently used as a common process for creating quality models. Its process flow starts by identifying quality goals,

⁹ European Programme for Research and Development in Information Technologies: <https://cordis.europa.eu/project/id/300>

followed by the related requirements needed to meet those goals, and finally completed by a set of metrics to monitor the extent to which the requirements have been met. The quality perspective focused with this approach and the resulting quality models cover product, manufacturer, and user views.

- **1995 - Dromey's "5 steps" approach and quality model** [49]: After noticing that quality assessment differs for each product and consequently requires a process to handle this behavior, Dromey defined a 5-step process to evaluate and construct a hierarchical quality model based on ISO/IEC 9126. His approach is crucial because it allows quality model verification by establishing criteria for deciding whether to include software properties as an integral part of a model, and a way of assessing whether the model is complete or not. The quality perspectives addressed here are user and product views.
- **1999: Horgan et al.'s essential views model** [84]: During his PhD on "Construction of a Quality Assurance and Measurement Framework for Software Project", Horgan noticed a specific behavior related to quality characteristics, or factors. Indeed, he found that some factors were key, that is to say, almost invariant whatever the project, product or stakeholders were (e.g., some quality factors that are always requested or required) and some factors were merely local to a project, product or stakeholders (e.g., a quality factor specific to a project and inadequate for another one). Horgan named them as the "*Key Quality Factors*" and "*Locally Defined Factors*". He merged them under the essential views model which corresponds to both product and user quality perspectives.
- **2001 - McGarry et al. practical software measurement** [157]: McGarry et al. constructed a practical guide related to measure and based on the experience. They describe a measurement information model to help on the definition and implementation of metrics, and a measurement process to guide the planning and execution of measurement activities. In addition, in both model and process, they included management concepts to ensure that the requirements and rationales for decision makers were taken into account in their definition (e.g., measurable concept, reason behind the metric) and risk was reduced during their implementation. At last, this body of knowledge, on practical software measurement, highlighted the importance of user feedback in measurement design. Hence, this contribution is a key support material about measurement for software quality modeling.
- **2010 - Abran's software metrics and software metrology** [136]: In the same line of work of McGarry et al., Abran not only analyzed knowledge about the most popular software measures in the industry (e.g., McCabe's cyclomatic complexity, Halstead's metrics, function points, use case points), but also addressed specifically two issues: "*how to correctly design software measures, and how to recognize if a software measure is well designed, and worth using as a basis for decision - making*". To do so, the author used the International Vocabulary of Basic and General Terms in Metrology [134] as the foundations of his studies on software metrology, software measure design and measurement methods, and then illustrated the COSMIC [158] design as an practical example of the lesson learned with those analyses. Finally, through this work, Abran aimed to avoid the main practical pitfalls with software measure use in the industry by setting up a trust relationship between software measure use and industry practitioner and managers.
- **2011 - the ISO/IEC 25010 quality model** [23]: Subsequent to the ISO/IEC 9126 standard, ISO/IEC 25010 [13] was initiated at the end of 2007 and the result was published in 2011. This is the current official standard for systems and software-quality models - with a wider scope than ISO/IEC 9126; consequently, it is actively and widely used. At the current time, ISO/IEC 25010 consists of two hierarchical quality model parts: a systems/software product quality model, with 8 quality characteristics and 31 quality sub-characteristics, and a quality-in-use model, with 5 quality characteristics and 9 quality sub-characteristics. This standard is supplemented with a data quality model, ISO/IEC 25012, for which there are 15 "data quality characteristics that are required or evaluated from inherent and/or system-dependent points of view" [133]. Suggested metrics are given by the ISO/IEC 2502n series. Like its predecessor, the quality perspective scope of ISO/IEC 25010 is based on a user and a product view.
- **2013 - Wagner' software product quality control** [27]: In 2011, Wagner started to synthesize and consolidate in a book the knowledge about software quality product. He integrated many research contributions in that field, particularly thanks to his participation as the project leader of Quamoco

project [160] in which he accumulated experience. This project focused on quality models and quality evaluation, and hence the book content covers those aspect. Indeed, the author began with miscellaneous definitions and terminologies related to quality, then continued with knowledge about quality models and software measures, description of quality planning and control, and ended the book by sharing and describing six practical experiences Wagner had during the 10 years prior to this book. So, the compiled knowledge about quality model and evaluation, complete with a 10 years' experience sharing makes this book a key contribution.

- **2015- Azgaldov et al. ABC of Qualimetry** [113]: Through this book in English, Azgaldov *et al.* extended the diffusion to qualimetry knowledge to a wider audience and continue to contribute to this science. We note that this book is a key contribution because, many contributions to that science are in Russian or address to some specific domain such as socioeconomics, and hopefully this book opens a bridge to other discipline, such as software engineering. This is the last key contribution of the timeline.

5. Qualimetry: the science of quality quantification

Research Sub-question 1c What is qualimetry, and is this approach the right one for our needs?

Qualimetry, from the Latin *qualis* "of what kind" and the Greek *μετρεω* "to measure", can be described as the science of method and problem solving for quality quantification of any kind of object such as service, product, people, project or process [113].

The concept of quality quantification is not recent as was indicating G.G. Azgaldov [113], the founder of qualimetry; Diez in [132] also clearly emphasized this fact, citing works done by Helmholtz in 1887 or by Campbell in 1920. However, until late 1960's the quantification of quality was exclusively done for only one specific type of object at a time, primarily product oriented one and without direct or explicit reuse or generalization over other similar objects. On the beginning of 1968, in former U.S.S.R., a scientist group of interest around problems linked to quality quantitative evaluation and control, published a common summary paper of their workshop [2]. The force of this group was due to the fact that its members were coming from a large variety of domain horizons (e.g., economists, architects, civil engineers, car makers) and shared the same concerns to unify theories and practices used for quality quantification. It was the ignition of an international discussion that led to the birth of qualimetry, a new scientific discipline, during 1968.

In addition, Azgaldov *et al.* [113] demonstrated that qualimetry was not only a scientific discipline but a real science by itself, reminding what Plato said on the 5th century B.C. "*Exclude from any science mathematics, measure and weight, and it is left with very little*". In 1981, that science was split into two distinct disciplines: theoretical qualimetry [161] (i.e., focusing on problems and method issues, with a mathematical view to object to evaluate) and applied qualimetry [162] (i.e., application of qualimetry to evaluate type of objects that were not evaluated before). We remark also that qualimetry relies on domain-independent concepts, or foundations, and therefore can be applied to any domains.

In order to quantify quality, Azgaldov *et al.* defined a general quality modeling process via a quality assessment algorithm [113]. This algorithm is composed of 9 sequential steps that rely on Quality Evaluation Method (QEM) design, planification and execution. A QEM is a method used to define and assess quality, including definition of evaluation context, quality characteristics, their measurable properties, or indices, their weight factors, to cite some of them. There exists three distinct QEM: rigorous, short-cut, approximate. A rigorous method aims to be exhaustive in the quality evaluation, to reduce error and to obtain the most reliable results, but it is expensive in term of resource and time (e.g., detailing with a high precision details quality characteristics and sub-characteristics, using all available measures to evaluate the quality). At the opposite, short-cut QEM tends to have narrow quality evaluation, less accurate results, but still acceptable, and therefore is cheaper in term of resource and time (e.g., applying ISO/IEC/IEEE 25010 with only one metric per quality characteristics). The last QEM, approximate, is between rigorous and short-cut. Azgaldov *et al.* [113] highlighted that short-cut is the most used internationally. Also, the team which is the source of information for QEM values can be either formed by expert, non-expert, or hybrid (i.e., mix of expert and non-expert) from the domain where the quality evaluation is required. In addition, the data origination used by qualimetry is a hierarchical quality model and to support the

analysis and the model elaboration, a set of about 30 quality model tree derivation rules have been defined. The 16 principal rules are listed in Table 8, for general rules, and in Table 9, for specific rules, while their details are available in Azgaldov *et al.* [113].

TABLE 8 - GENERAL RULES FOR QUALITY MODEL TREE DERIVATION

Id	Rule
Global rules	
1	Maximum height of tree
2	Branch a tree until only simple or quasi-simple properties remain at its top tier
3	Preference Indifference of Properties in a Group
4	Exhaustive consideration of the Application features of an object
5	Exclusion of reliability ¹⁰ properties (i.e., this must be part of integrated quality index => K_{use})
6	Structural rigidity of the primary tiers of a tree
General sub-tree rules	
7	Division by an equal characteristic
8	Functional orientation of property statements
9	Necessary and sufficient number of properties in a group
10	Reference number of purpose properties within a group

TABLE 9 - SPECIFIC RULES FOR QUALITY MODEL TREE DERIVATION

Id	Rule
Specific rules for the application of the expert method to weight factor	
11	Random order of properties in a group
12	Minimum number of properties in a group (maximum ≤ 9)
Specific rules to be used if the amount of information obtained in a quality assessment can be reduced thought the use of the rank scale	
13	Exclusion of equally expressed properties when the rank scale is admissible
14	Truncated tree when the rank scale is admissible
Specific rules to be used if the amount of information obtained in a quality assessment may/may not be reduced by more precise methods	
15	Incomplete tree when a short-cut assessment of quality is admissible
16	Complete tree when exact quality assessment alone is admissible

After this introduction and global outline of qualimetry, we must address the second part of our research question: *“is this approach the right one for our needs?”*. To recall our needs, we look for a unified, operational, and appropriate way to define, assess, control, or predict quality of embedded software, including handling of reuse, variant and respect of standards and regulations (cf. Chapter II.2). So, to answer to that question, we propose to compare with other existing approaches and then determine which one is the optimum one. We have identified eight main distinct streams of approaches for quality model development and use: Azgaldov *et al.* [113], with qualimetry, Basili [28], with GQM paradigm, Dromey [49], with the 5-step approach, Gilb [163], with the principles of software engineering management, Khoshgoftaar and Munson [124], to cover stream of statistic models, ISO / IEC 250nn [23], [121], [123], [145], [147], [164], [165], to cover stream of hierarchical quality models (including standard), Kitchenham [126], [127], with COQUAMO and SQUID, and Wagner [27], with QUAMOCO. We may note that there exist other works quite similar to these quality modeling approaches, nevertheless, these eight streams are a good synthesis of the current distinct approaches, and their main characteristics, from a quality modeling perspective summarized in Table 10. Moreover, these approaches can be regrouped into three categories: specific quality model solution, generic quality model solution and general methods.

First, the specific quality model solutions which have a narrow focus. This is the case with Khoshgoftaar and Munson [124] where the authors showed how to develop and use statistic or implicit quality models. This type of approach is often applied to a specific assessment or prediction case, and therefore it is hardly generalized or reused. Another type of specific solution is the one from Gilb [163] who proposed some principles of software engineering management to have a good software quality. He considered that quality requirements come from

¹⁰ Reliability include storability, faultless operation, maintainability and durability ; Published via a Russian regulatory documents (GOSTs) decree [113].

the required operational properties of software product, or from its maintenance environment. His solution finally is very specific and address neither development, not use of quality model.

TABLE 10 - COMPARISON OF EIGHT MAIN DISTINCT APPROACHES SUPPORTING QUALITY MODEL DEVELOPMENT AND USE

Stream of approach	Wagner [27]: QAMOCO	Kitchenham [126], [127]: COQUAMO and SQUID	ISO / IEC 250nn [23], [121], [123], [145], [147], [164], [165]: Stream of Hierarchical Quality models (and standard)	Khoshgoftaar & Munson [124]: Stream of Statistic software engineering models	Gilb [163]: Principles of software engineering management	Drome Y [49]: 5-step approach	Basili [28]: GQM Paradigm	Azgalov <i>et al.</i> [113]: Qualimetry
Quality model scope	Project and Software product	Software product	System & Software product and in use	Software product	Software product	Software product	Software product Any domain	Any domain
Evaluation context & plan	-	-	Evaluation plan	-	-	-	-	Evaluation context
Purposes	<ul style="list-style-type: none"> • Definition • Assessment • Prediction • Multi-purpose 	<ul style="list-style-type: none"> • Definition • Assessment 	<ul style="list-style-type: none"> • Definition • Assessment (evaluation part) • Prediction 	<ul style="list-style-type: none"> • Assessment • Prediction 	Assessment	Definition	Assessment	<ul style="list-style-type: none"> • Definition • Assessment
QEM: method to assess quality	Not specified but assumes short-cut method	short-cut method	Not specified but assumes short-cut method	Short-cut method	short-cut method	Not specified but assumes short-cut	Not specified but assumes short-cut method	<ul style="list-style-type: none"> • Rigorous method • Short-cut method • Approximate • Expert • Non-expert • Hybrid
QEM: source of information about values in QEM	Hybrid	Hybrid	Hybrid	Expert	Expert	Hybrid	Hybrid	Hybrid
Data organizational types	<ul style="list-style-type: none"> • Hierarchical • Meta-model • Statistical and Implicit 	<ul style="list-style-type: none"> • Hierarchical • Meta-model 	<ul style="list-style-type: none"> • Hierarchical • Meta-model 	Statistical and implicit	Hierarchical	Hierarchical	Hierarchical	Hierarchical
Weight factors	Per property / characteristic	-	Per property / characteristic	per variables	-	-	-	Per property / characteristic

The second category is the generic quality model solution. We choose ISO/IEC 25000 [23], [121], [123], [145], [147], [164], [165], the current standard for hierarchical quality modeling on software product, as the representative solution here. Indeed, it provides a good illustration of existing contributions to create and publish other hierarchical quality models, and that can be found in literature in general. ISO/IEC 25000 suffers from being too general since its quality characteristics / sub-characteristics definitions are relatively ambiguous (e.g., Abran *et al.* [96]) with the objective to cover a maximum of cases. In addition, this standard requires not only customization (e.g., Wagner *et al.* [83]), but also a method to complete, for example, aggregation and weight factor part.

The third and last category encompass the general methods for development and/or use of quality model. Dromey [49] 5-steps approach, is a method which focus on the quality model development and more particularly on the assessment of the built quality model. Regarding Basili's GQM paradigm [28] (generalization of McCall Factor/Criteria/Metric [41]), the approach is performed over 3 stages, starting from a series of goals, which are split into a proper set of questions and where set of metrics, associated to these questions, are answering to the goals (e.g., Shepperd [166]). This is a general method with a wide scope, but it doesn't pay attention to some details such as weight factors, aggregation or even operationalization. GQM result is subjective because of its process flow starting from goals. Kitchenham [126], [127] proposed methods for 'do your own model' with COQUAMO and SQUID. Thus, the approaches focus on only on quality model creation. Extending this work through QUAMOCO research project [160], Wagner [27] consolidated in a book a complete set of knowledge accompanied by methods to develop and use software product quality model. At last, Azgaldov *et al.* [113] represent the general scope of qualimetry approach, quality assessment algorithm that includes characteristic and property identification, tree construction, weight factors aggregations, and therefore covering both development and use of quality model too.

In conclusion, by analyzing all the cited approaches together, and then against our needs, two optimum approaches stand out:

- Wagner and his book, which is a consolidation of knowledge and experience in quality modeling,
- Azgaldov *et al.* and qualimetry, which is the science of quality quantification, merging methods and approaches from various domains.

Both Wagner's approach and qualimetry appear to be exhaustive and complete compare to the other approaches, dealing with theory and applied aspect. Nevertheless, compared to Wagner's approach, qualimetry theoretical and applied aspect are actively developed, supported by a large community of scientists from a variety of fields (e.g., architecture, economy, civil engineering, factory), and also addressing a larger scope of object to quantify, even if Wagner focused on software product which is well aligned with embedded software.

Consequently, qualimetry is the optimum approach for a unified, operational, and appropriate way to define, assess, control, or predict quality of embedded software.

However, we noticed that we could improve qualimetry with the following contributions.

6. Contributions to Qualimetry

a. The "House of Qualimetry"

The first contribution to qualimetry is linked to its comprehension. In fact, this relatively young science, which has a large scope, is not widely used even in software engineering and in systems engineering. Indeed, we encounter only specific applied qualimetry case studies which are mostly decorrelated from theoretical qualimetry. So, in order to leverage qualimetry to a large range of audience, foster its accurate understanding, and ensure that no major knowledges beneath it are eluded or forgotten, we propose a synthesized view of this science through the "House of Qualimetry" and its 6 pillars, depicted by Figure 20. We note that by consequence, this synthetic view answers to the first part of our research questions Q1 (i.e., "What are Qualimetry essentials which make Qualimetry a right answer to quality quantification?"): the qualimetry essentials.

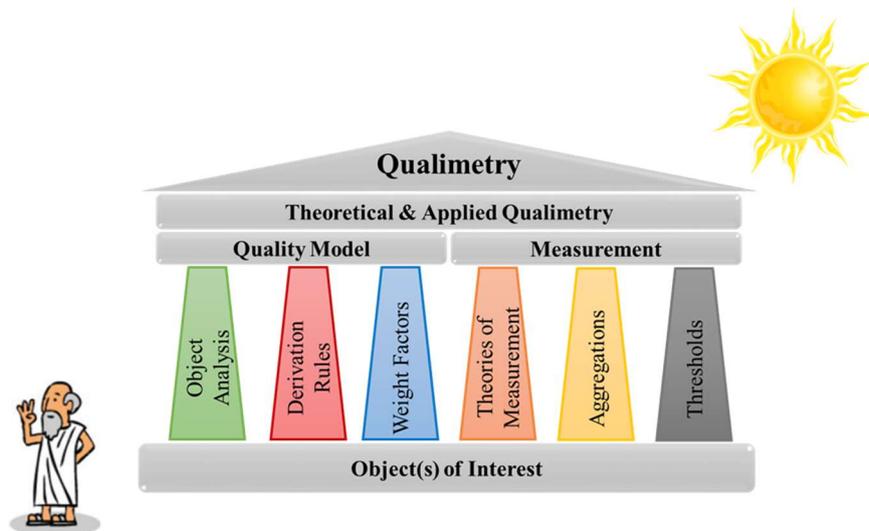


Figure 20 - The “House of Qualimetry” and its 6 pillars

We initiated the elaboration of this synthetic view by exploring and analyzing qualimetry, on both aspects, theoretical and applied one. Then, we used our knowledge and experience with regards to quality modeling in software engineering and systems engineering to determine the major qualimetry themes. In addition, we voluntarily limited their numbers because from experimental psychology, as Azgadnov *et al.* pointed out [113], we learn that the number of units of operational information an individual can handle simultaneously is limited to an average of 7 ± 2 . The result by Argotti *et al.* [167] is shown in Figure 20 and an overall explanation is given below with a global summary in Figure 21.

To describe the structure of these concepts, we make an analogy with architecture, borrowing Doric architecture vocabulary. As a science, qualimetry relies naturally on two interlaced and complementary disciplines: theoretical [161] and applied qualimetry [162]. These two disciplines are combined into an entablature which relies on two architraves: “quality model” and “measurement”. “Quality model” covers the identification, organization, and representation of the relevant quality characteristics while “measure” covers the evaluation, manipulation, and control of them. Furthermore, each of these two architraves is relying on a set of three pillars, all settled on a basement reflecting the object(s) of interest (i.e., the one(s) that is (are) aimed to be quality quantified).

i. The “Quality Model” pillars

While the first pillar (i.e., “object analysis”) is the major one, the other two are also mandatory in order to achieve the right quality model.

- **“Object analysis” pillar:** This pillar gathers the necessary knowledge and activities to understand, identify and organize the relevant quality goals, perspectives and characteristics linked to the analysis of our object of interest (i.e., the one that it aims to have its quality quantified). Thus, we first define the purpose of our analysis, aligned with the DAP classification (see Figure 11); we then analyze our object of interest in order to identify the quality objectives, perspectives, characteristics, sub-characteristics; attributes... that are relevant to us; finally, we decide how we are going to organize all this data. We can note that quite often the data organization is achieved *via* a hierarchical structure (i.e., tree structure) but can be also achieved through a statistical model.
- **“Derivation rules” pillar:** Here, the focus is with regards to global and specific qualimetry rules (see TABLE 8 and TABLE 9) to help support the analysis and optimize the design of the organizational data structure. For example, maximum tree height, division by equal characteristic, branch a tree until only simple or quasi-simple characteristics remain at its top tier.
- **“Weight factors” pillar:** Regularly forgotten, even in standards such as ISO/IEC/IEEE 25000 series where it is reduced to few words, the weighting factors are critical because they reflect the importance of quality characteristics among the same level of quality characteristics.

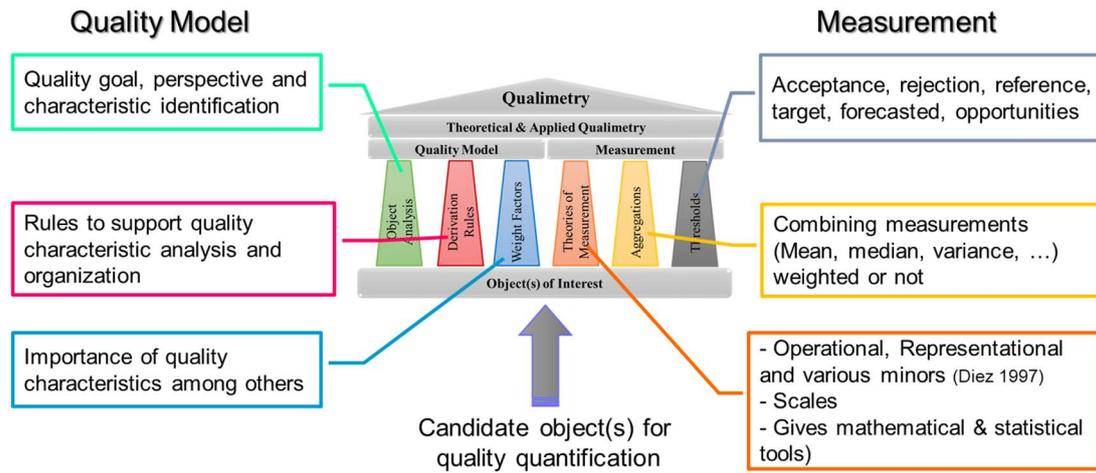


Figure 21 - The "House of Qualimetry" and its 6 pillars in a nutshell

ii. The "Measurement" pillars

As was the case for the previous set of pillars, these three pillars are all mandatory in order to proceed accurately on measurement taking, even if the "theories of measurement" pillar represents the main one.

- **"Theories of measurement" pillar:** This pillar is composed of three main streams of measurement theories. [132], [133]: operational measurement (i.e., how to operate or use the measure), representational measurement (i.e., how to represent the measure) and "various minor" theories. In a sense this is a fundamental pillar as it is bringing together all scale, mathematical and statistical tools for our measurements.
- **"Aggregations" pillar:** The aim is to deal with the way of combining (i.e., mean, median, variance and more [143]) together either all or a subset of the measurements depending on their purpose [27]. The aggregated measurements can either be weighted or un-weighted.
- **"Thresholds" pillar:** This pillar is associated with the measure of the ability to assess, control, or predict and therefore make the correct decision. In general, man is using two types of thresholds: acceptance and target. Acceptance is often confused with the rejection threshold even though they are not the same: the acceptance threshold is the worst-case threshold level that may be accepted, it lies just above the best case reject level. In fact, we counted six types of threshold exist as follows (refer to Chapter IV.3.e): rejection, acceptance, target, reference, forecasted and opportunities. Target corresponds to the threshold we are aiming for whereas reference corresponds to the reference value used in the industry or in the community at the time when the measurement is taken. Opportunities threshold corresponds to the level from which economical, health or environmental opportunities can occur. And lastly, forecasted allows to express estimated or predicted threshold, often used within a safety or a process maturity context.

Finally, by mean of the "House of Qualimetry", we popularize a complex concept with a synthetic view easy to remember, visualize, use, and understand. The Figure 22 illustrates an example of the application of two of three "quality model" pillars against "software product" as object to quality quantify, demonstrating that could elaborate the same quality models defined in the ISO/IEC/IEEE 25010 standard.

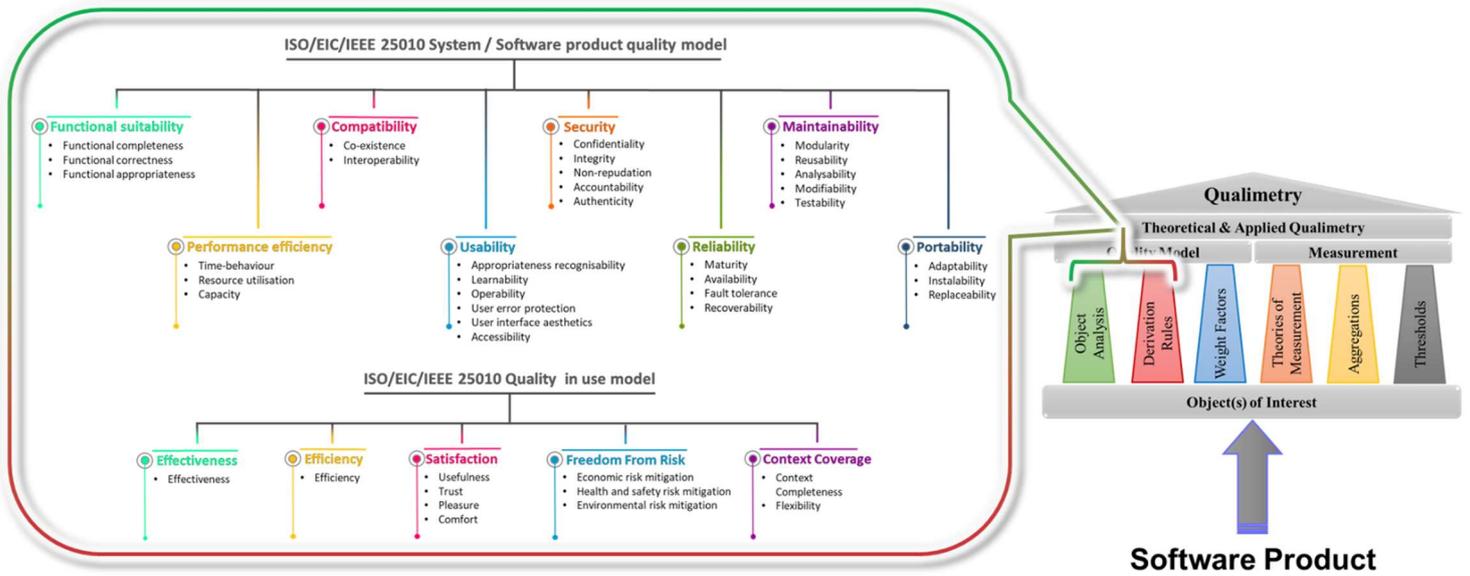


Figure 22 - Example of a result from the application of the two of the three pillars of quality model architecture against “software product”: the ISO/IEC/IEEE 25010 quality models [23].

b. Polymorphism applied to quality model

Research Sub-question 1d How to unify diversity and time evolution in quality modeling?

Within the goal to answer to this research sub-question 1d, the second contribution aims to overcome a lack in quality modeling, and therefore in qualimetry. Indeed, even if there already exist some elements of solution, there is no generic mechanism to link, unify and homogenize quality modeling, during development, use and reuse of quality model. This is key for similar objects (e.g., variants of product) that are candidate to be quality quantified. For instance, in 1999 Horgan *et al.* [84] identified the *Key Quality Factors* as the quality factors that are invariant over similar product or project, but not equal, and the *Locally Defined Factors* as the quality factors that must be customized to represent the differences between similar product or project: for instance different releases of a smartphone series, variants of a car (e.g., same type of car but different options like in premium versus low-cost, or different car types such as convertible, mini, sport, van for instance). Another example Grady and Caswell’s FURPS model [85] where depending on the development process stage, metrics are adapted to current stage (cf. example extracted from Grady and Caswell’s book and shown over Table 11).

TABLE 11 - EXAMPLE OF DISTINCT MEASURABLE OBJECTIVES USING FURPS QUALITY MODEL FOR EACH LIFE CYCLE PHASE (SOURCE: GRADY AND CASWELL [85], FIGURE 11-7, PAGE 161)

	Investigation / Specification	Design	Implementation	Testing	Support
Functional	<ul style="list-style-type: none"> # target uses to review spec or prototype % grade on report card from user % features competitive with other products # interfaces with existing products 	<ul style="list-style-type: none"> % spec included in design # changes to spec due to design requirements # users to review change if needed 	<ul style="list-style-type: none"> % designs included in code # code changes due to omissions discovered % features removed (reviewed by original target user) 	<ul style="list-style-type: none"> % features tested at alpha sites % user documentation tested against product # target alpha customers 	<ul style="list-style-type: none"> Known problem reports Sales act. Reports (esp. lost sales) User surveys Internal HP user surveys
Usability	<ul style="list-style-type: none"> # target uses to review spec or prototype % grade on documentation plan by target user 	<ul style="list-style-type: none"> % grade of design as compared to objectives # changes to prototype manuals after review 	<ul style="list-style-type: none"> % grade by other lab user % grade by product marketing documentation 	<ul style="list-style-type: none"> # changes to product after alpha test % grade from usability lab testing % grade by test sites 	<ul style="list-style-type: none"> # user misunderstandings

	<ul style="list-style-type: none"> • % grade on usability of prototype 		<ul style="list-style-type: none"> • % original users to review any change 		
Reliability	<ul style="list-style-type: none"> • # omissions noted in reviews of objectives (reliability goals) • # changes to project plan, test plan after review 	<ul style="list-style-type: none"> • # changes to design after review due to error • % grade of design as compared to objectives 	<ul style="list-style-type: none"> • % code changed due to reliability errors discovered in reviews • % code covered by test cases • # defects / KNCSS during module testing 	<ul style="list-style-type: none"> • MTTF (MTBF) • % hrs reliability testing • # defects / 1K hrs • # defects total • Defect rate before checkpoints 	<ul style="list-style-type: none"> • # known problem reports • # defects / KNCSS
Performance	<ul style="list-style-type: none"> • # changes to objectives after review • % grade on objectives by target user • % grade on objective by product managers 	<ul style="list-style-type: none"> • % product to be modeled in defined modeled environment 	<ul style="list-style-type: none"> • Performance tests achieve % of modeled expectations • % of code tested with targeted performance suite (module) 	<ul style="list-style-type: none"> • Achieve performance goal with regards to environment(s) tested • % of code tested with targeted performance suite (system) 	
Supportability	<ul style="list-style-type: none"> • # changes to support objectives after review by field & CPE 	<ul style="list-style-type: none"> • # design changes by CPE & field • # diagnostic / recovery changes by CPE & field input 	<ul style="list-style-type: none"> • MTTR Objective (time) • MTTC Objective (time) • time to train tester, use of documentation 	<ul style="list-style-type: none"> • MTTR Objective (time) • MTTC Objective (time) 	<ul style="list-style-type: none"> • MTTR Objective (time) • MTTC Objective (time)

Fortunately, in genetic (e.g., Joron *et al.* [168]) as well as in program-oriented object (e.g., Cardelli and Wegner [169]), we already have a mechanism for this type of behavior: the polymorphism concept. So, we propose to integrate the concept of polymorphism to quality modeling, quality model, and by extension we contribute to qualimetry.

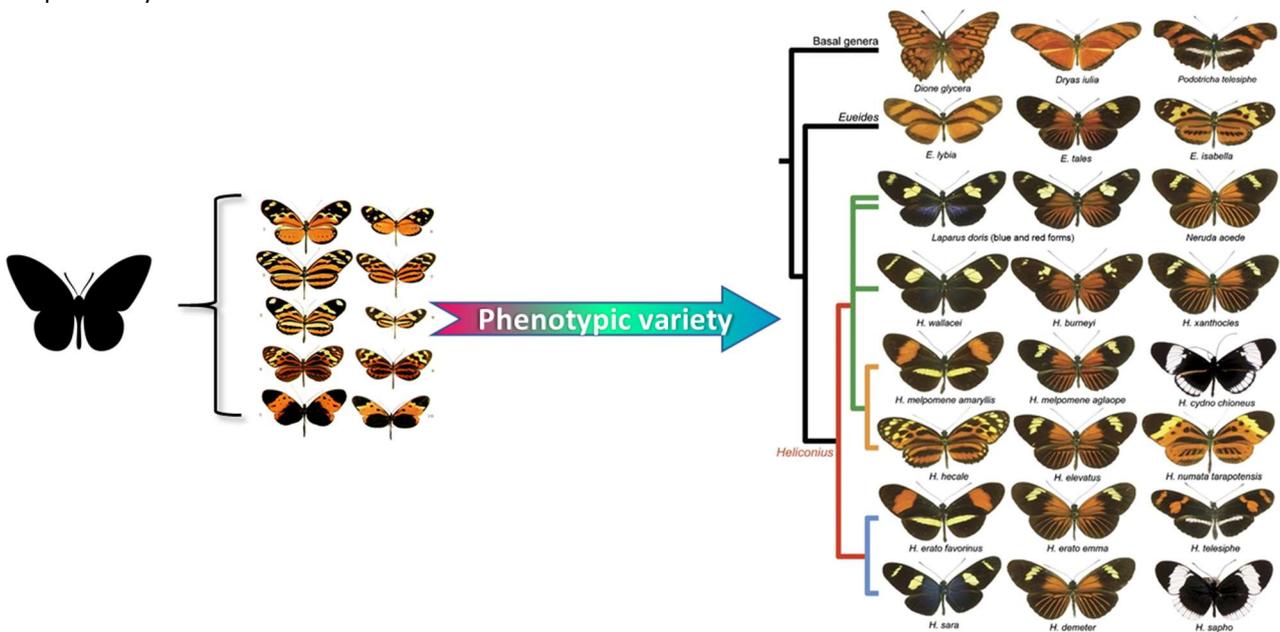


Figure 23 - Polymorphism mechanism showcased with butterfly analogy: example of a generic butterfly and its variety of butterflies with heredity links (source: [170] & [168])

Polymorphism means a multiplicity of morphs, or representations, which depends on phenotypic variety and / or temporal aspects. To explain how the polymorphism mechanism can be applied to quality models, we rely on an analogy with biology, substituting "butterfly" for "quality model". So, to describe the first aspect linked to phenotypic variety, we start with how a butterfly can be commonly perceived. A "generic" butterfly can be

defined based on common characteristics found in any butterfly, such as a trunk, two antennas, two wings covered with colored scales, three pairs of thoracic legs, a body divided into head, thorax and abdomen ... In the real world, this “generic” butterfly does not exist, but there are many different variants of it. And even though they are distinct from each other, all of them inherit from the traits of this “generic” butterfly. In Figure 23 our “generic” butterfly is symbolized by a “black butterfly” and on its immediate right, we see some of the varieties inherited from it. This illustrates one single level of polymorphism; however, we may have multiple levels of polymorphism and this is what the arrow of “phenotypic variety” is pointing out, showing an example of variety of butterflies with heredity links.

The second aspect of polymorphism is a temporal one. Continuing with the “butterfly” analogy, over its life, a butterfly, independently of its variant, evolves from eggs to caterpillar to chrysalid to new-born butterfly and when its wings are dry, to a fully-fledged (operational) butterfly flying away (see Figure 24).



Figure 24 - Polymorphism mechanism showcased with butterfly analogy: example of temporal evolution
(source: [171])

We remark that both phenotypic variety and temporal aspects demonstrate unification and homogenization properties. This, if we now transcribe polymorphism from butterfly to quality modeling, a polymorphic quality model means:

1. **For the similar type of objects (i.e., phenotypic variety aspect)**, referring to Cardelli and Wegner [169] and Cook *et al.* [172] regarding polymorphism, we may have
 - a. **Ad hoc polymorphism** can be understood as apparent common quality model characteristics, or “interface”. Nevertheless, it is defined by two type of behaviors: *overloading* (e.g., same characteristic name but different sub-characteristics or metrics depending on a context) and *coercion* (e.g., forcing a sub-characteristic to abort or to behave like its parent characteristic). For instance, ISO/IEC 9126 [24] or ISO/IEC/IEEE 25010 [23] mark the beginning of ad hoc polymorphism because they are expected to cover a large spectrum of cases and their customization demonstrates coercion and overloading. FURPS [85] and FURPS+ [152], for example, demonstrate ad hoc polymorphism behavior with the overloading of the metrics used.
 - b. **Universal polymorphism** can be understood as variations with heritage between quality models. It is defined by three types of behaviors: *inheritance* (e.g., creating a new quality model from an existing quality model, keeping its characteristic definition and implementation; this is also called sub-classing), *overriding* (e.g., replacing the sub-characteristics or metrics of a characteristic in a new inherited quality model), and *extension* (e.g., a new inherited quality model has some extended new characteristics to cover specific quality objectives). For example, Horgan *et al.* [84] with the Key Quality Factors vs Locally Defined Factors and Khaddaj and Horgan [148] with their adaptable quality model reflecting both ad hoc and universal polymorphism behavior (e.g., overriding of quality sub-characteristics related to a stakeholder view)
2. **For objects over their project or product life cycle (i.e., temporal aspect)**, for example, a quality model can evolve, change (e.g., in the design phase we have a different focus than in the maintenance one). This is the case with the FURPS [85] and FURPS+ [152] models since the measures of characteristics evolve along with the product life cycle stages. McCall's model [41] is another example because of its three software product activities (i.e., product operation, product revision and product transition).

To sum up, we propose the polymorphism mechanism to consolidate qualimetry, and therefore enabling built-in adaptation and evolution in quality modeling. This concept is transparent, unify many existing contributions in quality modeling domains, and foster development, use and reuse of quality model.

Furthermore, it unifies diversity and time evolution in quality modeling by using natural links between similar, or identical, objects and their evolution over the life cycle stages.

c. Quality model distance formula

Research Sub-question 2c

Considering at least two quality models, how to compare together quality models, and can we define a reliable distance formula between quality models?

Continuing the strengthening of qualimetry, our next contribution to quality modeling is devoted to determining a reliable distance formula between quality models, and consequently, addressing the second part of this research sub-question 2c.

A distance formula is certainly a relevant approach to measure how closed quality models are together. Indeed, such tool helps us to compare and classify quality models, to estimate and explain what the impacts and consequences are to change, update or adapt current quality model or to apply one quality model instead of another one. Furthermore, the consequences are directly linked to what we aim to do with quality model. For instance, let say that a company is currently using ISO/IEC 9126 standard and decides to be compliant with latest available standard, which is ISO/IEC/IEEE 25010. Then this distance will help to understand and estimate:

- what the risks linked to such change are, considering not only that “*small distance = low risk*” and therefore “*large distance = high risk*”, but also when a distance is small, change may be discarded, while a large distance reinforces the necessity to apply this change,
- which areas are the most impacted, and where more changes in quality modeling happen, using the distance on each quality characteristic for instance,
- how much work and resource it is going to cost to replace all, or partially, current quality model,
- where quality quantification, assessment, control, and prediction are changing,
- how deep the validation or evaluation path is changed, allowing to capture different types of issues possibly never found before and discarding other areas and paths,

Quality model changes can also occur due to modification or evolution of targeted product, or to the stages in the product life cycle. Thus, this formula can be used to support decision, and to control change or update of quality models, including the case of polymorphism. It fosters split of quality model changes into reasonable change increments, from an agile and risk point of view.

Before deciding which formula dovetails with measuring the distance between quality models, we must distinguish two types of computing distance to explore.

The first type deals with the quantitative aspect of quality models. Thus, the idea is to determine a distance between quality models from the results they give when they are exercised during quality evaluation activities. This is for instance the case of Khoshgotaar *et al.* [173]. With the aim to find which software quality model, among a set of statistic models, has the best cost benefit value, the authors performed a cost benefit analysis based on the quality model results obtained over multiple software releases. So, here the distance formula was the profit evaluation (i.e., “*Profit = [Benefit resulting of quality model use (e.g., early fix of fault prone components)] - [Cost of development and use of quality model]*”). Another example is the case of Juneja *et al.* [77] on software reliability growth models comparison. The quality model distance was the estimated failure rate for a set of reliability growth models, thanks to a simulation against a specific use case. However, in both cases, this type of distance approach requires a non-negligible effort to deploy and use quality model against specific use case set, and the resulting distance values may be consequently neither valid, nor reused for a different use-case set.

The second kind of computing distance examines the qualitative aspect of quality models (e.g., quality characteristics and sub-characteristics). Indeed, if we refer to the ontology shown by Figure 9, a quality model is generally composed of different levels of quality characteristics. In this case, we take the assumption that the

distance formula focuses on these characteristics, often expressed in natural language with words, or character strings. So, the most frequent and well-known word-based distances are:

- Hamming's distance [87] is a widely used distance between two general sequences of symbols. The distance represents the number of symbols which differs between these two sequences (see Annex 2 and Table 40).
- Levenshtein's distance [88] can be considered as a generalization of Hamming's distance but on strings. It measures the difference between two strings by counting the number of edit operations (i.e., deletion, insertion, replacement) required to go from one string to the other (see Annex 2 and Table 41).
- Damerau–Levenshtein's distance [89] is an extension of Levenshtein's distance and is also used in biology as a measure of variations on DNA sequence. This distance measures the difference between two strings by counting the number of transformation operations (i.e., string character edition, with deletion, insertion, and replacement, completed with transposition of two adjacent string characters) required to go from one string to the other. The additional operation (i.e; transposition) allows to compute a distance without being too much impacted by human typo (see Annex 2 and Table 42).
- Jaro's distance [90] is a measure of string similarity which is often used in string duplicate detection (see Annex 2 and Table 43).
- Jaro-Winkler's distance [91] is a variant of Jaro's distance, well adapted for evaluating similarity of short string since it includes a notion of string prefix. Moreover, the Jaro-Winkler distance is normalized and therefore a distance value of 0 means no similarity, while 1 means the two strings are identical (see Annex 2 and Table 44).

An alternate group of approaches to all these sequence-, or string-based distances is the similarity distance over sample set. We can cite as principal example Jaccard's distance [95] (see Annex 2) which is a general sample diversity distance based on sample attributes (i.e., the greater the distance value is, the lowest is the similarity). The idea is to evaluate how much the intersection of the sample sets is covering their union. If the intersection and the union are superposed, then the sample sets are identical. Hence, these string, sequence, or sample set similarity distance formulas are powerful tools to detect matches between any level of quality characteristics, with a certain level of robustness. Unfortunately, they are manipulating quality characteristics only from a character string set point of view, without any attention to lexical or semantic nuance. For example, *productivity* is a synonym of *efficiency*, and thus term relatively closed lexically and semantically. Furthermore, any of the previous distance formula applied to them shall indicate an opposite verdict: mismatch of the two words. Consequently, these distances don't dovetail with a proper distance between quality models.

Apropos semantic and lexical inclusion in distance formula, Motogna *et al.* [94] combined some natural language processing measures to perform comparison of quality models through similarities analysis. The authors noted that each quality model characteristic is defined not only by a set of words (i.e., name of the characteristic), but also a description and a set of sub-characteristics themselves defined by a set of words (i.e., name of the sub-characteristic) and a description. So, in their analysis, the authors determined the matching characteristics of two quality models by taking the average of characteristic word set similarities (e.g., with Jaccard' similarity distance) and lexical characteristic similarities (e.g., Mihalcea *et al.*'s "*corpus-based and knowledge-based measures of text semantic similarity*" [174]). To be more precise, the lexical similarity evaluation collected the semantic equivalent list of words that matches the characteristic related word stems thank to WordNet lexical database [175] - available online at Princeton University "About WordNet"¹¹. Nevertheless, their methodology didn't allow to determine a global distance between two quality models but rather locally to characteristics whenever a similarity can be calculated.

Another relevant contribution on semantic distance for quality model comparison is the research work done by Gordieiev *et al.*'s [92], [93]. They compared four quality models against a reference quality model, ISO/IEC/IEEE 25010, by using a cumulative semantic matching-based distance (cf equation 4). The authors decomposed every quality model into a set of model elements (i.e., characteristics and sub-characteristics) and a set of relationship between model element together. Then, the distance calculation is an aggregation of the semantic match result of each quality characteristic, and their sub-characteristics, with the ISO/IEC/IEEE 25010 ones. For example,

¹¹ WordNet Princeton University 2010: <https://wordnet.princeton.edu/>

Gordieiev *et al.* determined that the distance between ISO/IEC/IEEE 25010 and ISO/IEC 9126 gives 73.75% of similarity.

With this latest research work, we have a real distance for quality models. It handles quality model structure, relationship between its elements (i.e., characteristics and sub-characteristics) and semantic nuances between them. Nonetheless, there are few flaws that should be addressed. For instance, the authors assumed that the weight of characteristic similarities and the one of sub-characteristics similarities are equal. From our point of view, and from Motogna *et al.* one too, this is not a correct assumption because it is not taking into account the fact that a characteristic is already defined by its sub-characteristics, and therefore the characteristic similarity evaluation has to include that impact. The distance formula should be also more generic (e.g., distance between any two quality models).

$$d_{gordieiev} = \sum_{i=1}^m \left(CMM_i + \sum_{j=1}^{n_i} SMM_{i,j} \right) \quad (4)$$

where m is the number of quality model characteristics

n_i is the number of subcharacteristics of the i^{th} characteristics

CMM_i is the Characteristics Matching Metric of the i^{th} characteristics

$$CMM_i = \begin{cases} 0.5 & \text{if the } i^{th} \text{ characteristic matches semantically with ISO/IEC/IEEE 25010 characteristic} \\ 0 & \text{otherwise} \end{cases}$$

$SMM_{i,j}$ is the Subcharacteristics Matching Metric of the j^{th} subcharacteristics of the i^{th} characteristics

$$SMM_{i,j} = \begin{cases} \frac{0.5}{n_i} & \text{if the } j^{th} \text{ subcharacteristic matches semantically with ISO/IEC/IEEE 25010 subcharacteristic} \\ 0 & \text{otherwise} \end{cases}$$

However, we remark that all the distances described above are mainly dealing with similarity and not with variety to compare together quality model characteristics.

To handle properly variety, especially with a large set of quality models, we must have a statistical notion about frequency or presence of quality characteristics in quality model, for example. This is not the case of any of the above distances. Hopefully, with polymorphism in genetic, there is a well-known formula to compute the degree of nucleotide diversity, also known as the degree of polymorphism. This is the Nei and Li formula [86] (1979) depicted by equation 5.

This formula is made for DNA sequences from alleles and linked with the existence of single nucleotide polymorphism (i.e., variation of nucleotide at a specific DNA location for more than 1% of a population) [176]. Nevertheless, we can see an analogy between DNA sequences and quality characteristics sequences both of them coding behaviors (cf. Chapter VIII.2). So, in our case the π_{ij} represents the proportion of lexically and semantically different quality model elements (e.g., characteristics, sub-characteristics, attributes), and the x_i the frequency of the i^{th} quality model or characteristic sequence. In a case of quality model sequence, for instance, that frequency indicates the likelihood to find a quality model among the other models that are part of our sample set. In the same way for characteristic sequences, that frequency indicates the likelihood to find a characteristic among the other characteristics that are part of our sample set. For instance, if a characteristic recurrently appears in quality models, its probability is 1. If half of the time the characteristic is present and the other half is another close (i.e., not disjoint) characteristic, then their respective probabilities are 0.5.

$$\pi = \sum_{ij} x_i x_j \pi_{ij} \quad (5)$$

where π_{ij} proportion of different nucleotides between sequences i and j

x_i is the estimated frequency of the i^{th} sequence in the population

x_j is the estimated frequency of the j^{th} sequence in the population

About lexical and semantic match, we consider three cases, identical, similar, or gaps (i.e., disjoint) and therefore the calculation of the corresponding π_j is given by equation 6. Moreover, we use the unbiased estimate of π (cf. equation 7) to get normed distance result (i.e., between 0 and 1).

At last, to validate that our contribution to quality model distance formula is practicable and give right result, we decided to apply it against ISO/IEC 9126 and ISO/IEC/IEEE 25010 because we have a reference result with Gordieiev *et al.* So, we assume that we have two quality model sequences and the frequency of each sequence are equal (i.e., 50 % for ISO/IEC 9126 and 50 % for ISO/IEC/IEEE 25010). We identified: 52 distinct sub-characteristics, 31 unique, 8 similar (i.e., close but not identical: for instance, "Modifiability" versus "Changeability"), and 13 identical. This lands us to calculate, with equations 6 and 7, a diversity degree of 67.31% (cf. Annex 3 for the computation details) which also indicates that ISO/IEC 9126 and ISO/IEC/IEEE 25010 are similar¹² at 32.69%. We remark that distance result differs from Gordieiev *et al.* who found 73.75% of similarity between these two models. The main reason why the results diverge is because of Gordieiev *et al.* assumed that weight of characteristic similarities and the one of sub-characteristics similarities are equal, which is not our assumption. The example in Table 12, Table 13 and Figure 25, proves us right: for instance "Maintainability" characteristic is present on both models but their respective sub-characteristics are identical only at 28.57 %. So, taking the assumption that the characteristics are already identical at 50% (i.e., weight of characteristic with regard to its sub-characteristics) conducts to an incorrect result.

$$\pi_{ij} = \sum_{k=1}^n d_{match}(seq_{i_k}, seq_{j_k}) \tag{6}$$

where $n = \text{card}(seq_i \cup seq_j)$

$$d_{matc}(seq_{i_k}, seq_{j_k}) = \begin{cases} 0 & \text{if } seq_{i_k} \text{ and } seq_{j_k} \text{ are lexically and semantically equal} \\ \frac{0.5}{n} & \text{if } seq_{i_k} \text{ and } seq_{j_k} \text{ are lexically and semantically close but not equal} \\ \frac{1}{n} & \text{otherwise} \end{cases}$$

where seq_i is the i^{th} sequence and seq_{i_k} is the k^{th} element in that sequence

and seq_j is the j^{th} sequence and seq_{j_k} is the k^{th} element in that sequence

$$\hat{\pi} = \frac{n}{(n-1)} \sum_{ij} x_i x_j \pi_{ij} \tag{7}$$

To conclude on the quality model distance, our contribution reuses a proofed formula directly associated to polymorphism, or variety concept. Through is application against the two consecutive standards of software quality model, we assess that our formula is applicable and gives accurate results. We remark finally that our approach is close to Hamming's distance (i.e., number of symbols which differs between these two sequences), Gordieiev *et al.*'s distance (i.e., semantic comparison on characteristics and sub-characteristics) and Motogna *et al.* contribution (i.e., inclusion of description and sub-characteristics to define characteristics).

TABLE 12 - EXAMPLE OF LEXICAL AND SEMANTIC COMPARISON RESULT BETWEEN TWO CHARACTERISTICS FROM ISO/IEC 9126 AND ISO/IEC/IEEE 25010

ISO/IEC/IEEE 25010 (Sequence 1)	ISO/IEC 9126 (Sequence 2)	
Functional suitability		
Functional completeness	Functionality Compliance	Similar
Functional correctness	Accuracy	Similar
Functional appropriateness	Suitability	Similar
-	Security	Gap
-	Interoperability	Gap

¹² Comparison between these two quality models are also available in Annex A of ISO/IEC/IEEE 25010 [23]

Maintainability		
Modularity	-	Gap
Reusability	-	Gap
Analysability	Analysability	Identical
Modifiability	Changeability	Similar
Testability	Testability	Identical
-	Maintainability Compliance	Gap
-	Stability	Gap

TABLE 13 - INTERMEDIATE CALCULATION LINKED TO PREVIOUS EXAMPLE

	Gap	Similar	Identical	Total	π_{ij}
Functional suitability	2	3	0	5	0.7000
Maintainability	4	1	2	7	0.6429
SUM	6	4	2	12	0.6667

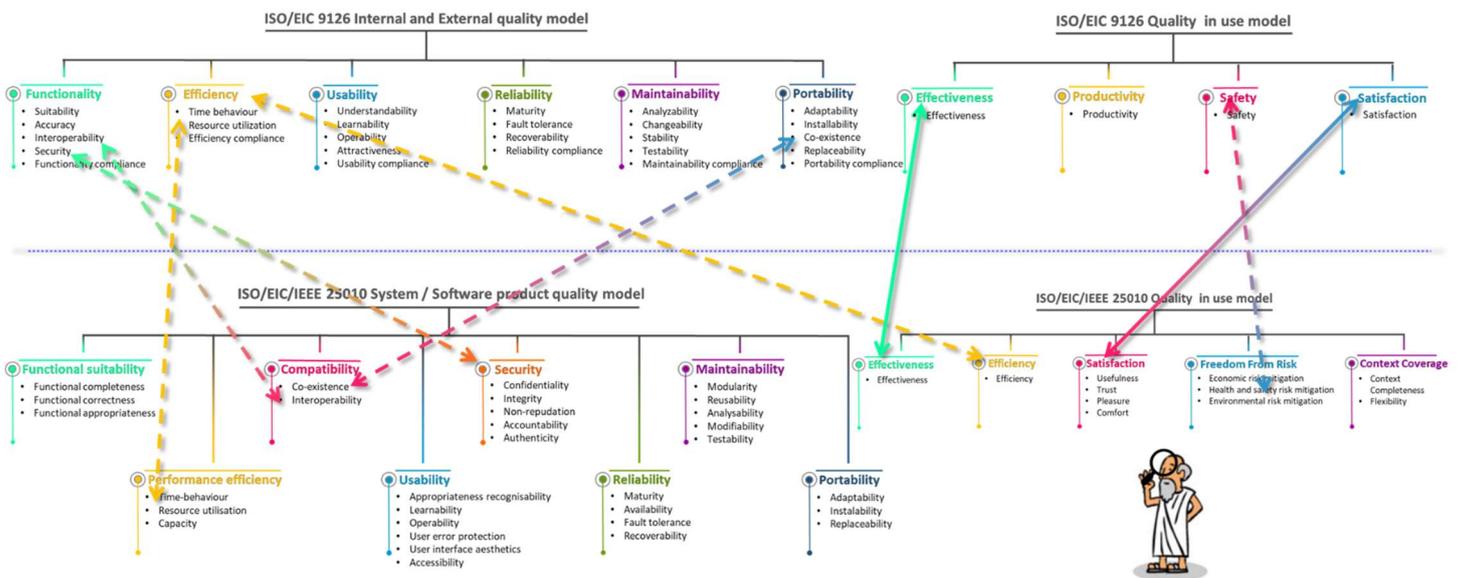


Figure 25 - Example of some differences between ISO/IEC 9126 & ISO/IEC/IEEE 25010

d. Measurement process

As we detailed with qualimetry in sections Chapter IV.5 and Chapter IV.6.a, a proper quality model is one side of the quality quantification problem. The other side concerns the measurement of the quality characteristics and especially all measurement process activities.

Indeed, the aim of a measurement process is not only to proceed on, or collect, measure but also to record and analyze the results, control quality, help on decision making, including doing some predictions and communicating the results to the right stakeholders. ISO/IEC 25040 standard [147] defines a coarse and linear evaluation process (see Figure 26) which define the main tasks that must be achieved for measurement, beginning by establishing the evaluation requirements and ending with the conclusion of the evaluation. This standard contains the same issue that we can detect with the rest of ISO/IEC 250nn standard series: it is not precise enough, willing to cover all case for computer system and software quality evaluation, and consequently, requires interpretation and strong complement.

Hopefully we may rely on practical work carried out for software related decision makers by, McGarry *et al.* [157] (see Figure 27) which introduces a process articulated around four activities: establish and sustain commitment, plan measurement, perform measurement and evaluate measurement. We note that this approach has been adopted by INCOSE, for example, for the Systems Engineering measurement primer and the technical measurement guide [177]. We can also find the measurement process introduction carried out by Miller *et al.* [178], which has a scope of systems engineering and the process published by Dekkers *et al.* [179], a US-CERT team on secure software development.

Moreover, we recall that in automobile – our current industrial context -, A-SPICE [21] guidelines points out a measurement process called MAN.6, part of management processes, but don't provide it. Instead, A-SPICE enumerate some recommendations linked to measurement process such as the type of process outputs for measures definition and collection, data and reports, some categories of measures to collect (i.e., product quality, field, project, risk, service level, process, personnel performance) and some base practices.



Figure 26 - Software product quality evaluation process defined by ISO/IEC 25040 [147]

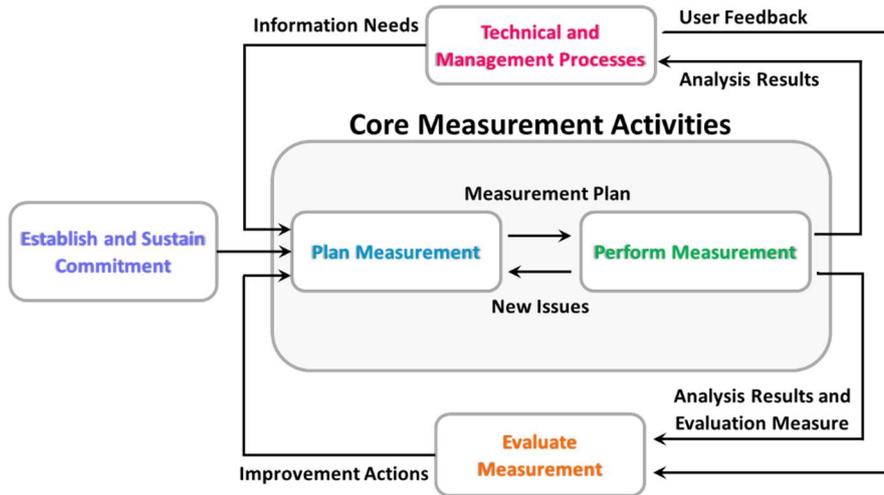


Figure 27 - Measurement process model of McGarry *et al.* [157]

Thus, based on all these measurement processes, and in order to be aligned with company requirements, we consolidate a measurement process based on A-SPICE MAN.6 process guidelines, McGarry *et al.* measurement process mode with regards to the core measurement activities, ISO/IEC 25040 for evaluation plan, inclusion of quality model and polymorphism in quality modeling which requires to be cadenced by the systems or software development life cycle. Our proposal of measurement process (cf. Figure 28) is therefore articulated into three sequential phases: Initial, Planning and Execution.

i. Initial phase

The purpose of this phase is to understand, identify and collect both requirements and context linked to measurement goals and activities. That phase is performed over three tasks which can be realized in parallel. The first task focuses on the identification and enumeration of all measurement objectives, taking measurement requirements as inputs. The second task is dedicated to the measurement context definition which can be understood as defining the scope, the boundaries, the dependencies, and the environment linked to measurement activities. The last task of the initial phase relates to process improvement. In the first iteration of these three phases, we may not yet have any lessons learned or post-mortem data from previous measurement activities to take into account, however, with time, we will be able to integrate this data in order to improve our current process. The different outputs of these three tasks will be merged and used as inputs to the second phase which is planning.

ii. Planning phase

During this phase, we transform the requirements, context and process improvement into an evaluation plan (see Annex 4), criteria and statistical and/or qualitative techniques to be ready for the execution of that plan. Since that plan must be aligned with the system development life cycle [13], [180], this one is also one input of the planning phase tasks. So, we start to transform measurement requirements and context into the quality model and measurement specifications. Once this has been done, we must plan for their treatment. First by planning for their collection and storage, where processes change, tools and training may be required, then for

their analysis procedure and criteria, or thresholds, to apply assessment, control, and prediction. The final task of this phase is the synthesis and organization of all outputs from these three previous tasks into one critical document: the evaluation plan.

iii. Execution phase

The last phase of our process corresponds to the execution of our evaluation plan which is aligned with the system development life cycle phase. The main task here is a loop to collect measurement data at the frequencies defined in the evaluation plan. Each time data is collected, it needs to be stored as well as analyzed and assessed. The results, containing analysis synthesis, predictions, recommendations, and conclusions, are generated under various forms -graphical dashboards, analyst summary, detailed results, and reports- which are then communicated to the stakeholders, for example, development teams, program managers and any key decision makers.

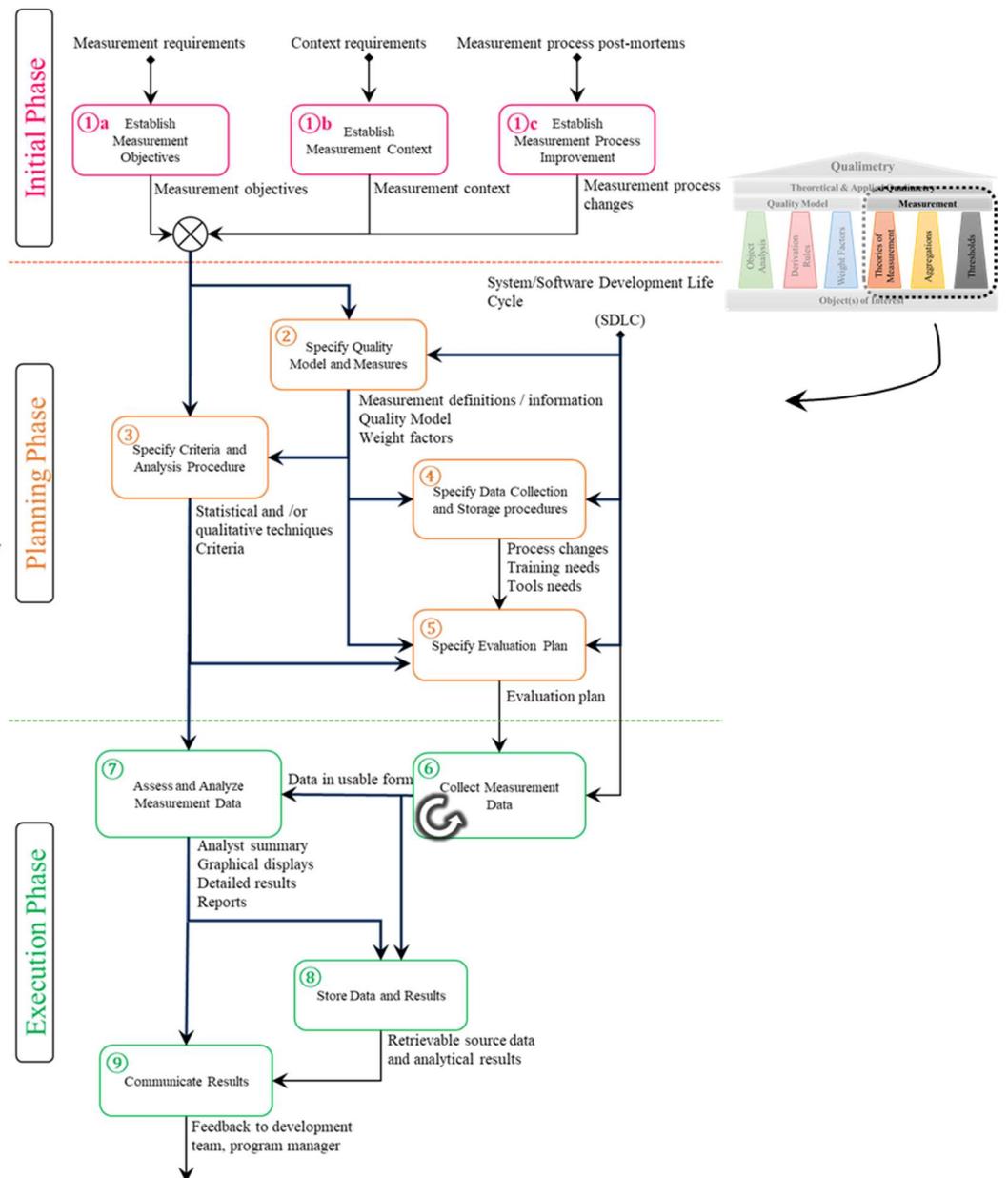
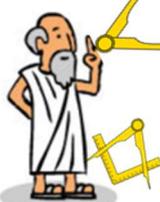


Figure 28 - Our measurement process proposal articulated over three phases and cadenced with SDLC

To conclude on this new process definition achievement, if we do an analogy, for instance, between ISO/IEC 25040 process (cf. Figure 26) and our proposal (cf. Figure 28) we can clearly conclude that the standard process is a sub-part of our proposed process. Indeed, "establish the evaluation requirements" is a subset of our initial



phase, both “*specify the evaluation*” and “*design the evaluation*” are included into our planning phase, and “*execute the evaluation*” with “*conclude the evaluation*” are also a subset of the tasks of our execution phase.

If we compare with McGarry *et al.* process (cf. Figure 27), our process corresponds to the “*core measurement activities*”. Our initial phase is located just before the “*plan measurement*” of McGarry *et al.* process, collecting requirements, goals, and improvement actions from the “*technical and management processes*”, “*establish and sustain commitment*” and “*evaluate measurement*”. The planning and execution phase are aligned respectively to the “*plan measurement*” and “*perform measurement*” activities.

So, our measurement process is directly compatible with these two well-known measurement processes.

Furthermore, our contribution takes benefit of qualimetry, putting quality modeling activity at the center of the measurement strategy, and remind that this process must be cadenced, like a processor, with systems or software development life cycle, impacting measurement tasks (e.g., change of measurement context, objectives and requirements; see FURPS example from TABLE 11.) and critical for handling accurately polymorphism behavior.

7- Threats to validity and discussions

Over this chapter we have deep dived into the concepts of quality, quality modeling and qualimetry, with the purpose to understand them enough to assess which research and technology approaches are the optimum one to answer to our main problematic: have a unified, operational, and appropriate way to define, assess, control, or predict quality of embedded software. However, there are several threats that may affect the validity of our analysis and contributions.

So, in the concept exploration, our intention was to clarify these three concepts and their main vocabulary, ensuring to have a strong grasp on them rather to be exhaustive and lost ourselves in tiny details. We rely also on standards as much as possible because they are reflecting international discussions and agreements between academic and industrial experts. And in automobile domain, for example, it is critical to rely on standards. In addition, we made the choice of focusing on quality modeling and qualimetry instead of considering only applied aspect of quality through the current usage of continuous, automated, and agile process, all of them often neglect quality requirements, related to quality characteristics (see ontology of Figure 9), in favor to fast delivery [181], for example. The reason behind this choice is that only quality modeling has the strength to structure, homogenize and get control on the quality, especially when dealing with a diversity of product composed of embedded software and systems.

On the key contributions to quality modeling of software, we didn’t perform a systematic literature review to retrieve these key contributions. Rather, we took advantage of the knowledge that we incrementally built during this thesis thought an exploratory literature review, on quality modeling applied to embedded software (see Chapter II.3), and a systematic literature review, on software quality model (see next 0.2). By consequence, we didn’t subjectively limit the number of key contributions, but took the research contributions in quality modeling that we noticed to be the most disruptive ones, or the most recurrent and cited ones. However, we have hesitated about the inclusion of some contributions since we had to deal with more than 50 years of continuous research works. For instance, Boehm’s COCOMO [151] is often referred in quality modeling contributions. Nevertheless, COCOMO is a cost model and not a quality model. Thus, its inclusion is not appropriate and could generate some confusion causing misinterpretation of the concepts. Another example is COSMIC [158]. This functional size measurement standard can be used to improve quality of requirements, or in the case of non-functional requirements¹³ which can be declined into functional requirements, they can be measured with COSMIC [182]. Nonetheless, we decided to exclude COSMIC from our current list of key contributions because it is independent to technology and quality considerations.

Once this list of key contributions achieved, we summarized them through a graphical timeline (cf. Figure 19) which is an interesting medium for knowledge sharing. In fact, we used graphical medium to leverage some complex concept and ensure people remind the principal information. This is the case for “*The House of*

¹³ Often associated to quality characteristics

Qualimetry” (cf. Figure 20) for the science of quality quantification, and for polymorphism with the butterfly analogy (cf. Figure 23, Figure 24).

Concerning the derivation rules from qualimetry, we cited only the most important ones that are part of the ABC of qualimetry book [113], but unfortunately, we were not able to retrieve the full set of 30 derivation rules pointed out by the authors. So, we trust the authors about the importance of these rules with regard to the missing ones, but potentially there is some specific ones that we may miss, and which may impact our case for quality modeling of embedded software.

On the scale topic, even if Stevens’ scale is well known and still widely used, it is subject to some criticism, such as Velleman et al. [140] claiming that “*Good data analysis does not assume data types*” and therefore scale type to be used must follow the analysis. In other words, we must know what we are planning to do first and how to use what we are planning to collect in order to select right scale and data type (for property, attribute or object). Moreover, some extension of Steven’ scale exists. We can cite for example Chrisman [142] who proposed an extension of the original Stevens’ scale list in 1998. That extended typology is interesting since it is not diverging from Stevens’ one and includes therefore 6 new scale types that are useful for some particular cases (e.g., gradation of membership is key in fuzzy set theory, log interval is widely use in stock market graphics, cyclical ratio is suitable for times and angles ...).

Regarding our quality model distance formula based on Nei and Li’s formula [86], we assumed implicitly that our distance, described by equations 6 and 7, is a metric from a mathematical point of view. To demonstrate this fact, a distance is a metric (i.e., in the metric space) if it respects the mathematical properties shown in 8, 9, 10 and 11 [183]. The non-negativity (8) is due to the fact that both equation 6 and 7 are positive equation (i.e., cannot provide any negative results). The identity of indiscernible (9) is directly derived from equation 7 which gives 0 only if x and y semantically match. Symmetry property (10) is verified since both matching and summation, composing equation 6 and 7 are commutative operations. About the subadditivity (11), we won’t demonstrate it, but our distance formula satisfies also that property. By satisfying these properties we can claim that our distance is a metrics in a mathematical sense. Conversely, Gordieiev *et al.* distance, equation 4, is not a metric. For instance, $d_{gordieiev}(x, x) \neq 0$ which invalid identity of indiscernible property.

on a set X , d is a function such: $d: X \times X \rightarrow [0, +\infty[\quad \forall x, y, z \in X$

$$d(x, y) \geq 0 \quad \text{Non – negativity} \quad (8)$$

$$d(x, y) = 0 \Leftrightarrow x = y \quad \text{Identity of indiscernible} \quad (9)$$

$$d(x, y) = d(y, x) \quad \text{Symmetry} \quad (10)$$

$$d(x, y) \leq d(x, z) \perp d(z, y) \quad \text{Subadditivity or triangle inequality} \quad (11)$$

Lastly, we would like to remark that two of our contributions get their inspiration from nature, and more precisely from genetic: this is the polymorphism concept and degree of polymorphism. The polymorphism is present also in software engineering (i.e., in object-oriented programming) but with the genetic analogy we were able to go further on that concept, and even explain it easier. For our last contribution on measurement process, we didn’t aim to replace existing measurement processes, but rather clarify and adapt it to foster polymorphic quality model development and usage. As we noticed too, our process can fit in the existing process ones, such as McGarry *et al.*’s measurement process model [157] (see Figure 27).

Chapter V. Quality Model Classification and Selection

1. Introduction

Following previous Chapter IV, we understand that quality model is a pivot concept in software quality quantification. Unfortunately, it appears also that there is an abundance of software quality models (e.g., [10]–[12]) while there is no obvious consolidated reference list of such models (see Chapter II) from which we could select the most appropriate model to solve our problematic about quality modeling for embedded software. In addition, identifying the most appropriate model among a pool of models requires to classify them and define a set of criteria for decision. This classification gives the hints as well to initiate the resolution of the research question 4 (i.e., “Can we have a unique reference quality model for software product?”).

Thus, these dilemmas can be summarized via the research question 2 and 4a:

Research Question 2

Considering the set of software quality models, how to identify and decide which quality model is the most suitable for embedded software?

Research Sub-question 4a

Is it possible to have a unique reference quality model for software product, or instead should we have a meta-model?

This chapter is organized around the refinement of research question 2 as followed.

As we highlighted in Chapter I and Chapter III, our research strategy aims to take advantage of existing quality models, selecting the most suitable quality model, if any, for automotive embedded software, instead of creating a new model. So, considering software scope, which encompasses embedded software one, we must retrieve and enumerate the published software quality models thanks to a systematic literature review, addressing consequently our research sub-question 2a: “considering software scope, what is the set of existing quality models?”

Once this enumeration done, we must classify these software quality models, preparing the set of models to be used for quality model selection purpose, for instance. So, the classification requires to identify the right methodology and criteria which we rephrase into the research sub-question 2b: “Considering a set of quality models, how to classify these quality models, what are the methodology, the criteria, and the characteristics to use?”.

Finally, the classified collection of software quality model together with the classification criteria foster the elaboration of a quality model landscape complementary to Kläs *et al.* one [10], expanding the knowledge on the current research contributions done on software quality models, and supporting the selection of the most appropriate quality model for embedded software in automotive (i.e., research sub-question 2d).

2. Systematic literature review on software quality model

Research Sub-question 2a

Considering software scope, what is the set of existing quality models?

During our early investigation on quality modeling and model, we noticed a plethora of existing systematic mappings or reviews with regards to the domain of software quality model. To name a few, we have for example, Kläs *et al.* [10] (2009) CQML classification scheme for comprehensive quality model landscape, R. S. Jamwal and D. Jamwal [184] (2009) with their 9-steps approach to evaluate software quality models, the comparative studies on software quality models done by AL-Badareen *et al.* [185] (2011) and Ahmad *et al.* [186] (2013), Thapar *et al.* [11] (2012) on the challenges linked to the development of standard software quality models, Polillo [187] (2012) about quality models for web 2.0 sites, the reviews of Adewumi *et al.* [188] (2013) and [189] (2016) on software quality models within the context of open source software, Oriol *et al.* [12] (2014) mapping of quality models for web-services, Miguel *et al.* [190] (2014) review on software product quality models, Gordieiev *et al.* [92] (2014) with regard to software quality model evolution against ISO/IEC/IEEE 25010 and their study [191] (2018) on prevailing quality characteristics for IT-oriented software quality models, Buglione [192] (2015) investigation on quality model evolution and perspectives, Motogna *et al.* [94] (2019) software quality models evaluation based

on natural language processing, and the systematic mapping study on software quality assessment model done by Yan *et al.* [193] (2019).

In addition, we notice that the studies related to software quality models rely in general on small sets of up to ten quality models (e.g., [92], [94], [184]–[187], [190], [192], [194]) with some minor exceptions: Kläs *et al.* [10] classified 22 quality models, from a publication range from 1978 to 2009, Thapar *et al.* [11] analyzed a reference set of 24 quality models from 1977 to 2011, Oriol *et al.* [12] collected and exploited 51 quality models published from 2002 to 2012, and Yan *et al.* [193] found 31 quality models that were published between 1998 and 2015. Note, the quality model sets from these last four studies are partially overlapped and cover a publication period going from 1977 to 2015.

Consequently, we can state that current available quality model collections aren't unified and should be consolidated at least for the published ones.

To build such consolidated list, we need also to understand how quality models are identified, compared, and classified, if there is a core subset of quality models, or rather a unique reference, that could be considered as basis for any related work on quality models, and what motivates their creation.

So, we reformulate and synthesize our goals into 3 research questions (i.e., *RQ*) with their rationale:

- ***RQ1 - Considering software scope, what is the set of existing quality models? (i.e., research sub-question 2a)***
Coming-up with a consolidated list of the main quality models in the software domain could serve as the reference starting quality model source of information, fostering academic or industrial survey work related to quality modeling. This list must be achieved without any time limitations on published year, and once the first iteration is achieved, it should be further maintained and completed to stay valuable.
- ***RQ2 - Considering a set of quality models, how to classify these quality models, what are the methodology, the criteria, and the characteristics to use? (i.e., research sub-question 2b)***
Identifying quality model classification and comparison methodologies, categories, and criteria teach us not only how to handle collection of quality models, but also how quality models are chosen and compared by our peers. Moreover, answering to this question is fundamental to have an unambiguous characterization of quality models which is mandatory to proceed properly on any type of operations (e.g., design, update, comparison, assessment) with or on them.
- ***RQ3 - Is it possible to have a unique reference quality model for software product? (i.e., first part of research sub-question 4a)***
We already know that many quality models exist, nevertheless we don't know if they are finally closed to a unique reference quality model for software product. Knowing if such unique reference exists is key to frame efficiently any quality modeling activities, and the classified quality model collection is the proper material requires to answer to that question.

In order to address these three research questions, we conduct a systematic literature review following Kitchenham and Charter's systematic literature review guidelines [31]. The systematic literature review process and its stages are detailed in Figure 29. Overall, we apply the same search strategy than Adewumi *et al.* [189] and Yan *et al.* [193], both based on Kitchenham and Charter's guidelines too. Likewise, we rely on the same five digital libraries, or electronic database sources (cf. Table 14) than Yan. *et al.* rather than Kitchenham and Charter recommendation ones, even if the focus of both is on software engineering. We made this choice because of the following five criteria:

- (1) source is a recognized and relevant reference in software engineering domain with a large covering period,
- (2) source is a collection of relevant conference papers, books, and journals,
- (3) their online search engine accepts extended search query,
- (4) query results are freely downloadable from our institution,
- (5) referenced documents are downloadable from the source (i.e., source is not only an aggregator engine).

TABLE 14 - LIST OF ELECTRONIC DATABASE SOURCES

Online source name	Online location
ACM digital library	dl.acm.org
IEEE Xplore	ieeexplore.ieee.org
Scopus	www.scopus.com
Springer	link.springer.com
Web of Science (also known as ISI Web of Knowledge)	webofknowledge.com

The electronics database source exploration and search are realized *via* search queries. These search queries, also named search strings, are composed of words to look for, combined with operators (e.g., OR, AND, NEAR, NOT), sometimes with some wildcard characters (i.e., "?" replaces one alphanumeric character, or "*" replaces zero to multiple alphanumeric characters) and often with indication to fields of narrow search interest in the database (e.g., title, abstract, keywords, date, author). Moreover, each online database search engine has its own query syntax and restriction (e.g., restriction on the number of allowed wildcards in search query). For instance, to search over title and abstract field, with *Scopus* we must use "**TITLE-ABS ()**" while with *ACM digital library*, we must use "**Title:**" and "**Abstract:**".

So, to create the right search strings from the research questions, independently to any search engine syntax, we structure our "words" around three main concepts: domain of interest, object of interest and type of research work. In this survey the domain of interest is the "*software*" domain. The objects of interest are the objects closed or associated to "*quality model*" concept and therefore we can find "*quality model*", "*quality factor*", "*quality characteristic*" and "*quality ontology*". For the type of research work, our focus must be aligned to the survey concept since we are performing a systematic literature review. We identified then "*survey*", "*study*", "*analysis*", "*review*", "*mapping*", "*comparison*", "*challenge*", "*evolution*" or "*taxonomy*". In addition to these words, we completed that list with some frequent acronyms (e.g., quality model as QM), plural forms and use of proper wildcards. The right search queries for each online database source can be then derived from the information summarized in Table 15 and are given in Table 16.

TABLE 15 - SEARCH QUERY ELEMENTS: KEYWORDS VS. OPERATORS VS. SEARCH FIELDS

Concept	Keywords	Operator	Fields
Domain of interest	software*	OR	Any
	SW		
		AND	
Object of interest	quality model*	OR	Any
	QM		
	quality ontology		
	quality ontologies		
	QO		
	quality factor*		
	QF		
quality characteri**			
		AND	
Type of research work	analysis	OR	Title Keywords
	challenge		
	challenges		
	classification		
	classifications		
	compar*		
	evolution		
	evolutions		
	mapping		
	mappings		
	taxonomy		
	review		
	reviews		
study			
studies			
survey			

TABLE 16 – THE FIVE DIGITAL LIBRARY SEARCH QUERIES

Online source name	Search query
ACM digital library	("software*" SW) AND ("quality model*" QM "quality ontology" "quality ontologies" QO "quality factor*" QF "quality characteri*" QC) AND acmdlTitle:(survey surveys study studies analysis review reviews "compar*" classification classifications evolution evolutions taxonomy challenge challenges mapping mappings)
IEEE Xplore	((("software*" OR SW) AND ("quality model*" OR QM OR "quality ontology" OR "quality ontologies" OR QO OR "quality factor*" OR "quality characteri*") AND (("Document Title": "survey") OR ("Document Title": "surveys") OR ("Document Title": "study") OR ("Document Title": "studies") OR ("Document Title": "analysis") OR ("Document Title": "review") OR ("Document Title": "reviews") OR ("Document Title": "compar*") OR ("Document Title": "classification") OR ("Document Title": "classifications") OR ("Document Title": "evolution") OR ("Document Title": "evolutions") OR ("Document Title": "taxonomy") OR ("Document Title": "challenge") OR ("Document Title": "challenges") OR ("Document Title": "mapping") OR ("Document Title": "mappings"))))
Scopus	(TITLE-ABS-KEY (("software*" OR sw)) AND TITLE-ABS-KEY (("quality model*" OR qm OR "quality ontology" OR "quality ontologies" OR qo OR "quality factor*" OR qf OR "quality characteri*")) AND TITLE ((survey OR surveys OR study OR studies OR analysis OR review OR reviews OR compar* OR classification OR classifications OR evolution OR evolutions OR taxonomy OR challenge OR challenges OR mapping OR mappings))) AND (LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "ENGI")) AND (LIMIT-TO (LANGUAGE , "English"))
Springer	("software*" OR SW) AND ("quality model*" OR QM OR "quality ontology" OR "quality ontologies" OR QO OR "quality factor*" OR QF OR "quality characteri*" OR QC) AND (("Document Title": "survey") ("Document Title": "surveys") OR ("Document Title": "study") OR ("Document Title": "studies") OR ("Document Title": "analysis") OR ("Document Title": "review") OR ("Document Title": "reviews") OR ("Document Title": "compar*") OR ("Document Title": "classification") OR ("Document Title": "classifications") OR ("Document Title": "evolution") OR ("Document Title": "evolutions") OR ("Document Title": "taxonomy") OR ("Document Title": "challenge") OR ("Document Title": "challenges") OR ("Document Title": "mapping") OR ("Document Title": "mappings"))
Web of Science	(ALL=(SW or software) AND TS=("quality model*" OR QM OR "quality ontology" OR "quality ontologies" OR QO OR "quality factor*" OR QF OR "quality characteri*") AND TI= (survey OR surveys OR study OR studies OR analysis OR review OR reviews OR classification OR classifications OR evolution OR evolutions OR "compar*" OR taxonomy OR challenge OR challenges OR mapping OR mappings)) AND LANGUAGE: (English)

Thus, once any search query is performed over one of the sources main electronic databases, we obtain a raw result of candidate documents. That raw result contains both relevant and irrelevant documents with our research questions, and therefore must be screened through a process of selection. Our process of study selection, shown in Figure 29, is a sequence of four filtering stages during which we assess the relevance of each collected documents with respect to our research questions. In addition, the document relevance done in each filtering stage relies on a set of inclusion and exclusion criteria, defined in Table 17, and like Oriol *et al.* [12], we are setting minimum document quality by assessing the ranking level of the corresponding journal or conferences.

TABLE 17 - INCLUSION AND EXCLUSION CRITERIA RELATED TO OUR SEARCH STRATEGY

Inclusion criteria	Exclusion criteria
Document focus is on software domain,	Sources of quality model citation are not provided
Document is referencing multiple quality models	Document is not only citing, enumerating quality models
Document contains a study, analysis, survey, or comparison on quality models, with a minimum or argumentation	Document is not a poster, cover, course, conference, or paper review
Document contains some description and details on enumerated quality models	Document is not a retracted publication
Document is from ranked ¹⁴ conferences, or journals	Full document is not accessible or downloadable
	Document is not in English

¹⁴ Ranking is obtained via Core Computing Research & Education Portal: <http://www.core.edu.au/conference-portal>

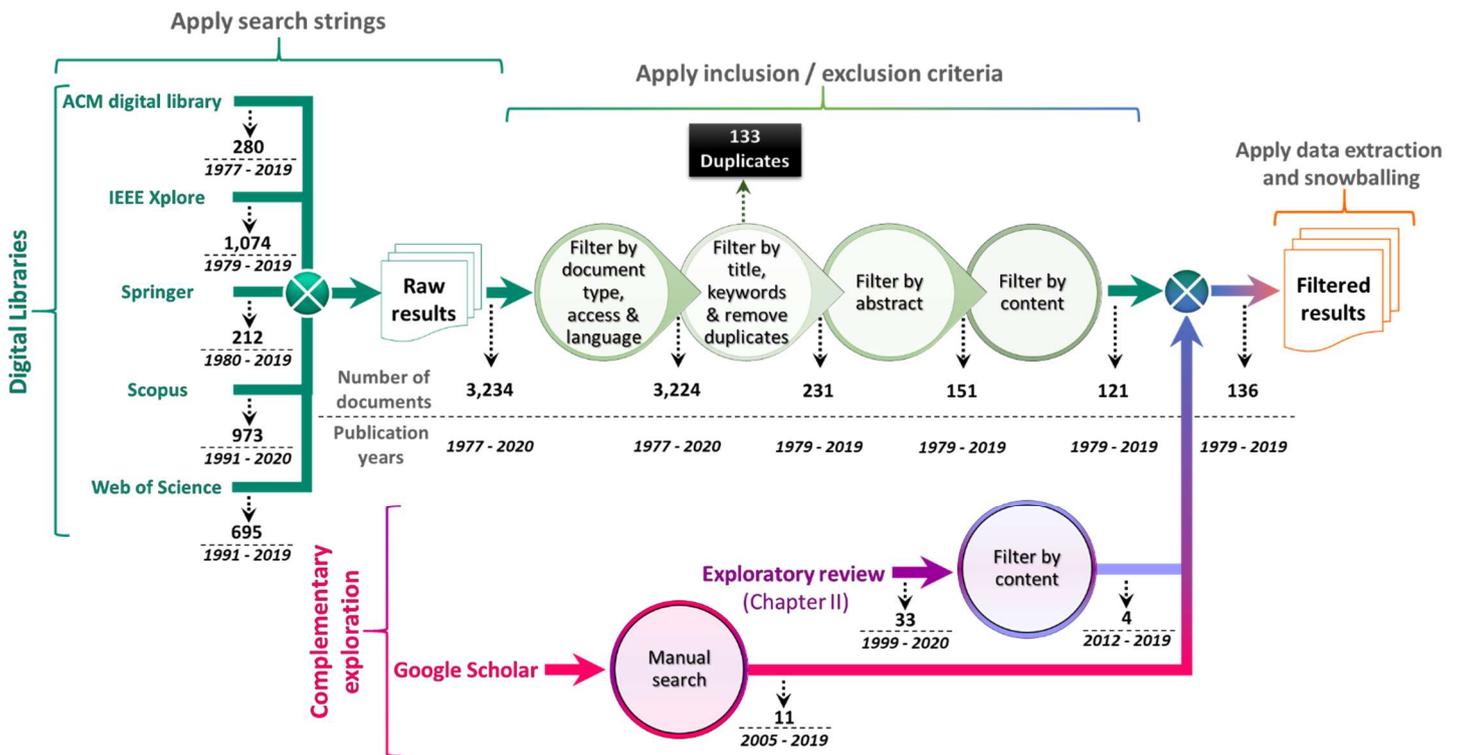


Figure 29 - The distinct stages of the systematic literature review process for study selection, with selected document numbers and publication years

The four screening, or filtering, stages are incremental, each stage taking as inputs the outputs of previous one, refining selection to the most relevant documents thanks to these inclusion and exclusion criteria. Furthermore, the effort to perform each selection stage increase incrementally as well. This behavior is explained by the fact that the more refined the list of selected documents is, the more thorough the analysis of the documents must be.

With the first filter, we have a very large number of potential documents to crawl and therefore, the first selection pass must be performed efficiently. Thus, the decision is made by considering only document type, language, and its accessibility. The second stage focus on title and duplicate document removal. The next stage needs more time investment on each document because the focus is on document abstract and therefore, it required to have each document abstract to be read. Note, paper abstracts are always available online. At the fourth and final stage, it is required to access, or download the full document and complete the last selection stage based on the document content. Hopefully at this stage, the number of candidate documents is reduced: in our current systematic, we started from a collection of 3,234 documents and reduced up to 151 documents at the input of the last stage, which represents a screening of 95.33%.

The result of this document cascade filtering is a list of 121 relevant documents (i.e., 39 articles from journals and 82 papers from conferences) covering a period between 1979 and 2019, and the major difficulties we faced during this systematic literature review were:

- Incorrect reference or pointer to retrieve the correct papers: sometimes there were some mistakes on the referenced paper, which was not pointing to correct quality model,
- Inaccessibility to publications: some published documents were not accessible or downloadable (e.g., old books),
- Bad association or wrong focus with quality model scope:
 - o A publication describes quality activity that is not associated quality model or modeling: for example, a QMS framework is not quality modeling with quality model,
 - o A model is not modeling quality: for instance, COCOMO [151] is a for cost modeling but not for quality modeling,
- Difficulties in understanding contributions: for example, complex papers referring to specific technical knowledge, or with unclear assumptions,

- Difficulties in the identification of quality model characters: for instance, retrieving which statistical method is used for a quality model, or its quality perspective.

Although the review collection and methodology were rigorous, we looked for potential biases due to the online database, the search query results or the filtering stages. So during this systematic literature review, we noted that several relevant papers, we previously found either through our exploratory review (see Chapter II.3) or some previous investigations, were missing. Consequently, we performed a complementary literature exploration (i.e., considering results from previous exploratory review and manual searches) finding 15 additional papers, four conference papers and 11 journal ones.

The result of the entire review process is a total of 136 publications done between 1979 and 2019, and summarized into Table 45 of Annex 5. Almost two third (i.e., 64%) of the 136 publications comes from conference while the source of the remaining 36% papers is journal (see Figure 30). Furthermore, most of these studies (i.e., 79.41%) were performed in an academic context, without any concrete practical consideration from industry partners whose are the primary demanders of these study results (see Figure 31). As depicted in Figure 32, 65.94% of the selected study contributions were done during the last decade, and this period includes the years with the most important number of selected studies: in 2014 we have 16 selected papers, in 2016 12 papers, and in both 2013 and 2018 11 papers. Therefore, we can infer that quality model study continues currently to be an active topic, and even get stronger interest also over the last decade. In parallel, we remark that beside one contribution from Mohanty [195] in 1979, no paper prior to 1994 passed our systematic review.

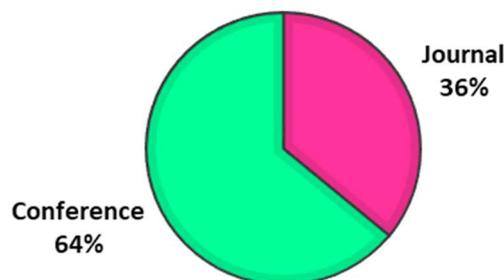


Figure 30 - Ratio of the two types (i.e., conference and journal) of selected studies

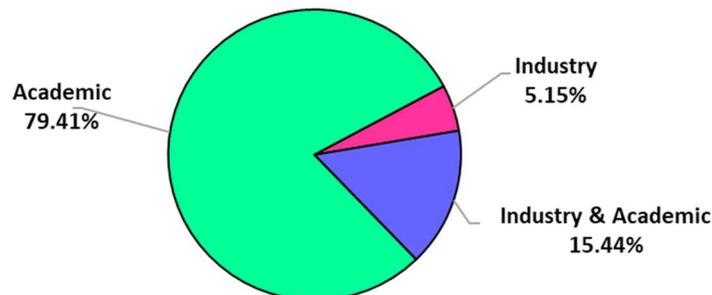


Figure 31 - Ratio of context (i.e., academic, industry, both) of the selected studies

Regarding the citations of these 136 papers, highlighting the publication impact factors, Figure 33 shows both cumulative and normalized distribution per year of the paper citation count. Frakes and Terry's journal paper [196] (1996) on metrics and models linked to software reuse is the most cited paper of our review selection. The cumulative citation bar graph confirms a high level of citation during last decade (e.g., in 2014, we can note a total of 459 citations) but the bar graph for normalized citation per paper number highlight that proportionately the highest ratio of the most cited happens before 2003, with few exceptions. Note, the 459 cumulative count in 2014 is due to a total of 16 papers, and thus the average citation is around 28 per paper. So, the average citation is around 33 per paper, and the 11 papers cited more than 100 are:

- Frakes and Terry's journal paper [196] (1996) "*Software Reuse: Metrics and Models*": 507 citations,
- Kitchenham and Pfleeger's journal paper [26] (1996) "*Software quality: the elusive target*": 230 citations,
- Khoshgoftaar *et al.*'s conference paper [197] (1999) "*Classification tree models of software quality over multiple releases*": 145 citations,
- Olsina *et al.*'s journal paper [198] (1999) "*Assessing the quality of academic websites: A case study*": 137 citations,

- Zhang and von Dran’s conference paper [199] (2001) “Expectations and rankings of Web site quality features: results of two studies on user perceptions”: 169 citations,
- Briand and Wüst’s journal paper [200] (2002) “Empirical studies of quality models in object-oriented systems”: 228 citations,
- Rawashdeh and Matakah’s journal paper [201] (2006) “A New Software Quality Model for Evaluating COTS Components”: 148 citations,
- Lincke *et al.*’s journal paper [202] (2008) “Comparing Software Metrics Tools”: 263 citations,
- Mohagheghi *et al.*’s journal paper [203] (2009) “Definitions and approaches to model quality in model-based software development - A review of literature”: 160 citations,
- Kritikos *et al.*’s journal paper [204] (2013) “A Survey on Service Quality Description”: 129 citations,
- Miguel *et al.*’s journal paper [190] (2014) “A Review of Software Quality Models for the Evaluation of Software Products”: 171 citations.

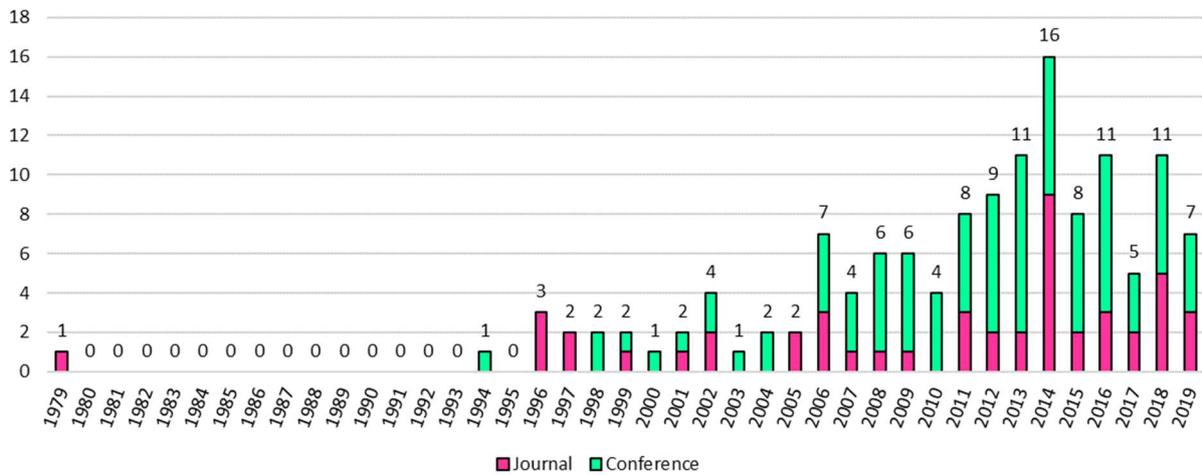


Figure 32 - Selected conference and journal paper distribution over publication year

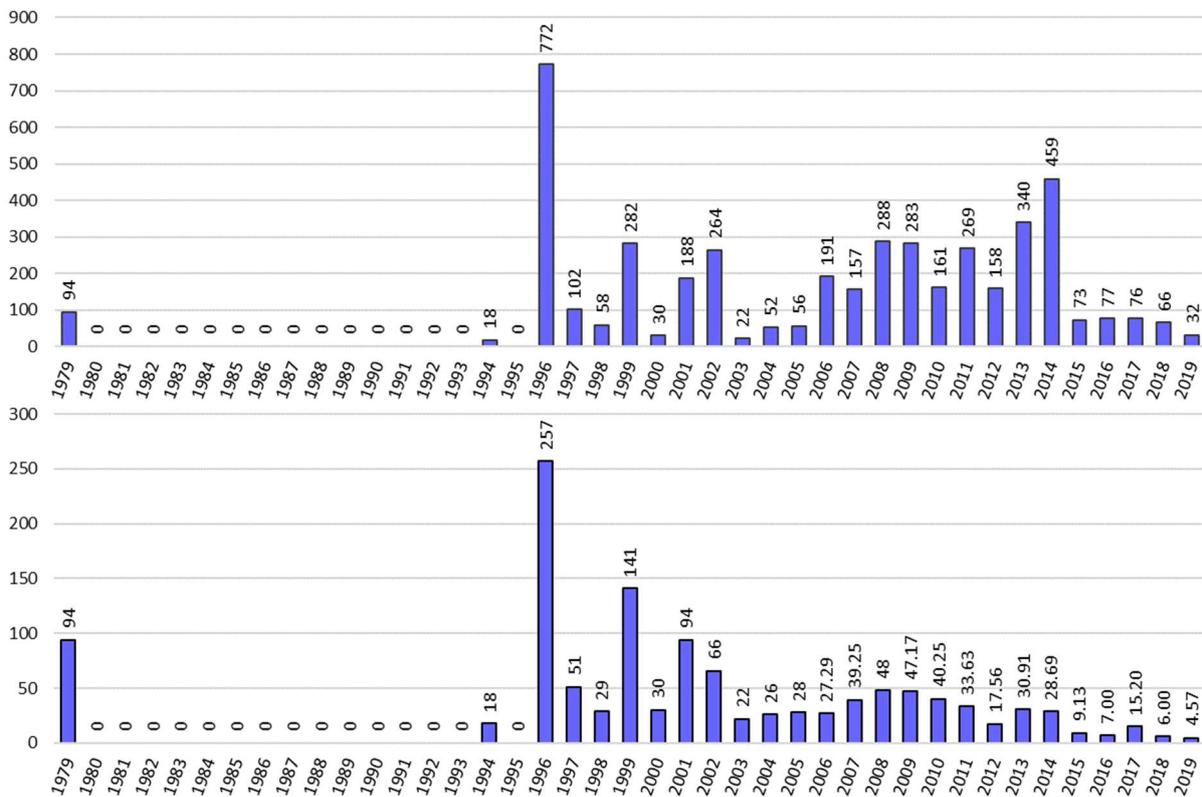


Figure 33 - Citation numbers of systematic literature review qualified papers per year; top bar graph, cumulative citation number, bottom bar graph, normalized citation number per number of papers

From this systematic literature review, 136 study papers, published between 1979 and 2019, were qualified and reviewed in order to address our three research questions.

Moreover, we remarked, based on these papers, that quality model study continues currently to be an active topic, even getting a stronger interest over the last decade, and is mostly lead by the academic community.

In the next following sections, taking advantage of that selected literature, we analyze it to find answers to the research questions, starting with software quality model classification aspect since this is linked to the topics and methodologies used in these papers.

3. Software quality model classification

Research Sub-question 2b

Considering a set of quality models, how to classify these quality models, what are the methodology, the criteria, and the characteristics to use?

In order to answer to this research question, we start our analysis of the 136 papers collection by identifying what kinds of study related to quality models each paper can be associated with. Knowing this information is crucial to learn how information related to quality models are exploited. Indeed, while a comparison study aims to identify and use relevant elements to classify and compare together quality models, a quality model creation study generally sets its focus on identifying limitations or gaps against a specific research case, and then addresses them. Thus, this analysis allows us to extract 65 types (see Table 46 of Annex 5) that we succeed to regroup into seven distinct types of study. These types are:

- *Analysis or survey on specific aspects or elements related to quality model*: the study aims to investigate a specific topic related to quality, quality characteristics or quality models.
- *Comparison of quality models or characteristics*: the purpose of such study is to identify a set of quality models and then uses some criteria to compare together the selection of quality models; the conclusion is often to identify the best quality model among that set.
- *Creation of new model*: the goal of this kind of survey is to reply to a need by creating a new quality model either from an analysis on a context or domain, and / or from existing quality models or quality characteristics.
- *Evaluation of quality models or characteristics*: this type of study is to perform against some specific use cases to evaluate how well a quality model perform; it can reveal not only the quality model advantages but also its disadvantages.
- *Quality model customization*: starting from a quality model, this kind of study describe the customization work done to adapt and customize a quality model to answer to some specific needs.
- *Quality model improvement*: the study object here is to improve some specific aspects of a quality model to answer to some predefined or analyzed constraints or limitations.
- *Systematic mapping or literature review*: the aim of this type of study is to address some specific research questions related to quality model in order to acquire and build further knowledge from existing literature and research work.

The mapping of the 136 papers against each of these seven types (cf. Table 18) reveals that they are not evenly distributed as depicted by Figure 34. Three types gather 81.25% of the studies. The most frequent study type concerns the creation of quality models, with 38.16% of the papers. This result reflects the fact that rather than reusing or customizing a quality model (i.e., 3.47% of the papers), researchers tend to create new quality models. Then, with respectively 26.39% and 16.67%, we have the analysis or survey on specific aspect, and comparison of quality models.

TABLE 18 - MAPPING OF THE 136 PAPER STUDIES AGAINST THE SEVEN TYPES OF STUDY

Type of study	Study Ids
Analysis or survey on specific aspects or elements related to quality model	SLR-S01, SLR-S03, SLR-S04, SLR-S05, SLR-S06, SLR-S13, SLR-S14, SLR-S16, SLR-S17, SLR-S26, SLR-S32, SLR-S42, SLR-S61, SLR-S62, SLR-S64, SLR-S65, SLR-S68, SLR-S69, SLR-S71, SLR-S74, SLR-S75, SLR-S80, SLR-S86, SLR-S87, SLR-S88, SLR-S89, SLR-S100, SLR-S101, SLR-S102, SLR-S107, SLR-S108, SLR-S110, SLR-S113, SLR-S119, SLR-S120, SLR-S128, SLR-S129, SLR-S135

Comparison of quality models or characteristics	SLR-S19, SLR-S25, SLR-S41, SLR-S44, SLR-S49, SLR-S50, SLR-S53, SLR-S66, SLR-S67, SLR-S70, SLR-S77, SLR-S81, SLR-S92, SLR-S93, SLR-S94, SLR-S96, SLR-S99, SLR-S109, SLR-S114, SLR-S115, SLR-S122, SLR-S126, SLR-S127, SLR-S136
Creation of new model	SLR-S07, SLR-S08, SLR-S09, SLR-S10, SLR-S11, SLR-S12, SLR-S15, SLR-S18, SLR-S21, SLR-S22, SLR-S23, SLR-S24, SLR-S27, SLR-S29, SLR-S30, SLR-S31, SLR-S34, SLR-S35, SLR-S36, SLR-S37, SLR-S38, SLR-S39, SLR-S43, SLR-S45, SLR-S46, SLR-S47, SLR-S48, SLR-S54, SLR-S55, SLR-S56, SLR-S58, SLR-S59, SLR-S60, SLR-S63, SLR-S72, SLR-S73, SLR-S76, SLR-S78, SLR-S83, SLR-S84, SLR-S85, SLR-S91, SLR-S97, SLR-S98, SLR-S103, SLR-S104, SLR-S105, SLR-S118, SLR-S121, SLR-S122, SLR-S123, SLR-S124, SLR-S125, SLR-S130, SLR-S132
Evaluation of quality models or characteristics	SLR-S19, SLR-S33, SLR-S99, SLR-S109, SLR-S114, SLR-S115, SLR-S136
Quality model customization	SLR-S40, SLR-S51, SLR-S52, SLR-S79, SLR-S106
Quality model improvement	SLR-S02, SLR-S20, SLR-S28, SLR-S94
Systematic mapping or literature review	SLR-S57, SLR-S82, SLR-S90, SLR-S95, SLR-S111, SLR-S112, SLR-S116, SLR-S117, SLR-S131, SLR-S133, SLR-S134

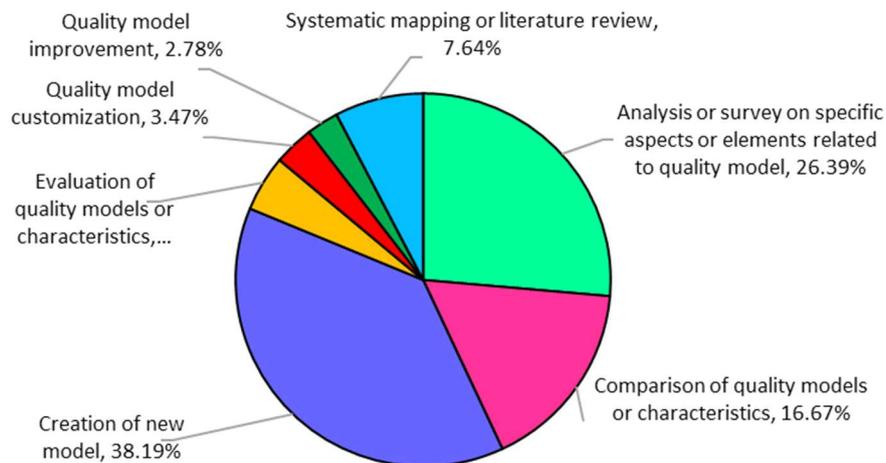


Figure 34 - Distributions of studies per type of study

Now that we have learnt the tendency of the different kinds of study that these 136 papers cover, our next step is to find which classification or comparison criteria elements have been used. So, similarly to the type of study analysis, we extracted from the papers, 114 distinct criteria combinations that we clustered into 8 unique criteria categories (see TABLE 19):

- *Against specific quality model or characteristics*: under this category, the goal is to compare or classify quality models against a specific quality model (e.g., the quality models evaluation against ISO/IEC/IEEE 25010 of Motogna et al. [94]) or some specific characteristics such as the evolution of quality characteristics linked to mobile software of Gezici et al. study [205], for example.
- *Context / purpose / scope / viewpoint*: the aim here is to rely on several usual aspects of a quality model; these are the context (e.g., from Yan et al. [206] academic, industrial, or both), the purpose (e.g., definition, assessment, prediction, multi-purpose from Deissenboeck's DAP [122]), the scope composed of the object of interest (e.g., web-service in Oriol et al. [12], open-source in Adewumi et al. [189] or in Petrinja et al. [207]) and the quality focus described in the CQML classification scheme of Kläs et al. [10] (e.g., general, specific, defects, maturity, cost), and the stakeholder viewpoint depicted by the essential views of Horgan et al. [84].
- *Description*: either a brief or complete description of quality models is performed, often including some historical motivation and context with authors, the main characteristics of the quality model, the application domain (e.g., IT, transportation, medical, socio-economic from Fath-Allah et al. [208]), the quality perspectives (e.g., the five quality perspectives of Garvin [101]: user, product, manufacturer, transcendental, value-based).
- *Evaluation / results / benefits vs limitations*: the object of this category is to compare or classify quality models based on the exercise and evaluation results of the model applied to sample use cases, or to analyze the model in order to identify its benefits and limitations.

- *Formalism*: the focus is put on the type of formalism used to model the quality ; we can rely on Wagner’s guidelines [27]: meta-model, hierarchical, and statistical or implicit.
- *None*: neither comparison, nor classification criterion is used; this concerns only one paper related describing a statistical quality model creation that is used to analyze and predict software defects in Hitachi company; we notice no reference to other similar work from the authors.
- *Quality characteristics / sub-characteristics / metrics / Tools*: the classifications or comparisons are made between the presence and the enumeration of quality characteristics, sub-characteristics, metrics and / or tools associated to quality model.
- *Quality model parent*: this criterion spots the attention on the quality model parent(s) when the quality model under focus is inheriting from one or several anterior quality models; Thapar *et al.* [11] (see Figure 35) qualified that relationship as basic versus tailored quality models ; the tailored quality models are issued from basic quality models.

TABLE 19 - MAPPING OF THE 136 PAPERS STUDIES AGAINST THE EIGHT CATEGORIES OF CLASSIFICATION OR COMPARISON CRITERIA

Classification or comparison criteria	Study Ids
Against specific quality model or characteristics	SLR-S16, SLR-S41, SLR-S47, SLR-S57, SLR-S60, SLR-S76, SLR-S86, SLR-S95, SLR-S99, SLR-S103, SLR-S106, SLR-S109, SLR-S110, SLR-S111, SLR-S112, SLR-S115, SLR-S125, SLR-S133, SLR-S135, SLR-S136
Context / purpose / scope / viewpoint	SLR-S03, SLR-S13, SLR-S15, SLR-S21, SLR-S40, SLR-S44, SLR-S51, SLR-S52, SLR-S54, SLR-S58, SLR-S59, SLR-S61, SLR-S62, SLR-S68, SLR-S69, SLR-S75, SLR-S78, SLR-S79, SLR-S80, SLR-S82, SLR-S84, SLR-S85, SLR-S89, SLR-S90, SLR-S91, SLR-S93, SLR-S94, SLR-S97, SLR-S100, SLR-S102, SLR-S105, SLR-S111, SLR-S118, SLR-S120, SLR-S121, SLR-S131, SLR-S135
Description	SLR-S06, SLR-S07, SLR-S17, SLR-S26, SLR-S36, SLR-S45, SLR-S51, SLR-S56, SLR-S59, SLR-S63, SLR-S64, SLR-S66, SLR-S69, SLR-S70, SLR-S72, SLR-S75, SLR-S78, SLR-S81, SLR-S84, SLR-S89, SLR-S91, SLR-S92, SLR-S94, SLR-S96, SLR-S100, SLR-S103, SLR-S108, SLR-S110, SLR-S112, SLR-S113, SLR-S114, SLR-S115, SLR-S118, SLR-S120, SLR-S121, SLR-S122, SLR-S124, SLR-S126, SLR-S131, SLR-S132, SLR-S133, SLR-S135
Evaluation / results / benefits vs limitations	SLR-S02, SLR-S04, SLR-S10, SLR-S12, SLR-S14, SLR-S17, SLR-S19, SLR-S20, SLR-S23, SLR-S25, SLR-S27, SLR-S29, SLR-S30, SLR-S32, SLR-S33, SLR-S35, SLR-S42, SLR-S43, SLR-S45, SLR-S46, SLR-S47, SLR-S48, SLR-S49, SLR-S50, SLR-S53, SLR-S55, SLR-S59, SLR-S65, SLR-S67, SLR-S69, SLR-S72, SLR-S73, SLR-S77, SLR-S83, SLR-S85, SLR-S90, SLR-S92, SLR-S96, SLR-S115, SLR-S122, SLR-S123, SLR-S124, SLR-S129, SLR-S130
Formalism	SLR-S10, SLR-S12, SLR-S14, SLR-S17, SLR-S20, SLR-S24, SLR-S29, SLR-S33, SLR-S45, SLR-S66, SLR-S69, SLR-S70, SLR-S82, SLR-S83, SLR-S85, SLR-S107, SLR-S108, SLR-S111, SLR-S112, SLR-S116, SLR-S117, SLR-S123, SLR-S126, SLR-S134
None	SLR-S09
Quality characteristics / sub-characteristics / metrics / Tools	SLR-S01, SLR-S05, SLR-S08, SLR-S11, SLR-S16, SLR-S18, SLR-S22, SLR-S23, SLR-S25, SLR-S26, SLR-S28, SLR-S30, SLR-S31, SLR-S34, SLR-S37, SLR-S38, SLR-S39, SLR-S41, SLR-S43, SLR-S44, SLR-S45, SLR-S46, SLR-S47, SLR-S50, SLR-S53, SLR-S57, SLR-S58, SLR-S61, SLR-S65, SLR-S66, SLR-S67, SLR-S68, SLR-S69, SLR-S70, SLR-S71, SLR-S75, SLR-S77, SLR-S80, SLR-S81, SLR-S83, SLR-S84, SLR-S85, SLR-S86, SLR-S87, SLR-S88, SLR-S89, SLR-S90, SLR-S91, SLR-S92, SLR-S93, SLR-S94, SLR-S95, SLR-S96, SLR-S97, SLR-S98, SLR-S99, SLR-S100, SLR-S101, SLR-S103, SLR-S105, SLR-S106, SLR-S107, SLR-S109, SLR-S110, SLR-S111, SLR-S112, SLR-S113, SLR-S116, SLR-S117, SLR-S118, SLR-S120, SLR-S121, SLR-S122, SLR-S123, SLR-S124, SLR-S125, SLR-S126, SLR-S127, SLR-S128, SLR-S129, SLR-S131, SLR-S133, SLR-S134, SLR-S135, SLR-S136
Quality model parent	SLR-S77, SLR-S79, SLR-S82, SLR-S90, SLR-S93, SLR-S117

Once again, the criteria categories (cf. Figure 36) for quality model classification or comparisons are not evenly distributed. Unsurprising, with 32.8% the predominant category relates to quality characteristics / sub-characteristics / metrics / tools because those elements are the ones which composed a quality model. Moreover, description, context / purpose /scope /viewpoint, and formalism, with respectively 16.22%, 14.29% and 9.27%, complete the portrayal of a quality model. Nevertheless, the second most important category doesn’t belong to them: this is the evaluation / results / benefits vs limitations. Therefore, we can conclude that after using quality model elements such as quality characteristics for instance, researchers prefer to construct their classification or comparison study on empirical or theoretical evaluations and determine quality model benefits and limitations.

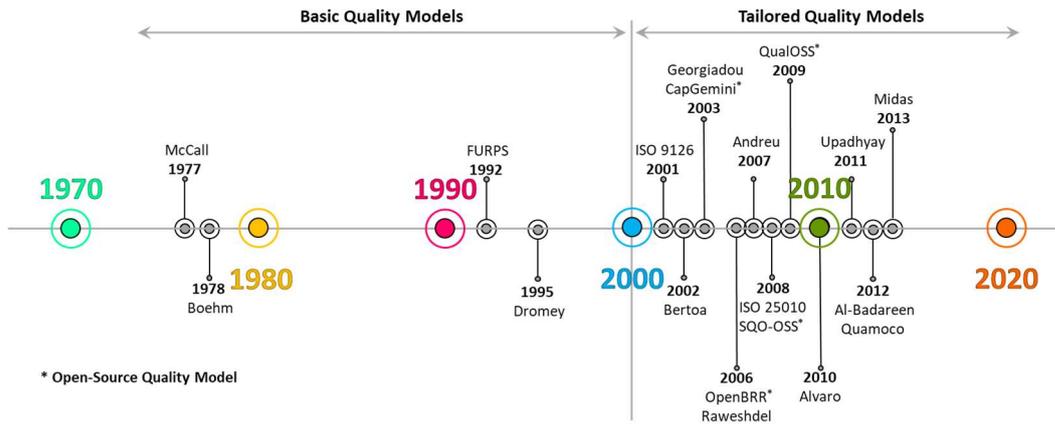


Figure 35 - Basic vs. Tailored quality model categorization (source: Thapar et al. [11])

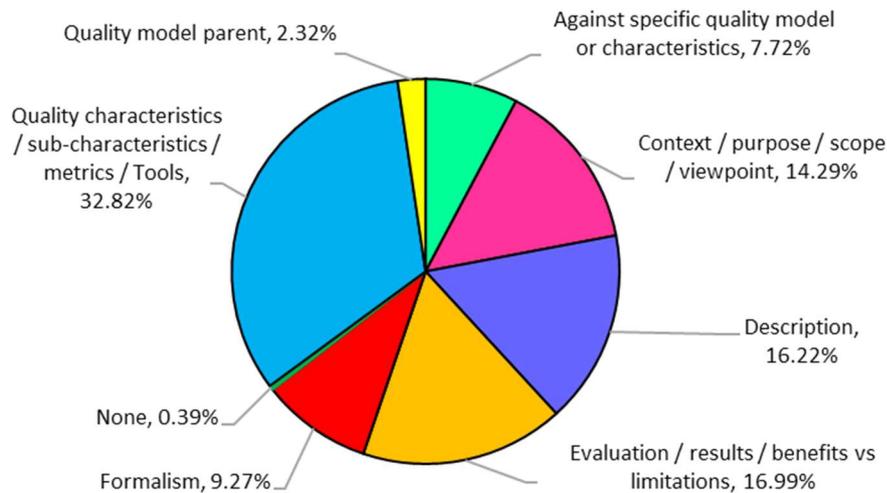


Figure 36 – Distributions of study classification or comparison criteria categories

So, based on these findings, five complementary themes emerge to describe and classify quality models. We have id, bibliographic, definition, scope and structural. Figure 38 provides the full overview of these five themes together with their refined distinct elements to practically describe, organize and classify quality models.

First, **id** is similar to an id card. It regroups the quality model name, its author(s), publication year, and its pedigree. Under pedigree, we include the quality model parents used to inspire, create, or customize the quality model.

Bibliographic theme integrates the reference source, availability and accessibility, and its importance in front of community thanks to citation counters. We have 3 citation counters: from the original source (i.e., publisher counter) when available, from study papers (i.e., how many times the quality model is cited over the papers issued from a systematic literature review) and from Google Scholar (<https://scholar.google.com/>). We note that from Snyder's analysis on citation counters [209], Google Scholar appears to provide a little bit optimistic results but they are closer to the reality than publisher ones. Indeed, publisher metrics often do not count for reference in papers, thesis, or another scientific document outside the publisher scope. Therefore, we decided to use Google Scholar as our counter reference but also keep the publisher citation metrics to strengthen the validity of our citation metrics. Moreover, by using Google Scholar citation information independently of the publishers, we have citation information generate in a consistent manner which brings confidence when we compare, in a relative way (e.g., as ordinal), papers based on their citation results. Figure 37 illustrates the differences in citation metrics depending on the metric source and highlighted by Snyder.

Next theme is the **definition** of three main elements that characterized a quality model. We have the basic or tailored nature of a model as defined by Thapar *et al.* [11] (cf. Figure 35) and then relayed by Miguel *et al.* [190], to which we can also attach standard nature (e.g., ISO/IEC 9126 or ISO/IEC/IEEE 25010). We recall that Wagner *et al.* [83] concluded that 28% of companies in their survey use quality model standards, even if 79% of them are

finally customized. The second element is the quality model purpose. This is aligned with the Deissenboeck *et al.*'s DAP classification [122] whose valid purposes are definition, assessment, prediction and multi-purpose. Model formalism is the third and last main characterization elements. It indicates whether the model is defined by a meta-model, a hierarchical model, or by an implicit or statistical model (see Wagner [27]).

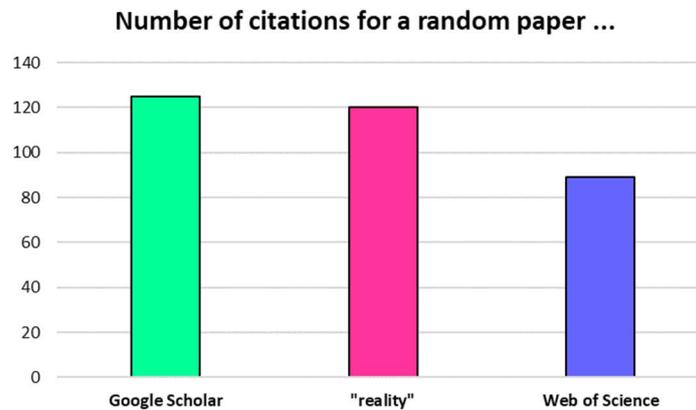


Figure 37 - Comparison done by Snyder [209] on citation metrics difference between Google Scholar, Web of Science publisher and the reality

Fourth, **scope** theme describes the quality model focus and its applicability areas. Thus, we have:

- the quality perspective as introduced by Garvin [101] (i.e., user, product, manufacturer, transcendental, and value-based),
- the quality focus likes in Kläs *et al.*'s CQML [10] (e.g., defects, maturity, general, functionality),
- the quality model domain used also by Fath-Allah *et al.* [208] (e.g., IT, transportation, medical, socio-economic),
- the quality model context (i.e., academic, industrial, or both) that we can find also in Yan *et al.* [206],
- the object of interest (e.g., web-service (Oriol *et al.* [12]), open-source (Adewumi *et al.* [189], Petrinja *et al.* [207]))
- the stakeholder viewpoint, identical to Horgan *et al.*'s essential views [84], and which we retrieved in the Kläs *et al.*'s CQML classification scheme [10] (e.g., sponsors, users, developers).

To complete the quality model characterization, the last theme is the **structural** one. This corresponds to the quality model constituent elements and their relationships. Consequently, we have first the vocable that used in the quality model (e.g., characteristics, attributes, factors, metrics), the characteristic / sub-characteristic list and definitions, the metrics list and definitions, and both qualitative and quantitative relationship (e.g., aggregation, correlation between metrics and quality characteristics) found also in Kläs *et al.*'s CQML [10]. Note that the characteristic / sub-characteristic and metrics could be either not defined, direct (i.e., available) and defined, direct and partially defined, indirect (i.e., available elsewhere, e.g. in another quality model) and defined, or indirect and partially defined. The notion of "direct" versus "indirect" was introduced and used by Thapar *et al.* [11] and Fath-Allah *et al.* [208]. Concerning the notion of "defined" versus "not defined", we rely on the work done Oriol *et al.* [12] bringing more nuance with their rating **Y** (i.e., explicitly defined), **Y+** (i.e., explicitly defined and contains subdivisions), **P** (i.e., partially, not explicitly defined but a quality attribute or metric can support that definition), **P+** (i.e., partially, not explicitly defined but several quality attributes or metrics can support that definition), and **ND** (i.e., not defined).

Finally, the result of the assembly of these five themes together with their characterization elements, shown in Figure 38, covers all the classification or comparison criteria that were extracted from the systematic literature review of 136 papers for a period from 1979 to 2019. This means that this consolidated result is therefore enough for classifying software quality models.

Nevertheless, extracting all information required by these classification quality model elements is the preliminary step in their classification. Next section addresses their use and organization to achieve the software quality model classification.

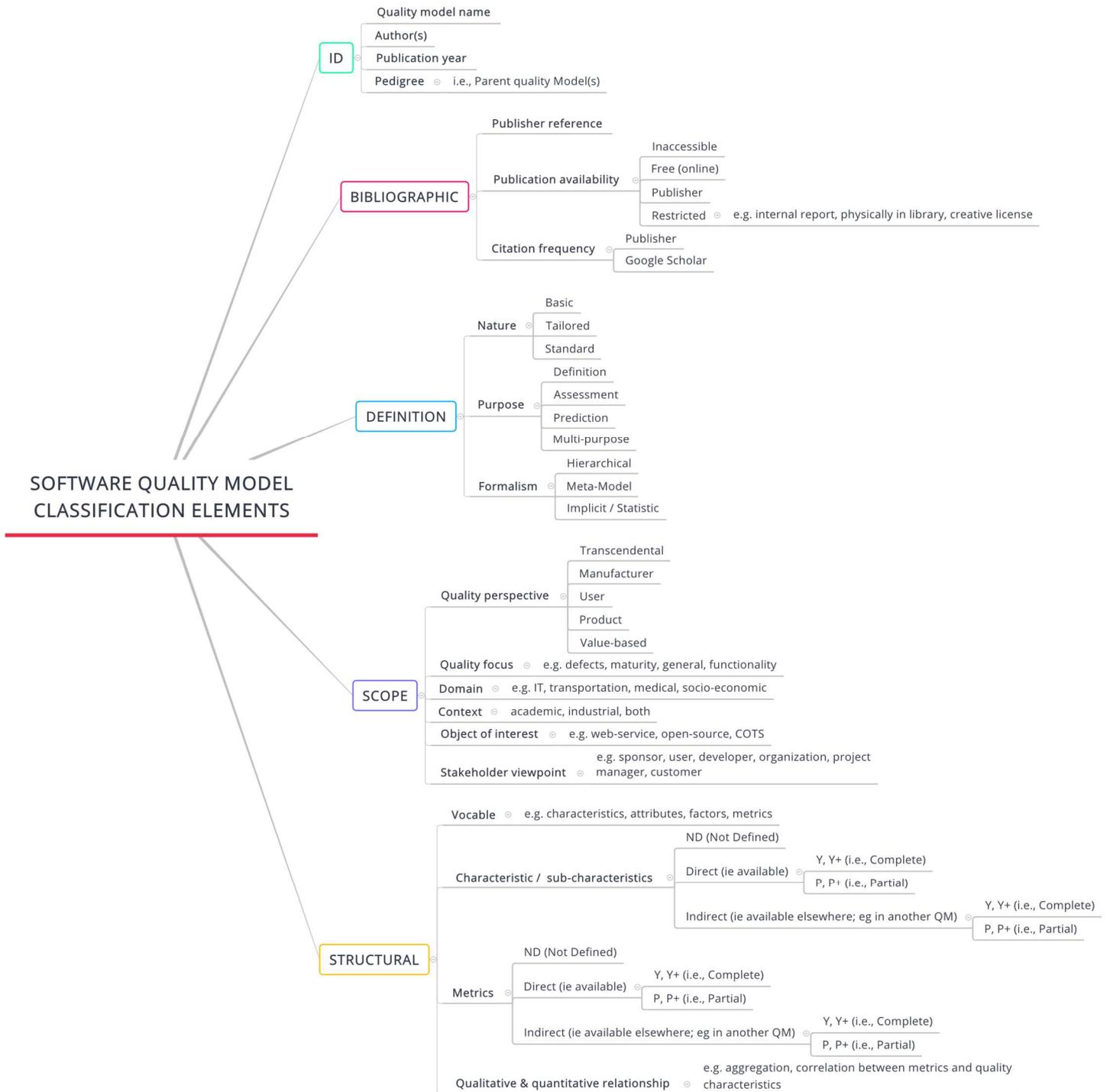


Figure 38 - The software quality model classification elements organized over five themes: id, bibliographic, definition, scope and structural

4. Contributions

a. Cladistic as Classification Method of Software Quality Models

Research Sub-question 2b Considering a set of quality models, how to classify these quality models, what are the methodology, the criteria, and the characteristics to use?

In Chapter V.3, we identified 20 software quality models classification elements that we organized into five distinct themes. Consequently, a straightforward and valid classification methodology for software quality

models can be achieved by categorizing software quality models into groups that share the similar classification element values. This kind of classification methodology is called a taxonomy, and its definition is “*a system for naming and organizing things, especially plants and animals, into groups that share similar qualities*” [210].

Nevertheless, the number of combinations to explore for this classification approach is huge. Indeed, regarding **definition** theme, we have 36 possible combinations (i.e., 3 nature possibilities x 4 purpose possibilities x 3 formalism possibilities). Likewise, the numbers of combinations for **bibliographic**, **scope** and **structural** theme are respectively 4, at least 3,240 and at least 100. So, the total number of all possible combinations to explore is at least 46,656,000 (i.e. 4 x 36 x 3,240 x 100), if we apply a taxonomy based on these elements. Note, we don't include **id** theme in this calculation since the only relevant element to consider here is the quality model pedigree, and its cardinality is difficult to evaluate, or guesstimate without collecting software quality models. However, knowing that there exists at least more than 46 millions of combination to explore is enough to conclude that this method is not efficient for our needs.

Thus, instead of dealing only with a taxonomy, we can also take benefit of the timeline and evolutionary knowledge contain in software quality model pedigree and which can be depicted likes the genealogical tree of quality models achieved by Oriol et al. [12].

Therefore, our classification method proposal becomes a composition between the use of taxa (i.e. “*taxonomic categories, as species or genera*” [211]), likes in taxonomy, but also, with the consideration of the timeline and evolutionary knowledge and reflected into homologies (i.e., “*a fundamental similarity based on common descent*” [212]). In other words, our proposal is to rely on cladistic, which is the “*classification of organisms based on the branching of descendant lineages from a common ancestor*” [213], as classification methodology. Furthermore, this proposal is directly aligned with the polymorphism mechanism introduced in Chapter IV.6.b integrating both the phenotypic variety (cf. Figure 23) and the evolution over time (cf. Figure 24).

When applying cladistic, the result is a cladogram as shown in Figure 23 for instance. The cladogram specifies degree of kinship relations between elements -in general organisms- which are then grouped into clade or taxon. Moreover, a cladogram is different from a genealogical tree because it does not show how ancestor and descendants are related together, but rather uses common ancestors and the study of shared characters to classify the taxa which is more relevant for classifying software quality models.

We applied on the software quality model classification elements the compatibility cladistic analysis from Estabrook *et al.* [214] where the objective is to optimize the element grouping (i.e., economy of hypotheses) thanks to the maximum mutually compatible characters. So, we identified five taxa whose definitions come directly from the **definition**, **scope**, and **structural** themes of the software quality model classification elements and are organized from the most generic characters (i.e. with less distinct cases) to the most specific one (i.e. with many distinct cases). We start with **definition**, considering “formalism” and “purpose” to describe the characters considered in the first taxon. We use the two elements together since they are both closely related. For instance, a meta-model formalism is often associated to definition purpose, while a statistic formalism is often used for prediction purpose.

The next level of taxon is about **structural** theme with again two closely related elements that are “elements (vocabulary)” and “qualitative & quantitative relationship”. This theme can be perceived as derived, or specialized, from the **definition**-based taxon, especially from the “formalism” element point of view. For the remaining three taxa, they are all linked to the **scope** theme. The most generic one is the “quality perspective” classification element. We voluntarily keep this element alone to define a taxon because it corresponds to the general perspectives that Garvin expressed about quality [101]. Then, two complementary and related viewpoints define the penultimate taxon. These are the “quality focus” and “stakeholder viewpoint”. Consequently, the final taxon composed of the “domain” and “object of interest” elements which are the closest elements together compare to the other **scope** elements.

As we highlighted, to complete the cladistic approach we must specify a homology in addition to these taxa. The homology supports the identification of common ancestors and the evolutionary arrangement through a timeline of evolution. This homology is defined by three classification elements: “publication year” and “pedigree” from **id** theme, and “nature” from **definition** theme.

The resulting software quality model clade is described in Figure 39, and integrates homology and taxa descriptions.

To proceed on software quality model classification, once software quality models are collected, their classification elements must be filled, or completed, and then applied against our proposed software quality model homology and taxa to generate the corresponding cladogram.

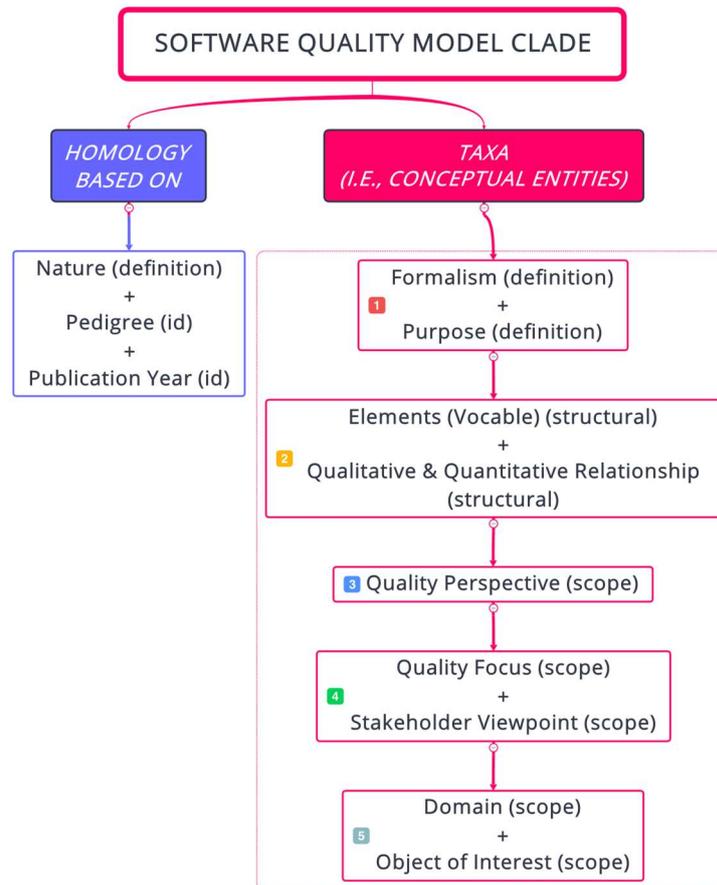


Figure 39 - Software Quality Model Clade based on a homology and five taxa

b. The first list of 492 software quality models

Research Sub-question 2a Considering software scope, what is the set of existing quality models?

The second contribution we achieve thanks to the systematic literature review is the elaboration of a consolidated software quality model list, answering consequently to the research sub-question 2a.

During the reading of the 136 retained papers, we first collected information (see Table 46 of Annex 5) about object of interest, domain and quality focus as specified in the taxa of the software quality model clade, described in Chapter V.4.a. Then, we look for software quality models refer in each of these papers. To do so, we follow snowballing approach as explained in Wohlin’s guidelines [215], [216]. The main approach idea is to collect further data by reviewing the reference papers as well that are cited in each the original systematic literature paper selection. Thus, we retrieved software quality models not only from the reference papers cited in each of the 136 papers, but also subsequently from the cited papers in the reference papers.

To include any identified software quality model into our collection, we aimed to retrieved and access to the source document, not only to confirm the relevance of the investigated contribution as software quality model but also to retrieve taxa related information likes quality perspective, pedigree, publication information (i.e., publication year, author(s), citation counters, publisher reference), purpose and formalism. In the case where the reference document was found but was not in English, or the reference was not accessible or retrievable, we

exclude it. This heavy literature review and investigation has resulted in the first list of 492 distinct software quality models, covering the period going from 1968 to 2019, for a total of 51 years. The complete enumeration of these 492 software quality models is available in Table 48 of Annex 6. We remark that 2008, 2009 and 2011 are the most productive years in term of software quality model creation and publication. Moreover, the last two decades concentrate 71.95% of the quality model production, particularly during the range 2000-2009 which represents 40.65% of the quality model production over the 51 years period. The details are shown in Figure 40. Note, the hierarchical formalism the most frequent one.

Regarding the associated normalized citation counters (i.e., for each year, this is the number of citation of the quality model reference papers per the number of quality models for that year), Figure 41 highlights the fact that despite a low software quality model production rate before 2000, this period contains the most cited contributions, even with a peak due to two widely cited contribution in 1984: Kano *et al.*'s quality model [58], and Basili and Weiss GQM quality model. [217].

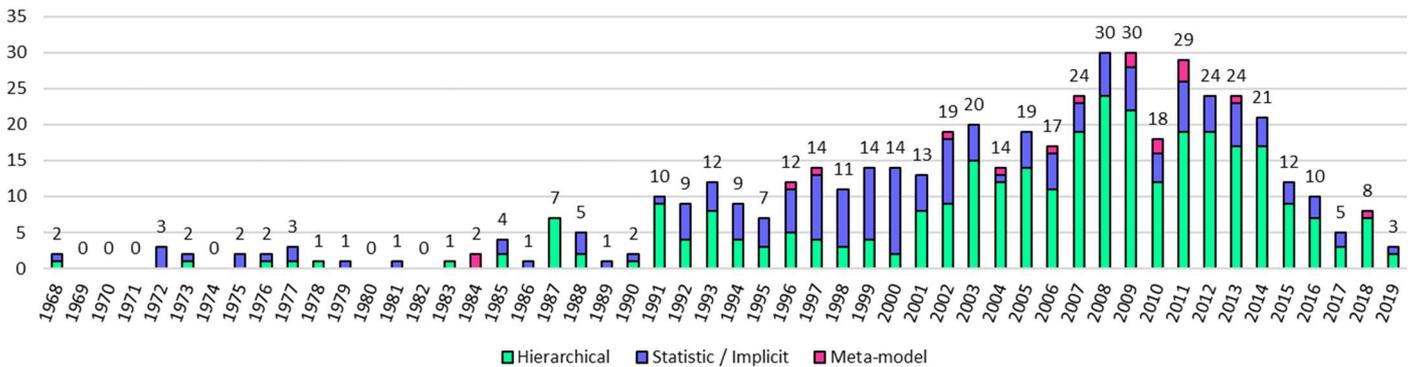


Figure 40 - The 492 created and published software quality models per formalism and year wise

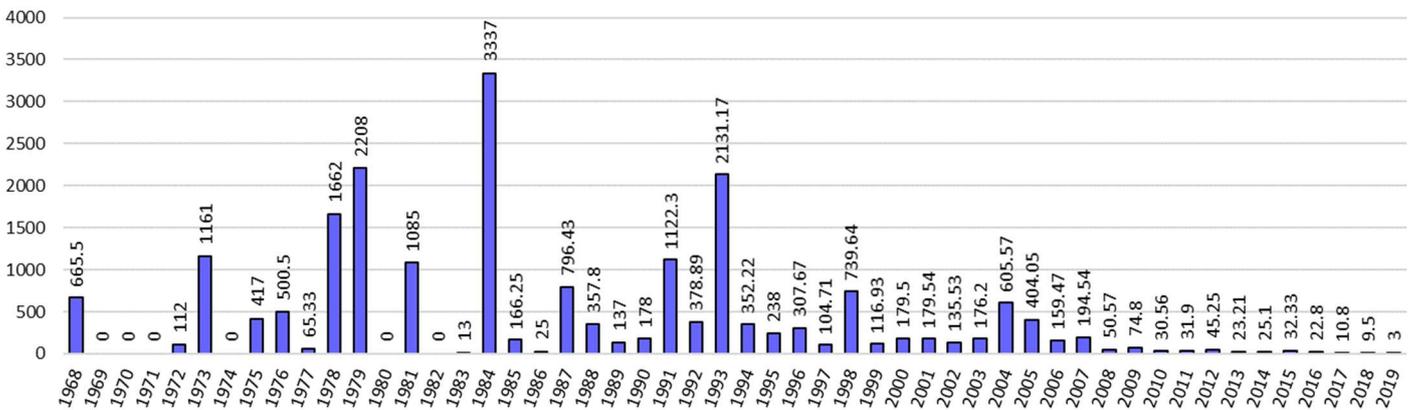


Figure 41 - The normalized citation numbers of the 492 published software quality model papers year wise

In parallel, we keep tracked of which software quality models were found or linked with each of the 136 study papers, with a maximum of 48 quality models for Oriol *et al* systematic mapping [12]. That information is aggregated into TABLE 47. As we can see in Figure 42 which represents the normal distribution of the number of retrieved quality models per study paper, there is an average (i.e., μ , value of gaussian distribution center) of 9.5 quality models found per study paper, with a standard deviation (i.e., σ) of 7.4. We recall that with a normal law distribution, a range of 2 times σ before and then after μ value includes 95.4499736% of samples.

The data drawn in Figure 41 come from Google Scholar citation counter, as explained in Chapter V.3. However, we noticed that for few software quality models, this type of citation counter didn't provide any answer. This is for instance the case with standard quality model such as ISO / IEC 9126. Hopefully, the data that we generate from the mapping of retrieved quality models against each study paper can fill these gaps, and ensure that the citation results are collected homogenously over the 136 study papers, which we cannot assess for data from Google Scholar counter. The result is shown in Figure 43, where we succeed to get citation data for all the 492 software quality models. We observe also that 7.11% of the 492 soft quality models (i.e., 35 over 492) represents

44.38% of the cited quality models in these studies. Figure 44 is a zoom on these 35 most cited software quality models (i.e., they are cited more than 5 times over the 136 study papers).

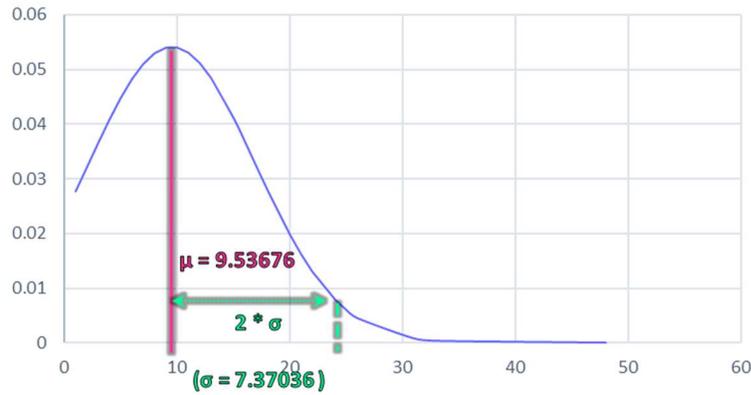


Figure 42 - The normal distribution related to the number of quality models per study paper

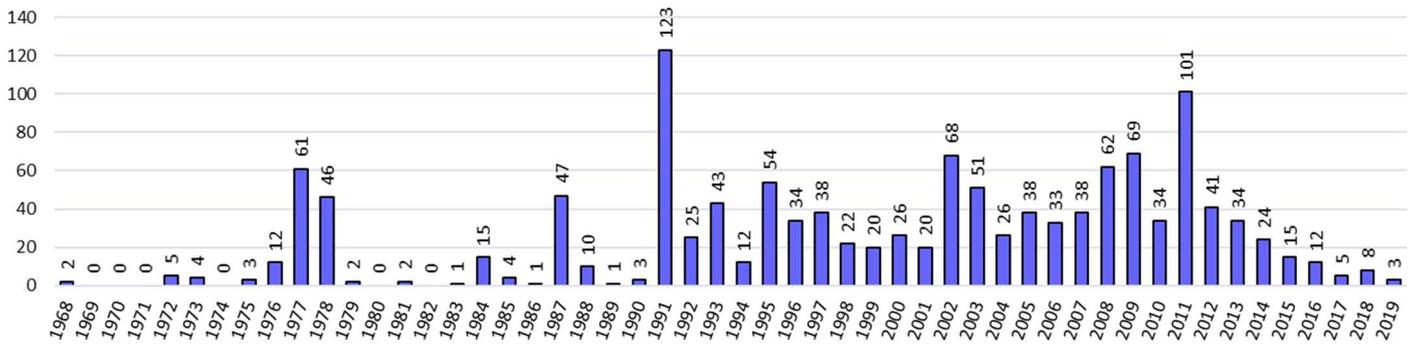


Figure 43 - Cumulative number of quality model citations in the 136 study papers year wise

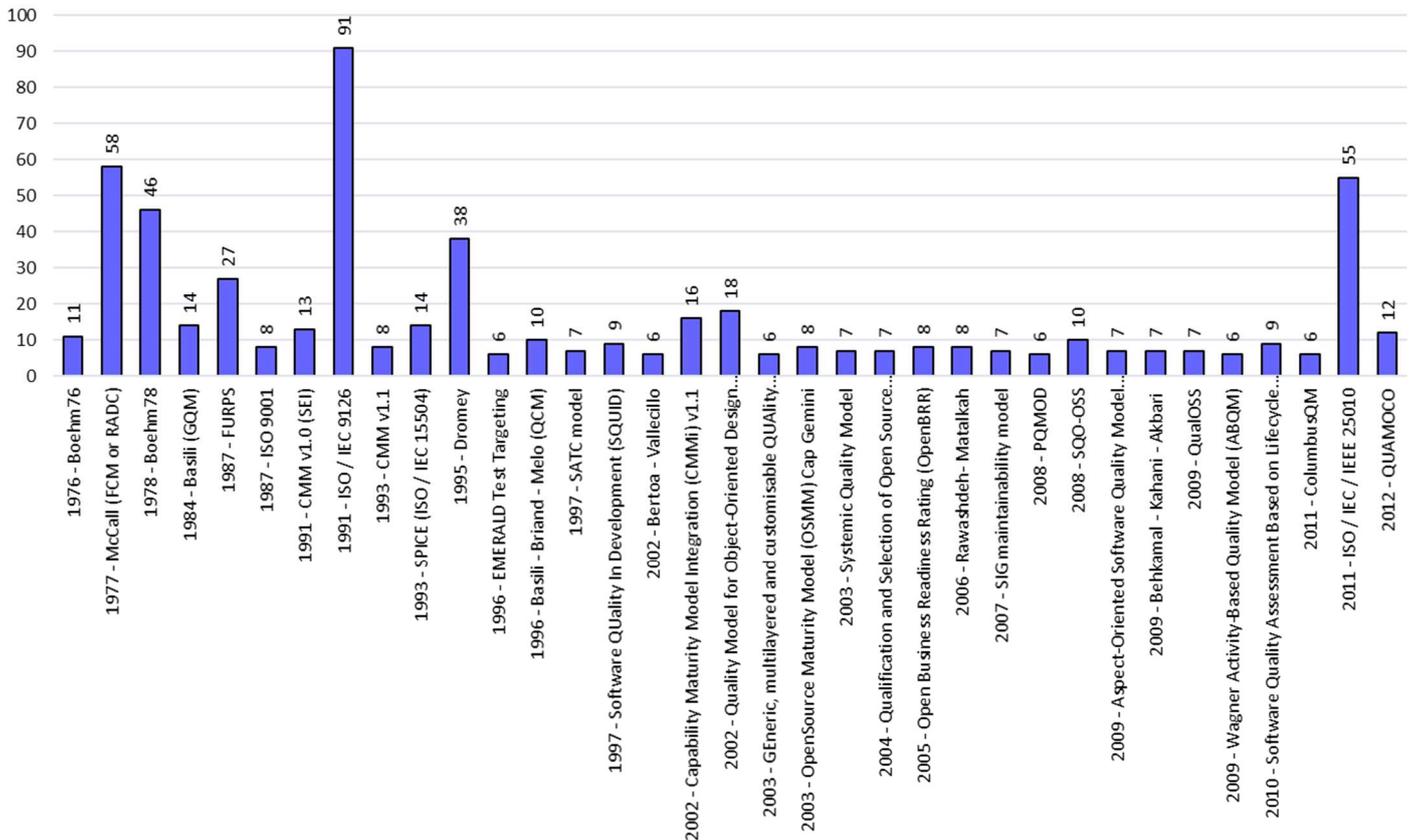


Figure 44 - The 35 most cited (i.e., cited more than 5 times) software quality models in the 136 study papers, order by chronologic order

To conclude this section, the 15 most cited software quality models from 1698 to 2019, and counting for 33.36% of the total of citations, are:

- **ISO / IEC 9126** [24]: published in 1991 and cited in 66.91% of the study papers (i.e., 91 times),
- **McCall (also known as FCM or RADDC)** [41]: published in 1977 and cited in 42.65% of the study papers (i.e., 58 times),
- **ISO / IEC / IEEE 25010 [23]**: published in 2011 and cited in 40.44% of the study papers (i.e., 55 times),
- **Boehm** [128]: published in 1978 and cited in 33.82% of the study papers (i.e., 46 times),
- **Dromey** [49]: published in 1995 and cited in 27.94% of the study papers (i.e., 38 times),
- **FURPS** [85]: published in 1987 and cited in 19.85% of the study papers (i.e., 27 times),
- **Quality Model for Object-Oriented Design (QMOOD)** [218]: published in 2002 and cited in 13.24% of the study papers (i.e., 18 times),
- **Capability Maturity Model Integration (CMMi) v1.1** [219]: published in 2002 and cited in 11.76% of the study papers (i.e., 16 times),
- **Basili (also known as GQM)** [217]: published in 1984 and cited in 10.29% of the study papers (i.e., 14 times),
- **SPICE (ISO / IEC 15504)** [220]: published in 1993 and cited in 10.29% of the study papers (i.e., 14 times),
- **CMM v1.0 from SEI team** [154]: published on 1991 and cited in 9.56% of the study papers (i.e., 13 times),
- **QUAMOCO** [160]: published in 2012 and cited in 8.82% of the study papers (i.e., 12 times),
- **Boehm** [42]: published in 1976 and cited in 8.09% of the study papers (i.e., 11 times),
- **Basili - Briand - Melo (also known as QCM)** [221]: published in 1996 and cited in 7.35% of the study papers (i.e., 10 times),
- **SQO-OSS** [222]: published in 2008 and cited in 7.35% of the study papers (i.e., 10 times).

c. Software quality model landscape and the selection question

Research Sub-question 2d	What is the most appropriate quality model is for embedded software in automotive?
Research Sub-question 4a	Is it possible to have a unique reference quality model for software product, or instead should we have a meta-model?

To answer to both research sub-question 2d and 4a, we must take profit of the underlying knowledge resulting from this unique collection of 492 software quality models by building a software quality model landscape, and then assess if there is a convergence to a unique reference quality model for software product, and which is the most appropriate quality model to select for embedded software in automotive.

First, the 492 software quality models categorization against the Deissenboeck’s DAP classification [122] shows, via Figure 45, that a majority (i.e., more than 58%) of published software quality models are assessment models, followed by the prediction quality models with almost 28% of the models. This teaches us that the main objective for modeling quality is put on the quality assessment and then on its prediction. This statement sounds logical because we must know how to assess or control before being able to predict.

The next aspect of this landscape is regarding the formalism used for software quality model and illustrated by Figure 46: the overall result is similar to the DAP classification one. Indeed, the principal formalism is the hierarchical one, with around 63% of the cases, and, with about 33%, we find the statistic, or implicit models. We could expect this kind of proportion because in most of the prediction times, a statistic, or implicit model is construct, relying on historical data and specific expert knowledge to predict quality. Concerning the hierarchical quality models, their usages are widely spread since this is the easiest formalism to understand and explain. We remark that meta-model formalism represents only 3.45% of the cases while we have a bigger proportion (i.e., 10.14%) for definition model. The reason of this difference is due to the fact that definition models are either described through meta-model and hierarchical model.

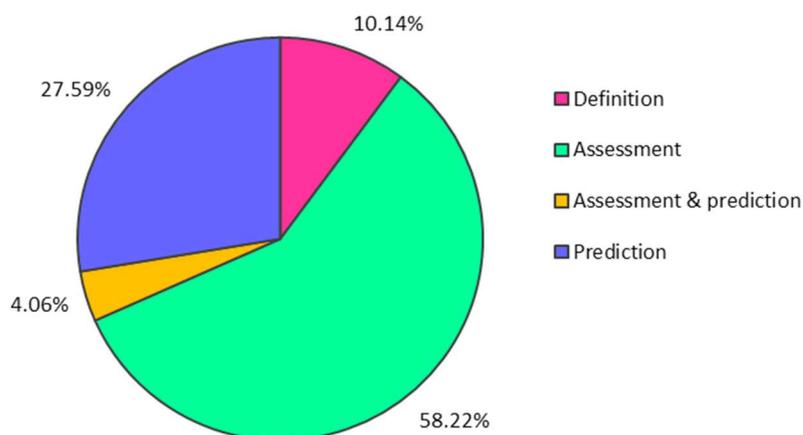


Figure 45 - DAP type distribution of the 492 software quality model samples

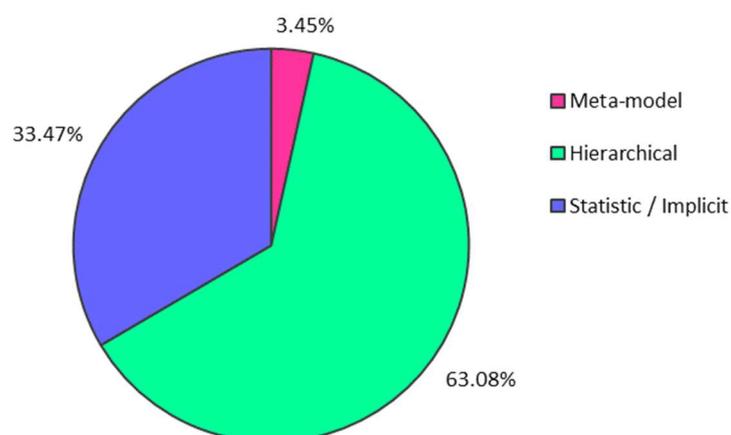


Figure 46 - Formalism type distribution of the 492 software quality model samples

Going one step further in the statistic or implicit formalism comprehension, we aimed to understand and retrieve the main applied methods. We found 29 data mining methods, going from regression to classification, from fuzzy logic to Bayesian or neural networks. We noticed that more than one method was often used together to strengthen the prediction, but in our collected data, we counted only the predominant one for each prediction software quality model.

So, with 17.58%, the statistical methods are the most applied type of data mining methods. There are often based on various statistical analysis of historical data. The second most used method (i.e., 11.52%) is the logistic regression due to its particular ‘S’ curve result. Its popularity sounds to recognize a better-adapted predictor for software quality compared to classical regression analysis, used in 6.06% of the cases. Equally with 9.09% of usage, fuzzy logic and neural network are the two next frequent methods, but their implementations are more elaborated compare to regression methods. Another interesting recurrent (i.e., 8.48%) used method is the capture – recapture approach. This is a statistical inference method usually used in ecology. Its main principle is to consider only a subset of the studied population to infer, or predict, some specific result against the entire population. Moving on with the next less frequent data mining methods, we retrieve the regression analysis with 6.06% likes the Bayesian network which can associated to a hierarchical model, and the following three methods, classification, classification tree and genetic algorithm, that are applied each one in 3.03% of the statistic or implicit quality model cases.

About the 19 other data mining methods, their usages occur in some minor cases. The complete result of the 29 method is enumerated through Figure 47.

To continue the construction of the software quality model landscape, we look for the quality perspectives, or views, as described by Garvin [101], considered in the 492 quality models. The accumulated data demonstrate an equi-distribution among three perspectives (cf. Figure 48): user, product, and manufacturer. Nevertheless, the

finer details of this distribution per software quality model reveal a more nuanced landscape (see Figure 49). In fact, while we still have a single quality perspective in 61% of software quality models, for almost 39% of the cases, software quality models integrate a mix of two or three quality perspectives. The single manufacturer perspective is a little bit apart from the rest of the result since it counts for 31.44% of software quality models. This result is explained by the fact that prediction quality models usually address a manufacturer quality perspective.

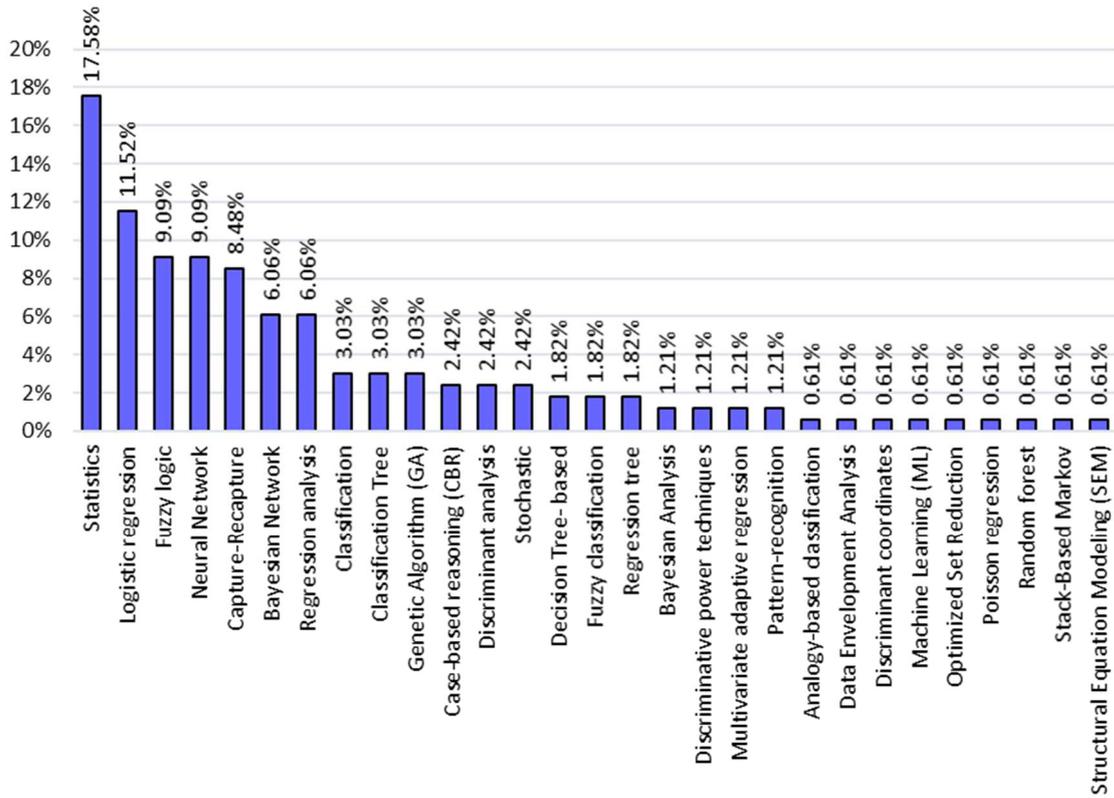


Figure 47 - Insight on the main prediction method distribution of the 492 software quality model samples

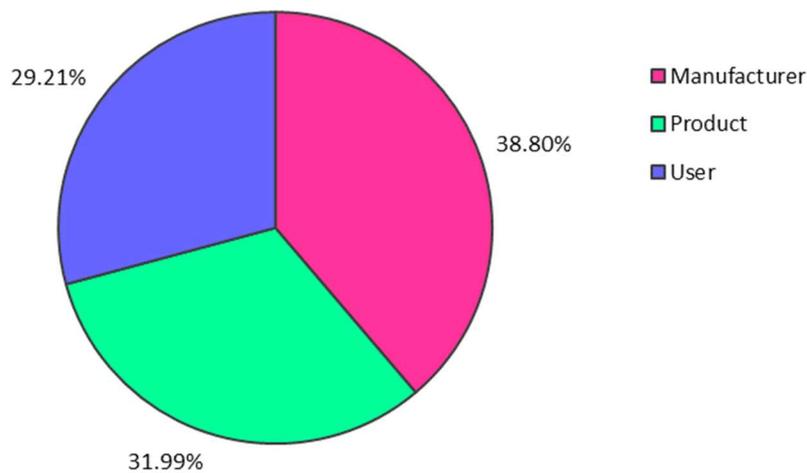


Figure 48 - Main quality perspectives, distribution of the 492 software quality model samples

Regarding software quality model scope depicted by Figure 50, the main concern that emerges from the 492 quality models is with regards to the product scope alone. This scope happens in more than 79% of these 492 models, proving that quality for software is first a matter of product quality. The second most frequent scope is the quality related to service software with only a little bit more than 12%, and then we found process, 3.45%, or product and process, 3.25%. Despite that remaining scopes cover a very small minority of the concerns, this doesn't mean that they are insignificant or should be discarded. On the contrary, they highlight future directions to investigate.

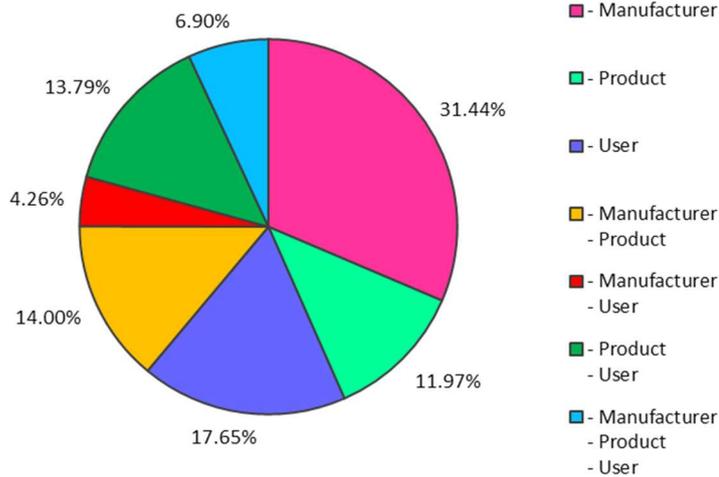


Figure 49 - Nuance in quality perspectives, distribution of the 492 software quality model samples

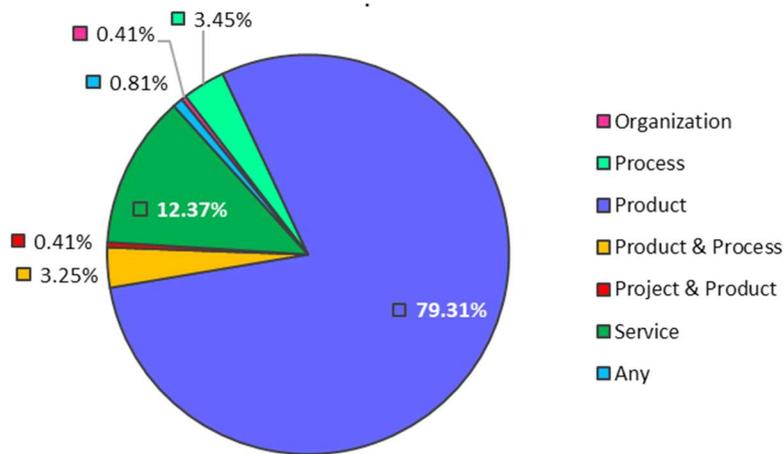


Figure 50 - Scope distribution of the 492 software quality model samples

For the last landscape facet, we aim to verify Thapar *et al.*'s postulate [11] (see also Figure 35) stating that before year 2000, we have basic quality models, and since 2000, we have tailored quality models. From the 492 software quality models, we distinguished the quality model with no parent model (i.e., new model creation) with the quality model with at least one parent model (i.e., derived, adapted, or tailored quality model). Figure 51 displays the result year wise. The result tends to affirm globally what Thapar *et al.* previously indicated but with some nuances.

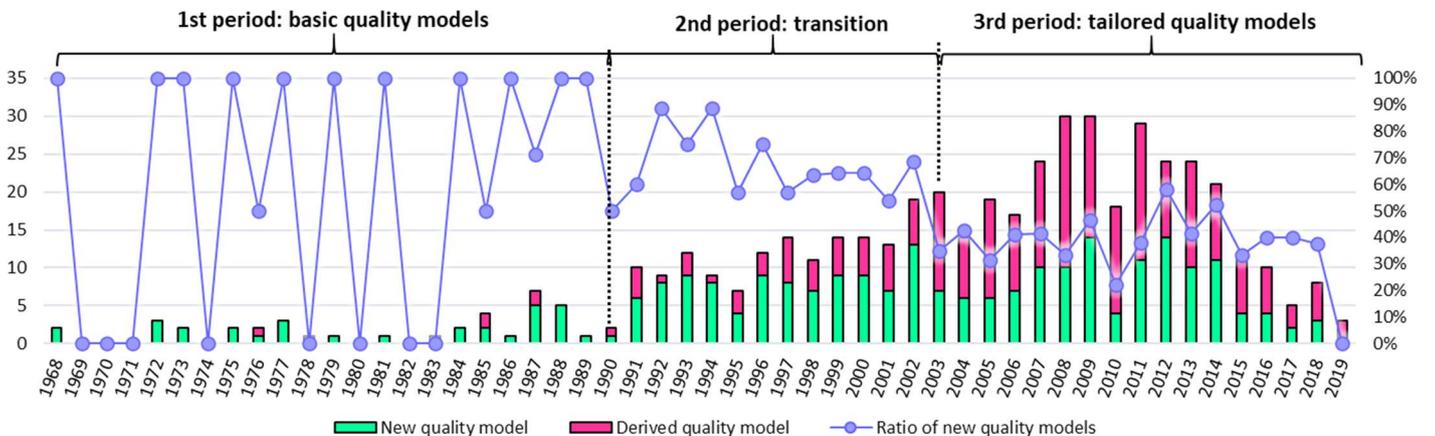


Figure 51 - Comparison between creation of new quality models and derived quality models over the 492 software quality model samples

First of all, we have three periods. Up to 1990, we have only few quality models that are mostly new quality models. So, we retrieve here the basic. Then from 1990 to 2003, there is a transition period where tailoring of published quality models is starting, but with still an important production of new quality model creation. Finally, since 2003, less new quality models are created, and more new quality models are resulted from the customization of other published quality models. Note, the two most customized quality models are the standard quality models ISO / IEC 9126 and ISO / IEC / IEEE 25010, as depicted by Figure 44 also reflecting quality model citation data.

Consequently, the second key nuance against Thapar *et al*'s postulate is the distinction between three type of quality models rather than two: we have basic, tailored, and standard quality models.

Through the analysis and construction of this landscape, we see the main tendencies that characterize software quality models, including their evolution since 1968. Therefore, published software quality models focus principally on quality assessment, and then on prediction. They are commonly hierarchical except for prediction where the adopted formalism is statistic or implicit. The quality perspectives are equally distributed over manufacturer, user, and product perspectives, but the scope is usually about product. Finally, the software quality model evolution is articulated around three periods: up to 1990, we have the basic quality model period, from 1990 to 2003, the transition period, and since 2003, we are in the quality model tailoring period.

Coming back to the research sub-questions 2d (i.e., “*What is the most appropriate quality model is for embedded software in automotive?*”) and 4a (i.e., “*Is it possible to have a unique reference quality model for software product, or instead should we have a meta-model?*”), from the landscape knowledge, it appears that there is not one distinct quality model emerging from these 492 software quality models, especially because the quality model depends on various variables such as the DAP, the quality perspective, or the object of interest, to cite a few of them.

Moreover, as we saw in our systematic literature review, the quality model comparison studies represent 16.67% of the 136 study papers (see Figure 34), with the latest study dealing on this topic published in 2019: Motogna *et al.* [94]. This is a good indicator highlighting that the quest to find the best quality model -and not a unique one- is still an active topic.

Thus, the answer to 4a is undoubtedly that it is not possible to have a unique reference quality model for software product, and, with this list of 492 quality models, we can also conclude that having a unique quality model covering all software product cases is an elusive target.

Regarding 2a, again from both the systematic literature review, combined with snowballing, and the exploratory literature review (cf. Chapter II.3), no quality model for embedded software in automotive clearly emerged. Nevertheless, if we refer to Figure 44, two of the three most cited software quality models among all the 492 models are the standard quality model ISO / IEC 9126 and ISO / IEC / IEEE 2501. And since we are in the quality model tailoring period, an appropriate approach is similar to Arhens *et al.* [51] who customized ISO / IEC 9126 to address their needs for a software quality model for automotive software architecture.

In conclusion, the answer to 2a is that there is no quality model ready yet for embedded software in automotive, but rather, we select the latest standard, ISO / IEC / IEEE 25010, and then customized it to generate the most appropriate model to reply to our needs in automotive domain.

5. Threats to validity and discussions

This chapter described a quite long work of investigation performed on quality models in current literature. This investigation was possible thanks to the online available libraries that allowed us to retrieve and get access to a plethora of candidate papers during our systematic literature review. Furthermore, the snowballing adjunction to the review has assured us a successful harvest of 492 quality models. Nevertheless, this lengthy investigative work and the corresponding contributions may have been impacted by several threats.

Firstly, concerning literature bias, we didn't really consider the grey literature (i.e., the material and research published outside the traditional commercial academic publishers). We use Google Scholar as a complementary

source to find some further candidate study papers, but in a minor way, and certainly not as Haddaway *et al.* [223] described about the role of Google Scholar in evidence reviews and grey literature searching. Rather, we decided to use traditional commercial academic publishers because of the mandatory peer reviews to publish papers through this type of media channel, and which automatically bring a certain level of scientific peer recognition of the contribution. On the other hand, during our paper screening, we didn't retain any study paper in the range of 1980-1993 (see Figure 32). At last, this was not a blocker to our survey, but as a potential improvement to fill that gaps, the grey literature could support us to identify any additional suitable contributions within that range.

The 136 study papers were the first elements of our quest for quality models. Thus, for each reference about quality models in those papers, we looked for retrieving the original reference papers. However, sometimes it was not possible to retrieve or access to all the documents, even, the ones recommended by the authors and consequently we missed some models. For example, this is the case with April *et al.*'s study [224] where some of the quality models indicated by the authors were not available anymore. So, our choice was to exclude by default any quality model for which we couldn't get the original publications otherwise we could not confirm that the model is really a quality model, and then collect multiple data about them for a later usage and the software quality model landscape creation.

In addition, each time we found an original reference publication about quality model, we scrutinized the referenced papers to find further quality models. This was the snowballing part in our systematic literature review. We include also as valid quality model definition, the results of surveys where we have list of quality attributes due to the nature of this type of quality model where the aim is mainly to describe or define the key quality characteristics, or attributes. This the case for instance of Åkerholm *et al.* study [64] on the most important quality attribute for vehicular software.

At the opposite, we exclude models for cost prediction (e.g., COCOMO [151]), for measure (e.g., COSMIC [158] on functional size, IEEE 1045:1992 [212] on productivity, IEEE 1061:1998 [40] on metrics methodology) even if studies like the Kläs *et al.* CQML classification [10] include them as valid quality model. We decided to not include these models because they don't directly define, assess, control, or predict quality but they may influence quality. Moreover, we didn't include quality model for information quality (e.g., Knight and Burn IQIP quality model [226]) since their focus is on data quality rather than software quality, despite the fact that quality of data may be linked also to the quality of software. This is the same for Delone and McLean IS success model [44] which includes quality aspect, but this model aims to predict the effect of Information System against the organization performance and not on software quality.

Another potential threat is about the analysis performed on each quality model, and more particularly on statistical or implicit quality models. It appears sometimes that the data mining method combine different approaches together and therefore it could be challenging to identify accurately the right main method. Moreover, our analysis was manual and because of the high volume of data to analyze, possibly some mistakes could happen. We trust that their number is minimum since we crossed check the extracted from multiple perspective (e.g., from other authors, during a second reading or data collection).

Lastly, concerning the appliance of our cladistic-based classification method, we didn't use all the defined taxa. The reasons are the manual treatment of a high volume of data and the lack of proper tool to support and handle that work. Hopefully, we dispose of all the software quality model reference sources and, once an adapt tool is available, we can perform the full classification with all taxa and homology (see Figure 39). Furthermore, this tool should provide an online portal with free access to the quality model list, the classification elements, and allow to the community to contribute such an open-source project to maintain and consolidate that list. We must gather as well this cladistic method as an open-source structure jointly with the corresponding manipulation library otherwise this research work will become deprecated and unused.

Chapter VI. Quality Model Operationalization

1. Introduction

As we saw in our study on quality modeling applied to embedded software development, in Chapter II.3, often quality model solutions are not, or cannot be operational and use as they are defined. The quality model operationalization requires a particular attention that is usually neglect but it is possible to find contributions where one of the focus is to achieve an operational quality model. This is the case, for example, of Ahrens *et al.* study [51]. Thus, it follows the purpose of this chapter, summarized through the research question 3:

Research Question 3

Considering a quality model for a software product, how to operationalize it?

By operation, we aim the development and use of quality model. So, to answer to this question, we must first understand what type of issues, if any, can prevent development and use of quality model. We are rephrasing this under the sub-questions 3a: *What are the main challenges and the issues that prevent operationalization of quality model?*

Once the issues and the challenges identified, we then need to identify the proper solutions to resolve, or at least to work-around, them. Nevertheless, we must avoid too general solutions which won't bring any solutions but rather bring more questions or issues. So, we must identify practical solution (i.e., solutions related "to actual experience or to the use of knowledge in activities rather than to knowledge only or ideas" [227]) either from the reuse of existing academic or industrial solutions, or from the creation new and tailored solution. This leads us to our next research sub-question 3b: *What are the practical solutions to those challenges and issues?*

Finally, the final step is to have a repeatable and systematic way to operationalize, based on the findings of 3b, a software quality model. To achieve repeatable and systematic behavior here, we must define a process that handle the operationalization (i.e., development and use) of a software quality model. Therefore, we land the research sub-question 3c: *What is the process to ensure quality model operationalization?* In the following sections, we are addressing each of these questions sequentially because of their respective dependencies: 3c depends on 3b results, which depends on 3a results.

2. Operational challenges and issues with quality model

Research Sub-question 3a

What are the main challenges and the issues that prevent operationalization of quality model?

In 2012, Thapar *et al.* performed a comparative study on quality models with the aim to identify "what challenges are posed by quality models" [11]. They first identified a group of 24 quality models that they considered significantly representative for the purpose of their study. They categorized the quality models that emerged until 2001 as basic quality model, and the later quality model as tailored quality models because these models were basic quality models that were modified or completed to answer to the increase of software industry needs on quality evaluation and improvement. We note that the authors' basic quality model set was made of five quality models that we also referred to as key contributions to quality modeling of software: Boehm's quality model [42], McCall's quality model [41], Dromey's quality model [49], FURPS' quality model [85] and ISO/IEC 9126 quality model [24]. Then, Thapar *et al.* investigated the main obstacles, or challenging issues which impede the development and practical use of quality models before applying these findings to compare the 24 models together. Thus, the authors identified 9 main potential issues. These are:

- **Association:** The software development process and quality model are not, or not sufficiently, formally associated together, thus affecting the implementation of quality characteristics and measures. For example, in agile (i.e., incremental and iterative software development aligned with the agile manifesto [228]) and rapid (e.g., Scrum, Extreme Programming) software developments, often being customers centric, mis-association or lack of involvement with quality modeling may results, for instance, in late consideration of quality characteristics and requirements, in business

value priority versus quality, in lack of awareness quality characteristics and requirements by customer or DevOps team.

- **Evolution:** Limited evolution of quality models may lead to them becoming obsolescent, prevent their being adapted or their scaling during the development process. This is often the case statistic quality models, for example, since they are designed from specific data set. Another example is Dromey's quality model which is quite abstract and limited within its four quality characteristics (i.e., correctness, internal, contextual, and descriptive) to evolve easily.
- **General:** Quality characteristics and sub-characteristics of quality models are too general to be applicable as is, or too specific to be applicable to multiple use cases. This is often the case with standards (e.g., ISO/IEC/IEEE 25010) which tend to cover a large set of possibilities; their drawback is that their quality characteristics and sub-characteristics are too general. Wagner *et al.* conducted a survey [83], [98] on quality model in international German companies and they found that 79% of the companies that rely on quality models taken from standards have customized these models to use them.
- **Guidelines:** Quality model guidelines or documentation are frequently not complete enough to enable stakeholders, including software architects and developers, to understand or use the specified quality model. For instance, the Garcés *et al.* quality model for ambient assisted living systems [56] is not sufficiently detailed to be used because the authors explained only its main definition and principles. At the opposite, McCall [41] and ISO/IEC 9126 [24], [146], [155], [156] quality models are two examples described with enough definition, details and context explanation to be deployed and use in software development.
- **Maintainable:** Quality models which cannot be maintained regularly may greatly impede development due to having to fix defects (e.g., wrong quality sub-characteristic decomposition, or incorrect metrics) or integrate new requirements. We saw in Chapter IV.2.b with Kano's model [58], quality perception like stakeholder quality requirements changes over times and therefore quality models require regular changes to avoid becoming quickly obsolete or reflect a wrong view of current quality. Moreover, under the maintainable scope we consider another aspect not covered by Thapar *et al.* This is linked to configuration management and versioning of quality model. Indeed, transportation system regulation (e.g., automotive, aeronautic) requires having mechanism to keep and recover configuration elements of produced systems. This includes not only documentation, source code, test material, development tools, to cite few, but also quality modeling material.
- **Risk-Driven:** The risk-driven aspect is limited or missing in the quality model, and therefore makes early risk mitigation difficult to achieve. In development, for example, this is particularly important with the case of quality characteristics describing non-functional requirements (e.g., usability, security, reliability, or performance efficiency). Indeed, they are often deferred at a later stage of the development process, and consequently their achievements may be in difficulty if the related quality modeling is not connected with risk management.
- **Stakeholders:** Insufficient involvement, or participation of stakeholders in quality model development, quality evaluation and quality framework challenge the quality model buy-in by stakeholders, as well as the correct quality perception and expectation of final customer. In *Rapid* development (e.g., extreme programming), for example, customer works closely with developers to foster communication, feedback, and shorten loopback on what is being developed to successfully match customer expectations.
- **Subjective evaluation:** Lack of objective quality evaluation, due to a lack of sufficiently detailed and complete metrics, results in a less than objective assessment, as Ahrens *et al.* highlighted also in their survey [51]. This is also the case where the quality assessment chain, including metrics, is partially or not automated. Indeed, automation allows systematic reproducibility of outcomes, increasing the objectivity. At the opposite, a lack of automation is a source of preface to subjectivity. However, subjective evaluation happens in *Rapid* software development too, where continuous integration and delivery rely on automated building, deployment, and testing capability. The cause

of this subjectivity is the time constraint with short iterations which result in limited testing (i.e., in assessment or control activity), for example [181].

- **Validation fairness:** The idea behind this quality modeling challenge is the fact that we cannot be judge and party. Thus, once a quality model is developed, it must be validated by independent experts - ideally-, otherwise the lack of independency in the quality model validation shall result in biased quality assessment, control, or prediction.

In Thapar *et al.* studies, the 24 quality models were thus compared to each other based on the number of issues found in each model. The study concluded that at least three issues - not always the same ones - were found in each of these quality models, and five of them only had three issues. The short list of those with only three issues includes ISO/IEC 9126 [24], GEQUAMO [229], Bawane [103], Alvaro [36] and Kalaimagal [102].

Although the authors detailed for each issue its rationale and an idea of solution, they did not evaluate the relative importance of the issues, nor did they put forward practical solutions for meeting the corresponding challenges. So, taking the data of Thapar *et al.*, we analyzed the frequency of occurrence of the 9 issues with this set of 24 quality models – see Figure 52. We found that the three most frequent issues (risk-driven, association and validation fairness) accounted for ~49% (cf. Figure 52) of the issues and, when we extended that list with the next three most frequent issues (guidelines, evolution and maintainable), we found ~86% of the challenges. These results mean that by solving risk-driven, association and validation fairness issues, ~49% of the challenges that impede the development and use of quality models can be met; further, by also addressing the next three challenges, ~86% in total of those issues were covered. But, according to the analysis of general, subjective evaluation and stakeholder, these issues did not seem to occur often, which is surprising because the two last issues occur more frequently in reality.

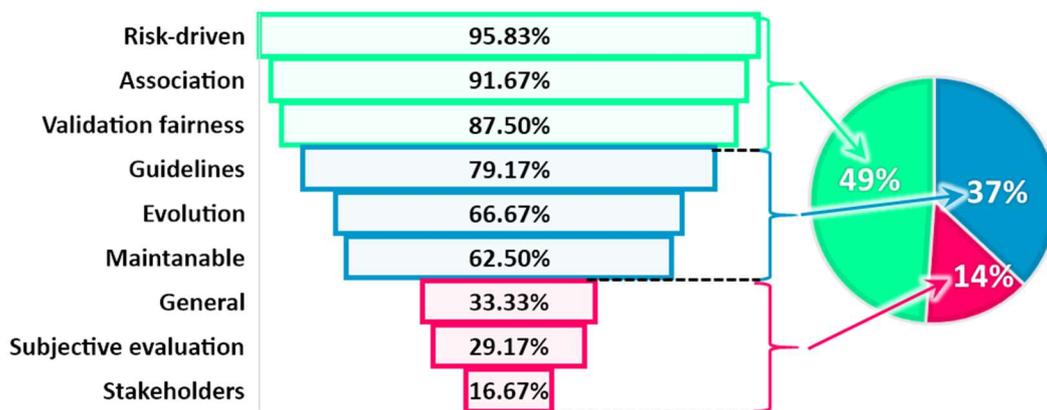


Figure 52 - Frequencies of the 9 main issues identified by Thapar *et al.* [11], which impede the development and use of quality models

According to our readings and experience, these 9 potential issues are not the only main ones to be taken into account. Indeed, when modeling quality, it is critically important that all new or reused terms, concepts and associated definitions be unambiguous, agreed upon and understood by all stakeholders.

Moreover, attention needs to be paid to redundant or duplicated elements because they usually result in rephrasing, thus increasing the risk of inconsistency and making knowledge more complex. These are terminology and redundancy issues. There are references to them in some of the harmonization issues with ISO-based quality models that were identified by Abran *et al.* [96]. So, to summarize:

- **Terminology:** Lack of unambiguity, consistency, agreement and understanding for reused terms, concepts, and their definitions. This issue occurs particularly when standard glossaries (e.g., ISO/IEC/IEEE 24765 International Standard Systems and software engineering—Vocabulary [117], the international vocabulary of metrology [134]) are ignored in favor of proprietary definitions or company jargon.
- **Redundancy:** Redundant or duplicated elements in quality model that usually result in rephrasing. For example, over the ISO/IEC SQuaRE series, there are duplication of terms and definitions mainly defined in the ISO/IEC 25000 [25] and ISO/IEC/IEEE 25010 [23] which is “contrary to the ISO practice” as Abran

et al. [96] pointed out. Indeed, ISO/IEC/IEEE 25010 and ISO/IEC 25012 [155] duplicate terms and definitions from ISO/IEC 25000, ISO/IEC 25020 [121] duplicates terms and definitions from ISO/IEC 25000 and ISO/IEC/IEEE 15939 [118], ISO/IEC 25021 [160] duplicates terms and definitions from ISO/IEC 25000, ISO/IEC 25010, ISO/IEC 25020 and ISO/IEC/IEEE 15939, ... Additionally we remark that some duplications are performed jointly with an adaptation of the original definition (e.g., it is written in ISO/IEC 25021 that definition of external measure of software quality is adapted from ISO/IEC/IEEE 25010 definition). Nevertheless, duplication and rephrasing can be sometime saving. In Boehm's quality model [42] some quality sub-characteristics are linked to two or three quality characteristics (e.g., accessibility is linked to efficiency, human engineering and testability ; see Figure 14). Duplication of these sub-characteristics surely helps to have a proper description of them based on the characteristic contexts.

Within a predictive quality modeling scheme, Khoshgoftaar *et al.* [173] performed cost-analysis on several predictive quality-modelling software. They showed that accuracy and reliability - based on the effectiveness and efficiency of quality models - are significant factors or issues as well.

- **Accuracy:** Lack quality model of accuracy, including in repeatable prediction results, where the quality model results are not in the defined precision range and therefore produce invalid results.
- **Reliability:** Lack of quality model effectiveness or efficiency causing diverged results over repeatable uses of quality model within the same conditions (cf. reliability in measurement in Figure 15).

Synthetizing both our research and our practical experience in quality modeling, we identified three other issues which also affect the development and practical use of quality models.

First is their maturity. Maturity is not only a matter of age or youth of the quality model. It is a factor which combines the intensity and history of model use, the development context (research vs. academic vs. industrial), its application for an actual use case, its review history, its evolution, and revision history.

The next issue is expertise in quality modeling knowledge and activities. Without expertise, or expert people the development and deployment of quality model is uncertain or at least shall suffer from lack of properly answering to quality requirements. As well, a loss of expertise, or familiarity, due to a loss of competency or an unmastered complexity increase in quality modeling leads to the same consequence. This issue can be therefore assimilated to the fifth Lehman's law of software evolution, "*Conservation of Familiarity*" [82].

Moreover, the importance of quality modeling may be underestimate or even overlooked. This leads us to the final issues: neglect.

The neglect issue is the most critical one. Indeed, if quality modeling or quality model is purely and simply neglected, ignored or underestimate then no quality modeling can happen. This is therefore the most critical and first issue to solved.

- **Maturity:** Lack of quality model fully grown, developed, reused, and consolidate. A good illustration of mature quality model is the ISO/IEC/IEEE 25010 quality models. These matured models are not only evolution of the ones from ISO/IEC 9126, but also the results of the ISO document life cycle process, involving international academic and industrial participants.
- **Expertise:** Lack of expertise in quality model development or use may inhibit quality modeling, evaluation plan, measurement, quality assessment, control, or prediction activities. Likewise, without security or safety expert, for instance, security or safety quality requirements implementation, testing - manual or automated- and compliance to standard are extremely difficult tasks.
- **Neglect:** Lack of consideration of quality model, neglecting consciously or unconsciously quality modeling activities is the most important threat to quality modeling. In Behutiye *et al'* systematic study on the management of quality requirements in agile and rapid software [181], the authors identified several causes of the neglect of quality requirements, and consequently quality characteristic and modeling, by agile software development teams. We can cite for example those teams focusing on shorten delivery time (i.e., implementation and validation of quality requirements may be time consuming) and on functional requirements, using of existing internal infrastructure, or waiting for a request or need to proceed on the quality requirements.

To conclude our exploration and analysis on issues preventing operational quality model development and use, we end up with a list of 16 potential issues that are summarized in TABLE 20, by alphabetical order.

TABLE 20 – ALPHABETICALLY SORTED LIST OF MAIN POTENTIAL ISSUES THAT CHALLENGE DEVELOPMENT AND USE OF QUALITY MODELS

Id	Name	Description
01	Association	Lack of association between software development process and quality models
02	Accuracy	Lack quality model of accuracy, including in repeatable prediction results
03	Evolution	Lack of quality model evolution possibility, influencing its obsolescence, or preventing its adaptation during development process
04	Expertise	Lack of expertise in quality model development or use
05	General	Quality characteristics and sub-characteristics too general to be applicable, or too specific to be generalized to use cases
06	Guidelines	Lack of quality model guidelines or documentation
07	Maintainable	Lack of quality model maintenance possibility
08	Maturity	Lack of quality model fully grown, developed, reused, and consolidated
09	Neglect	Lack of consideration of quality model, neglecting consciously or unconsciously quality modeling activities
10	Redundancy	Redundant or duplicated elements that usually result in rephrasing
11	Reliability	Lack of quality model effectiveness or efficiency
12	Risk-Driven	Lack of risk-driven aspect in quality model
13	Subjective evaluation	Lack of objectively quality evaluation, enough detailed and sufficient metrics
14	Stakeholders	Lack of stakeholder participation in quality model development, quality evaluation and quality framework
15	Terminology	Lack of unambiguity, agreement and understanding for reused terms, concepts, and their definitions
16	Validation fairness	Lack of quality model validation fairness

3. Practical solutions to the operational issues

Research Sub-question 3b What are the practical solutions to those challenges and issues?

In previous section, we have identified 16 obstacles that prevent operation of software quality model (i.e., development and use). Thus, in the following paragraphs, we are exploring how to resolve or workaround practically these obstacles whenever it is possible, addressing then the research sub-question 3b: “*What are the practical solutions to those challenges and issues?*”.

By practical solution, we mean solutions related to experiences, real situations or actions that are possible to reproduce, reuse or deploy. Thereby, we aim to avoid enumeration of hints or ideas like in Thaphar *et al.* study [11], which lead to inefficient resolutions because they require a non-negligible effort to make them in practice, even when these ideas are good sense.

Hopefully, in sections from Chapter IV.3 to Chapter IV.6, we already explored quality modeling and qualimetry technologies. So, we decided to take advantage of them since they are well documented, studied, and proven practice.

- **Association:** To address this issue, the practical solution must aim to foster the association between quality modeling and software development process. Grady and Caswell's FURPS [85] and then FURPS+ [152] quality modeling approach is a good illustration of what must be achieved to meet this challenge. Indeed, they took into consideration the software development life cycle in the definition and instantiation of their quality model, keeping for the same quality characteristics over the different development life cycle stages but they associated a specific set of metrics to each stage. The interest with that approach is that quality modeling remains constantly aligned with the current software development process stage. An example is shown in TABLE 11.
- **Accuracy:** As we indicated previously this type of issue is similar to the accuracy measurement problem exposed over Figure 15. So, the *accuracy* issue needs to be addressed by measuring, analyzing, and then making the right adjustments and enhancements to quality models. Such practical solutions may be inspired by the method of Khoshgoftaar *et al.* [173] with regard to accuracy aspect. The authors used *test* data to evaluate the accuracy of their model, and after analyzing the results, they made decision on the adjustments to propagate to their quality model.
- **Evolution:** Qualimetry offers a solution for meeting the *evolution* challenge. Its polymorphism mechanism [230] (see also Chapter IV.6.b) enables built-in adaptation and evolution in quality modeling, allowing smooth and natural evolution of quality modeling, deriving new quality models from previous ones without breaking existing overload or inheritance behaviors. Furthermore, the polymorphic quality model simplifies the building and using of quality models independently from their general characteristics and the quality modeling target.
- **Expertise:** The practical solution for this issue is first linked to people management. Thus, recruiting and developing people expertise in quality modeling is fundamental as well as retaining the talented people [231] in that domain. An organization should be able to count on some experts that can drive the quality modeling strategy, even on data science, and disseminate the knowledge over the organization. The second element is to control or maintain the mastery of the quality modeling overall complexity. This means that evolving development and use in quality modeling must be achieved with the respect of the fifth Lehman's law, "*Conservation of Familiarity*" [82]. Moreover, if we follow the Agile Manifesto [228], "*Continuous attention to technical excellence and good design enhances agility*". So, quality modeling expertise cannot be skipped even in the modern software development methodology like Scaled Agile Framework (SAFe) [232] where cross-functional agile teams concentrate all type of expertise around the customer.
- **General:** Likes for evolution the polymorphism mechanism provided by qualimetry allows to derive polymorphic quality models from general quality model or quality characteristics that are overly general, independently of the quality modeling target. An example of practical build of polymorphic quality model is given in next Chapter VII.5.
- **Guidelines:** Concerning *guidelines* issue, the solution is to follow the ISO 9001 recommendation or guidelines [233] stating that it is critical to document everything with the right level of detail, for instance. Moreover, the documentation must be versioned and easily accessible to the entire organization using the dedicated and appropriate document management tool.
- **Maintainable:** To strengthen the *maintainable* aspect of quality model, the Miyoshi & Azuma [234] recommended that the number of key factors, or characteristics should be kept between three to eight. We find the same guidelines in qualimetry, where Azgadnov *et al.* [113] referred, without citing it, to Miller' study in experimental psychology [235] about the human capacity for processing information. Thus, the recommended number is "*7±2*" and it is nicknamed "*the magic number*" or "*Miller's law*". In both cases, we noted that these numbers are quite similar. Additionally, compliance with tree derivation rules [38] (cf. TABLE 8 and TABLE 9) for controlling the analysis of quality characteristics, their organization and the quality model complexity, formalized in qualimetry, fosters a high level of maintainability in the case of hierarchical models. In parallel to these quality model attributes, a proper level of technical documentation, aligned also to the *Guidelines* issue, is an important factor influencing the maintenance capability such as documented source code can do.

- Maturity:** Due to the nature of *maturity*, this issue deserves a specific focus since it evolves over time, and depends greatly on the changes, or decisions made in the quality modeling which should thus be done carefully, while anticipating the next steps as in a game of chess. However, by applying a capability maturity model such as CMM [154], CMMi [236], [237], or A-SPICE [21] to quality modeling activities (i.e., as a process, and to quality model characteristics, sub-characteristics and metrics), we can assess, control, and enhance maturity, and consequently address this issue. Since our industrial context in automotive, we recommend A-SPICE and its 5 capability levels shown in Figure 53. Highest level means highest maturity, and in our case, level 0 indicates that there is no quality modeling performed.

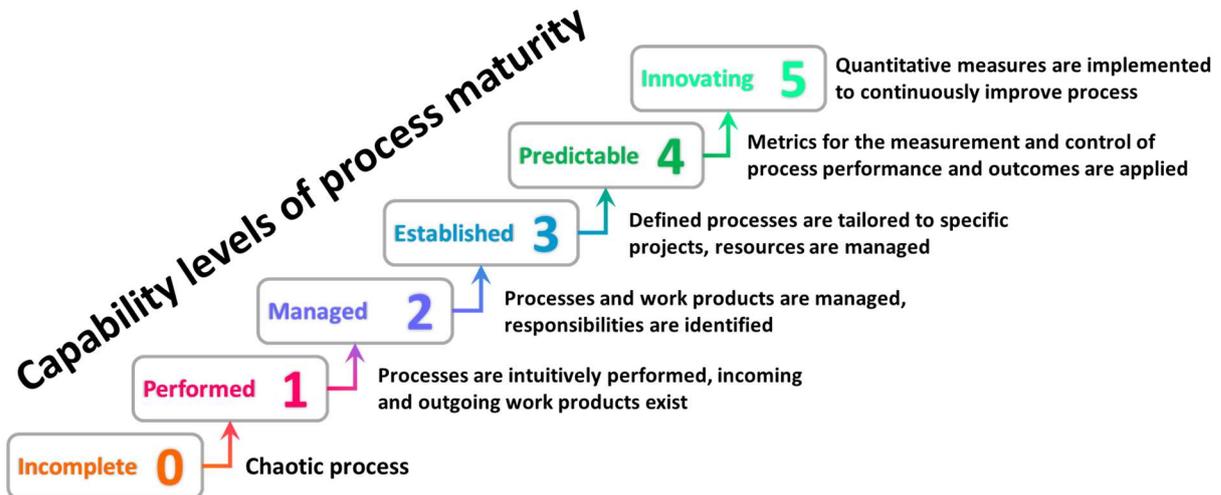


Figure 53 - The A-SPICE capability levels of process maturity [21]

- Neglect:** This issue is complex to solve without a strong and unwavering support to quality modeling, or qualimetry, from the academic and industrial leaders. Therefore, role modeling, showcasing of big win examples thanks to quality modeling (e.g., industry success in certification, academic student evaluation), and proper explicit communication to software engineering and systems engineering teams are the right tools for practical solution. One of the Agile Manifesto principle [228] is “*Continuous attention to technical excellence and good design enhances agility*”. *De facto*, applying it means that even quality modeling shouldn’t be neglected in software development, especially with agile and *Rapid* software development methodology where quality requirements are often neglected or overlooked, and consequently
- Redundancy:** Following the ISO practices which advise to avoid duplication, it obviously makes sense to be careful, rigorous and keep in mind this advice to limit redundancy when developing and using of quality models. Moreover, factorization of concepts, terms, quality modeling elements (e.g., quality characteristics, perspectives, metrics, quality model) jointly with references to standards such as the international vocabulary of metrology [134], for example, decrease the risk that redundancy issue occurs.
- Reliability:** The *reliability* issue resolution is close to the *accuracy* one. Thus, it needs to be addressed by measuring, analyzing, and then making the right adjustment and enhancements to quality models. Khoshgoftaar *et al.* [173] method with regard to reliability aspect can also be considered. However, there is a difference with the practical solution for accuracy. Indeed, to address *reliability* issue, we should rely on automation to ensure repeatable quality model results.
- Risk-driven:** Inspiration for resolving risk-driven issues can be found by combining ISO/IEC 25010 [22] and ISO/IEC 25022 [74]: the “*freedom from risk*” quality characteristic, its sub-characteristics (both defined in ISO/IEC 25010) and their measurements (defined in ISO/IEC 25022) demonstrate how the risk aspect can be included in quality modeling. The ISO/IEC 25022 standard is of major importance because it describes how the risk is managed with two quality threshold levels. This is risk versus opportunity associated with the level of quality (see Figure 18). If the measured level of quality is below the lowest threshold, the risk is unacceptable, or not manageable. If the measured level of quality is between these two thresholds, then the risk is acceptable or manageable. Finally, if the quality level is above the highest

threshold, it is providing a development opportunity. In fact, by generalizing the use of these two quality threshold levels instead of a single quality threshold level (e.g., acceptance threshold), we automatically encompass the risk aspect in our quality modeling.

- Subjective evaluation:** Regarding *subjective evaluation*, a complete metrics specification with enough detail to apply them is required, and ideally most of them should be automated. We can use Boehm [42], McCall [41] or ISO/IEC 9126-2,3,4 [104], [105], [106] as beginning source of information for metrics. Then we have to select and apply correctly aggregation operators. There are many different types of aggregation operators [143] (e.g., arithmetic mean, weighted mean, symmetric sum, k-order statistics) and their choice depends on the purpose for which they are intended, according to Wagner [27] (e.g., assessment, prediction, comparison, hotspot identification and trend analysis). Chapter IV.3.d and TABLE 39 from Annex 1 give further details on aggregation topic. An alternate aggregation approach is the “Logic Scoring of Preference” of Dujmovic and Bayucan [144] shown in Figure 17 and by the equation 3. The main idea is to specify the relationships between the inputs and the output based on trade-off between conjunction, neutral and disjunction relationships. Concerning the inclusion of quality characteristics and sub-characteristics relationship, Khaddaj and Horgan [148] established quality characteristics by means of a factor polarity profile (see TABLE 7) with direct, neutral and inverse relationships. From another perspective, Perry’s quality control checklists [238] can be adapted to have objective check procedures to quality modeling activities. Finally, to shorten continuous integration, testing and delivery, *agile* and *Rapid* software development method encourage automation which contributes as well as reducing subjectivity in the evaluation with repeatable and constant method to evaluate quality.
- Stakeholders:** For this issue, all stakeholders must be involved as early as possible to build or customize a quality model that takes all stakeholders' visions into account in accordance with Horgan's views [84]: “*Key Quality Factors*” are those quality factors which are relatively invariant per project, i.e., product and stakeholders, while “*Locally Defined Factors*” are the quality factors that change, depending on the project, product, or stakeholders. Over the “*Quality Assessment Algorithm*” of qualimetry [113], expert and non-expert stakeholders participate in quality characteristics, sub-characteristics, and weight factor determination through meeting sessions or surveys. Thus, since the stakeholders are part of the quality model development, their vision is shared and aligned with the resulting quality model.
- Terminology:** Like for redundancy, to reduce *terminology* issue, it is also vital to rely on terminology standards likes the international vocabulary of metrology [134], ISO/IEC/IEEE 24765 International Standard Systems and software engineering—Vocabulary [117], or the international software testing qualifications board vocabulary [111], to cite few. These standards ensure that a set of common vocabulary and concepts are clearly specified and shared broadly. Concerning other than standard vocabulary or concepts used in quality modeling during development or use, the solution consists in ensuring compliance of the terminology with the “*five C’s*” of requirements engineering (i.e., correctness, completeness, consistency, conciseness, and clarity) [9].
- Validation fairness:** Resolution of *validation fairness* issue may be achieved thanks to qualimetry science and its “*Quality Assessment Algorithm*” [113]. This algorithm guides quality model developers and helps them assess quality models by upstream involvement of experts and non-experts through technical, expert, and steering groups. A particular attention must be made with regard to the independency of involved parties. This validation with expert can also be completed with a joint validation between empirical results (e.g., results from the use of quality model to assess quality of a certain product) and expert (e.g., results from validation expert who assesses quality of the same product that empirical one). This approach was applied by to validate the QMOOD quality model they built [218].

So, we identified and proposed a complete set of practical solutions for the issues that may prevent the operationalization of quality models. Figure 54 gives an overview of those practical solutions mapped to the corresponding issues. Next section focuses on the construction of two processes that take advantage of these solutions to guide efficiently development and use of quality models in quality evaluation.

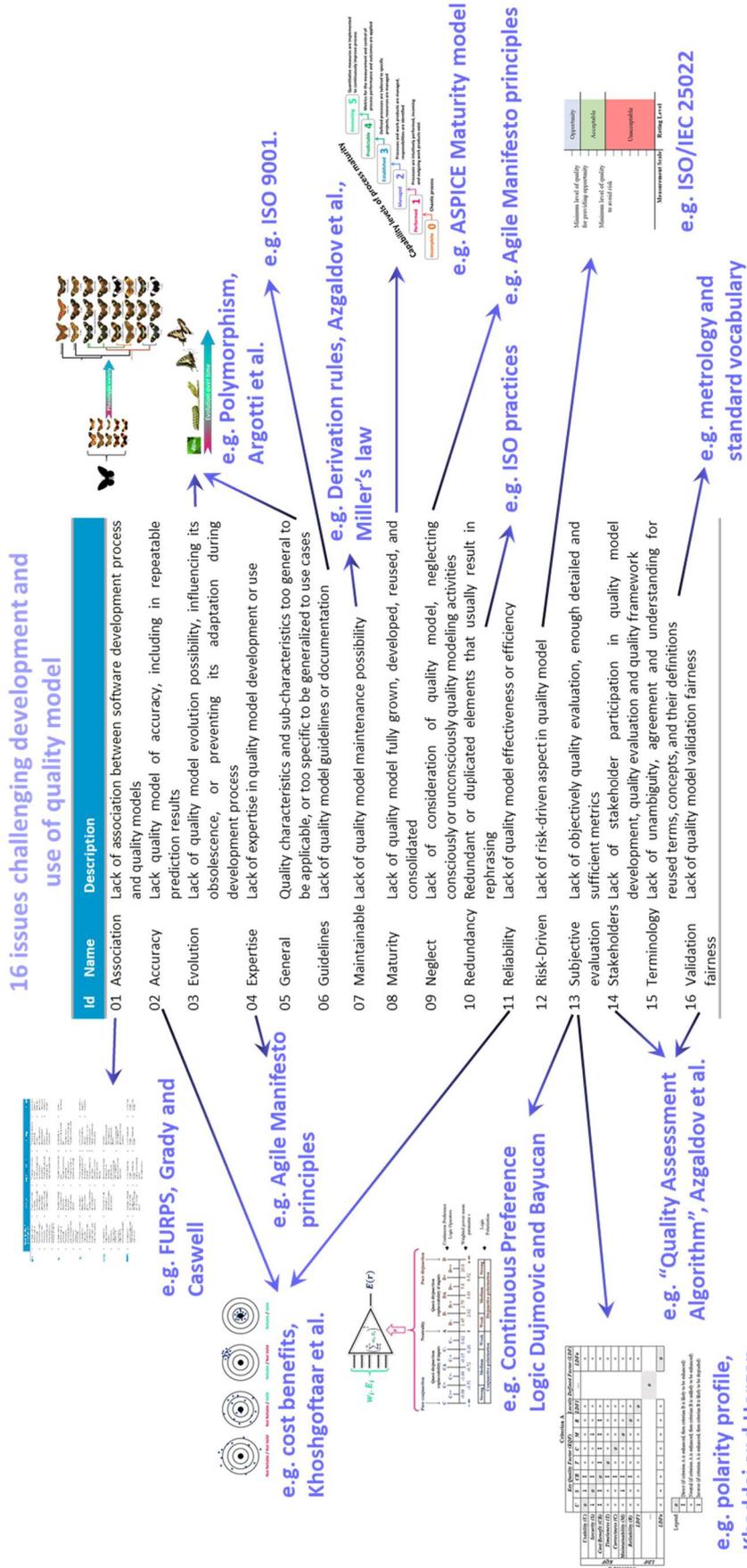


Figure 54 - Mapping of practical solutions against the 16 issues preventing development and use of quality models

4. Operational Contributions

Research Sub-question 3c What is the process to ensure quality model operationalization?

After the analysis and finding of 16 issues that may prevent development and use of quality models, and then the identification of practical solutions for each of them, we now explore how we can take benefit of them during quality model operations. Thus, the result of this exploration responds to 3c: *“What is the process to ensure quality model operationalization?”*.

We remark that the definition and deployment of such process linked to quality model development and use is aligned to the process maturity capability level 2 (i.e., Managed) expectations (see Figure 53). Moreover, the tailoring of that process to specific projects raises the process maturity to capability level 3 (i.e., Established).

a. Quality model operational use: The “Quality Thermometer”

The purpose of this process is to cover the operational development and use of quality models in a context of monitoring project progress and status, including product quality.

There are various framework solutions for the visualization and monitoring of project progress and status such as dashboards or scorecards [239].

A dashboard usually displays key performance indicators graphically, via graphics, gauges, or meters that are updated frequently (i.e., hourly, daily), but users must interpret the data.

A scorecard consists in metrics, targets, and trends, showing how the measures performed against the targets. The scorecard is usually updated on a weekly, monthly or quarterly basis, and like the balance scorecard [240], it facilitates the alignment of company strategy and project objectives.

In our case, we decided to define and use a project scorecard with the necessary details to accurately monitor status and progress for project, process and product. In addition, the division of our project scorecard (cf. Figure 55, Table 49 and Annex 7 for further details) into 6 main blocks was the result of two brainstorming sessions organized with our main stakeholders. Each block addresses a specific status area of the project.



Figure 55 - The software project scorecard including indicators and metrics for project, process, and product

Thus, the main project scorecard blocks are:

- **Project id:** main identification details about the project, the current milestone or also its safety, or Automotive Safety Integrity Level (ASIL) since we are working in the automotive domain (in aeronautics we could use the Design Assurance Level (DAL)).

- **Project dimensions:** details about project and product sizes; these are used for the purpose of normalization, weighting and/or for monitoring complex systems.
- **Project indicators:** a combination of metrics from a project (e.g., sprint or feature completion, scope creep [241]), process (e.g., lead and cycle time) and product (e.g., software product quality).
- **Quality performance indicators:** a synthesis of the conformity to A-SPICE processes and their work-products.
- **Safety & regulation performance indicators:** a synthesis of the conformity to safety and regulation processes and their work-products, with regards to applicable standards.
- **Internal and external software metrics:** a collection of internal and external software metrics linked to some quality characteristics and sub-characteristics of quality model; this collection is made up of the most frequent metrics used by the development, verification, and validation teams.

In the project indicators block, we can see the reference to software product quality which is computed thanks to a quality model, and a set of internal and external software metrics. We also note in passing that this quality model and the required metric set depend on both the project scope and the current project milestone, or life cycle stage.

However, even though the quality model usage appears to be transparent, hidden behind the unique software quality product indicator in our current case, its use must be framed by a formal process in order to be clear and consistent. As a result, we defined a seven-step process that is symbolized under the shape of the “quality thermometer” for the project (shown in Figure 56), as “taking the temperature” of the software product quality.

The description of this process is intended to:

1. **Identify the project characteristics:** We capture the current project and product definitions, the current targeted milestone, the type of indicators (i.e., leading vs. lagging) and which quality perspective(s) we aim to consider between product, user, manufacturer.
2. **Define and customize the Quality Model:** We build, or customize the right quality model, in conjunction with polymorphism to match the project characteristics (e.g., type of product, project, and stage in the product life cycle). Since the development of quality model requires multiple steps to integrate properly practical solutions for association, evolution, general, maintainable, stakeholder involvement, and terminology, we decided to build an independent and complementary process for quality model development. The detail of this process is given in next Chapter VI.4.b.
3. **Define the metrics and update the scorecard:** We first identify the software metrics set associated with the quality model characteristics and sub-characteristics, starting from the internal and external software metrics pre-listed in the scorecard, completing this list as needed. This association is done accordingly to the same approach than Grady and Caswell did with FURPS [85] and then FURPS+ [152], taking into account development life cycle impact. Afterwards, we update the scorecard block for internal and external software metrics to reflect the selected metrics set.
4. **Define relationships and aggregation:** During this stage we identify the weights and the relationships between quality characteristics, sub-characteristics, and metrics. We rely on Logic Scoring of Preference – combinations of conjunction and disjunction relationships between inputs and outputs-, polarity profile, and aggregation operators.
5. **Collect software metrics:** We now have all the elements we need to start measuring the software product quality by collecting the selected software metrics - from stage 3. We apply as necessary the polymorphism behavior regarding evolution linked to time, or more precisely, aligned to the product development life cycle stages. Moreover, we dispose of historical metric data allowing to perform any data analysis or graphical display at that step.
6. **Build quality indicator:** To compute the quality indicator, we aggregate these metrics, quality characteristics, and sub-characteristics based on the defined quality model - from stage 2-, then weights, and relationships - from stage 4.

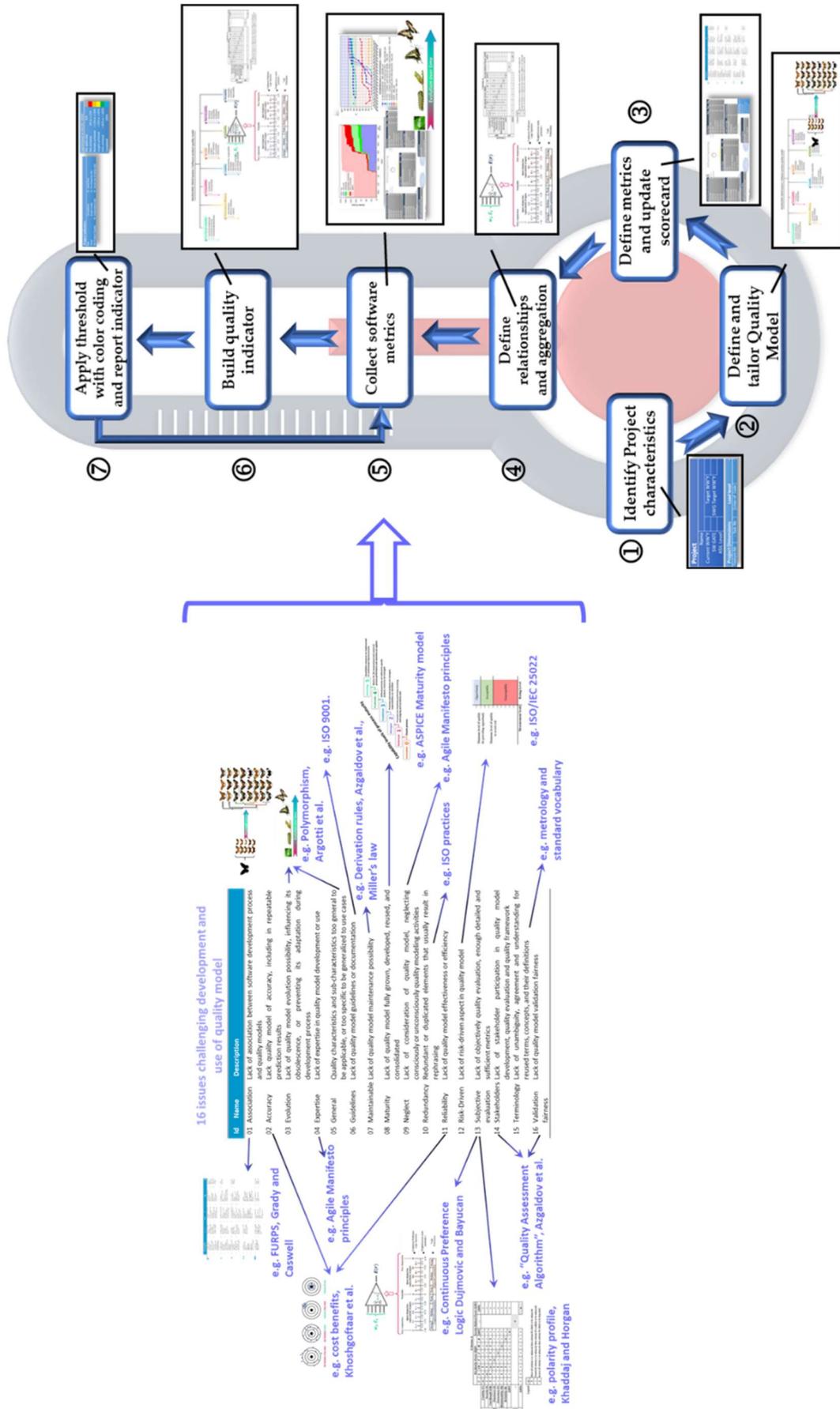


Figure 56 – Inheriting of the practical solutions to the operational issues, the “Quality Thermometer” process for project

7. **Apply threshold with color coding and report indicator:** Finally, we report the quality indicator on the project scorecard. We also use color coding associated with the range breakdown in ISO/IEC 33020 [138] (see Figure 57) with the percentage of this quality indicator over the targeted threshold. This allows us to give simple and direct visual feedback on the currently achieved quality level.

Legend based on ISO/IEC 33020	
Not applicable	N/A
Not achieved	0% to ≤ 15%
Partially achieved	> 15% to ≤ 50%
Largely achieved	> 50% to ≤ 85%
Fully achieved	> 85% to < 100%
Completed	100%

Figure 57 - Visual display of quality level done via color coding associated to range decomposition coming from ISO/IEC 33020 [141]

Epilogue: Once this quality indicator has been reported, we are ready to loop, proceeding again - until it is required - in stage 5, collecting new software metrics, building a new quality indicator, reporting it, and so forth.

Note that the scorecard gives an instantaneous view of the project status, and that it must thus be completed with a set of trend charts, displaying dynamic views, showing trendlines - linear or not-, that may be useful for prediction. In addition, we use two levels of thresholds based on ISO/IEC 25022 [142] (cf. Figure 18) – risk/opportunities - to indicate when quality is at a risk level and thus manageable or not, and when we move to an opportunity.

Finally, over this process a certain number of practical solutions have been explicitly or implicitly included to address association (i.e., at stage 3 and 5 levels), evolution (i.e., at stage 2, 3, 4, 5 and 6 levels), general (i.e., at stage 2, 3, 4, 5 and 6 levels), risk-driven (i.e., at stage 3, 5,6 and 7 levels), subjective evaluation (i.e., at stage 3 level) and maturity (i.e., at process level) issues.

Moreover, the main benefits of this approach are that it offers synthetic homogeneous views between projects, whether this be for supplier or internal development teams, as well as bridging different disciplinary teams thanks to a common vocabulary from polymorphic quality models.

The next section describes the quality model build process, completing the “Quality Thermometer” process for project.

b. Quality model operational development: a 6 stages process

The purpose of this process is to cover the development aspect of the quality model operationalization. In order to build the right quality polymorphic model, addressing then *general* and *evolution* issues, we must follow a sequence of six stages as summarized in Figure 60: identify the origin of the quality model, build a survey item list, organize the survey, analyze the survey results, construct quality models and determine which common quality model to use as a basis.

1. **Identify the origin of the quality model:** In order to initiate the development of a quality model that can be required by a project (i.e., in the scope of the “Quality Thermometer” process for project), for instance, we begin our process by identifying coarsely a quality model to serve as starting model. Indeed, with more than 50 years of active research contributions in quality modeling, there exists many valuable quality models.

Therefore, our assumption is that by fostering reuse of quality model rather than creating from scratch a new one, it allows us to better take advantage of existing contributions. So, by default we consider as original model current standard for systems and software product: ISO/IEC/IEEE 25010 quality models that cover both product and user views of quality and are conform to A-SPICE. However, this origin can be different. For example, if a company has in-house quality models, it is recommended to use them as origin.

Regarding common and specific product characteristics, the aim is to associate each of them with some quality characteristics from the original quality model if possible, otherwise to complete them. For example, “safety” can be associated with “freedom from risk” and also “security”, “human-machine

interface” with “satisfaction” and “usability”. Moreover, we model them by applying the ad hoc and universal polymorphism mechanism to the quality model.

Nevertheless, such analysis and association are not straightforward. They require a large spectrum of knowledge including the project, systems or software product, quality characteristics and quality modeling. Furthermore, the stakeholders, who depend on that quality model, must buy-in to the resulting quality model. We solved that requirement by involving the main experts and stakeholders in the quality modeling activity, and more particularly by getting them to identify quality characteristics and sub-characteristics thanks to a survey, as our next building stage.

2. **Build the survey item list:** The survey is divided into 4 stages. The first one introduced the objective of the survey and collected the contributors' roles and projects. In term of contributor audience, we should consider our main stakeholders likes project managers, architects, verification and validation leaders, and assurance quality engineers, for instance.

For the second stage, we asked to the survey participants to rank the importance of each of the quality characteristics - from the quality model(s) identified as origin and from product characteristics - on a scale from -2 (“not important at all”) to +2 (“extremely important”) based on their technical knowledge and their own vision of the product and project quality. We chose to use a 5 points Likert’s scale [242] (e.g., Figure 58) because usually all the stakeholders are already familiar with it. This scale is symmetric with the central value (i.e., point three here) considered to be the neutral value, and Likert’s items can be assigned to values (e.g., 1 to 5, or -2 to +2) to facilitate their processing and analysis.



Figure 58 - Example of a 5 points Likert's scale

The survey's third stage also focused on ranking by order of importance, but rather on all the quality sub-characteristics of the quality model(s) identified as origin and the and the product sub-characteristics. We separated the assessment of quality characteristics from that of the sub-characteristics to capture the stakeholders' perceptions of them. For instance, if a quality characteristic was ranked -2 and all its sub-characteristics were ranked +1 or +2, it means that this quality characteristic was misunderstood or was wrongly ranked. Another example could be, if one of the sub-characteristics of a quality characteristic was highly ranked compared to the other sub-characteristics, then it became the determinant sub-characteristic. Furthermore, we added a sixth ranking choice, +3 (“I don’t know”) because there were non-experts participating in the survey. This is because they might not have an opinion, or correctly understand some quality sub-characteristics which are in fact more subtle than quality characteristics. With this sixth ranking choice we avoided them biasing the overall survey results by selecting another default answer.



Figure 59 - Example of a 6 points Likert's scale

The last stage consists in an open question, offering the survey participant to suggest some missing characteristics and sub-characteristics, and share additional comments that may influence quality modeling. Note that the survey elements (e.g., questions, answer proposal, descriptions) must rely as much as possible on standard glossary and metrology vocabulary to avoid any confusion or rephrasing in terminology.

3. **Organize the survey (or hybrid method):** This approach is similar to the qualimetry hybrid method [113], where a panel of domain experts and non-experts is asked to evaluate the values of a set of quality characteristics and sub-characteristics. So, during this stage each of the targeted participants are contacted and asked to answer the survey based on their own understanding and perception of the quality. Consequently, we incite the stakeholders to participate in the quality model development, ensuring to get their implicit buy-in on the resulting quality model.

4. **Analyze the survey results:** Once the survey had been answered and its data collected, we analyze the data, our fourth building stage, by verifying the participant agreement consensus, or interrater reliability, based on Cohen’s kappa κ [243] (see equation 12) in the case of two raters (in this case the stakeholders who contributed to our survey) for a set of answers and Fleiss’ kappa κ [244] (see equation 13) for three or more raters (see Chapter VII.4 for examples and explanation on the calculations of Cohen’s kappa κ and Fleiss’ kappa κ).

For each project, we checked whether Fleiss κ was available for all roles and if it was at least greater than 0.4 (i.e., Moderate from Table 21). We thus selected a data set based on the highest Fleiss κ for each project, or product, and all roles, or where a role had the highest κ . When this was not sufficient, we used Cohen κ as a decision criterion.

If no kappa showed at least a fair agreement, then we looked for consensus at the specific role level and all projects, or products. In case of poor or slight consensus between participants, we perform another survey round, restarting at stage 2 of the process but with highlighting the areas of divergences and areas of convergences on quality characteristics and sub-characteristics.

$$\kappa_{Cohen} = \kappa_c = \frac{p_o - p_e}{1 - p_e} \quad (12)$$

$$\text{where } p_e = \frac{1}{N^2} \sum_{i=1}^k n_{i1} n_{i2}$$

p_o : the relative observed perfect agreement among the 2 raters

n_{i1} : number of times rater 1 predicted category i

n_{i2} : number of times rater 2 predicted category i

N : number of observations to categorize

k : number of categories

$$\kappa_{Fleiss} = \kappa_f = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \quad (13)$$

$$\text{where } \bar{P} = \frac{1}{N} \sum_{i=1}^N P_i = \frac{1}{N \cdot n(n-1)} \left(\sum_{i=1}^N \sum_{j=1}^k n_{ij}^2 - N \cdot n \right)$$

$$\bar{P}_e = \sum_{j=1}^k p_j^2 \quad \text{and } p_j = \frac{1}{N \cdot n} \sum_{i=1}^N n_{ij} \quad \text{with } 1 = \sum_{j=1}^k p_j$$

n_{ij} : number of raters who assigned the i^{th} subject to the j^{th} category

N : number of subjects

n : number of ratings per subject

k : number of categories

TABLE 21 - KAPPA INTERPRETATION (SOURCE LANDIS AND KOCH [245])

Kappa κ	Strength of Agreement
< 0	Poor
0.01 – 0.20	Slight
0.21 – 0.40	Fair
0.41 – 0.60	Moderate
0.61 – 0.80	Substantial
0.81 – 1.00	Almost Perfect

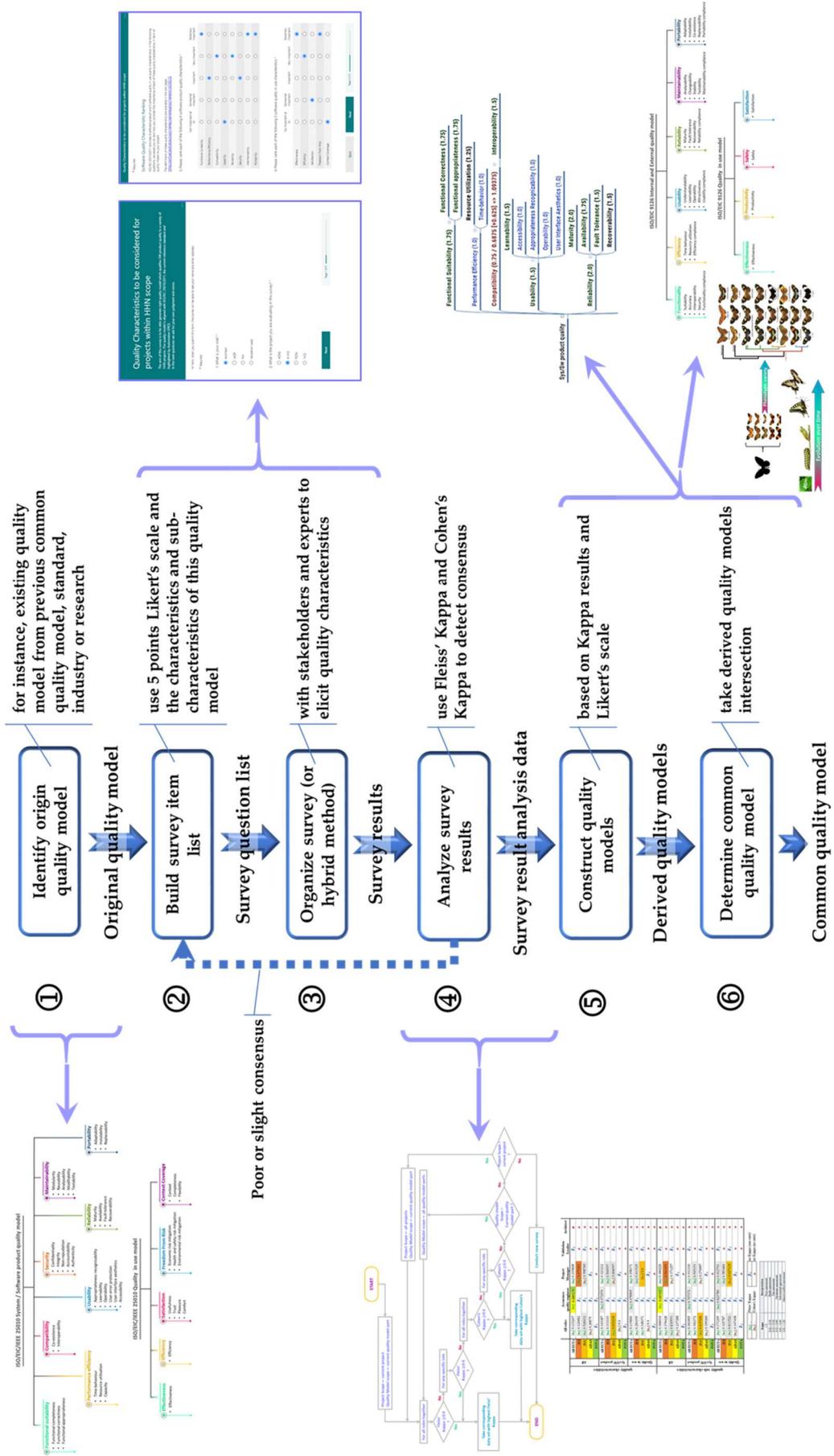


Figure 60 - 6-stages process for quality model development

So, our analysis depicted by the algorithm in Figure 61 assumes that at least two raters participated to the survey for each project. Moreover, it integrates the fact that a quality model can be composed of sub-quality model corresponding to a specific perspective. For example, with ISO/IEC/IEEE 25010, we can distinguish two quality perspective: system/software product quality and quality in use (i.e., user).

Then, for each project, and each specific perspective, we checked whether Fleiss κ was available for all roles (i.e., all raters for the project or ECU) and if it was at least greater than 0.4 (i.e., Moderate). We thus selected the data set based on the highest Fleiss κ for that project, or ECU, and all roles. If no Fleiss κ is at least moderate for all roles, we look for result for specific role. Otherwise, we used Cohen κ as a decision criterion starting from all roles and then specific role result, if any. If no kappa showed at least a moderate agreement, then we first extend our analysis to all quality perspective for the current project or ECU, applying the method describe above. If we still don't succeed to find a proper Fleiss or Cohen's κ then we look for consensus at all projects, or ECUs, level, starting from the current quality perspective with all roles and applying again our analysis heuristic. Finally, in the case where no consensus can be found at project, role, and quality perspective level, we must conduct a new survey.

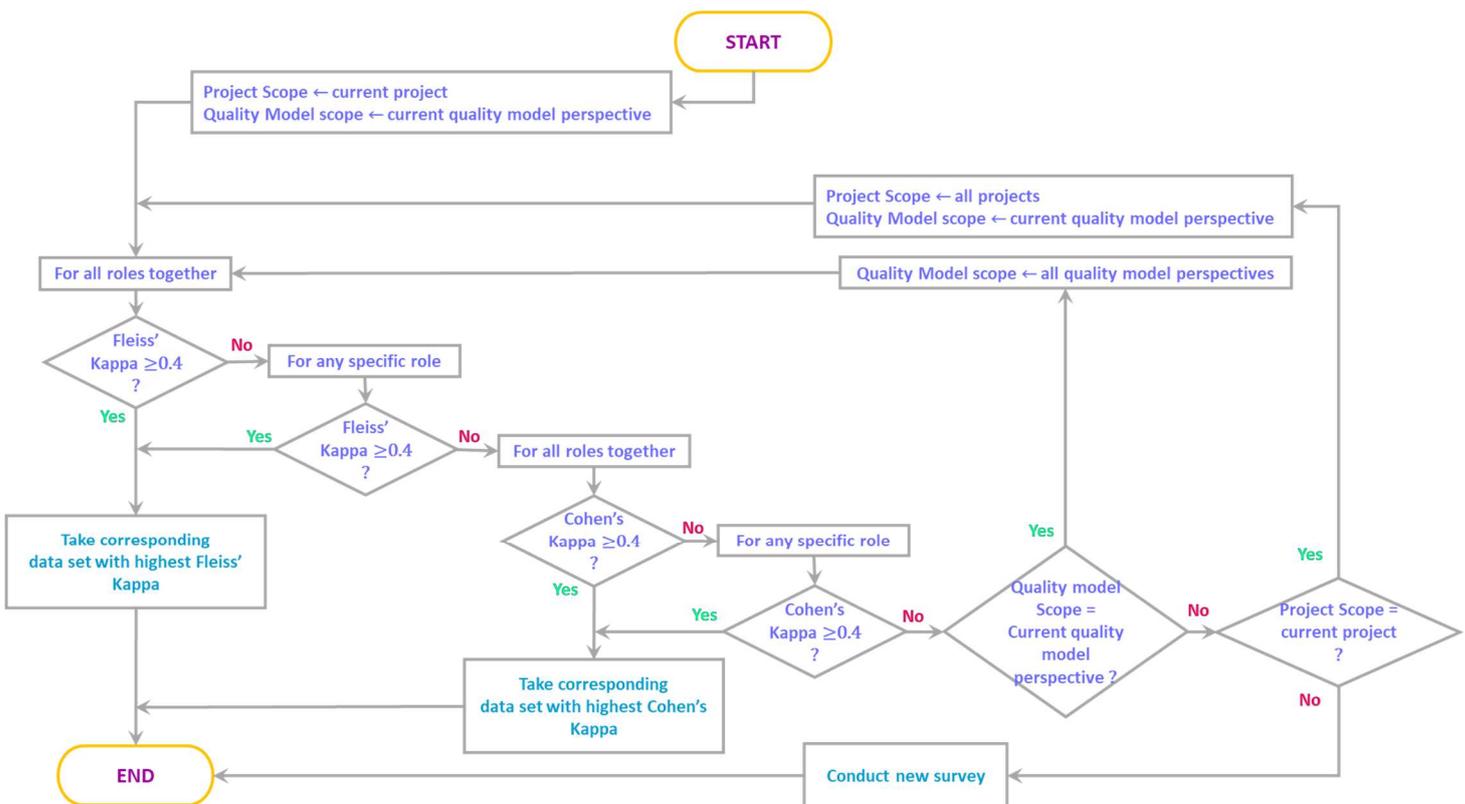


Figure 61 - Algorithm of our analysis based on Fleiss and Cohen's kappa

5. **Construct the quality models:** Once the data set had been selected, to build our polymorphic quality models we determined the most important characteristics and sub-characteristics using the scale set value set: $\{-2; -1; 0; +1; +2\}$.

First for all characteristics and sub-characteristics, we determine their importance value by calculating the mean values of their ranking values from “not at all important” (i.e., -2) to “extremely important” (i.e., +2). For example, two raters answered “not at all important” and “extremely important” for a characteristic, its importance value is: $\frac{(-1)+(+2)}{2} = 0.5$. So, all characteristics, with an importance value at least very important (i.e., +1), are included in the polymorphic quality model. Regarding their sub-characteristics, we apply the same rule and therefore include only sub-characteristics that are have an importance value of at least very important (i.e., +1).

Furthermore, for characteristics where the importance is less than +1, we look for the influence of their sub-characteristics. Indeed, if a characteristic is rejected during first step but has extremely important

sub-characteristics, then the characteristics should be finally included. To do so, we compute a new characteristic importance value by considering for a half the current importance values of the characteristic and combined with the importance values of its sub-characteristics s (i.e., complete influence of the sub-characteristics like Gordieiev *et al.* [92]), and the second half the average value of its sub-characteristics importance value that equal or greater than +1. Once a previous rejected characteristic is finally included, we apply the include its quality sub-characteristics with an importance value of at least +1. In the case where we have more than two levels of characteristics, we replicate this mechanism to the different level of quality model hierarchy.

Calculation explanation and example are provided in Chapter VII.5.a. During this construction, we take care to follow the Miller's law and the tree derivation rules whenever they apply, to facilitate the maintainability of quality model. In parallel, the polymorphic quality model construction prevents the *evolution* and *general* operational issues.

6. **Determine the common quality model:** After we model quality for several products, for our final building stage we take the mathematical intersection of these quality models to generate a common quality model. This last model will then be used as the basis for other products for which we don't already have a quality model. It can be also used for the next generation of the same products and also as a basis for building a new architecture: quality requirements drive systems and software feature requirements, and then the corresponding architecture requirements.

To conclude on this contribution of the 6-stages process for building quality models, even if this process can be used independently to the "Quality Thermometer" process, the two processes address two complementary operational aspect: development and use of quality model.

In addition, the stages of this process include practical solutions to prevent the evolution, general, maintainable, stakeholder and terminology operational issues. We notice that the elaboration of such process contributes also to reach at least a maturity level 2 (i.e., managed) for quality modeling development.

Finally, rather than a creation of a new quality model that may be optimum only for a particular project and product, we based our quality model development strategy on the reuse of existing, and valuable, quality models.

5. "Quality thermometer" and "6-stages" process comparison against current ISO/IEC standards

In the previous section of this chapter, we described two processes which addresses the development and the use of quality models. During these process elaborations, we referred to and embedded a certain number of operationalization concepts coming from standards (e.g., ISO 9001 [233], ISO/IEC 25022 [145]) but without using any of them as the backbone of our processes. And yet, we found that most of the documents in the ISO 250nn series already address various aspects of operationalizing the quality models contained in ISO/IEC 2501n (i.e.,) ISO/IEC/IEEE 25010 [23], ISO/IEC TS 25011 [246] and ISO/IEC 25012 [159]), particularly on the quality requirements and model evaluation. Similarly, ISO 15939 addresses such aspects but from a measurement process perspective.

Consequently, the objective of this section is to perform a comparison to verify the relevance of the two proposed processes against the current relevant standards which are

- ISO/IEC 250nn series [25], the system and software quality requirement and evaluation standards, that define an evaluation process using software quality models and measurements,
- ISO/IEC/IEEE 15939 [119], the measurement process standard.

The result of the comparison analysis is summarized through Table 22 and Table 23.

First, we note that in the "Quality Thermometer" and "6-Stages" process, any quality models can be used as entry model while for ISO/IEC 250nn, the aim is to use the standard quality models defined in ISO/IEC/IEEE 25010 [23] (i.e., systems and software product quality and quality in use models), ISO/IEC TS 25011 [246] (i.e., service quality models) and ISO/IEC 25012 [159] (i.e., data quality model).

Furthermore, quality model customization is natively included in our proposed approach which is not the case with the ISO/IEC 250nn, where deletion or addition of quality characteristics is not advised. This customization aspect is enabled thanks to the built-in evolution mechanism of polymorphism for product and life cycle. About the standards, that evolution aspect is lighter. Indeed, there are only 3 levels (i.e., internal, external and in use) of product quality life-cycle phase with a measurement focus of ISO/IEC 25020 [123] (i.e., the measurement reference model and guide) and the optional life cycle applicability criteria of ISO/IEC/IEEE 1593.

Regarding relationships, aggregations, and weight factors, again, they are both embedded and refined into the two proposed processes, especially their determinations are included in the “6-Stages” process. On their side, the studied standards assume that aggregation should be achieved through measurements functions and therefore, they do not address or cover them. This is the same case with weight factors where they are considered optional and manually defined without any further detail. In consequence, the result is subjective for the standard approach while we succeeded to setup an objective one in our processes.

Concerning reporting, both ISO/IEC 25040 [147] (i.e., evaluation process) and ISO/IEC/IEEE 15939 suggest some informative evaluation guidelines which is less effective and consistent than the scorecard based solution of the *Quality Thermometer*. But since the *Quality Thermometer* is close to ISO/IEC 25040, this process is therefore compatible with both ISO/IEC 250nn and ISO/IEC/IEEE 15939, and then it can benefit from them, and vice-versa.

However, even if we found several absent or limited concepts in the current standard compared to our two proposed processes, we remarked that we missed the traceability with requirement which is highlighted in ISO/IEC 25030 [247] (i.e., software product quality requirements). So, we will have to take into account and integrate this traceability concept not only in the *Quality Thermometer* process but as well in the “6-Stages” process.

TABLE 22 - COMPARISON BETWEEN THE "QUALITY THERMOMETER" PROCESS AND THE CURRENT RELEVANT STANDARDS: ISO/IEC 250NN SERIES AND ISO/IEC/IEEE 15939

"Quality Thermometer" process	ISO/IEC 250nn & ISO/IEC/IEEE 15939
Any quality models as entry with customization	ISO/IEC 2501n
Built-in evolution with polymorphism for product and life cycle	Product Quality Life-cycle Phase (Measurement Focus): 3 levels – internal/external/in use – ISO/IEC 25020 Life cycle applicability – ISO/IEC/IEEE 15939
Aggregation method and relationship definition	Hierarchical relationship - ISO 2501n Aggregation via measurement functions – ISO/IEC/IEEE 15939, ISO/IEC 2502n
Scorecard for consistent reporting	Informative evaluation report guidelines – ISO/IEC 25040, ISO/IEC/IEEE 15939
Close to ISO/IEC 25040 : compatible with ISO/IEC 250nn and ISO/IEC/IEEE 15939	

TABLE 23 - COMPARISON BETWEEN THE "6-STAGES" PROCESS AND THE CURRENT RELEVANT STANDARD: ISO/IEC 250NN SERIES AND ISO/IEC/IEEE 15939

"6-stages" process	ISO/IEC 250nn & ISO 15939
Any quality models as entry with customization	ISO/IEC 2501n
Built-in evolution with polymorphism for product and life cycle	Product Quality Life-cycle Phase (Measurement Focus): 3 levels – internal/external/in use – ISO/IEC 25020 Life cycle applicability – ISO/IEC/IEEE 15939
Automatic weight determination	Optional manual weight – ISO/IEC/IEEE 15939
Objective stakeholder consensus via computation	Subjective validation and approval of quality requirements – ISO/IEC 25030
Missing traceability with requirements	Traceability with quality requirements – ISO/IEC 25030

To conclude, this comparison confirms that, even if both ISO/IEC 250nn series and ISO/IEC/IEEE 15939 standards already cover some operationalization aspects for quality model and evaluation, the “Quality Thermometer” and the “6-stages” process are relevant and bring additional values by filling gaps in the existing standards.

6. Threats to validity and discussions

The main motivation behind the study described in this chapter is about how to develop and use quality model from a practical point of view. We observed that over the eight main approaches to support quality model development and use (see Chapter IV.5 and Table 10), qualimetry is the most appropriate approach to proceed for quality modeling. Moreover, as we highlighted previously (e.g., via the “*House of Qualimetry*”, Figure 20), qualimetry covers both theoretical and applied aspect. Unfortunately, this applied aspect focuses more on applying qualimetry to evaluate type of objects that were not evaluated before avoiding some pitfalls in qualimetric analysis or quality evaluations [162], [248], rather than on challenges or issues that may prevent applying qualimetry, or quality modeling.

We found only few contributions about the quality model operational issues in the literature likes Thapar et al. [11], Ahrens *et al.* [51], Abran *et al.* [96], or Khoshgoftaar *et al.* [173]. Nevertheless, Thapar *et al.* study was the most exhaustive exploration on this problematic, identifying a total of 9 issues that prevent quality model development or use. In their study, the authors describe each issue and gave an idea of solution about it. However, these solution hints were not practical solutions and the main authors’ focus was to identify the best quality model over a list of 24 models using number of issues per model as comparison criteria.

Our survey explored further that problematic, identifying a total of 16 issues and consolidating for the first time not only findings from previous contributions but also proposing three new ones. On the one hand, we succeeded to find for each of these 16 issues some real practical solutions, based either on previous research and industrial works, or our personal experience. On the other hand, we didn’t proceed on an additional analysis about the frequency, the importance, or the relevance of the issues to each other.

In Chapter IV.6.d, we defined a measurement process (cf. Figure 28) aligned to other measurement processes such as McGarry et al. [157] and ISO/IEC 25040 [144]. That process is designed more for expert audience. With the “*Quality Thermometer*” process, we do not redefine another measurement process; in fact, this new process is a practical stepped approach for operational (i.e., development and use) quality modeling. This process is designed for a wider audience, especially to get buy-in from stakeholders. Thus, both processes are complementary and can be aligned together: “*initial phase*” with “*stage 1*”, “*plan phase*” with “*stage 2, 3 and 4*”, and “*execution phase*” with “*stage 5, 6 and 7*”. Moreover, the “*Quality Thermometer*” process is generic enough to be applied to “*take the quality temperature*” of project, products, process, resource, supplier, organization... Finally, we choose this name and graphical representation split into seven stages (i.e., applying Miller’s law) to ensure that people keep in mind this key process.

Regarding the 6-stages process for quality development model, we decided to foster the reuse of previous quality models rather than creating new one from scratch because we do not want to discard any valuable existing contributions, give to stakeholders the perception that we reinvent the wheel for the umpteenth time, and many quality models get their inspiration from other previous models (i.e., as we are proposing to do here). Then, the construction, or model customization, starts from the selection of quality attributes that are the most relevant according to the opinion of the interviewed stakeholders, present or not in the proposed quality model. The selection of existing quality attributes joint with their weight factors form a set of polymorphic quality models that derived from a mother quality model, itself derived from the chosen quality model in stage 1. The survey together with the selection of quality attributes is not innovative, however the automatic construction of their corresponding weight factors and creation of polymorphic quality models with their mother model are innovative contributions.

Finally, although we have not mentioned quality perception in the development and use of quality model. This concept is implicitly managed through objective quality evaluation and through quality modeling evolution managed with polymorphism (i.e., evolution of the quality perception over time).

Chapter VII. Put into Practice

1. Introduction

Through the previous chapters we cleared our thesis subject “*Study of Qualimetry essential to embedded software development*” from a general perspective of theory and practice. We studied current contributions on quality modeling applied to embedded software, acknowledging that none of them was answering properly to our needs (see Chapter II). Then we reviewed theory about quality, quality modeling, qualimetry, confirming that qualimetry is the right approach to our problematic before contributing to its consolidation (cf. Chapter IV). Next, we explored the broad set of existing software quality models (cf. Chapter V) and finally investigated how to go from theory to practical operation of quality modeling, the result being the proposal of two processes for development and use of quality models (see Chapter VI).

Since now we have all the key findings to practically perform quality modeling, we are ready to exercise them to answer to our company needs, “*applying quality modeling to embedded software*”, which are also summarized in our research sub-question:

Research Sub-question 3d

What is the practical answer to our needs on automotive embedded software case?

Consequently, this chapter is about a demonstration that our approach, detailed in the previous chapters, successfully answers the company needs on automotive embedded software. Thus, we select four distinct and representative software elements from the automotive real-world (i.e., three ECU embedded software and one transverse software function) against which we apply quality modeling to develop the proper quality models for these software elements.

In the following sections, we begin by the analysis of our automobile real-world use case. Then we apply the 6-stages process to construct the embedded software polymorphic quality models, calculate their weight factors and connect their quality sub-characteristics to a proper set of metrics. Lastly, we generate the quality model that is common to these polymorphic quality models.

2. Quality modeling on a real-world use case: embedded software for the automotive industry

From a systems engineering point of view, a car is a complex system [13], itself part of a system of systems (e.g., multiple connected cars within a road system and its infrastructures). Furthermore, a vehicle results from a mass-produced of a vehicle platform instantiation. Consequently, a vehicle platform is one designed and developed generic complex system from which a set of vehicle variants (e.g., mini-compacts, crossovers, super cars, vans, utility vehicles, convertibles, etc.), with a set of available options (e.g., color, enhanced multimedia features, leather seats) are derived.

This complex system is composed of more than 40 systems, themselves decomposed in several sub-systems (ECU). To give an idea, a vehicle accounts for more than 60 ECUs combining hardware and embedded software. Therewith, it is important to note that each ECU has:

- a set of common characteristics with other ECUs: e.g., it performs diagnostic functions and has a connection interface, power, limited resources,
- a set of specific characteristics: e.g., human-machine interface, communication and safety features, responsiveness,
- a specific context: e.g., door control, engine control, telematics, seat control, air conditioning.

As a matter of fact, we note that each ECU has a design specification, architecture, terminology, and quality quantification which vary more or less with regards to the other sub-systems. This complexity therefore impacts the quality modeling of our real-world use case, a selected subset of embedded software a vehicle-platform.

We chose to limit our real-world use case to a subset of four high priority embedded software projects: three ECUs (In-Vehicle Infotainment (IVI) ECU, In-Vehicle Communication (IVC) ECU and Advanced Driver-Assistance Systems (ADAS) ECU) and one ECU-transverse embedded software (Firmware Over The Air (FOTA)).

Their primary description and characteristics are the following:

- **IVI:** This ECU is responsible for the infotainment (e.g., radio, audio-Bluetooth, CD, DVD), navigation system and is the principal human-machine interface to control the vehicle (e.g., air conditioning, ambient light control, smartphone pairing, rear view camera). Thus, the performance efficiency is not as important as quality in use aspect which must include efficiency, effectiveness, and satisfaction (e.g., quality characteristics such as pleasure or comfort). Indeed, since human-machine interface is key for this ECU, the right performance criteria must be relying on the user perception of the performance rather than pure processing time for instance. This ECU is part of the vehicle infotainment domain and mainly rely on a class D network (speeds over 1 Mbit/s) because of the high volume of video data transmitted from rear-camera or DVD, for instance. We note that network classification into 4 classes was defined in 1994 by Society for Automotive Engineers (SAE) : SAE j2056 [249], and is based on transmission speed and network performance.
- **IVC:** This ECU is responsible for vehicle telematics based on wireless communication (i.e., Bluetooth, Wi-Fi, 3G and 4G/LTE). It has no direct interaction with user (i.e., there is no. human-machine interface on IVC) but it is connected directly to IVI system by a class D ethernet network. Therefore, IVC is part of the vehicle infotainment domain. Moreover, cyber security must be addressed carefully for this ECU since it is the wireless remote access point to the vehicle network.
- **ADAS:** The advance driver assistance systems is responsible for assisting driver for safer driving and easier parking. We can distinguish four groups of functionalities:
 - o Longitudinal control systems: e.g., anti-lock braking systems (i.e., ABS), adaptive cruise control (i.e., ACC), hill descent control,
 - o *Lateral control systems:* e.g., electronic stability control (i.e., ESC), lane centering,
 - o *Alert systems:* e.g., automotive night vision, lane departure warning system (i.e., LDW), blind spot monitor,
 - o *Park assist systems:* e.g., parking sensor, automatic parking.

Its real-time functionalities (for instance, rear view camera for park assistance, lane departure detection, night vision assistance for pedestrian recognition) require processing high volumes of data, and consequently its processing performance and network bandwidth needs are high. In addition to the performance quality characteristics, a particular attention is required on security and safety, included into freedom from risk quality characteristic, because ADAS interacts with many critical vehicle systems (e.g., engine, braking system, steering) and a failure at its level can cause serious accident with injury. Like IVC, there is no direct interaction with the user (i.e., there is no human-machine interface in the ECU).

- **FOTA:** FOTA is a transverse embedded software, responsible for updating software and firmware. It may also be used to change a car configuration or enable / disable remotely some functionalities. This is for instance the case with Tesla which disabled remotely the autopilot feature on used Model S car, because the company considered that the new car owners didn't pay for that feature [250]. However, this OTA technology is not yet widely used in automotive domain, but it is progressively adopted by car manufacturers since 2012 (see Figure 62).



Figure 62 – Over-The-Air adoption timeline of major automobile manufacturers (sources: [251], [252])

FOTA is a strong tool to reduce or prevent the car obsolescence like what is done with evolving and corrective software maintenance on smartphone, tablet, and computer (e.g., the regular operating

system updates of Microsoft Windows, Linux distributions and Mac OSX) since several decades. The corrective maintenance focus on a continuous quality increase, with performance increase, security, and bug fixes. The evolving maintenance addresses the first Lehman's law [82], "*Continuing Change*": "*an E-type¹⁵ system must be continually adapted or it becomes progressively less satisfactory*". Hence, FOTA provide the support to adapt software solution to customer needs, and to push innovation or new content (e.g., navigation maps) towards vehicles. FOTA working principle is depicted by the example of Figure 63: we have the corrective or evolving packages, generated on the company servers, that are received wirelessly through the IVC ECU and dispatched to the target ECUs. Nevertheless, there are many challenges when developing and using FOTA for automotive systems:

- The multiplicity of system, software architectures and configuration to manage remotely,
- The multiple geographical locations to consider, with their own regulations, network capability and infrastructures,
- The compliance on cybersecurity, safety, and regulation requirements to achieve,
- The size of data to be transferred: from kilobytes to gigabytes.

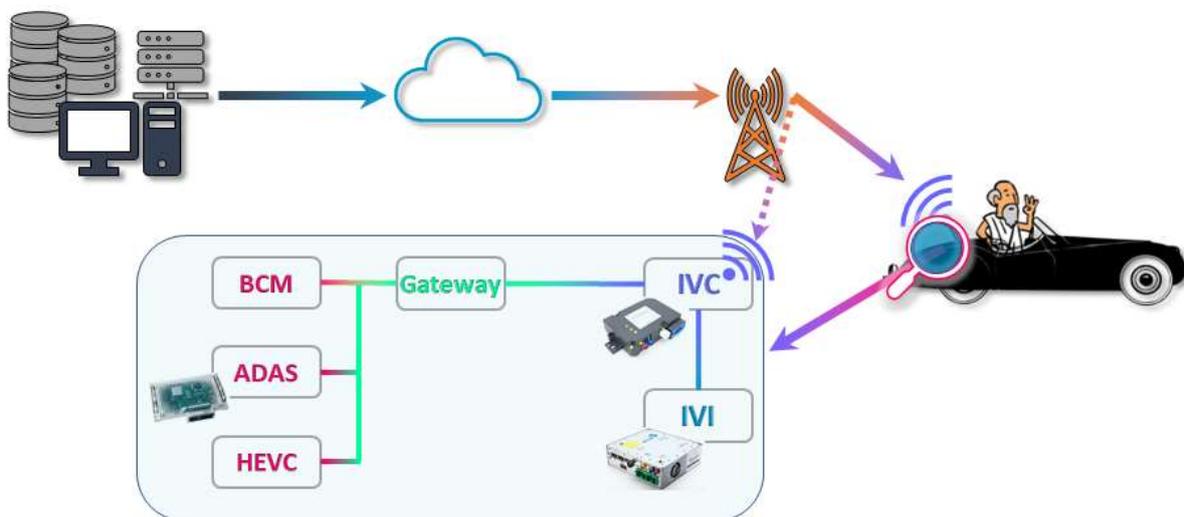


Figure 63 - Example of FOTA working principle with IVC, IVI and ADAS ECUs

So, thanks to these four overviews, we saw that the embedded software of the three ECUs and the FOTA transverse feature have their own specificities but as well a set of commonalities and links which can be retrieved through Figure 63.

In addition, in the automotive domain, software engineering must comply with regulations and standards such as A-SPICE [21]. A-SPICE focuses on software development and management processes and improvement and assessment of the process capability level. A-SPICE is neither a product quality assessment process nor a product quality control process, but its guidelines recommend using the ISO/IEC 25010 [22]¹⁶ software product quality model (i.e., product view) and quality-in-use model (i.e., user view). Some associated metrics samples are available from ISO/IEC 25022 [74] for quality-in-use and ISO/IEC 25023 [84] for software product quality. Thus, we must include this recommendation in the quality modeling of our real-world use case.

3. Initiating quality model construction via the 6-stages process

If we refer to Chapter VI.4.b and the 6-stages process, to initiate the construction of a quality model for the real-world use case that is introduced in previous Chapter VII.2, the first stage is the identification or selection of a reference model.

Because of our industrial automotive context, and as we saw in above, A-SPICE guidelines must be followed for the development of all ECU embedded software. This means that all software may have one unique and identical

¹⁵ System to address real-world activity

¹⁶ In the previous Automotive-SPICE version, up to 2.5 [253], the recommendation was to use ISO/IEC 9126 [24]

quality model, ISO/IEC/IEEE 25010. Nevertheless, even though the ISO/IEC/IEEE 25010 quality model is a general quality model, it does not take into consideration the specificities of each ECU: i.e., a specific context, but also common and specific characteristics. This standard quality model must therefore be customized for each ECU in order to be used properly. Note that the customization behavior has also been confirmed by a study on the use of quality models done by Wagner et al. in 2009 [98] and finalized in 2012 [83]. Indeed, this study emphasized that 28% of the companies interviewed use quality models taken from standards (e.g., CMM, ISO/IEC 9126) and that 79% of these 28% customize these standard quality models.

So, we assume that our original model comes from ISO/IEC/IEEE 25010 (cf. Figure 64 and Annex 8). This model covers both product and user quality perspectives and conforms to A-SPIICE. Indeed, the product quality perspective is defined by the system / software product quality model.

This model is structured into 8 quality characteristics themselves refined into 31 quality sub-characteristics. One of these characteristics, the "functional suitability", is dedicated to the functional quality requirements, while the other seven are for non-functional quality requirements (i.e., performance efficiency, compatibility, usability, reliability, security, maintainability, and portability).

The measures associated to this quality model are both internal and external measures. The internal measures are the measures done directly on the software without any need to run the software. These are for example, static code analysis, including McCabe cyclomatic complexity of model and code [254], functional requirement implementation coverage. The external measures correspond to the measures performed dynamically against the running software, and therefore requires a proper execution environment: built and lined software deployed on hardware with an operating system and all necessary data for its execution. As to user quality perspective, it is covered by the quality in use model with a set of five quality characteristics and 9 sub-characteristics. Furthermore, the associated quality in use measures are performed specific contexts of use and therefore require the complete system, simulated or not, as prerequisite of their realization.

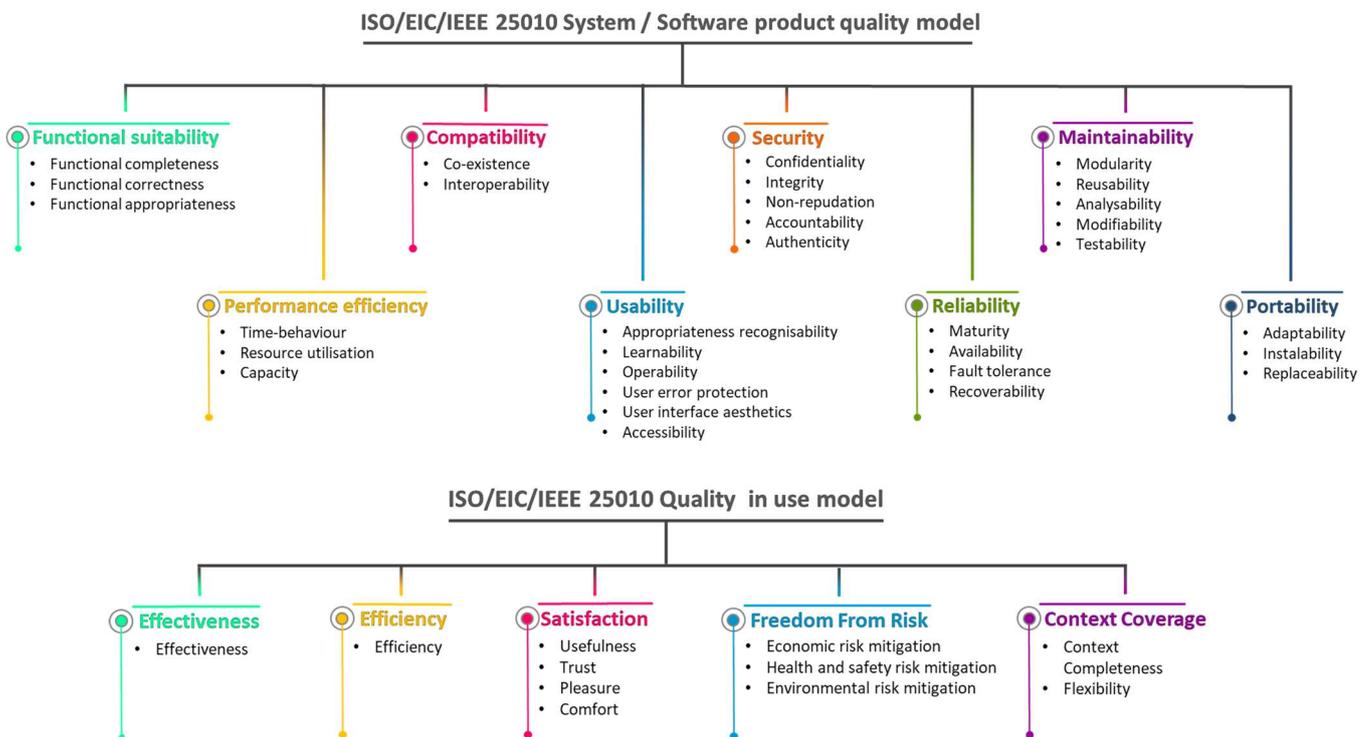


Figure 64 - The ISO/IEC/IEEE 25010 quality models: "System / Software product quality model" and "Quality in use model"

The Figure 65 summarizes the relationship between the quality models, the quality properties, the measures and an automotive sample. We note the influence and dependency links between the different quality properties, including the process quality properties coming from the software development process, for example.

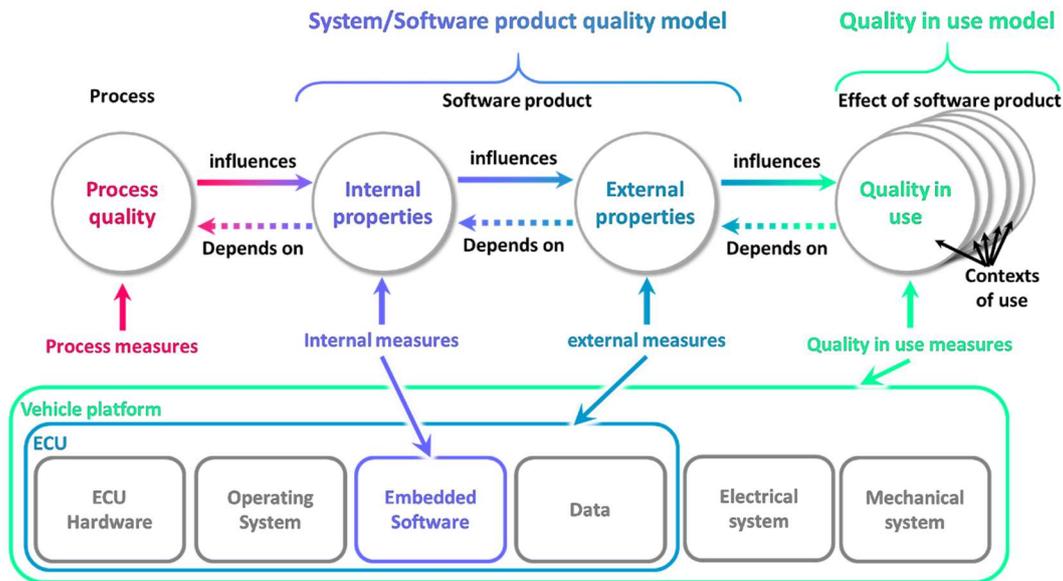


Figure 65 - Example of a mapping between quality models, properties, measures and an automotive systems and software (source and inspiration from ISO/IEC/IEEE 25010 [23] & ISO/IEC 25030 [254])

Regarding common and specific characteristics of ECUs, the aim is to associate each of them with some quality characteristics from the original quality model. For example, “*safety*” can be associated with “*freedom from risk*” and “*security*”, “*human-machine interface*” with “*satisfaction*” and “*usability*”. In addition, we model them by applying the *ad hoc* and *universal* polymorphism mechanism to the quality model (refer to Chapter IV.6.b).

The second and third steps of our 6-stages process are the construction and deployment of a survey. We remind that the survey purpose is to involve the main experts and stakeholders in the quality modeling. So, we chose to limit our survey to a subset of four high priority projects (i.e., corresponding to our real-world use case): IVI, IVC, ADAS and FOTA. Our contributor audience was made up of our main stakeholders: project managers, architects, verification and validation leaders, and assurance quality engineers who are involved in those embedded software projects.

As we described previously in Chapter VI.4.b, the survey is organized into four parts and uses Likert scales [242] for the enumeration of the possible answers (see Annex 9 for a copy of the survey). In the first part, we remind to the participant the objective of the survey usage which is to be able generate right quality model, and aligned A-SPICE, to qualify SW product quality of a variety of vehicle projects. We also ask to select their role and project in which they are involved (cf. Figure 66). In the case of multiple role or project involvement, we asked to the participant to answer several times the questionnaire.

1. What is your role? *

Architect

IAQP

PM

Validation Lead

2. What is the project you are evaluating in this survey? *

ADAS

A-IVI2

FOTA

IVC2

Figure 66 - Survey extract: participant role and project

The next part of the survey consists in the evaluation by order of importance each of the 13 ISO/IEC 25010 quality characteristics. Thus, for each of these quality characteristics, we ask to the participants to rank with a five points Likert’s scale (see Figure 58 and Figure 67) the importance of the characteristics thanks to their technical knowledge and own vision of the ECU and project quality. Note to avoid confusion or interpretation, the quality characteristic definitions are documented in a wiki page and accessible to any participants.

3. Please, rank each of the following 8 software product quality characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important
Functional Suitability	<input type="radio"/>				
Performance Efficiency	<input type="radio"/>				
Compatibility	<input type="radio"/>				
Usability	<input type="radio"/>				
Reliability	<input type="radio"/>				
Security	<input type="radio"/>				
Maintainability	<input type="radio"/>				
Portability	<input type="radio"/>				

Figure 67 - Survey extract: the ranking choice of quality characteristics

The survey's third part also focused on importance ranking, but rather on all the 40 ISO/IEC 25010 quality sub-characteristics. Their definitions are also documented and accessible to any participants. Moreover, we introduced a sixth choice in this part to give flexibility to the participant as explained in Chapter VI.4.b (see Figure 69).

5. Please, rank each of the following 3 functional suitability quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Functional Completeness	<input type="radio"/>					
Functional Correctness	<input type="radio"/>					
Functional appropriateness	<input type="radio"/>					

Figure 68 - Survey extract: the ranking choice of quality sub-characteristics

Last part of the survey proposes to the participants to share their vision on quality and complete these quality models through an open question shown in Figure 69.

16. Please, provide any additional quality characteristic(s) (not listed above) that is(are) very or extremely important for your project.

Enter your answer

Figure 69 - Survey extract: the final open question of the survey

The survey construction doesn't require any advanced functionality from a tool form. Therefore, we decided to build the survey with *Microsoft Form* because this online tool is freely usable and accessible in our company, it is easy to use, provide short report on survey responses and allows to export the survey responses in a Microsoft Excel file. After the survey implementation in *Microsoft Form*, we invited a total of 25 participants to contribute to the survey.

4. Survey result analysis

Once the survey had been answered and its data collected (cf. Annex 10 for the survey responses), we began analyzing the data, our fourth building stage, by noticing a total of 48% participation (i.e., we collected 12 responses out of 25 invitations) and an average response time of 16 minutes and 23 seconds, with a rather scattered standard deviation of 7 minutes and 26 seconds (if we exclude the exception from one stakeholder who spent 3 hours and 22 minutes to complete the survey; after a quick inquiry, we figured out that that stakeholder was interrupted during the survey and left the survey session open). So, since the survey was made of 16 questions, we can conclude that it took about 1 minute to answer to a question and the standard deviation taught us that for around half of the participant they were familiar to these quality characteristics.

Regarding the ranking responses, both quality characteristics and sub-characteristics were homogenous (cf. Figure 70) with a pick, respectively at 46.15% and 38.49%, of quality characteristics and sub-characteristics that were judged as extremely important. Almost none of them were considered as not important at all, and interestingly 12.30% of the sub-characteristics were no tanked by participant. We also remarked that no additional quality characteristic or sub-characteristic have been suggested *via* the survey open question.

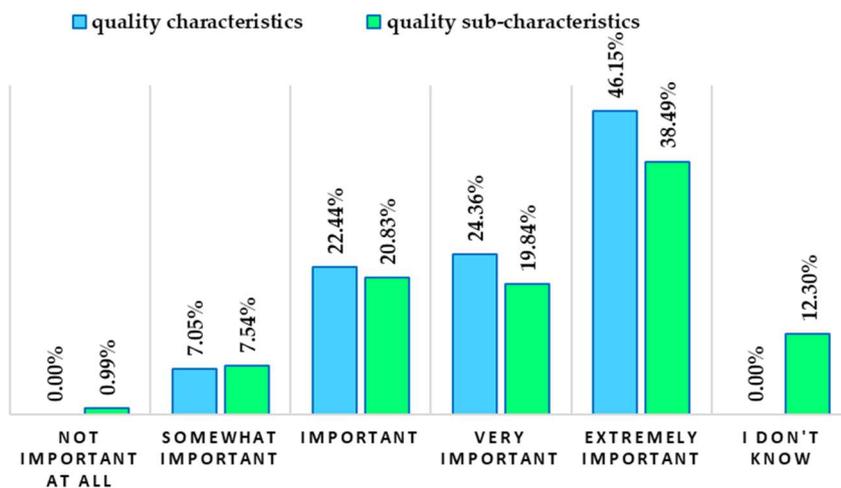


Figure 70 - Response distribution per order of importance ranking

Next, we verified the participant agreement consensus, or inter-rater reliability, based on Cohen's kappa κ [243] in the case of two raters (in this case the stakeholders who contributed to our survey) for a set of answers and Fleiss' kappa κ [244] for three or more raters. The following calculations exemplify both kappa computations on two sample sets extracted from the survey responses.

- **Cohen's kappa κ example:** The sample set for this example is obtained for the quality characteristics of the quality in use model on IVC for any project managers. We noticed that only 2 raters provide answers for IVC. Thus, we have to use Cohen's kappa κ to determine the level of consensus regarding these quality characteristics. The response counts are given in TABLE 24, however since we are using Cohen's kappa, we must rather extract and use the ranking response matches between all possible response combinations. These response match data are given in TABLE 25. Note the green cells highlighting perfect match. We remind that the Cohen's kappa κ equation is $\kappa_c = \frac{p_o - p_e}{1 - p_e}$. So, the calculations are then:

$$p_e = \frac{1}{N^2} \sum_{i=1}^k n_{i1} n_{i2} = \frac{1}{5^2} (0 * 0 + 0 * 0 + 1 * 3 + 1 * 0 + 3 * 2) = \frac{1}{25} (3 + 6) = \frac{9}{25} = 0.36$$

$$p_o = \frac{1 + 2}{1 + 1 + 1 + 2} = \frac{3}{5} = 0.6$$

Finally, $\kappa_c = \frac{p_o - p_e}{1 - p_e} = \frac{0.6 - 0.36}{1 - 0.36} = 0.375$

TABLE 24 - EXTRACTED NUMBER OF SURVEY RESPONSES PER QUALITY IN USE QUALITY CHARACTERISTICS AND IMPORTANCE FOR IVC AND PROJECT MANAGER ROLE

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important
Effectiveness	0	0	0	0	2
Efficiency	0	0	1	0	1
Satisfaction	0	0	0	0	2
Freedom From Risk	0	0	2	0	0
Context Coverage	0	0	1	1	0

TABLE 25 – THE ASSOCIATED RANKING RESPONSE MATCHES BETWEEN ALL POSSIBLE RESPONSE COMBINATIONS; THE GREEN CELLS INDICATE PERFECT MATCH

		Project Manager 1				
		Not Important at All	Somewhat Important	Important	Very Important	Extremely Important
Project Manager 2	Not Important at All	0	0	0	0	0
	Somewhat Important	0	0	0	0	0
	Important	0	0	1	1	1
	Very Important	0	0	0	0	0
	Extremely Important	0	0	0	0	2

- **Fleiss' kappa κ example:** The sample set for this example is obtained for the quality characteristics of the system/software product quality model on IVC for any roles. The response counts are given in TABLE 26. Moreover, since we consider any roles, we have 4 raters (i.e., $n = 4$) for IVC, and thus we can use Fleiss' kappa κ to determine the level of consensus regarding these quality characteristics (i.e., subjects; $N = 8$). The levels of importance used in their ranking correspond to the categories are (i.e., $k = 5$). We remind that the Fleiss' kappa κ equation is $\kappa_f = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$. So, the calculations are then:

$$\begin{aligned} \bar{P}_e &= \sum_{j=1}^k p_j^2 = \left(\frac{1}{8 \times 4}(0)\right)^2 \\ &+ \left(\frac{1}{8 \times 4}(1)\right)^2 + \left(\frac{1}{8 \times 4}(2 + 3 + 3 + 1)\right)^2 + \left(\frac{1}{8 \times 4}(1 + 3 + 2 + 2 + 1 + 3)\right)^2 \\ &+ \left(\frac{1}{8 \times 4}(1 + 3 + 2 + 2 + 1 + 3)\right)^2 = \frac{1 + 9^2 + 12^2 + 10^2}{32^2} = \frac{326}{1024} \\ &= 0.31836 \end{aligned}$$

$$\begin{aligned} \bar{P} &= \frac{1}{N} \sum_{i=1}^N P_i \\ &= \frac{1}{8} \times \frac{((1 + 9) + (9 + 1) + (4 + 4) + (1 + 4 + 1) + (4 + 4) + (9 + 1) + (9 + 1) + (9 + 1) - 4)}{4(4 - 1)} \\ &= \frac{1}{96} (10 + 10 + 8 + 6 + 8 + 10 + 10 + 10 - 4) \\ &= \frac{1}{96} (10 + 10 + 8 + 6 + 8 + 10 + 10 + 10 - 32) = \frac{68}{96} = 0.708333 \end{aligned}$$

Finally, $\kappa_f = \frac{0.708333 - 0.31836}{1 - 0.31836} = 0.57211$

TABLE 26 – EXTRACTED NUMBER OF SURVEY RESPONSES PER SYSTEM/SOFTWARE PRODUCT QUALITY CHARACTERISTICS AND IMPORTANCE FOR IVC AND ANY ROLE

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important
Functional Suitability	0	0	0	1	3
Performance Efficiency	0	0	0	3	1
Compatibility	0	0	0	2	2
Usability	0	1	2	0	1
Reliability	0	0	0	2	2
Security	0	0	3	0	1
Maintainability	0	0	3	1	0
Portability	0	0	1	3	0

As detailed in Chapter VI.4.b and algorithm shown in Figure 61, for each project, we checked whether Fleiss κ was available for current quality perspective and all roles. If it was at least greater than 0.4 (i.e., Moderate). We thus selected the data set based on the highest Fleiss κ for each project, or ECU, and all roles, or where a role had the highest κ . When it was not sufficient, we used Cohen κ as a decision criterion. If no kappa showed at least a moderate agreement, then we looked for consensus at the specific role level, all quality perspectives and all projects, or ECUs. For instance, for IVI ECU, project managers had the highest κ . For ADAS ECU, we took assurance quality engineers for all projects, or ECUs. In case where no consensus can be found at project or role level (i.e., no kappa greater or equal to 0.4 for any roles for the project and then for any roles when considering all projects together), we conduct a new survey. The complete results of our kappa-based data analysis are summarized in Table 27.

The result for sub-characteristics shown here excludes answers such as “I don’t know” simply because the overall κ values with and without this sixth rating were respectively 0.058276 and 0.120398: including these answers reduces the agreement between raters. In addition, we noted that no architects and only one validation leader contributed to our survey. This explains the cells where there is no κ value. Furthermore, κ values, 0.122846 and 0.120398, obtained for all projects and roles reveal that there is a slight agreement over all projects and between all parties. Consequently, the quality view is different for each combination of project and roles, hence the necessity to create a set of polymorphic quality models.

Furthermore, on both quality characteristics and sub-characteristics, Assurance Quality engineers were quite well aligned over all projects, with a Fleiss κ on characteristics of 0.444330 (i.e., Moderate agreement) and on sub-characteristics of 0.577705 (i.e., Moderate agreement, but close to substantial agreement), while project managers were aligned per project only: their Fleiss κ is from to 0.328165 (i.e., Fair agreement) to 0.433414 (i.e., Moderate agreement).

TABLE 27 - SURVEY DATA ANALYSIS WITH COHEN K AND FLEISS K. COLORED CELLS HIGHLIGHT K BASED CHOICE FOR EACH ECU; GRAYED CELLS HIGHLIGHT AT LEAST MODERATE AGREEMENT

		All roles	Assurance quality engineer	Project Manager	Validation Leader	Architect	
quality characteristics	All	All ECUs	[κ_F] 0.122846	[κ_F] 0.444330	[κ_F] 0.313977	\exists_1	\emptyset
		IVI	[κ_F] 0.291964	\exists_1	[κ_F] 0.433414	\emptyset	\emptyset
		IVC	[κ_F] 0.488525	\exists_1	[κ_C] 0.356436	\exists_1	\emptyset
		ADAS	[κ_C] 0.025	\exists_1	\exists_1	\emptyset	\emptyset
		FOTA	\exists_1	\exists_1	\emptyset	\emptyset	\emptyset
	Sys/SW product	All ECUs	[κ_F] 0.126425	[κ_F] 0.412305	[κ_F] 0.286275	\exists_1	\emptyset
		IVI	[κ_F] 0.284314	\exists_1	[κ_F] 0.429208	\emptyset	\emptyset
		IVC	[κ_F] 0.572111	\exists_1	[κ_C] 0.368421	\exists_1	\emptyset
		ADAS	[κ_C] 0	\exists_1	\exists_1	\emptyset	\emptyset

quality sub-characteristics ²	Quality in use	FOTA	\exists_1	\exists_1	\emptyset	\emptyset	\emptyset
		All ECUs	[K _F] 0.088664	[K _F] 0.385343	[K _F] 0.3013	\exists_1	\emptyset
		IVI	[K _F] 0.222350	\exists_1	[K _F] 0.328165	\emptyset	\emptyset
		IVC	[K _F] 0.248120	\exists_1	[K _C] 0.375	\exists_1	\emptyset
		ADAS	[K _C] 0	\exists_1	\exists_1	\emptyset	\emptyset
		FOTA	\exists_1	\exists_1	\emptyset	\emptyset	\emptyset
	All	All ECUs	[K _F] 0.120398	[K _F] 0.577705	[K _F] 0.273017	\exists_1	\emptyset
		IVI	[K _F] 0.256983	\exists_1	[K _F] 0.367483	\emptyset	\emptyset
		IVC	[K _C] 0.630553	\exists_1	[K _C] 0.147727	\exists_1	\emptyset
		ADAS	[K _C] 0.004739	\exists_1	\exists_1	\emptyset	\emptyset
		FOTA	\exists_1	\exists_1	\emptyset	\emptyset	\emptyset
		Sys/SW product	All ECUs	[K _F] 0.102039	[K _F] 0.520416	[K _F] 0.245621	\exists_1
	IVI		[K _F] 0.263481	\exists_1	[K _F] 0.353019	\emptyset	\emptyset
	IVC		[K _F] 0.581892	\exists_1	[K _C] 0.067335	\exists_1	\emptyset
	ADAS		[K _C] -0.033333	\exists_1	\exists_1	\emptyset	\emptyset
	FOTA		\exists_1	\exists_1	\emptyset	\emptyset	\emptyset
	Quality in use		All ECUs	[K _F] 0.137114	[K _F] 0.644681	[K _F] 0.295497	\exists_1
		IVI	[K _F] 0.185935	\exists_1	[K _F] 0.333542	\emptyset	\emptyset
IVC		[K _F] 0.697851	\exists_1	[K _C] 0.352941	\exists_1	\emptyset	
ADAS		[K _C] 0.104651	\exists_1	\exists_1	\emptyset	\emptyset	
FOTA		\exists_1	\exists_1	\emptyset	\emptyset	\emptyset	

[K _F] ...	Fleiss' Kappa	\exists_1	no Kappa (one rater)
[K _C] ...	Cohen's Kappa	\emptyset	no Kappa (no rater)

5. Contributions

Based on the survey data results and our analysis with Cohen and Fleiss κ , we are able to finalize the elaboration of the 3 polymorphic quality models, their weight factors, the metrics and the quality model which common (i.e., which is inherited by the 3 polymorphic quality models). These contributions are the evidences to confirm that the approach that we have defined in the previous chapters is practically working and giving us right solution.

a. Importance values and weight factors

This is the first step of the construction because we are proceeding on some computation to determine what characteristics or sub-characteristics is kept based on the analysis of the survey results. So, these computations give us automatically the importance of each of these characteristics / sub-characteristics and by consequence, they represent their weight factors once normalized.

For example, we determined for IVC ECU that we have rater consensus on quality characteristics between all raters. The corresponding answers are therefore extracted from the survey for IVC ECU:

Quality perspective	Characteristics	IVI Raters			
		Rater 1	Rater 2	Rater 3	Rater 4
System / Software product quality	Functional Suitability	Extremely Important	Extremely Important	Very Important	Extremely Important
	Performance Efficiency	Very Important	Very Important	Very Important	Extremely Important
	Compatibility	Very Important	Extremely Important	Very Important	Extremely Important
	Usability	Important	Extremely Important	Somewhat Important	Important
	Reliability	Very Important	Extremely Important	Very Important	Extremely Important

Put into Practice

Quality in use	Security	Important	Important	Extremely Important	Important
	Maintainability	Very Important	Important	Important	Important
	Portability	Very Important	Important	Very Important	Very Important
	Effectiveness	Very Important	Extremely Important	Very Important	Extremely Important
	Efficiency	Very Important	Important	Important	Extremely Important
	Satisfaction	Important	Extremely Important	Very Important	Extremely Important
	Freedom From Risk	Very Important	Important	Extremely Important	Important
	Context Coverage	Important	Important	Very Important	Very Important

To calculate the quality characteristics importance value from these answers, and then determine which of them are included or rejected (i.e., we remind that inclusion criteria is “importance value $\geq +1$ ”), we apply the rules explained in Chapter VI.4.b together with the 5 point Likert’ scale described in Figure 58.

Thus, the IVI quality characteristic importance values are (calculation details are shown):

Quality perspective	Characteristics	Importance value	Decision
System / Software product quality	Functional Suitability	$\frac{(+2) + (+2) + (+1) + (+2)}{4} = \frac{7}{4} = 1.75$	Included
	Performance Efficiency	$\frac{(+1) + (+1) + (+1) + (+2)}{4} = \frac{5}{4} = 1.25$	Included
	Compatibility	$\frac{(+1) + (+2) + (+1) + (+2)}{4} = \frac{6}{4} = 1.5$	Included
	Usability	$\frac{(0) + (+2) + (-1) + (0)}{4} = \frac{1}{4} = 0.25$	Excluded
	Reliability	$\frac{(+1) + (+2) + (+1) + (+2)}{4} = \frac{6}{4} = 1.5$	Included
	Security	$\frac{(0) + (0) + (2) + (0)}{4} = \frac{2}{4} = 0.5$	Excluded
	Maintainability	$\frac{(+1) + (0) + (0) + (0)}{4} = \frac{1}{4} = 0.25$	Excluded
	Portability	$\frac{(+1) + (0) + (+1) + (+1)}{4} = \frac{3}{4} = 0.75$	Excluded
Quality in use	Effectiveness	$\frac{(+1) + (+2) + (+1) + (+2)}{4} = \frac{6}{4} = 1.5$	Included
	Efficiency	$\frac{(+1) + (0) + (0) + (+2)}{4} = \frac{3}{4} = 0.75$	Excluded
	Satisfaction	$\frac{(0) + (+2) + (+1) + (+2)}{4} = \frac{5}{4} = 1.25$	Included
	Freedom From Risk	$\frac{(+1) + (0) + (+2) + (0)}{4} = \frac{3}{4} = 0.75$	Excluded
	Context Coverage	$\frac{(0) + (0) + (+1) + (+1)}{4} = \frac{2}{4} = 0.5$	Excluded

So, over the 13 quality characteristics, we found that six of them must be included in the quality model. We proceed the same for the quality sub-characteristics to calculate the importance values and the decision to include or exclude them. Next is to evaluate another time all excluded characteristics, taking into account the influence of their sub-characteristics. In our example we have 7 characteristics excluded and therefore candidate for this further consideration. We detail below the case for the “Usability” characteristic.

Its following sub-characteristics and their importance value computed as above:

Usability sub-characteristics	Importance value	Usability sub-characteristics mean importance value
Appropriateness Recognizability	1	$\frac{1 + 0.33333 + 1 + 0.33333 - 0.66667 + 0.33333}{6} = 0.38889$
Learnability	$\frac{1}{3} = 0.33333$	
Operability	1	
User Error Protection	$\frac{1}{3} = 0.33333$	
User Interface Aesthetics	$\frac{-2}{3} = -0.66667$	
Accessibility	$\frac{1}{3} = 0.33333$	

Now, following Chapter VI.4.b explanations on the calculation of the new importance value of previously excluded characteristics, we have for the seven characteristics of our example the following new inclusion / exclusion analysis results:

Characteristics	New importance value	Final Decision
Usability	$0.5 \times \left(\frac{0.25 + 0.38889}{2}\right) + 0.5 \times \left(\frac{1 + 1}{2}\right) = \mathbf{0.6597222}$	Excluded
Security	$0.5 \times \left(\frac{0.5 + 0.8}{2}\right) + 0.5 \times \left(\frac{1.3333 + 1.3333 + 1}{2}\right) = \mathbf{0.936111}$	Excluded
Maintainability	$0.5 \times \left(\frac{0.25 + 1.2}{2}\right) + 0.5 \times \left(\frac{1.3333 + 1.3333 + 1 + 1 + 1.3333}{5}\right) = \mathbf{0.9625}$	Excluded
Portability	$0.5 \times \left(\frac{0.75 + 0.77778}{2}\right) + 0.5 \times (1.3333) = \mathbf{1.0486111}$	Included
Efficiency	$0.5 \times \left(\frac{0.75 + 0.75}{2}\right) + 0.5 \times (0.75) = \mathbf{0.75}$	Excluded
Freedom From Risk	$0.5 \times \left(\frac{0.75 + 0.8889}{2}\right) + 0.5 \times \left(\frac{1 + 1.3333}{2}\right) = \mathbf{0.9930556}$	Excluded
Context Coverage	$0.5 \times \left(\frac{0.5 + 0.16667}{2}\right) + 0.5 \times (0) = \mathbf{0.166667}$	Excluded

We recall, for instance, that for the “Usability” characteristic the current importance value is 0.25, 0.38889 is the “Usability” sub-characteristics mean importance value and two of these sub-characteristics are very important (i.e., +1 value in the 5-point Likert’ scale). Thereby, we found that only “Portability” characteristic received enough influence from its sub-characteristics to be finally included. To summarize all the IVI ECU quality characteristics with their final importance values and inclusion / exclusion decision. Note, the included quality characteristics are the ones defining the IVI ECU quality model, and since their total number is seven, we respect the Miller’s law [235].

Quality perspective	Characteristics	Final importance value	Final Decision
System / Software product quality	Functional Suitability	1.75	Included
	Performance Efficiency	1.25	Included
	Compatibility	1.5	Included
	Usability	0.6597222	Excluded
	Reliability	1.5	Included
	Security	0.936111	Excluded
	Maintainability	0.9625	Excluded
	Portability	1.0486111	Included
Quality in use	Effectiveness	1.5	Included
	Efficiency	0.75	Excluded
	Satisfaction	1.25	Included
	Freedom From Risk	0.9930556	Excluded
	Context Coverage	0.166667	Excluded

To continue, the quality characteristic and sub-characteristic importance values are not only allowing us to determine which of the quality characteristics and sub-characteristics must be taken into account to compose our quality model, but these values provide implicitly the weight factors of the quality characteristics and sub-characteristics. Indeed, by definition these values reflect the importance of a characteristics among the others. We have the same behavior with the importance values and the sub-characteristics. Therefore, to obtain the corresponding weight factors, considering for example a sum aggregator operator, the remaining step is simply to norm the importance values. So, we first determine the total sum $\sum_{importance\ values}$ of the included characteristic importance value, per quality perspective, and lastly divide each of these importance values by $\sum_{importance\ values}$. The sum of the resulting weight factors gives 1.

Continuing with our example, we have:

$$\sum_{importance\ values_{SYS/SW\ product\ quality}} = 1.75 + 1.25 + 1.5 + 1.5 + 1.048611 = \mathbf{7.0486111}$$

$$\sum_{importance\ values_{quality\ in\ use}} = 1.5 + 1.25 = \mathbf{2.75}$$

Regarding the IVC embedded software quality model, Figure 72 depicts this hierarchical quality model with its weight factors. This model is structured over two quality perspectives, seven quality characteristics and 12 quality sub-characteristics. Thus, this model covers 53.85% of the quality characteristics of ISO/IEC/IEEE 25010 and only 28.57% of its quality sub-characteristics. We remark that functional suitability, compatibility reliability and effectiveness are the most key quality characteristics for the IVC ECU.

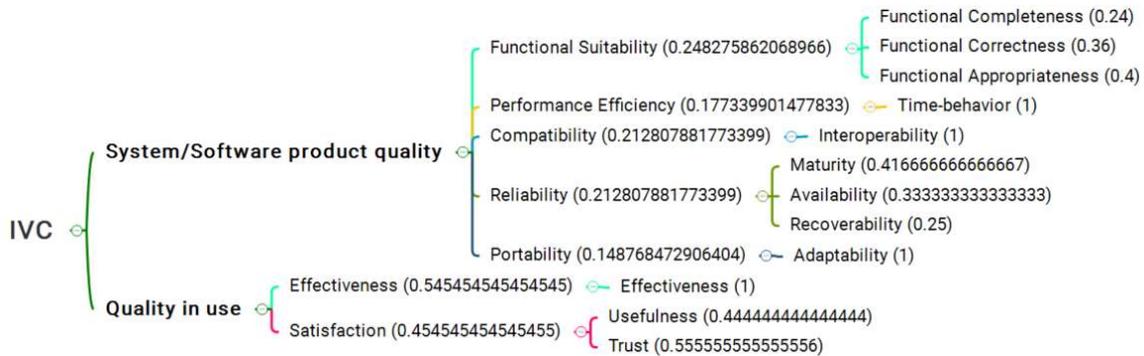


Figure 72 - The resulting IVC embedded software quality model; numbers in parenthesis are characteristic /sub-characteristic weight factors

The last embedded software quality model illustrated by Figure 73 covers both ADAS and FOTA embedded software quality model. Indeed, the analysis results for ADAS and FOTA showed rater consensus at higher level of project scope rather than specifically to either FOTA or ADAS, where rater agreements were slight or poor. This hierarchical quality model is organized over two quality perspectives, 10 quality characteristics and 25 quality sub-characteristics. Consequently, this model uses 76.92% of quality characteristics and 59.52% of quality sub-characteristics of ISO/IEC/IEEE 25010. Functional suitability, performance efficiency, effectiveness, and efficiency are the most critical quality characteristics to consider for ADAS ECU and FOTA.

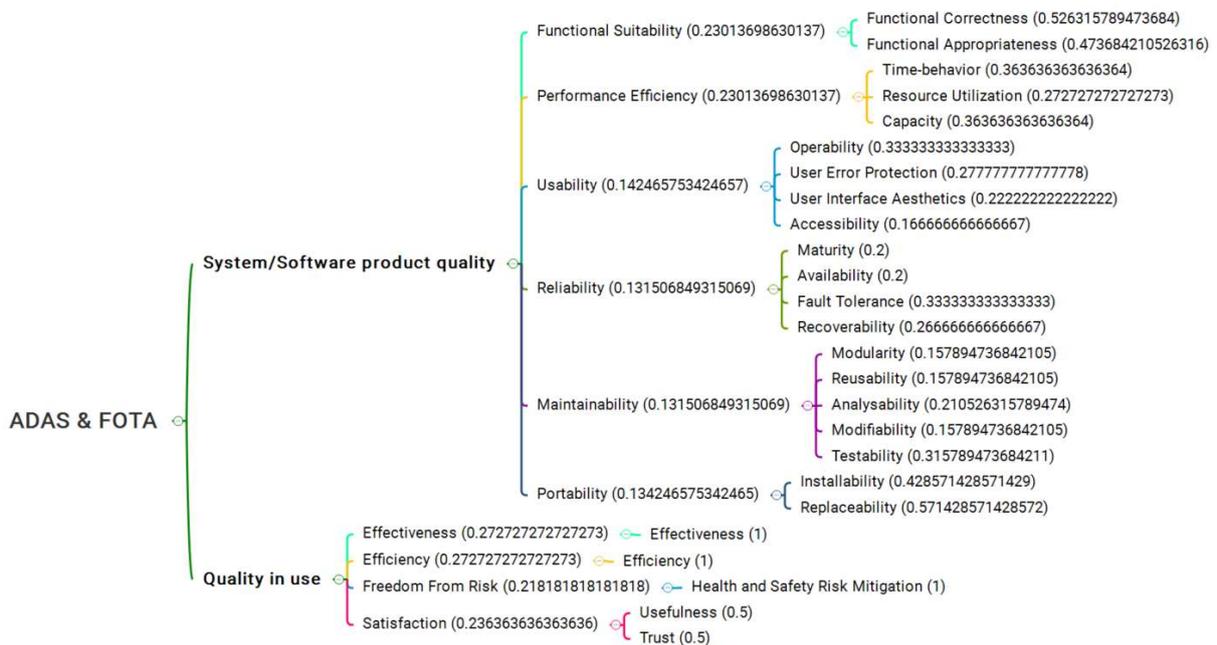


Figure 73 - The resulting ADAS & FOTA embedded software quality model; numbers in parenthesis are characteristic /sub-characteristic weight factors

Finally, with these three quality models, we see evidences that each embedded software has specific quality characteristics, and thus quality requirements, that must be took into account during development. Moreover, even if we note a certain level of similarity coverage between these quality models with regards to their quality characteristic and sub-characteristic, their instantiations through weight factors clearly differ, showing that the quality evaluation with these models is objective. Concerning ISO/IEC/IEEE 25010 standard, the three achieved

quality models also demonstrate the needs of tailoring standard quality models because with these model adaptations, almost 60%, and down to 29%, of the standard quality sub-characteristics are envisaged for our real-world use case quality models. At last, each of the quality model sub-characteristics are completed by a proper basic set of measures borrowed from ISO/IEC 25023 [161] (i.e., measures of system / software product quality), and ISO/IEC 25022 [142] (i.e., measures of quality in use) to strengthen the operationalization of these three quality models. This basic set of measures is described in Annex 12.

c. The polymorphic quality models

We reach our final building stage: the emphasis of polymorphism behavior with the construction of a common quality model from which the other quality models of our real-world use case are finally derived.

To proceed, we take the mathematical intersection of these constructed quality models. This mathematical operation gives a model with all the quality characteristics and sub-characteristics present in our constructed quality models. The result is shown in Figure 74. This common model is made of six quality characteristics and 9 quality sub-characteristics. Moreover, this model has no weight factors since it can be considered as the common “*ancestor*” of the quality models of the embedded software from our real-world use case, and not one specific quality model instantiation.

We remark also that it is possible to create a secondary common quality model, which derived from that common quality model and be the common “*ancestor*” of IVI, and ADAS & FOTA quality models. The construction is therefore similar to the common quality model but the intersection is reduced to only to these subset of quality models (cf. Figure 75). Behind these “*ancestor*” relationships is the concept of polymorphism (refer to Chapter IV.6.b and Figure 23). The Figure 76 describes the polymorphism tree structure with the polymorphism five quality models: two common quality models and three embedded software quality models.



Figure 74 - The common quality model from which the other quality models are derived

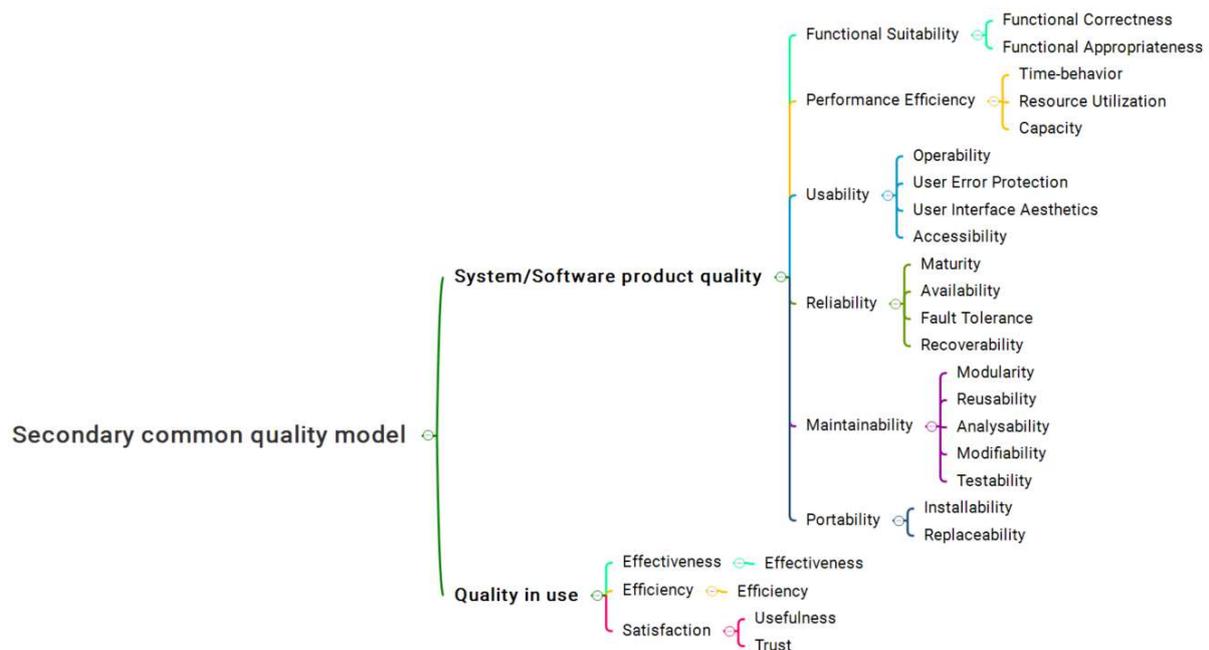


Figure 75 - The secondary common quality model from which IVI and ADAS & FOTA quality models are derived

To conclude on this section, these two common quality models and the polymorphism tree structure representation showcased the interest and how polymorphism is concretely applied.

They can serve as the basis for other ECU embedded software for which we don't already have a quality model. They can be also used for the next generation of the same ECUs and as a basis for building a new architecture: quality requirements drive systems and software feature requirements, and then the corresponding architecture requirements.

6. Threats to validity and discussions

The achievements covered by this chapter successfully demonstrate the validity of the quality modeling operationalization we have built and detail in previous Chapter VI. Moreover, by constructing quality models against embedded software real-world use case, not only we answered to the company needs, but also, we concretely proved what polymorphic quality model concept is not a theoretical concept and is an efficient quality modeling tool. Nevertheless, we notice several questionable and perfectible contribution elements.

First, our methodology relies on the reuse of quality models rather than creating a new model from scratch. Indeed, by following the reuse approach, we benefit of more than 50 years of valuable contributions in software quality models, including quality model analysis and consolidation studies that tend to mature some quality models. This is the case, for example, of ISO/IEC/IEEE 25010 which is at the center of attention of many surveys (e.g., Gordieiev *et al.* studies on quality model evolutions compare to ISO/IEC/IEEE 25010 [92], [191]). In addition, the selection of the reference model is certainly perfectible. We didn't define a specific process for this selection but instead, we consider our real-world use case constraints from automotive industry to be at the state of the art when developing a vehicle. So, we relied on A-SPICE and ISO/IEC/IEEE 25010, because we didn't find any specific and unique quality model for automotive in our exploratory and systematic literature review (see chapters Chapter II.2 and 0.2).

Another element which requires some further attention is the number of survey participants which are missing for some categories. For example, none of the invited software architects and only one validation leader replied to the survey. The main drawback is that it can weaken our results and certainly the quality model buy-in of software architects. However, we got strong support from project managers and assurance quality engineers, and as we indicated previously, these results constitute a starting point and the quality models must continue to evolve along to the projects and the platform vehicles. So, the next evolutions are obvious: we must first involve and get contributions from software architect and validation leaders, and then perform the quality modeling over to entire vehicle platform.

Regarding the calculated Kappa, a particular attention needs to be payed to Kappa interpretation because it appears that there is no universal agreement on the meaning of the table from Landis and Koch [245]. This is mainly due to the fact that Kappa interpretation is subjective and depend on the number of categories for instance. In addition, to explore further our Kappa based analysis, we may calculate Kappa values for each characteristics and sub-characteristic rather than calculating one Kappa value for all characteristics and one for all sub-characteristics per quality perspective. The idea behind this refinement is to look for finer groups of characteristics or sub-characteristics where consensus between raters exists.

The basic set of metrics, given in Annex 12, should be consolidated, or refactored to take into account the project milestone or development life cycle likes in FURPS [85] and FURPS+ [152], and therefore the temporal polymorphism applied to measures (see Chapter IV.6.b). Here our goal was to demonstrate the practicability of the solution. Moreover, concerning the thresholds, beside some internal company ones (e.g., no critical bug is allowed to pass a milestone, or a certain number of volatile and non-volatile memory footprint is allowed), we adopt the same approach than Ahrens *et al.* [51] where the aims is to do better for each subsequent releases: continue to improve through relative thresholds.

In conclusion, this chapter emphasizes the benefits to develop operational quality models, as well as the practicality of the solution against real-world use cases. Our approach allows to homogenize quality models and vocabulary over company engineering domains (e.g., systems, software, electrical, mechanical), keeping the relationship between the vehicle platform elements, and between the projects together. It is also a unique and

simple way to determine weight factors jointly with quality model. Indeed, this is not only a matter of conducting and using a survey, but rather benefiting of the survey results to determine which characteristics and sub-characteristics are the most meaningful and contributing for the project under consideration. Finally, our approach is a breakthrough by reinstating quality modeling activity as the backbone of quality evaluation, particularly compare to the current company solution, or even with usual Agile or Rapid development methodology, that relies on measurement based on tools, quality assessment via a checklist and customer issue reports or feedback.

Chapter VIII. Meta-Model: Software Quality Model Genome

1. Introduction

In Chapter V, the systematic literature review on software quality models and their classification allowed us to answer to the research sub-question 4a *“Is it possible to have a unique reference quality model for software product, or instead should we have a meta-model?”*. The conclusion was that there is no unique reference quality model for software product. Indeed, we retrieved a huge number of software product quality models either for quality definition, assessment, or prediction, and each of these quality models has a good reason to exist and to be different from the other models.

Moreover, the current standard ISO/IEC/IEEE 25010 is defined as the reference of system and software product quality model, but as the Wagner et al.’ survey [83] pointed out, the standard quality models are usually customized due to the particularity of the products, projects or company requirements. Thus, because of this customization necessity, we cannot conclude that even the current standard can be considered as the unique software product quality model.

Consequently, the purpose of this chapter is to explore the alternate case of a meta-model as the preferable solution instead of a unique reference quality model for software product.

As we did when we proposed polymorphism concept for quality model, we continue to take benefit of genetic knowledge for the creation of a meta-model. We choose genetic since we notice an analogy between quality model variations of the same quality concepts through different quality characteristics, and DNA variations between alleles of same gene. This meta-model can be then used as the beginning quality model of polymorphic quality model construction and offers a mechanism to include any quality models.

So, to elaborate such meta-model, we must respond to the research sub-question 4b:

Research Sub-question 4b	What is the construction algorithm for such meta-model?
---------------------------------	---

Once answered, we are ready to execute this construction algorithm and initiate the meta-model construction, delivering finally the first metal model results. This step is summarized over:

Research Sub-question 4c	What is the first result of the meta-model construction?
---------------------------------	--

2. Motivation and analogy with genetic

Our main motivation behind the elaboration of a meta-model is to avoid the comparison and selection of a quality model among a set of candidate quality model, resulting on discarding many potential valuable contributions for the benefit of one. Instead, we prefer to privilege union of quality models to build a quality meta-model, and then get benefit of all research works rather than selecting a subset of them. Our credo is *“all research, academic and industry work, should count”*.

During our exploratory literature mapping and systematic literature review, we noticed that frequently same quality concepts were defined more or less similarly between distinct contributions. The nuances were mainly on the wording, definition, interpretation, or on the sub-elements (e.g., distinct set of sub-characteristics for same characteristic in different quality models). For instance, in Boehm’s quality model [42], maintainability is refined into modifiability, testability and understandability. In ISO/IEC 9126 [24], it is defined by analyzability, changeability, maintainability compliance, stability, testability. And in ISO/IEC/IEEE 25010 [23], it corresponds to the following sub-characteristics: analyzability, modifiability, modularity, reusability, testability. We remark that over these three variations of the maintainability characteristic, only testability is common to these three models. We note also that modifiability is also present in Boehm’s model and ISO/IEC/IEEE 25010, and both ISO/9126 and ISO/IEC/IEEE 25010 share analyzability sub-characteristics.

So, our objectives are to identify not only convergences but also exceptions of quality perspectives, characteristics, and sub-characteristics (e.g., same concept but different wording or set of sub-characteristics) and represent these variations with statistics (e.g., probability to have a specific variation). For instance, in the

previous case with ISO/IEC 9126, ISO/IEC/IEEE 25010 and Boehm’s quality models, the probability to find testability sub-characteristic for maintainability is 100%, for modifiability two third (i.e., 66.67%) and for analyzability also two third (i.e., 66.67%). This type of variations jointly with likelihood consideration is frequent in genetic. We can cite for example Nei and Li’s formula [86] (see Chapter IV.6.c) to evaluate the degree of polymorphism or variety between several alleles, the non-parametric linkage technic [255] used for genetic disease identification.

Therefore, our observations make us propose the *Genetic–Quality* analogy, and more precisely the analogy between DNA sequences –sequence of nucleotides-, which encode genetic characters through protein information, with sequences of quality characteristics and sub-characteristics, which implicitly encode quality characters and requirements.

Thus, since in genetic a chromosome is composed of two identical copies of DNA sequences called chromatid, by applying this analogy, a quality perspective can be assimilated to a chromatid while two copies of this quality perspective correspond to a chromosome (see Figure 77).

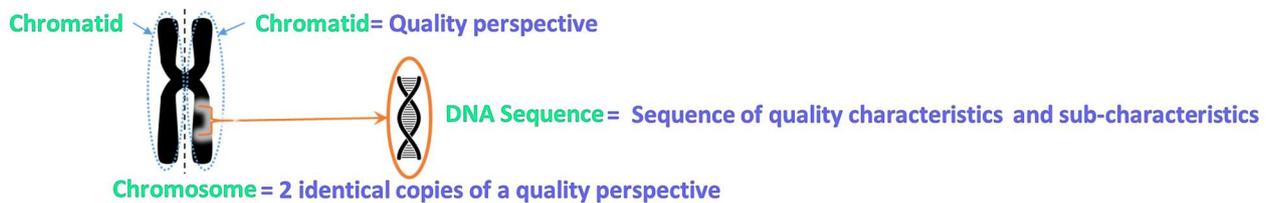


Figure 77 - Genetic-Quality analogy: global overview with chromosome, chromatids, and DNA sequence (genetic terminology is in green, quality terminology is in purple)

To go further, in genetic too, a gene is a subset of a DNA sequence “controlling the development of particular characteristics” [256] and starts at a specific location called locus. This DNA sequence subset is unitary composed of nucleotides located at specific sites. Consequently, by using our analogy *Genetic-Quality*, we associate a gene with a quality characteristic, starting at a specific location (i.e., locus) in the sequence of quality characteristics and sub-characteristics of a quality perspective. As to the nucleotide sites, they correspond to these quality perspective characteristics and sub-characteristics. In addition, a gene (e.g., a gene responsible for the eye color) in a pair of homologue chromosomes is represented by two alleles (i.e., one for each chromosome). These alleles can be identical (e.g., same eye color), or different (e.g., different eye color), but in any case, each of the allele variations has a certain probability to exist. We have the same mechanism in quality domain, where variations of quality characteristic and sub-characteristics have their own likelihood to exist. Finally, we note that “a gene is said to be polymorphic if more than one allele occupies that gene’s locus within a population” [257], retrieving the polymorphism behavior for quality model we introduced in Chapter IV.6.b. The Figure 78 illustrates these analogy details and TABLE 28 summarizes the terminology analogy.

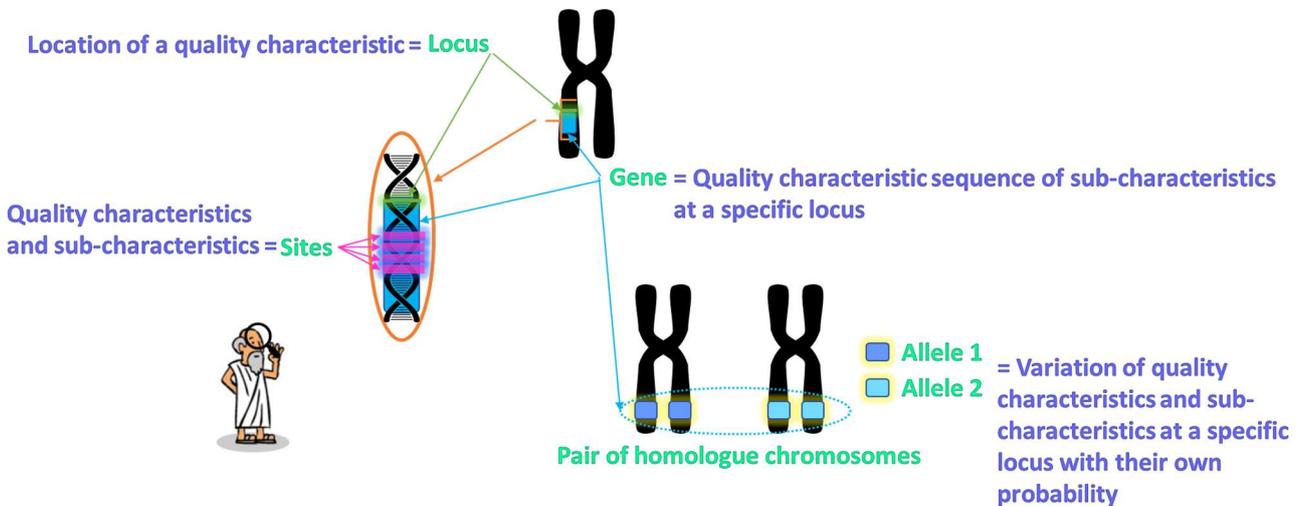


Figure 78 - Genetic-Quality analogy: detail overview with locus, sites, genes and alleles (genetic terminology is in green, quality terminology is in purple)

Figure 79 illustrates and fosters the comprehension of the proposed *Genetic-Quality* analogy through an example where this analogy is applied against the current standard ISO/IEC/IEEE 25010 quality models. In this example, the two quality perspectives, software product quality and quality in use, are assimilated to two distinct chromosomes. We notice the sequence of characteristics (i.e., genes) starting at specific locus and composed of sequences of sub-characteristics themselves located at specific sites on those chromosomes.



Figure 79 - *Genetic-Quality* analogy: an example based on ISO/IEC/IEEE 25010 quality models

At last, we consolidate the description of this *Genetic-Quality* analogy proposal by detailing the associated ontology (cf. Figure 80). This ontology described the relationship with their numeration between each concept used in this analogy and will serve to structure our meta-model construction. Furthermore, we decided to reuse the quality aligned genetic terminology (see Table 28) for conciseness purpose. Thus, a chromosome can have none or many polymorphic chromosome variations and is composed of exactly two chromatids (i.e., quality perspective). Each chromatid is itself composed of a sequence of DNA (i.e., sequence of quality characteristics and their quality sub-characteristics). Moreover, and as we saw above, a DNA sequence can be decomposed into subsets of genes, starting at specific locus in that sequence. Genes and DNA sequences can be also defined as sequences of sites (i.e., quality characteristics and sub-characteristics) which can be themselves consider as other variations of sites. Finally, a gene can have none to many variants, called allele, constituted of sites and each allele is exactly one polymorphic variation of a gene.

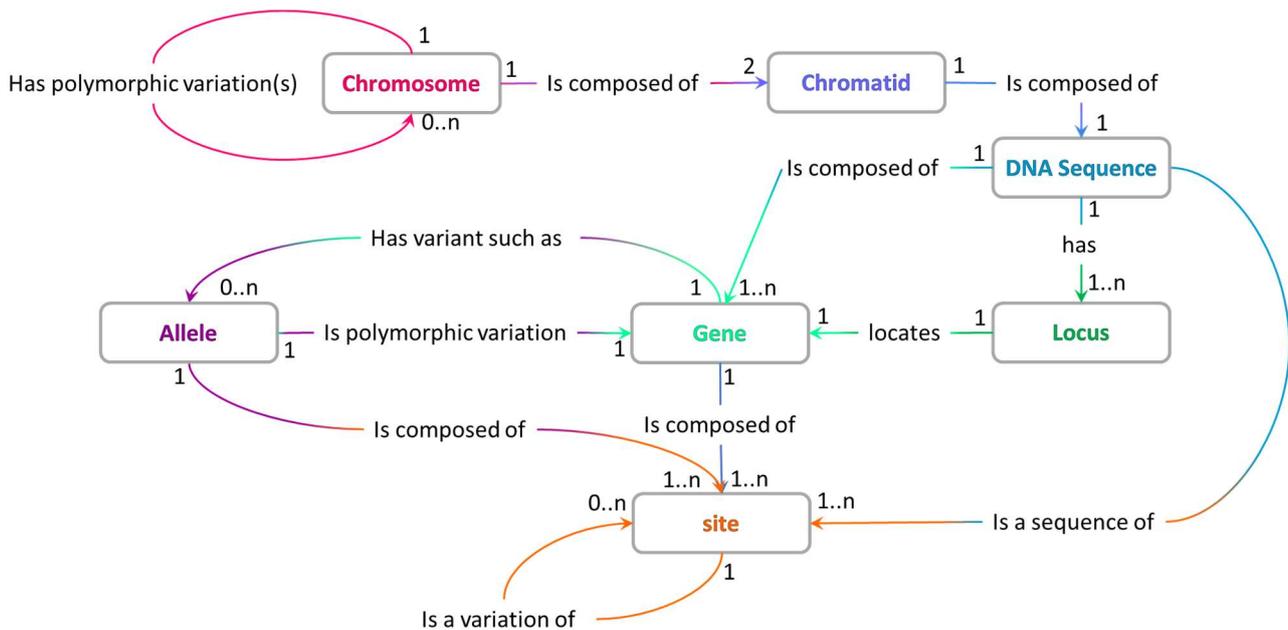


Figure 80 - *Genetic-Quality* analogy: Meta-model ontology

TABLE 28 - GENETIC-QUALITY ANALOGY: TERMINOLOGY SUMMARY

Genetic domain	Quality domain
Chromatid	Quality perspective
Chromosome	Two identical copies of a quality perspective
DNA sequence	Sequence of quality characteristics and sub-characteristics
Sites	Quality characteristic and sub-characteristics
Locus	Location of a quality characteristic in a quality perspective
Gene	Quality characteristic sequence of sub-characteristics at a specific locus
Polymorphic gene	Existence of more than one variation at a specific locus of quality characteristics and sub-characteristics
Alleles	Variations of quality characteristics and sub-characteristics at a specific locus with their own probability

3. Software quality model genome meta-model construction algorithm

a. Construction methodology

Based on the *Genetic-Quality* analogy described in previous section, we aim to elaborate an algorithm to construct a meta-model for the whole group of genes, also known as genome in genetic, of a set of software quality models.

Our construction methodology is organized around 3 stages. We start with a set of quality models as the sources of the meta-model elements, then extract and prepare groups of quality characteristics from these models, before ending with the genome construction. The Figure 81 shows the 3 stages with the associated seven steps.

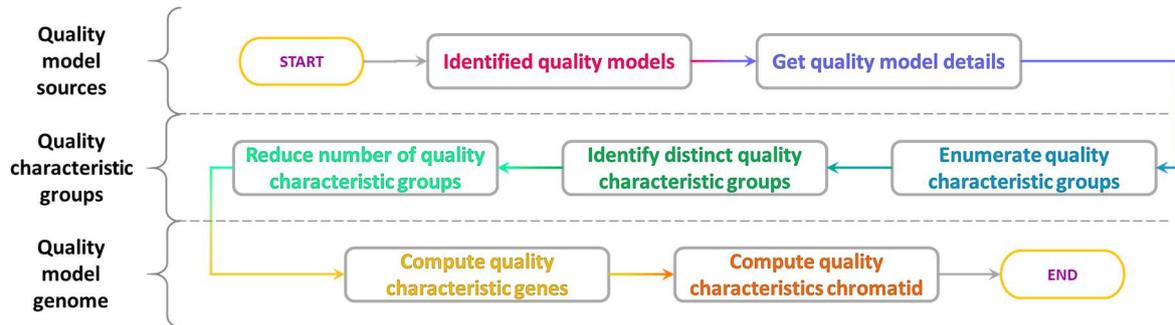


Figure 81 - The seven steps in the software quality model genome meta-model construction algorithm

The first step consists in the identification of the quality models used as source for the meta-model construction. In fact, we plan to use their characteristics, sub-characteristics, and all sub-subsequent level of characteristics, as entries for the meta-model site and gene definitions. We remark the constraint for including a quality model in that source set: to be qualified, the quality model must be clearly defined, with all clear relationship between its elements (e.g., characteristics with their sub-characteristics) and all element definitions. Once the selection is done, the next step is the quality model detail collection. Thus, all quality model structures, with their quality characteristics, quality sub-characteristics and definitions are extracted to initiate the meta-model construction.

During the third step, for each selected quality model quality characteristic groups are enumerated. We defined a quality characteristic group as a quality characteristic that owns at least two sub-characteristics. In a case where a quality characteristic has only one sub-characteristic, we named it orphan characteristic group and we consider the name of this quality characteristic to have a likelihood of 50% for the quality characteristic name and 50% for the quality sub-characteristic name. Through the next step, these quality characteristic groups are compared together in order to identify the distinct ones. Note, during the comparison only the quality characteristic name is used and not yet the sub-characteristics which may be different since they come from distinct quality models. The last step of the quality characteristic groups stage (cf. Figure 81) focuses on reducing the number of distinct quality characteristic groups by matching the similar ones, limiting potential redundancy with quality characteristics. The match criteria are based on quality characteristic group name synonyms, their definitions, and their list of sub-characteristics. For example, the “*functionality*” from ISO/IEC 9126 quality model [24]

matches the “functional suitability” from ISO/IEC/IEEE 25010 [23], the “efficiency” from Boehm [42] and ISO/IEC 9126 quality models matches the “performance efficiency” from ISO/IEC/IEEE 25010. After this fifth, we have collected all the necessary elements to compute quality model genome (i.e., whole genes and their respective chromatids).

Consequently, over the sixth the quality characteristic genes are calculated from the reduced and distinct quality characteristics groups. So, by computing each gene we mean to identify for each quality characteristic all possible quality sub-characteristics which are linked to the characteristic, with their respective likelihood to happen. The different possible set of sub-characteristics from the quality models can be seen as the set of sites forming gene alleles (see previous Chapter VIII.2 and Table 29 for an example on “portability” quality characteristics) from each quality model source where it exists. Therefore, after identifying matching sites between alleles, for each distinct site (i.e., sub-characteristic), we take the most dominant name as site name, determine statistics about the probability to have this site (i.e., number of alleles where we find the site over the number of considered alleles) and the frequency related to each similar name we find in the site (e.g., all identical names over considered alleles give 100%, but if half of them are similar, we may have 50% or less: see result sample on “portability” quality characteristics in Table 30). Note, group characteristic likelihood is computed from the number of quality model sources which reference its quality characteristic.

TABLE 29 - ALLELE AND SITE EXAMPLE FOR "PORTABILITY" QUALITY CHARACTERISTIC

Gene: Portability		Allele 1	Allele 2	Allele 3	Allele 4	Allele 5	Allele 6	Allele 7	Allele 8
site 1	Adaptability	Adaptability	Adaptability			Adaptability		Adaptability	Adaptability
site 2	Instalability	Instalability	Instalability			Deployability		Instalability	Instalability
site 3	Replaceability	Replaceability	Replaceability			Replaceability		Replaceability	Replaceability
site 4	Conformance		Portability Compliance						Conformance
site 5	Modularity			Self- containedness	Modularity				
site 6	Machine independence			Device independence	Machine independence				
site 7	Self-Descriptiveness Software system				Self-Descriptiveness Software system			Portability	
site 8	independence				independence				
site 9	Co-existence		Co-existence						
site 10	Reusability					Reusability			

TABLE 30 - GENE COMPUTATION EXAMPLE FOR "PORTABILITY" QUALITY CHARACTERISTIC

Gene: Portability		
site 1	Adaptability	Adaptability = 100%
site 2	Instalability	Instalability = 80% Deployability = 20%
site 3	Replaceability	Replaceability = 100%
site 4	Conformance	Conformance = 50% Portability Compliance = 50%
site 5	Modularity	Modularity = 50% Self-containedness = 50%
site 6	Machine independence	Machine independence = 50% Device independence = 50%
site 7	Self-Descriptiveness Software system	Self-Descriptiveness = 50% Portability = 50% [i.e., Portability documentation = 25%, Portability complexity = 25%]
site 8	independence	Software system independence = 100%
site 9	Co-existence	Co-existence = 100%
site 10	Reusability	Reusability = 100%

Over the seventh and final step in the meta-model construction, we regroup quality characteristic genes together based on their quality characteristic and sub-characteristic relationship to build chromatids. A chromatid is identified as a top-level gene, that is to say, it is not a child of another gene. So, for each identified chromatid, we take its genes, sub-genes, and descent to sites in order to combine their probability values and list all sites (cf. example depicted by Figure 82).

Moreover, to optimize the number of those sites, we try to merge identical sites together by applying the following rules:

- 1- if inside a gene, two sites are identical, or similar (i.e., slight variations), and are not part of any sub-genes, we replace these two sites by only one site which has a likelihood equal to the sum of the probability values of these two sites (with a maximum value of 1 or 100%),
- 2- if inside a gene, two sites are identical, or similar (i.e., slight variations), and one of them is part of a sub-gene, replace the two sites by only the sub-gene; then sum their probability values (with a maximum value of 1 or 100%),
- 3- if inside a gene, two sites are identical, or similar (i.e., slight variations), and the two sites are part of two distinct sub-genes, replace the two sites by
 - a. the merge of the two sub-genes, summing their probability values, if the gene joint is possible without breaking any other existing gene joint,
 - b. otherwise use site variation to name them and refactor each sub-gene into distinct sites,

During merge, or refactor, we must try to reduce the number of sites by

- A- Merge identical sites including their probability value, to reduce redundancy,
- B- Integrate sites not belonging to sub-genes into sub-genes, whenever it is possible,
- C- Optimize merge between sub-genes by maximizing sub-gene overlap, reducing the number of sites through consideration to name variations (e.g., "simplicity" with "self-descriptness")

Moreover, if a quality characteristic loop exists (e.g., quality characteristic A point to quality characteristic B, which point back to characteristic A), we must use chromatid context and common sense to remove the most irrelevant relationship.

The Chapter VIII.4 and Chapter VIII.5 illustrate in details against the first metal-model construction that methodology.

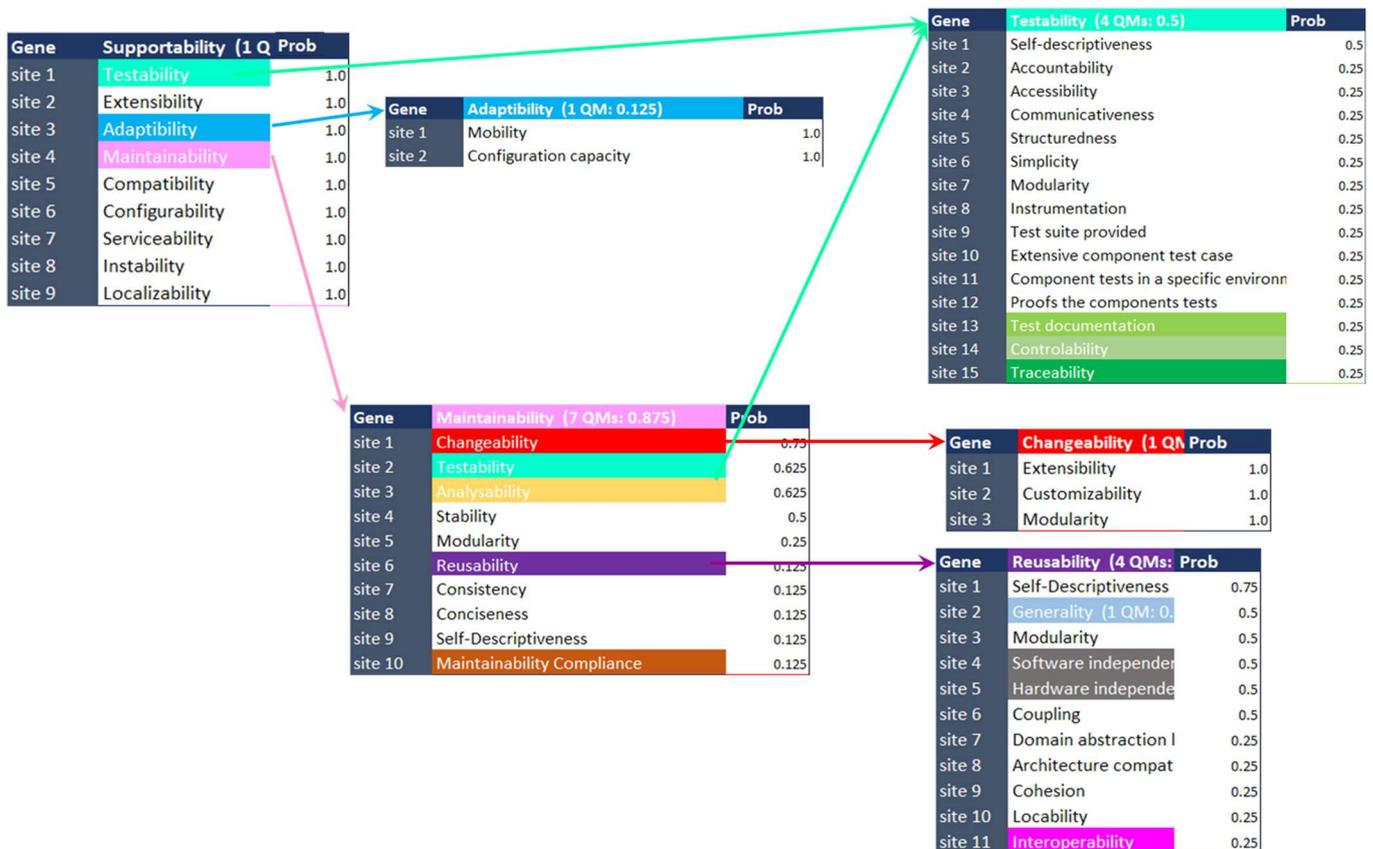


Figure 82 - Example of "Supportability" chromatid computation with the main gene and some of its sub-genes (testability, adaptability, maintainability, changeability and reusability); sub-genes are site names with color background

b. Construction algorithm

Research Sub-question 4b What is the construction algorithm for such meta-model?

To apply concretely this construction methodology, we create a software quality model genome meta-model construction algorithm. It is organized around the three stages and their steps seen in the methodology (cf. Figure 81). The main construction algorithm is defined in the next three solid line boxes (i.e., one per construction stage) and should be used sequentially. Moreover, to facilitate its applicability, we also describe three functions that require a particular attention: the extraction of quality characteristic group elements, the estimation of distance between two groups of quality characteristics and the creation of a gene node graph. Each of these three functions are put in a dashed box.

Each comment in the algorithm starts by a # character, and the main variables are:

- **Chromatids:** list of identified chromatids
- **Distance_Acceptance_Threshold:** Maximum acceptance threshold to accept that two characteristics are lexically and semantically closed,
- **Genome_Metamodel:** Software quality model genome meta-model (i.e., the result meta-model),
- **Grp:** Groups of distinct and reduced quality characteristics used to build chromatid and software quality model genome meta-model,
- **Grp_Graph:** graph containing all characteristics groups with their sub-characteristics (i.e., their children),
- **Pool_of_QM:** pool of quality model sources to merge,
- **Push:** function to put new element on the top of the destination structure,
- **QM_Grp:** Groups of quality characteristics with their sub-characteristics extracted from the quality model sources.

In the first stage, we assumed that quality model sources are already identified, and their data (i.e., quality characteristics, sub-characteristics, their definition, their relationships) are available in loadable structures (e.g., quality model data store in files). Thus, the data initialization is achieved by loading the data from all quality model sources into the data structure “*Pool_of_QM*”.

With the second stage, the aim is to extract and optimize all the quality characteristic groups “*QM_Grp*” from the quality model sources “*Pool_of_QM*”. As indicated in the methodology, this optimization consists in limiting the number of distinct quality characteristic groups by regrouping the similar ones. So, we crawl each quality characteristic group from the quality model sources and verify if any similar quality characteristic group exist in these current data. If this is the case, we keep only one instance of these data to avoid redundancy. However, before any removing action, we merge the quality characteristic variations (e.g., “*functionality*” versus “*functional suitability*”) and their respective likelihood to happen into the kept instance.

While the first two stages were responsible to extract and prepare the data related to quality characteristic groups, the genome real meta-model construction is performed in the third and last stage. It is organized in three phases. The first one is to structure the genes resulting from the groups of distinct and reduced quality characteristics, “*Grp*”, under a graph “*Grp_Graph*”. The graph nodes (i.e., vertices) correspond to each quality characteristics from “*Grp*” and their edges are the children relationship between theses quality characteristics and their sub-characteristics which are also considered as part of the graph nodes. Next phase is to find all chromatids “*Chromatids*”. Thus, for all nodes in the graph, we check if that node has a parent. If this node has none, it is a chromatid. And finally, for each chromatid, we build list of from descendent gene sub- graphs and add the result to our software quality model genome meta-model “*Genome_Metamodel*”

```
# Create Genome quality meta-model from pool of n quality models
# Stage 1: Quality model sources
#   Data initialization
Loop: i from 1 to n
    Pool_of_QM[n] ← Load data of quality model n
End loop
```

```

# Stage 2: Quality characteristic groups
#   Collect quality characteristic group data
Loop: i from 1 to n
    QM_Grp[i] ← Get groups of quality characteristics w/sub-characteristics in
    Pool_of_QM[i]
End loop

#   Identify and reduce distinct quality characteristic groups
#   by default, the first quality characteristic groups are is the ones from first quality model
Grp ← QM_Grp[1]

#   Now, check the other groups from the remaining quality models
Loop: i from 2 to n
    # Extract candidate group elements based on distance with already identified group elements
    Candidate_Grp ← Extract element of QM_Grp [j] to be mapped to Grp
    If Candidate_Grp is not empty
        # Insert Candidate_Grp quality characteristics and sub-characteristics
        # into Grp and combine their statistic data accordingly
        Merge Candidate_Grp with Grp
    End loop
End loop

```

```

# Stage 3: Quality model genome
#   Construct gene graph
Grp_Graph ← empty
Loop: i from 1 to size (Grp)
    # Create a graph node for the i-th element of Grp: the current element i, with child list, if any
    Grp_Graph_Node ← Get Graph Node for i -th element in Grp
    Grp_Graph ← Push Grp_Graph_Node
End loop

#   Identify chromatids
Chromatids ← empty
Loop: i from 1 to size (Grp_Graph)
    # By default, we consider that current graph node is a candidate chromatid, until we refute it
    Candidate_Chromatid ← i
    Loop: j from 1 to size (Grp_Graph)
        # If i is one of the children of Grp_Graph[j] child, this means that i has a parent
        # and cannot be considered anymore as a chromatid
        If (j != i) and
            (i in child list of Grp_Graph[j])
                Candidate_Chromatid ← empty
                break
        End loop
    # Finally, if we still have a Candidate_Chromatid at that point, we can add it to the list of valid chromatids
    If Candidate_Chromatid != empty
        Chromatids ← push Candidate_Chromatid
    End loop

#   Construct software quality model genome meta-model based on those chromatids
Genome_Metamodel ← empty
Loop: i from 1 to size (Chromatids)
    # This is done by expanding all children levels from Grp_Graph graph for each chromatid
    Genome_Metamodel ← push (Grp_Graph[i], expand all children of Grp_Graph[i])
End loop

# Final result of the software quality model genome meta-model construction
# is stored in "Genome_Metamodel" variable

```

The extraction and optimization feature are performed through the “*Extract element of QM_Grp_Set to be mapped to Grp*” function. This function is straightforward. Indeed, it goes through all the elements for current quality model group set *QM_Grp_Set* and look for which one of these elements has the closest distance to the quality characteristic groups *Grp*. If there is one element found, it is removed from the quality model group set *QM_Grp_Set* and the found element is returned. When two elements are distinct, or too far from each other, the distance function (i.e., *Get distance between Ref_Grp and Tgt_Grp*) gives a value of *infinity*. Therefore, if all elements of the quality model group set *QM_Grp_Set* are distinct or too far from the quality characteristic groups *Grp*, no result is found because the “*If Tmp_Distance < Candidate_Distance*” comparison will be always false (i.e., this comparison becomes “*If infinity < infinity*”).

Regarding the distance function, “*Get distance between Ref_Grp and Tgt_Grp*”, we first test if the two characteristic “words” (i.e., “*Ref_Grp*” and “*Tgt_Grp*”) are identical. In the positive case, the distance is equal to 0. Otherwise, we verify if the two characteristics “*Ref_Grp*” and “*Tgt_Grp*” are closed synonym, by are verifying their lexical and semantic distance.

For this task we use both WordNet lexical database [175] - available online at Princeton University¹¹, and the online semantic Altas (i.e., <http://www.atlas-semanticques.eu/>) allowing to determine multiple levels of synonym constellations (cf. example in Figure 83 of three synonym constellations for “*efficiency*” word). A synonym constellation is a group of synonyms semantically close based on cliques (i.e., minimum semantic units with very fine granularity).

So, if “*Tgt_Grp*” is within the “*Ref_Grp*” synonym constellations, or in the contrary case, if the intersection of “*Ref_Grp*” and “*Tgt_Grp*” synonym constellations is not empty, then the distance result is the minimum constellation distance. Finally, to avoid too long distance, the distance result is compared to a threshold value set by user to acknowledge range of valid distance values. However, we consider that if this distance is higher to this threshold, but both “*Ref_Grp*” and “*Tgt_Grp*” definitions and sub-characteristics are matching (see Motogna *et al.* [94]) then we accept this distance. In all other cases, the distance is fixed to *infinity* to reflect the disjunction.

The “*Get Graph Node for i -th element in Grp*” function create a graph node, or vertex, for the *i*th element of the groups of distinct and reduced quality characteristics, “*Grp*”. A graph node is made of the current “*Grp*” index *i*, and the children list, if any exist, of this *i*th element of “*Grp*”. The children retrieval is achieved by crawling all elements of “*Grp*” and then verifying if they are one of its sub-characteristics.

```
# Find the closest element of QM_Grp_Set to Grp, if any, and remove it from QM_Grp_Set
Extract element of QM_Grp_Set to be mapped to Grp
Candidate_Grp ← empty
Candidate_Distance ← infinity
Candidate_Element_Id ← 0

# Crawl all elements in QM_Grp_Set to find the closest element to Grp if it exists
Loop: i from 1 to size (QM_Grp_Set)
    Tmp_Grp ← Get element i from QM_Grp_Set
    # Determine the distance between these two elements
    Tmp_Distance ← Get distance between Grp and Tmp_Grp
    # If the distance is lower than previous one (note if distance is always infinity,
    # then elements are considered too far from each other: don't take them)
    If Tmp_Distance < Candidate_Distance
        Candidate_Grp ← Tmp_Grp
        Candidate_Distance ← Tmp_Distance
        Candidate_Element_Id ← i

End loop

# We found a candidate element, so remove it from QM_Grp_Set
If Candidate_Element_Id != 0
    Remove element Candidate_Element_Id from QM_Grp_Set
Return Candidate_Grp
```

```

# Compute distance between two groups of characteristics: Ref_Grp and Tgt_Grp
Get distance between Ref_Grp and Tgt_Grp
  Distance ← infinity
  # Test first if the two characteristic "words" are identical
  If Ref_Grp = Tgt_Grp
    Distance ← 0
  Else
    # Otherwise check if Tgt_Grp is within the Ref_Grp synonym constellation
    Cons_Ref_Grp ← Get synonym constellation for Ref_Grp from
      (https://wordnet.princeton.edu/ and
      http://www.atlas-semanticques.eu/)
    If Tgt_Grp is in Cons_Ref_Grp
      Distance ← Get constellation closest distance of Tgt_Grp in Const_Ref_Grp
    Else
      # Otherwise check the intersection of Tgt_Grp and Ref_Grp synonym constellations is not empty
      Const_Tgt_Grp ← Get synonym constellation for Tgt_Grp from
        (https://wordnet.princeton.edu/ and
        http://www.atlas-semanticques.eu/)
      If (Const_Ref_Grp ∩ Const_Tgt_Grp) is not empty
        Distance ← Get constellation closest distance of (Const_Ref_Grp ∩
          Const_Tgt_Grp)
      # Finally, assess that the distance is not too high, so require user to acknowledge (above process is automatic)
      If Distance > Distance_Acceptance_Threshold
        # Give a last chance through a test checking the definition of characteristics Tgt_Grp
        If Not (Ref_Grp can be mapped to Tgt_Grp based on their definitions and sub-
          characteristics)
          Distance ← infinity
      Return Distance
  
```

```

# Create a graph node for the i-th element of Grp: the current element i, with children list, if any
Get Graph Node for i -th element in Grp
  Grp_Element ← Grp [i]
  Grp_Children ← empty
  # crawl all Grp elements to verify which ones are sub-characteristics of the current element
  Loop: j from 1 to size (Grp)
    if (j != i) and
      (Grp[j] is in sub-characteristics of Grp_Element)
      Grp_Children ← push j
  End loop
  Return (i, Grp_Children)
  
```

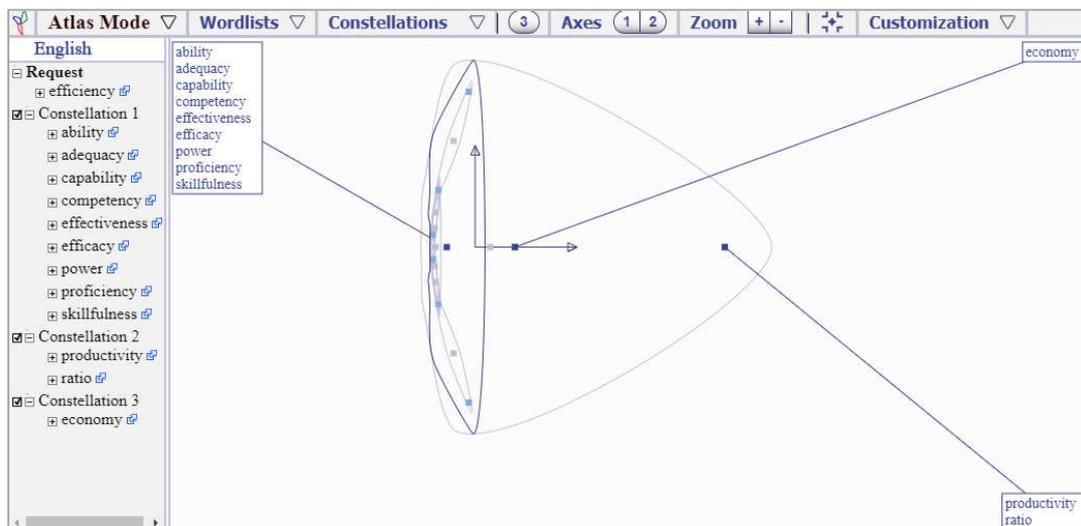


Figure 83 - Example from the online Semantic Atlas: three synonym constellations for "efficiency" word as quality characteristic (source: <http://www.atlas-semanticques.eu/>)

4. Software quality model genome meta-model construction

a. List of quality models

As indicated in Chapter VIII.3, the preliminary assumption of the software quality model genome meta-model construction algorithm is the existence of a set of software quality models to be used for the meta-model build. Indeed, the algorithm generate a list of “quality” genes from the quality model characteristics and sub-characteristics, identify the corresponding “quality” chromatids and finally synthetize a software quality model genome meta-model resulting of these input quality models.

Moreover, these quality models must have their quality characteristics and sub-characteristics, together with their definition and relationships, clearly defined to be exploitable. Note, as Oriol *et al.* study [12] highlighted, definition completeness of quality models is often partial or incomplete. In their survey on quality models for web-services, they found that only 51% of them had their definitions 100% complete.

So, considering Thapar *et al.* [11] study conclusion with regard to the challenges and issues in software quality model development and use, we select four of the five quality models which have only three identified issues: Alvaro [36], Bawane [103], ISO/IEC 9126 [24], and Kalaimagal’s Q’Facto 12 [102]. We reject GEQUAMO quality model [229] from this selection since we were not able to retrieve all details (i.e., definitions and all sub-characteristics) about this quality model. Furthermore, we complete that selection list with four additional and widely used software quality models for which reference papers with quality model details, including definitions, are available. These four quality models are Boehm [42], McCall [41], FURPS [85],and ISO/IEC/IEEE 25010 [23].

Thus, we identified a total of eight software quality models (i.e., Boehm, McCall, FURPS, ISO / IEC 9126, Alvaro, Bawane, Kalaimagal’s Q’Facto 12, and ISO / IEC / IEEE 25010) to construct the first software quality model genome meta-model. The quality model details are provided in Annex 13 while Table 31 is an overview of these models.

TABLE 31 - OVERVIEW OF THE EIGHT SELECTED QUALITY MODELS FOR SOFTWARE QUALITY MODEL GENOME META-MODEL CONSTRUCTION

Quality model	Reference	Date	Quality model element numbering and type
Boehm	[42]	1976	3 high-level qualities
			7 intermediate-level qualities
			15 primitive quality characteristics
McCall	[41]	1977	3 Perspectives
			11 quality factors
			23 quality criteria
FURPS	[85]	1987	5 Components
			25 Sub-components
ISO/IEC 9126	[24]	1991	2 quality perspectives
			10 quality characteristics
			31 quality sub-characteristics
Alvaro	[36]	2010	6 quality characteristics
			23 quality sub-characteristics
			48 attributes
Bawane	[103]	2010	2 quality perspectives
			11 quality characteristics
			28 quality sub-characteristics
Kalaimagal’s Q’Facto 12	[102]	2010	12 quality factors
			30 quality criteria
			44 quality measures
ISO/IEC/IEEE 25010	[23]	2011	2 quality perspectives
			13 quality characteristics
			42 quality sub-characteristics

We remark that these quality models total 396 quality characteristics and sub-characteristics. This corresponds to an average of 49.5 quality characteristics and sub-characteristics per quality model, with a minimum of 30 with FURPS quality model and a maximum of 86 with Kalaimagal’s Q’Facto 12 quality model.

Note, we don't need to necessarily have distinct quality models here. Indeed, ISO/IEC/IEEE 25010 is the evolution of ISO/IEC 9126, and consequently are not "purely" distinct, but it is valid to consider these two quality models as well in the genome construction, because the evolution is a natural variation mechanism of a genome.

b. List of genes with their variations

Before generating the list of quality characteristic genes from these eight selected software quality models, there are several intermediary steps to complete as shown in Figure 81.

Firstly, we must enumerate the groups of quality characteristics from these quality models. We qualified as group of quality characteristics, a quality characteristic that owns at least one sub-characteristic. Thus, we are able to retrieve 103 quality characteristic groups over the eight quality models, which corresponds to an average of 12.875 groups per quality model, and with the same two models for the extrema: five groups for FURPS and 25 for Kalaimagal Q'Facto 12. Moreover, a part of these 103 groups are identical and therefore can be clustered into 55 distinct quality characteristic groups.

The complete group list per quality model is described in Table 32. The two left columns summarize the name of the 55 distinct quality characteristic groups and their number of occurrences over the eight models. For example, "Efficiency" is used in six of these eight quality models (i.e., Boehm, McCall, Alvaro, ISO/IEC 9126, Bawane and Kalaimagal Q'Facto 12) while "Compatibility" is defined only in Kalaimagal Q'Facto 12 and ISO/IEC/IEEE 25010. The "Portability" group is indicated twice at this step because we retrieve this quality characteristic as quality characteristic and quality sub-characteristic in Kalaimagal Q'Facto 12 model. In the next step, this redundancy is removed, counting for only one group for that model.

TABLE 32 – THE 55 DISTINCT QUALITY CHARACTERISTIC GROUPS FROM THE EIGHT SELECTED SOFTWARE QUALITY MODELS

Total	Quality Characteristic Name	Boehm [42]	McCall [41]	FURPS [85]	ISO/IEC 9126 [24]	Alvaro [36]	Bawane [103]	Kalaimagal Q'Facto 12 [102]	ISO/IEC/IEEE 25010 [23]
1	Adaptability					Adaptability			
1	Changeability					Changeability			
2	Compatibility							Compatibility	Compatibility
1	Compliance					Compliance			
1	Context coverage								Context coverage
1	Controllability							Controllability	
1	Correctness		Correctness						
6	Efficiency	Efficiency	Efficiency		Efficiency	Efficiency	Efficiency	Efficiency	
1	Fault-tolerance					Fault Tolerance			
1	Freedom from risk								Freedom from risk
1	Flexibility		Flexibility						
5	Functionality			Functionality	Functionality	Functionality	Functionality	Functionality	
1	Functional suitability								Functional suitability
1	General Utility	General Utility							
1	Generality							Generality	
1	Hardware/Software Independence							Hardware/Software Independence	
1	Human engineering	Human engineering							
1	Instability							Instability	
1	Integrity (security)		Integrity (security)						
2	Interoperability		Interoperability					Interoperability	
1	Learnability							Learnability	
7	Maintainability	Maintainability	Maintainability		Maintainability	Maintainability	Maintainability	Maintainability	Maintainability
1	Maturity					Maturity			
1	Modifiability	Modifiability							
2	Operability					Operability		Operability	
1	Performance			Performance					
1	Performance efficiency								Performance efficiency
7	Portability	Portability	Portability		Portability	Portability	Portability	Portability	Portability

Meta-Model: Software Quality Model Genome

1	Portability							Portability
2	Product in use			Product in use		Product in use		
1	Product operation		Product operation					
1	Product revision		Product revision					
1	Product transition		Product transition					
1	Quality in use							Quality in use
1	Recoverability							Recoverability
8	Reliability	Reliability	Reliability	Reliability	Reliability	Reliability	Reliability	Reliability
1	Resource behavior					Resource behavior		
4	Reusability		Reusability			Reusability	Reusability	Reusability
1	Safety in use							Safety in use
1	Satisfaction							Satisfaction
3	Security					Security		Security
1	Self-Contained							Self-Contained
2	Software Product			Software Product		Software Product		
1	Standardization / Certification							Standardization / Certification
1	Suitability					Suitability		
1	Supportability			Supportability				
1	System / Software product							System / Software product
1	Test documentation							Test documentation
4	Testability	Testability	Testability			Testability		Testability
1	Time behavior					Time behavior		
1	Traceability							Traceability
2	Understandability	Understandability				Understandability		
7	Usability		Usability	Usability	Usability	Usability	Usability	Usability
1	Usability (As-is utility)	Usability (As-is utility)						
1	Usability in use							Usability in use

Secondly, the similar quality characteristic groups are merged together in order to avoid, or at least to reduce, the lexical and semantic overlap between quality characteristic groups. This behavior is not only aligned with the concept of word constellations but also with the gene variations of same genetic character. For example, “Efficiency” and “Performance efficiency” have similar definition and sub-characteristics. Therefore, we can conclude that they both cover the same quality characteristic concept and be regrouped then under “Efficiency”.

The regrouping results are shown in Table 33, and its consequence is a decrease of 12 distinct quality characteristic groups, moving from 55 to 43 distinct quality characteristic groups for the same quality modeling coverage. Moreover, 27 over the 43 groups (i.e., 62.28%) come from only one quality model, and 10 over the 43 groups (i.e., 23.26%) are quality characteristic groups found at least in four of the eight quality models, with three of these groups (i.e., “Efficiency”, “Reliability” and “Usability”) present in the eight quality models.

TABLE 33 - THE 10 MERGED QUALITY CHARACTERISTIC GROUPS (GREEN BACKGROUND CELL INDICATES A MERGED ENTRY)

Total	Quality Characteristic Name	Boehm [42]	McCall [41]	FURPS [85]	ISO/IEC 9126 [24]	Alvaro [36]	Bawane [103]	Kalaimagal Q'Facto 12 [102]	ISO/IEC/IEEE 25010 [23]
2	Compliance					Compliance		Standardization / Certification	
8	Efficiency	Efficiency	Efficiency	Performance	Efficiency	Efficiency	Efficiency	Efficiency	Performance efficiency
2	Fault-tolerance					Fault Tolerance		Recoverability	
6	Functionality			Functionality	Functionality	Functionality	Functionality	Functionality	Functional suitability
3	Interoperability		Interoperability					Interoperability	Compatibility
7	Portability	Portability	Portability		Portability	Portability	Portability	Portability	Portability
4	Product in use				Product in use		Product in use	Usability in use	Quality in use
4	Security		Integrity (security)			Security		Security	Security
3	Software Product				Software Product		Software Product		System / Software product
8	Usability	Usability (As-is utility)	Usability	Usability	Usability	Usability	Usability	Usability	Usability

Finally, these 43 quality characteristic groups can be transformed into quality characteristic genes with their own quality characteristic sites (i.e., equivalent to quality sub-characteristics as shown in Chapter VIII.2) and their probability to happen.

Thus, the gene sites result from the aggregation of the quality sub-characteristics of the current quality characteristic genes, regrouping the lexically and semantically similar sub-characteristics, or sites, together like for the genes. An example with regards to “Portability” gene is described by Table 29 and Table 30, and the complete list of genes with their sites, including their lexical and semantic variations, is given into Table 50 and Table 51 of Annex 14.

The likelihood of a gene is determined by the number of its occurrence over the total number of merged quality models -in our current case this is 8. So, for “Adaptability” gene, which is present only in Alavaro’s quality model, the probability to happen is 0.125 (i.e., $\frac{1}{8} = 0.125$). About “Portability” gene, found in all the selected quality models except FURPS, its probability is 0.875 (i.e., $\frac{7}{8} = 0.875$). Concerning the site probability, the calculations are identical and an example related to “Portability” gene is given in Table 30.

- The final result is a list of 43 genes split into two parts:
- A list of the 27 genes resulting from a single quality model (cf. Table 34). The likelihood of these gene sites is therefore 1 since each gene is only present in one quality model. There is a total of 81 sites for these 27 genes, which means an average of 3 sites per gene with a minimum of 2 sites (e.g., “Adaptability” gene) and a maximum of 9 sites for “Supportability” gene. To facilitate identification in later usage, we identify these genes with the prefix “A” and a number from 1 to 27.
 - A list of the 16 genes resulting from more than one quality model (cf. Table 35). There is a total of 128 sites, and consequently an average of 8 sites per gene, with a minimum of 2 sites for “Compliance” gene and a maximum of 15 sites for “Testability” gene. To facilitate identification in later usage, we identify these genes with the prefix “B” and a number from 1 to 16.

TABLE 34 - LIST OF THE 27 SINGLE QUALITY MODEL GENES

Gene ID	Type	Quality Characteristic Name	Probability	Gene ID	Type	Quality Characteristic Name	Probability						
A01	Gene	Adaptability	0.125	A16	Gene	Product operation	0.125						
		site 1	Mobility			1.0	site 1	Correctness	1.0				
		site 2	Configuration capacity			1.0	site 2	Reliability	1.0				
A02	Gene	Changeability	0.125	site 3	Efficiency	1.0	A17	Gene	Product revision	0.125			
		site 1	Extensibility	1.0	site 4	Integrity (security)					1.0		
		site 2	Customizability	1.0	site 5	Usability					1.0		
site 3	Modularity	1.0	site 1	Maintainability	1.0	A18	Gene	Product transition	0.125				
A03	Gene	Context coverage	0.125	site 2	Flexibility					1.0	site 1	Portability	1.0
				site 1	context completeness					1.0	site 2	Reusability	1.0
site 2	Flexibility	1.0	site 3	Testability	1.0	A19	Gene	Resource behavior	0.125				
A04	Gene	Controllability	0.125	site 1	Traceability					1.0	site 1	Memory utilization	1.0
				site 1	Component execution control					1.0	site 2	Disk utilization	1.0
				site 2	Component environment control	1.0	A20	Gene	Safety in use	0.125	site 1	Risk of software	1.0
site 3	Component function feature control	1.0	site 2	Commercial risk in use	1.0								
A05	Gene	Correctness	0.125	site 3	Completeness	1.0					site 3	Risk to the operation in use	1.0
				site 1	Traceability	1.0	site 4	Risk to the public in use	1.0				
				site 2	Consistency	1.0	A21	Gene	Satisfaction	0.125	site 1	Usefulness	1.0
A06	Gene	Freedom from risk	0.125	site 1	Economic risk mitigation	1.0							
				site 2	Health and safety risk Mitigation	1.0							
				site 3	Environmental risk mitigation	1.0							
A07	Gene	Flexibility	0.125	site 1	Modularity	1.0							

Meta-Model: Software Quality Model Genome

	site 2	Generality	1.0		site 2	Trust	1.0
	site 3	Expandability	1.0		site 3	Pleasure	1.0
	site 4	Self-Descriptiveness	1.0		site 4	Comfort	1.0
A08	Gene	General utility	0.125	A22	Gene	Self-Contained	0.125
	site 1	Portability	1.0		site 1	Presence of precondition & postconditions	1.0
	site 2	Usability (As-is utility)	1.0		site 2	Modularity	1.0
	site 3	Maintainability	1.0	A23	Gene	Suitability	0.125
A09	Gene	Generality	0.125		site 1	Coverage	1.0
	site 1	Presence of domain abstraction	1.0		site 2	Completeness	1.0
	site 2	Reuse history	1.0		site 3	Pre- and Post-conditioned	1.0
A10	Gene	Hardware/Software Independence	0.125		site 4	Proofs of Pre- and Post-conditions	1.0
	site 1	Hardware independence	1.0	A24	Gene	Supportability	0.125
	site 2	Software independence	1.0		site 1	Testability	1.0
A11	Gene	Human engineering	0.125		site 2	Extensibility	1.0
	site 1	Robustness / Integrity	1.0		site 3	Adaptability	1.0
	site 2	Accessibility	1.0		site 4	Maintainability	1.0
	site 3	Communicativeness	1.0		site 5	Compatibility	1.0
A12	Gene	Instalability	0.125		site 6	Configurability	1.0
	site 1	Instalability documentation	1.0		site 7	Serviceability	1.0
	site 2	Instalability complexity	1.0		site 8	Instability	1.0
A13	Gene	Learnability	0.125		site 9	Localizability	1.0
	site 1	Training	1.0	A25	Gene	Test documentation	0.125
	site 2	Presence of demonstration	1.0		site 1	Presence of test suites	1.0
A14	Gene	Maturity	0.125		site 2	Proofs of previous tests	1.0
	site 1	Volatility	1.0	A26	Gene	Time behavior	0.125
	site 2	Failure removal	1.0		site 1	Response time	1.0
A15	Gene	Modifiability	0.125		site 2	Latency throughput ("out")	1.0
	site 1	Structuredness	1.0		site 3	Latency processing capacity ("in")	1.0
	site 2	Augmentability	1.0	A27	Gene	Traceability	0.125
					site 1	Error trace	1.0
					site 2	Performance trace	1.0

TABLE 35 - LIST OF THE 16 MULTI-QUALITY MODELS GENES

Gene ID	Type	Quality Characteristic Name	Probability	Gene ID	Type	Quality Characteristic Name	Probability
B01	Gene	Efficiency	1.0	B10	Gene	Reusability	0.5
	site 1	Time-behavior	0.75		site 1	Self-Descriptiveness	0.75
	site 2	Resource utilization	0.625		site 2	Generality	0.5
	site 3	Storage efficiency	0.25		site 3	Modularity	0.5
	site 4	Execution efficiency	0.25		site 4	Software independence	0.5
	site 5	Efficiency Compliance	0.25		site 5	Hardware independence	0.5
	site 6	Capacity	0.125		site 6	Coupling	0.5
	site 7	Accountability	0.125		site 7	Domain abstraction level	0.25
	site 8	Accessibility	0.125		site 8	Architecture compatibility	0.25
	site 9	Scalability	0.125		site 9	Cohesion	0.25
	site 10	Availability	0.125		site 10	Locability	0.25
	site 11	Time of answers	0.125		site 11	Interoperability	0.25
	site 12	Time of recovery	0.125	B11	Gene	Security	0.5
B02	Gene	Fault tolerance	0.25		site 1	Confidentiality	1
	site 1	Mechanism availability	1.0		site 2	Auditability	0.5
	site 2	Mechanism efficiency	0.5		site 3	Cipherability	0.5
	site 3	Persistence	0.5		site 4	Integrity	0.25
B03	Gene	Functionality	0.75		site 5	Non-repudiation	0.25
	site 1	Accuracy	0.8333		site 6	Accountability	0.25
	site 2	Security	0.6667		site 7	Authenticity	0.25

site 3	Suitability	0.6667	B12 Gene	Software product	0.375
site 4	Interoperability	0.5	site 1	Functionality	1.0
site 5	Compliance	0.5	site 2	Usability	1.0
site 6	Self-contained	0.3333	site 3	Reliability	1.0
site 7	Functional Completeness	0.3333	site 4	Efficiency	1.0
site 8	Capacities	0.1667	site 5	Maintainability	1.0
B04 Gene	Interoperability	0.375	site 6	Portability	1.0
site 1	Data commonality	0.6667	site 7	Compatibility	0.3333
site 2	Communication commonality	0.6667	site 8	Security	0.3333
site 3	Co-existence	0.6667	site 9	Reusability	0.3333
site 4	Modularity	0.33333	B13 Gene	Compliance	0.25
site 5	Version compatibility	0.33333	site 1	Standardization	1.0
B05 Gene	Maintainability	0.875	site 2	Certification	1.0
site 1	Changeability	0.75	B14 Gene	Testability	0.5
site 2	Testability	0.625	site 1	Self-descriptiveness	0.5
site 3	Analyzability	0.625	site 2	Accountability	0.25
site 4	Stability	0.5	site 3	Accessibility	0.25
site 5	Modularity	0.25	site 4	Communicativeness	0.25
site 6	Reusability	0.125	site 5	Structuredness	0.25
site 7	Consistency	0.125	site 6	Simplicity	0.25
site 8	Conciseness	0.125	site 7	Modularity	0.25
site 9	Self-Descriptiveness	0.125	site 8	Instrumentation	0.25
site 10	Maintainability Compliance	0.125	site 9	Test suite provided	0.25
B06 Gene	Operability	0.25	site 10	Extensive component test case	0.25
site 1	Effort to operate	1.0	site 11	Component tests in a specific environment	0.25
site 2	Complexity level	0.5	site 12	Proofs the components tests	0.25
site 3	Provided Interfaces	0.5	site 13	Test documentation	0.25
site 4	Required Interfaces	0.5	site 14	Controllability	0.25
site 5	Effort to configure	0.5	site 15	Traceability	0.25
B07 Gene	Portability	0.875	B15 Gene	Understandability	0.25
site 1	Adaptability	0.625	site 1	Self-descriptiveness	1
site 2	Replaceability	0.625	site 2	Consistency	0.5
site 3	Instability	0.625	site 3	Structuredness	0.5
site 4	Safety	0.5	site 4	Conciseness	0.5
site 5	Freedom from risk	0.25	site 5	Legibility	0.5
site 6	Machine independence	0.25	site 6	Documentation availability	0.5
site 7	Self-Descriptiveness	0.25	site 7	Documentation readability and quality	0.5
site 8	Software system independence	0.125	B16 Gene	Usability	1.0
site 9	Co-existence	0.125	site 1	Learnability	0.875
site 10	Reusability	0.125	site 2	Operability	0.75
B08 Gene	Product in use	0.5	site 3	Understandability	0.75
site 1	Effectiveness	1	site 4	Attractiveness	0.5
site 2	Satisfaction	1	site 5	Human engineering	0.5
site 3	Productivity	0.75	site 6	Reliability	0.25
site 4	Safety	0.5	site 7	Usability compliance	0.125
site 5	Freedom from risk	0.25	site 8	Efficiency	0.125
B09 Gene	Reliability	1.0			
site 1	Fault tolerance	0.875			
site 2	Recoverability	0.625			
site 3	Maturity	0.5			
site 4	Accuracy	0.25			
site 5	Consistency	0.25			
site 6	Availability	0.25			
site 7	Reliability Compliance	0.25			
site 8	Simplicity	0.25			
site 9	Frequency and severity of failures	0.125			

c. Relationship links between genes and chromatid identification

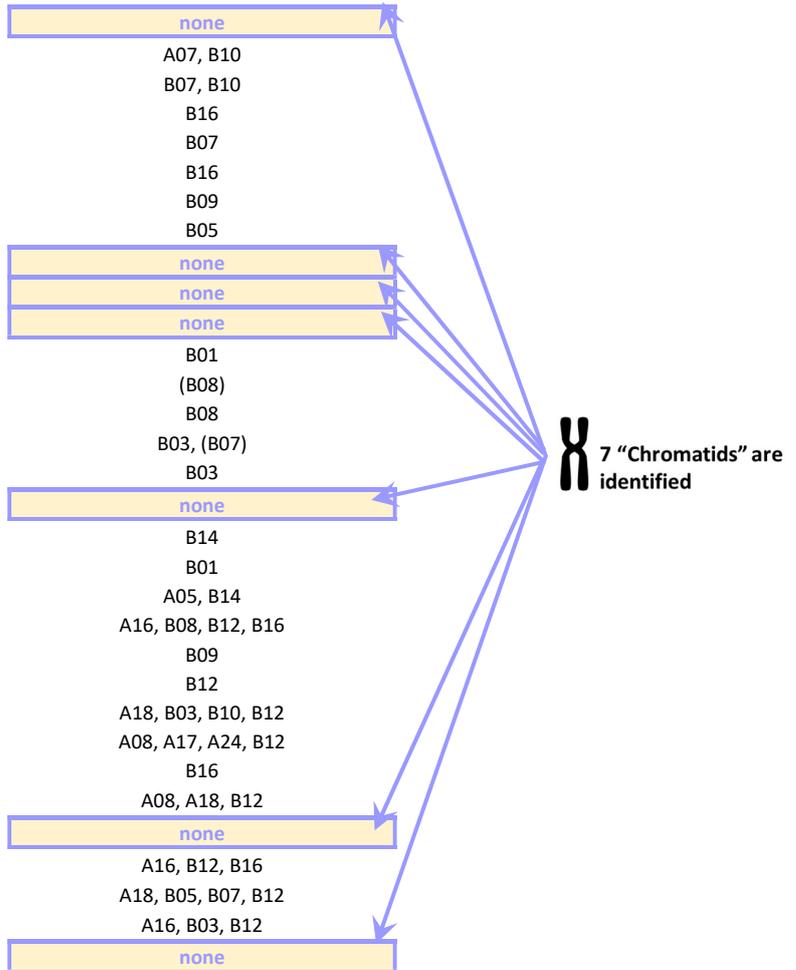
Following the identification, construction, and definition of the 43 quality characteristic genes, the next step in the construction of the software quality model genome meta-model is to look for any chromatid. A chromatid corresponds to a quality perspective and can be retrieved as a top-level gene (i.e., the gene is not a child, or site, of another gene).

So, to determine which gene, if any, is a top-level gene, we must find first which genes are sites of other genes, and finally if a gene is not belonging, or linked to any other gene(s), we can conclude that it is a top-level gene, or chromatid. To retrieve these gene relationship links, we look for quality characteristics that are quality sub-characteristics in other quality characteristic(s), taking into account also the fact that a characteristic, or sub-characteristic, can vary in its naming (see Table 50 and Table 51 of Annex 14).

The result of this analysis, synthetized in Table 36, shows 7 chromatids: A08 “General utility”, A16 “Product operation”, A17 “Product revision”, A18 “Product transition”, A24 “Supportability”, B08 “Product in use”, and B12 “Software product”.

TABLE 36 - THE RELATIONSHIP LINKS BETWEEN THE QUALITY CHARACTERISTIC GENES

Gene ID	Quality Characteristic Name	Found in gene(s)
A01	Adaptability	A24, B07
A02	Changeability	B05
A03	Context coverage	B08
A04	Controllability	B11, B14
A05	Correctness	A16, B03
A06	Freedom from risk	B08
A07	Flexibility	A03, A17
A08	General utility	none
A09	Generality	A07, B10
A10	Hardware/Software Independence	B07, B10
A11	Human engineering	B16
A12	Instalability	B07
A13	Learnability	B16
A14	Maturity	B09
A15	Modifiability	B05
A16	Product operation	none
A17	Product revision	none
A18	Product transition	none
A19	Resource behavior	B01
A20	Safety in use	(B08)
A21	Satisfaction	B08
A22	Self-Contained	B03, (B07)
A23	Suitability	B03
A24	Supportability	none
A25	Test documentation	B14
A26	Time behavior	B01
A27	Traceability	A05, B14
B01	Efficiency	A16, B08, B12, B16
B02	Fault tolerance	B09
B03	Functionality	B12
B04	Interoperability	A18, B03, B10, B12
B05	Maintainability	A08, A17, A24, B12
B06	Operability	B16
B07	Portability	A08, A18, B12
B08	Product in use	none
B09	Reliability	A16, B12, B16
B10	Reusability	A18, B05, B07, B12
B11	Security	A16, B03, B12
B12	Software product	none
B13	Compliance	B03, (B01, B05, B07, B09, B16)
B14	Testability	A17, A24, B05
B15	Understandability	B05, B16



Note, in “found in gene(s)” column of Table 36, there are few genes in parenthesis. The parenthesis indicates that the relationship between the two genes (i.e., gene as site and gene as parent gene) require to take into account the context of the parent gene. For example, for gene A20 “Safety in use”, we can find it as a site of gene B08 “Product in use”. However, in “Product in use” gene the corresponding site to “Safety in use” is site 4 “Safety” (see Table 51 from Annex 14). Thus, the parenthesis points out that a context nuance must be considered carefully when using the “Safety in use” gene to detail “Safety” site.

A visual representation of Table 36 is performed through the oriented graph representation shown by Figure 84. Each node is one of the 43 genes, the root nodes with a chromosome symbol are the chromatids, and the arrows depict the links between genes. The beginning of each arrow represents the gene as site, and the dotted arrows signal that the parent gene context has to be considered (i.e., the dotted arrows are equivalent to the parenthesis highlight).

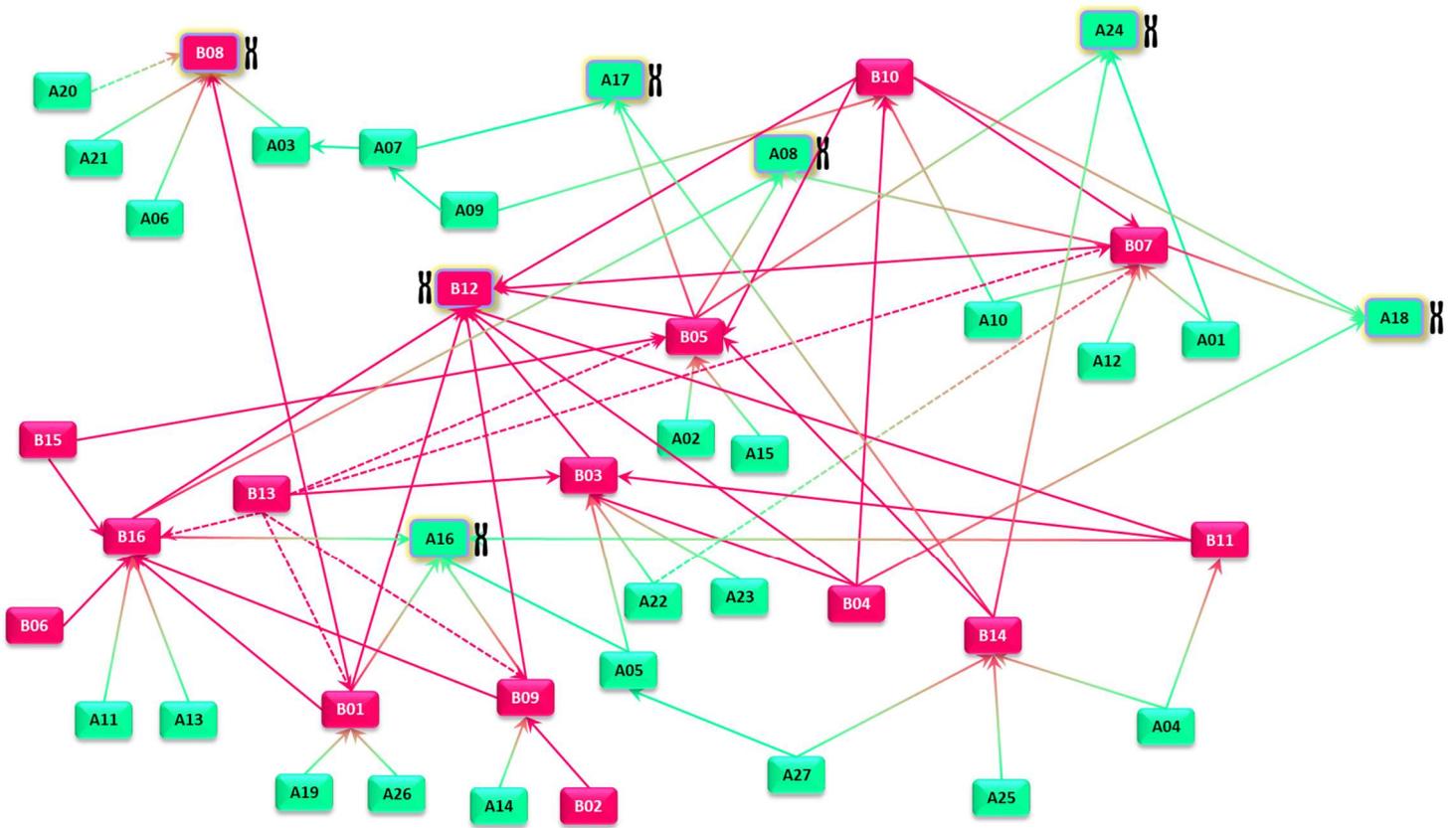


Figure 84 – Links between the 43 genes; chromatids are identified by X and dotted arrows indicate a relationship link that requires to take into account parent gene context

The remaining task to finalize the software quality model genome meta-model is the construction of each these seven chromatids elaborated from this graph representation and the full gene details from Table 34 and Table 35.

5. Contributions: The 7 Chromatids of SW Quality Model Genome

Research Sub-question 4c	What is the first result of the meta-model construction?
---------------------------------	--

This section is the concluding construction step of the software quality model genome meta-model. From the selected set of eight quality models, we successively extracted 43 quality characteristic genes with their own sites, or quality sub-characteristics, calculated their likelihood values, determined the relationship links between all genes and identified seven chromatids.

Thus, this final step consists in detailing each of these seven chromatids that constitute the software quality model genome meta-model.

From the complete linked gene map shown through Figure 84, we extract a subset of this map by isolating only the genes that are directly or indirectly linked to a specific chromatid. Once this sub-map done, we generate the detailed chromatid site sequence, including probability values, by enumerating each gene sites starting from the chromatid up to the furthest linked genes (see example in Figure 82 about a site enumeration beginning for "Supportability" chromatid). Figure 85 illustrates that final chromatid creation chain.

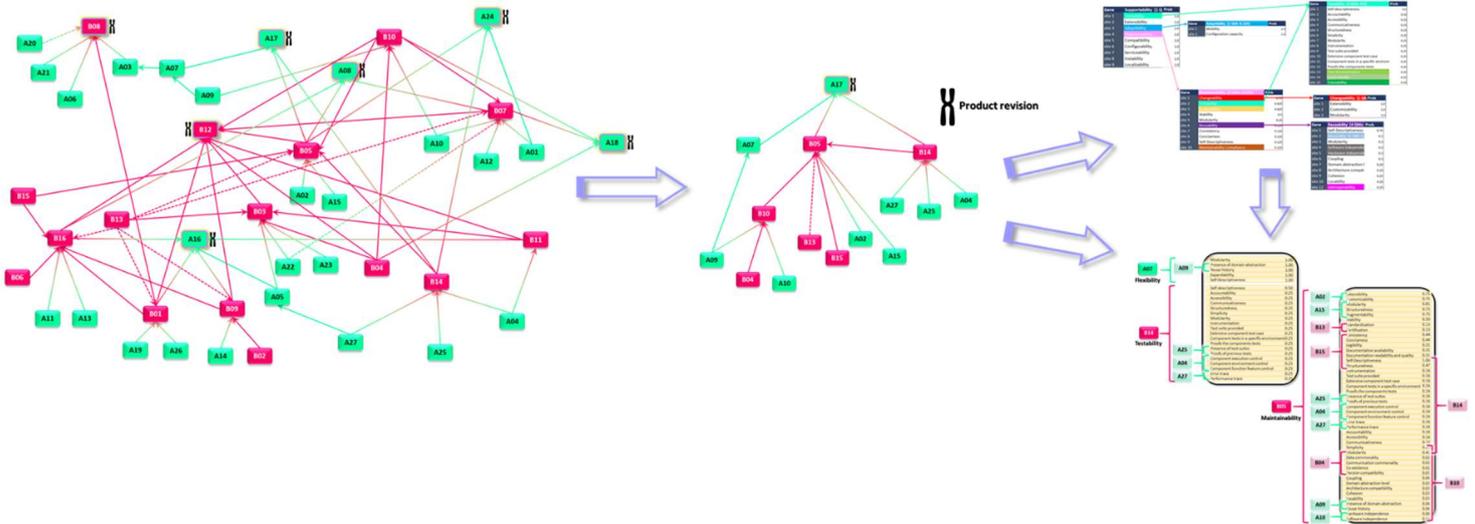


Figure 85 - From the 43 linked genes to a specific the chromatid linked genes and the corresponding site sequence

So, the final expected result for each chromatid is a complete detailed sequence of sites with their likelihood values and the localization of each gene, and any sub-sequent levels of genes, composing the chromatid.

However, during the enumeration task of the site sequence, there are three aspects to be caution when replacing a site by the corresponding gene and its sites. For example, in gene A24 "Supportability", its site 4 is "Maintainability", but this site can be replaced by gene B05 "Maintainability" which possesses itself 10 sites.

The first aspect concerns the site likelihood values. In our example, when site 4 of A24 is replaced by a detailed version made of the 10 sites of B05, each likelihood values of these 10 sites are combined, or multiplied by the likelihood of site 4 (e.g., site 1 "Changeability" of B05 is 0.875 and in A024 gene scope this value becomes $0.875 * 1.0$ (from site 4) = 0.875). In the case of more levels of site-gene, we propagate the calculation over all the levels.

The second aspect is about the optimization of the number of identical sites. Indeed, again with our example, in gene B05, its site 5 is "Modularity" which is also present as site 7 of gene B14 "Testability" which replace "Testability" site 2 in gene B05. Consequently, we sum the two probability values with a maximum of 1, paying attention that we must first combine likelihood values of site 2 of B05 with site 7 of B14 (i.e., site replaced by a gene with its site and therefore it requires to combine together the likelihood values).

The third aspect is related to the identical site optimization number. In fact, two identical sites sometimes cannot be merged into one site because the result will be an incompatible gene overlap. Furthermore, since one of the expected results for chromatid is to be able to localize each gene which composes the chromatid, and not only to list all the distinct sites that compose the chromatid, a gene must be localized accurately at a specific locus but also its sequence must be contiguous. Therefore, if a merge of two identical sites may result in the failure of one of these two conditions, then the two identical sites must remain unmerged. Note, the site optimization may result in the overlap of some genes whoever, we impose that the genes directly linked to the chromatid must not be overlapped to retrieve the exact main quality characteristics of the chromatid.

Table 37 gives an example of two site sequence results for gene B05 "Maintainability". The first one, on the left, is the sequence obtained by replacing each site by the corresponding gene and its sites. The second one, on the right, is the same sequence but optimized to reduce the number of similar sites. The quality coverage is identical, and we remark that, for instance, we reduced the number of "modularity" sites from five occurrence to only two.

TABLE 37 - EXAMPLE OF GENE 05 "MAINTAINABILITY" DETAILED WITH ALL ITS SUB-GENES AND SITES: ON THE LEFT SIDE, THE DIRECT DETAILED SITE SEQUENCE, ON THE RIGHT THE DETAILED AND OPTIMIZED SITE SEQUENCE WITH SAME QUALITY COVERAGE

Detailed gene B05 "Maintainability"		Detailed and optimized gene B05 "Maintainability"		
Genes	Sites	Genes	Sites	
B05	A02	B05	Extensibility	
	A15		Customizability	
			Modularity	
	Structuredness			
	B14		Augmentability	
			Self-descriptiveness	
			Accountability	
			Accessibility	
			Communicativeness	
			Structuredness	
			Simplicity	
			Modularity	
			Instrumentation	
			Test suite provided	
	B15		Extensive component test case	
			Component tests in a specific environment	
			Proofs the components tests	
			A25	
			Presence of test suites	
			Proofs of previous tests	
			A04	Component execution control
				Component environment control
			A27	Component function feature control
				Error trace
	B10		Performance trace	
			Code Readability	
			Consistency	
Structuredness				
B15		Conciseness		
		Legibility		
Documentation availability				
Documentation readability and quality				
Stability				
Modularity				
Self-Descriptiveness				
A09				
Presence of domain abstraction				
Reuse history				
B10	Modularity			
	A10			
Hardware independence				
Software independence				
Coupling				
Domain abstraction level				
Architecture compatibility				
Cohesion				
Locability				
B05	B14	B05	Extensibility	
			A02	
			Customizability	
			Modularity	
			Structuredness	
			A15	
			Augmentability	
			Stability	
			B13	
			Standardization	
			Certification	
			Consistency	
			Conciseness	
			Legibility	
			B15	
			Documentation availability	
			Documentation readability and quality	
			Self-Descriptiveness	
			Structuredness	
			Instrumentation	
			Test suite provided	
			Extensive component test case	
			Component tests in a specific environment	
			Proofs the components tests	
			A25	
			Presence of test suites	
			Proofs of previous tests	
A04	Component execution control			
	Component environment control			
A27	Component function feature control			
	Error trace			
B10	Performance trace			
	Code Readability			
	Consistency			
	Structuredness			
	B15	Conciseness		
		Legibility		
	Documentation availability			
	Documentation readability and quality			
	Stability			
	Modularity			
Self-Descriptiveness				
B04				
Presence of domain abstraction				
Reuse history				
B10	Modularity			
	A09			
Hardware independence				
Software independence				
Coupling				
Domain abstraction level				
Architecture compatibility				
Cohesion				
Locability				
B05	B10	B05	Extensibility	
			A09	
			Reuse history	
			A10	
			Hardware independence	
			Coupling	
			Domain abstraction level	
			Architecture compatibility	
			Cohesion	
			Locability	
			Presence of domain abstraction	
			Reuse history	
			A10	
			Hardware independence	
			Coupling	
			Domain abstraction level	
			Architecture compatibility	
			Cohesion	
			Locability	
			Presence of domain abstraction	
			Reuse history	
			A09	
			Reuse history	
			A10	
			Hardware independence	



By applying that final construction step against each of the seven chromatids, we are providing the sub-map of chromatid linked genes and their corresponding site sequence, including likelihood values, in the following seven sub-sections.

a. "General Utility" Chromatid (A08)

General utility perspective "reflects the actual uses to which evaluation of software quality would be put. In general, when one is acquiring a software package, one is mainly concerned with three questions:

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

(Source : Boehm et al. [42]).

This chromatid is composed of 27 quality characteristic genes linked together as depicted by Figure 86, subset of the graph shown in Figure 84, and its complete "DNA" sequence made of 114 quality characteristic sites disclosed in Figure 87.

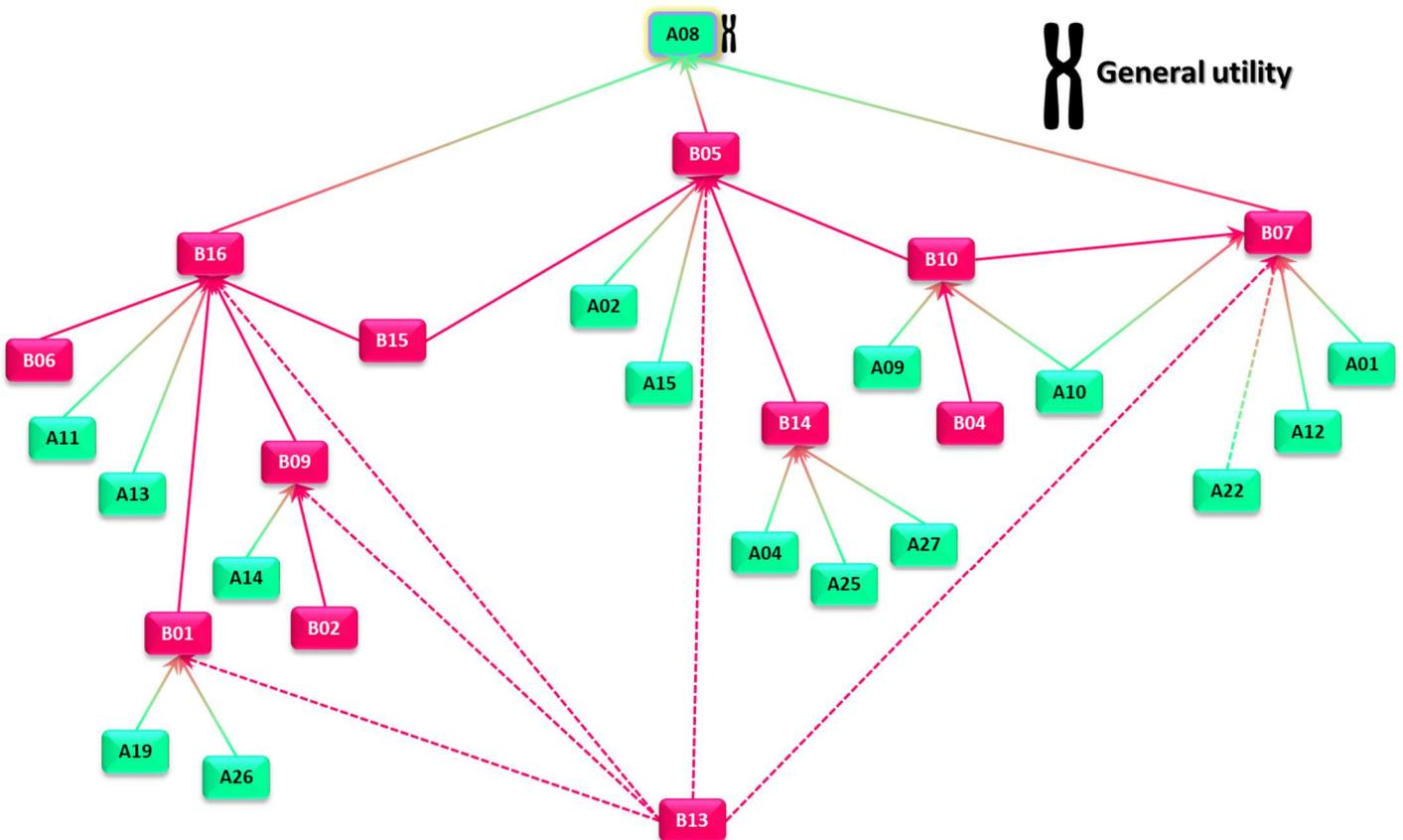


Figure 86 - The 27 genes of "General utility" chromatid, with their respective links

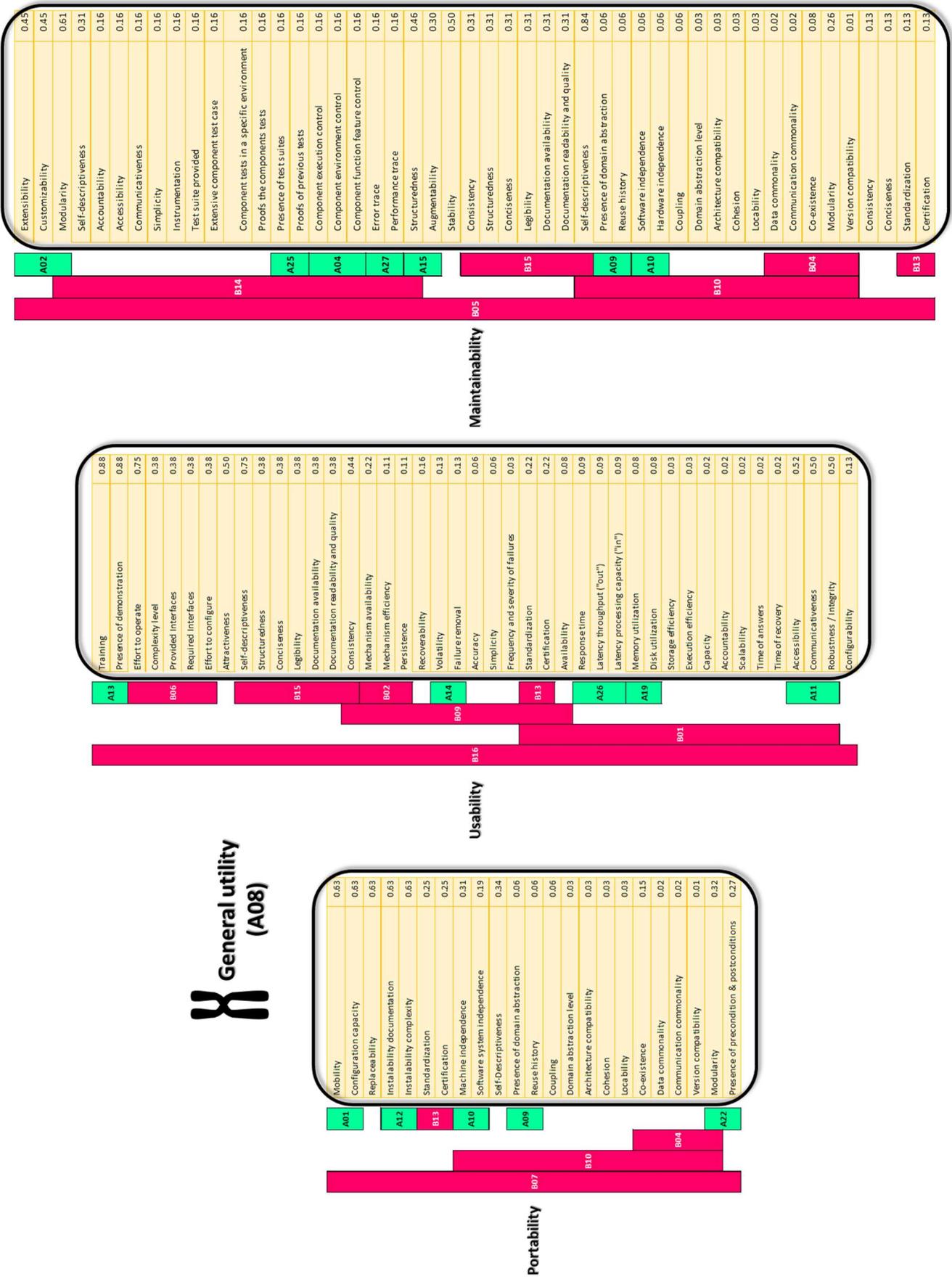


Figure 87 - "General utility" chromatid quality characteristic sequence with its 27 genes and the 114 sites, including sites likelihood

b. "Product Operation" Chromatid (A16)

The product operations perspective identifies "quality factors that influence the extent to which the software fulfils its specification" (source: <http://www.sqa.net/softwarequalityattributes.html> and McCall et al. [41]).

This chromatid is composed of 17 quality characteristic genes linked together as depicted by Figure 88, subset of the graph shown in Figure 84, and its complete "DNA" sequence made of 85 quality characteristic sites disclosed in Figure 89.

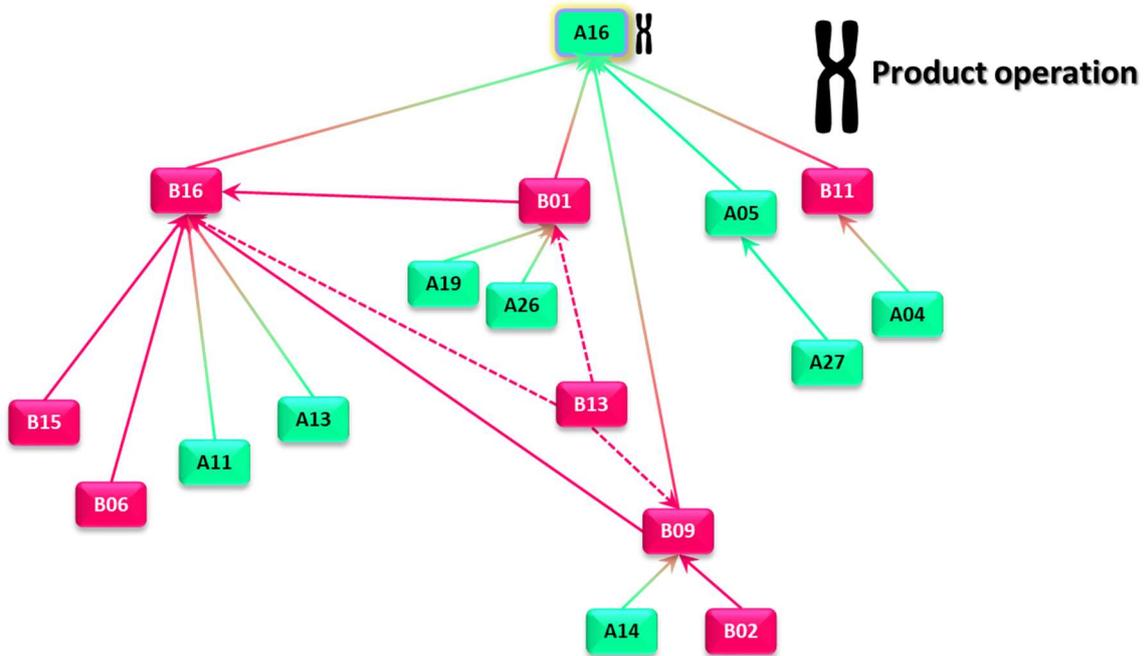


Figure 88 - The 17 genes of "Product operation" chromatid, with their respective links

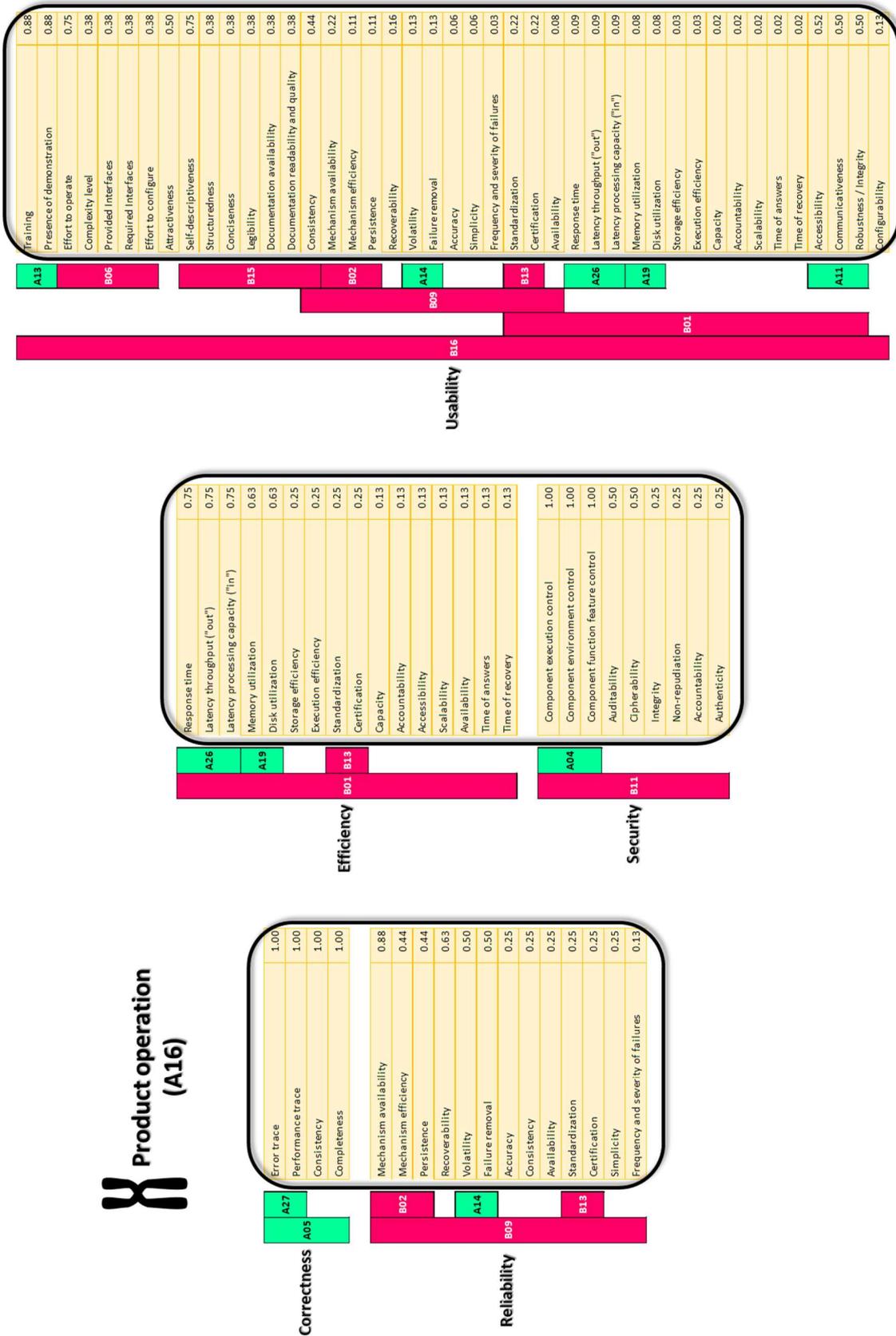


Figure 89 - "Product operation" chromatid quality characteristic sequence with its 17 genes and the 85 sites, including sites likelihood

c. "Product Revision" Chromatic (A17)

The product revision perspective identifies "quality factors that influence the ability to change the software product" (source: <http://www.sqa.net/softwarequalityattributes.html> and McCall et al. [41]).

This chromatid is composed of 15 quality characteristic genes linked together as depicted by Figure 90, subset of the graph shown in Figure 84, and its complete "DNA" sequence made of 69 quality characteristic sites disclosed in Figure 91.

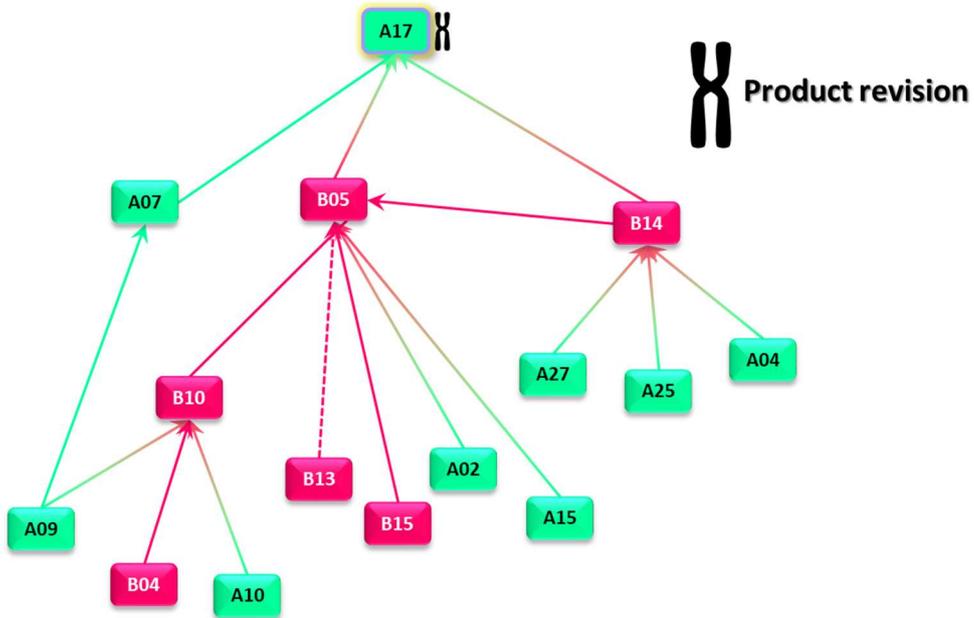


Figure 90 - The 15 genes of "Product revision" chromatid, with their respective links

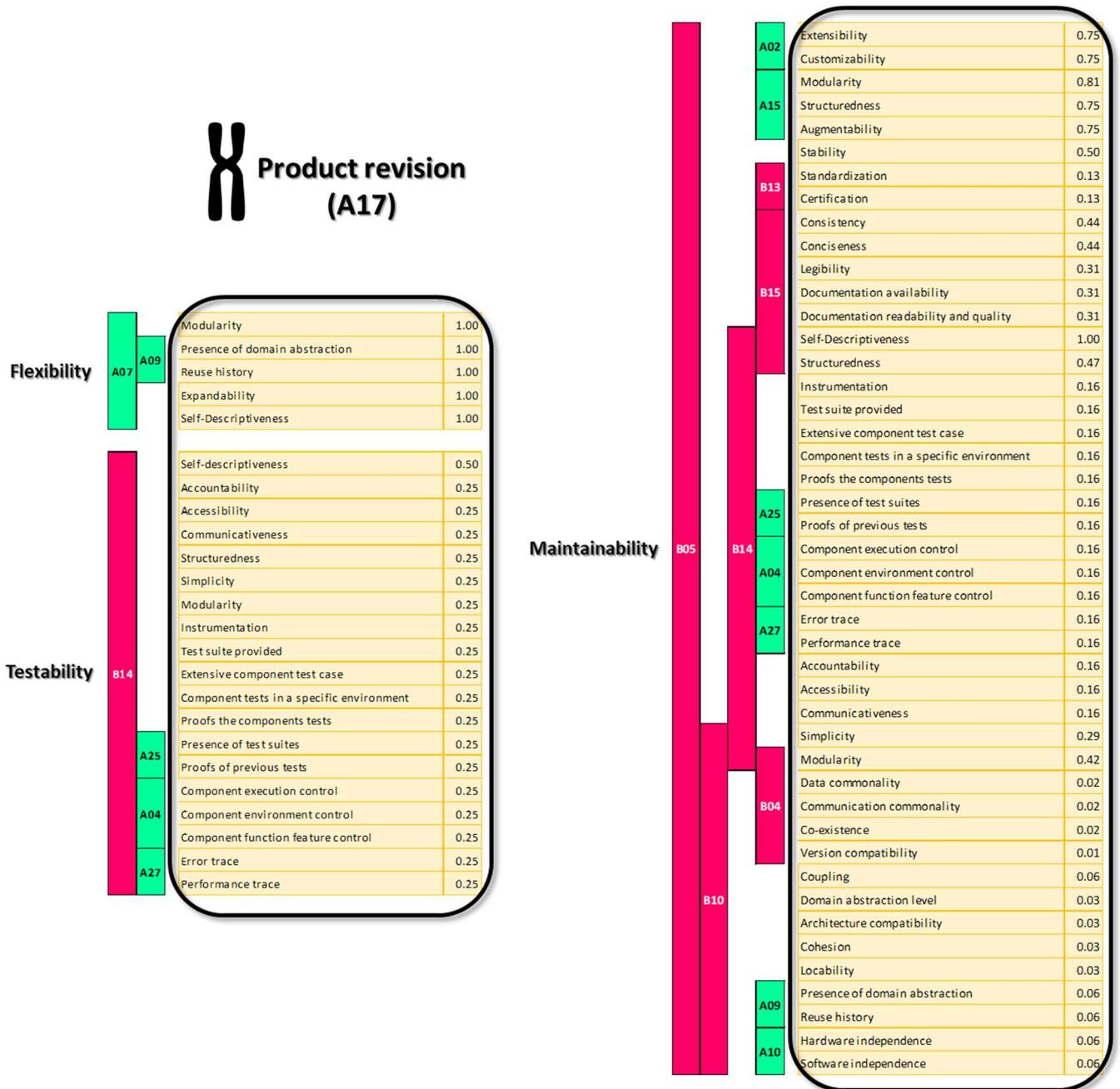


Figure 91 - "Product revision" chromatid quality characteristic sequence with its 15 genes and the 96 sites, including sites likelihood

d. "Product Transition" Chromatid (A18)

The product transition perspective identifies "quality factors that influence the ability to adapt the software to new environments" (source: <http://www.sqa.net/softwarequalityattributes.html> and McCall et al. [41]).

This chromatid is composed of 10 quality characteristic genes linked together as depicted by Figure 92, subset of the graph shown in Figure 84, and its complete "DNA" sequence made of 43 quality characteristic sites disclosed in Figure 93.

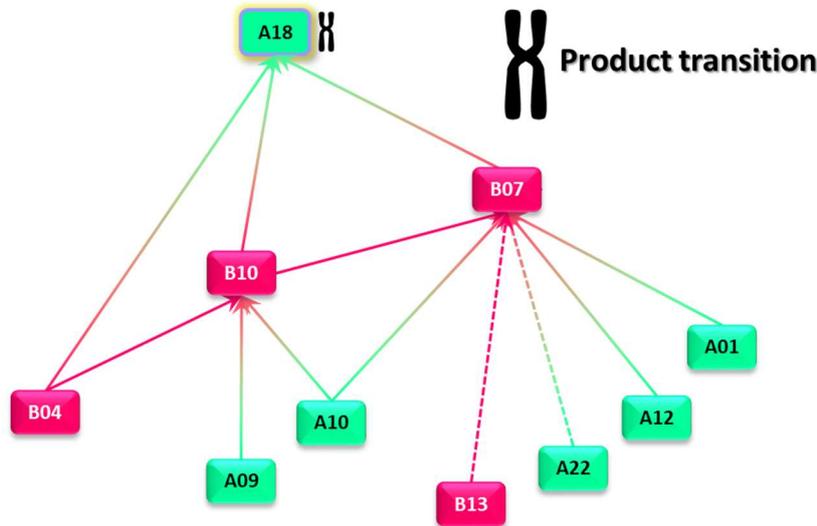


Figure 92 - The 10 genes of "Product transition" chromatid, with their respective links

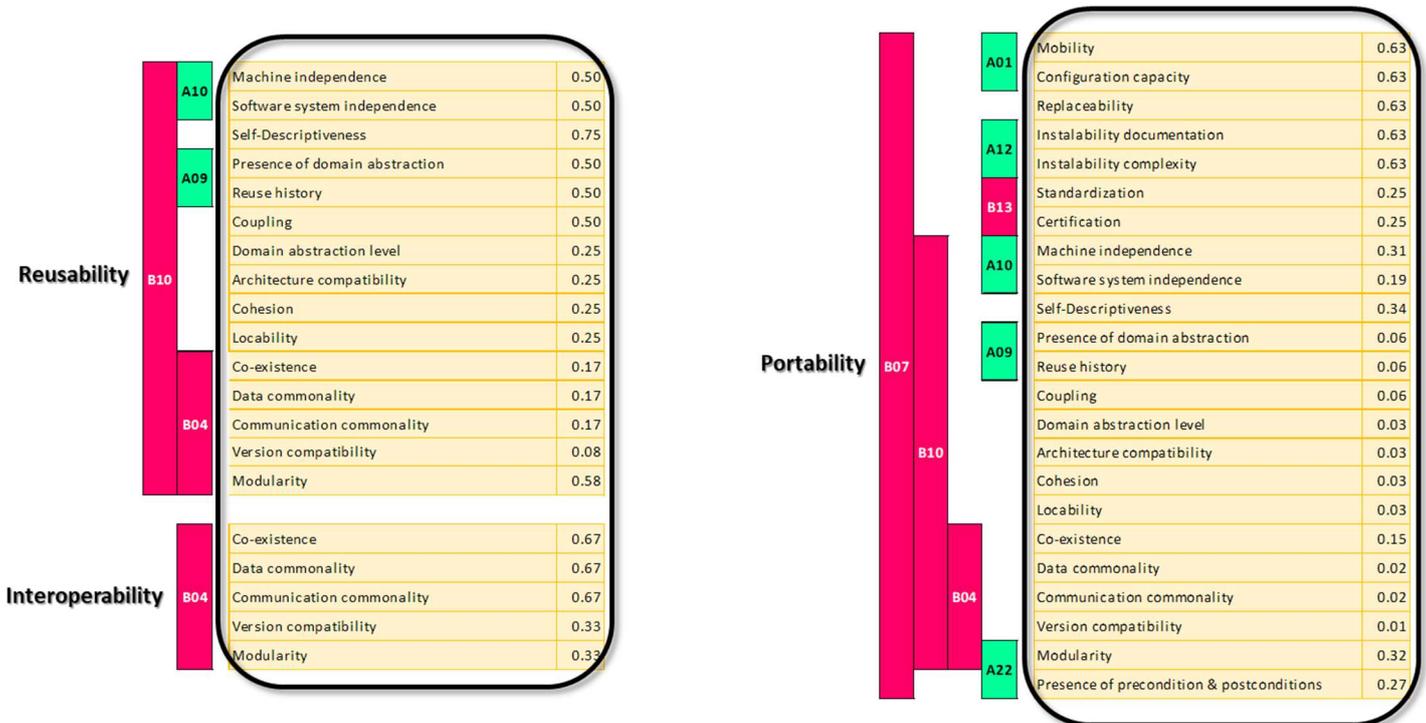
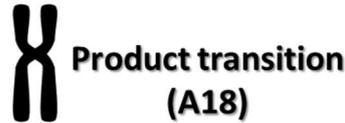


Figure 93 - "Product transition" chromatid quality characteristic sequence with its 10 genes and the 43 sites, including sites likelihood

e. "Supportability" Chromatid (A24)

Supportability perspective can be defined through a set of questions "Is it testable, extensible, serviceable, installable, and configurable? Can it be monitored? How will system be extended? Who maintains the system?" (source: Grady and Caswell [85]).

This chromatid is composed of 15 quality characteristic genes linked together as depicted by Figure 94, subset of the graph shown in Figure 84, and its complete "DNA" sequence made of 72 quality characteristic sites disclosed in Figure 95.

Note, this chromatid is close to "product revision" chromatid. Indeed, they are around 89% similar and thus "Supportability" chromatid is a polymorphism variation of "product revision" chromatid, and vice-versa.

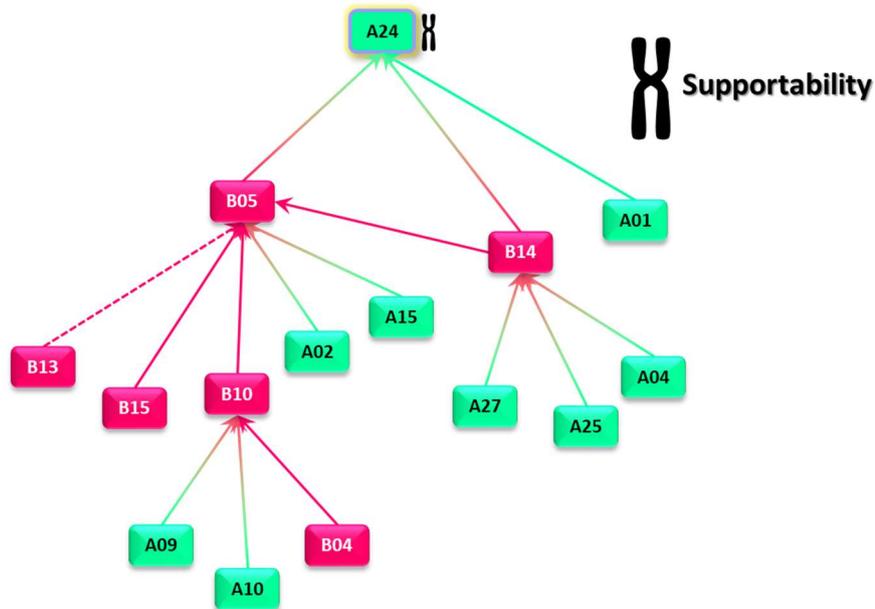


Figure 94 - The 15 genes of "Supportability" chromatid, with their respective links

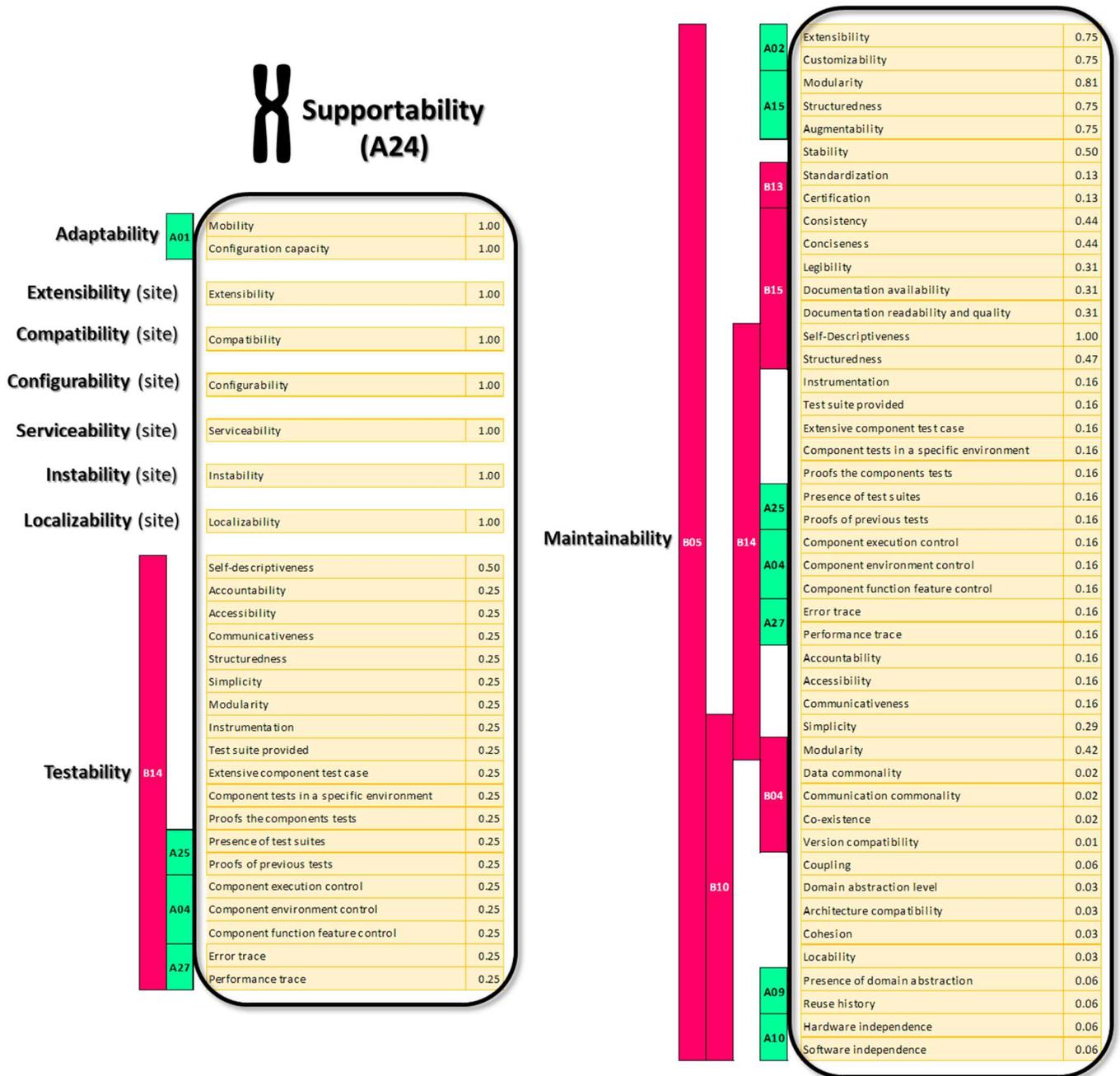


Figure 95 - "Supportability" chromatid quality characteristic sequence with its 15 genes and the 72 sites, including sites likelihood

f. "Product in Use" Chromatid (B08)

Product in use perspective "is the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use" (source: ISO/IEC 25010 [23]).

This chromatid is composed of 11 quality characteristic genes linked together as depicted by Figure 96, subset of the graph shown in Figure 84, and its complete "DNA" sequence made of 34 quality characteristic sites disclosed in Figure 97.

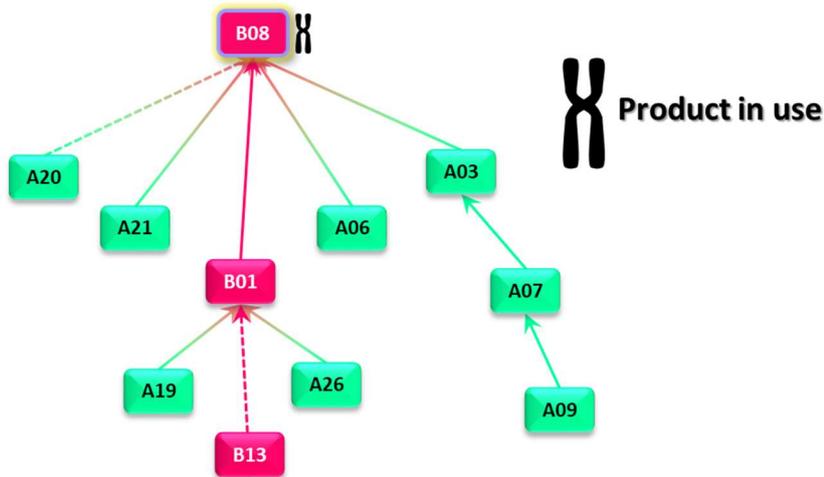


Figure 96 - The 11 genes of "Product in use" chromatid, with their respective links

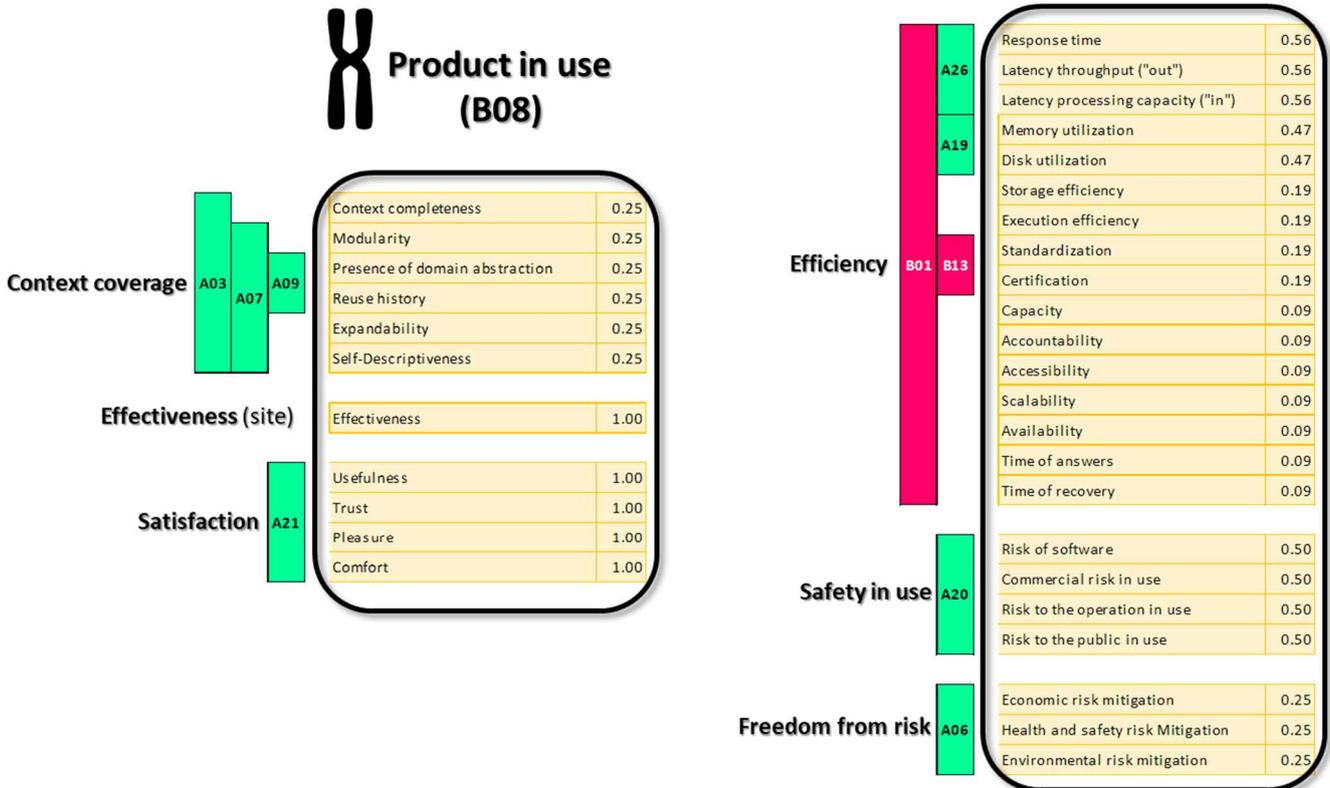


Figure 97 - "Product in use" chromatid quality characteristic sequence with its 11 genes and the 34 sites, including sites likelihood

g. “Software Product” Chromatid (B12)

Software quality degree to which a software product satisfies stated and implied needs when used under specified conditions (software quality has the same meaning as software product quality; (source: ISO/IEC 25010 [23])).

This chromatid is composed of 32 quality characteristic genes linked together as depicted by Figure 98, subset of the graph shown in Figure 84, and its complete “DNA” sequence made of 195 quality characteristic sites disclosed in Figure 99 and Figure 100.

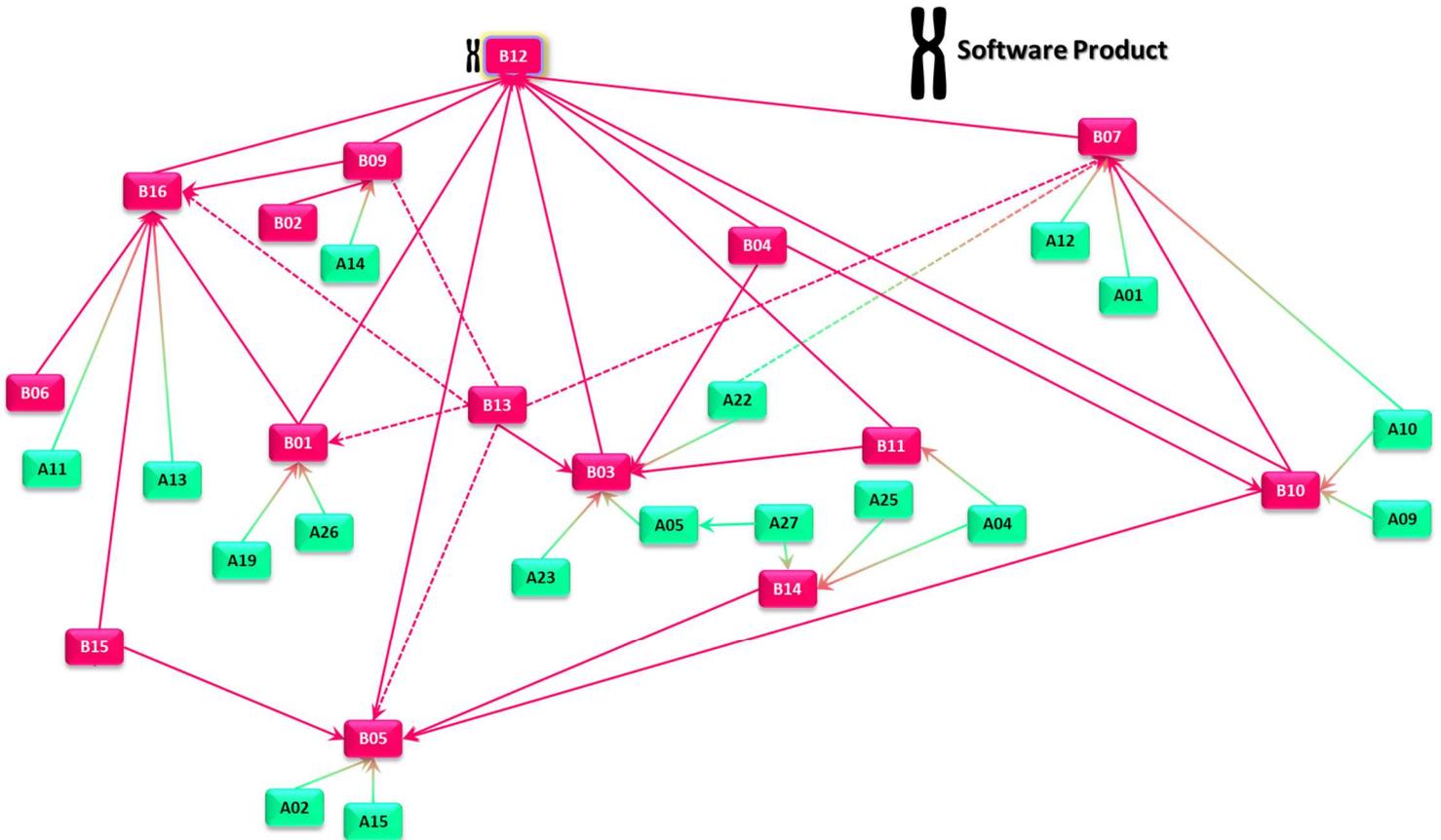


Figure 98 - The 32 genes of “Software product” chromatid, with their respective links

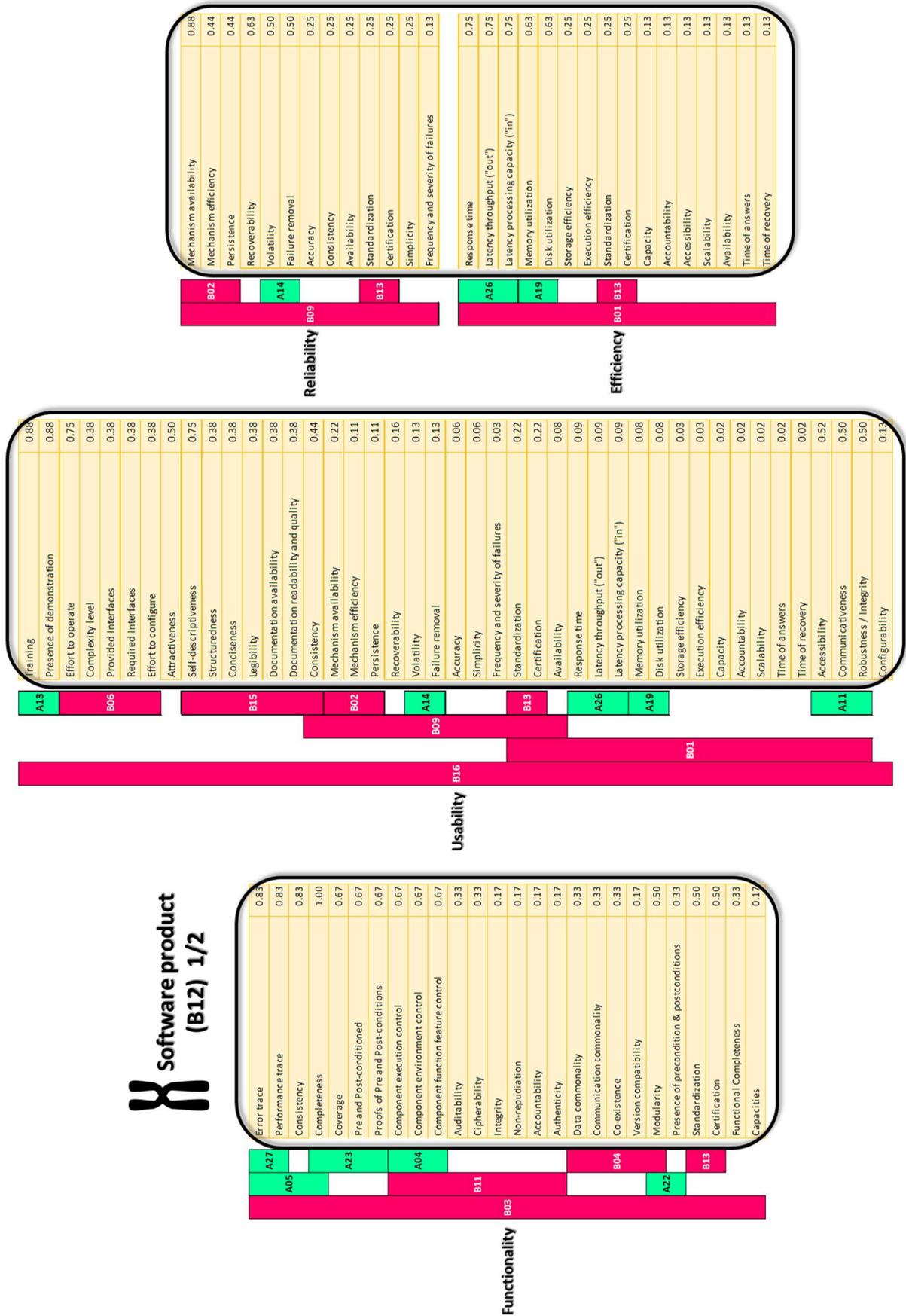


Figure 99 - "Software product" chromatid quality characteristic sequence with its 32 genes and the 195 sites, including sites likelihood (part 1 of 2)

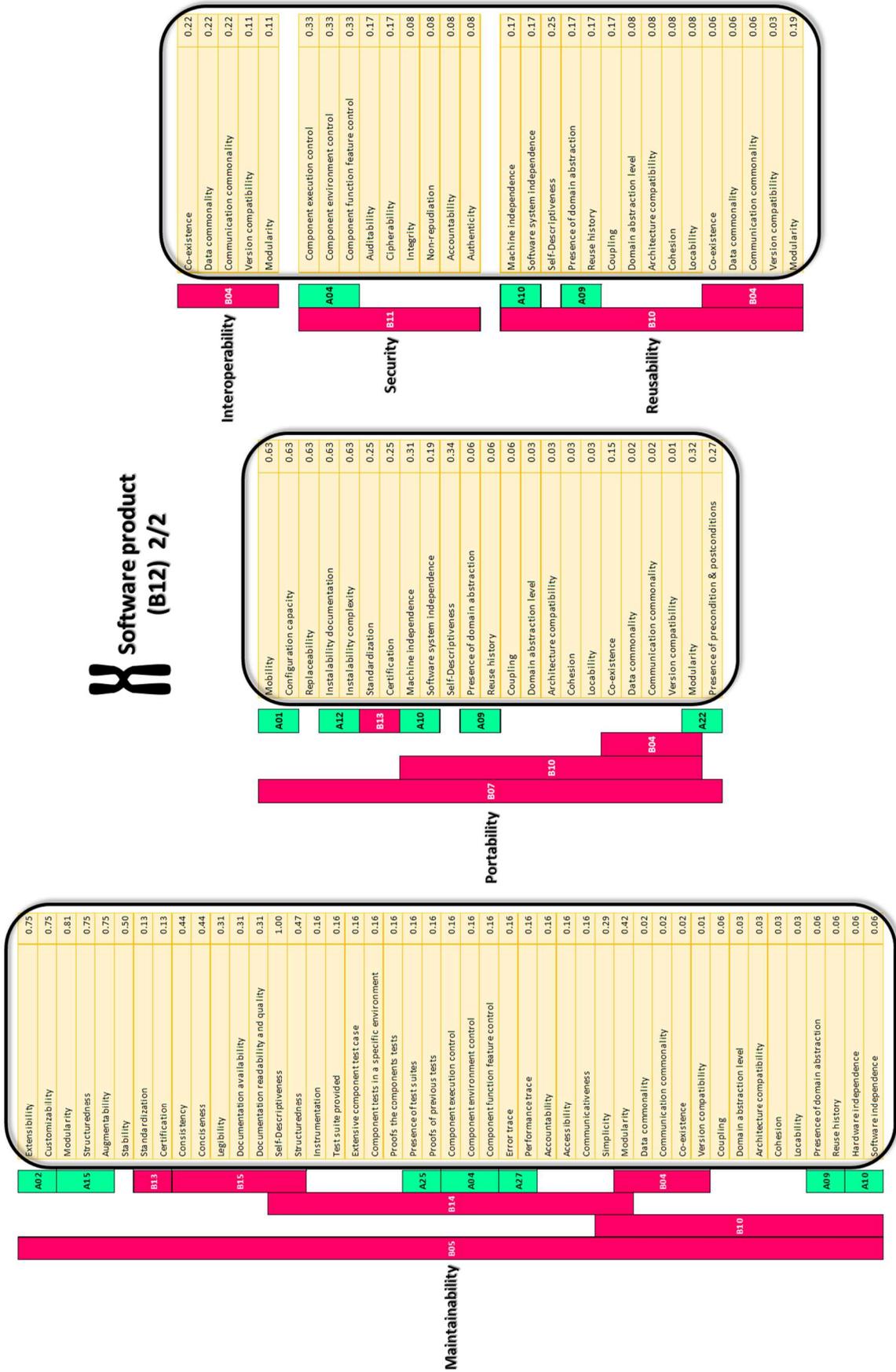


Figure 100 - "Software product" chromatid quality characteristic sequence with its 32 genes and the 195 sites, including sites likelihood (part 2 of 2)

6. Threats to validity and discussions

Through this eighth chapter, we detail the creation of a meta-model resulting from the aggregation of eight existing software quality models. This aggregation is not a direct merge of these quality models but rather uses an analogy with genetic to take into account not only the likelihood for each quality characteristics and sub-characteristics to happen or exists in a particular quality perspective, but also to consider that a quality characteristic, or sub-characteristic, may have some variations in their definition and naming, with a certain level of probability too. The contributions are an analogy definition between quality modeling and genetic, a construction algorithm and the first meta-model of software quality model genome made of seven quality chromatids with their quality characteristic sequences, including their variations defined in Annex 14, and their likelihoods to happen in each chromatid genes.

The overview shown in Figure 101 points out the complexity and length of each of these seven chromatids, as well as the existence of two polymorphic chromatids: production revision and supportability.

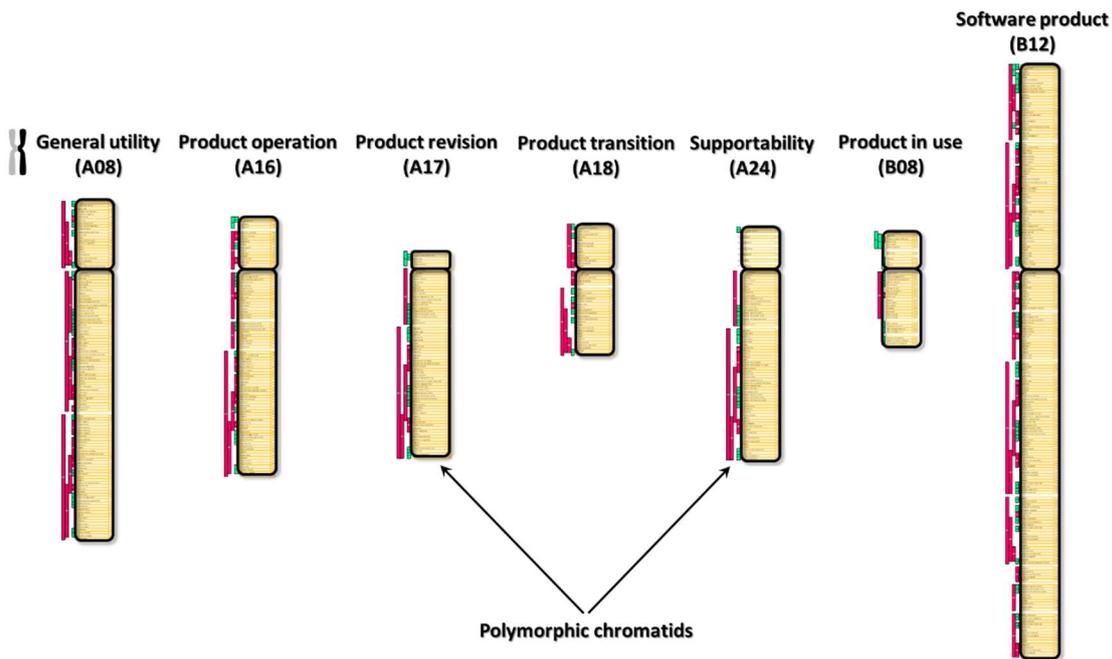


Figure 101 - Overview of the software quality model genome meta-model composed of 7 chromatids

Moreover, during the meta-model elaboration, we noticed a certain level of subjectivity in grouping together quality characteristics, or sub-characteristics, considering that they are potentially variations from each other. Indeed, if they are similar (i.e., their names are identical, direct synonym, or have an identical definition) or are distinct (i.e., their names are different, not synonym, and have disjoint definition), the grouping decision is thus obvious and objective. Otherwise, the decision may be more or less subjective. This subjectivity impacts the confidence of our meta-model, and therefore, to estimate the degree of subjectivity of the meta-model, we calculate the ratio between the characteristic and sub-characteristic associations we have made subjectively over the total number of characteristics and sub-characteristics. We find that the overall degree of subjectivity of the meta-model is 20.65%, with supportability chromatid as the least subjective (i.e., 5.56%) and product transition as the most subjective one (i.e., 34.88%). The calculation results are given in TABLE 38.

TABLE 38 - DEGREE OF SUBJECTIVITY FOR EACH CHROMATID OF THE SOFTWARE QUALITY MODEL GENOME META-MODEL

Chromatid	Number of subjective sites	Number of sites	Percentage of subjectivity
General utility	23	114	20.18%
Product operation	23	85	27.06%
Product revision	4	69	5.80%
Product transition	15	43	34.88%
Supportability	4	72	5.56%
Product in use	6	34	17.65%
Software product	52	195	26.67%
Overall	127	615	20.65%

To add further quality characteristic nuances, include more valuable quality modeling contributions, and thus consolidate these meta-model results, we must continue to integrate more software quality models, extending the diversity scope to other software quality models (e.g., web-site [187], [258], web-services [12], COTS [201], [259], open-source [189], [207], [260], IT software [191]). Note, the quality models that are candidate to be integrated must have their definition with a proper completeness level. We aim “Y”, “Y+” and possible “P+” levels, if we use Oriol *et al.* classification definition completeness level [12]:

- **Y (Yes):** *The (sub)characteristic is explicitly defined in the quality model.*
- **Y+ (Yes+):** *The (sub)characteristic is explicitly defined in the quality model and contains further subdivisions.*
- **P (Partially):** *The (sub)characteristic is not explicitly defined, but the quality model has a quality attribute or metric which can be classified into this (sub)characteristic.*
- **P+ (Partially+):** *The (sub)characteristic is not explicitly defined, but the quality model has several quality attributes or metrics which can be classified into this (sub)characteristic.*
- **ND (Not Defined):** *The (sub)characteristic is not defined, neither its quality attributes nor metrics.”*

Regarding use of this meta-model, it was not the purpose of this chapter. However, the meta-model can be used either as the input reference quality model of the 6-stages process for quality model development (see Chapter VI.4.b), with quality characteristics and sub-characteristics likelihood emphasizing the most important ones, or as is, with the likelihood values as weight factors, or again to develop a customized software quality model jointly, or not, with GQM paradigm [28] for example.

Another kind of usage is the detection of characteristics that prevail to a specific domain, like Gordieiev *et al.* [191] who were looking for the most important quality characteristics for IT-oriented software quality models based on expert review and assessment. The analogy with genetic offers an alternate approach to resolve this problematic. Indeed, the identification of prevailing quality characteristics for a specific domain can be solved similarly to the genetic disease identification thanks to non-parametric linkage technic [255]: genes at specific set of locus are looked for in a sample set, combining both healthy and sick samples; if a correlation is found between a set of those locus and the disease under investigation, then the disease is a genetic one and is associated to this specific set of locus. Nevertheless, a study must be performed to confirm the benefit of this genetic approach on sample sets, for instance, for web-services [12] and open-source [189], [207], [260] software quality models.

Finally, through a unique and innovative approach relying on an analogy with genetic, we successfully construct a meta-model against the software engineering domain. Nevertheless, there is no restriction to apply such construction of quality model genome meta-model to any types of domain (e.g., architecture, socio-economic, systems engineering) which confirms that this achievement is another original contribution to qualimetry, the science of quality quantification.

Chapter IX. General Synthesis and Research Perspectives

The objective of the thesis conducted us to define a theoretical framework for the supervision and piloting of engineering processes and product development through quality.

This work has led to multiple contributions, partially valued in the form of two international conference papers (cf. Argotti *et al.* [167], [230]). Above all, it was highlighted the need to have an exhaustive and in-depth study of existing quality models in the literature in order to be able to go further in consolidated conceptual and methodological proposals, and before being able to consider any application.

The purpose of this final thesis chapter is to conclude the three research work years synthesized in this document. Consequently, in the next two sections, we are successively reviewing a synthesis of the research path together with the related contributions, and then explore the resulting main research perspectives.

1. General synthesis

In Chapter I, we posed the thesis research problematic, “*Study of Qualimetry essentials applied to embedded software development*”, that we rephrased into “*strengthen and unify the definition, assessment, control, or prediction of the embedded software quality*”, with regard to our industrial context. We initiated its analysis, identifying the four following research questions:

Research Question 1	Is Qualimetry, as the science of quality quantification, the right approach and what are quality and Qualimetry essentials?
Research Question 2	Considering the set of software quality models, how to identify and decide which quality model is the most suitable for embedded software?
Research Question 3	Considering a quality model for a software product, how to operationalize it?
Research Question 4	Can we have a unique reference quality model for software product?

We deepened then this analysis through Chapter II. We remarked that the automotive industrial context jointly with the vehicle as complex system, the development model with suppliers, and the current standard and regulation requirements are raising the overall complexity of our problematic. Therefore, in this context it is critical to have a unified, operational, and appropriate way to define, assess, control, or predict quality of embedded software.

In order to verify if such unified, operational and appropriate solution for quality of embedded software already existed, we performed an exploratory literature review about “*how quality modeling is applied to embedded software*”. We found the existence of a myriad of possible embedded systems and software, each of them with their own specificities, quality characteristics, and possibly a diversity of quality models. So, it appeared that there was no right and unique solution yet to our question.

Consequently, we refined these four research questions into 15 research sub-questions and then detailed our research methodology in Chapter III. In this chapter, furthermore, we explained our research methodology realignment highlighting not only the difficulties in the selection of a proper quality model for embedded software, but also the consequences of such selection in discarding many valuable contributions.

Afterwards, we addressed these research questions from Chapter IV to Chapter VIII.

Research Question 1	Is Qualimetry, as the science of quality quantification, the right approach and what are quality and Qualimetry essentials?
----------------------------	---

Thus, through Chapter IV, we explored the essence of quality (i.e. research sub-question **1a**), and quality modeling particularly in software field (i.e., research sub-questions **1b**). We defined, clarified these knowledges and the related concepts (e.g., perceived quality, quality perspectives, quality dimensions and characteristics, quality model, measurements, scale), and concluded this exploration with the build of the first timeline of the key contributions to software quality modeling, going from 1965 with the first appearance of software engineering

concept [32] to 2015 with Azgaldov *et al.* and the ABC of Qualimetry [113]. Next, we looked for Qualimetry, acknowledging that we had the correct comprehension of the young science of quality quantification, before demonstrating that the right approach for our needs is Qualimetry (i.e., research sub-question **1c**).

During this investigation, we noted that Qualimetry was often misunderstood. Therefore, we start to contribute to that science by popularizing it, summarizing its major concepts under a synthetic view: the “*House of Qualimetry*” and its 6 pillars. We also remarked that it was possible to unify diversity and time evolution in quality modeling (i.e., research sub-question **1d**) by finding our inspiration in genetic, and thus introducing the concept of polymorphism (i.e., ad hoc, universal and temporal polymorphism) in quality modeling. To complete this contribution, we proposed and proved that using a genetic diversity-based formula [86] was more appropriate for comparing quality models together (i.e., research sub-question **2c**) than Hamming’s distance, for example, and proposed a new measurement process cadenced with system and software life cycle to integrate temporal polymorphism.

Research Question 2

Considering the set of software quality models, how to identify and decide which quality model is the most suitable for embedded software?

The focus of Chapter V was to deep dive in the literature to retrieve existing software quality models, and then determine which quality model could be selected to serve our needs with embedded software (i.e., research question **2**). To carry out this undertaking, we conducted a systematic literature review where we identified and analyzed 136 study papers published during a period from 1979 to 2019. The result of this review, combined with snowballing approach, as described by Wohlin [215], [216] and which consists in exploiting each referenced papers as additional source of study papers, was the retrieval of 492 software quality models from 1968 to 2019 (i.e., research sub-question **2a**). This software quality model list is a unique contribution since it represents a collection of 10 times the maximum we found in published papers: Oriol *et al.* [12] enumerated 48 quality models linked to web-services. Note, in Kläs *et al.* [97], the authors claimed that they have provided a classification for about 80 quality models, nevertheless we failed to retrieve that list of quality models, even in the referenced papers of that study or in the authors publication.

Strong of the systematic literature review results and speaking about classification, our next contribution was to propose the usage of cladistic as classification method for the software quality models (i.e., research sub-question **2b**). For that reason, the classification scheme was made of 20 software quality models classification elements organized over five themes (i.e., id, bibliographic, definition, scope and structural), and then declined into software quality model clades: homology-based (i.e., similarity related to shared ancestry) and taxa (i.e., conceptual entities).

Although we started to use a subset of these taxa to classify these 492 quality models, they were enough to succeed on depicting a software quality model landscape. We found that these models were designed principally for quality assessment and then for prediction, they are usually hierarchical except for prediction, where statistic or implicit formalism is better adapted, with a scope often put on product, and a quality perspective equally distributed over manufacturer, user, and product perspectives. In addition, our contribution on software quality model landscape rectified Thapar *et al.* [11] postulate about quality model evolution (i.e., basic quality models before 2000 and tailored quality models since 2000). Indeed, we showed that this evolution is articulated around three periods: up to 1990, we have the basic quality model period, from 1990 to 2003, the transition period, and since 2003, we are in the quality model tailoring period.

At last, the conclusion of this chapter confirmed the inadequacy to have a unique reference quality model covering all software product cases (i.e., research sub-question **4a**), and suggested the selection, as well the customization, of the latest quality model standard, ISO / IEC / IEEE 25010, to generate an appropriate model for embedded software in automotive domain (i.e., research sub-question **2d**)

Research Question 3

Considering a quality model for a software product, how to operationalize it?

In Chapter VI, our aim was to investigate the transition from quality model theory to practice, and more particularly about the quality model operationalization (i.e., research question **3**). This operational aspect is

critical to develop and deploy quality model against real word use case, or to succeed in replicating and benefiting of quality model studies.

So, during our study, we identified a list of 16 distinct challenges or issues that prevent development and use of software quality models (i.e., research sub-question **3a**), and we succeeded afterwards to associate practical solutions (i.e., solutions related to experiences, real situations or actions that are possible to reproduce, reuse or deploy) to each of these 16 challenges (i.e., research sub-question **3b**).

The consolidated synthesis of these issue identification and resolution was achieved through the proposal of two complementary processes (i.e., research sub-question **3c**):

- The “6 stages” process focuses on quality model operational development, with an analysis algorithm based on survey, Fleiss and Cohen Kappa ; this algorithm, used for quality model construction, takes into account constraint, stakeholder point of view and allows to determine automatically the quality characteristics and sub-characteristics weight factors.
- The “Quality Thermometer” process focuses on quality model operational use; therefore, it includes the “6-stages” process since one of the early stages of quality model usage concerns the quality model development.

The innovative parts of these two process contributions are the transparent encapsulation of the practical solutions and the use of polymorphism concept.

Next to the thinking and proposals about theory to practice transition, Chapter VII reflected the put into practice of our findings and contributions against our real-world use case: embedded software for the automotive industry (i.e., research sub-question **3d**).

Thus, we decided to apply them against a subset of the vehicle embedded software (i.e., three electronic control units – IVI, IVC, ADAS- with their own embedded software and a transversal embedded software functionality – FOTA). The result was the creation of three distinct polymorphic quality models with their respective weight factors. We noticed in the result the existence of two levels of polymorphic quality model inheritance, and a joint quality model for ADAS and FOTA. Furthermore, all the construction steps, detailed in this chapter, can serve as proofed guidelines to perform quality modeling against any software or systems.

Finally, this success on quality model operational development for a real word use case from automotive domain, not only allowed us to answer to the company needs, but also demonstrated the merits and relevance of our findings and contributions.

Research Question 4

Can we have a unique reference quality model for software product?

As a subsidiary chapter, since we already answered to the company demand, Chapter VIII objective was to go one step further in the exploration of a reference software quality model. Indeed, Chapter V concludes on the inadequacy to have a unique reference quality model covering all software product cases and accordingly we could elaborate rather a quality meta-model, gathering knowledge from existing quality models, which could be used a basis for developing new quality model.

Continuing with genetic, we found that a certain level of analogy could be achieved between DNA sequences and quality characteristics and sub-characteristics sequence. Moreover, likes in DNA sequences, variation of quality characteristics and sub-characteristics may exist with a certain level of likelihood, recalling the polymorphism concept. So, the basis of our quality meta-model contribution relied on this analogy which was also captured into a meta-model ontology.

After the detailed design of the meta-model construction algorithm (i.e., research sub-question **4b**), we selected a set of eight existing software quality models to initiate the creation of the meta-model first version (i.e., research sub-question **4c**). The result of this unique and final contribution is the software quality genome composed of 7 chromatids: general utility, product operation, product revision, product transition, supportability, product in use and software product.

The thesis research work and achievements are summarized also in the global synthesis done in Figure 102.

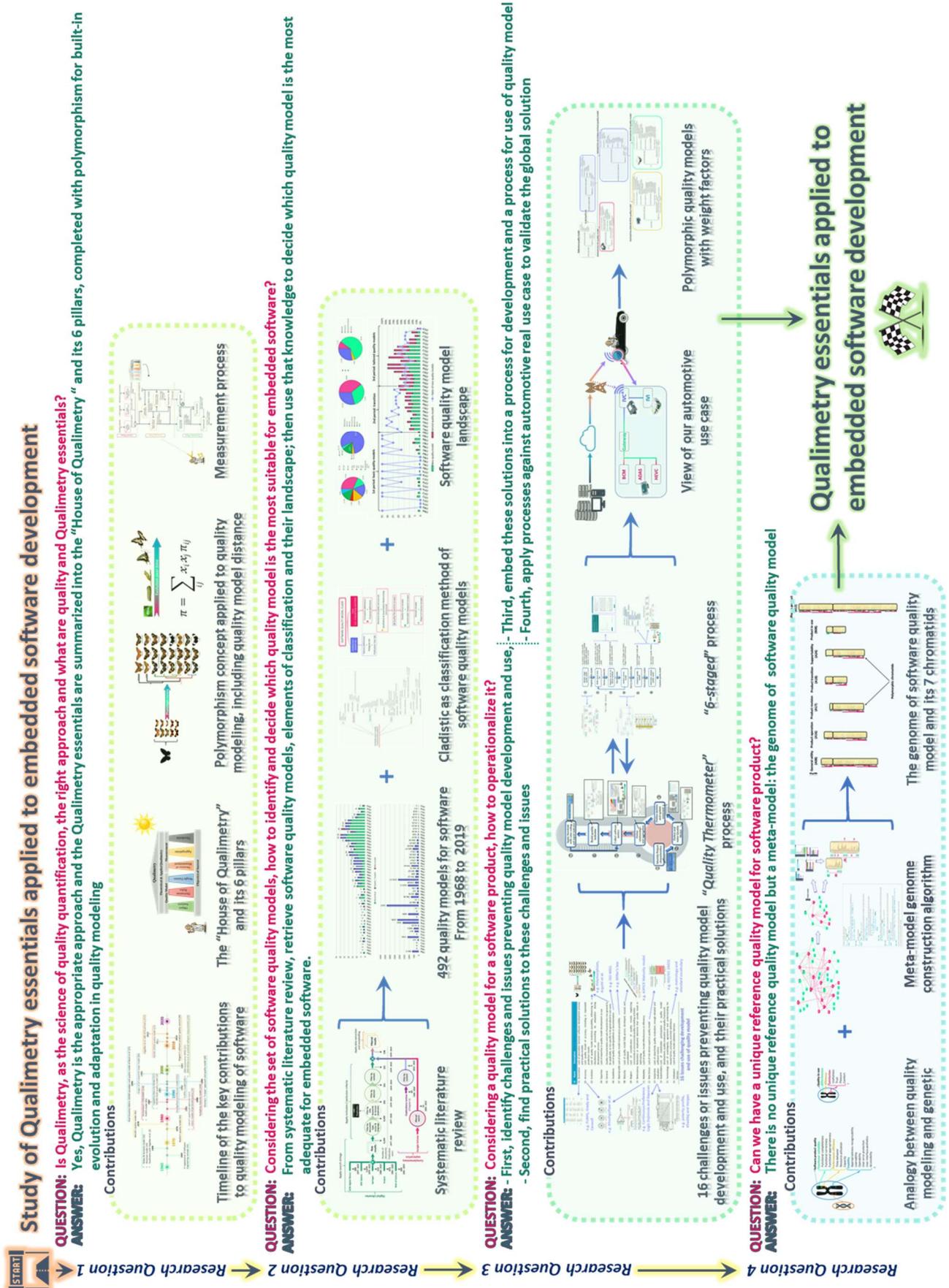


Figure 102 - General synthesis of the thesis research work and achievements

2. Research Perspectives

This comprehensive and in-depth research study on software quality models is the beginning of an exciting but hectic journey in the field of Qualimetry. The resulting research perspectives shed light on the first directions this journey should take. There are three of them, namely: valorization, consolidation, and exploration.

- **Valorization perspectives:** the intend behind these perspectives is not only to share more widely our research findings and contributions with the academic and industrial community, but also to improve our proposal through feedback and measurement of effectiveness, as well as to promote its appropriation and adaptation, through a strategy of deployment in company accompanied by training and tooling. The timeframe for this kind of perspective is mainly between short-term and mid-term period.

Thus, one way to achieve the information sharing is to rely on literature media. We planned to gather the following finding and contribution into several research paper submissions:

- o Systematic literature review with snowballing resulting in a unique list of 492 software quality model,
- o Software quality model classification based on cladistic,
- o Software quality model landscape and correction of Thapar et al.'s postulate,
- o From theory to practice analysis resulting in 16 operationalization challenges and their practical solutions,
- o The processes to develop and use quality model, encapsulating practical solutions, and exemplified against an example from automotive,
- o Polymorphism quality model in practice,
- o Software quality model genome meta-model, including the construction algorithm, and the 7 chromatids of the first meta-model result.

Furthermore, and as we already raised in Chapter V.5, an online portal tool must be created to spread the sharing of the 492 software quality model collection and enable the collaboration for their use, completion and maintenance. The motivation behind is to allow the academic and industrial community to collaborate on this collection and avoid that this list becomes obsolete within the coming years.

An alternate way to enhance the value of our research results is to industrialize, scale and deploy against actual production systems the thesis contributions. However, if we measure the technology maturity of our thesis achievements using the Technology Readiness Level (TRL) [261] scale, we are currently reaching level 4, that is to say technology has been validated in laboratory environment, while industrialization, scaling and deployment mean a level of 9. Hopefully, we can use the TRL to guide us on the path of the technology readiness.

- **Consolidation perspectives:** this second kind of research perspectives lead us to continue the consolidation of our current findings and contributions both from a research and development point of views. The timeframe for this type of research perspective is at least mid-term. The following paragraphs briefly describe the primary research and development directions.

Regarding the quality model collection and classification, the consolidation means that we have to create the right tool and data model to gather, store and classify properly at minimum the 492 found quality models. Then, we should be able to perform full classification based on cladistics and takes all benefits from these related contributions.

On the meta-model side, that research perspective indicates that we must first complete the implementation of a tool for the automatic meta-model construction, and next, integrate more quality models in the meta-model during its construction. One of the expected results linked to more quality model inclusions in the meta-model is to strengthen the convergence of the most relevant and important quality characteristics and sub-characteristics. A further meta-model enhancement will be the integration of metrics.

Concerning quality model development consolidation perspective, we aim to use the meta-model as the referenced quality model, foster quality model reuse through polymorphism, and build a tool to automate the "6-stages" process for quality model development, including metrics consideration. In parallel, the practical quality modeling for an entire complex system such as an entire vehicle should be handled.

Similarly, to consolidate quality model usage with the “*Quality Thermometer*” process, the tooling aspect must be addressed and should cover at least: the transparent use of polymorphic quality model, automated operational deployment and execution of quality models and their metrics, online dashboard and scorecard with data mining capability to enable prediction and prescription.

- **Exploration perspectives:** the objectives of such research perspectives are to explore, and study open problems related to Qualimetry, quality model or modeling. Consequently, the corresponding timeframe basis for that research work is long-term. A certain number of open problems has been already captured.

The first problem is about the assessing, or predicting, the value brought by quality model development and use. This is a recurrent question often coming from company leaders to accept Qualimetry activity cost. However, this question remains unanswered despite few research studies such as Khoshgoftaar *et al.* [173] whose initiate the construction of quality modeling activity cost-benefit model based on the assumption of quality model reuse over multiple software releases, Porta [262] with a survey on cost-benefit -analysis model for quality assurance, or the integral quality composed of quality and cost effectiveness (see Chapter IV.2.a).

A second type of problem is the formal definition and generalization of thresholds to assess, control or predict objectively that a quality level of a product, for instance, is good. Unfortunately, we usually have neither universal (i.e., commonly agreed) acceptance, reference, nor target threshold. One way to go around that problem is to define a target, or an acceptance threshold based on previous results obtained from identical product, like a previous software release. Thus, we remove the problem by considering only the progress compare to previous achievements. Nevertheless, the original problem remains intact even if there are few minor industrial attempts like the 15 acceptance ranges from the automotive HIS source code metrics [263].

The third category of problem relates to the modeling generalization of quality trajectory and its velocity. One parallel problem is about the discontinuity that may exist between assessment and prediction model with identical scope and quality perspective but with distinct model formalisms.

Finally, the last set of problem encompasses modeling of contemporary quality area of interest and where some research studies have been initiated but not yet solved the problem. We can cite for example:

- Software greenness and sustainability,
- Software aging and obsolescence,
- Quality data for connected software systems and services.

References

- [1] “Enquête Nationale : les Coûts de la Non-Qualité dans l’Industrie.” Afnor Group (in French), Oct. 2017. [Online]. Available: <http://auditeur.afnor.org/wp-content/uploads/2018/02/Enqu%C3%AAtte-nationale-les-co%C3%BBts-de-la-non-qualit%C3%A9-dans-lindustrie.pdf>
- [2] G. G. Azgaldov *et al.*, “Qualimetry: the Science of Product Quality Assessment,” *Standart y i kachest vo*, no. 1, 1968.
- [3] “ISO 26262-6:2011 - Road vehicles - Functional safety - Part 6: Product development at the software level,” *International Organization for Standardization*, 2011.
- [4] “ARP4754A - Guidelines for Development of Civil Aircraft and Systems,” *SAE International*, Dec. 2010, [Online]. Available: <https://www.sae.org/standards/content/arp4754a/>
- [5] “DO-178C - Software Considerations in Airborne Systems and Equipment Certification,” *Radio Technical Commission for Aeronautics*, Dec. 2011, [Online]. Available: https://my.rtca.org/NC__Product?id=a1B36000001lcmqEAC
- [6] Zouheyr Tamrabet, Toufik Marir, and Farid MOKHATI, “A Survey on Quality Attributes and Quality Models for Embedded Software,” *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 9, no. 2, pp. 1–17, 2018, doi: 10.4018/IJERTCS.2018070101.
- [7] P. Rioux, “Le CNRS et Lacroix créent à Toulouse un laboratoire dédié aux systèmes pyrotechniques,” *La Dépêche du Midi*, Nov. 2016, [Online]. Available: <https://www.ladepeche.fr/article/2016/11/22/2463658-cnrs-lacroix-creent-toulouse-laboratoire-dedie-systemes-pyrotechniques.html>
- [8] “LAAS - Laboratory presentation.” <https://www.laas.fr/public/en/laboratory-presentation> (accessed Oct. 15, 2020).
- [9] R. E. Fairley, *Systems Engineering of Software-Enabled Systems*, IEEE Press. John Wiley & Sons, 2019.
- [10] M. Kläs, J. Heidrich, J. Münch, and A. Trendowicz, “CQML Scheme: A Classification Scheme for Comprehensive Quality Model Landscapes,” in *2009 35th Euromicro Conference on Software Engineering and Advanced Applications*, Aug. 2009, pp. 243–250. doi: 10.1109/SEAA.2009.88.
- [11] S. S. Thapar, P. Singh, and S. Rani, “Challenges to the Development of Standard Software Quality Model,” *International Journal of Computer Applications*, vol. 49, no. 10, Jul. 2012.
- [12] M. Oriol, J. Marco, and X. Franch, “Quality Models for Web Services: A Systematic Mapping,” *Information and Software Technology*, vol. 56, no. 10, pp. 1167–1182, Oct. 2014, doi: 10.1016/j.infsof.2014.03.012.
- [13] D. D. Walden, G. J. Roedler, K. J. Forsberg, D. R. Hamelin, and T. M. Shortell, *Systems Engineering Handbook: a Guide for System Life Cycle Processes and Activities*, Fourth. Wiley, 2015.
- [14] United States Department of Transportation, “Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT) - version 8.3 of the National ITS Reference Architecture,” Oct. 14, 2019. <http://local.iteris.com/arc-it/index.html> (accessed Nov. 02, 2020).
- [15] D. Crolla, D. E. Foster, T. Kobayashi, and N. Vaughan, “Encyclopedia of Automotive Engineering , Chapter14.” p. 4101, Feb. 2015.
- [16] “ISO/PAS 21448:2019 Road vehicles - Safety of the intended functionality,” *International Organization for Standardization*, Jan. 2019, [Online]. Available: <https://www.iso.org/standard/70939.html>
- [17] “ISO/SAE DIS 21434 Road vehicles - Cybersecurity engineering (under development),” *International Organization for Standardization*, 2020, [Online]. Available: <https://www.iso.org/standard/70918.html>
- [18] L. Caudet, V. Von Hammerstein-Gesmond, and M. Noyon, “Agreement on Commission proposal to tighten rules for safer and cleaner cars,” *European Commission*, Dec. 2017, [Online]. Available: https://ec.europa.eu/commission/presscorner/detail/en/IP_17_5131
- [19] European Parliament, “Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation),” Apr. 27, 2016. <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679>

- [20] C. Perez, *Technological Revolutions and Financial Capital: The Dynamics of Bubbles and Golden Ages*. Edward Elgar Publishing Ltd, 2003.
- [21] VDA QMC Working Group 13 / Automotive SIG, "Automotive SPICE Process Assessment / Reference Model, version 3.1 - revision 656." Nov. 01, 2017. [Online]. Available: http://www.automotivespice.com/fileadmin/software-download/AutomotiveSPICE_PAM_31.pdf
- [22] K. Forsberg and H. Mooz, "The Relationship of Systems Engineering to the Project Cycle," *First Annual Symposium of the National Council On Systems Engineering (NCOSE)*, Oct. 1991.
- [23] "ISO/IEC 25010:2011 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," *International Organization for Standardization*, 2011, [Online]. Available: <https://www.iso.org/standard/35733.html>
- [24] "ISO/IEC 9126-1:2001 - Software engineering - Product quality - Part1: Quality Model," *International Organization for Standardization*, 2001, [Online]. Available: <https://www.iso.org/standard/22749.html>
- [25] "ISO/CEI 25000:2005 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Planning and management," *International Organization for Standardization*, 2005, [Online]. Available: <https://www.iso.org/standard/35683.html>
- [26] B. Kitchenham and S. Pfleeger, "Software quality: the elusive target," *IEEE Software*, vol. 13, no. 1, pp. 12–21, 1996.
- [27] S. Wagner, *Software Product Quality Control*, Springer-Verlag Berlin Heidelberg. 2013.
- [28] V. Basili, G. Caldiera, and H. D. Rombach, "Goal Question Metric Approach," *Encyclopedia of Software Engineering*, John Wiley & Sons, Inc., pp. 528–532, 1994.
- [29] H. Y. Jeong and Y. H. Kim, "A Quality Model of Lightweight Component for Embedded System," *Applied Mechanics and Materials*, vol. 121–126, pp. 4907–4911, 2012, doi: 10.4028/www.scientific.net/AMM.121-126.4907.
- [30] J. Adams, H. T. Khan, R. Raeside, and D. I. White, *Research methods for graduate business and social science students*. SAGE publications India, 2007. [Online]. Available: <http://lib.mitc.edu.vn/bitstream/123456789/13168/1/7.pdf>
- [31] B. Kitchenham and S. Charters, *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007.
- [32] E. C. Berkeley, M. M. Berlin, L. L. Lovett, and N. D. MacDonald, Eds., "The Computer Directory and Buyer's Guide," in *Computers and Automation*, Jun. 1965, vol. 14. [Online]. Available: <http://bitsavers.trailing-edge.com/pdf/computersAndAutomation/196506.pdf>
- [33] R. J. Rubey and R. D. Hartwick, "Quantitative measurement of program quality," in *Proceedings of the 1968 23rd ACM national conference (ACM '68)*, New York, NY, USA, 1968, pp. 671–677. doi: <http://dx.doi.org/10.1145/800186.810631>.
- [34] M. L. Shooman, *Probabilistic reliability: an engineering approach*. New York, N.Y.: McGraw-Hill, 1968.
- [35] J. Lepistö, "Embedded Software Testing Methods," Bachelor of Engineering Thesis, Helsinki Metropolia University of Applied Sciences, 2012. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/46873/Lepisto_Juho.pdf?sequence=1
- [36] A. Alvaro, E. S. De Almeida, and S. L. Meira, "A Software Component Quality Model: A Preliminary Evaluation," in *32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06)*, Sep. 2006, pp. 28–37. doi: 10.1109/EUROMICRO.2006.13.
- [37] F. Carvalho and S. Meira, "Towards an Embedded Software Component Quality Verification Framework," in *2009 14th IEEE International Conference on Engineering of Complex Computer Systems*, Jun. 2009, pp. 248–257. doi: 10.1109/ICECCS.2009.26.
- [38] M. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, "Capability Maturity Model for Software v1.1," Software Engineering Institute, Pittsburgh, Pennsylvania, CMU/SEI-93-TR-24, 1993.
- [39] Y. Choi, S. Lee, H. Song, J. Park, and S. Kim, "Practical S/W component quality evaluation model," 2008, vol. 1, pp. 259–264. doi: 10.1109/ICACT.2008.4493757.

- [40] IEEE Std 1061-1998, "textitIEEE Standard for a Software Quality Metrics Methodology, R2009, Revision of IEEE Std 1061." 1992.
- [41] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality," *Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command*, 1977.
- [42] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative Evaluation of Software Quality," in *Proceedings of the 2nd International Conference on Software Engineering (ICSE '76)*, Los Alamitos, CA, USA, 1976, pp. 592–605. [Online]. Available: <http://dl.acm.org/citation.cfm?id=800253.807736>
- [43] C. Peper and D. Schneider, "On runtime service quality models in adaptive ad-hoc systems," in *Proceedings of the 2009 ESEC/FSE workshop on Software integration and evolution @ runtime*, Amsterdam, The Netherlands, 2009, pp. 11–18. [Online]. Available: <https://doi.org/10.1145/1596495.1596500>
- [44] W. H. DeLone and E. R. McLean, "Information systems success: The quest for the dependent variable," *Information systems research*, vol. 3, no. 1, pp. 60–95, 1992.
- [45] H. Y. Jeong and Y. H. Kim, "A Design of Software Quality Model for Embedded System," *Applied Mechanics and Materials*, vol. 157–158, pp. 680–683, 2012, doi: 10.4028/www.scientific.net/AMM.157-158.680.
- [46] H.-Y. Jeong and Y.-H. Kim, "A system software quality model using DeLone & McLean model and ISO/IEC 9126," *International Journal of Digital Content Technology and its Applications*, vol. 6, no. 5, pp. 181–188, 2012.
- [47] H. Jeong, "The Practical Quality Model for Embedded System and Software," in *2013 16th International Conference on Network-Based Information Systems*, Sep. 2013, pp. 288–291. doi: 10.1109/NBiS.2013.44.
- [48] A. Mayr, R. Plösch, M. Kläs, C. Lampasona, and M. Saft, "A Comprehensive Code-Based Quality Model for Embedded Systems: Systematic Development and Validation by Industrial Projects," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering*, Nov. 2012, pp. 281–290. doi: 10.1109/ISSRE.2012.4.
- [49] G. Dromey, "A Model for Software Product Quality," *IEEE Transactions on Software Engineering*, vol. 146, no. 21, 1995.
- [50] A. Trendowicz and T. Punter, "Quality Modeling for Software Product Lines," 2003.
- [51] D. Ahrens, A. Frey, A. Pfeiffer, and T. Bertram, "Objective evaluation of software architectures in driver assistance systems," *Computer Science - Research and Development*, vol. 28, no. 1, pp. 23–43, Feb. 2013, doi: 10.1007/s00450-011-0185-x.
- [52] K. T. Al-Sarayreh, "A Quality Requirements Safety Model for Embedded and Real Time Software Product Quality," in *Recent Advances in Computer Science*, Kuala Lumpur, Malaysia, Apr. 2015, pp. 200–206.
- [53] N. Silva and M. Vieira, "Software for Embedded Systems: A Quality Assessment Based on Improved ODC Taxonomy," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, New York, NY, USA, 2016, pp. 1780–1783. doi: 10.1145/2851613.2851908.
- [54] R. Chillarege *et al.*, "Orthogonal defect classification—a concept for in-process measurements," *IEEE Transactions on software Engineering*, vol. 18, no. 11, pp. 943–956, Nov. 1992, doi: 10.1109/32.177364.
- [55] L. Garcés, A. Ampatzoglou, P. Avgeriou, and E. Y. Nakagaw, "Quality attributes and quality models for ambient assisted living software systems: A systematic mapping," *INFORMATION AND SOFTWARE TECHNOLOGY*, vol. 82, pp. 121–138, Feb. 2017, doi: 10.1016/j.infsof.2016.10.005.
- [56] L. Garcés, F. Oquendo, and E. Y. Nakagawa, "A Quality Model for AAL Software Systems," in *2016 IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS)*, Jun. 2016, pp. 175–180. doi: 10.1109/CBMS.2016.46.
- [57] S. Kasiviswanathan and D. Ramalingam, "Development and application of user review quality model for embedded system," *Microprocessors and Microsystems*, vol. 74, p. 103029, Apr. 2020, doi: 10.1016/j.micpro.2020.103029.
- [58] N. Kano, N. Seraku, F. Takahashi, and S. Tsuji, "Attractive Quality and Must-Be Quality [in Japanese]," *Journal of the Japanese Society for Quality Control*, vol. 14, no. 2, pp. 147–156, Apr. 1984.
- [59] R. W. Saaty, "The analytic hierarchy process—what it is and how it is used," *Mathematical Modelling*, vol. 9, no. 3, pp. 161–176, Jan. 1987, doi: 10.1016/0270-0255(87)90473-8.

- [60] Y.-M. Zhu, "Software Failure Mode and Effects Analysis," in *Failure-Modes-Based Software Reading*, Y.-M. Zhu, Ed. Cham: Springer International Publishing, 2017, pp. 7–15. doi: 10.1007/978-3-319-65103-3_2.
- [61] J. G. Wijnstra, "Quality attributes and aspects of a medical product family," *Proceedings of the Hawaii International Conference on System Sciences*, p. 284, 2001, doi: 10.1109/HICSS.2001.927254.
- [62] A. Purhonen, "Quality attribute taxonomies for DSP software architecture design," in *Lecture Notes in Computer Science*, Berlin, Apr. 2002, vol. 2290, pp. 238–247. doi: 10.1007/3-540-47833-7_21.
- [63] M. Barbacci, M. H. Klein, T. A. Longstaff, and C. B. Weinstock, "Quality Attributes," Carnegie Mellon University, Technical Report (CMU/SEI-95-TR-021), ESC-TR-95-021, 1995.
- [64] M. Akerholm, J. Fredriksson, K. Sandström, and I. Crnkovic, "Quality attribute support in a component technology for vehicular software," Linköping, Sweden, 2004, pp. 1–9.
- [65] M. Larsson, "Predicting Quality Attributes in Component-based Software Systems," PhD Thesis, Mälardalen University, 2004. [Online]. Available: <http://www.es.mdh.se/publications/575->
- [66] M. Paulitsch, H. Ruess, and M. Sorea, "Non-functional Avionics Requirements," in *Communications in Computer and Information Science*, Berlin, Heidelberg, 2008, vol. 17, pp. 369–384. doi: 10.1007/978-3-540-88479-8_26.
- [67] T. Sherman, "Quality attributes for embedded systems," 2008, pp. 536–539. doi: 10.1007/978-1-4020-8741-7_95.
- [68] M. Guessi, E. Y. Nakagawa, F. Oquendo, and J. C. Maldonado, "Architectural description of embedded systems: a systematic review," in *Proceedings of the 3rd international ACM SIGSOFT symposium on Architecting Critical Systems*, Bertinoro, Italy, 2012, pp. 31–40. [Online]. Available: <https://doi.org/10.1145/2304656.2304661>
- [69] L. Oliveira *et al.*, "An investigation on quality models and quality attributes for embedded systems," *ICSEA*, vol. 13, pp. 1–6, 2013.
- [70] H.-Y. Jeong, J. H. Park, and Y.-S. Jeong, "An ANP-Based Practical Quality Model for a Secure Embedded System with Sensor Network," *International Journal of Distributed Sensor Networks*, vol. 10, no. 2, p. 505242, Feb. 2014, doi: 10.1155/2014/505242.
- [71] T. Bianchi, D. S. Santos, and K. R. Felizardo, "Quality Attributes of Systems-of-Systems: A Systematic Literature Review," in *THIRD INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING FOR SYSTEMS-OF-SYSTEMS SESOS 2015*, 2015, pp. 23–30. doi: 10.1109/SESOS.2015.12.
- [72] T. Khoshgoftaar and E. B. Allen, "Predicting fault-prone software modules in embedded systems with classification trees," in *Proceedings 4th IEEE International Symposium on High-Assurance Systems Engineering*, Nov. 1999, pp. 105–112. doi: 10.1109/HASE.1999.809481.
- [73] T. Khoshgoftaar, B. Cukic, and N. Seliya, "Predicting Fault-Prone Modules in Embedded Systems Using Analogy-Based Classification Models," *Int. J. Soft. Eng. Knowl. Eng.*, vol. 12, no. 02, pp. 201–221, Apr. 2002, doi: 10.1142/S0218194002000883.
- [74] X. He and Y. Li, "Software reliability analysis on embedded system based on SFMEA and SFTA model," in *2012 International Conference on Systems and Informatics (ICSAI2012)*, May 2012, pp. 2471–2474. doi: 10.1109/ICSAI.2012.6223554.
- [75] J. Liu, Y. Chen, L. Zhang, J. Deng, and W. Zhang, "The Evaluation of the Embedded Software Quality Based on the Binary Code," in *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Aug. 2016, pp. 167–170. doi: 10.1109/QRS-C.2016.26.
- [76] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Transactions on Reliability*, vol. R-28, no. 3, pp. 206–211, Aug. 1979, doi: 10.1109/TR.1979.5220566.
- [77] S. Juneja, A. Juneja, and R. Anand, "Reliability Modeling for Embedded System Environment compared to available Software Reliability Growth Models," in *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, Apr. 2019, pp. 379–382. doi: 10.1109/ICACTM.2019.8776814.

- [78] K. Rohloff, J. Loyall, and R. Schantz, "Quality Measures for Embedded Systems and Their Application to Control and Certification," *SIGBED Rev.*, vol. 3, no. 4, pp. 58–62, Oct. 2006, doi: 10.1145/1183088.1183095.
- [79] M. F. S. Oliveira, R. M. Redin, L. Carro, L. d. C. Lamb, and F. R. Wagner, "Software Quality Metrics and their Impact on Embedded Software," in *2008 5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software*, Apr. 2008, pp. 68–77. doi: 10.1109/MOMPES.2008.11.
- [80] I. Stürmer and H. Pohlheim, "Model Quality Assessment in Practice: How to Measure and Assess the Quality of Software Models During the Embedded Software Development Process," Toulouse, France, Feb. 2012. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02263433>
- [81] J. Bouquet *et al.*, "Model Quality Objectives for embedded software development with MATLAB and Simulink," Toulouse, France, Jan. 2018. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02156122>
- [82] M. M. Lehman, "Programs, Life Cycles, and Laws of Software Evolution," *In Proceedings of the IEEE*, vol. 68, no. 8, pp. 1060–1076, Sep. 1980.
- [83] S. Wagner, K. Lochmann, S. Winter, A. Goeb, M. Kläs, and S. Nunnenmacher, "Software Quality Models in Practice: Survey Results," Technische Universität München Institut für Informatik, TUM-I19, 2012. [Online]. Available: <http://mediatum.ub.tum.de/doc/1110601/1110601.pdf>
- [84] G. Horgan, S. Khaddaj, and P. Forte, "An essential views model for software quality assurance," 1999, pp. 387–396.
- [85] R. B. Grady and D. L. Caswell, *Software Metrics: Establishing a Company-Wide Program*. Englewood Cliffs, New Jersey, USA: Prentice Hall, Inc., 1987.
- [86] M. Nei and W.-H. Li, "Mathematical model for studying genetic variation in terms of restriction endonucleases," in *In Proceedings of the National Academy of Science of the USA*, Oct. 1979, vol. 76, pp. 5269–5273.
- [87] R. Hamming, "Error-Detecting and Error-Correcting Codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.
- [88] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb. 1966.
- [89] F. J. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors," *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, Mar. 1964, doi: 10.1145/363958.363994.
- [90] M. A. Jaro, "Advances in record linking methodology as applied to the 1985 census of Tampa Florida," *Journal of the American Statistical Society*, vol. 84, no. 406, pp. 414–420, 1989.
- [91] W. E. Winkler, "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage," *Proceedings of the Section on Survey Research Methods, American Statistical Association*, pp. 354–359, 1990.
- [92] O. Gordieiev, V. Kharchenko, N. Fominykh, and V. Sklyar, "Evolution of Software Quality Models in Context of the Standard ISO 25010," in *Proceedings of the 9-th International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30 – July 4, 2014, Brunów, Poland, 2014*, pp. 223–232.
- [93] O. Gordieiev, V. Kharchenko, and M. Fusani, "Software Quality Standards and Models Evolution: Greenness and Reliability Issues," in *INFORMATION AND COMMUNICATION TECHNOLOGIES IN EDUCATION, RESEARCH, AND INDUSTRIAL APPLICATIONS*, HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY, 2016, vol. 594, pp. 38–55. doi: 10.1007/978-3-319-30246-1_3.
- [94] S. Motogna, D. Lupsa, and L. Ciuciu, "A NLP Approach to Software Quality Models Evaluation," Feb. 2019, vol. 11231, pp. 207–217.
- [95] P. Jaccard, "Distribution de la Flore Alpine dans le Bassin des Dranses et dans quelques Régions Voisines," *Bulletin de la Société vaudoise des sciences naturelles*, vol. 37, pp. 241–272, Jan. 1901.
- [96] A. Abran, R. Al Qutaish, J. Desharnais, and N. Habra, "ISO-based models to measure software product quality," *Institute of Chartered Financial Analysts of India (ICFAI)-ICFAI Books*, 2007.
- [97] M. Kläs, C. Lampasona, and J. Munch, "Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results," in *2011 37TH EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS (SEAA 2011)*, 2011, pp. 341–348. doi: 10.1109/SEAA.2011.62.

- [98] S. Wagner, K. Lochmann, S. Winter, A. Goeb, and M. Kläs, "Quality Models in Practice: A Preliminary Analysis," in *ESEM 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 464–467.
- [99] M. . M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski, "Metrics and Laws of software Evolution - The nineties View, journal," in *Proceedings of the 4th International Symposium on Software Metrics, IEEE*, 1997, pp. 20–32.
- [100] E. Arch, "Lehman's Laws of Software Evolution and the Staged-Model," 2011. https://blogs.msdn.microsoft.com/karchworld_identity/2011/04/01/lehmans-laws-of-software-evolution-and-the-staged-model/
- [101] D. Garvin, "What does 'product quality' really mean?," *Sloan Management Review*, vol. 26, pp. 25–45, 1984.
- [102] S. Kalaimagal and R. Srinivasan, "Q'Facto 12: an improved quality model for COTS components," *SIGSOFT Softw. Eng. Notes*, vol. 35, no. 2, pp. 1–4, 2010.
- [103] N. Bawane and C. V. Srikrishna, "A Novel Method for Quantitative Assessment of Software Quality," *International Journal of Computer Science and Security (IJCSS)*, vol. 3, no. 6, pp. 508–517, Jan. 2010.
- [104] D. A. Garvin, "Competing in the Eight Dimensions of Quality," *Harvard Business Review*, pp. 101–109, 1987.
- [105] G. Roedler and D. H. Rhodes, "Systems engineering leading indicators guide - version 1.0," *LAI, INCOSE, PSM & SEARI*, p. 67, Jun. 2007.
- [106] G. Roedler, D. H. Rhodes, H. Schimmoller, and C. Jones, "Systems engineering leading indicators guide - version 2.0," *INCOSE Technical Product Number: INCOSE-TP-2005-001-03*, p. 146, Jan. 2010.
- [107] P. Antman, "From Aristotle to Descartes a Brief history of quality," May 08, 2013. <https://blog.smartbear.com/software-quality/from-aristotle-to-descartes-a-brief-history-of-quality/>
- [108] G. P. Stavropoulos, *The Complete Aristotle*. Free GPS Library, 2013.
- [109] "ISO/IEC 9000:2015 - Quality management systems - Fundamentals and vocabulary," *International Organization for Standardization*, 2015, [Online]. Available: <https://www.iso.org/standard/45481.html>
- [110] "ISO/IEC/IEEE 24765 International Standard - Systems and software engineering--Vocabulary," *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, Aug. 2017, doi: 10.1109/IEEESTD.2017.8016712.
- [111] "ISTQB glossary 3.1," <https://www.istqb.org/downloads/category/20-istqb-glossary.html>.
- [112] "IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990." Institute of Electrical and Electronic Engineers, Inc., New York, NY, Reaffirmed on 12-9-2002, Dec. 10, 1990.
- [113] G. Azgaldov, A. Kostin, and A. Padilla Omiste, *The ABC of Qualimetry, toolkit for measuring the immeasurable*, Ridero. 2015.
- [114] A. S. Lobanov, "The Basic Concepts of Qualimetry," *Scientific and Technical Information Processing*, vol. 40, no. 2, pp. 72–82, 2013.
- [115] G. von Dran, P. Zhang, and R. Small, "Quality Websites: An Application of the Kano Model to Website Design," in *AMCIS 1999 Proceedings*. 314, 1999, pp. 898–900. [Online]. Available: <https://aisel.aisnet.org/amcis1999/314>
- [116] S.-W. Liang, H.-P. Lu, and T.-K. Kuo, "A Study on Using the Kano Two-Dimensional Quality Model to Evaluate the Service Quality of Government Websites," *JOURNAL OF INTERNET TECHNOLOGY*, vol. 15, no. 2, SI, pp. 149–162, Mar. 2014, doi: 10.6138/JIT.2014.15.2.01.
- [117] W.-M. Han, "Evaluating perceived and estimated data quality for Web 2.0 applications: a gap analysis," *SOFTWARE QUALITY JOURNAL*, vol. 26, no. 2, pp. 367–383, Jun. 2018, doi: 10.1007/s11219-017-9365-7.
- [118] "Glossary of Terms used in the Planning and Design of the IAEA Technical Cooperation Programme." International Atomic Energy Agency (IAEA), Jul. 04, 2016. [Online]. Available: https://pcmf.iaea.org/DesktopModules/PCMF/docs/2017_18_Docs/other/Planning_and_Design_Glossary_2016_07_05.pdf

References

- [119] ISO/IEC JTC 1/SC 7 Software and systems engineering, "ISO/IEC/IEEE 15939:2017 Systems and software engineering - Measurement process," *International Organization for Standardization*, May 1999, [Online]. Available: <https://www.iso.org/standard/71197.html>
- [120] ISO/IEC JTC 1/SC 7 Software and systems engineering, "ISO/IEC 14598-1:1999 Information technology - Software product evaluation - Part 1: General overview," *International Organization for Standardization*, Apr. 1999, [Online]. Available: <https://www.iso.org/standard/24902.html>
- [121] "ISO/IEC 25021:2012 - Systems and software engineering - System and software product Quality Requirements and Evaluation (SQuaRE) - Quality measure elements," *International Organization for Standardization*, 2012.
- [122] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner, "Software quality models: Purposes, usage scenarios and requirements," 2009.
- [123] "ISO/IEC 25020:2007 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Measurement reference model and guide," *International Organization for Standardization*, 2007.
- [124] T. M. Khoshgoftaar and J. C. Munson, "Predicting software development errors using software complexity metrics," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 2, pp. 253–261, Feb. 1990, doi: 10.1109/49.46879.
- [125] T. M. Khoshgoftaar and R. M. Szabo, "A Poisson Regression Model of Software Quality: A Comparative Study," in *Reliability Modeling, Analysis and Optimization*, vol. Volume 9, 0 vols., WORLD SCIENTIFIC, 2006, pp. 131–154. doi: 10.1142/9789812707147_0007.
- [126] B. Kitchenham, "Towards a constructive quality model. Part 1: Software quality modelling, measurement and prediction," *Software Engineering Journal*, vol. 2, no. 4, pp. 105-126(21), Jul. 1987.
- [127] B. Kitchenham, S. Linkman, A. Pasquini, and V. Nanni, "The SQUID approach to defining a quality model," *Software Quality Journal*, vol. 6, no. 3, pp. 211–233, Sep. 1997, doi: 10.1023/A:1018516103435.
- [128] B. W. Boehm, "Characteristics of Software Quality," *TRW Series of Software Technology*, Jan. 1978.
- [129] D. J. Hand, "Statistics and the Theory of Measurement," *Journal of the Royal Statistical Society*, no. 159, pp. 445–492, 1996.
- [130] H. von Helmholtz, *Epistemological Writings, The Paul Hertz/Moritz Schlick centenary edition of 1921, with notes and commentary by the editors, Chapter 3: Numbering and Measuring from an Epistemological Viewpoint*, vol. 79. 1977.
- [131] P. W. Bridgman, *The Logic of Modern Physics*. New York : Macmillan, 1927. [Online]. Available: <https://archive.org/details/logicofmodernphy00brid>
- [132] J. A. Diez, "A Hundred Years of Numbers. An Historical Introduction to Measurement Theory 1887-1990 Part I: The Formation Period. Two Lines of Research: Axiomatics and Real Morphisms, Scales and Invariance," *Studies in History and Philosophy of Science*, vol. 28, no. 1, pp. 167–185, 1997.
- [133] J. A. Diez, "A Hundred Years of Numbers. An Historical Introduction to Measurement Theory 1887-1990 Part II: Suppes and the Mature Theory and Uniqueness Representation," *Studies in History and Philosophy of Science*, vol. 28, no. 2, pp. 237–265, 1997.
- [134] I. BIPM, I. IFCC, I. IUPAC, and O. ISO, "The international vocabulary of metrology—basic and general concepts and associated terms (VIM), 3rd edition JCGM 200: 2012," *JCGM (Joint Committee for Guides in Metrology)*, p. 108, 2012.
- [135] N. F. Schneidewind, "Methodology for validating software metrics," *IEEE Transactions on Software Engineering*, vol. 18, no. 5, pp. 410–422, May 1992, doi: 10.1109/32.135774.
- [136] A. Abran, *Software Metrics and Software Metrology*, EEE Computer Society / Wiley Partnership. 2010. [Online]. Available: <https://onlinelibrary.wiley.com/doi/book/10.1002/9780470606834>
- [137] L. B. Mokkink *et al.*, "The COSMIN study reached international consensus on taxonomy, terminology, and definitions of measurement properties for health-related patient-reported outcomes," *Journal of Clinical Epidemiology*, vol. 63, pp. 733–745, 2010.

- [138] A. C. de Souza, N. M. C. Alexandre, and E. de Brito Guirardello, "Psychometric properties in instruments evaluation of reliability and validity," in *Epidemiol. Serv. Saude*, Brasília, Sep. 2017, vol. 26. doi: 10.5123/S1679-4974201700030002.
- [139] S. S. Stevens, "On the Theory of Scales of Measurement," *Science, New Series*, vol. 103, no. 2687, pp. 677–680, Jun. 1946.
- [140] P. Velleman and L. Wilkinson, "Nominal, Ordinal, Interval, and Ratio Typologies are Misleading," *The American Statistician*, vol. 47, no. 1, pp. 65–72, 1993.
- [141] "ISO/IEC 33020:2015 - Information technology — Process assessment — Process measurement framework for assessment of process capability," *International Organization for Standardization*, 2015, [Online]. Available: <https://www.iso.org/standard/54195.html>
- [142] N. R. Chrisman, *Rethinking levels of measurement for cartography*, vol. 25. Cartography and geographic information systems, 1998.
- [143] M. Detyniecki, "Fundamentals on Aggregation Operators," Computer Science Division University of California, Berkeley United States of America, 2001. [Online]. Available: <http://www.cs.berkeley.edu/~marcin/agop.pdf>
- [144] J. J. Dujmovic and A. Bayucan, "A Quantitative Method for Software Evaluation and its Application in Evaluating Windowed Environments," 1997.
- [145] "ISO/IEC 25022:2016 - Systems and software engineering - Systems and software product Quality Requirements and Evaluation (SQuaRE) - Measurement of internal quality," *International Organization for Standardization*, 2016.
- [146] "ISO/IEC TR 9126-4:2004 - Software engineering - Product quality - Part4: Quality in use Metrics," *International Organization for Standardization*, 2004, [Online]. Available: <https://www.iso.org/standard/39752.html>
- [147] "ISO/IEC 25040:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) -- Evaluation process," *International Organization for Standardization*, 2011.
- [148] S. Khaddaj and G. Horgan, "A Proposed Adaptable Quality Model for Software Quality Assurance," *Journal of Computer Sciences*, vol. 1, no. 4, pp. 481–486, Apr. 2005.
- [149] A. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society, London Mathematical Society*, 1936, doi: 10.1112/PLMS/S2-42.1.230.
- [150] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, and M. J. Merrit, "Characteristics of Software Quality," Document #25201-6001-RU-00, Dec. 1973.
- [151] B. W. Boehm, *Software Engineering Economics*, Prentice Hall, Inc. Englewood Cliffs, New Jersey, USA, 1981.
- [152] R. B. Grady, *Practical Software Metrics for Project Management and Process Improvement*. USA: Prentice-Hall, Inc., 1992.
- [153] W. S. Humphrey, "Characterizing the software process: a maturity framework," *IEEE Software*, vol. 5, no. 2, pp. 73–79, Mar. 1988, doi: 10.1109/52.2014.
- [154] M. Paulk, B. Curtis, and M. B. Chrissis, "Capability Maturity Model for Software v1.0," Software Engineering Institute, Pittsburgh, Pennsylvania, CMU/SEI-91-TR-24, 1991.
- [155] "ISO/IEC TR 9126-2:2003 - Software engineering - Product quality - Part2: External Metrics," *International Organization for Standardization*, 2003, [Online]. Available: <https://www.iso.org/standard/22750.html>
- [156] "ISO/IEC TR 9126-3:2003 - Software engineering - Product quality - Part3: Internal Metrics," *International Organization for Standardization*, 2003, [Online]. Available: <https://www.iso.org/standard/22891.html>
- [157] J. McGarry *et al.*, *Practical Software Measurement: Objective Information for Decision Makers*. Addison-Wesley, 2001.

References

- [158] "GSO ISO/IEC 19761:2017, Software engineering - COSMIC: a functional size measurement method," GCC Standardization Organization, 2017, [Online]. Available: <https://www.gso.org.sa/store/gso/standards/GSO:745477/GSO%20ISO-IEC%2019761:2017?lang=en>
- [159] "ISO/IEC 25012:2008 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Data quality model," *International Organization for Standardization*, 2008.
- [160] S. Wagner *et al.*, "The Quamoco Product Quality Modelling and Assessment Approach," 2012.
- [161] P. A. Florenskii, "Some Remarks on Product Quality Assessment," *Vestn. teor. eksperiment. elektrotehniki*, no. 11, 1928.
- [162] G. G. Azgaldov and A. V. Kostin, "Applied qualimetry: its origins, errors and misconceptions," *Benchmarking: An International Journal*, vol. 18, no. 3, pp. 428–444, 2011.
- [163] T. Gilb, *Principles of Software Engineering Management*. Addison-Wesley, Reading, Mass., 1987.
- [164] "ISO/IEC 25023:2016 - Systems and software engineering - System and software product Quality Requirements and Evaluation (SQuaRE) - Measurement of system and software product quality," *International Organization for Standardization*, 2016.
- [165] "ISO/IEC 25024:2015 - Systems and Software engineering - Systems and Software product Quality Requirements and Evaluation (SQuaRE) - Measurement of data quality," *International Organization for Standardization*, 2015.
- [166] M. Shepperd, "Early life-cycle metrics and software quality models," *Information and Software Technology*, vol. 32, no. 4, pp. 311–316, May 1990, doi: 10.1016/0950-5849(90)90065-Y.
- [167] Y. Argotti, C. Baron, and P. Esteban, "Quality quantification in Systems Engineering from the Qualimetry Eye," presented at the 13th Annual IEEE International Systems Conference (SysCon), Orlando, USA, Apr. 2019.
- [168] M. Joron, C. D. Jiggins, A. Papanicolaou, and W. O. McMillan, "Heliconius wing patterns: an evo-devo model for understanding phenotypic diversity," *Heredity*, vol. 97, no. 3, pp. 157–167, Sep. 2006, doi: 10.1038/sj.hdy.6800873.
- [169] L. Cardelli and P. Wegner, "On Understanding Types, Data Abstraction, and Polymorphism," *ACM Comput. Surv.*, vol. 17, no. 4, pp. 471–523, Dec. 1985, doi: 10.1145/6041.6042.
- [170] A. Meyer, "Repeating Patterns of Mimicry," *PLOS Biology*, vol. 4, no. 10, p. e341, Oct. 2006, doi: 10.1371/journal.pbio.0040341.
- [171] R. Deshmukh, S. Baral, A. Gandhimathi, M. Kuwalekar, and K. Kunte, "Mimicry in butterflies: co-option and a bag of magnificent developmental genetic tricks," *WIREs Developmental Biology*, 2017, [Online]. Available: <https://www.advancedsciencenews.com/mimicry-butterflies-muse-palette-artist/>
- [172] W. R. Cook, W. Hill, and P. S. Canning, "Inheritance is Not Subtyping," in *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, New York, NY, USA, 1989, pp. 125–135. doi: 10.1145/96709.96721.
- [173] T. Khoshgoftaar, E. Allen, W. Jones, and J. Huddephl, "Cost-benefit analysis of software quality models," *SOFTWARE QUALITY JOURNAL*, vol. 9, no. 1, pp. 9–30, Jan. 2001, doi: 10.1023/A:1016621219262.
- [174] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, Boston, Massachusetts, 2006, pp. 775–780.
- [175] G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [176] "Single Nucleotide Polymorphism (SNP)," *Scitable by Nature Education*. Accessed: Dec. 05, 2020. [Online]. Available: <https://www.nature.com/scitable/definition/snp-295/>
- [177] G. Roedler and R. Madachy, "Measurement, SEBoK Guide to the Systems Engineering Body of Knowledge." <https://www.sebokwiki.org/wiki/Masurement> (accessed Dec. 05, 2020).
- [178] C. Miller, R. S. Carson, S. Fowler, D. J. Gantzer, and G. Roedler, "Systems Engineering Measurement Primer: a Basic Introduction to Measurement Concepts and Use for Systems Engineering," *INCOSE, San*

- Diego, CA, Nov. 2010, [Online]. Available: <https://www.incose.org/docs/default-source/ProductsPublications/systems-engineering-measurement-primer---december-2010.pdf?sfvrsn=2&sfvrsn=2>
- [179] C. Dekkers, D. Zubrow, and J. McCurley, "Measures and Measurement for Secure Software Development," 2013. <https://www.us-cert.gov/bsi/articles/best-practices/measurement/measures-and-measurement-secure-software-development>
- [180] "ISO/IEC/IEEE 15288:2015 -Systems and software engineering -- System life cycle processes," *International Organization for Standardization*, 2015, [Online]. Available: <https://www.iso.org/standard/63711.html>
- [181] W. Behutiye *et al.*, "Management of quality requirements in agile and rapid software development: A systematic mapping study," *Information and Software Technology*, vol. 123, p. 106225, Jul. 2020, doi: 10.1016/j.infsof.2019.106225.
- [182] A. Abran, P. Fagg, and A. Lesterhuis, *COSMIC Measurement Manual for ISO 19761 - Part 2: Guidelines, version 5.0*. 2020. [Online]. Available: <https://cosmic-sizing.org/measurement-manual/>
- [183] A. V. Arkhangel'skii and L. S. Pontryagin, *General Topology I: Basic Concepts and Constructions Dimension Theory*, Springer-Verlag Berlin Heidelberg., vol. 17. 1990.
- [184] R. S. Jamwal and D. Jamwal, "Issues & Factors For Evaluation of Software Quality Models," presented at the INDIACom-2009, Computing For Nation Developmen, New Delhi, India, Feb. 2009.
- [185] A. B. AL-Badareen, M. H. Selamat, M. A. Jabar, J. Din, and S. Turaev, "Software Quality Models: A Comparative Study," in *Software Engineering and Computer Systems*, Berlin, Heidelberg, 2011, pp. 46–55.
- [186] S. F. Ahmad, M. Rizwan Beg, and M. Haleem, "A Comparative Study of Software Quality Models," *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 2, no. 1, pp. 172–176, Jan. 2013.
- [187] R. Polillo, "Quality Models for Web [2.0] Sites: A Methodological Approach and a Proposal," in *Current Trends in Web Engineering*, 2011, pp. 251–265.
- [188] A. Adewumi, S. Misra, and N. Omoregbe, "A Review of Models for Evaluating Quality in Open Source Software," *IERI Procedia*, vol. 4, pp. 88–92, Jan. 2013, doi: 10.1016/j.ieri.2013.11.014.
- [189] A. Adewumi, S. Misra, N. Omoregbe, and B. Crawford, "A systematic literature review of open source software quality assessment models," *SpringerPlus*, vol. 2016, p. 1936, Nov. 2016, doi: 10.1186/s40064-016-3612-4.
- [190] J. P. Miguel, D. Mauricio, and G. Rodriguez, "A Review of Software Quality Models for the Evaluation of Software Products," *International Journal of Software Engineering & Applications (IJSEA)*, vol. 5, no. 6, pp. 31–53, Nov. 2014.
- [191] O. Gordieiev and V. Kharchenko, "IT-oriented software quality models and evolution of the prevailing characteristics," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, May 2018, pp. 375–380.
- [192] L. Buglione, "Some thoughts on quality models: Evolution and perspectives," *International Measurement Confederation (IMEKO)*, vol. 4, no. 3, pp. 72–79, Sep. 2015, doi: 10.21014/acta_imeko.v4i3.248.
- [193] M. Yan, X. Xia, X. Zhang, L. Xu, D. Yang, and S. Li, "Software quality assessment model: a systematic mapping study," *Science China Information Sciences*, vol. 62, pp. 1–18, 2019.
- [194] I. Tervonen, "A Unifying Model for Software Quality Engineering," in *Software Quality and Productivity: Theory, practice, education and training*, M. Lee, B.-Z. Barta, and P. Juliff, Eds. Boston, MA: Springer US, 1995, pp. 200–205. doi: 10.1007/978-0-387-34848-3_30.
- [195] S. N. Mohanty, "Models and Measurements for Quality Assessment of Software," *ACM Comput. Surv.*, vol. 11, no. 3, pp. 251–275, Sep. 1979, doi: 10.1145/356778.356783.
- [196] W. Frakes and C. Terry, "Software Reuse: Metrics and Models," *ACM Comput. Surv.*, vol. 28, no. 2, pp. 415–435, Jun. 1996, doi: 10.1145/234528.234531.

References

- [197] T. M. Khoshgoftaar, E. B. Allen, W. D. Jones, and J. I. Hudepohl, "Classification tree models of software quality over multiple releases," in *Proceedings 10th International Symposium on Software Reliability Engineering (Cat. No. PR00443)*, Nov. 1999, pp. 116–125. doi: 10.1109/ISSRE.1999.809316.
- [198] L. Olsina, D. Godoy, G. Lafuente, and G. Rossi, "Assessing the quality of academic websites: a case study," *New Review of Hypermedia and Multimedia*, vol. 5, no. 1, pp. 81–103, Jan. 1999, doi: 10.1080/13614569908914709.
- [199] Ping Zhang and G. von Dran, "Expectations and rankings of Web site quality features: results of two studies on user perceptions," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, Jan. 2001, p. 10 pp. doi: 10.1109/HICSS.2001.927050.
- [200] L. Briand and J. Wust, "Empirical studies of quality models in object-oriented systems," in *ADVANCES IN COMPUTERS, VOL 56*, vol. 56, Zelkowitz, MV, Ed. 525 B STREET, SUITE 1900, SAN DIEGO, CA 92101-4495 USA: ELSEVIER ACADEMIC PRESS INC, 2002, pp. 97–166. doi: 10.1016/S0065-2458(02)80005-5.
- [201] A. Rawashdeh and B. Matakah, "A New Software Quality Model for Evaluating COTS Components," *Journal of Computer Science*, vol. 2, no. 4, pp. 373–381, 2006.
- [202] R. Lincke, J. Lundberg, and W. Löwe, "Comparing Software Metrics Tools," in *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, New York, NY, USA, 2008, pp. 131–142. doi: 10.1145/1390630.1390648.
- [203] P. Mohagheghi, V. Dehlen, and T. Neple, "Definitions and approaches to model quality in model-based software development - A review of literature," *INFORMATION AND SOFTWARE TECHNOLOGY*, vol. 51, no. 12, pp. 1646–1669, Dec. 2009, doi: 10.1016/j.infsof.2009.04.004.
- [204] K. Kritikos *et al.*, "A Survey on Service Quality Description," *ACM Comput. Surv.*, vol. 46, no. 1, Jul. 2013, doi: 10.1145/2522968.2522969.
- [205] B. Gezici, A. Tarhan, and O. Chouseinoglou, "Internal and external quality in the evolution of mobile software: An exploratory study in open-source market," *INFORMATION AND SOFTWARE TECHNOLOGY*, vol. 112, pp. 178–200, Aug. 2019, doi: 10.1016/j.infsof.2019.04.002.
- [206] M. Yan, X. Xia, X. Zhang, L. Xu, and D. Yang, "A Systematic Mapping Study of Quality Assessment Models for Software Products," in *2017 International Conference on Software Analysis, Testing and Evolution (SATE)*, Nov. 2017, pp. 63–71. doi: 10.1109/SATE.2017.16.
- [207] E. Petrinja, A. Sillitti, and G. Succi, "Comparing OpenBRR, QSOS, and OMM Assessment Models," in *Open Source Software: New Horizons*, Notre Dame, IN, USA, Jun. 2010, vol. 319, pp. 224–238.
- [208] A. Fath-Allah, L. Cheikhi, R. E. Al-Qutaish, and A. Idri, "A Comparative Analysis of E-Government Quality Models," *International Journal of Social, Education, Economics and Management Engineering*, vol. 8, no. 4, pp. 3646–3650, 2014.
- [209] J. Snyder, "Google Scholar vs. Scopus & Web of Science," Feb. 29, 2012. <http://www.functionalneurogenesis.com/blog/2012/02/google-scholar-vs-scopus-web-of-science/>
- [210] "Definition of 'Taxonomy,'" *Cambridge Online English Dictionary*. Dec. 08, 2020. Accessed: Dec. 08, 2020. [Online]. Available: <https://dictionary.cambridge.org/us/dictionary/english/taxonomy>
- [211] "Definition of 'Taxon', plural 'taxa,'" *Dictionary.com (3rd version)*. Dec. 08, 2020. Accessed: Dec. 08, 2020. [Online]. Available: <https://www.dictionary.com/browse/taxon>
- [212] "Definition of 'homology,'" *Dictionary.com (3rd version)*. Dec. 08, 2020. Accessed: Dec. 08, 2020. [Online]. Available: <https://www.dictionary.com/browse/homology>
- [213] "Definition of 'Cladistic,'" *Dictionary.com (3rd version)*. [Online]. Available: <https://www.dictionary.com/browse/cladistics>
- [214] G. F. Estabrook, C. S. Johnson, and F. R. McMorris, "A mathematical foundation for the analysis of cladistic character compatibility," *Mathematical Biosciences*, vol. 29, no. 1, pp. 181–187, Jan. 1976, doi: 10.1016/0025-5564(76)90035-3.
- [215] C. Wohlin, "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering," New York, NY, USA, 2014. doi: 10.1145/2601248.2601268.

- [216] C. Wohlin, "Second-Generation Systematic Literature Studies Using Snowballing," New York, NY, USA, 2016. doi: 10.1145/2915970.2916006.
- [217] V. R. Basili and D. M. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 6, pp. 728–738, Nov. 1984, doi: 10.1109/TSE.1984.5010301.
- [218] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 4–17, Jan. 2002, doi: 10.1109/32.979986.
- [219] C. P. Team, "Capability Maturity Model Integration for Software Engineering Version 1.1 Staged Representation," Technical Report, CMU/SEI-2002-TR-012, Software Engineering Institute ..., 2002.
- [220] "ISO/IEC 15504-1:2004 - Information technology – Process assessment – Part 1: Concepts and vocabulary," *International Organization for Standardization*, 2004, [Online]. Available: <https://www.iso.org/standard/38932.html>
- [221] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 751–761, Oct. 1996, doi: 10.1109/32.544352.
- [222] I. Samoladas, G. Gousios, D. Spinellis, and I. Stamelos, "The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation," in *Open Source Development, Communities and Quality*, Boston, MA, 2008, pp. 237–248.
- [223] N. R. Haddaway, A. M. Collins, D. Coughlin, and S. Kirk, "The Role of Google Scholar in Evidence Reviews and Its Applicability to Grey Literature Searching," *PLOS ONE*, vol. 10, no. 9, p. e0138237, Sep. 2015, doi: 10.1371/journal.pone.0138237.
- [224] A. April, J. Huffman Hayes, A. Abran, and R. Dumke, "Software Maintenance Maturity Model (SMmm): the software maintenance process model," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 17, no. 3, pp. 197–223, May 2005.
- [225] 1045_WG - Std for Software Productivity Metrics Working Group, "IEEE 1045-1992 - IEEE Standard for Software Productivity Metrics." 1992. [Online]. Available: <https://standards.ieee.org/standard/1045-1992.html>
- [226] S. Knight and J. Burn, "Developing a framework for assessing information quality on the World Wide Web.," *Informing Science*, vol. 8, pp. 159–172, 2005.
- [227] "Definition of 'Practical,'" *Cambridge Online English Dictionary*. Dec. 08, 2020. Accessed: Dec. 08, 2020. [Online]. Available: <https://dictionary.cambridge.org/us/dictionary/english/practical>
- [228] K. Beck *et al.*, "Manifesto for Agile Software Development," 2001. <http://agilemanifesto.org/>
- [229] E. Georgiadou, "GEQUAMO—A Generic, Multilayered, Customisable, Software Quality Model," *Software Quality Journal*, vol. 11, no. 4, pp. 313–323, Nov. 2003, doi: 10.1023/A:1025817312035.
- [230] Y. Argotti, C. Baron, P. Esteban, and D. Chaton, "Quality Quantification Applied to Automotive Embedded Systems and Software," presented at the Embedded Real Time Systems (ERTS) 10th Edition, Toulouse, France, Jan. 2020.
- [231] Brooks, "No Silver Bullet Essence and Accidents of Software Engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987, doi: 10.1109/MC.1987.1663532.
- [232] Scaled Agile, Inc., "SAFe: Team and Technical Agility Team," Sep. 24, 2019. <https://www.scaledagileframework.com/team-and-technical-agility/> (accessed Dec. 10, 2020).
- [233] ISO/TC 176/SC 2 Quality systems, "ISO 9001:1987 Quality systems — Model for quality assurance in design/development, production, installation and servicing." International Organization for Standardization, Mar. 1987. [Online]. Available: <https://www.iso.org/standard/16533.html>
- [234] T. Miyoshi and M. Azuma, "An Empirical-Study of Evaluating Software-Development Environment Quality," *IEEE Transactions on Software Engineering*, vol. 19, no. 5, pp. 425–435, May 1993, doi: 10.1109/32.232010.
- [235] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information.," *Psychological Review*, vol. 63, no. 2, pp. 81–97, Mar. 1956, doi: 10.1037/h0043158.

References

- [236] M. B. Chrissis, M. Konrad, and S. Shrum, *CMMI for Development: Guidelines for Process Integration and Product Improvement*, 3rd ed. Addison-Wesley Professional, 2011.
- [237] CMMI Institute, *CMMI V2.0, model at-a-glance*. 2019. Accessed: Dec. 12, 2020. [Online]. Available: <https://cmmiinstitute.com/getattachment/a9b733ec-dcee-4b98-8c37-8fbd30f731de/attachment.aspx>
- [238] W. Perry, *Effective Methods for Software Testing, Third Edition*. USA: John Wiley & Sons, Inc., 2006.
- [239] J. Solomon, "Dashboards Vs Scorecards - An Insight," Oct. 05, 2006. <https://ezinearticles.com/?Dashboards-Vs-Scorecards---An-Insight&id=319136> (accessed May 05, 2006).
- [240] R. S. Kaplan and D. P. Norton, "The Balanced Scorecard—Measures that Drive Performance," Feb. 1992. <https://hbr.org/1992/01/the-balanced-scorecard-measures-that-drive-performance-2> (accessed May 29, 2020).
- [241] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Sixth Edition*. Project Management Institute, Global Standard, 2017.
- [242] R. Likert, *A Technique for the Measurement of Attitudes*, R. S. Woodworth, Editor., vol. 140. New-York, 1932. [Online]. Available: https://legacy.voteview.com/pdf/Likert_1932.pdf
- [243] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, Apr. 1960, doi: 10.1177/001316446002000104.
- [244] J. L. Fleiss, "Measuring nominal scale agreement among many raters.," *Psychological Bulletin*, vol. 76, no. 5, pp. 378–382, 1971, doi: 10.1037/h0031619.
- [245] J. R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977, doi: 10.2307/2529310.
- [246] "ISO/IEC TS 25011:2017 - Information technology - Systems and software Quality Requirements and Evaluation (SQuaRE) - Service quality models," *International Organization for Standardization*, 2017.
- [247] "ISO/IEC 25030:2007 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality requirements," *International Organization for Standardization*, 2007.
- [248] G. G. Azgaldov, *The Theory and Practice of Product Quality Assessment. Essentials of Qualimetry*, Moscow: Ekonomika (in Russian). 1982. [Online]. Available: http://www.labrate.ru/azgaldov/azgaldov_theory_and_practice_of_quality-assessment.pdf
- [249] Society of Automotive Engineers, "Class C Application Requirement Considerations J2056/1_199306," *SAE International*, no. Vehicle Architecture For Data Communications Standards, Jun. 1993, doi: 10.4271/J2056/1_199306.
- [250] N. Statt, "Tesla remotely disables Autopilot on used Model S after it was sold," *The Verge*, Feb. 2020, Accessed: Dec. 17, 2020. [Online]. Available: <https://www.theverge.com/2020/2/6/21127243/tesla-model-s-autopilot-disabled-remotely-used-car-update>
- [251] FutureBridge Analysis, "Evolution of Over-the-air Software Updates in Automotive," *Over-the-Air Software Updates – Reaping Benefits for the Automotive Industry*, Jan. 22, 2020. <https://www.futurebridge.com/blog/over-the-air-software-updates-reaping-benefits-for-the-automotive-industry/> (accessed Dec. 17, 2020).
- [252] J. Gill, J. Hu, L. Shan, S. Thackray, and S. Unger, "Over-the-Air Updates with Electric Vehicles," *New Media in Business Blog*, Aug. 15, 2020. https://www.newmediabusinessblog.org/index.php/Over-the-Air_Updates_with_Electric_Vehicles (accessed Dec. 17, 2020).
- [253] Automotive SIG, VDA, "Automotive SPICE Process Assessment, version 2.5." May 10, 2010. [Online]. Available: http://www.automotivespice.com/fileadmin/software-download/automotiveSIG_PAM_v25.pdf
- [254] T. J. McCabe, "A Complexity Measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, Dec. 1976, doi: 10.1109/TSE.1976.233837.
- [255] J. K. Haseman and R. C. Elston, "The investigation of linkage between a quantitative trait and a marker locus," *Behavior Genetics*, vol. 2, no. 1, pp. 3–19, Mar. 1972, doi: 10.1007/BF01066731.
- [256] "Definition of 'Gene,'" *Cambridge Online English Dictionary*. Dec. 08, 2020. Accessed: Dec. 08, 2020. [Online]. Available: <https://dictionary.cambridge.org/dictionary/english/gene>

- [257] “Genetic polymorphism,” *Biology-Online Dictionary*. Dec. 27, 2020. Accessed: Dec. 27, 2020. [Online]. Available: <https://www.biologyonline.com/dictionary/genetic-polymorphism>
- [258] Z. Bo and Z. Yan, “FIT ANALYSIS OF WEB QUALITY MODEL BASED ON SEM APPROACH,” in *2011 3RD INTERNATIONAL CONFERENCE ON COMPUTER TECHNOLOGY AND DEVELOPMENT (ICCTD 2011), VOL 2, THREE PARK AVENUE, NEW YORK, NY 10016-5990 USA, 2012*, pp. 359–363.
- [259] C. Alves, X. Franch, J. Carvallo, and A. Finkelstein, “Using goals and quality models to support the matching analysis during COTS selection,” in *COTS-BASED SOFTWARE SYSTEMS, PROCEEDINGS, HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY, 2005*, vol. 3412, pp. 146–156.
- [260] A. Adewumi, S. Misra, and N. Omoregbe, “A Review of Models for Evaluating Quality in Open Source Software,” *IERI Procedia*, vol. 4, pp. 88–92, Jan. 2013, doi: 10.1016/j.ieri.2013.11.014.
- [261] European Space Agency and TEC-SHS, “Technology Readiness Levels Handbook for Sapce Applications,” TEC-SHS/5551/MG/ap, Sep. 2008. [Online]. Available: https://artes.esa.int/sites/default/files/TRL_Handbook.pdf
- [262] N. F. Porta, “Towards a Model for Cost-Benefit-Analysis of Quality Assurance in the Automotive E/E Development,” in *Proceedings of the 1st International Workshop on Business Impact of Process Improvements*, New York, NY, USA, 2008, pp. 33–38. doi: 10.1145/1370837.1370844.
- [263] H. Kuder *et al.*, “HIS source code metrics, version 1.3.1,” *Hersteller Initiative Software-AK Softwaretest*, vol. 1, no. 1, 2008.
- [264] T. M. Khoshgoftaar and R. M. Szabo, “Improving neural network predictions of software quality using principal components analysis,” in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, Jun. 1994, vol. 5, pp. 3295–3300 vol.5. doi: 10.1109/ICNN.1994.374764.
- [265] T. M. Khoshgoftaar, E. B. Allen, K. S. Kalaichelvan, and N. Goel, “The impact of software evolution and reuse on software quality,” *Empirical Software Engineering*, vol. 1, no. 1, pp. 31–44, Jan. 1996, doi: 10.1007/BF00125810.
- [266] T. Dahlberg and J. Jarvinen, “Challenges to IS quality,” *INFORMATION AND SOFTWARE TECHNOLOGY*, vol. 39, no. 12, pp. 809–818, Dec. 1997, doi: 10.1016/S0950-5849(97)00039-6.
- [267] J. C. Granja-Alvarez and M. J. Barranco-García, “A Method for Estimating Maintenance Cost in a Software Project: A Case Study,” *Journal of Software Maintenance: Research and Practice*, vol. 9, no. 3, pp. 161–175, May 1997, doi: 10.1002/(SICI)1096-908X(199705)9:3<161::AID-SMR148>3.0.CO;2-8.
- [268] T. Khoshgoftaar, E. Allen, A. Naik, W. Jones, and J. Hudepohl, “Using classification trees for software quality models: Lessons learned,” in *THIRD IEEE INTERNATIONAL HIGH-ASSURANCE SYSTEMS ENGINEERING SYMPOSIUM, PROCEEDINGS*, 1998, pp. 82–89. doi: 10.1109/HASE.1998.731598.
- [269] Y. Yokoyama and M. Kodaira, “Software cost and quality analysis by statistical approaches,” in *Proceedings of the 20th International Conference on Software Engineering*, Apr. 1998, pp. 465–467. doi: 10.1109/ICSE.1998.671607.
- [270] T. M. Khoshgoftaar, R. Shan, and E. B. Allen, “Improving tree-based models of software quality with principal components analysis,” in *Proceedings 11th International Symposium on Software Reliability Engineering. ISSRE 2000*, Oct. 2000, pp. 198–209. doi: 10.1109/ISSRE.2000.885872.
- [271] T. M. Khoshgoftaar, E. B. Allen, W. D. Jones, and J. P. Hudepohl, “Cost-Benefit Analysis of Software Quality Models,” *Software Quality Journal*, vol. 9, no. 1, pp. 9–30, 2001, doi: 10.1023/A:1016621219262.
- [272] H. Zhu, Y. Zhang, Q. Huo, and S. Greenwood, “Application of hazard analysis to software quality modelling,” in *Proceedings 26th Annual International Computer Software and Applications*, Aug. 2002, pp. 139–144. doi: 10.1109/CMPSAC.2002.1044544.
- [273] C. V. Ramamoorthy, “Evolution and evaluation of software quality models,” in *14th IEEE International Conference on Tools with Artificial Intelligence, 2002. (ICTAI 2002). Proceedings.*, Nov. 2002, pp. 543–545. doi: 10.1109/TAI.2002.1180850.
- [274] L. E. Mendoza, A. C. Grimán, M. A. Pérez, and T. Rojas, “Evaluation of Environments for Portals Development: A Case Study,” *Information Systems Management*, vol. 19, no. 2, pp. 70–84, Feb. 2002, doi: 10.1201/1078/43200.19.2.20020228/35141.7.

-
- [275] T. M. Khoshgoftaar, E. Geleyn, and L. Nguyen, "Empirical case studies of combining software quality classification models," in *Third International Conference on Quality Software, 2003. Proceedings.*, Nov. 2003, pp. 40–49. doi: 10.1109/QSIC.2003.1319084.
- [276] Y. Liu and T. Khoshgoftaar, "Reducing overfitting in genetic programming models for software quality classification," in *Eighth IEEE International Symposium on High Assurance Systems Engineering, 2004. Proceedings.*, Mar. 2004, pp. 56–65. doi: 10.1109/HASE.2004.1281730.
- [277] J. Oh, D. Park, B. Lee, J. Lee, E. Hong, and C. Wu, "Certification of Software Packages Using Hierarchical Classification," in *Software Engineering Research and Applications*, Berlin, Heidelberg, 2004, pp. 209–224.
- [278] M.-A. Côté, W. Suryan, C. Y. Laporte, and R. A. Martin, "The evolution path for industrial software quality evaluation methods applying ISO/IEC 9126:2001 quality model: Example of MITRE's SQAE method," *Software Quality Journal*, vol. 13, no. 1, pp. 17–30, 2005, doi: 10.1007/s11219-004-5259-6.
- [279] Q. Zhang, J. Wu, and H. Zhu, "Tool Support to Model-based Quality Analysis of Software Architecture," in *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, Sep. 2006, vol. 1, pp. 121–128. doi: 10.1109/COMPSAC.2006.82.
- [280] A. Mishra and D. Mishra, "Software quality assurance models in small and medium organisations: A comparison," *International Journal of Information Technology and Management*, vol. 5, no. 1, pp. 4–20, 2006, doi: 10.1504/IJITM.2006.008710.
- [281] H. Jung, W. Jung, and H. Yang, "A study on the standard of software quality testing," in *COMPUTATIONAL SCIENCE AND ITS APPLICATIONS - ICCSA 2006, PT 4*, 2006, vol. 3983, pp. 1052–1059.
- [282] C.-T. Wang, C.-C. Lo, and T.-F. Jean, "Probabilistic models for software quality analysis," *Journal of the Chinese Institute of Industrial Engineers*, vol. 23, no. 4, pp. 328–336, 2006, doi: 10.1080/10170660609509329.
- [283] J. Ruiz, C. Calero, and M. Piattini, "Web metrics selection through a practitioners' survey," in *ICSOFT 2006: Proceedings of the First International Conference on Software and Data Technologies, Vol 1*, 2006, pp. 238–244.
- [284] O. Ormandjieva, I. Hussain, and L. Kosseim, "Toward a text classification system for the quality assessment of software requirements written in natural language," in *Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting*, Dubrovnik, Croatia, 2007, pp. 39–45. [Online]. Available: <https://doi.org/10.1145/1295074.1295082>
- [285] P. L. Roden, S. Virani, L. H. Etkorn, and S. Messimer, "An Empirical Study of the Relationship of Stability Metrics and the QMOOD Quality Models Over Software Developed Using Highly Iterative or Agile Software Processes," in *Seventh IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM 2007)*, Oct. 2007, pp. 171–179. doi: 10.1109/SCAM.2007.29.
- [286] Y. Ma and B. Cukic, "Adequate and Precise Evaluation of Quality Models in Software Engineering Studies," in *Third International Workshop on Predictor Models in Software Engineering (PROMISE'07: ICSE Workshops 2007)*, May 2007, pp. 1–1. doi: 10.1109/PROMISE.2007.1.
- [287] S. Neti and H. A. Muller, "Quality Criteria and an Analysis Framework for Self-Healing Systems," in *International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '07)*, May 2007, pp. 6–6. doi: 10.1109/SEAMS.2007.15.
- [288] L. Zhang, L. Li, and H. Gao, "2-D Software Quality Model and Case Study in Software Flexibility Research," in *2008 International Conference on Computational Intelligence for Modelling Control & Automation*, Dec. 2008, pp. 1147–1152. doi: 10.1109/CIMCA.2008.70.
- [289] A. A. Hamada, M. N. Moustafa, and H. I. Shaheen, "Software Quality model Analysis Program," in *2008 International Conference on Computer Engineering & Systems*, Nov. 2008, pp. 296–300. doi: 10.1109/ICCES.2008.4773015.
- [290] X. Feng and Y. Liu, "A Study on Evaluation Model of Information Sharing Quality in Virtual Teams," in *2008 International Conference on Computer Science and Software Engineering*, Dec. 2008, vol. 5, pp. 117–120. doi: 10.1109/CSSE.2008.150.

- [291] O. Alfonzo, K. Domínguez, L. Rivas, M. Pérez, L. Mendoza, and M. Ortega, "Quality Measurement Model for Analysis and Design Tools Based on FLOSS," in *19th Australian Conference on Software Engineering (aswec 2008)*, Mar. 2008, pp. 258–268. doi: 10.1109/ASWEC.2008.4483214.
- [292] M. Bombardieri and F. A. Fontana, "A specialisation of the SQuaRE quality model for the evaluation of the software evolution and maintenance activity," in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops*, Sep. 2008, pp. 110–113. doi: 10.1109/ASEW.2008.4686328.
- [293] H. P. Breivold and I. Crnkovic, "Analysis of Software Evolvability in Quality Models," in *2009 35th Euromicro Conference on Software Engineering and Advanced Applications*, Aug. 2009, pp. 279–282. doi: 10.1109/SEAA.2009.10.
- [294] F. Khomh, "SQUAD: Software Quality Understanding through the Analysis of Design," in *2009 16th Working Conference on Reverse Engineering*, Oct. 2009, pp. 303–306. doi: 10.1109/WCRE.2009.22.
- [295] R. Brcina, S. Bode, and M. Riebisch, "Optimisation Process for Maintaining Evolvability during Software Evolution," in *2009 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, Apr. 2009, pp. 196–205. doi: 10.1109/ECBS.2009.20.
- [296] Q. Yu-dong, Z. Ai-hong, X. Xiao-fang, and Y. Xiao-bin, "Analysis of contribution of conceptual model quality to software reliability," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, Oct. 2010, vol. 10, pp. V10-386. doi: 10.1109/ICCASM.2010.5622740.
- [297] J. Letouzey and T. Coq, "The SQuALE Analysis Model: An Analysis Model Compliant with the Representation Condition for Assessing the Quality of Software Source Code," in *2010 Second International Conference on Advances in System Testing and Validation Lifecycle*, Aug. 2010, pp. 43–48. doi: 10.1109/VALID.2010.31.
- [298] R. Lincke, T. Gutzmann, and W. Löwe, "Software Quality Prediction Models Compared," in *2010 10th International Conference on Quality Software*, Jul. 2010, pp. 82–91. doi: 10.1109/QSIC.2010.9.
- [299] E. Chandra, D. Francis Xavier Christopher, and S. D. Vijaykumar, "Study of CMMI based process framework for quality models," in *3rd International Conference on Trendz in Information Sciences & Computing (TISC2011)*, Dec. 2011, pp. 173–176. doi: 10.1109/TISC.2011.6169109.
- [300] T. Coq and J.-P. Rosen, "The SQuALE Quality and Analysis Models for Assessing the Quality of Ada Source Code," in *RELIABLE SOFTWARE TECHNOLOGIES - ADA-EUROPE 2011*, 2011, vol. 6652, pp. 61–74.
- [301] K. Lochmann and L. Heinemann, "Integrating quality models and static analysis for comprehensive quality assessment," in *Proceedings - International Conference on Software Engineering (ICSE)*, May 2011, pp. 5–11.
- [302] D. Nabil, A. Mosad, and H. A. Hefny, "Web-Based Applications quality factors: A survey and a proposed conceptual model," *Egyptian Informatics Journal*, vol. 12, no. 3, pp. 211–217, Nov. 2011.
- [303] S. Montagud, S. Abrahao, and E. Insfran, "A systematic review of quality attributes and measures for software product lines," *SOFTWARE QUALITY JOURNAL*, vol. 20, no. 3–4, SI, pp. 425–486, Sep. 2012, doi: 10.1007/s11219-011-9146-7.
- [304] L. Cheikhi, A. Abran, and L.-M. Desharnais, "Analysis of the ISBSG Software Repository from the ISO 9126 View of Software Product Quality," in *38TH ANNUAL CONFERENCE ON IEEE INDUSTRIAL ELECTRONICS SOCIETY (IECON 2012)*, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2012, pp. 3086–3094.
- [305] K. Lochmann, D. M. Fernandez, and S. Wagner, "A Case Study on Specifying Quality Requirements Using a Quality Model," in *2012 19TH ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC), VOL 1*, 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA, 2012, pp. 577–582. doi: 10.1109/APSEC.2012.57.
- [306] M. Galster and P. Avgeriou, "Qualitative Analysis of the Impact of SOA Patterns on Quality Attributes," in *2012 12TH INTERNATIONAL CONFERENCE ON QUALITY SOFTWARE (QSIC)*, 2012, pp. 167–170. doi: 10.1109/QSIC.2012.35.
- [307] H. Wan-jiang and L. Tian-bo, "Study on quality evaluation model of communication system," in *2012 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization*, Oct. 2012, vol. 1, pp. 1–4. doi: 10.1109/ICSSEM.2012.6340726.

- [308] G. A. García-Mireles, M. Á. Ángeles Moraga, and F. García, "Development of maturity models: A systematic literature review," in *16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, May 2012, pp. 279–283. doi: 10.1049/ic.2012.0036.
- [309] H.-J. Jeong and S.-J. Hong, "The survey of quality model for software and system," in *Lecture Notes in Electrical Engineering*, Dec. 2012, vol. 114, pp. 569–577.
- [310] S. K. Dubey, S. Ghosh, and A. Rana, "Comparison of Software Quality Models: An Analytical Approach," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 2, pp. 111–119, Feb. 2012.
- [311] B. Singh and S. P. Kannoja, "A Review on Software Quality Models," in *2013 INTERNATIONAL CONFERENCE ON COMMUNICATION SYSTEMS AND NETWORK TECHNOLOGIES (CSNT 2013)*, 2013, pp. 801–806. doi: 10.1109/CSNT.2013.171.
- [312] C. Calero, M. F. Bertoa, and M. A. Moraga, "A Systematic Literature Review for Software Sustainability Measures," in *2013 2ND INTERNATIONAL WORKSHOP ON GREEN AND SUSTAINABLE SOFTWARE (GREENS)*, 2013, pp. 46–53.
- [313] A. Adewumi, N. Omoregbe, S. Misra, and L. Fernandez, "Quantitative Quality Model for Evaluating Open Source Web Applications: Case Study of Repository Software," in *2013 IEEE 16TH INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND ENGINEERING (CSE 2013)*, 2013, pp. 1207–1213. doi: 10.1109/CSE.2013.179.
- [314] K. Li, J. Xiao, Y. Wang, and Q. Wang, "Analysis of the Key Factors for Software Quality in Crowdsourcing Development: An Empirical Study on TopCoder.com," in *2013 IEEE 37TH ANNUAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC)*, 2013, pp. 812–817. doi: 10.1109/COMPSAC.2013.133.
- [315] M. Ericsson, W. Lowe, T. Olsson, D. Toll, and A. Wingkvist, "A Study of the Effect of Data Normalization on Software and Information Quality Assessment," in *2013 20TH ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC 2013), VOL 2*, 2013, pp. 55–60. doi: 10.1109/APSEC.2013.112.
- [316] H. K. A. Bakar and R. Razali, "A preliminary review of legacy information systems evaluation models," in *2013 International Conference on Research and Innovation in Information Systems (ICRIIS)*, Nov. 2013, pp. 314–318. doi: 10.1109/ICRIIS.2013.6716728.
- [317] P. Hegedüs, "A probabilistic quality model for C# -an industrial case study," in *Acta Cybernetica, Volume 21, Issue 1*, 2013, pp. 135–147.
- [318] A. B. Tomar and V. M. Thakare, "A Customized Model on Software Quality Assurance & Reuse," *International Journal Of Computer Science And Applications*, vol. 6, no. 2, Apr. 2013, [Online]. Available: <http://researchpublications.org/IJCSA/NCAICN-13/194.pdf>
- [319] S. Gupta and H. K. Singh, "A Semiautomated Method for Classifying Program Analysis Rules into a Quality Model," in *Proceedings of the 22nd International Conference on Program Comprehension*, New York, NY, USA, 2014, pp. 266–270. doi: 10.1145/2597008.2597808.
- [320] R. Goyal, P. Chandra, and Y. Singh, "Why Interaction between Metrics Should Be Considered in the Development of Software Quality Models: A Preliminary Study," *SIGSOFT Softw. Eng. Notes*, vol. 39, no. 4, pp. 1–4, Aug. 2014, doi: 10.1145/2632434.2659853.
- [321] T. Davuluru, J. Medida, and V. S. K. Reddy, "A Study of Software Quality Models," 345 E 47TH ST, NEW YORK, NY 10017 USA, 2014.
- [322] S. Ouhbi, A. Idri, J. L. Fernandez Aleman, and A. Toval, "Evaluating Software Product Quality: A Systematic Mapping Study," in *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, Oct. 2014, pp. 141–151. doi: 10.1109/IWSM.Mensura.2014.30.
- [323] H. Zhu, Q. Zhang, and Y. Zhang, "HASARD: A Model-Based Method for Quality Analysis of Software Architecture," *Relating System Quality and Software Architecture*, pp. 123–156, Jul. 2014.
- [324] E. Yildiz, S. Bilgen, G. Tokdemir, N. E. Cagiltay, and Y. N. Erturan, "Analysis of B2C mobile application characteristics and quality factors based on ISO 25010 quality model," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8640, pp. 261–274.

- [325] E. Ronchieri, M. Canaparo, and D. Salomoni, "A software quality model by using discriminant analysis predictive technique," *Journal of Integrated Design and Process Science*, vol. 18, no. 4, pp. 25–59, 2014.
- [326] S. S.-S. Cherfi, A. D. Tuan, and I. Comyn-Wattiau, "An Exploratory Study on Websites Quality Assessment," in *ADVANCES IN CONCEPTUAL MODELING, ER 2013*, 2014, vol. 8697, pp. 170–179.
- [327] M. Sarrab and O. M. H. Rehman, "Empirical study of open source software selection for adoption, based on software quality characteristics," *ADVANCES IN ENGINEERING SOFTWARE*, vol. 69, pp. 1–11, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.001.
- [328] L. Buglione, "Software product quality: Some thoughts about its evolution and perspectives," in *20th IMEKO TC4 Symposium on Measurements of Electrical Quantities: Research on Electrical and Electronic Measurement for the Economic Upturn, Together with 18th TC4 International Workshop on ADC and DCA Modeling and Testing*, Sep. 2014, pp. 737–742.
- [329] D. Gupta, A. Ahlawat, and K. Sagar, "A Critical Analysis of A Hierarchy Based Usability Model," in *2014 INTERNATIONAL CONFERENCE ON CONTEMPORARY COMPUTING AND INFORMATICS (IC3I)*, 2014, pp. 255–260.
- [330] S. Manoj Wadhwa, "A Comparative Study of Software Quality Models," *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 5, no. 4, pp. 5634–5638, 2014.
- [331] K. Sheoran and O. P. Sangwan, "An Insight of software quality models applied in predicting software quality attributes: A comparative analysis," in *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, Sep. 2015, pp. 1–5. doi: 10.1109/ICRITO.2015.7359355.
- [332] P. Hegedus, "Advances in Software Product Quality Measurement and Its Applications in Software Evolution," in *2015 31ST INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE AND EVOLUTION (ICSME) PROCEEDINGS*, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2015, pp. 590–593.
- [333] M. K. Chawla and I. Chhabra, "SQMMA: Software quality model for maintainability analysis," in *ACM International Conference Proceeding Series*, Oct. 2015, pp. 9–17.
- [334] F. Imeri, L. Antovski, and M. Hamiti, "Empirical Analysis of Quality Models in Practice in Small IT Companies in SEE Region," *Procedia - Social and Behavioral Sciences*, vol. 191, pp. 969–974, Jun. 2015, doi: 10.1016/j.sbspro.2015.04.490.
- [335] A. Ganser, H. Lichter, A. Roth, and B. Rumpe, "Staged model evolution and proactive quality guidance for model libraries," *SOFTWARE QUALITY JOURNAL*, vol. 24, no. 3, pp. 675–708, Sep. 2016, doi: 10.1007/s11219-015-9298-y.
- [336] M. A. Kabir, M. U. Rehman, and S. I. Majumdar, "An Analytical and Comparative Study of Software Usability Quality Factors Usability Model in Software Engineering Literature," in *PROCEEDINGS OF 2016 IEEE 7TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND SERVICE SCIENCE (ICSESS 2016)*, 2016, pp. 800–803.
- [337] D. Di Ruscio, D. S. Kolovos, Y. Korkontzelos, N. Matragkas, and J. Vinju, "Supporting Custom Quality Models to Analyse and Compare Open-Source Software," in *PROCEEDINGS 2016 10TH INTERNATIONAL CONFERENCE ON THE QUALITY OF INFORMATION AND COMMUNICATIONS TECHNOLOGY (QUATIC)*, 2016, pp. 94–99. doi: 10.1109/QUATIC.2016.23.
- [338] Z. Qian, C. Wan, and Y. Chen, "Evaluating quality-in-use of FLOSS through analyzing user reviews," in *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Jun. 2016, pp. 547–552. doi: 10.1109/SNPD.2016.7515956.
- [339] L. Sergio, P. Silva, S. C. B. Sampaio, E. R. de Souza, R. T. Moreira, and A. M. L. Vasconcelos, "Mapping between the guide of it solution contract and CMMI models: A qualitative analysis," in *PROCEEDINGS 2016 10TH INTERNATIONAL CONFERENCE ON THE QUALITY OF INFORMATION AND COMMUNICATIONS TECHNOLOGY (QUATIC)*, 2016, pp. 150–153. doi: 10.1109/QUATIC.2016.32.
- [340] U. Devi, A. Sharma, and N. Kesswani, "A Review on Quality Models to Analyse the Impact of Refactored Code on Maintainability with reference to Software Product Line," in *PROCEEDINGS OF THE 10TH INDIACOM - 2016 3RD INTERNATIONAL CONFERENCE ON COMPUTING FOR SUSTAINABLE GLOBAL DEVELOPMENT*, 2016, pp. 3705–3708.

- [341] S. Pattnaik and B. K. Pattanayak, "A survey on machine learning techniques used for software quality prediction," *International Journal of Reasoning-based Intelligent Systems*, vol. 8, no. 1/2, pp. 3–14, 2016, doi: 10.1504/IJRIS.2016.080058.
- [342] C. I. M. Bezerra, R. M. C. Andrade, and J. M. Monteiro, "Exploring quality measures for the evaluation of feature models: a case study," *JOURNAL OF SYSTEMS AND SOFTWARE*, vol. 131, pp. 366–385, Sep. 2017, doi: 10.1016/j.jss.2016.07.040.
- [343] M. Santos, P. J. Afonso, P. H. Bermejo, and H. Costa, "Metrics and Statistical Techniques Used to Evaluate Internal Quality of Object-Oriented Software: A Systematic Mapping," 2016.
- [344] N. R. M. Suradi, S. Kahar, and N. A. A. Jamaludin, "A Review on Software Quality Attributes for Web-Based Application," in *PROCEEDING OF THE 1ST INTERNATIONAL CONFERENCE OF ENGINEERING AND APPLIED SCIENCE (INCEAS 2016)*, 2016, pp. 180–190.
- [345] I. Griffith, C. Izurieta, and C. Huvaere, "An Industry Perspective to Comparing the SQA and Quamoco Software Quality Models," in *11TH ACM/IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT (ESEM 2017)*, 2017, pp. 287–296. doi: 10.1109/ESEM.2017.42.
- [346] K. Moumane and A. Idri, "Software quality in mobile environments: A comparative study," in *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, Apr. 2017, pp. 1123–1128. doi: 10.1109/CoDIT.2017.8102750.
- [347] T. Wahyuningrum and K. Mustofa, "A systematic mapping review of software quality measurement: Research trends, model, and method," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 5, pp. 2847–2854, Oct. 2017.
- [348] F. D. Giraldo, S. España, Ó. Pastor, and W. J. Giraldo, "Considerations about quality in model-driven engineering: Current state and challenges," *Software Quality Journal*, vol. 26, no. 2, pp. 685–750, Jun. 2018.
- [349] D. Russo, P. Ciancarini, T. Falasconi, and M. Tomasi, "A Meta-Model for Information Systems Quality: A Mixed Study of the Financial Sector," *ACM TRANSACTIONS ON MANAGEMENT INFORMATION SYSTEMS*, vol. 9, no. 3, Nov. 2018, doi: 10.1145/3230713.
- [350] R. Wahdiniwati, E. B. Setiawan, and D. A. Wahab, "Comparative Analysis of Software Quality Model In The Selection of Marketplace E-Commerce," in *2018 INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY SYSTEMS AND INNOVATION (ICITSI)*, 2018, pp. 386–391.
- [351] D. Gatica, F. Ponce, R. Noël, and H. Astudillo, "Characterizing Architectural Evaluations and Identifying Quality Attributes addressed in Systems-of-Systems: A Systematic Mapping Study," in *2018 37th International Conference of the Chilean Computer Science Society (SCCC)*, Nov. 2018, pp. 1–7. doi: 10.1109/SCCC.2018.8705229.
- [352] A. J. Abdellatif, B. McCollum, and P. McMullan, "Serious Games: Quality Characteristics Evaluation Framework and Case Study," in *PROCEEDINGS OF THE 8TH IEEE INTEGRATED STEM EDUCATION CONFERENCE (ISEC 2018)*, 2018, pp. 112–119.
- [353] N. Zighed, N. Bounour, and A.-D. Seriai, "Comparative Analysis of Object-Oriented Software Maintainability Prediction Models," *FOUNDATIONS OF COMPUTING AND DECISION SCIENCES*, vol. 43, no. 4, pp. 359–374, Dec. 2018, doi: 10.1515/fcds-2018-0018.
- [354] M. Rai and K. S. Virk, "Software Component Quality Models: A Survey," in *Advances in Intelligent Systems and Computing*, 2018, pp. 247–255.
- [355] V. Nikolic *et al.*, "Survey of quality models of e-learning systems," *PHYSICA A-STATISTICAL MECHANICS AND ITS APPLICATIONS*, vol. 511, pp. 324–330, Dec. 2018, doi: 10.1016/j.physa.2018.07.058.
- [356] K. Mossakowska and A. Jarzebowicz, "A Survey Investigating the Influence of Business Analysis Techniques on Software Quality Characteristics," in *TOWARDS A SYNERGISTIC COMBINATION OF RESEARCH AND PRACTICE IN SOFTWARE ENGINEERING*, 2018, vol. 733, pp. 135–148. doi: 10.1007/978-3-319-65208-5_10.
- [357] P. Nistala, K. V. Nori, and R. Reddy, "Software Quality Models: A Systematic Mapping Study," in *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, May 2019, pp. 125–134. doi: 10.1109/ICSSP.2019.00025.

- [358] N. Condori-Fernandez and P. Lago, "Towards a Software Sustainability-Quality Model: Insights from a Multi-Case Study," in *2019 13th International Conference on Research Challenges in Information Science (RCIS)*, May 2019, pp. 1–11. doi: 10.1109/RCIS.2019.8877084.
- [359] G. Arcos-Medina and D. Mauricio, "Aspects of software quality applied to the process of agile software development: a systematic literature review," *International Journal of Systems Assurance Engineering and Management*, vol. 10, no. 5, pp. 867–897, Oct. 2019.
- [360] J. Jelinski and P. B. Moranda, "Software reability research," *W. Freiberger (Ed), Academic Press, New York*, pp. 485–502, 1972.
- [361] G. J. Schick and R. W. Wolverton, "Assessment of software reliability," presented at the 11th Annum Meeting, German Operation Research Society, Hamburg, Germany, Sep. 1972.
- [362] H. Mills, "On the statistical validation of computer programs," IBM Federal Systems Division, Technical Report FSC-72-6015, 1972.
- [363] B. Littlewood and J. L. Verrall, "A Bayesian Reliability Growth Model for Computer Software," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 22, no. 3, pp. 332–346, 1973, doi: 10.2307/2346781.
- [364] J. D. Musa, "A theory of software reliability and its application," *IEEE Transactions on Software Engineering*, vol. SE-1, no. 3, pp. 312–327, Sep. 1975, doi: 10.1109/TSE.1975.6312856.
- [365] Y. Funami and M. H. Halstead, "A software physics analysis of Aklyama's debugging data," in *Proc. MRI Symp. Computer Software Engineering*, 1975, pp. 133–138.
- [366] N. S. Mohanty and M. Adamowicz, "Proposed measures for the evaluation of software," in *Proc. Symposium on Computer Soft. Eng., MRI, New York Polytechnic*, 1976, pp. 485–497. [Online]. Available: <https://www.oldcomputerbooks.com/pages/books/C811093/jerome-fox-polytechnic-press-of-the-polytechnic-institute-of-new-york/proceedings-of-the-symposium-on-computer-software-engineering-new-york-1976-microwave-research>
- [367] R. K. Klobert, "Calculation of error process of computer program," in *Proc. Computers*, Los Angeles, Nov. 1977, pp. 442–426.
- [368] N. F. Schneidewind, "The use of simulation in the evaluation of software," *Computer*, vol. 10, no. 4, pp. 47–53, Apr. 1977.
- [369] S. Henry and D. Kafura, "Software Structure Metrics Based on Information Flow," *IEEE Transactions on Software Engineering*, vol. SE-7, no. 5, pp. 510–518, Sep. 1981, doi: 10.1109/TSE.1981.231113.
- [370] T. P. Bowen, J. V. Post, J. Tsai, P. E. Presson, and R. L. Schmidt, "Software Quality Measurement for Distributed Systems. Volume 2. Guidebook for Software Quality Measurement," Rome Air Development Center, Air Force Systems Command, Griffis Air Force Base, NY, RADC-TR-83-175-VOL-2, Jul. 1983. [Online]. Available: <https://apps.dtic.mil/docs/citations/ADA137956>
- [371] T. P. Bowen, G. B. Wigle, and I. T. Tsai, "Specification of software quality attributes," Rome Air Development Center, RADC-TR-85-37, vols. I & II, 1985.
- [372] N. Langberg and N. D. Singpurwalla, "A Unification of Some Software Reliability Models," *SIAM J. Sci. and Stat. Comput.*, vol. 6, no. 3, pp. 781–790, Jul. 1985, doi: 10.1137/0906053.
- [373] V. Y. Shen, T. Yu, S. M. Thebaut, and L. R. Paulsen, "Identifying Error-Prone Software—An Empirical Study," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 4, pp. 317–324, Apr. 1985, doi: 10.1109/TSE.1985.232222.
- [374] T. Sunazuka, M. Azuma, and N. Yamagishi, "Software Quality Assessment Technology," in *Proceedings of the 8th International Conference on Software Engineering*, Washington, DC, USA, 1985, pp. 142–148.
- [375] T. K. Nayak, "Software Reliability: Statistical Modeling & Estimation," *IEEE Transactions on Reliability*, vol. 35, no. 5, pp. 566–570, Dec. 1986, doi: 10.1109/TR.1986.4335548.
- [376] National Institute of Standards and Technology, "The Malcolm Baldrige National Quality Award 1996, Application Forms and Instructions," presented at the United States Department of Commerce, Gaithersburg, GA, 1996.
- [377] M. Evans and J. Marciniak, "Software Quality Assurance and Management," *New York: John Wiley*, 1987.

References

- [378] D. N. Card and W. W. Agresti, "Measuring software design complexity," *Journal of Systems and Software*, vol. 8, no. 3, pp. 185–197, Jun. 1988, doi: 10.1016/0164-1212(88)90021-0.
- [379] S. Wake and S. Henry, "A model based on software quality factors which predicts maintainability," in *Proceedings. Conference on Software Maintenance, 1988.*, Oct. 1988, pp. 382–387. doi: 10.1109/ICSM.1988.10191.
- [380] R. W. Selby and A. A. Porter, "Learning from examples: generation and evaluation of decision trees for software resource analysis," *IEEE Transactions on Software Engineering*, vol. 14, no. 12, pp. 1743–1757, Dec. 1988, doi: 10.1109/32.9061.
- [381] V. A. Zeithaml, L. L. Berry, and A. Parasuraman, "SERVQUAL: a multiple-item scale for measuring consumer perceptions of service quality," *Journal of retailing*, vol. 64, no. 1, pp. 12–40, 1988.
- [382] J. Gaffney and T. Durek, "Software reuse-key to enhanced productivity: some quantitative models," *Information and Software Technology*, vol. 31, no. 5, pp. 258–267, Jun. 1989, doi: 10.1016/0950-5849(89)90005-0.
- [383] S. Henry and S. Wake, "Predicting maintainability with software quality metrics," *Journal of Software Maintenance: Research and Practice*, vol. 3, no. 3, pp. 129–143, Sep. 1991, doi: 10.1002/smr.4360030302.
- [384] P. Koltun and A. Hudson, "A reuse maturity model," presented at the Fourth Annual Workshop on Software Reuse, Herndon, VA, USA, 1991.
- [385] I. Eriksson and A. Törn, "A Model for IS Quality," *Software Engineering Journal*, pp. 152–158, Jul. 1991.
- [386] ISO/IEC JTC 1/SC 7 Software and systems engineering, "ISO 9126:1991 Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for Their Use." International Organization for Standardization, Dec. 1991. [Online]. Available: <https://www.iso.org/standard/16722.html>
- [387] V. A. Zeithaml, L. L. Berry, and A. Parasuraman, "Refinement and reassess of the SERVQUAL scale," *Journal of retailing*, vol. 67, no. 4, pp. 420–450, 1991.
- [388] W. E. Perry, *Quality Assurance for Information Systems: Methods, Tools, and Techniques*. USA: QED Information Sciences, Inc., 1991.
- [389] K. A. Dyson, "Quality Evaluation System (QUES) Software Quality Framework as Implemented in QUES. Volume 2," Rome Air Development Center, Rome, Defense Technical Information Center, RL-TR-91-407, Vol II (of two), Dec. 1991. [Online]. Available: https://archive.org/details/DTIC_ADA252976/mode/2up
- [390] Ghezzi, C. M. Jazayeri, and D. Mandrioli, "Fundamental of software Engineering," *Prentice-Hall, NJ, USA*, 1991.
- [391] B. Shackel, "Usability—Context, Framework, Definition, Design and Evaluation," in *Human Factors for Informatics Usability*, USA: Cambridge University Press, 1991, pp. 21–37.
- [392] W. W. Agresti and W. M. Evanco, "Projecting software defects from analyzing Ada designs," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 988–997, Nov. 1992, doi: 10.1109/32.177368.
- [393] T. M. Khoshgoftaar, A. S. Pandya, and H. B. More, "A neural network approach for predicting software development faults," in *[1992] Proceedings Third International Symposium on Software Reliability Engineering*, Oct. 1992, pp. 83–89. doi: 10.1109/ISSRE.1992.285855.
- [394] L. C. Briand, V. R. Basili, and W. M. Thomas, "A pattern recognition approach for software engineering data analysis," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 931–942, Nov. 1992, doi: 10.1109/32.177363.
- [395] A. Gillies, "Modelling software quality in the commercial environment," *Software Quality Journal*, vol. 1, no. 3, pp. 175–191, Sep. 1992, doi: 10.1007/BF01720924.
- [396] S. G. Eick, C. R. Loader, M. D. Long, L. G. Votta, and S. Vander Wiel, "Estimating Software Fault Content before Coding," in *Proceedings of the 14th International Conference on Software Engineering*, New York, NY, USA, 1992, pp. 59–65. doi: 10.1145/143062.143090.
- [397] P. Oman and J. Hagemester, "Metrics for assessing a software system's maintainability," in *Proceedings Conference on Software Maintenance 1992*, Nov. 1992, pp. 337–344. doi: 10.1109/ICSM.1992.242525.

- [398] N. Karunanithi, D. Whitley, and Y. K. Malaiya, "Using neural networks in reliability prediction," *IEEE Software*, vol. 9, no. 4, pp. 53–59, Jul. 1992, doi: 10.1109/52.143107.
- [399] S. G. Eick, C. R. Loader, S. A. Vander Wiel, and L. G. Votta, "How many errors remain in a software design document after inspection?," *Proceedings of the 25th Symposium on the Interface*, pp. 195–202, 1993.
- [400] W. M. Zage and D. M. Zage, "Evaluating design metrics on large-scale software," *IEEE Software*, vol. 10, no. 4, pp. 75–81, Jul. 1993, doi: 10.1109/52.219620.
- [401] T. Davis, "The reuse capability model: a basis for improving an organization's reuse capability," in *[1993] Proceedings Advances in Software Reuse*, Mar. 1993, pp. 126–133. doi: 10.1109/ASR.1993.291710.
- [402] A. Dorling, "SPICE: Software process improvement and capability dEtermination," *Information and Software Technology*, vol. 35, no. 6, pp. 404–406, Jun. 1993, doi: 10.1016/0950-5849(93)90011-Q.
- [403] L. C. Briand, V. R. Brasili, and C. J. Hetmanski, "Developing interpretable models with optimized set reduction for identifying high-risk software components," *IEEE Transactions on Software Engineering*, vol. 19, no. 11, pp. 1028–1044, Nov. 1993, doi: 10.1109/32.256851.
- [404] "IEEE Standard for Software Maintenance," *IEEE Std 1219-1993*, pp. 1–45, Jun. 1993, doi: 10.1109/IEEESTD.1993.115570.
- [405] "Bootstrap: Europe's assessment method," *IEEE Software*, vol. 10, no. 3, pp. 93–95, May 1993, doi: 10.1109/52.210613.
- [406] J. Nielsen, *Usability engineering*. Morgan Kaufmann, 1993.
- [407] J. Preece, D. Benyon, G. Davies, L. Keller, and Y. Rogers, *A guide to usability: Human factors in computing*. Addison-Wesley Longman Publishing Co., Inc., 1993.
- [408] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," *Journal of Systems and Software*, vol. 23, no. 2, pp. 111–122, Nov. 1993, doi: 10.1016/0164-1212(93)90077-B.
- [409] T. M. Khoshgoftaar, D. L. Lanning, and A. S. Pandya, "A comparative study of pattern recognition techniques for quality evaluation of telecommunications software," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 2, pp. 279–291, Feb. 1994, doi: 10.1109/49.272878.
- [410] European Foundation for Quality Management (EFQM), "The European Foundation for Quality Management, Guidelines for Identifying and Addressing Business Excellence Issues," presented at the The European Foundation for Quality Management, Brussels, Belgium, 1994.
- [411] W. M. Evanco and W. W. Agresti, "A composite complexity approach for software defect modelling," *Software Quality Journal*, vol. 3, no. 1, pp. 27–44, Mar. 1994, doi: 10.1007/BF00426946.
- [412] W. J. Kettinger and C. C. Lee, "Perceived Service Quality and User Satisfaction with the Information Services Function*," *Decision Sciences*, vol. 25, no. 5-6, pp. 737–766, Sep. 1994, doi: 10.1111/j.1540-5915.1994.tb01868.x.
- [413] S. H. Kan, V. R. Basili, and L. N. Shapiro, "Software quality: An overview from the perspective of total quality management," *IBM Systems Journal*, vol. 33, no. 1, pp. 4–19, 1994, doi: 10.1147/sj.331.0004.
- [414] D. Coleman, D. Ash, B. Lowther, and P. Oman, "Using metrics to evaluate software system maintainability," *Computer*, vol. 27, no. 8, pp. 44–49, Aug. 1994, doi: 10.1109/2.303623.
- [415] O. I. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling," *IEEE Software*, vol. 11, no. 2, pp. 42–49, Mar. 1994, doi: 10.1109/52.268955.
- [416] P. Zeephongsekul, G. Xia, and S. Kumar, "Software-reliability growth model: primary-failures generate secondary-faults under imperfect debugging," *IEEE Transactions on Reliability*, vol. 43, no. 3, pp. 408–413, Sep. 1994, doi: 10.1109/24.326435.
- [417] T. M. Khoshgoftaar and D. L. Lanning, "A neural network approach for early detection of program modules having high risk in the maintenance phase," *Journal of Systems and Software*, vol. 29, pp. 85–91, 1995.
- [418] N. F. Schneidewind, "Software metrics validation: Space Shuttle flight software example," *Annals of Software Engineering*, vol. 1, no. 1, pp. 287–309, Dec. 1995, doi: 10.1007/BF02249054.

References

- [419] J. Troster and J. Tian, "Measurement and defect modeling for a legacy software system," *Annals of Software Engineering*, vol. 1, no. 1, pp. 95–118, Dec. 1995, doi: 10.1007/BF02249047.
- [420] N. Bevan, "Measuring usability as quality of use," *Software Quality Journal*, vol. 4, no. 2, pp. 115–130, Jun. 1995, doi: 10.1007/BF00402715.
- [421] C. Wohlin, P. Runeson, and J. Brantestam, "An experimental evaluation of capture-recapture in software inspections," *Software Testing, Verification and Reliability*, vol. 5, no. 4, pp. 213–232, Jan. 1995, doi: 10.1002/stvr.4370050403.
- [422] A. April and F. Coallier, "Trillium: a model for the assessment of telecom software system development and maintenance capability," in *Proceedings of Software Engineering Standards Symposium*, Aug. 1995, pp. 175–183. doi: 10.1109/SESS.1995.525963.
- [423] J. P. Hudepohl, S. J. Aud, T. M. Khoshgoftaar, E. B. Allen, and J. Mayrand, "Emerald: software metrics and models on the desktop," *IEEE Software*, vol. 13, no. 5, pp. 56–60, Sep. 1996, doi: 10.1109/52.536459.
- [424] M. Azuma, "Software products evaluation system: quality models, metrics and processes—International Standards and Japanese practice," *Information and Software Technology*, vol. 38, no. 3, pp. 145–154, Mar. 1996, doi: 10.1016/0950-5849(95)01069-6.
- [425] T. M. Khoshgoftaar, E. B. Allen, K. S. Kalaichelvan, and N. Goel, "Early quality prediction: a case study in telecommunications," *IEEE Software*, vol. 13, no. 1, pp. 65–71, Jan. 1996, doi: 10.1109/52.476287.
- [426] C. Ebert, "Fuzzy classification for software criticality analysis," *EXPERT SYSTEMS WITH APPLICATIONS*, vol. 11, no. 3, pp. 323–342, 1996, doi: 10.1016/S0957-4174(96)00048-6.
- [427] J. J. Dujmovic, "A method for evaluation and selection of complex hardware and software systems," in *CMG 96 Proceedings*, 1996, pp. 368–378.
- [428] M. D. Levi and F. G. Conrad, "A Heuristic Evaluation of a World Wide Web Prototype," *Interactions*, vol. 3, no. 4, pp. 50–61, Jul. 1996, doi: 10.1145/234813.234819.
- [429] R. A. Martin, L. H. Shafer, and M. H. Auditorium, "Providing a framework for effective software quality assessment," *The MITRE Corporation*, 1996.
- [430] Software Analysis Team at Headquarters (HQ) AFOTEC, "Software Maintainability Evaluation Guide," Department of the Air Force, HQ Air Force Operational Test and Evaluation Center, Kirtland AFB, New Mexico, AFOTEC Pamphlet 99-102, Vol. 3, Sep. 1996. [Online]. Available: https://archive.org/details/DTIC_ADA324619/page/n3/mode/2up
- [431] R. E. Park, W. B. Goethert, and W. A. Florac, "Goal-Driven Software Measurement. A Guidebook.," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, Handbook CMU/SEI-96-HB-002, 1996. [Online]. Available: <https://apps.dtic.mil/dtic/tr/fulltext/u2/a313946.pdf>
- [432] F. Brito e Abreu and W. Melo, "Evaluating the impact of object-oriented design on software quality," in *Proceedings of the 3rd International Software Metrics Symposium*, Mar. 1996, pp. 90–99. doi: 10.1109/METRIC.1996.492446.
- [433] R. Harrison, L. G. Samaraweera, M. R. Dobie, and P. H. Lewis, "An evaluation of code metrics for object-oriented programs," *Information and Software Technology*, vol. 38, no. 7, pp. 443–450, Jan. 1996, doi: 10.1016/0950-5849(95)01081-5.
- [434] S. S. Gokhale and M. R. Lyu, "Regression tree modeling for the prediction of software quality," in *proceedings of the Third ISSAT International Conference on Reliability and Quality in Design*, 1997, pp. 31–36.
- [435] R. Takahashi, Y. Muraoka, and Y. Nakamura, "Building software quality classification trees: approach, experimentation, evaluation," in *Proceedings The Eighth International Symposium on Software Reliability Engineering*, Nov. 1997, pp. 222–233. doi: 10.1109/ISSRE.1997.630869.
- [436] N. F. Schneidewind, "Software metrics model for integrating quality control and prediction," in *Proceedings The Eighth International Symposium on Software Reliability Engineering*, Nov. 1997, pp. 402–415. doi: 10.1109/ISSRE.1997.630888.
- [437] R. V. Small, "Assessing the Motivational Quality of World Wide Websites," ERIC Clearinghouse on Information & Technology, Syracuse, NY, USA, Document available only on microfiche ED 407930, May 1997.

- [438] T. M. Khoshgoftaar, E. B. Allen, R. Halstead, G. P. Trio, and R. Flass, "Process measures for predicting software quality," in *Proceedings 1997 High-Assurance Engineering Workshop*, Aug. 1997, pp. 155–160. doi: 10.1109/HASE.1997.648056.
- [439] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, "Automated Analysis of Requirement Specifications," in *Proceedings of the 19th International Conference on Software Engineering*, New York, NY, USA, 1997, pp. 161–171. doi: 10.1145/253228.253258.
- [440] T. M. Khoshgoftaar *et al.*, "Predicting fault-prone modules with case-based reasoning," in *Proceedings The Eighth International Symposium on Software Reliability Engineering*, Nov. 1997, pp. 27–35. doi: 10.1109/ISSRE.1997.630845.
- [441] N. B. Ebrahimi, "On the statistical analysis of the number of errors remaining in a software design document after inspection," *IEEE Transactions on Software Engineering*, vol. 23, no. 8, pp. 529–532, Aug. 1997, doi: 10.1109/32.624308.
- [442] K. D. Welker, P. W. Oman, and G. G. Atkinson, "Development and Application of an Automated Source Code Maintainability Index," *Journal of Software Maintenance: Research and Practice*, vol. 9, no. 3, pp. 127–159, 1997, doi: [https://doi.org/10.1002/\(SICI\)1096-908X\(199705\)9:3<127::AID-SMR149>3.0.CO;2-S](https://doi.org/10.1002/(SICI)1096-908X(199705)9:3<127::AID-SMR149>3.0.CO;2-S).
- [443] T. M. Khoshgoftaar, E. B. Allen, J. P. Hudepohl, and S. J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system," *IEEE Transactions on Neural Networks*, vol. 8, no. 4, pp. 902–909, Jul. 1997, doi: 10.1109/72.595888.
- [444] B. Sabata, S. Chatterjee, M. Davis, J. J. Sydir, and T. F. Lawrence, "Taxonomy for QoS specifications," in *Proceedings Third International Workshop on Object-Oriented Real-Time Dependable Systems*, Feb. 1997, pp. 100–107. doi: 10.1109/WORDS.1997.609931.
- [445] T. M. Khoshgoftaar and E. B. Allen, "Classification of Fault-Prone Software Modules: Prior Probabilities, Costs, and Model Evaluation," *Empirical Software Engineering*, vol. 3, no. 3, pp. 275–298, Sep. 1998, doi: 10.1023/A:1009736205722.
- [446] N. F. Schneidewind, "An integrated process and product model," in *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262)*, Nov. 1998, pp. 224–234. doi: 10.1109/METRIC.1998.731249.
- [447] J. Kirakowski and B. Cierlik, "Measuring the Usability of Web Sites," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 42, no. 4, pp. 424–428, Oct. 1998, doi: 10.1177/154193129804200405.
- [448] N. Ohlsson, M. Zhao, and M. Helander, "Application of multivariate analysis for software fault prediction," *Software Quality Journal*, vol. 7, no. 1, pp. 51–66, Mar. 1998, doi: 10.1023/B:SQJO.0000042059.16470.f0.
- [449] L. C. Briand, K. E. Emam, and B. G. Freimut, "A comparison and integration of capture-recapture models and the detection profile method," in *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No.98TB100257)*, Nov. 1998, pp. 32–41. doi: 10.1109/ISSRE.1998.730766.
- [450] P. Runeson and C. Wohlin, "An Experimental Evaluation of an Experience-Based Capture-Recapture Method in Software Code Inspections," *Empirical Software Engineering*, vol. 3, no. 4, pp. 381–406, Dec. 1998, doi: 10.1023/A:1009728205264.
- [451] A. Dix, J. Finlay, G. D. Abowd, and R. Beale, *Human-computer interaction, 2nd Edition*. Prentice Hall, Inc., 1998.
- [452] W. Pedrycz, J. F. Peters, and S. Ramanna, "Software quality measurement: concepts and fuzzy neural relational model," in *1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36228)*, May 1998, vol. 2, pp. 1026–1031 vol.2. doi: 10.1109/FUZZY.1998.686259.
- [453] E. Veenendaal, "Questionnaire based usability testing," *Proceedings of European Software Quality Week*, pp. 1–9, Nov. 1998.
- [454] W. D. Jones, J. P. Hudepohl, T. M. Khoshgoftaar, and E. B. Allen, "Application of a usage profile in software quality models," in *Proceedings of the Third European Conference on Software Maintenance and Reengineering (Cat. No. PR00090)*, Mar. 1999, pp. 148–157. doi: 10.1109/CSMR.1999.756692.

References

- [455] T. M. Khoshgoftaar and E. B. Allen, "Logistic Regression Modeling of Software Quality," *Int. J. Rel. Qual. Saf. Eng.*, vol. 06, no. 04, pp. 303–317, Dec. 1999, doi: 10.1142/S0218539399000292.
- [456] T. M. Khoshgoftaar, E. B. Allen, W. D. Jones, and J. P. Hudepohl, "Data Mining For Predictors of Software Quality," *Int. J. Soft. Eng. Knowl. Eng.*, vol. 09, no. 05, pp. 547–563, Oct. 1999, doi: 10.1142/S0218194099000309.
- [457] T. M. Khoshgoftaar, E. B. Allen, X. Yuan, W. D. Jones, and J. P. Huderpohl, "Preparing measurements of legacy software for predicting operational faults," in *Proceedings IEEE International Conference on Software Maintenance - 1999 (ICSM'99). "Software Maintenance for Business Change" (Cat. No.99CB36360)*, Sep. 1999, pp. 359–368. doi: 10.1109/ICSM.1999.792634.
- [458] D. Gehrke and E. Turban, "Determinants of successful Website design: relative importance and recommendations for effectiveness," in *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, Jan. 1999, vol. Track5, p. 8 pp. doi: 10.1109/HICSS.1999.772943.
- [459] S. Chulani and B. Boehm, "Modeling software defect introduction and removal: COQUALMO (CONstructive QUALity MOdel).," presented at the USC-CSE, 1999.
- [460] J. Voas, "Certification: reducing the hidden costs of poor quality," *IEEE Software*, vol. 16, no. 4, pp. 22–25, Aug. 1999, doi: 10.1109/MS.1999.776944.
- [461] H. Petersson and C. Wohlin, "An empirical study of experience-based software defect content estimation methods," in *Proceedings 10th International Symposium on Software Reliability Engineering (Cat. No.PR00443)*, Nov. 1999, pp. 126–135. doi: 10.1109/ISSRE.1999.809317.
- [462] S. Benlarbi and W. L. Melo, "Polymorphism Measures for Early Risk Prediction," in *Proceedings of the 21st International Conference on Software Engineering*, New York, NY, USA, 1999, pp. 334–344. doi: 10.1145/302405.302652.
- [463] K. El Emam, S. Benlarbi, N. Goel, and S. Rai, "A validation of object-oriented metrics," Citeseer, Technical Report ERB-1063, NRG., 1999.
- [464] Mei-Huei Tang, Ming-Hung Kao, and Mei-Hwa Chen, "An empirical study on object-oriented metrics," in *Proceedings Sixth International Software Metrics Symposium (Cat. No.PR00403)*, Nov. 1999, pp. 242–249. doi: 10.1109/METRIC.1999.809745.
- [465] T. M. Khoshgoftaar, E. B. Allen, W. D. Jones, and J. P. Hudepohl, "Accuracy of software quality models over multiple releases," *Annals of Software Engineering*, vol. 9, no. 1, pp. 103–116, May 2000, doi: 10.1023/A:1018972607783.
- [466] T. Khoshgoftaar and E. Allen, "Improving tree-based models of software quality with principal components analysis," in *11TH INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING, PROCEEDINGS*, 2000, pp. 198–209. doi: 10.1109/ISSRE.2000.885872.
- [467] P. Zhang, R. V. Small, G. M. V. Dran, and S. Barcellos, "A two factor theory for Website design," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Jan. 2000, p. 10 pp. vol.1. doi: 10.1109/HICSS.2000.926847.
- [468] T. Khoshgoftaar and E. Allen, "A practical classification-rule for software-quality models," *IEEE TRANSACTIONS ON RELIABILITY*, vol. 49, no. 2, pp. 209–216, Jun. 2000, doi: 10.1109/24.877340.
- [469] H. Petersson and C. Wohlin, "Evaluating defect content estimation rules in software inspections," 2000.
- [470] T. Thelin and P. Runeson, "Robust estimations of fault content with capture–recapture and detection profile estimators," *Journal of Systems and Software*, vol. 52, no. 2, pp. 139–148, Jun. 2000, doi: 10.1016/S0164-1212(99)00140-5.
- [471] L. C. Briand, J. Wüst, J. W. Daly, and D. V. Porter, "Exploring the relationships between design measures and software quality in object-oriented systems," *Journal of Systems and Software*, vol. 51, no. 3, pp. 245–273, 2000, doi: [https://doi.org/10.1016/S0164-1212\(99\)00102-8](https://doi.org/10.1016/S0164-1212(99)00102-8).
- [472] S. Barnes and R. Vidgen, "WebQual: An Exploration of Website Quality," presented at the 8th European Conf. on Information Systems, Jul. 2000.

- [473] S. Muthanna, K. Kontogiannis, K. Ponnambalam, and B. Stacey, "A maintainability model for industrial software systems using design level metrics," in *Proceedings Seventh Working Conference on Reverse Engineering*, Nov. 2000, pp. 248–256. doi: 10.1109/WCRE.2000.891476.
- [474] N. E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software Engineering*, vol. 26, no. 8, pp. 797–814, Aug. 2000, doi: 10.1109/32.879815.
- [475] X. Yuan, T. M. Khoshgoftaar, E. B. Allen, and K. Ganesan, "An application of fuzzy clustering to software quality prediction," in *Proceedings 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, Mar. 2000, pp. 85–90. doi: 10.1109/ASSET.2000.888052.
- [476] Ping Guo and M. R. Lyu, "Software quality prediction using mixture models with EM algorithm," in *Proceedings First Asia-Pacific Conference on Quality Software*, Oct. 2000, pp. 69–78. doi: 10.1109/APAQ.2000.883780.
- [477] S. Benlarbi, K. El Emam, N. Goel, and S. Rai, "Thresholds for object-oriented measures," in *Proceedings 11th International Symposium on Software Reliability Engineering. ISSRE 2000*, Oct. 2000, pp. 24–38. doi: 10.1109/ISSRE.2000.885858.
- [478] D. Glasberg, K. El Emam, W. Melo, and N. Madhavji, "Validating object-oriented design metrics on a commercial java application," Citeseer, TR ERB-1080, NRC, 2000.
- [479] Y. Liu and T. M. Khoshgoftaar, "Genetic programming model for software quality classification," in *Proceedings Sixth IEEE International Symposium on High Assurance Systems Engineering. Special Topic: Impact of Networking*, Oct. 2001, pp. 127–136. doi: 10.1109/HASE.2001.966814.
- [480] O. Balci, "A Methodology for Certification of Modeling and Simulation Applications," *ACM Trans. Model. Comput. Simul.*, vol. 11, no. 4, pp. 352–377, Oct. 2001, doi: 10.1145/508366.508369.
- [481] H. K. N. Leung, "Quality metrics for intranet applications," *Information & Management*, vol. 38, no. 3, pp. 137–152, Jan. 2001, doi: 10.1016/S0378-7206(00)00060-4.
- [482] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "An automatic quality evaluation for natural language requirements," in *Proceedings of the Seventh International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ*, 2001, vol. 1, pp. 4–5.
- [483] F. Losavio, L. Chirinos, and M. Pérez, "Attribute-Based Techniques to Evaluate Architectural Styles for Interactive Systems," *Journal of Object Oriented Programming*, pp. 130–138, 2001.
- [484] M. Kajko-Mattsson, S. Forssander, and U. Olsson, "Corrective maintenance maturity model (CM/sup 3/): maintainer's education and training," in *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, May 2001, pp. 610–619. doi: 10.1109/ICSE.2001.919135.
- [485] S. Biffel and W. Grossmann, "Evaluating the accuracy of defect estimation models based on inspection data from two inspection cycles," in *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, May 2001, pp. 145–154. doi: 10.1109/ICSE.2001.919089.
- [486] L. Olsina, G. Lafuente, and G. Rossi, "Specifying Quality Characteristics and Attributes for Websites," in *Web Engineering: Managing Diversity and Complexity of Web Application Development*, S. Murugesan and Y. Deshpande, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 266–278. doi: 10.1007/3-540-45144-7_26.
- [487] K. El Emam, W. Melo, and J. C. Machado, "The prediction of faulty classes using object-oriented design metrics," *Journal of Systems and Software*, vol. 56, no. 1, pp. 63–75, Feb. 2001, doi: 10.1016/S0164-1212(00)00086-8.
- [488] L. C. Briand, J. Wüst, and H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs," *Empirical Software Engineering*, vol. 6, no. 1, pp. 11–58, Mar. 2001, doi: 10.1023/A:1009815306478.
- [489] K. El Emam, S. Benlarbi, N. Goel, and S. N. Rai, "The confounding effect of class size on the validity of object-oriented metrics," *IEEE Transactions on Software Engineering*, vol. 27, no. 7, pp. 630–650, Jul. 2001, doi: 10.1109/32.935855.
- [490] T. M. Khoshgoftaar, E. B. Allen, and J. Deng, "Using regression trees to classify fault-prone software modules," *IEEE Transactions on Reliability*, vol. 51, no. 4, pp. 455–462, Dec. 2002, doi: 10.1109/TR.2002.804488.

- [491] N. J. Pizzi, A. R. Summers, and W. Pedrycz, "Software quality prediction using median-adjusted class labels," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, May 2002, vol. 3, pp. 2405–2409 vol.3. doi: 10.1109/IJCNN.2002.1007518.
- [492] L. C. Briand, W. L. Melo, and J. Wust, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 706–720, Jul. 2002, doi: 10.1109/TSE.2002.1019484.
- [493] S. S. So, S. D. Cha, and Y. R. Kwon, "Empirical evaluation of a fuzzy logic-based software quality prediction model," *Fuzzy Sets and Systems*, vol. 127, no. 2, pp. 199–208, Apr. 2002, doi: 10.1016/S0165-0114(01)00128-2.
- [494] R. Ramler, E. Weippl, M. Winterer, W. Schwinger, and J. Altmann, "A quality-driven approach to web testing," in *Iberoamerican Conference on Web Engineering, ICWE, 2002*, vol. 2, pp. 81–95.
- [495] T. M. Khoshgoftaar and N. Seliya, "Tree-based software quality estimation models for fault prediction," in *Proceedings Eighth IEEE Symposium on Software Metrics*, Jun. 2002, pp. 203–214. doi: 10.1109/METRIC.2002.1011339.
- [496] J. Miller, F. Macdonald, and J. Ferguson, "ASSISTing Management Decisions in the Software Inspection Process," *Information Technology and Management*, vol. 3, no. 1, pp. 67–83, Jan. 2002, doi: 10.1023/A:1013112826330.
- [497] F. Padberg, "Empirical Interval Estimates for the Defect Content after an Inspection," in *Proceedings of the 24th International Conference on Software Engineering*, New York, NY, USA, 2002, pp. 58–68. doi: 10.1145/581339.581350.
- [498] M. Bertoa and A. Vallecillo, "Quality Attributes for COTS Components," Spain, 2002.
- [499] A. B. Albuquerque and A. D. Belchior, "E-commerce websites: a qualitative evaluation," *11th international WWW conference proceedings - Poster Session. Hawaii*, pp. 294–300, 2002.
- [500] P. Schubert, "Extended web assessment method (EWAM): evaluation of electronic commerce applications from the customer's viewpoint," *International Journal of Electronic Commerce*, vol. 7, no. 2, pp. 51–80, 2002.
- [501] A. Mani, "Understanding quality of service for Web services," <http://www-106.ibm.com/developerworks/library/ws-quality.html>, 2002.
- [502] K. El Emam, S. Benlarbi, N. Goel, W. Melo, H. Lounis, and S. N. Rai, "The optimal class size for object-oriented software," *IEEE Transactions on Software Engineering*, vol. 28, no. 5, pp. 494–509, May 2002, doi: 10.1109/TSE.2002.1000452.
- [503] T. M. Khoshgoftaar, N. Seliya, and Y. Liu, "Genetic programming-based decision trees for software quality classification," in *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, Nov. 2003, pp. 374–383. doi: 10.1109/TAI.2003.1250214.
- [504] R. Zeineddine and N. Mansour, "Software Quality Improvement Model for Small Organizations," in *Computer and Information Sciences - ISCIS 2003*, Berlin, Heidelberg, 2003, pp. 1027–1034.
- [505] E. M. Herrera and R. A. T. Ramírez, "A Methodology for Self-Diagnosis for Software Quality Assurance in Small and Medium-Sized Industries in Latin America," *The Electronic Journal of Information Systems in Developing Countries*, vol. 15, no. 1, pp. 1–13, Oct. 2003, doi: 10.1002/j.1681-4835.2003.tb00100.x.
- [506] T.-S. Quah and M. M. T. Thwin, "Application of neural networks for software quality prediction using object-oriented metrics," in *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.*, Sep. 2003, pp. 116–125. doi: 10.1109/ICSM.2003.1235412.
- [507] J. Ruiz, Coral Calero, and M. Piattini, "A Three Dimensional Web Quality Model," in *Web Engineering*, Berlin, Heidelberg, 2003, pp. 384–385.
- [508] M. Ortega, M. Pérez, and T. Rojas, "Construction of a Systemic Quality Model for Evaluating a Software Product," *Software Quality Journal*, vol. 11, no. 3, pp. 219–242, Jul. 2003, doi: 10.1023/A:1025166710988.
- [509] R. P. S. Simão and A. D. Belchior, "Quality Characteristics for Software Components: Hierarchy and Quality Guides," *Component-Based Software Quality. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg.*, vol. 2693, pp. 184–206, 2003, doi: https://doi.org/10.1007/978-3-540-45064-1_9.

- [510] F.-W. Duijnhouwer and C. Widdows, "Capgemini Expert Letter Open Source Maturity Model," *Capgemini Expert Letter*, pp. 1–18, 2003.
- [511] A. Stefani, M. Xenos, and D. Stavrinoudis, "Modelling e-commerce systems' quality with belief networks," in *IEEE International Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2003. VECIMS '03. 2003*, Jul. 2003, pp. 13–18. doi: 10.1109/VECIMS.2003.1227023.
- [512] S. Golubic, "On software quality verification in the object-oriented development environment," in *Proceedings of the 7th International Conference on Telecommunications, 2003. ConTEL 2003.*, Jun. 2003, vol. 2, pp. 557–563 vol.2. doi: 10.1109/CONTEL.2003.176961.
- [513] L. Mich, M. Franch, and L. Gaio, "Evaluating and designing the quality of web sites," *IEEE MultiMedia*, vol. 10, no. 1, pp. 34–43, 2003.
- [514] H. Zhang, S. Jarzabek, and B. Yang, "Quality Prediction and Assessment for Product Lines," in *Advanced Information Systems Engineering*, Berlin, Heidelberg, 2003, pp. 681–695.
- [515] T. M. Khoshgoftaar and N. Seliya, "Analogy-Based Practical Classification Rules for Software Quality Estimation," *Empirical Software Engineering*, vol. 8, no. 4, pp. 325–350, Dec. 2003, doi: 10.1023/A:1025316301168.
- [516] A. Abran, A. Khelifi, W. Suryan, and A. Seffah, "Usability Meanings and Interpretations in ISO Standards," *Software Quality Journal*, vol. 11, no. 4, pp. 325–338, Nov. 2003, doi: 10.1023/A:1025869312943.
- [517] S. Ran, "A Model for Web Services Discovery with QoS," *SIGecom Exch.*, vol. 4, no. 1, pp. 1–10, Mar. 2003, doi: 10.1145/844357.844360.
- [518] K. Lee, J. Jeon, W. Lee, S.-H. Jeong, and S.-W. Park, "Qos for web services: Requirements and possible approaches," *W3C working group note*, vol. 25, no. 3, p. 119, 2003.
- [519] C. Patel, K. Supekar, and Y. Lee, "A QoS oriented framework for adaptive management of web service based workflows," in *International Conference on Database and Expert Systems Applications, 2003*, pp. 826–835. doi: 10.1007/978-3-540-45227-0_80.
- [520] M. MORISIO, I. STAMELOS, and A. TSOUKIAS, "SOFTWARE PRODUCT AND PROCESS ASSESSMENT THROUGH PROFILE-BASED EVALUATION," *Int. J. Soft. Eng. Knowl. Eng.*, vol. 13, no. 05, pp. 495–512, Oct. 2003, doi: 10.1142/S0218194003001433.
- [521] L. Guo, Y. Ma, B. Cukic, and Harshinder Singh, "Robust prediction of fault-proneness by random forests," in *15th International Symposium on Software Reliability Engineering*, Nov. 2004, pp. 417–428. doi: 10.1109/ISSRE.2004.35.
- [522] Navica Inc., "The Open Source Maturity Model is a vital tool for planning open source success," 2004. <http://www.navicasoft.com/pages/osmm.htm>
- [523] Atos Origin, "Method for Qualification and Selection of Open Source Software (QSOS)," 2004. <http://www.qsos.org/> (accessed Jan. 23, 2021).
- [524] K. Khosravi and Y.-G. Guéhéneuc, "A quality model for design patterns," University of Montreal, Technical report 1249, Sep. 2004.
- [525] G. A. Di Lucca, A. R. Fasolino, P. Tramontana, and C. A. Visaggio, "Towards the definition of a maintainability model for Web applications," in *Eighth European Conference on Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings.*, Mar. 2004, pp. 279–287. doi: 10.1109/CSMR.2004.1281430.
- [526] Webb Harold W. and Webb Linda A., "SiteQual: an integrated measure of Web site quality," *Journal of Enterprise Information Management*, vol. 17, no. 6, pp. 430–440, Jan. 2004, doi: 10.1108/17410390410566724.
- [527] E. M. Maximilien and M. P. Singh, "A framework and ontology for dynamic Web services selection," *IEEE Internet Computing*, vol. 8, no. 5, pp. 84–93, Oct. 2004, doi: 10.1109/MIC.2004.27.
- [528] A. Avizienis, J. -. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, Mar. 2004, doi: 10.1109/TDSC.2004.2.

- [529] C. Patel, K. Supekar, and Yugyung Lee, "Provisioning resilient, adaptive Web services-based workflow: a semantic modeling approach," in *Proceedings. IEEE International Conference on Web Services, 2004.*, Jul. 2004, pp. 480–487. doi: 10.1109/ICWS.2004.1314773.
- [530] N. Looker, M. Munro, and J. Xu, "Assessing web service quality of service with fault injection," 2004.
- [531] R. Marinescu and D. Ratiu, "Quantifying the quality of object-oriented design: the factor-strategy model," in *11th Working Conference on Reverse Engineering*, Nov. 2004, pp. 192–201. doi: 10.1109/WCRE.2004.31.
- [532] B. B. Chua and L. E. Dyson, "Applying the ISO 9126 model to the evaluation of an e-learning system," in *Proc. of ASCILITE, 2004*, pp. 5–8.
- [533] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, "Non-functional requirements in industry - three case studies adopting an experience-based NFR method," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, Sep. 2005, pp. 373–382. doi: 10.1109/RE.2005.47.
- [534] B. Freimut, C. Denger, and M. Ketterer, "An industrial case study of implementing and validating defect classification for process improvement and quality management," in *11th IEEE International Software Metrics Symposium (METRICS'05)*, Sep. 2005, pp. 10 pp. – 19. doi: 10.1109/METRICS.2005.10.
- [535] Kilsup Lee and Sung Jong Lee, "A quantitative software quality evaluation model for the artifacts of component based development," in *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, May 2005, pp. 20–25. doi: 10.1109/SNPD-SAWN.2005.7.
- [536] M. M. T. Thwin and T.-S. Quah, "Application of neural networks for software quality prediction using object-oriented metrics," *Journal of Systems and Software*, vol. 76, no. 2, pp. 147–156, 2005, doi: <https://doi.org/10.1016/j.jss.2004.05.001>.
- [537] A. Wasserman, M. Pal, and C. Chan, "Business Readiness Rating Project," *BRR Whitepaper 2005 RFC 1*, pp. 1–22, 2005.
- [538] O. Signore, "A comprehensive model for Web sites quality," in *Seventh IEEE International Symposium on Web Site Evolution*, Sep. 2005, pp. 30–36. doi: 10.1109/WSE.2005.1.
- [539] A. Alvaro, E. Almeida, and S. Meira, "Quality attributes for a component quality model," *10th WCOP/19th ECCOP, Glasgow, Scotland*, pp. 31–37, 2005.
- [540] S. He, E. Qi, Z. He, and B. Nie, "A study on quality controlling of semiconductor assembly based on principal component analysis," in *PROCEEDINGS OF THE 12TH INTERNATIONAL CONFERENCE ON INDUSTRIAL ENGINEERING AND ENGINEERING MANAGEMENT, VOLS 1 AND 2: MODERN INDUSTRIAL ENGINEERING AND INNOVATION IN ENTERPRISE MANAGEMENT*, 2005, pp. 418–420.
- [541] A. Parasuraman, V. A. Zeithaml, and A. Malhotra, "E-S-QUAL: A Multiple-Item Scale for Assessing Electronic Service Quality," *Journal of Service Research*, vol. 7, no. 3, pp. 123–233, Feb. 2005, doi: 10.1177/1094670504271156.
- [542] G. Dobson, R. Lock, and I. Sommerville, "QoSOnt: a QoS ontology for service-centric systems," in *31st EUROMICRO Conference on Software Engineering and Advanced Applications*, Sep. 2005, pp. 80–87. doi: 10.1109/EUROMICRO.2005.49.
- [543] J. De Bruijn *et al.*, "Web service modeling ontology (wsmo) (working draft version)," *Interface*, vol. 5, no. 1, p. 50, 2005.
- [544] S. Amasaki, Y. Takagi, O. Mizuno, and T. Kikuno, "Constructing a Bayesian belief network to predict final quality in embedded system development," *IEICE Transactions on Information and Systems*, vol. 88, no. 6, pp. 1134–1141, 2005.
- [545] Fei Xing, Ping Guo, and M. R. Lyu, "A novel method for early software quality prediction based on support vector machine," in *16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*, Nov. 2005, pp. 10 pp. – 222. doi: 10.1109/ISSRE.2005.6.
- [546] D. Janakiram and M. S. Rajasree, "ReQuEst: Requirements-Driven Quality Estimator," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 1, p. 4, Jan. 2005, doi: 10.1145/1039174.1039194.

- [547] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Transactions on Software Engineering*, vol. 31, no. 10, pp. 897–910, Oct. 2005, doi: 10.1109/TSE.2005.112.
- [548] C. van Koten and A. R. Gray, "An application of Bayesian network for predicting object-oriented software maintainability," *Information and Software Technology*, vol. 48, no. 1, pp. 59–67, 2006, doi: <https://doi.org/10.1016/j.infsof.2005.03.002>.
- [549] F. J. Miranda, R. Cortés, and C. Barriuso, "Quantitative evaluation of e-banking web sites: An empirical study of Spanish banks.," *Electronic Journal of Information Systems Evaluation*, vol. 9, no. 2, pp. 73–82, 2006.
- [550] J. P. Carvallo and X. Franch, "Extending the ISO/IEC 9126-1 Quality Model with Non-Technical Factors for COTS Components Selection," in *Proceedings of the 2006 International Workshop on Software Quality*, New York, NY, USA, 2006, pp. 9–14. doi: 10.1145/1137702.1137706.
- [551] T. M. Khoshgoftaar, N. Seliya, and N. Sundaresh, "An empirical study of predicting software faults with case-based reasoning," *Software Quality Journal*, vol. 14, no. 2, pp. 85–111, Jun. 2006, doi: 10.1007/s11219-006-7597-z.
- [552] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda, "Usability measurement and metrics: A consolidated model," *Software Quality Journal*, vol. 14, no. 2, pp. 159–178, Jun. 2006, doi: 10.1007/s11219-006-7600-8.
- [553] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, "A qos-aware selection model for semantic web services," in *International Conference on Service-Oriented Computing*, 2006, pp. 390–401. doi: 10.1007/11948148_32.
- [554] S. Jiang and F. A. Aagesen, "An approach to integrated semantic service discovery," in *IFIP TC6 International Conference on Autonomic Networking*, 2006, pp. 159–171.
- [555] Gwyduk Yeom, Taewoong Yun, and Dugki Min, "A QoS model and testing mechanism for quality-driven Web services selection," in *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, Apr. 2006, p. 6 pp. doi: 10.1109/SEUS-WCCIA.2006.34.
- [556] D. T. Tsesmetzis, I. G. Roussaki, I. V. Papaioannou, and M. E. Anagnostou, "QoS awareness support in Web-Service semantics," in *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, Feb. 2006, pp. 128–128. doi: 10.1109/AICT-ICIW.2006.156.
- [557] D. Z. G. Garcia and M. B. F. de Toledo, "Semantics-Enriched QoS Policies for Web Service Interactions," in *Proceedings of the 12th Brazilian Symposium on Multimedia and the Web*, New York, NY, USA, 2006, pp. 35–44. doi: 10.1145/1186595.1186601.
- [558] H. Truong, R. Samborski, and T. Fahringer, "Towards a Framework for Monitoring and Analyzing QoS Metrics of Grid Services," in *2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, Dec. 2006, pp. 65–65. doi: 10.1109/E-SCIENCE.2006.261149.
- [559] Yuming Zhou and Hareton Leung, "Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults," *IEEE Transactions on Software Engineering*, vol. 32, no. 10, pp. 771–789, Oct. 2006, doi: 10.1109/TSE.2006.102.
- [560] K. Kritikos and D. Plexousakis, "Semantic QoS Metric Matching," in *2006 European Conference on Web Services (ECOWS'06)*, Dec. 2006, pp. 265–274. doi: 10.1109/ECOWS.2006.34.
- [561] S. Mavromoustakos and A. S. Andreou, "WAQE: a Web Application Quality Evaluation model," *International Journal of Web Engineering and Technology*, vol. 3, no. 1, pp. 96–120, Dec. 2006, doi: 10.1504/IJWET.2007.011529.
- [562] A. S. Andreou and M. Tziakouris, "A quality framework for developing and evaluating original software components," *Information and Software Technology*, vol. 49, no. 2, pp. 122–141, 2007, doi: <https://doi.org/10.1016/j.infsof.2006.03.007>.
- [563] Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," *Journal of Systems and Software*, vol. 80, no. 8, pp. 1349–1361, 2007, doi: <https://doi.org/10.1016/j.jss.2006.10.049>.
- [564] D. Taibi, L. Lavazza, and S. Morasca, "OpenBQR: a framework for the assessment of OSS," in *Open Source Development, Adoption and Innovation*, Boston, MA, 2007, pp. 173–186.

- [565] Mbusi Sibisi and C. C. van Waveren, "A process framework for customising software quality models," in *AFRICON 2007*, Sep. 2007, pp. 1–8. doi: 10.1109/AFRCON.2007.4401495.
- [566] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J. - Girard, "An Activity-Based Quality Model for Maintainability," in *2007 IEEE International Conference on Software Maintenance*, Oct. 2007, pp. 184–193. doi: 10.1109/ICSM.2007.4362631.
- [567] S. Winter, S. Wagner, and F. Deissenboeck, "A Comprehensive Model of Usability," in *Engineering Interactive Systems*, Berlin, Heidelberg, 2007, pp. 106–122.
- [568] I. Heitlager, T. Kuipers, and J. Visser, "A Practical Model for Measuring Maintainability," in *6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007)*, Sep. 2007, pp. 30–39. doi: 10.1109/QUATIC.2007.8.
- [569] S. Kim, T. Zimmermann, E. J. Whitehead Jr., and A. Zeller, "Predicting Faults from Cached History," in *29th International Conference on Software Engineering (ICSE'07)*, May 2007, pp. 489–498. doi: 10.1109/ICSE.2007.66.
- [570] S. Kanmani, V. R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," *Information and Software Technology*, vol. 49, no. 5, pp. 483–492, May 2007, doi: 10.1016/j.infsof.2006.07.005.
- [571] G. Quirchmayr, S. Funilkul, and W. Chutimaskul, "A quality model of e-government services based on the ISO/IEC 9126 standard," in *International Legal Informatics Symposium (IRIS)*, 2007, pp. 45–53.
- [572] T. S. Dagger, J. C. Sweeney, and L. W. Johnson, "A hierarchical model of health service quality: scale development and investigation of an integrated model," *Journal of Service Research*, vol. 10, no. 2, pp. 123–142, 2007.
- [573] A. Henriksson, Y. Yi, B. Frost, and M. Middleton, "Evaluation instrument for e-government websites," *Electronic Government, an International Journal (EG)*, vol. 4, no. 2, pp. 204–226, 2007.
- [574] K. Ren, J. Chen, T. Chen, J. Song, and N. Xiao, "Grid-Based Semantic Web Service Discovery Model with QoS Constraints," in *Third International Conference on Semantics, Knowledge and Grid (SKG 2007)*, Oct. 2007, pp. 479–482. doi: 10.1109/SKG.2007.118.
- [575] J. De Bruijn *et al.*, "D2v1.4. Web service modeling ontology (wsmo)," *Interface*, p. 35, 2007.
- [576] W. D. Yu, R. B. Radhakrishna, S. Pingali, and V. Kolluri, "Modeling the measurements of QoS requirements in web service systems," *Simulation*, vol. 83, no. 1, pp. 75–91, 2007.
- [577] Y. Kang, "Extended Model Design for Quality Factor Based Web Service Management," in *Future Generation Communication and Networking (FGCN 2007)*, Dec. 2007, vol. 2, pp. 484–487. doi: 10.1109/FGCN.2007.123.
- [578] E. Giallonardo and E. Zimeo, "More Semantics in QoS Matching," in *IEEE International Conference on Service-Oriented Computing and Applications (SOCA '07)*, Jun. 2007, pp. 163–171. doi: 10.1109/SOCA.2007.30.
- [579] Y. Lee and G. Yeom, "A Quality Chain Modeling Methodology for Ternary Web Services Quality View," in *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007)*, Aug. 2007, pp. 91–97. doi: 10.1109/SERA.2007.26.
- [580] W. J. Sung, J. H. Kim, and S. Y. Rhew, "A Quality Model for Open Source Software Selection," in *Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007)*, Aug. 2007, pp. 515–519. doi: 10.1109/ALPIT.2007.81.
- [581] G. J. Pai and J. Bechta Dugan, "Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods," *IEEE Transactions on Software Engineering*, vol. 33, no. 10, pp. 675–686, Oct. 2007, doi: 10.1109/TSE.2007.70722.
- [582] S. Wagner and F. Deissenboeck, "An Integrated Approach to Quality Modelling," in *Fifth International Workshop on Software Quality (WoSQ'07: ICSE Workshops 2007)*, May 2007, pp. 1–1. doi: 10.1109/WOSQ.2007.3.
- [583] H. M. Selim, "Critical success factors for e-learning acceptance: Confirmatory factor models," *Computers & Education*, vol. 49, no. 2, pp. 396–413, Sep. 2007, doi: 10.1016/j.compedu.2005.09.004.

- [584] F. Khomh and Y.-G. Guéhéneuc, "DEQUALITE: Building Design-Based Software Quality Models," New York, NY, USA, 2008. doi: 10.1145/1753196.1753199.
- [585] H. Fang, "Modeling and Analysis for Educational Software Quality Hierarchy Triangle," in *2008 Seventh International Conference on Web-based Learning*, Aug. 2008, pp. 14–18. doi: 10.1109/ICWL.2008.19.
- [586] C.-W. Chang, C.-R. Wu, and H.-L. Lin, "Integrating fuzzy theory and hierarchy concepts to evaluate software quality," *Software Quality Journal*, vol. 16, no. 2, pp. 263–276, Jun. 2008, doi: 10.1007/s11219-007-9035-2.
- [587] A. Sharma, R. Kumar, and P. S. Grover, "Estimation of quality for software components: an empirical approach," *ACM SIGSOFT Software Engineering Notes*, vol. 33, no. 6, pp. 1–10, Nov. 2008.
- [588] A. I. Eldesouky, H. Arafat, and H. Ramzey, "Toward complex academic Web-Sites Quality evaluation method (QEM) framework: quality requirements phase definition and specification," *Computer and Systems Engineering Department*, 2008.
- [589] B. Shim, S. Choue, S. Kim, and S. Park, "A Design Quality Model for Service-Oriented Architecture," in *2008 15th Asia-Pacific Software Engineering Conference*, Dec. 2008, pp. 403–410. doi: 10.1109/APSEC.2008.32.
- [590] CITY *et al.*, "Quality Reference Model for SBA," *Contractual Deliverable #CD-JRA-1.3.2 , S-Cube, the European Network of Excellence in Software Services and Systems*, p. 64, Mar. 2008.
- [591] A. Stefani and M. Xenos, "E-commerce system quality assessment using a model based on ISO 9126 and Belief Networks," *Software Quality Journal*, vol. 16, no. 1, pp. 107–129, Mar. 2008, doi: 10.1007/s11219-007-9032-5.
- [592] P. M. Heck and M. van Eekelen, *LaQuSo software product certification model (LSPCM)*. Technische Universiteit Eindhoven, 2008.
- [593] R. Plösch *et al.*, "The EMISQ method and its tool support-expert-based evaluation of internal software quality," *Innovations in Systems and Software Engineering*, vol. 4, no. 1, pp. 3–15, Apr. 2008, doi: 10.1007/s11334-007-0039-7.
- [594] J. Laval, A. Bergel, and S. Ducasse, "Assessing the Quality of your Software with MoQam," Antwerp, Belgium, Oct. 2008. [Online]. Available: <https://hal.inria.fr/inria-00498482>
- [595] I. Gondra, "Applying machine learning to software fault-proneness prediction," *Journal of Systems and Software*, vol. 81, no. 2, pp. 186–195, Feb. 2008, doi: 10.1016/j.jss.2007.05.035.
- [596] S. Cimino, S. Sperone, and F. Micali, "Web Q-Model: a new approach to the quality," Apr. 2008.
- [597] W. Chutimaskul, S. Funilkul, and V. Chongsuphajaisiddhi, "The Quality Framework of E-Government Development," in *Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance*, New York, NY, USA, 2008, pp. 105–109. doi: 10.1145/1509096.1509117.
- [598] W. Abramowicz, R. Hofman, W. Suryan, and D. Zyskowski, "SQuaRE based web services quality model," in *Proceedings of The International MultiConference of Engineers and Computer Scientists 2008*, 2008, pp. 827–835.
- [599] N. Artaïam and T. Senivongse, "Enhancing Service-Side QoS Monitoring for Web Services," in *2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, Aug. 2008, pp. 765–770. doi: 10.1109/SNPD.2008.157.
- [600] V. X. Tran, "WS-QoSOnto: A QoS Ontology for Web Services," in *2008 IEEE International Symposium on Service-Oriented System Engineering*, Dec. 2008, pp. 233–238. doi: 10.1109/SOSE.2008.17.
- [601] H. Mittal, P. Bhatia, and P. Goswami, "Software quality assessment based on fuzzy logic technique," *International Journal of Software Computing Applications*, vol. 34, no. 3, pp. 105–112, 2008.
- [602] L. Etxeberria and G. Sagardui, "Quality assessment in software product lines," in *International Conference on Software Reuse*, 2008, pp. 178–181.
- [603] L. Etxeberria and G. Sagardui, "Evaluation of Quality Attribute Variability in Software Product Families," in *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008)*, Apr. 2008, pp. 255–264. doi: 10.1109/ECBS.2008.14.

- [604] E. Raffoul, K. Domínguez, M. Pérez, L. E. Mendoza, and A. C. Grimán, "Quality model for the selection of FLOSS-based Issue tracking system," in *Proceedings of the IASTED international conference on software engineering, Innsbruck, Austria, 2008*, vol. 12.
- [605] H. M. Olague, L. H. Etkorn, S. L. Messimer, and H. S. Delugach, "An empirical validation of object-oriented class complexity metrics and their ability to predict error-prone classes in highly iterative, or agile, software: a case study," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 20, no. 3, pp. 171–197, May 2008, doi: 10.1002/smr.366.
- [606] M. Kläs, H. Nakao, F. Elberzhager, and J. Münch, "Support planning and controlling of early quality assurance by combining expert judgment and defect data—a case study," *Empirical Software Engineering*, vol. 15, no. 4, pp. 423–454, Aug. 2010, doi: 10.1007/s10664-009-9112-1.
- [607] A. Bergel *et al.*, "SQUALE – Software QUALity Enhancement," in *2009 13th European Conference on Software Maintenance and Reengineering*, Mar. 2009, pp. 285–288. doi: 10.1109/CSMR.2009.13.
- [608] E. Petrinja, R. Nambakam, and A. Sillitti, "Introducing the OpenSource Maturity Model," in *2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, May 2009, pp. 37–41. doi: 10.1109/FLOSS.2009.5071358.
- [609] B. Behkamal, M. Kahani, and M. K. Akbari, "Customizing ISO 9126 quality model for evaluation of B2B applications," *Information and Software Technology*, vol. 51, no. 3, pp. 599–609, 2009, doi: 10.1016/j.infsof.2008.08.001.
- [610] A. Kumar, P. S. Grover, and R. Kumar, "A quantitative evaluation of aspect-oriented software quality model (AOSQUAMO)," *ACM SIGSOFT Software Engineering Notes*, vol. 34, no. 5, pp. 1–9, Sep. 2009.
- [611] P. R. Srivastava and K. Kumar, "An Approach towards Software Quality Assessment," in *Information Systems, Technology and Management*, Berlin, Heidelberg, 2009, pp. 150–160.
- [612] S. Wagner, "A Bayesian Network Approach to Assess and Predict Software Quality Using Activity-Based Quality Models," Vancouver, BC, Canada, May 2009. doi: 10.1145/1540438.1540447.
- [613] M. Á. Moraga, C. Calero, J. Garzás, and M. Piattini, "Assessment of portlet quality: Collecting real experience," *Computer Standards & Interfaces*, vol. 31, no. 2, pp. 336–347, 2009, doi: <https://doi.org/10.1016/j.csi.2008.05.001>.
- [614] M. Soto and M. Ciolkowski, "The QualOSS open source assessment model measuring the performance of open source communities," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, Oct. 2009, pp. 498–501. doi: 10.1109/ESEM.2009.5314237.
- [615] R. Plösch, H. Gruber, C. Körner, G. Pomberger, and S. Schiffer, "A proposal for a quality model based on a technical topic classification," in *SQMB 2009 Workshop*, 2009, vol. 2.
- [616] A. Hussain and M. Kutar, "Usability metric framework for mobile phone application," 2009.
- [617] V. del Bianco, L. Lavazza, S. Morasca, and D. Taibi, "Quality of Open Source Software: The QualiPSO Trustworthiness Model," *Boldyreff C., Crowston K., Lundell B., Wasserman A.I. (eds) Open Source Ecosystems: Diverse Communities Interacting. OSS 2009. IFIP Advances in Information and Communication Technology*, vol. 299, pp. 199–212, doi: 10.1007/978-3-642-02032-2_18.
- [618] X. Papadomichelaki and G. Mentzas, "A Multiple-Item Scale for Assessing E-Government Service Quality," in *Wimmer M.A., Scholl H.J., Janssen M., Traunmüller R. (eds) Electronic Government. EGOV 2009. Lecture Notes in Computer Science*, 2009, vol. 5693, pp. 163–175. doi: 10.1007/978-3-642-03516-6_14.
- [619] H. M. Frutos, I. Kotsiopoulos, L. M. V. Gonzalez, and L. R. Merino, "Enhancing service selection by semantic qos," in *European Semantic Web Conference*, 2009, pp. 565–577.
- [620] H. Chang and K. Lee, "Quality-Driven Web Service Composition for Ubiquitous Computing Environment," in *2009 International Conference on New Trends in Information and Service Science*, Jul. 2009, pp. 156–161. doi: 10.1109/NISS.2009.117.
- [621] E. Al-Masri and Q. H. Mahmoud, "Understanding web service discovery goals," in *2009 IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2009, pp. 3714–3719. doi: 10.1109/ICSMC.2009.5346882.

- [622] Tong Hongxia, Cao Jian, Zhang ShenSheng, and Mou Yujie, "A fuzzy evaluation system for web services selection using extended QoS model," *Kybernetes*, vol. 38, no. 3/4, pp. 513–521, Jan. 2009, doi: 10.1108/03684920910944236.
- [623] M. Comuzzi and B. Pernici, "A Framework for QoS-Based Web Service Contracting," *ACM Trans. Web*, vol. 3, no. 3, Jul. 2009, doi: 10.1145/1541822.1541825.
- [624] Z. Balfagih and M. F. Hassan, "Quality Model for Web Services from Multi-stakeholders' Perspective," in *2009 International Conference on Information Management and Engineering*, Apr. 2009, pp. 287–291. doi: 10.1109/ICIME.2009.11.
- [625] S. Li and J. Zhou, "The WSMO-QoS Semantic Web Service Discovery Framework," in *2009 International Conference on Computational Intelligence and Software Engineering*, Dec. 2009, pp. 1–5. doi: 10.1109/CISE.2009.5366383.
- [626] K. K. Reddy D, K. Maralla, R. K. G, and M. Thirumaran, "A Greedy Approach with Criteria Factors for QoS Based Web Service Discovery," New York, NY, USA, 2009. doi: 10.1145/1517303.1517317.
- [627] A. Marchetto, "OQMw: An OO Quality Model for Web Applications," *Journal of Applied Science and Engineering*, vol. 12, no. 4, pp. 459–470, Dec. 2009, doi: 10.6180/jase.2009.12.4.10.
- [628] S. Ozkan and R. Koseler, "Multi-dimensional students' evaluation of e-learning systems in the higher education context: An empirical investigation," *Computers & Education*, vol. 53, no. 4, pp. 1285–1296, Dec. 2009, doi: 10.1016/j.compedu.2009.06.011.
- [629] C. Cappiello *et al.*, "A quality model for service monitoring and adaptation," in *Workshop on Service Monitoring, Adaptation and Beyond*, 2009, p. 29.
- [630] N. B. Mabrouk, N. Georgantas, and V. Issarny, "A semantic end-to-end QoS model for dynamic service oriented environments," in *2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*, May 2009, pp. 34–41. doi: 10.1109/PESOS.2009.5068817.
- [631] D. Alonso-Ríos, A. Vázquez-García, E. Mosqueira-Rey, and V. Moret-Bonillo, "Usability: A Critical Analysis and a Taxonomy," *null*, vol. 26, no. 1, pp. 53–74, Dec. 2009, doi: 10.1080/10447310903025552.
- [632] I. Castillo, F. Losavio, A. Matteo, and J. Bøegh, "REquirements, Aspects and Software Quality: the REASQ model," *Journal Object Technology*, vol. 9, no. 4, pp. 69–91, 2010.
- [633] A. Alvaro, E. S. de Almeida, and S. R. de L. Meira, "A software component quality framework," *SIGSOFT Softw. Eng. Notes*, vol. 35, no. 1, pp. 1–18, 2010.
- [634] S. Kalaimagal and R. Srinivasan, "Q'Facto 10-A commercial off-the-shelf component quality model proposal," *Journal of Software Engineering*, vol. 4, no. 1, pp. 1–15, 2010.
- [635] G. Malak, H. Sahraoui, L. Badri, and M. Badri, "Modeling Web Quality Using a Probabilistic Approach: An Empirical Validation," *ACM Trans. Web*, vol. 4, no. 3, Jul. 2010, doi: 10.1145/1806916.1806918.
- [636] P. Lew, L. Olsina, and L. Zhang, "Quality, Quality in Use, Actual Usability and User Experience as Key Drivers for Web Application Evaluation," in *Web Engineering*, Berlin, Heidelberg, 2010, pp. 218–232.
- [637] M. Herrera, M. Á. Moraga, I. Caballero, and C. Calero, "Quality in Use Model for Web Portals (QiUWeP)," in *Current Trends in Web Engineering*, Berlin, Heidelberg, 2010, pp. 91–101.
- [638] Miao Fan, Yi Luo, Guoshi Wu, and Xiangling Fu, "An improved analytic hierarchy process model on Software Quality Evaluation," in *The 2nd International Conference on Information Science and Engineering*, Dec. 2010, pp. 1838–1842. doi: 10.1109/ICISE.2010.5690372.
- [639] M. Luckey, A. Baumann, D. Méndez, and S. Wagner, "Reusing Security Requirements Using an Extended Quality Model," in *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems*, New York, NY, USA, 2010, pp. 1–7. doi: 10.1145/1809100.1809101.
- [640] R. Mohanty, V. Ravi, and M. R. Patra, "Web-services classification using intelligent techniques," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5484–5490, Jul. 2010, doi: 10.1016/j.eswa.2010.02.063.
- [641] B. Yin, H. Yang, P. Fu, and X. Chen, "A semantic web services discovery algorithm based on QoS ontology," in *International Conference on Active Media Technology*, 2010, vol. 6335, pp. 166–173.

-
- [642] Z. Pan and J. Baik, "A QOS Enhanced Framework and Trust Model for Effective Web Services Selection.," *Journal of Web Engineering*, vol. 9, no. 2, pp. 186–204, 2010.
- [643] S. Zhang and M. Song, "An architecture design of life cycle based SLA management," in *2010 The 12th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2010, vol. 2, pp. 1351–1355.
- [644] U. B. Corrêa, L. Lamb, L. Carro, L. Brisolará, and J. Mattos, "Towards Estimating Physical Properties of Embedded Systems using Software Quality Metrics," in *2010 10th IEEE International Conference on Computer and Information Technology*, Jul. 2010, pp. 2381–2386. doi: 10.1109/CIT.2010.409.
- [645] F. J. Domínguez-Mayo, M. J. Escalona, M. Mejías, and A. H. Torres, "A Quality Model in a Quality Evaluation Framework for MDWE methodologies," in *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)*, May 2010, pp. 495–506. doi: 10.1109/RCIS.2010.5507323.
- [646] S. M. Hwang and S. Im, "Korean Software Process Quality Certification Model," in *2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, May 2011, pp. 123–128. doi: 10.1109/CNSI.2011.55.
- [647] N. Upadhyay, B. M. Despande, and V. P. Agrawal, "Towards a Software Component Quality Model," in *Advances in Computer Science and Information Technology*, Berlin, Heidelberg, 2011, pp. 398–412.
- [648] A. B. AL-Badareen, M. H. Selamat, J. Din, M. A. Jabar, and S. Turaev, "Software Quality Evaluation: User's View," *International Journal of Applied Mathematics and Informatics*, vol. 5, no. 3, pp. 200–207, 2011.
- [649] K. Lochmann and A. Goeb, "A Software Quality Model for SOA," in *Proceedings of the 8th International Workshop on Software Quality*, New York, NY, USA, 2011, pp. 18–25. doi: 10.1145/2024587.2024593.
- [650] T. Bakota, P. Hegedús, P. Körtvélyesi, R. Ferenc, and T. Gyimóthy, "A probabilistic software quality model," in *2011 27th IEEE International Conference on Software Maintenance (ICSM)*, Sep. 2011, pp. 243–252. doi: 10.1109/ICSM.2011.6080791.
- [651] L. Olsina, P. Lew, A. Dieser, and B. Rivera, "Using web quality models and a strategy for purpose-oriented evaluations," *Journal of Web Engineering*, vol. 10, no. 4, pp. 316–352, 2011.
- [652] P. Murthy, S. K. V, T. Sharma, and K. Rao, "Quality Model Driven Dynamic Analysis," in *2011 IEEE 35th Annual Computer Software and Applications Conference*, Jul. 2011, pp. 360–365. doi: 10.1109/COMPSAC.2011.54.
- [653] P. D. D. Dominic and H. Jati, "A comparison of Asian airlines websites quality: using a non-parametric test," *International Journal of Business Innovation and Research (IJBIR)*, vol. 5, no. 5, pp. 499–523, 2011.
- [654] R. Rekik and I. Kallel, "Fuzzy reduced method for evaluating the quality of institutional web sites," in *2011 7th International Conference on Next Generation Web Services Practices*, Oct. 2011, pp. 296–301. doi: 10.1109/NWeSP.2011.6088194.
- [655] Y. Singh, R. Malhotra, and P. Gupta, "Empirical validation of web metrics for improving the quality of web page," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 5, pp. 22–28, 2011.
- [656] P. Bocciarelli and A. D'Ambrogio, "A model-driven method for describing and predicting the reliability of composite services," *Software & Systems Modeling*, vol. 10, no. 2, pp. 265–280, 2011.
- [657] J. Qiu and F. Yu, "Research on Semantic Dynamic Service Combination Module," in *2011 International Conference on Internet Technology and Applications*, Aug. 2011, pp. 1–4. doi: 10.1109/ITAP.2011.6006212.
- [658] N. Debnath, P. Martellotto, M. Daniele, D. Riesco, and G. Montejano, "A method to evaluate QoS of web services required by a workflow," in *2011 11th International Conference on ITS Telecommunications*, Aug. 2011, pp. 640–645. doi: 10.1109/ITST.2011.6060134.
- [659] T. Mens, L. Doctors, N. Habra, B. Vanderose, and F. Kamseu, "QUALGEN: Modeling and Analysing the Quality of Evolving Software Systems," in *2011 15th European Conference on Software Maintenance and Reengineering*, Mar. 2011, pp. 351–354. doi: 10.1109/CSMR.2011.50.
- [660] D. Azar and J. Vybihal, "An ant colony optimization algorithm to improve software quality prediction models: Case of class stability," *Information and Software Technology*, vol. 53, no. 4, pp. 388–393, Apr. 2011, doi: 10.1016/j.infsof.2010.11.013.

- [661] \Lukasz Radliński, "A conceptual Bayesian net model for integrated software quality prediction," *Annales Universitatis Mariae Curie-Sklodowska, sectio AI-Informatica*, vol. 11, no. 4, pp. 49–60, 2011.
- [662] E. Bagheri and D. Gasevic, "Assessing the maintainability of software product line feature models using structural metrics," *Software Quality Journal*, vol. 19, no. 3, pp. 579–612, Sep. 2011, doi: 10.1007/s11219-010-9127-2.
- [663] Müller Tristan, "How to choose a free and open source integrated library system," *OCLC Systems & Services: International digital library perspectives*, vol. 27, no. 1, pp. 57–78, Jan. 2011, doi: 10.1108/10650751111106573.
- [664] C. Chirila, D. Juratoni, D. Tudor, and V. Crețu, "Towards a software quality assessment model based on open-source static code analyzers," in *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, May 2011, pp. 341–346. doi: 10.1109/SACI.2011.5873026.
- [665] J. S. Challa, A. Paul, Y. Dada, V. Nerella, P. R. Srivastava, and A. P. Singh, "Integrated software quality evaluation: a fuzzy multi-criteria approach," *Journal of Information Processing Systems*, vol. 7, no. 3, pp. 473–518, 2011.
- [666] M. Espinilla, F. Domínguez-Mayo, M. Escalona, M. Mejías, M. Ross, and G. Staples, "A Method Based on AHP to Define the Quality Model of QuEF," in *Knowledge Engineering and Management*, Springer, 2011, pp. 685–694. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-25661-5_85
- [667] M. Abdellatif, A. B. M. Sultan, M. A. Jabar, and R. Abdullah, "A technique for quality evaluation of e-learning from developers perspective," *American Journal of Economics and Business Administration*, vol. 3, no. 1, pp. 157–164, 2011.
- [668] T. Um, N. Kim, D. Lee, and H. P. In, "A Quality Attributes Evaluation Method for an Agile Approach," in *2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, May 2011, pp. 460–461. doi: 10.1109/CNSI.2011.93.
- [669] D. Franke, S. Kowalewski, and C. Weise, "A Mobile Software Quality Model," in *2012 12th International Conference on Quality Software*, Aug. 2012, pp. 154–157. doi: 10.1109/QSIC.2012.49.
- [670] L. Yu and A. Mishra, "Experience in Predicting Fault-Prone Software Modules Using Complexity Metrics," *null*, vol. 9, no. 4, pp. 421–434, Jan. 2012, doi: 10.1080/16843703.2012.11673302.
- [671] B. Singh and S. P. Kannoja, "A Model for Software Product Quality Prediction," *Journal of Software Engineering and Applications*, vol. 5, no. 6, pp. 395–401, 2012, doi: 10.4236/jsea.2012.56046.
- [672] R. Malhotra and A. Jain, "Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality," *Journal of Information Processing Systems*, vol. 8, no. 2, pp. 241–262, Jun. 2012, doi: 10.3745/JIPS.2012.8.2.241.
- [673] S. K. Dubey, A. Gulati, and A. Rana, "Integrated model for software usability," *International Journal on Computer Science and Engineering*, vol. 4, no. 3, p. 429, 2012.
- [674] Bhattacharya Debjani, Gulla Umesh, and Gupta M.P., "E-service quality model for Indian government portals: citizens' perspective," *Journal of Enterprise Information Management*, vol. 25, no. 3, pp. 246–271, Jan. 2012, doi: 10.1108/17410391211224408.
- [675] S. Elling, L. Lentz, M. de Jong, and H. van den Bergh, "Measuring the quality of governmental websites in a controlled versus an online setting with the 'Website Evaluation Questionnaire,'" *Government Information Quarterly*, vol. 29, no. 3, pp. 383–393, Jul. 2012, doi: 10.1016/j.giq.2011.11.004.
- [676] O. Moser, F. Rosenberg, and S. Dustdar, "Domain-Specific Service Selection for Composite Services," *IEEE Transactions on Software Engineering*, vol. 38, no. 4, pp. 828–843, Aug. 2012, doi: 10.1109/TSE.2011.43.
- [677] O. Cabrera and X. Franch, "A quality model for analysing web service monitoring tools," in *2012 Sixth International Conference on Research Challenges in Information Science (RCIS)*, May 2012, pp. 1–12. doi: 10.1109/RCIS.2012.6240444.
- [678] OASIS Web Services Quality Model Technical Committee, "Web Services Quality Factors Version 1.0," Oct. 31, 2012. <http://docs.oasisopen.org/wsqm/wsqr/v1.0/WS-Quality-Factors.pdf>
- [679] R. Phalnikar and P. A. Khutade, "Survey of QoS based web service discovery," in *2012 World Congress on Information and Communication Technologies*, Nov. 2012, pp. 657–661. doi: 10.1109/WICT.2012.6409157.

- [680] P. Nandanam and R. Rajmohan, "QoS evaluation for web services in cloud computing," in *2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)*, Jul. 2012, pp. 1–8. doi: 10.1109/ICCCNT.2012.6395991.
- [681] E. Rashid, S. Patnaik, and V. Bhattacharjee, "Software quality estimation using machine learning: Case-Based reasoning technique," *International Journal of Computer Applications*, vol. 58, no. 14, 2012.
- [682] A. Raza, L. F. Capretz, and F. Ahmed, "An open source usability maturity model (OS-UMM)," *Computers in Human Behavior*, vol. 28, no. 4, pp. 1109–1121, Jul. 2012, doi: 10.1016/j.chb.2012.01.018.
- [683] E. K. El-Rayyes and I. M. Abu-Zaid, "New Model to Achieve Software Quality Assurance (SQA) in Web Application," *International Journal of Science and Technology*, vol. 2, no. 7, pp. 423–426, 2012.
- [684] D. Masoumi and B. Lindström, "Quality in e-learning: a framework for promoting and assuring quality in virtual institutions," *Journal of Computer Assisted Learning*, vol. 28, no. 1, pp. 27–41, Feb. 2012, doi: 10.1111/j.1365-2729.2011.00440.x.
- [685] J. Park, H. Kim, J. Shin, and J. Baik, "An Embedded Software Reliability Model with Consideration of Hardware Related Software Failures," in *2012 IEEE Sixth International Conference on Software Security and Reliability*, Jun. 2012, pp. 207–214. doi: 10.1109/SERE.2012.10.
- [686] C. Calero and M. Bertoa, "25010+ S: A software quality model with sustainable characteristics. Sustainability as an element of software quality," in *Proceeding of the Green in Software Engineering Green by Software Engineerin (GIBSE), co-located with AOSD 2013*, Fukuoka, Japan, Mar. 2013, vol. 18.
- [687] R. D. Venkatasubramanyam and S. Nayak, "An Overview of Technical Models for Dynamic Analysis," *Lecture Notes on Software Engineering*, vol. 1, no. 2, p. 160, 2013.
- [688] X. Wang, M. Ceberio, S. Virani, A. Garcia, and J. Cummins, "A hybrid algorithm to extract fuzzy measures for software quality assessment," *Journal of Uncertain Systems*, vol. 7, no. 3, pp. 219–237, 2013.
- [689] N. Baliyan and S. Kumar, "Quality Assessment of Software as a Service on Cloud Using Fuzzy Logic," in *2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, Oct. 2013, pp. 1–6. doi: 10.1109/CCEM.2013.6684439.
- [690] S. Zahra, A. Khalid, and A. Javed, "An efficient and effective new generation objective quality model for mobile applications," *International Journal of Modern Education and Computer Science*, vol. 5, no. 4, p. 36, 2013.
- [691] M. Ulman, V. Vostrovskiy, and J. Tyrychtr, "Agricultural e-government: Design of quality evaluation method based on ISO square quality model," *AGRIS on-line Papers in Economics and Informatics*, vol. 5, no. 665-2016–44964, pp. 211–222, 2013.
- [692] O. Rababah, T. Hamtini, O. Harfoushi, B. Al-Shboul, R. Obiedat, and S. Nawafleh, "Towards developing successful e-government websites," *Journal of Software Engineering and Applications*, vol. 6, no. 11, p. 559, 2013.
- [693] D. Dixit, "CBQM: Component Based Quality Model," in *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on*, 2013, pp. 1–5.
- [694] M. A. Ahmed and H. A. Al-Jamimi, "Machine learning approaches for predicting software maintainability: a fuzzy-based transparent model," *IET Software*, vol. 7, no. 6, pp. 317–326(9), Dec. 2013.
- [695] Y. Duan, A. Kattepur, F. Getahun, A. Elfakiz, and W. Du, "Releasing the Power of Variability: Towards Constraint Driven Quality Assurance," in *2013 Second IIAI International Conference on Advanced Applied Informatics*, Sep. 2013, pp. 15–20. doi: 10.1109/IIAI-AAI.2013.23.
- [696] L. Aversano and M. Tortorella, "Quality evaluation of floss projects: Application to ERP systems," *Information and Software Technology*, vol. 55, no. 7, pp. 1260–1276, Jul. 2013, doi: 10.1016/j.infsof.2013.01.007.
- [697] N. J. Pizzi, "A fuzzy classifier approach to estimating software quality," *Information Sciences*, vol. 241, pp. 1–11, Aug. 2013, doi: 10.1016/j.ins.2013.04.027.
- [698] A. Mayr, R. Plösch, and M. Saft, "Objective Measurement of Safety in the Context of IEC 61508-3," in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, Sep. 2013, pp. 45–52. doi: 10.1109/SEAA.2013.32.

- [699] S. Srivastava and R. Kumar, "Indirect method to measure software quality using CK-OO suite," in *2013 International Conference on Intelligent Systems and Signal Processing (ISSP)*, Mar. 2013, pp. 47–51. doi: 10.1109/ISSP.2013.6526872.
- [700] G. Samarthyam, G. Suryanarayana, T. Sharma, and S. Gupta, "MIDAS: A design quality assessment method for industrial software," in *2013 35th International Conference on Software Engineering (ICSE)*, May 2013, pp. 911–920. doi: 10.1109/ICSE.2013.6606640.
- [701] B. Penzenstadler and H. Femmer, "A Generic Model for Sustainability with Process- and Product-Specific Instances," in *Proceedings of the 2013 Workshop on Green in/by Software Engineering*, New York, NY, USA, 2013, pp. 3–8. doi: 10.1145/2451605.2451609.
- [702] A. Roth, A. Ganser, H. Lichter, and B. Rumpe, "Staged Evolution with Quality Gates for Model Libraries," in *the International Workshop on Document Changes: Modeling, Detection, Storage and Visualization*, Sep. 2013, vol. 1008. [Online]. Available: <https://arxiv.org/abs/1408.5707>
- [703] R. V. Small and M. P. Arnone, "WebCHECK: The Website Evaluation Instrument," *Knowledge Quest*, vol. 42, no. 3, pp. 58–63, Feb. 2014.
- [704] N. M. Hien, "A study on evaluation of e-government service quality," *International Journal of Humanities and Social Sciences*, vol. 8, no. 1, pp. 16–19, 2014.
- [705] Z. Masood, S. Xuequn, and J. Yousaf, "Usability evaluation framework for software engineering methodologies," *Lecture Notes on Software Engineering*, vol. 2, no. 3, p. 225, 2014.
- [706] A. Adline and M. Ramachandran, "Predicting the software fault using the method of genetic algorithm," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 3, no. 2, pp. 390–398, 2014.
- [707] A. Puri and H. Singh, "Genetic algorithm based approach for finding faulty modules in open source software systems," *International Journal of Computer Science and Engineering Survey*, vol. 5, no. 3, p. 29, 2014.
- [708] G. Zhang, H. Ye, and Y. Lin, "Quality attribute modeling and quality aware product configuration in software product lines," *Software Quality Journal*, vol. 22, no. 3, pp. 365–401, Sep. 2014, doi: 10.1007/s11219-013-9197-z.
- [709] Y. Kuwata, K. Takeda, and H. Miura, "A Study on Maturity Model of Open Source Software Community to Estimate the Quality of Products," *Procedia Computer Science*, vol. 35, pp. 1711–1717, Jan. 2014, doi: 10.1016/j.procs.2014.08.264.
- [710] X. Zheng, P. Martin, K. Brohman, and L. D. Xu, "CLOUDQUAL: A Quality Model for Cloud Services," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1527–1536, May 2014, doi: 10.1109/TII.2014.2306329.
- [711] S. Gupta, H. K. Singh, R. D. Venkatasubramanyam, and U. Uppili, "SCQAM: A Scalable Structured Code Quality Assessment Method for Industrial Software," in *Proceedings of the 22nd International Conference on Program Comprehension*, New York, NY, USA, 2014, pp. 244–252. doi: 10.1145/2597008.2597806.
- [712] D. Athanasiou, A. Nugroho, J. Visser, and A. Zaidman, "Test Code Quality and Its Relation to Issue Handling Performance," *IEEE Transactions on Software Engineering*, vol. 40, no. 11, pp. 1100–1125, Nov. 2014, doi: 10.1109/TSE.2014.2342227.
- [713] I. Le stari and B. Hendradjaya, "The application model of learning management system quality in asynchronous blended learning system," in *2014 International Conference on Electrical Engineering and Computer Science (ICEECS)*, Nov. 2014, pp. 223–228. doi: 10.1109/ICEECS.2014.7045251.
- [714] U. L. Yuhana, A. B. Raharjo, and S. Rochimah, "Academic information system quality measurement using quality instrument: A proposed model," in *2014 International Conference on Data and Software Engineering (ICODSE)*, Nov. 2014, pp. 1–6. doi: 10.1109/ICODSE.2014.7062684.
- [715] E. Ziemba, T. Papaj, and D. Descours, "Factors affecting success of e-government portals: a perspective of software quality model," in *Proceedings of European Conference on eGovernment*, 2014, pp. 252–262.
- [716] T. A. Alrawashdeh, M. I. Muhairat, and S. M. Alqatawneh, "A Quantitative Evaluation of ERP Systems Quality Model," in *2014 11th International Conference on Information Technology: New Generations*, Apr. 2014, pp. 46–49. doi: 10.1109/ITNG.2014.37.

References

- [717] M. U. Malik, H. Nasir, and A. Javed, "An efficient objective quality model for agile application development," *International Journal of Computer Applications*, vol. 85, no. 8, 2014.
- [718] A. A. B. Baqais, M. Alshayeb, and Z. A. Baig, "Hybrid intelligent model for software maintenance prediction," *Proceedings of World Congress on Engineering*, pp. 358–362, 2014.
- [719] C. I. Bezerra, R. M. Andrade, and J. M. S. Monteiro, "Measures for quality evaluation of feature models," in *International Conference on Software Reuse*, 2015, pp. 282–297.
- [720] P. Sudhaman and C. Thangavel, "Efficiency analysis of ERP projects—software quality perspective," *International Journal of Project Management*, vol. 33, no. 4, pp. 961–970, May 2015, doi: 10.1016/j.ijproman.2014.10.011.
- [721] H.-J. Sohn, M.-G. Lee, B.-M. Seong, and J.-B. Kim, "Quality evaluation criteria based on open source mobile HTML5 UI framework for development of cross-platform," *International Journal of Software Engineering and Its Applications*, vol. 9, no. 6, pp. 1–12, 2015.
- [722] J. M. Alves, C. Wangenheim, T. Lacerda, A. Savaris, and A. Wangenheim, "Adequate software quality evaluation model v1. 0," *Instituto Nacional para Convergência Digital–INCoD, Tech. Rep*, 2015.
- [723] G. Ladányi, Z. Tóth, R. Ferenc, and T. Keresztesi, "A software quality model for RPG," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Mar. 2015, pp. 91–100. doi: 10.1109/SANER.2015.7081819.
- [724] S. Rochimah, H. I. Rahmani, and U. L. Yuhana, "Usability characteristic evaluation on administration module of Academic Information System using ISO/IEC 9126 quality model," in *2015 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, May 2015, pp. 363–368. doi: 10.1109/ISITIA.2015.7220007.
- [725] D. D. J. Suwawi, E. Darwiyanto, and M. Rochmani, "Evaluation of academic website using ISO/IEC 9126," in *2015 3rd International Conference on Information and Communication Technology (ICoICT)*, May 2015, pp. 222–227. doi: 10.1109/ICoICT.2015.7231426.
- [726] A. Calderón and M. Ruiz, "A systematic literature review on serious games evaluation: An application to software project management," *Computers & Education*, vol. 87, pp. 396–422, Sep. 2015, doi: 10.1016/j.compedu.2015.07.011.
- [727] W. Zhang, L. Huang, V. Ng, and J. Ge, "SMPLeArner: learning to predict software maintainability," *Automated Software Engineering*, vol. 22, no. 1, pp. 111–141, Mar. 2015, doi: 10.1007/s10515-014-0161-3.
- [728] R. Jindal, R. Malhotra, and A. Jain, "Predicting Software Maintenance effort using neural networks," in *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, Sep. 2015, pp. 1–6. doi: 10.1109/ICRITO.2015.7359258.
- [729] C. Sharma and S. K. Dubey, "Reliability evaluation of software system using AHP and Fuzzy Topsis approach," in *Proceedings of Fifth International Conference on Soft Computing for Problem Solving*, 2016, pp. 81–92.
- [730] R. Andrian, B. Hendradjaya, and W. D. Sunindyo, "Software assessment model using metrics products for e-Government in the G2B model," in *2016 4th International Conference on Information and Communication Technology (ICoICT)*, May 2016, pp. 1–6. doi: 10.1109/ICoICT.2016.7571931.
- [731] T. Marir, F. Mokhati, H. Bouchlaghem-Seridi, Y. Acid, and M. Bouzid, "QM4MAS: a quality model for multi-agent systems," *International Journal of Computer Applications in Technology*, vol. 54, no. 4, pp. 297–310, 2016.
- [732] M. Sarrab, M. Elbasir, and S. Alnaeli, "Towards a quality model of technical aspects for mobile learning services: An empirical investigation," *Computers in Human Behavior*, vol. 55, pp. 100–112, Feb. 2016, doi: 10.1016/j.chb.2015.09.003.
- [733] A. Jain, S. Tarwani, and A. Chug, "An empirical investigation of evolutionary algorithm for software maintainability prediction," in *2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, Mar. 2016, pp. 1–6. doi: 10.1109/SCEECS.2016.7509314.
- [734] S. Forouzani, Y. K. Chiam, and S. Forouzani, "Method for Assessing Software Quality Using Source Code Analysis," in *Proceedings of the Fifth International Conference on Network, Communication and Computing*, New York, NY, USA, 2016, pp. 166–170. doi: 10.1145/3033288.3033316.

- [735] N. Kumar, R. Dadhich, and A. Shastri, "MAQM: a generic object-oriented framework to build quality models for Web-based applications," *International Journal of System Assurance Engineering and Management*, vol. 8, no. 2, pp. 716–729, Nov. 2017, doi: 10.1007/s13198-016-0512-5.
- [736] R. M. Wibowo, P. A. Erna, and I. Hidayah, "Heuristic evaluation and user testing with ISO 9126 in evaluating of decision support system for recommendation of outstanding marketing officer," in *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, Nov. 2017, pp. 454–458. doi: 10.1109/SIET.2017.8304181.
- [737] A. Tabassum, S. Nazir Bhatti Bahria, A. Rida Asghar Bahria, I. Manzoor, and A. Imtiaz, "Optimized Quality Model for Agile Development: Extreme Programming (XP) as a Case Scenario," *IJACSA International Journal of Advanced Computer Science and Applications*, vol. 8, no. 4, 2017.
- [738] Ramadiani, Azainil, U. Haryaka, F. Agus, and A. H. Kridalaksana, "User Satisfaction Model for e-Learning Using Smartphone," *Procedia Computer Science*, vol. 116, pp. 373–380, Jan. 2017, doi: 10.1016/j.procs.2017.10.070.
- [739] S. Anwer, A. Adbellatif, M. Alshayeb, and M. S. Anjum, "Effect of coupling on software faults: An empirical study," in *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*, Mar. 2017, pp. 211–215. doi: 10.1109/C-CODE.2017.7918930.
- [740] N. R. M. Suradi, S. Kahar, and N. A. A. Jamaluddin, "Identification of software quality characteristics on academic application in higher education institution (HEI)," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 10, no. 2–7, pp. 133–136, 2018.
- [741] N. Condori-Fernandez and P. Lago, "Characterizing the contribution of quality requirements to software sustainability," *Journal of Systems and Software*, vol. 137, pp. 289–305, Mar. 2018, doi: 10.1016/j.jss.2017.12.005.

Annexes

Annex 1. Catalog of the main existing aggregation operators (from Chapter IV.3.d)

TABLE 39 - CATALOG OF THE MAIN EXISTING AGGREGATION OPERATORS, BASED ON DETYNIECKI [143], WAGNER [27] AND DUJMOVIC & BAYUCAN [144]

Group of aggregation operators	Aggregation operator	Formula
Basic operators	Arithmetic mean	$\mu = M(x_1, x_2, \dots, x_n) = \sum_{i=1}^n \left(\frac{1}{n} \cdot x_i\right)$
	Weighted mean	$M_{w_1, \dots, w_n}(x_1, x_2, \dots, x_n) = \sum_{i=1}^n (w_i \cdot x_i) \text{ where } w_i \geq 0 \text{ and } \sum_{i=1}^n w_i = 1$
	Mode	$M_m(x_1, x_2, \dots, x_k) = x_j \text{ and } n_j = \max(n_1, n_2, \dots, n_k)$ where (n_1, n_2, \dots, n_k) are frequencies of (x_1, x_2, \dots, x_k)
	Median	$M_{0.5}(x_1, x_2, \dots, x_k) = \begin{cases} x_{((n+1)/2)} & \text{if } n \text{ is odd} \\ \frac{1}{2}(x_{(n/2)} + x_{(n/2+1)}) & \text{otherwise} \end{cases}$
	Median absolute deviation	$D(x_1, x_2, \dots, x_n) = \sum_{i=1}^n \left(\frac{1}{n} \cdot x_i - M_{0.5}(x_1, x_2, \dots, x_n) \right)$
	Minimum	$\forall x_i, y \leq x_i \Rightarrow y = \min(x_1, x_2, \dots, x_k)$
	Maximum	$\forall x_i, y \geq x_i \Rightarrow y = \max(x_1, x_2, \dots, x_k)$
	Weighted minimum	$\min_{w_1, \dots, w_n}^{\otimes}(x_1, x_2, \dots, x_n)$ $= \sum_{i=1}^n [i \cdot (w_{\sigma(i)-} - w_{\sigma(i+1)}) \cdot \min(x_{\sigma(i)}, \dots, x_{\sigma(i)})]$ where $w_i \geq 0, \sum_{i=1}^n w_i = 1, \sigma^{(i)}$ and $w_{\sigma(1)} \geq w_{\sigma(2)} \geq \dots \geq w_{\sigma(n)}, w_{\sigma(i+1)} = 0$
	Weighted maximum	$\max_{w_1, \dots, w_n}^{\otimes}(x_1, x_2, \dots, x_n)$ $= \sum_{i=1}^n [i \cdot (w_{\sigma(i)-} - w_{\sigma(i+1)}) \cdot \max(x_{\sigma(i)}, \dots, x_{\sigma(i)})]$ where $w_i \geq 0, \sum_{i=1}^n w_i = 1, \sigma^{(i)}$ and $w_{\sigma(1)} \geq w_{\sigma(2)} \geq \dots \geq w_{\sigma(n)}, w_{\sigma(i+1)} = 0$

	Range	$R(x_1, x_2, \dots, x_k) = \max(x_1, x_2, \dots, x_k) - \min(x_1, x_2, \dots, x_k)$
	Variance	$\sigma^2 = S^2(x_1, x_2, \dots, x_k) = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$ $= \frac{1}{n} \sum_{i=1}^n \left(x_i - \sum_{i=1}^n \left(\frac{1}{n} \cdot x_i \right) \right)^2$
	Standard deviation	$\sigma = S(x_1, x_2, \dots, x_k) = \sqrt{\sigma^2}$ $= \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}$
Quasi-arithmetic means	Geometric mean	$M_{geometric}(x_1, x_2, \dots, x_n) = \left(\prod_{i=1}^n x_i \right)^{\frac{1}{n}}$
	Harmonic mean	$M_{harmonic}(x_1, x_2, \dots, x_n) = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \dots + \frac{1}{x_n}}$
	Generic quasi-arithmetic mean	$M(x_1, x_2, \dots, x_n) = \left[\sum_{i=1}^n \left(\frac{1}{n} \cdot x_i^\alpha \right) \right]^{\frac{1}{\alpha}}$
Additive generated symmetric sum	$S(x_1, x_2, \dots, x_n) = f^{-1} \left[\sum_{i=1}^n f(x_i) \right]$ <p>where f is a strictly monotone continuous function and $f(x) + f(1 - x) = 0$</p>	
Gini coefficient	$G_n = \frac{2 \sum_{i=1}^n x_{(i)} - (n+1) \sum_{i=1}^n x_{(i)}}{n \sum_{i=1}^n x_{(i)}}$	
Ordered Weighted Averaging Operators	Ordered Weighted Averaging	$OWA(x_1, x_2, \dots, x_n) = \sum_{i=1}^n w_i \cdot x_{\sigma(i)}$ <p>where $w_i \geq 0, \sum_{i=1}^n w_i = 1$ and $0 \leq w_{\sigma(1)} \leq w_{\sigma(2)} \leq \dots \leq w_{\sigma(n)}$</p>
	Degree of maxness (or orness)	$maxness(w_1, w_2, \dots, w_n) = \sum_{i=1}^n w_i \cdot \frac{i-1}{n-1}$

	Degree of dispersion	$dispersion(w_1, w_2, \dots, w_n) = - \sum_{i=1}^n w_i \cdot \ln(w_i)$
Choquet and Sugeno discrete Fuzzy Integrals	Discrete Sugeno Integrals	$Sugeno_{\mu}(x_1, x_2, \dots, x_n) = \max_{i=1}^n \left(\min(x_{\sigma(i)}, \mu(C_{\sigma(i)})) \right)$ <p style="text-align: center;">where $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \dots \leq x_{\sigma(n)}$</p>
	Discrete Choquet Integrals	$Choquet_{\mu}(x_1, x_2, \dots, x_n) = \sum_{i=1}^n (x_{\sigma(i)} - x_{\sigma(i-1)}) \cdot (\mu(C_{\sigma(i)}))$ <p style="text-align: center;">where $x_{\sigma(0)} = 0$ and $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \dots \leq x_{\sigma(n)}$</p>
Fusion operators	Bayesian fusion	$P(x x_1, x_2) = \frac{P(x_1 x) \cdot P(x_2 x) \cdot P(x)}{P(x_1, x_2)}$
	Probabilistic fusion: prioritized conjunction	$\pi_{1 \wedge 2} = \min(\pi_1, \max(\pi_2, 1 - h(\pi_1, \pi_2)))$ <p style="text-align: center;">where $h(\pi_1, \pi_2) = \sup_x(\min(\pi_1(x), \pi_2(x)))$</p>
	Probabilistic fusion: prioritized disjunction	$\pi_{1 \vee 2} = \max(\pi_1, \min(\pi_2, h(\pi_1, \pi_2)))$ <p style="text-align: center;">where $h(\pi_1, \pi_2) = \sup_x(\min(\pi_1(x), \pi_2(x)))$</p>
Compensatory operators	Zimmermann and Zysno	$Z_{\gamma}(x_1, \dots, x_n) = \left(\prod_{i=1}^n x_i \right)^{1-\gamma} \cdot \left(1 - \prod_{i=1}^n (1 - x_i) \right)^{\gamma}$ <p style="text-align: center;">where γ degree of compensation</p>
	Exponential compensatory operators	$E_{\gamma}^{T,S}(x_1, \dots, x_n) = (T(x_1, \dots, x_n))^{1-\gamma} \cdot (S(x_1, \dots, x_n))^{\gamma}$ <p style="text-align: center;">where γ degree of compensation T is a t – norm and S is a t – conorm</p>
	Convex-linear compensatory operator	$L_{\gamma}^{T,S}(x_1, \dots, x_n) = (1 - \gamma) \cdot T(x_1, \dots, x_n) + \gamma \cdot S(x_1, \dots, x_n)$ <p style="text-align: center;">where γ degree of compensation T is a t – norm and S is a t – conorm</p>

	Associative compensatory operator	$C(x, y) = f^{-1}(f(x) + f(y))$ $\text{where } f(x) = \begin{cases} -g\left(\frac{x}{e}\right) & \text{if } x \leq e \\ -g\left(\frac{x-e}{1-e}\right) & \text{if } x \geq e \end{cases}$ <p>where g is an additive generator of a t – norm, h is an additive generator of a t – conorm e is a neutral element</p>
Uninorms	Minimal uninorms	$U_{min}(x, y) = \begin{cases} e.T\left(\frac{x}{e}, \frac{y}{e}\right) & \text{for } x \leq e \text{ and } y \leq e \\ e + (1 - e).S\left(\frac{x-e}{1-e}, \frac{y-e}{1-e}\right) & \text{for } x > e \text{ and } y > e \\ \min(x, y) & \text{elsewhere} \end{cases}$
	Maximal uninorms	$U_{max}(x, y) = \begin{cases} e.T\left(\frac{x}{e}, \frac{y}{e}\right) & \text{for } x \leq e \text{ and } y \leq e \\ e + (1 - e).S\left(\frac{x-e}{1-e}, \frac{y-e}{1-e}\right) & \text{for } x > e \text{ and } y > e \\ \min(x, y) & \text{elsewhere} \end{cases}$
	Generated uninorms	$U(x, y) = g^{(-1)}(g(x) + g(y))$
	Nullnorms	$N(x, y) = \begin{cases} a.S\left(\frac{x}{a}, \frac{y}{a}\right) & \text{for } x \leq a \text{ and } y \leq a \\ a + (1 - a).T\left(\frac{x-a}{1-a}, \frac{y-a}{1-a}\right) & \text{for } x > a \text{ and } y > a \\ a & \text{elsewhere} \end{cases}$
Weighted power mean “Continuous Preference Logic” parameter of Dujmovic and Bayucan [144]		$E(r) = \begin{cases} \left(\sum_{i=1}^n w_i \cdot E_i\right)^{\frac{1}{r}} & \text{for } r \neq +\infty \text{ and } r \neq -\infty \\ \max(E_1, \dots, E_n) & \text{for } r = +\infty \\ \min(E_1, \dots, E_n) & \text{for } r = -\infty \end{cases}$ <p>where $0 \leq E_i \leq 1$, $0 < w_i < 1$ for $i \in [1; n]$ and $\sum_{i=1}^n w_i = 1$</p>

Annex 2. Main sequence, string, or similarity distance formulas

Distance name	Formula
Hamming's distance [87]	$d_{hamming}(x, y) = \sum_{i=0}^{n-1} (x_i \oplus y_i)$ <p>where x and y are 2 sequences of symbols and $(x_i \oplus y_i) = \begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{otherwise} \end{cases}$</p>

TABLE 40 - EXAMPLES OF HAMMING'S DISTANCE COMPUTATION: RED CHARACTERS INDICATE DIFFERING CHARACTERS

Example 1	Distance	Example 2	Distance	Example 3	Distance	Example 4	Distance
solar		imperial		acknowledgement		123456AD90	
vs.	1	vs.	3	vs.	12	vs.	5
polar		interval		accomplishments		124357AC97	

Levenshtein's distance [88]	$d_{lev_{x,y}}(i, j) = \begin{cases} \max(i, j) & \text{if } \min(i, j) = 0 \\ \min \begin{cases} lev_{x,y}(i-1, j) + 1 \\ lev_{x,y}(i, j-1) + 1 \\ lev_{x,y}(i-1, j-1) + 1_{(x_i \neq y_j)} \end{cases} & \text{otherwise} \end{cases}$ <p>where x and y are 2 strings</p>
-----------------------------	---

TABLE 41 - EXAMPLES OF LEVENSHTAIN'S DISTANCE COMPUTATION: RED CHARACTERS INDICATE DIFFERING CHARACTERS

Example 1	Distance	Example 2	Distance	Example 3	Distance	Example 4	Distance
solar		solar		acknowledgement		123456AD90	
vs.	1	vs.	7	vs.	9	vs.	4
polar		solarization		accomplishments		12457AC97	

Damerau–Levenshtein's distance [89]	similar to Levenshtein's distance but includes the additional operation: transposition of two adjacent string characters.
-------------------------------------	---

TABLE 42 - EXAMPLES OF DAMERAU-LEVENSHTAIN'S DISTANCE COMPUTATION: RED CHARACTERS INDICATE DIFFERING CHARACTERS

Example 1	Distance	Example 2	Distance	Example 3	Distance	Example 4	Distance
solar		solar		acknowledgement		123456AD90	
vs.	1	vs.	7	vs.	3	vs.	4
polar		solarization		acknowledgemnt		124357AC97	

Jaro's distance [90]	$d_{jaro}(x, y) = \frac{1}{3} \left(\frac{m}{ x } + \frac{m}{ y } + \frac{m-t}{m} \right)$
----------------------	---

where x and y are 2 strings,
 $|x|$ is the character size of string x ,
 t is the number of character transpositions
 m is the number of matching characters, with a position difference

$$\leq \left\lfloor \frac{\max(|x|, |y|)}{2} \right\rfloor - 1$$

TABLE 43 - EXAMPLES OF JARO'S DISTANCE COMPUTATION: RED CHARACTERS INDICATE DIFFERING CHARACTERS

Example 1	Distance	Example 2	Distance	Example 3	Distance	Example 4	Distance
solar	0.86667	solar	0.80556	acknowledgement	0.97778	123456AD90	0.78333
vs.		vs.		vs.		vs.	
polar		Solarization		acknowlegdemnt		124357AC97	

Jaro-Winkler's distance [91]

$$d_{jaro-winkler}(x, y) = d_{jaro}(x, y) + \left(l \cdot p \left(1 - d_{jaro}(x, y) \right) \right)$$

where x and y are 2 strings
 l is the prefix length, with $l \leq 4$
 p is a coefficient to foster strings with prefix. Winkler suggested $p = 0.1$

TABLE 44 - EXAMPLES OF JARO-WINKLER'S DISTANCE COMPUTATION WITH $l = 2$: RED CHARACTERS INDICATE DIFFERING CHARACTERS

Example 1	Distance	Example 2	Distance	Example 3	Distance	Example 4	Distance
Solar	0.89333	solar	0.84444	acknowledgement	0.98222	123456AD90	0.78766
vs.		vs.		vs.		vs.	
polar		Solarization		Aknowlegdemnt		124357AC97	

Jaccard's distance [95]

$$d_{jaccard}(x, y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|}$$

where x and y are 2 sample sets

Annex 3. Example of distance calculation: diversity or polymorphism degree applied to ISO/IEC/IEEE 25010 and ISO/IEC 9126 quality models

This annex presents the calculation of quality model distance based on degree of polymorphism, described in Chapter IV.6.c and defined by the equations 6 and 7, between the two consecutive standards for quality model in software engineering: ISO/IEC 9126 [24] and ISO/IEC/IEEE 25010 [23]. We note that a comparison between these two quality models are also available in the Annex A of ISO/IEC/IEEE 25010.

To initiate the calculation, we have to identify the sequences, usually linked to alleles in genetic field, and their frequencies. So, in the current case we consider two sequences, each of them directly associated to either ISO/IEC 9126 or ISO/IEC/IEEE 25010. Moreover, we assume that both model frequencies are identical, that is to say there is equal chance to use for software one quality model or the other. Thus, both frequencies are equal to 50%.

Next step is to enumerate the two sequences together (i.e., list their quality characteristics and quality sub-characteristics) and identify whether lexically and semantically they are identical, similar, or different. In this last situation, we mark this as a gap like gap in a DNA sequence. So, the enumeration combined with difference results are:

ISO/IEC/IEEE 25010 (Allele 1)	ISO/IEC 916 (Allele 2)	
System / Software product		
Functional suitability	Functionality	Similar
Performance efficiency	Efficiency	Similar
Compatibility	-	Gap
Usability	Usability	identical
Reliability	Reliability	identical
Security	-	Gap
Maintainability	Maintainability	identical
Portability	Portability	identical
Quality in use		
Effectiveness	Effectiveness	Identical
Efficiency	Productivity	Similar
Satisfaction	Satisfaction	Identical
Freedom from risk	Safety	Similar
Context coverage	-	Gap

System / Software product quality model part

ISO/IEC/IEEE 25010 (Allele 1)	ISO/IEC 916 (Allele 2)	
Functional suitability		
Functional completeness	Functionality Compliance	Similar
Functional correctness	Accuracy	Similar
Functional appropriateness	Suitability	Similar
-	Security	Gap
-	Interoperability	Gap
Performance efficiency		
Time-behavior	Time-behavior	Identical
Resource utilization	Resource utilization	Identical
Capacity	-	Gap
-	Efficiency Compliance	Gap
Compatibility		
Co-existence	-	Gap
Interoperability	-	Gap

Usability		
Appropriateness recognizability	Attractiveness	Similar
Learnability	Learnability	Identical
Operability	Operability	Identical
User error protection	-	Gap
User interface aesthetics	-	Gap
Accessibility	-	Gap
-	Understandability	Gap
-	Usability Compliance	Gap
Reliability		
Maturity	Maturity	Identical
Availability	-	Gap
Fault tolerance	Fault tolerance	Identical
Recoverability	Recoverability	Identical
-	Reliability Compliance	Gap
Security		
Confidentiality	-	Gap
Integrity	-	Gap
Non-repudiation	-	Gap
Accountability	-	Gap
Authenticity	-	Gap
Maintainability		
Modularity	-	Gap
Reusability	-	Gap
Analyzability	Analyzability	Identical
Modifiability	Changeability	Similar
Testability	Testability	Identical
-	Maintainability Compliance	Gap
-	Stability	Gap
Portability		
Adaptability	Adaptability	Identical
Instalability	Instalability	Identical
Replaceability	Replaceability	Identical
-	Portability Compliance	Gap
-	Co-existence	Gap

Quality in use model part

ISO/IEC/IEEE 25010 (Allele 1)	ISO/IEC 916 (Allele 2)	
Effectiveness		
Effectiveness	Effectiveness	Identical
Efficiency		
Efficiency	Productivity	Similar
Satisfaction		
Usefulness	-	Gap
Trust	-	Gap
Pleasure	Satisfaction	Similar
Comfort	-	Gap
Freedom from risk		
Economic risk mitigation	-	Gap
Health and safety risk mitigation	Safety	Similar
Environmental risk mitigation	-	Gap

Context coverage		
Context completeness	-	Gap
Flexibility	-	Gap

If we resume the findings of the enumeration with difference per characteristics, we can determine the π_{ij} of each characteristic based on the numbers of similar, identical, and total sub-characteristics. Furthermore, we can estimate the numbers of similar, identical, and total sub-characteristics for the two sequences, as well as the corresponding π_{ij} ; this is shown in “**Global SUM**” line of the below table.

	Gap	Similar	Identical	Total	π_{ij}
Functional suitability	2	3	0	5	0.7000
Performance efficiency	2	0	2	4	0.5000
Compatibility	2	0	0	2	1.0000
Usability	5	1	2	8	0.6875
Reliability	2	0	3	5	0.4000
Security	5	0	0	5	1.0000
Maintainability	4	1	2	7	0.6429
Portability	2	0	3	5	0.4000
SUM	24	5	12	41	0.6463
Effectiveness	0	0	1	1	0.0000
Efficiency	0	1	0	1	0.5000
Satisfaction	3	1	0	4	0.8750
Freedom from risk	2	1	0	3	0.8333
Context coverage	2	0	0	2	1.0000
SUM	7	3	1	11	0.7727
Global SUM	31	8	13	52	0.6731

The final step is then to calculate the degree of polymorphism π between these two software quality models through equation 6, so we have:

$$\hat{\pi} = \frac{n}{(n-1)} \sum_{ij} x_i x_j \pi_{ij} = \frac{2}{(2-1)} * \left(0.5 * 0.5 * \frac{31 + 0.5 * 8}{52} \right) * \left(0.5 * 0.5 * \frac{31 + 0.5 * 8}{52} \right) = \mathbf{0.673076923}$$

In conclusion, the degree of polymorphism between ISO/IEC 9126 and ISO/IEC/IEEE 25010 is ~ 0.6731 , which means that the two quality models are $\sim 67.31\%$ different.

Annex 4. Measurement process key document: Evaluation plan template

1 Context																								
1.1 Purpose																								
<i>This chapter should provide the purpose of the quality evaluation plan, introducing the context and situation of the evaluation.</i>																								
1.2 Audience																								
<i>Organizations involved in the evaluation, such as the independent evaluation organization, product developers and acquirer's organizational units.</i>																								
1.3 Intended																								
<i>This chapter should provide information products expected from the evaluation.</i>																								
5.1.4 Definition of responsibilities																								
<i>This chapter should define all responsibilities associated with the implementation of the Plan. This includes system(s) and/or software quality requirements specification, all data collection, analysis tasks, implementation of other supporting requirements, reporting, follow up and similar requirements. [ISO/IEC 25001]</i>																								

2 Objectives

2.1 Evaluation Objectives

This chapter should provide a clear statement about the objective(s) of the evaluation and the intended application of the system(s) or software. This can be stated in terms of business needs. However, they should be useable for the purpose of specifying quality requirements and setting quality objectives and respective criteria. [ISO/IEC 25001]

2.2 Quality Objectives

This chapter should provide quantifiable quality objectives (target values), which are verified against values measured at interim or final phases of the project development. [ISO/IEC 25001]

2.3 List of priorities

This chapter should prioritize the above characteristics and should provide a supporting rationale for these priorities. [ISO/IEC 25001]

2.4 Level of evaluation and compromises

This chapter should address questions such as accuracy and precision level, dedicated labor effort, repeatability and generalization, or any acceptable vs non acceptable compromises/ trade off

3 Evaluation Elements

3.1 Systems and software quality requirements and applicable quality characteristics

This chapter should provide statements of the quality characteristics (e.g. ISO/IEC 25010) resulting from the specification of system(s) or software quality requirements, which support the objectives prescribed in 5.2.1 "Evaluation Objectives".

NOTE: *The stated quality objectives may be both product and process oriented. The purpose of this plan is to address the product quality objectives only. [ISO/IEC 25001]*

3.2 Evaluation design

This chapter should define the measurements, which are planned to be carried out and cover required scope of quality evaluation. The chapter should indicate at what phase(s) of the development cycle these measurements are to be carried out, what evaluation process should be applied (from ISO/IEC 25041), how often they should be repeated, what techniques or tools should be used to aid data capture and analysis, and what actions should be undertaken if there are divergences from the stated objectives. [ISO/IEC 25001]

4 Execution Plan

4.1 Evaluation planning and execution

This chapter should provide a clear plan of activities with milestones and stated deliverables. [ISO/IEC 25001]

4.2 Using and analyzing data

This chapter should define how data is to be analysed, what, if any, statistical methods are to be employed and what presentation techniques are to be used. It should make references to previously stated responsibilities, supporting tools and forms. It should also state how the information is to be integrated into the progress tracking process or into the product acceptance process. [ISO/IEC 25001]. With regard to ISO2626: for ASIL A and greater we must have process improvement. Therefore forecasted values (e.g. at the end of the development) should also be given, considering the ratio obtained when comparing the planned measurement values to the current values.

4.3 Reporting

This chapter should define all relevant reporting requirements. [ISO/IEC 25001]

5 Other Requirements

This chapter should include requirements not covered previously, e.g. it can include the following information: [ISO/IEC 25001]

5.1 Techniques and methods employed

Provide a full description (or references to other material) of the techniques and methods used, (e.g. method for sizing; development maturity assessment; inspection method for error detection; defect removal model for predicting error rates). [ISO/IEC 25001]

5.2 Supporting tools

Describe or provide requirements and references for the supporting tools. This can include guides for the use of databases, spreadsheet and statistical packages. [ISO/IEC 25001]

5.3 Supporting trainings

Describe or provide requirements and references for the supporting trainings

5.4 Relevant standards and guides

Refer to applicable standards and supporting guides. Describe their use and benefits relevant to the systems and software product quality requirements and evaluation processes (e.g. ISO/IEC 25000; ISO 9001; ISO/IEC 90003). [ISO/IEC 25001]

5.5 Suppliers' evaluation

Include evaluation and measurement procedures for the effective quantitative assessment of the systems or software product suppliers. This can cover the number of released copies, current error status, surveys about post installation support performance, statistics about past and current users' satisfaction, management [ISO/IEC 25001]

Annex 5. Systematic literature review results

TABLE 45 - LIST OF FILTERED PUBLISHED STUDIES RESULTING THE SYSTEMATIC LITERATURE REVIEW

Id	Year	Ref.	Source ¹⁷	Type Citations		Authors	Title	Uri ²⁰
				¹⁸	¹⁹			
SLR-S01	1979	[195]	ACM digital library	J	94	Siba N. Mohanty	Models and Measurements for Quality Assessment of Software	https://dl.acm.org/doi/10.1145/356778.356783
SLR-S02	1994	[264]	IEEE Xplore	C	18	T. M. Khoshgoftaar R. M. Szabo	Improving neural network predictions of software quality using principal components analysis	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=374764
SLR-S03	1996	[196]	ACM digital library	J	507	William Frakes Carol Terry	Software Reuse: Metrics and Models	https://dl.acm.org/doi/10.1145/234528.234531
SLR-S04	1996	[265]	Scopus	J	35	Khoshgoftaar T.M. Allen E.B. Kalaichelvan K.S. Goel N.	The impact of software evolution and reuse on software quality	https://www.scopus.com/inward/record.uri?eid=2-s2.0-0029726553&doi=10.1007%2fBF00125810&partnerID=40&md5=22f6cf0451a62716aa68a29a14e0ca89
SLR-S05	1996	[26]	IEEE Xplore	J	230	B. Kitchenham S.L. Pfleeger	Software quality: the elusive target [special issues section]	https://ieeexplore.ieee.org/document/476281
SLR-S06	1997	[266]	Scopus	J	39	Dahlberg T. Jarvinen J.	Challenges to IS quality	https://www.scopus.com/inward/record.uri?eid=2-s2.0-0031381685&doi=10.1016%2fS0950-5849%2897%2900039-6&partnerID=40&md5=79fa87af5c97be907d56f64dc820839f
SLR-S07	1997	[267]	Web of Science	J	63	Granja Alvarez, JC Barranco Garcia, MJ	A method for estimating maintenance cost in a software project: A case study	https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SIC%291096-908X%28199705%299%3A3%3C161%3A%3AAID-SMR148%3E3.0.CO%3B2-8
SLR-S08	1998	[268]	IEEE Xplore	C	44	T. M. Khoshgoftaar E. B. Allen A. Naik W. D. Jones J. Hudepohl	Using classification trees for software quality models: lessons learned	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=731598
SLR-S09	1998	[269]	Web of Science	C	14	Yokoyama, Y Kodaira, M	Software cost and quality analysis by statistical approaches	https://ieeexplore.ieee.org/document/671607
SLR-S10	1999	[197]	IEEE Xplore	C	145	T. M. Khoshgoftaar E. B. Allen W. D. Jones J. I. Hudepohl	Classification tree models of software quality over multiple releases	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=809316
SLR-S11	1999	[198]	Scopus	J	137	Olsina L. Godoy D. Lafuente G. Rossi G.	Assessing the quality of academic websites: A case study	https://www.scopus.com/inward/record.uri?eid=2-s2.0-0033339710&doi=10.1080%2f13614569908914709&partnerID=40&md5=48291fac8c23aabaaa12f022d99a8855
SLR-S12	2000	[270]	IEEE Xplore	C	30	T. M. Khoshgoftaar Ruqun Shan E. B. Allen	Improving tree-based models of software quality with principal components analysis	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=885872
SLR-S13	2001	[199]	IEEE Xplore	C	169	Ping Zhang G. von Dran	Expectations and rankings of Web site quality features: results of two studies on user perceptions	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=927050

¹⁷ Source: digital libraries for the main systematic literature review stream, or study found during exploratory review, or further manual search.

¹⁸ Type of document: C = Conference paper, J = Journal paper

¹⁹ Study citations are retrieved from google scholar for harmonization data despite the digital library source.

²⁰ Main url to access to online paper

SLR-S14	2001	[271]	Scopus	J	19	Khoshgoftaar T.M. Allen E.B. Jones W.D. Hudepohl J.P.	Cost-Benefit Analysis of Software Quality Models	https://www.scopus.com/inward/record.uri?eid=2-s2.0-0038138707&doi=10.1023%2fA%3a1016621219262&partnerID=40&md5=e236c7fd7357975ed5580c2801139eb4
SLR-S15	2002	[272]	IEEE Xplore	C	14	Hong Zhu Yanlong Zhang Qingning Huo S. Greenwood	Application of hazard analysis to software quality modelling	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1044544
SLR-S16	2002	[273]	IEEE Xplore	C	5	C. V. Ramamoorthy	Evolution and evaluation of software quality models	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1180850
SLR-S17	2002	[200]	Scopus	J	228	Briand L.C. Wüst J.	Empirical studies of quality models in object-oriented systems	https://www.scopus.com/inward/record.uri?eid=2-s2.0-77957159740&doi=10.1016%2fS0065-2458%2802%2980005-5&partnerID=40&md5=1256de9aefbf034462dacc6007f42689
SLR-S18	2002	[274]	Web of Science	J	17	Mendoza, LE Griman, AC Perez, MA Rojas, T	Evaluation of environments for portals development: A case study	https://www.tandfonline.com/doi/abs/10.1201/1078/43200.19.2.20020228/35141.7
SLR-S19	2003	[275]	IEEE Xplore	C	22	T. M. Khoshgoftaar E. Geleyn L. Nguyen	Empirical case studies of combining software quality classification models	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1319084
SLR-S20	2004	[276]	IEEE Xplore	C	47	Yi Liu T. Khoshgoftaar	Reducing overfitting in genetic programming models for software quality classification	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1281730
SLR-S21	2004	[277]	Web of Science	C	5	Oh, J Park, D Lee, B Lee, J Hong, E Wu, C	Certification of software packages using hierarchical classification	https://link.springer.com/chapter/10.1007/978-3-540-24675-6_17
SLR-S22	2005	[278]	Scopus	J	32	Côté M.-A. Suryan W. Laporte C.Y. Martin R.A.	The evolution path for industrial software quality evaluation methods applying ISO/IEC 9126:2001 quality model: Example of MITRE's SQA method	https://www.scopus.com/inward/record.uri?eid=2-s2.0-17444388547&doi=10.1007%2f11219-004-5259-6&partnerID=40&md5=09ec05c221610b4c799ab6111def6568
SLR-S23	2005	[148]	Manual search	J	24	S. Khaddaj G. Horgan	A Proposed Adaptable Quality Model for Software Quality Assurance	https://www.semanticscholar.org/paper/A-Proposed-Adaptable-Quality-Model-for-Software-Khaddaj-Horgan/815debd01444997d7d20656204be767d7296d660
SLR-S24	2006	[279]	IEEE Xplore	C	6	Q. Zhang J. Wu H. Zhu	Tool Support to Model-based Quality Analysis of Software Architecture	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4020069
SLR-S25	2006	[280]	Scopus	J	26	Mishra A. Mishra D.	Software quality assurance models in small and medium organisations: A comparison	https://www.scopus.com/inward/record.uri?eid=2-s2.0-33845719178&doi=10.1504%2fIJITM.2006.008710&partnerID=40&md5=42359aa8dd07631d8be7f435c7e1f634
SLR-S26	2006	[281]	Scopus	C	11	Jung H.-J. Jung W.-T. Yang H.-S.	A study on the standard of software quality testing	https://www.scopus.com/inward/record.uri?eid=2-s2.0-33745922863&doi=10.1007%2f11751632_113&partnerID=40&md5=535bc40cf6bbac584b698ede7532b4c5
SLR-S27	2006	[282]	Scopus	J	0	Wang C.-T. Lo C.-C. Jean T.-F.	Probabilistic models for software quality analysis	https://www.scopus.com/inward/record.uri?eid=2-s2.0-33746862135&doi=10.1080%2f10170660609509329&partnerID=40&md5=41cc8711f0c0c0eba87345026edcf678
SLR-S28	2006	[283]	Web of Science	C	0	Ruiz, Julian Calero, Coral Piattini, Mario	Web metrics selection through a practitioners' survey	https://www.semanticscholar.org/paper/Web-metrics-selection-through-a-practitioners'-Ruiz-Calero/cf756d83396f40005ff49d4ffbf3f467812472c

Annexes

SLR-S29	2006	[125]	Web of Science	C	0	Khoshgoftaar, TM Szabo, RM	A Poisson regression model of software quality: A comparative study	https://www.worldscientific.com/doi/10.1142/9789812707147_0007
SLR-S30	2006	[201]	Manual search	J	148	A. Rawashdeh B. Matakah	A New Software Quality Model for Evaluating COTS Components	https://www.semanticscholar.org/paper/A-New-Software-Quality-Model-for-Evaluating-COTS-Rawashdeh-Matakah/579ceadfed8bc605961742fb254bcc938927843a
SLR-S31	2007	[284]	ACM digital library	J	55	Olga Ormandjieva Ishrar Hussain Leila Kosseim	Toward a Text Classification System for the Quality Assessment of Software Requirements Written in Natural Language	https://dl.acm.org/doi/10.1145/1295074.1295082
SLR-S32	2007	[285]	IEEE Xplore	C	17	P. L. Roden S. Virani L. H. Etzkorn S. Messimer	An Empirical Study of the Relationship of Stability Metrics and the QMOOD Quality Models Over Software Developed Using Highly Iterative or Agile Software Processes	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4362911
SLR-S33	2007	[286]	IEEE Xplore	C	48	Y. Ma B. Cukic	Adequate and Precise Evaluation of Quality Models in Software Engineering Studies	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4273257
SLR-S34	2007	[287]	IEEE Xplore	C	37	S. Neti H. A. Muller	Quality Criteria and an Analysis Framework for Self-Healing Systems	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4228606
SLR-S35	2008	[202]	ACM digital library	J	263	Rüdiger Lincke Jonas Lundberg Welf Löwe	Comparing Software Metrics Tools	https://dl.acm.org/doi/10.1145/1390630.1390648
SLR-S36	2008	[288]	IEEE Xplore	C	7	L. Zhang L. Li H. Gao	2-D Software Quality Model and Case Study in Software Flexibility Research	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5172787
SLR-S37	2008	[289]	IEEE Xplore	C	2	A. A. Hamada M. N. Moustafa H. I. Shaheen	Software Quality model Analysis Program	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4773015
SLR-S38	2008	[290]	IEEE Xplore	C	1	X. Feng Y. Liu	A Study on Evaluation Model of Information Sharing Quality in Virtual Teams	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4722856
SLR-S39	2008	[291]	IEEE Xplore	C	9	O. Alfonzo K. Domínguez L. Rivas M. Pérez L. Mendoza M. Ortega	Quality Measurement Model for Analysis and Design Tools Based on FLOSS	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4483214
SLR-S40	2008	[292]	IEEE Xplore	C	6	M. Bombardieri F. A. Fontana	A specialisation of the SQuaRE quality model for the evaluation of the software evolution and maintenance activity	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4686328
SLR-S41	2009	[293]	IEEE Xplore	C	10	H. P. Breivold I. Crnkovic	Analysis of Software Evolvability in Quality Models	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5349964
SLR-S42	2009	[98]	IEEE Xplore	C	32	S. Wagner K. Lochmann S. Winter A. Goeb M. Kläs	Quality models in practice: A preliminary analysis	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5316003
SLR-S43	2009	[294]	IEEE Xplore	C	8	F. Khomh	SQUAD: Software Quality Understanding through the Analysis of Design	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5328744

SLR-S44	2009	[10]	IEEE Xplore	C	42	M. Kläs J. Heidrich J. Münch A. Trendowicz	CQML Scheme: A Classification Scheme for Comprehensive Quality Model Landscapes	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5349845
SLR-S45	2009	[203]	Scopus	J	160	Mohagheghi P. Dehlen V. Nepel T.	Definitions and approaches to model quality in model-based software development - A review of literature	https://www.scopus.com/inward/record.uri?eid=2-s2.0-70349567623&doi=10.1016%2fj.infsoc.2009.04.004&partnerID=40&md5=8a2fb74f0c7b611214be792e6bb810b1
SLR-S46	2009	[295]	Scopus	C	31	Brcina R. Bode S. Riebisch M.	Optimisation process for maintaining evolvability during software evolution	https://www.scopus.com/inward/record.uri?eid=2-s2.0-67650330218&doi=10.1109%2fECSB.2009.20&partnerID=40&md5=37457feb9b3cc283cf5f63de0710b7bd
SLR-S47	2010	[296]	IEEE Xplore	C	3	Qi Yu-dong Zhu Ai-hong Xie Xiao-fang Yan Xiao-bin	Analysis of contribution of conceptual model quality to software reliability	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5622740
SLR-S48	2010	[297]	IEEE Xplore	C	62	J. Letouzey T. Coq	The SQALE Analysis Model: An Analysis Model Compliant with the Representation Condition for Assessing the Quality of Software Source Code	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5617180
SLR-S49	2010	[298]	IEEE Xplore	C	26	R. Lincke T. Gutzmann W. Löwe	Software Quality Prediction Models Compared	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5562947
SLR-S50	2010	[207]	Manual search	C	70	E. Petrinja A. Sillitti G. Succi	Comparing OpenBRR, QSOS, and OMM Assessment Models	https://hal.inria.fr/hal-01056052/
SLR-S51	2011	[299]	IEEE Xplore	C	3	E. Chandra D. Francis Xavier Christopher S. D. Vijaykumar	Study of CMMI based process framework for quality models	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6169109
SLR-S52	2011	[97]	IEEE Xplore	C	30	M. Kläs C. Lampasona J. Münch	Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6068366
SLR-S53	2011	[185]	Scopus	C	54	Al-Badareen A.B. Selamat M.H. A. Jabar M. Din J. Turaev S.	Software quality models: A comparative study	https://www.scopus.com/inward/record.uri?eid=2-s2.0-79960369917&doi=10.1007%2f978-3-642-22170-5_4&partnerID=40&md5=7e35cefc32c4677e451f68a330a436c9
SLR-S54	2011	[300]	Scopus	C	6	Coq T. Rosen J.-P.	The SQALE quality and analysis models for assessing the quality of Ada source code	https://www.scopus.com/inward/record.uri?eid=2-s2.0-79960266130&doi=10.1007%2f978-3-642-21338-0_5&partnerID=40&md5=4bd77c3b9f3b5d7b424d295f1dd2d563
SLR-S55	2011	[301]	Scopus	C	15	Lochmann K. Heinemann L.	Integrating quality models and static analysis for comprehensive quality assessment	https://www.scopus.com/inward/record.uri?eid=2-s2.0-79959854657&doi=10.1145%2f1985374.1985378&partnerID=40&md5=ddd65225476a4c260011649f9c9a8066
SLR-S56	2011	[302]	Scopus	J	44	Nabil D. Mosad A. Hefny H.A.	Web-Based Applications quality factors: A survey and a proposed conceptual model	https://www.scopus.com/inward/record.uri?eid=2-s2.0-83555173273&doi=10.1016%2fj.eij.2011.09.003&partnerID=40&md5=6fb780a59c13bbd0dbbb2bc29caa2958
SLR-S57	2011	[303]	Scopus	J	80	Montagud S. Abrahão S. Insfran E.	A systematic review of quality attributes and measures for software product lines	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84865626740&doi=10.1007%2fs11219-011-9146-7&partnerID=40&md5=33b25cd0a25fc09685e06cbc9c988f14

Annexes

SLR-S58	2011	[187]	Manual search	J	37	R. Polillo	Quality Models for Web [2.0] Sites: A Methodological Approach and a Proposal	https://link.springer.com/chapter/10.1007/978-3-642-27997-3_25
SLR-S59	2012	[48]	Exploratory review	C	18	A. Mayr R. Plösch M. Kläs C. Lampasona, and M. Saft,	A Comprehensive Code-Based Quality Model for Embedded Systems: Systematic Development and Validation by Industrial Projects	https://ieeexplore.ieee.org/document/6405376
SLR-S60	2012	[304]	IEEE Xplore	C	3	L. Cheikhi A. Abran J. Desharnais	Analysis of the ISBSG software repository from the ISO 9126 view of software product quality	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6389405
SLR-S61	2012	[305]	IEEE Xplore	C	6	K. Lochmann D. M. Fernandez S. Wagner	A Case Study on Specifying Quality Requirements Using a Quality Model	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6462713
SLR-S62	2012	[306]	IEEE Xplore	C	14	M. Galster P. Avgeriou	Qualitative Analysis of the Impact of SOA Patterns on Quality Attributes	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6319243
SLR-S63	2012	[307]	IEEE Xplore	C	5	H. Wan-jiang L. Tian-bo	Study on quality evaluation model of communication system	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6340726
SLR-S64	2012	[308]	IEEE Xplore	C	44	G. A. Garcia-Mireles M. Angeles Moraga F. Garcia	Development of maturity models: A systematic literature review	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6272525
SLR-S65	2012	[309]	Scopus	C	1	Jeong H.-J. Hong S.-J.	The survey of quality model for software and system	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84255199062&doi=10.1007%2f978-94-007-2792-2_54&partnerID=40&md5=790805eba6ac067ee6871ed520a8804f
SLR-S66	2012	[310]	Manual search	J	48	S. K. Dubey S. Ghosh A. Rana	Comparison of Software Quality Models: An Analytical Approach	https://www.semanticscholar.org/paper/Comparison-of-Software-Quality-Models%3A-An-Approach-Dubey-Ghosh/2fae30444bbd03a682753b75ddd8adcbe5350feb
SLR-S67	2012	[11]	Manual search	J	19	S. S. Thapar P. Singh S. Rani	Challenges to the Development of Standard Software Quality Model	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.6465&rep=rep1&type=pdf
SLR-S68	2013	[69]	Exploratory review	C	14	Oliveira, L. B. R. Guessi, M. Feitosa, D. Manteuffel, C. Galster, M. Oquendo, F. Nakagawa, E. Y.	An investigation on quality models and quality attributes for embedded systems	https://www.semanticscholar.org/paper/An-Investigation-on-Quality-Models-and-Quality-for-Oliveira-Guessi/7eff018a46d1be728ea101ca246ec06c0871f6ca
SLR-S69	2013	[204]	ACM digital library	C	129	Kyriakos Kritikos Barbara Pernici Pierluigi Plebani Cinzia Cappiello Marco Comuzzi Salima Benrernou Ivona Brandic Attila Kertész Michael Parkin Manuel Carro	A Survey on Service Quality Description	https://dl.acm.org/doi/10.1145/2522968.2522969
SLR-S70	2013	[311]	IEEE Xplore	C	25	B. Singh S. P. Kannoja	A Review on Software Quality Models	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6524514

SLR-S71	2013	[312]	IEEE Xplore	C	57	C. Calero M. F. Bertoa M. A. Moraga	A systematic literature review for software sustainability measures	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6606421
SLR-S72	2013	[313]	IEEE Xplore	C	4	A. Adewumi N. Omoregbe S. Misra L. Fernandez	Quantitative Quality Model for Evaluating Open Source Web Applications: Case Study of Repository Software	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6755361
SLR-S73	2013	[314]	IEEE Xplore	C	45	K. Li J. Xiao Y. Wang Q. Wang	Analysis of the Key Factors for Software Quality in Crowdsourcing Development: An Empirical Study on TopCoder.com	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6649922
SLR-S74	2013	[315]	IEEE Xplore	C	8	M. Ericsson W. Lowe T. Olsson D. Toll A. Wingkvist	A Study of the Effect of Data Normalization on Software and Information Quality Assessment	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6754351
SLR-S75	2013	[316]	IEEE Xplore	C	4	H. K. A. Bakar R. Razali	A preliminary review of legacy information systems evaluation models	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6716728
SLR-S76	2013	[317]	Scopus	C	16	Hegedüs P.	A probabilistic quality model for C# -an industrial case study	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84885225033&doi=10.14232%2factacyb.21.1.2013.10&partnerID=40&md5=4d29f1e00f36804bac31cdf6f995f34d
SLR-S77	2013	[260]	Web of Science	C	35	Adewumi, Adewole Misra, Sanjay Omoregbe, Nicholas	A Review of Models for Evaluating Quality in Open Source Software	https://www.sciencedirect.com/science/article/pii/S2212667813000178
SLR-S78	2013	[318]	Manual search	J	3	A. B. Tomar V. M. Thakare	A Customized Model on Software Quality Assurance & Reuse	https://www.semanticscholar.org/paper/A-Customized-Model-on-Software-Quality-Assurance-%26-Tomar-Thakare/a3ff1ff3517fc8fb4670ba1e24f5f53c3459594d
SLR-S79	2014	[319]	ACM digital library	J	2	Shrinath Gupta Himanshu Kumar Singh	A Semiautomated Method for Classifying Program Analysis Rules into a Quality Model	https://dl.acm.org/doi/10.1145/2597008.2597808
SLR-S80	2014	[320]	ACM digital library	J	2	Rinkaj Goyal Pravin Chandra Yogesh Singh	Why Interaction Between Metrics Should Be Considered in the Development of Software Quality Models: A Preliminary Study	https://dl.acm.org/doi/10.1145/2632434.2659853
SLR-S81	2014	[321]	IEEE Xplore	C	5	T. Davuluru J. Medida V. S. K. Reddy	A study of software quality models	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7012958
SLR-S82	2014	[322]	IEEE Xplore	C	19	S. Ouhbi A. Idri J. L. F. Aleman A. Toval	Evaluating Software Product Quality: A Systematic Mapping Study	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7000093
SLR-S83	2014	[323]	Scopus	J	1	Zhu H. Zhang Q. Zhang Y.	HASARD: A Model-Based Method for Quality Analysis of Software Architecture	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84942245243&doi=10.1016%2fb978-0-12-417009-4.00005-3&partnerID=40&md5=5dd4f34512f800291a453d8b8c6b4f8e
SLR-S84	2014	[324]	Scopus	C	4	Yildiz E. Bilgen S. Tokdemir G. Cagiltay N.E. Erturan Y.N.	Analysis of B2C mobile application characteristics and quality factors based on ISO 25010 quality model	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84906751837&doi=10.1007%2f978-3-319-10359-4_21&partnerID=40&md5=607063f7979ec7d03c477248d68647bf

Annexes

SLR-S85	2014	[325]	Scopus	J	2	Ronchieri E. Canaparo M. Salomoni D.	A software quality model by using discriminant analysis predictive technique	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84922863544&doi=10.3233%2fjid-2014-0016&partnerID=40&md5=d74f98344fedb535c2e0506a8ae96112
SLR-S86	2014	[326]	Scopus	C	3	Cherfi S.S.-S. Tuan A.D. Comyn-Wattiau I.	An exploratory study on websites quality assessment	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84919692097&partnerID=40&md5=5b6330cd813306c387af6dce524ca189
SLR-S87	2014	[327]	Scopus	J	58	Sarrab M. Rehman O.M.H.	Empirical study of open source software selection for adoption, based on software quality characteristics	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84892601718&doi=10.1016%2fj.advengsoft.2013.12.001&partnerID=40&md5=102b2e3d948a33890ff3d25521fd15c5
SLR-S88	2014	[92]	Scopus	C	40	Gordieiev O. Kharchenko V. Fominykh N. Sklyar V.	Evolution of software quality models in context of the standard ISO 25010	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84917706898&doi=10.1007%2f978-3-319-07013-1_21&partnerID=40&md5=1f0d9169e64a51c5d55af5b61f46eb2a
SLR-S89	2014	[328]	Scopus	C	0	Buglione L.	Software product quality: Some thoughts about its evolution and perspectives	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84918789815&partnerID=40&md5=c2ccd5c92585f827c4614c819db93b2d
SLR-S90	2014	[12]	Web of Science	J	83	Oriol, Marc Marco, Jordi Franch, Xavier	Quality models for web services: A systematic mapping	https://www.sciencedirect.com/science/article/abs/pii/S0950584914000822
SLR-S91	2014	[329]	Web of Science	C	33	Gupta, Deepak Ahlawat, Anil Sagar, Kalpna	A Critical Analysis of A Hierarchy Based Usability Model	https://ieeexplore.ieee.org/document/7019810
SLR-S92	2014	[330]	Manual search	J	28	S. Manoj Wadhwa	A Comparative Study of Software Quality Models	http://ijcsit.com/docs/Volume%205/vol5issue04/ijcsit20140504177.pdf
SLR-S93	2014	[208]	Manual search	J	8	A. Fath-Allah L. Cheikhi R. E. Al-Qutaish A. Idri	A Comparative Analysis of E-Government Quality Models	https://www.semanticscholar.org/paper/A-Comparative-Analysis-of-E-Government-Quality-Fath-Allah-Cheikhi/b1c5dff0d65d1db8c81cafbd30a2a209651434b4
SLR-S94	2014	[190]	Manual search	J	171	J. P. Miguel D. Mauricio G. Rodriguez	A Review of Software Quality Models for the Evaluation of Software Products	https://www.semanticscholar.org/paper/A-Review-of-Software-Quality-Models-for-the-of-Miguel-Mauricio/0f6f7c5eaa44279c5208fa14f5489f710ecf2683
SLR-S95	2015	[71]	IEEE Xplore	C	28	T. Bianchi D. S. Santos K. R. Felizardo	Quality Attributes of Systems-of-Systems: A Systematic Literature Review	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7179220
SLR-S96	2015	[331]	IEEE Xplore	C	13	K. Sheoran O. P. Sangwan	An Insight of software quality models applied in predicting software quality attributes: A comparative analysis	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7359355
SLR-S97	2015	[332]	IEEE Xplore	C	3	Hegedus, Peter	Advances in software product quality measurement and its applications in software evolution	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7332520
SLR-S98	2015	[333]	Scopus	C	9	Chawla M.K., Chhabra I.	SQMMA: Software quality model for maintainability analysis	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84959474131&doi=10.1145%2f2835043.2835062&partnerID=40&md5=131c1f1bcb41c56c81a08c011d12596d
SLR-S99	2015	[93]	Scopus	C	13	Gordieiev O. Kharchenko V. Fusani M.	Evolution of software quality models: Green and reliability issues	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84930363322&partnerID=40&md5=32efb2b85551118b519e0ca8d176b314

SLR-S100	2015	[192]	Scopus	J	3	Buglione L.	Some thoughts on quality models: Evolution and perspectives	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84944450337&partnerID=40&md5=5b4de18eb17b941f18d3400811e14eef
SLR-S101	2015	[334]	Web of Science	C	0	Imeri, Florinda Antovski, Ljupcho Hamiti, Mentor	Empirical Analysis of Quality Models in Practice in Small IT Companies in SEE Region	https://www.sciencedirect.com/science/article/pii/S1877042815027500
SLR-S102	2016	[335]	Web of Science	J	4	Ganser, Andreas Lichter, Horst Roth, Alexander Rumpe, Bernhard	Staged model evolution and proactive quality guidance for model libraries	https://link.springer.com/article/10.1007/s11219-015-9298-y
SLR-S103	2016	[336]	IEEE Xplore	C	12	M. A. Kabir M. U. Rehman S. I. Majumdar	An analytical and comparative study of software usability quality factors	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7883188
SLR-S104	2016	[337]	IEEE Xplore	C	3	D. d. Ruscio D. S. Kolovos Y. Korkontzelos N. Matragkas J. Vinju	Supporting Custom Quality Models to Analyse and Compare Open-Source Software	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7814523
SLR-S105	2016	[338]	IEEE Xplore	C	4	Z. Qian C. Wan Y. Chen	Evaluating quality-in-use of FLOSS through analyzing user reviews	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7515956
SLR-S106	2016	[339]	IEEE Xplore	C	0	L. S. P. Silva S. C. B. Sampaio E. R. d. Souza R. T. Moreira A. M. L. Vasconcelos	Mapping between the Guide of IT Solution Contract and CMMI Models: A Qualitative Analysis	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7814535
SLR-S107	2016	[340]	Scopus	C	3	Devi U. Sharma A. Kesswani N.	A review on quality models to analyse the impact of refactored code on maintainability with reference to software product line	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84997503403&partnerID=40&md5=c1cc2086115f825aa0f6e2a0cf776b4e
SLR-S108	2016	[341]	Scopus	J	4	Pattnaik S. Pattanayak B.K.	A survey on machine learning techniques used for software quality prediction	https://www.inderscience.com/info/inarticle.php?articleid=80058
SLR-S109	2016	[93]	Scopus	C	5	Gordiev O. Kharchenko V. Fusani M.	Software quality standards and models evolution: Greenness and reliability issues	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84960404079&doi=10.1007%2f978-3-319-30246-1_3&partnerID=40&md5=7f0c7074dbbceb0e60779c8e1a7d67f2
SLR-S110	2016	[342]	Web of Science	J	14	Bezerra, Carla I. M. Andrade, Rossana M. C. Monteiro, Jose Maria	Exploring quality measures for the evaluation of feature models: a case study	https://www.sciencedirect.com/science/article/abs/pii/S0164121216301340
SLR-S111	2016	[189]	Web of Science	J	26	Adewumi, Adewole Misra, Sanjay Omogbe, Nicholas Crawford, Broderick Soto, Ricardo	A systematic literature review of open source software quality assessment models	https://springerplus.springeropen.com/articles/10.1186/s40064-016-3612-4
SLR-S112	2016	[343]	Web of Science	C	6	Santos, Mariana Afonso, Paulo Junior Bermejo, Paulo Henrique Costa, Heitor	Metrics and Statistical Techniques Used to Evaluate Internal Quality of Object-Oriented Software: A Systematic Mapping	https://ieeexplore.ieee.org/document/7836021
SLR-S113	2016	[344]	Web of Science	C	0	Suradi, Nur Razia Mohd Kahar, Saliyah Jamaludin, Nor Azliana Akmal	A Review on Software Quality Attributes for Web-Based Application	https://www.researchgate.net/publication/311453941_A_Review_On_Software_Quality_Attributes_for_Web-based_Application

Annexes

SLR-S114	2017	[345]	IEEE Xplore	C	11	C. Izurieta I. Griffith C. Huvaere	An Industry Perspective to Comparing the SQALE and Quamoco Software Quality Models	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8170115
SLR-S115	2017	[346]	IEEE Xplore	C	1	K. Moumane A. Idri	Software quality in mobile environments: A comparative study	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8102750
SLR-S116	2017	[206]	IEEE Xplore	C	6	M. Yan X. Xia X. Zhang L. Xu D. Yang	A Systematic Mapping Study of Quality Assessment Models for Software Products	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8118522
SLR-S117	2017	[347]	Scopus	J	12	Wahyuningrum T. Mustofa K.	A systematic mapping review of software quality measurement: Research trends, model, and method	https://www.scopus.com/inward/record.uri?eid=2-s2.0-85050120161&doi=10.11591%2fijece.v7i5.pp2847-2854&partnerID=40&md5=4d2e29c8dd5634ee56182bc4fc5b04aa
SLR-S118	2017	[55]	Scopus	J	46	Garcés L. Ampatzoglou A. Avgeriou P. Nakagawa E.Y.	Quality attributes and quality models for ambient assisted living software systems: A systematic mapping	https://www.scopus.com/inward/record.uri?eid=2-s2.0-84994350835&doi=10.1016%2fj.infsof.2016.10.005&partnerID=40&md5=924f259ce9824d9727b190ab5c627919
SLR-S119	2018	[348]	Scopus	J	9	Giraldo F.D. España S. Pastor Ó. Giraldo W.J.	Considerations about quality in model-driven engineering: Current state and challenges	https://www.scopus.com/inward/record.uri?eid=2-s2.0-85006410514&doi=10.1007%2fs11219-016-9350-6&partnerID=40&md5=4586f43287e5bbaf7f7aeca584938bcf
SLR-S120	2018	[6]	Exploratory review	J	1	Zouheyr Tamrabet Toufik Marir Farid MOKHATI	A Survey on Quality Attributes and Quality Models for Embedded Software	https://dl.acm.org/doi/abs/10.4018/IJERTCS.2018070101
SLR-S121	2018	[349]	ACM digital library	J	8	Daniel Russo Paolo Ciancarini and Tommaso Falasconi Massimo Tomasi	A Meta-Model for Information Systems Quality: A Mixed Study of the Financial Sector	https://dl.acm.org/doi/10.1145/3230713
SLR-S122	2018	[350]	IEEE Xplore	C	0	R. Wahdiniwaty E. B. Setiawan D. A. Wahab	Comparative Analysis of Software Quality Model In The Selection of Marketplace E-Commerce	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8696074
SLR-S123	2018	[351]	IEEE Xplore	C	1	D. Gatica F. Ponce R. Noël H. Astudillo	Characterizing Architectural Evaluations and Identifying Quality Attributes addressed in Systems-of-Systems: A Systematic Mapping Study	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8705229
SLR-S124	2018	[352]	IEEE Xplore	C	19	A. J. Abdellatif B. McCollum P. McMullan	Serious games: Quality characteristics evaluation framework and case study	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8340460
SLR-S125	2018	[191]	IEEE Xplore	C	2	O. Gordieiev V. Kharchenko	IT-oriented software quality models and evolution of the prevailing characteristics	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8409162
SLR-S126	2018	[353]	Scopus	J	4	Zighed N. Bounour N. Seriai A.-D.	Comparative Analysis of Object-Oriented Software Maintainability Prediction Models	https://www.scopus.com/inward/record.uri?eid=2-s2.0-85060063384&doi=10.1515%2ffcds-2018-0018&partnerID=40&md5=4be59f714c130c780d51af467729c0cd
SLR-S127	2018	[354]	Scopus	C	0	Rai M. Virk K.S.	Software Component Quality Models: A Survey	https://www.scopus.com/inward/record.uri?eid=2-s2.0-85045842336&doi=10.1007%2f978-981-10-5903-2_27&partnerID=40&md5=9480facd0c4cd316df56fe7fb58abc

SLR-S128	2018	[355]	Web of Science	J	20	Nikolic, Vlastimir Kaljevic, Jelena Jovic, Srdan Petkovic, Dalibor Milovancevic, Milos Dimitrov, Ljubomir Dachkinov, Pancho	Survey of quality models of e-learning systems	https://www.sciencedirect.com/science/article/abs/pii/S0378437118309300
SLR-S129	2018	[356]	Web of Science	C	2	Mossakowska, Katarzyna Jarzebowicz, Aleksander	A Survey Investigating the Influence of Business Analysis Techniques on Software Quality Characteristics	https://link.springer.com/chapter/10.1007/978-3-319-65208-5_10
SLR-S130	2019	[77]	Exploratory review	C	0	S. Juneja A. Juneja R. Anand	Reliability Modeling for Embedded System Environment compared to available Software Reliability Growth Models	https://ieeexplore.ieee.org/document/8776814
SLR-S131	2019	[357]	IEEE Xplore	C	12	P. Nistala K. V. Nori R. Reddy	Software Quality Models: A Systematic Mapping Study	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8812848
SLR-S132	2019	[358]	IEEE Xplore	C	3	N. Condori-Fernandez P. Lago	Towards a Software Sustainability-Quality Model: Insights from a Multi-Case Study	https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8877084
SLR-S133	2019	[359]	Scopus	J	4	Arcos-Medina G. Mauricio D.	Aspects of software quality applied to the process of agile software development: a systematic literature review	https://www.scopus.com/inward/record.uri?eid=2-s2.0-85074445827&doi=10.1007%2fs13198-019-00840-7&partnerID=40&md5=3a8c86a837f49472a44dc60b848013e9
SLR-S134	2019	[193]	Scopus	J	6	Yan M. Xia X. Zhang X. Xu L. Yang D. Li S.	Software quality assessment model: a systematic mapping study	https://www.scopus.com/inward/record.uri?eid=2-s2.0-85069895120&doi=10.1007%2fs11432-018-9608-3&partnerID=40&md5=a2fbcdb0b67be249834e19658827f8c7
SLR-S135	2019	[205]	Web of Science	J	6	Gezici, Bahar Tarhan, Ayca Chouseinoglou, Oumout	Internal and external quality in the evolution of mobile software: An exploratory study in open-source market	https://www.sciencedirect.com/science/article/abs/pii/S0950584918301290
SLR-S136	2019	[94]	Manual search	C	1	S. Motogna D. Lupsa L. Ciuciu	A NLP Approach to Software Quality Models Evaluation	https://link.springer.com/chapter/10.1007/978-3-030-11683-5_24

TABLE 46 - MAIN RAW RESULTS FROM THE SYSTEMATIC LITERATURE REVIEW

Id	Type of studies	Classification criteria	Context: Industry vs academic	Domain: medical, socio -eco	Main Quality focus (CQML): defects, maturity, general, functionality	SW domain (object of interest)
SLR-S01	Survey about models and metrics for quality assessment	quality characteristics and metrics	academic	not defined	reliability	software
SLR-S02	Improvement of neural network QM	average absolute error (i.e., predict vs get)	industry & academic	not defined	reliability, fault prediction	operating system
SLR-S03	Survey about models and metrics for reuse metrics and models	purpose / scope	industry & academic	not defined	reuse, maturity, cost, failure (cost benefit analysis, maturity, amount of reuse, failure models analysis, reusability assessment, reuse library metrics)	software

Annexes

SLR-S04	Investigation on software evolution & reuse impact on quality	quality model result: w/ and w/o reuse	industry & academic	telecommunication	reuse, maintenance	telecommunication
SLR-S05	survey about quality and quality model	quality characteristics	academic	not defined	quality in general	software
SLR-S06	survey about challenges	main description	industry	information technology (IT)	quality in general	information system
SLR-S07	creation of new model	main description	industry & academic	accounting and commercial management	maintenance, cost	accounting and commercial management
SLR-S08	creation of new model	product and process metrics	industry & academic	telecommunication	reliability, fault prediction	telecommunication
SLR-S09	creation of new model	none	industry	not defined	cost, defect	software
SLR-S10	creation of new model	Statistical approach (i.e. construction method or formula)	industry & academic	telecommunication	reliability, fault prediction	telecommunication
SLR-S11	creation of new model	quality characteristics and attributes	academic	university	usability, reliability, functionality, efficiency	website
SLR-S12	creation of new model	Statistical approach (i.e. construction method or formula)	academic	telecommunication	fault prediction	telecommunication
SLR-S13	Exploratory study	purpose / scope	academic	not defined	user perception	website
SLR-S14	Survey on quality models role in business process reengineering	Statistical approach (i.e. construction method or formula)	industry & academic	telecommunication	reliability, cost, accuracy, return on investment	telecommunication
SLR-S15	Quality model construction and development methodology	purpose / scope	academic	not defined	safety	information system, website
SLR-S16	Survey on quality from different point of view	link to maturity and reliability characteristic	academic	not defined	reliability, maturity, maintainability	software
SLR-S17	Empirical study of quality models in OO systems	detailed description, formalism, statistical / implicit method, model evaluation, dependent & independent variables, benefits vs limitations	academic	object-oriented systems	reliability, maintainability, fault prediction	object-oriented software
SLR-S18	creation of new model	quality characteristics and sub-characteristics	industry	commercial business	functionality, usability, efficiency	business web portal development environment

SLR-S19	Comparative evaluation	quality model result	academic	telecommunication	reliability	telecommunication, wireless
SLR-S20	Improvement of genetic programming models for QM	Statistical approach (i.e. construction method or formula)	academic	telecommunication	reliability, fault prediction	telecommunication
SLR-S21	creation of new model	purpose / scope	academic	telecommunication	certification, process	Commercial-Off-The-Shelf (COTS), telecommunication
SLR-S22	creation of new model	structure, quality characteristics and sub-characteristics, and metrics	industry & academic	not defined	quality in general	software
SLR-S23	creation of new model	characteristics and limitations	academic	not defined	global quality (maintainability, usability, cost/benefit, security, reliability, timeliness, correctness), local quality	software
SLR-S24	creation of new model	type of model (hierarchical vs relational)	academic	not defined	safety, risk, quality hazard	e-commerce
SLR-S25	comparison of quality models	benefits vs limitations, main characteristics	academic	not defined	general quality, process	software
SLR-S26	Study of standard quality models	description, metrics	academic	not defined	quality standard, reliability	software
SLR-S27	creation of new model	quality model result	academic	not defined	complexity, reliability, cost	information system
SLR-S28	Improvement of quality model metrics	metrics	academic	not defined	usability, cost	web system
SLR-S29	creation of new model	Statistical approach (i.e. construction method or formula)	industry & academic	military	reliability, fault prediction	telecommunication
SLR-S30	creation of new model	characteristics and limitations	academic	government, aeronautic, space	functionality, reliability, usability, efficiency, maintainability, manageability	COTS
SLR-S31	creation of new model	characteristics	academic	not defined	reliability, surface(literal) & conceptual understandability	requirements
SLR-S32	Study between some quality model metrics	metric results	academic	not defined	stability	software developed iteratively / agile
SLR-S33	Predictive model evaluation	Statistical approach (i.e. construction method or formula) and evaluation result	academic	aeronautic, space	fault prediction, performance	software
SLR-S34	creation of new model	quality characteristics	academic	not defined	maintainability, reliability, supportability	software architecture, self-healing system

Annexes

SLR-S35	creation of new model	metrics result	academic	air traffic management	maintainability	software
SLR-S36	creation of new model	description	academic	not defined	flexibility, complexity	software
SLR-S37	creation of new model	quality characteristics	academic	not defined	quality in general	software
SLR-S38	creation of new model	quality characteristics	academic	international web/online course	sharing quality	information sharing system, website
SLR-S39	creation of new model	quality characteristics	academic	not defined	functionality, usability, maintainability	Free/Libre open source software
SLR-S40	Tailoring of quality model	scope	industry & academic	application performance management	evolution, maintainability, reliability	software
SLR-S41	comparison of quality models	evolvability sub-characteristic	industry & academic	industrial systems	evolvability	software
SLR-S42	survey on quality models in practice	interviewee results	industry & academic	software development	quality in general	software
SLR-S43	creation of new model	characteristics vs object-oriented patterns	academic	not defined	design (expandability, simplicity, reusability), implementation (learnability, understandability, modularity), runtime (generality, modularity at runtime, scalability, robustness)	object-oriented software
SLR-S44	comparison of quality models	Purpose, quality focus, viewpoint, quality factors, validation factors, relationships (quantitative & qualitative)	academic	not defined	defect, maturity, effectiveness, reliability, quality in general	software
SLR-S45	creation of new model	description, type of models, practice and impact on quality goals, tool, demo/empirical approach	industry & academic	not defined	correctness, completeness, consistency, comprehensibility, confinement, changeability	model-based software development
SLR-S46	creation of new model	characteristic, limitations	academic	product lines, IT infrastructure	evolvability	software
SLR-S47	creation of new model	perception, interpretation, survey to map quality characteristics / sub-characteristics with conceptual quality characteristics/sub-characteristics (semantic, pragmatic, syntactic quality)	academic	naval aeronautical and astronautical	reliability, conceptual model quality	software

SLR-S48	creation of new model	limitation against directive to compute aggregate indices (from metrics and quality characteristics)	industry	IT systems	quality in general	software
SLR-S49	comparison of quality models	statistical comparison of quality model application against same set of software systems	academic	not defined	maintainability, correctness	open source software
SLR-S50	comparison of quality models	benefits vs limitations from experimental use, main characteristics	academic	not defined	functionality, usability, general quality	Free/Libre open source software, web-client
SLR-S51	Method to customize quality model	purpose / scope, description	academic	Software Factory Data Warehouse	general quality, process	software
SLR-S52	Method to customize quality model	Scope, purpose, viewpoint, focus, context	academic	not defined	perceived consistency, perceived appropriateness, perceived efficiency, completeness, correctness, efficiency	software
SLR-S53	comparison of quality models	characteristics and overall weighted sum based on those characteristics (weight set by experts)	academic	not defined	quality in general	software
SLR-S54	creation of new model	Check point and threshold purpose	industry	not defined	quality in general	Ada software
SLR-S55	creation of new model	Operationalization limitations between quality attributes and measurements	academic	not defined	quality in general	java software, open source
SLR-S56	creation of new model	Description	academic	not defined	quality in general	web-based applications
SLR-S57	Systematic review on quality attributes and metrics	Against ISO 25010 quality characteristics and ISO 9126 metrics	academic	not defined	quality in general	software product lines
SLR-S58	creation of new model	purpose, scope, characteristics	academic	not defined	internal and external quality, quality in use	web [2.0] sites
SLR-S59	creation of new model	description, scope, purpose, limitations against embedded systems	industry & academic	embedded systems	quality in general	embedded systems software
SLR-S60	creation of new model	Mapping to International Software Benchmarking Standards Group (ISBSG) data collection questionnaires	academic	software engineering	functionality, reliability, maintainability, satisfaction, productivity	software
SLR-S61	Study of benefit to use quality model	Scope, quality characteristics	industry & academic	traffic control system	structuredness, traceability, productivity, maintainability	software
SLR-S62	Study of architecture patterns impact on quality attributes	scope	academic	not defined	performance, dependability, costs, security, usability, quality of use	Service-Oriented Architecture (SOA), service-based application

Annexes

SLR-S63	creation of new model	description	academic	Communication systems	functionality, reliability, efficiency, usability, portability, hardware standard	embedded systems software
SLR-S64	survey on development of maturity models	description	academic	not defined	process	software
SLR-S65	survey on quality characteristic importance for quality model	coverage completeness of quality characteristics, sub-characteristics	academic	E-type systems (evolutionary)	quality in general	E-type software (evolutionary)
SLR-S66	comparison of quality models	description, quality characteristics and structure	academic	not defined	quality in general	software
SLR-S67	comparison of quality models	category (basic, tailored), challenges and issues	academic	not defined	quality in general	component based software development
SLR-S68	Survey on quality models and quality attributes	scope, quality characteristics, level of evidences	academic	embedded systems	quality in general	embedded systems software
SLR-S69	Survey on service quality description (i.e., includes quality model as one mean of description)	description, scope, completeness of metrics and details, formalism, association with assessment guidelines	academic	web-based services	quality of services	software service, software-support (i.e., infrastructural) service
SLR-S70	comparison of quality models	description, quality characteristics and structure	academic	not defined	product quality, process quality	software
SLR-S71	survey on sustainability metrics	metrics type, quality characteristics	academic	not defined	sustainability	green software
SLR-S72	creation of new model	description, limitations	industry & academic	university	community activity, maintainability, reliability, release activity	open source, web application, software repository
SLR-S73	creation of new model	impact on metrics	academic	software crowdsourcing platform	efficiency, cost, platform quality factor, project quality project	crowdsourcing-based software
SLR-S74	Survey on quality model metrics normalization impact	relationship with metrics to build quality model	industry & academic	telecom infrastructure, games, hydraulic control	quality in general	software
SLR-S75	Survey on quality model for Legacy Information System assessment	description, scope/purpose, characteristics	academic	Legacy Information Systems (LIS)	quality in general	information systems software
SLR-S76	creation of new model	difference, limitations against a specific quality model	industry & academic	IT systems	maintainability	C# software, IT systems software
SLR-S77	comparison of quality models	benefits vs limitations, main characteristics, quality model origin	academic	not defined	quality in general	open source software

SLR-S78	creation of a reuse model	purpose, scope, description	academic	not defined	quality in general, reuse	software
SLR-S79	classification of program analysis rules into quality model	scope, origin or use of previous quality model	industry	Industry, Healthcare, Energy, Infrastructure and Cities sectors	internal software quality	software code
SLR-S80	Statistical analysis of metrics interactions in quality model	purpose, scope, relationship between characteristics and metrics to build quality model	academic	not defined	quality in general	software
SLR-S81	comparison of quality models	complete description, presence of quality characteristics	academic	not defined	product quality, process quality	software
SLR-S82	Systematic mapping study on software product quality evaluation	scope, approach, origin of previous quality model	academic	not defined	product quality, process quality	software
SLR-S83	creation of new model	structure, approach, limitations, characteristics	academic	Information Systems, Web-based information systems	quality in general	software architecture
SLR-S84	creation of new model	purpose, scope, description, characteristics	academic	b2c, mobile commerce application	quality in general	mobile software applications
SLR-S85	creation of new model	purpose, scope, approach, limitations, characteristics	academic	Scientific computing infrastructures	reliability	distributed software
SLR-S86	survey on quality models for websites against ISO 9126 quality characteristics	quality characteristics mapping against ISO 9126 quality characteristics	academic	not defined	quality in general	websites
SLR-S87	survey on software selection based on quality model	quality characteristics	academic	government organization, IT systems	system quality, information quality and service quality.	open source software
SLR-S88	survey on complexity and completeness quality model evolution against ISO 25010 quality characteristics	quality characteristics, specific metrics relying on characteristics presence aggregation	academic	not defined	quality in general	software
SLR-S89	survey on software product quality evolution and perspective	description, purpose, perspective / viewpoint, measurement	academic	not defined	quality in general	software
SLR-S90	Systematic review on quality models	purpose, structure, quality model origin, quality characteristics, completeness of quality model definition	academic	not defined	quality of services	websites, web-portal, services
SLR-S91	creation of new model	description, quality characteristics	academic	not defined	usability	software systems
SLR-S92	comparison of quality models	description, limitations, quality characteristics	academic	not defined	quality in general	software

Annexes

SLR-S93	comparison of quality models	purpose, scope, metrics, characteristics, quality dimensions, quality model origin	academic	government	quality in use, quality of services	websites, web-portal, services
SLR-S94	comparison of quality models and consolidation of quality characteristic definition	description, scope, characteristics, quality model type between basic, tailored and open source	academic	not defined	quality in general	software
SLR-S95	Systematic review on quality characteristics, creation of new quality model	quality characteristics for systems of systems, belonging to ISO 25010	industry & academic	systems of systems	quality in general	software
SLR-S96	comparison of quality models	description, limitations, quality characteristics	academic	not defined	quality in general	software component
SLR-S97	creation of new model	purpose / scope, characteristics	academic	industry, nuclear facilities, flight control system	maintainability	software systems
SLR-S98	creation of new model	most well-known quality model, maintainability sub-characteristics	academic	web server	maintainability	open-source, software
SLR-S99	comparison / evaluation of quality models	quality characteristics, specific metrics relying on characteristics presence aggregation for greenness and reliability	academic	not defined	greenness, reliability	software
SLR-S100	survey on software product quality evolution and perspective	description, purpose, stakeholder, perspective / viewpoint, measurement	academic	not defined	quality in general	software
SLR-S101	Analysis of quality model use	most well-known quality model	academic	small IT	quality in general	IT software
SLR-S102	Study of model libraries evolution	purpose, scope	academic	not defined	syntactic, semantic, pragmatic, emotional	UML model libraries for software
SLR-S103	creation of new model	description, quality characteristics with regards to usability	academic	not defined	usability	software
SLR-S104	creation of new model	Quality model operationalization: support of tool, automation, and reconfiguration	academic	Open source project	analysis and comparison	open source software
SLR-S105	creation of new model	scope, quality in use quality characteristics	academic	not defined	quality in use	Free / Libre and Open Source Software (FLOSS)
SLR-S106	Mapping against maturity process quality model	process area	academic	IT systems, government	process	software

SLR-S107	Study of quality models for impact analysis	Approach (i.e., quantification method), quality characteristics	academic	software product line maturity		code cloning in object-oriented software
SLR-S108	Survey on machine learning quality models	Description, approach / modeling method	academic	not defined	quality prediction, reliability fault-prediction	software
SLR-S109	comparison / evaluation of quality models	quality characteristics, specific metrics relying on characteristics presence aggregation for greenness and reliability characteristics	academic	Green IT	greenness, reliability	IT software, service
SLR-S110	Exploratory study of measure against feature models	description, quality characteristics related to maintainability	academic	software product line maintainability, variability		feature model, model based designed
SLR-S111	Systematic literature review on quality model related to open-source software	scope, purpose, quality characteristics mapping against ISO 25010, approach / modeling method	academic	not defined	quality in general	open source software (OSS)
SLR-S112	Systematic mapping study on metrics and statistical techniques used for internal quality	brief description (title, author, year, citation), purpose, quality characteristics mapping against ISO 25010, approach / modeling method	academic	not defined	internal quality	object-oriented software
SLR-S113	Survey on software quality attributes	description, quality characteristics	academic	not defined	quality in general	web-based application (WBA)
SLR-S114	comparison / evaluation of quality models	description, comparison of quality model results against real use-case	industry & academic	sustainment management systems	maintainability, reliability, security	open source software, commercial software in C#
SLR-S115	comparison / evaluation of software quality	description of two ISO 9126-based framework, results against real use-case	academic	mobile device, mobile network	product quality, quality in use, quality of service	mobile application
SLR-S116	Systematic mapping of software product quality assessment models	most frequent occurrences of quality characteristics, assessment / validation approach quality model method, tool	academic	not defined	quality in general	software
SLR-S117	Systematic mapping of software quality measurements	Origin model (ISO 9126, ISO 25010, McCall, combined), Approach (i.e., quantification / evaluation method), quality characteristics	academic	business of information and technology	quality in general	software
SLR-S118	creation of new model	description, purpose, quality characteristics,	academic	ambient assisted living (AAL) system	quality in general	AAL software, embedded software

Annexes

SLR-S119	survey on quality in model-driven engineering	quality definition categories (e.g., quality model for MDWE, framework, quality in model transformation), author references	academic	not defined	quality in general	model-driven engineering
SLR-S120	Survey on quality models and quality attributes, and creation of new quality model	description, purpose, scope, type, quality characteristic mapping	academic	not defined	quality in general	embedded software
SLR-S121	creation of new model	description, scope, quality characteristics	industry & academic	Information Systems, financial sector	general quality, process	information systems software
SLR-S122	Comparison of quality models, creation of new quality model	description, quality characteristics, assessment result against use case	academic	e-commerce	quality in general	websites, web-portal
SLR-S123	creation of new model	architectural evaluation types, quality characteristics	academic	systems of systems	quality in general, performance, robustness	architecture, design
SLR-S124	creation of new model	description, limitations, quality characteristics	academic	serious game, education, videogames	quality in general	programmed serious game
SLR-S125	creation of new model	Occurrence of ISO 25010 quality characteristic occurrences in IT technologies	academic	IT, service-based systems (internet of things, green IT, virtual reality, augmented reality, artificial intelligence, cloud computing, blockchain, web)		IT software, service
SLR-S126	comparison of maintainability index quality models	description, approach / techniques method, metrics, dataset used in quality model elaboration	academic	not defined	maintainability	object-oriented software
SLR-S127	comparison of quality models	most well-known / cited quality model for software component, citation occurrence total and year-wise	academic	not defined	quality in general	software component, component model-based software
SLR-S128	survey on quality models of e-learning systems	quality indicator frequency, perspective	academic	e-learning systems	quality in general	e-learning software, website
SLR-S129	Survey on influence of requirement engineering & business analysis on quality characteristics	quality characteristics, influence of requirement engineering & business analysis on quality characteristics	academic	Requirement Engineering (RE), business analysis (BA)	quality in general	software
SLR-S130	creation of new model	comparison of results from MATLAB modeling of quality models	academic	embedded systems, systems (aircrafts, automobile, nuclear power plants and various robotic medical application)	reliability, reliability growth model	embedded software, software
SLR-S131	Systematic mapping study on quality model	description, quality goal, characteristics, scope / coverage, meta-model element	industry	not defined	quality in general, meta-model	software

SLR-S132	creation of new model	scope, purpose, brief description	academic	software-intensive systems	sustainability	software
SLR-S133	Systematic review of software quality aspects with regards to agile	description, quality characteristics / sub-characteristics, link with agile software development	academic	not defined	quality in general	agile software development
SLR-S134	Systematic mapping of software product quality assessment models	most frequent occurrences of metrics, quality characteristics, aggregations and evaluation method, tools	academic	not defined	quality in general	software
SLR-S135	Study on quality in software evolution and creation of new model	description, scope, quality characteristics linked to evolution of quality	academic	mobile applications, open-source market	success, evolution, cohesion, (in)stability, portability, understandability	mobile software, open-source software (OSS), object-oriented software
SLR-S136	comparison / evaluation of quality models	Natural language processing on quality characteristics against ISO 25010 quality characteristics	academic	not defined	performance efficiency, reliability, portability, usability, maintainability, compatibility, security	software

TABLE 47 - MAPPING SYSTEMATIC LITERATURE REVIEW STUDIES WITH QUALITY MODELS

Study Id	Number of quality model	Quality model names
SLR-S01	11	- Rubey - Hartwick - Shooman - Jelinski - Moranda - Schick - Wolverton - Littlewood - Verall - Musa - Mohanty - Adamowicz - Funami - Hasteed - Klobert - Mc Call (FCM) - Schneidewind77
SLR-S02	2	- Khoshgoftaar - Pandya - More - Khoshgoftaar - Lanning - Pandya
SLR-S03	7	- Card - Agresti - Gaffney - Durek - Agresti - Evanco - Process Maturity Framework - CMM v1.0 (SEI) - Koltun - Hudson - SPC Reuse Capability
SLR-S04	8	- Henry - Kafura - Card - Agresti - Shepperd - Henry - Wake - Zage - Khoshgoftaar - Lanning - Pandya - Khoshgoftaar - Lanning - Khoshgoftaar - Allen - Kalaichelvan - Goel

Annexes

SLR-S05	10	<ul style="list-style-type: none"> Product - Mc Call (FCM) - Basili (GQM) - Evans - Marciniak - Gilb - ISO 9126 - Khoshgoftaar - Lanning - Dromey Process - Malcolm Baldrige Criteria for Performance Excellence - ISO 9001 - CMM v1.1
SLR-S06	9	<ul style="list-style-type: none"> Product - Boehm78 - Basili (GQM) - Garvin Eight dimension of Quality - FURPS Process - ISO 9001 - CMM v1.1 - SPICE Organization - Malcolm Baldrige Criteria for Performance Excellence - EFQM Excellence
SLR-S07	2	<ul style="list-style-type: none"> - Wake-Henry - Granja-Alvarez - BarrancoGarcia Maintenance Cost
SLR-S08	2	<ul style="list-style-type: none"> - EMERALD Test Targeting - Khoshgoftaar - Allen - Naik - Jones- Hudepohl
SLR-S09	1	<ul style="list-style-type: none"> - SDCH
SLR-S10	15	<ul style="list-style-type: none"> - Henry - Kafura - Selby - Porter - Briand - Basili - Thomas - EMERALD Test Targeting - Evanco - Agresti - Khoshgoftaar - Lanning - Schneidewind95 - Troster - Tian - Khoshgoftaar - Allen - Kalaichelvan - Goel - Basili - Briand - Melo (QCM) - Ebert - Gokhale - Lyu - Khoshgoftaar - Allen - Naik - Jones- Hudepohl - Schneidewind98 - Jones - Hudepohl - Khoshgoftaar - Allen
SLR-S11	5	<ul style="list-style-type: none"> - ISO 9126 - Dujmovic LSP - Dujmovic - Bayucan - WAMMI - Website QEM

SLR-S12	22	<ul style="list-style-type: none"> - Selby - Porter - Briand - Brasili- Hetmanski - EMERALD Test Targeting - Khoshgoftaar - Lanning - Schneidewind95 - Troster - Tian - Khoshgoftaar - Allen - Kalaichelvan - Goel - Basili - Briand - Melo - Ebert - Gokhale - Lyu - Takahashi - Muraoka - Nakamura - Schneidewind97 - Khoshgoftaar - Allen - Naik - Jones- Hudepohl - Khoshgoftaar - Allen98 - Ohlsson - Zhao - Helander - Khoshgoftaar - Allen - Jones- Hudepohl - 99a - Jones - Hudepohl - Khoshgoftaar - Allen - Khoshgoftaar - Allen99 - Khoshgoftaar - Allen - Jones- Hudepohl - 99b - Khoshgoftaar - Allen - Yuan - Jones - Huderpohl - Khoshgoftaar - Allen - Jones- Hudepohl - 2000 - Khoshgoftaar - Shan - Allen
SLR-S13	10	<ul style="list-style-type: none"> - Kano - SERVQUAL - SERVQUAL91 - USISF - SERVQUAL - Levi - Conrad - WebMAC - Gehrken – Turban - Von Dran - Zhang - Small - Zhang et al. Website Quality Model - Expanded Website Quality Model
SLR-S14	4	<ul style="list-style-type: none"> - EMERALD Test Targeting - Khoshgoftaar - Allen- Halstead - Trio - Flass - Khoshgoftaar - Allen98 - Khoshgoftaar - Allen2000
SLR-S15	10	<ul style="list-style-type: none"> - Mc Call (FCM) - Boehm78 - Constructive QUALity MOdel (COQUAMO) - SOLE - ISO 9126 - Perry - Gillies - Dromey - Website QEM - Quality Model for Object-Oriented Design (QMOOD) - HASARD model
SLR-S16	8	<ul style="list-style-type: none"> - Musa - Schneidewind77 - Schneidewind95 - Schneidewind97 - Schneidewind98 - CMM v1.0 - CMM v1.1 - Khoshgoftaar - Allen - Kalaichelvan - Goel

Annexes

SLR-S17	13	<ul style="list-style-type: none">- Brito e Abreu - Melo MOOD- Basili - Briand - Melo (QCM)- Briand - Wüst - Daly - Porter- MARS model- Briand - Wüst - Lounis- El Emam, - Melo - Machado- Benlarbi - Melo- El Emam - Benlarbi - Goel - Rai 01- El Emam - Benlarbi - Goel - Rai 99- El Emam - Benlarbi - Goel - Melo - Lounis - Rai- Glasberg - El Emam - Melo - Madhavji- Harrison - Samaraweera - Dobie - Lewis- Tang - Kao - Chen
SLR-S18	2	<ul style="list-style-type: none">- ISO 9126- Business Portal Development Environment (PBDE) quality model
SLR-S19	2	<ul style="list-style-type: none">- EMERALD Test Targeting- Khoshgoftaar - Allen - Jones- Hudepohl - 99a
SLR-S20	8	<ul style="list-style-type: none">- EMERALD Test Targeting- Ohlsson - Zhao - Helander- Liu - Khoshgoftaar- Khoshgoftaar - Allen - Deng 2002- Khoshgoftaar - Cukic - Seliya 2002- Pizzi - Summers - Pedrycz- MARS model- Khoshgoftaar - Seliya - Liu
SLR-S21	8	<ul style="list-style-type: none">- CMM v1.0- ISO 9126- CMM v1.1- SPICE- INSTAC model- MUSiC- Modeling & Simulation Application (MSA) certification model- Certification model
SLR-S22	8	<ul style="list-style-type: none">- Boehm73- Mc Call (FCM or RADC)- Boehm78- ISO 9126- QUES model (RADC)- Dromey- MITRE Software Quality Assessment Exercise (SQAE)- AFOTEC Maintainability- Enhanced SQAE
SLR-S23	7	<ul style="list-style-type: none">- Boehm78- Gillies- Dromey- Garvin Eight dimension of Quality- Constructive QUALity MOdel (COQUAMO)- ISO 9126- ADEQUATE
SLR-S24	8	<ul style="list-style-type: none">- Mc Call (FCM or RADC)- Boehm78- ISO 9126- Perry- Gillies- Dromey- Quality Model for Object-Oriented Design (QMOOD)- HASARD model

SLR-S25	8	<ul style="list-style-type: none"> - Basili (GQM) - ISO 9001 - ISO 9126 - CMM v1.1 - SPICE - GQIM - goal-driven model - Software Quality Model for Small Organizations (SQIMSO) - Self-diagnosis Herrera - Ramirez model
SLR-S26	7	<ul style="list-style-type: none"> - Jelinski - Moranda - Littlewood - Verall - Goel - Okumoto (NHPP) - Langberg - Singpurwalla - Nayak - ISO 9126 (-ISO 25010)
SLR-S27	4	<ul style="list-style-type: none"> - Intranet Application Quality Model - So - Cha - Kwon - Quah - Thwin - Stack-based Markov (SBM) Model
SLR-S28	2	<ul style="list-style-type: none"> - Ramler - Weippl - Winterer - Schwinger -Altmann - QUINT2
SLR-S29	9	<ul style="list-style-type: none"> - Shen - Yu - Thebaut -Paulsen - Khoshgoftaar - Munson - Henry - Wake - Khoshgoftaar - Pandya - More - Khoshgoftaar - Lanning - Pandya - Khoshgoftaar - Szabo94 - Khoshgoftaar - Allen2000 - Poisson Regression Model Fault
SLR-S30	7	<ul style="list-style-type: none"> - McCall - Boehm78 - FURPS - Dromey - ISO 9126 - Software quality certification triangle - Rawashdeh- Matakah
SLR-S31	4	<ul style="list-style-type: none"> - SATC model - Natural Language Software Requirements - Specification (NLSRS) quality model - Ormandjieva - Hussain - Kosseim
SLR-S32	4	<ul style="list-style-type: none"> - Mc Call (FCM or RADC) - ISO 9126 - Dromey - Quality Model for Object-Oriented Design (QMOOD)
SLR-S33	6	<ul style="list-style-type: none"> - Basili - Briand - Melo (QCM) - Gokhale - Lyu - Khoshgoftaar - Lanning - Khoshgoftaar - Ganesan - Allen - Ross - Munikoti - Goel - Nandi - Khoshgoftaar - Seliya - Guo - Ma - Cukic - Singh
SLR-S34	7	<ul style="list-style-type: none"> - ISO 9126 - Dromey - Bowen - Post - Tsai - Presson - Schmidt - Bowen - Wigle - Tsai - SQUID - Losavio - Chirinos - Perez - Neti - Muller

Annexes

SLR-S35	4	<ul style="list-style-type: none">- McCall- Dromey- ISO 9126- Lincke - Lundberg - Löwe
SLR-S36	5	<ul style="list-style-type: none">- McCall- Boehm78- Perry- ISO 9126- 2D Software Quality Model Zhang - Li - Gao
SLR-S37	10	<ul style="list-style-type: none">-McCall- Boehm78- Basili (GQM)- SQUID- Dromey- Gilb- FURPS- ISO 9126- CUPRIMDSO- FUPRIMDSO
SLR-S38	1	<ul style="list-style-type: none">-Feng - Liu
SLR-S39	5	<ul style="list-style-type: none">-Basili (GQM)- Dromey- ISO 9126- Systemic Quality Model- MOSCA
SLR-S40	12	<ul style="list-style-type: none">- Boehm76- McCall- CMM v1.0 (SEI)- CMMi- CM3 maturity model- SMmm- Basili - Briand - Melo (QCM)- INSTAC model- Evans - Marciniak- Rawashdeh- Matakah- Lee - Lee- ISO 25010 (early version)
SLR-S41	7	<ul style="list-style-type: none">- McCall- Boehm78- Dromey- FURPS- ISO 9126- Systemic Quality Model- Rawashdeh- Matakah
SLR-S42	2	<ul style="list-style-type: none">- ISO 9001- ISO 9126
SLR-S43	2	<ul style="list-style-type: none">- Quality Model for Object-Oriented Design (QMOOD)- PQMOD

SLR-S44	25	<ul style="list-style-type: none"> - CMMi - SPICE (ISO / IEC 15504) - SQUID - ISO 9126 - Basili - Briand - Melo (QCM) - GEneric, multilayered and customisable QUALity MOdel (GEQUAMO) - COConstructive QUALity MOdel (COQUALMO) - Hybrid Defect Content and Effectiveness Early Prediction (HyDEEP) - Orthogonal Defect Classification (ODC) - Defect Flow Model (DFM) - Experience-based NFR quality model - Mills - Eick - Loader - Long - Votta - Wiel Capture-Recapture - Eick - Loader - Wiel - Votta Capture-Recapture - Wiel - Votta Capture-Recapture - Wohlin - Runeson - Brantestam Capture-Recapture - Ebrahimi Capture-Recapture - Briand - Emam - Freimut Capture-Recapture - Runeson - Wohlin Capture-Recapture - Petersson - Wohlin Capture-Recapture99 - Petersson - Wohlin Capture-Recapture00 - Thelin - Runeson Capture-Recapture - Biffel - Grossmann Inspection-Reinspection - Miller - Macdonald - Ferguson Capture-Recapture - Padberg Cpature-Recapture
SLR-S45	1	<ul style="list-style-type: none"> - Mohagheghi - Dehlen -Neple
SLR-S46	6	<ul style="list-style-type: none"> - ISO 9126 - McCall - Basili (GQM) - Prometheus - FMSQE - Brcina - Bode -Riebisch evolvability
SLR-S47	2	<ul style="list-style-type: none"> - ISO 25010 - Conceptual Model Quality
SLR-S48	4	<ul style="list-style-type: none"> - Boehm78 - ISO 9126 - ISO 25010 - Software Quality Assessment Based on Lifecycle Expectation (SQALE)
SLR-S49	7	<ul style="list-style-type: none"> - McCall - ISO 9126 - Basili - Briand - Melo (QCM) - Thwin - Quah - van Koten - Gray - Zhou - Leung MARS model - Welker - Oman - Atkinson Maintainability index
SLR-S50	7	<ul style="list-style-type: none"> - Capability Maturity Model Integration (CMMi) v1.1 - OpenSource Maturity Model (OSMM) Cap Gemini - OpenSource Maturity Model (OSMM) Navica - Qualification and Selection of Open Source software (QSOS) - Open Business Readiness Rating (OpenBRR) - Open Business Quality Rating (Open BQR) - QualitPso Open Source Maturity Model (OSMM)

Annexes

SLR-S51	10	<ul style="list-style-type: none"> - CMM v1.0 (SEI) - Capability Maturity Model Integration (CMMi) v1.1 - ISO 9001 - Bootstrap - SPICE (ISO / IEC 15504) - Basili (GQM) - GQIM - goal-driven model - ISO 9126 - HASARD model - Golubic quality build-in based quality model
SLR-S52	31	<ul style="list-style-type: none"> - McCall - CMMi - SPICE (ISO / IEC 15504) - Software QQuality In Development (SQUID) - ISO 9126 - Basili (GQM) - Basili - Briand - Melo (QCM) - ADEQUATE - SOLE - GGeneric, multilayered and customisable QUALity MOdel (GEQUAMO) - Hybrid Defect Content and Effectiveness Early Prediction (HyDEEP) - Orthogonal Defect Classification (ODC) - Defect Flow Model (DFM) - Experience-based NFR quality model - Eick - Loader - Long - Votta - Wiel Capture-Recapture - Eick - Loader - Wiel - Votta Capture-Recapture - Wiel - Votta Capture-Recapture - Wohlin - Runeson - Brantestam Capture-Recapture - Ebrahimi Capture-Recapture - Briand - Emam - Freimut Capture-Recapture - Runeson - Wohlin Capture-Recapture - Petersson - Wohlin Capture-Recapture99 - Petersson - Wohlin Capture-Recapture00 - Thelin - Runeson Capture-Recapture - Biffi - Grossmann Inspection-Reinspection - Miller - Macdonald - Ferguson Capture-Recapture - Padberg Cpature-Recapture - Behkamal - Kahani - Akbari - Original software components quality model (OSCQM) - Benlarbi - El Emam - Goel - Rai- QUAMOCO
SLR-S53	11	<ul style="list-style-type: none"> - McCall - Boehm78 - FURPS - Dromey - ISO 9126 - Rawashdeh- Matakah - Behkamal - Kahani - Akbari - Software Quality STAR model - Stefani - Xenos - Stavrinoudis - Aspect-Oriented Software Quality Model (AOSQUAMO) - Bertoa - Vallecillo - Educational Software Quality Hierarchy Triandgle (ESHTri) model
SLR-S54	4	<ul style="list-style-type: none"> - Boehm78 - McCall - Software Quality Assessment Based on Lifecycle Expectation (SQALE) - Software Quality Assessment Based on Lifecycle Expectation (SQALE) ADA
SLR-S55	8	<ul style="list-style-type: none"> - McCall - Boehm78 - Dromey - Software QQuality In Development (SQUID) - ISO 25010

		<ul style="list-style-type: none"> - Deissenboeck - Wagner - Pizka - Teuchert - Girard maintainability 2D model - Winter - Wagner - Deissenboeck usability 2D model - Lochmann - Heinemann
SLR-S56	10	<ul style="list-style-type: none"> - McCall - ISO 9126 - Dromey - WBA quality model (WBAQM) - Web Assessment Index (WAI) - Signore - Web-site Quality Evaluation Methodology (QEM) model - 2QCV3Q - Fuzzy Model for Software Quality Evaluation (FMSQE) - Web-site Quality Evaluation Method (QEM) framework
SLR-S57	4	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - QUINT2 - Zhang - Jarzabek - Yang
SLR-S58	16	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - Web-site Quality Evaluation Methodology (QEM) model - 2QCV3Q - QUINT2 - Signore - Stefani - Xenos - Stavrinoudis - Fuzzy Model for Software Quality Evaluation (FMSQE) - Web-site Quality Evaluation Method (QEM) framework - Polillo - Extended Web Assessment Method (EWAM) - Web Application Quality Evaluation model (WAQE) - Malak - Sahraoui - Badri - Badri Web quality model - 2Q2U - Quality in Use Model for Web Portals (QiUWeP) - Polillo

Annexes

SLR-S59	33	<ul style="list-style-type: none"> - McCall - Boehm78 - Dromey - ISO 9126 - ISO 25010 - CMMi - SPICE (ISO / IEC 15504) - SQUID - Basili - Briand - Melo (QCM) - GEneric, multilayered and customisable QUAlity MOdel (GEQUAMO) - Hybrid Defect Content and Effectiveness Early Prediction (HyDEEP) - Orthogonal Defect Classification (ODC) - Defect Flow Model (DFM) - Experience-based NFR quality model - Eick - Loader - Long - Votta - Wiel Capture-Recapture - Eick - Loader - Wiel - Votta Capture-Recapture - Wiel - Votta Capture-Recapture - Wohlin - Runeson - Brantestam Capture-Recapture - Ebrahimi Capture-Recapture - Briand - Emam - Freimut Capture-Recapture - Runeson - Wohlin Capture-Recapture - Petersson - Wohlin Capture-Recapture99 - Petersson - Wohlin Capture-Recapture00 - Thelin - Runeson Capture-Recapture - Biffi - Grossmann Inspection-Reinspection - Miller - Macdonald - Ferguson Capture-Recapture - Padberg Cpature-Recapture - QUAMOCO - Software Component Quality Model (CQM) v1.0 - Embedded software component quality model (EQM) - Coleman - Ash - Lowther - Oman Maintainability Index - Wagner Activity-Based Quality Model (ABQM) - Embedded Systems software Quality Model (ESQM)
SLR-S60	3	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - ISBSG quality model
SLR-S61	6	<ul style="list-style-type: none"> - SQUID - ISO 9126 - ISO 25010 - Wagner Activity-Based Quality Model (ABQM) - Service-Oriented Architecture (SOA) Design quality model - Lochmann - Goeb Unifying Model - QUAMOCO
SLR-S62	2	<ul style="list-style-type: none"> - ISO 9126 - SCube Quality Reference Model
SLR-S63	7	<ul style="list-style-type: none"> - Boehm76 - McCall - ISO 9126 - Perry - Gillies - Dromey - Wan-Jiang - Tian-Bo
SLR-S64	4	<ul style="list-style-type: none"> - CMM v1.0 (SEI) - Capability Maturity Model Integration (CMMi) v1.1 - SPICE (ISO / IEC 15504) - Capability Maturity Model Integration (CMMi) v1.1
SLR-S65	3	<ul style="list-style-type: none"> - ISO 9126 - Behkamal - Kahani - Akbari - Portlet Quality Model (PtQM)

SLR-S66	13	<ul style="list-style-type: none"> - McCall - Boehm76, Boehm78 - FURPS - Dromey - ISO 9126 - PQMOD - Software Quality STAR model - Aspect-Oriented Software Quality Model (AOSQUAMO) - SATC model - Quality Model for Object-Oriented Design (QMOOD) - REquirements, Aspects and Software Quality (REASQ) model - Chang - Wu - Lin - IEEE model for software maintenance
SLR-S67	26	<ul style="list-style-type: none"> - McCall - Boehm78 - FURPS - Dromey - ISO 9126 - Bertoa - Vallecillo - Prometheus - Systemic Quality Model - GEneric, multilayered and customisable QUALity MOdel (GEQUAMO) - Software Quality STAR model - Rawashdeh- Matakah - Original software components quality model (OSQCM) - Mbusi - Van Waveren - Sharma - Kumar - Grover - Behkamal - Kahani - Akbari - Aspect-Oriented Software Quality Model(AOSQUAMO) - Embedded software component quality model (EQM) - Srivastava - Kumar - Jamwal - Jamwal - Bawane - Srikrishna - Software Component Quality Model (CQM) v1.1 - Samsung s/w Component Quality evaluation Model (SCQM) - Q'Facto 10 - Q'Facto 12 - Upadhyay - Despande - Agrawal Software Component Quality Model (SCQM) - AL-Badareen - Selamat - Din - Jabar - Turaev
SLR-S68	10	<ul style="list-style-type: none"> - McCall - Boehm76 - ISO 25010 - Purhonen - Samsung s/w Component Quality evaluation Model (SCQM) - Embedded software component quality model (EQM) - Jeong - Kim v1.0 - Jeong - Kim v1.1 - Jeong - Ahrens - Frey - Pfeiffer - Bertram

Annexes

SLR-S69	13	<ul style="list-style-type: none">- ISO 9126- BREIN QoS ontology- Truong - Samborski - Fahringer- Ran- Cappiello - Kritikos - Metzger - Parkin - Pernici - Plebani - Treiber- Mabrouk - Georgantas - Issarny- WSQM- Sabata - Chatterjee - Davis - Sydir - Lawrence- WSAF-QoS- QoSOnt- Web Services Modeling Ontology (WSMO) QoS- onQoS-QL- OWL-Q
SLR-S70	9	<ul style="list-style-type: none">- McCall- Boehm78- Dromey- FURPS- ISO 9126- ISO 25010- CMM v1.0 (SEI)- SPICE (ISO / IEC 15504)- Capability Maturity Model Integration (CMMi) v1.1
SLR-S71	10	<ul style="list-style-type: none">- ISO 9126- ISO 25010- 25010+S- Capability Maturity Model Integration (CMMi) v1.1- SPICE (ISO / IEC 15504)- QualOSS- Qualification and Selection of Open Source software (QSOS)- OpenSource Maturity Model (OSMM) Cap Gemini- OpenSource Maturity Model (OSMM) Navica- Open Business Readiness Rating (OpenBRR)
SLR-S72	9	<ul style="list-style-type: none">- ISO 9126- OpenSource Maturity Model (OSMM) Cap Gemini- Open Business Readiness Rating (OpenBRR)- Qualification and Selection of Open Source software (QSOS)- Stefani - Xenos- Malak - Sahraoui - Badri - Badri Web quality model- Benlarbi - El Emam - Goel - Rai- SQO-OSS- Adewumi - Omoregbe - Misra - Fernandez
SLR-S73	2	<ul style="list-style-type: none">- Briand - Wüst - Daly - Porter- Li - Xiao - Wang - Wang
SLR-S74	5	<ul style="list-style-type: none">- McCall- Basili (GQM)- ISO 9126- Fan - Luo - Wu - Fu- ColumbusQM
SLR-S75	7	<ul style="list-style-type: none">-ISO 9126- ISO 25010- Behkamal - Kahani - Akbari- 2Q2U- 2Q2U v2- Quality in Use Model for Web Portals (QiUWeP)- WebQual

SLR-S76	10	<ul style="list-style-type: none"> - ISO 9126 - ColumbusQM - Quality Model for Object-Oriented Design (QMOOD) - Software Quality Assessment Based on Lifecycle Expectation (SQALE) - Technical Topic Classification (TTC) quality model - Muthanna - Kontogiannis - Ponnambalam - Stacey - SIG maintainability model - Carvallo - Franch - LaQuSo software product certification model (LSPCM) - Hegedús
SLR-S77	12	<ul style="list-style-type: none"> - McCall - Boehm76 - Boehm78 - ISO 9126 - CMM v1.0 (SEI) - OpenSource Maturity Model (OSMM) Cap Gemini - OpenSource Maturity Model (OSMM) Navica - Qualification and Selection of Open Source software (QSOS) - QualOSS - QualitPso Open Source Maturity Model (OSMM) - Open Business Readiness Rating (OpenBRR) - SQO-OSS
SLR-S78	8	<ul style="list-style-type: none"> - McCall - Boehm78 - ISO 9126 - ISO 9001 - SPICE (ISO / IEC 15504) - CMM v1.0 (SEI) - Korean Software Process Quality Certification Model - SERVQUAL
SLR-S79	10	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - FURPS - SATC model - Evaluation Method for Internal Software Quality (EMISQ) - Dynamic Analysis for Internal Software Quality (DAISQ) model - Venkatasubramanyam, Radhika D and Nayak, Snigdha - Quality Model for Object-Oriented Design (QMOOD) - Qualixo model - Software Quality Assessment Based on Lifecycle Expectation (SQALE)
SLR-S80	4	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - Wagner Activity-Based Quality Model (ABQM) - Khoshgoftaar - Seliya - Sundaresh
SLR-S81	6	<ul style="list-style-type: none"> - McCall - Boehm78 - Dromey - FURPS - ISO 9126 - CMM v1.1

SLR-S82	20	<ul style="list-style-type: none"> - McCall - Boehm78 - Dromey - Basili (GQM) - ISO 9126 - ISO 25010 - QUAMOCO - CMM v1.1 - SPICE (ISO / IEC 15504) - ISO 9001 - Quality Model for Object-Oriented Design (QMOOD) - Wang - Ceberio - Virani - Garcia - Cummins - Baliyan - Kumar - Wagner Activity-Based Quality Model (ABQM) - Aspect-Oriented Software Quality Model (AOSQUAMO) - Chang - Wu - Lin - Lee - Lee - Briand - Wüst - Daly - Porter - Takahashi - Muraoka - Nakamura
SLR-S83	20	<ul style="list-style-type: none"> - McCall - Boehm78 - ISO 9126 - ISO 25010 - Dromey - Quality Model for Object-Oriented Design (QMOOD) - Service-Oriented Architecture (SOA) quality model - SCube Quality Reference Model - Mobile Software Quality Model - Gillies - Perry - Dromey - FURPS - Deissenboeck - Wagner - Pizka - Teuchert - Girard maintainability 2D model - Wagner Activity-Based Quality Model (ABQM) - Lochmann - Goeb Unifying Model - QUAMOCO - Winter - Wagner - Deissenboeck usability 2D model - Extended Activity-Based Quality Model (ABQM) - HASARD model
SLR-S84	10	<ul style="list-style-type: none"> - McCall - Boehm78 - ISO 9126 - ISO 25010 - Basili (GQM) - Mobile Software Quality Model - Mobile Application Quality Model - Hussain - Kutar - Quality in Use Integrated Measurement (QUIM) model - Yildiz - Bilgen - Tokdemir - Cagiltay - Erturan

SLR-S85	22	<ul style="list-style-type: none"> - Ronchieri - Canaparo - Salomoni - Khoshgoftaar - Allen - Kalaichelvan - Goel - Khoshgoftaar - Seliya - Liu - Ohlsson - Zhao - Helander - El Emam, - Melo - Machado - Yu - Mishra - Schneidewind97 - Briand - Brasili- Hetmanski - Khoshgoftaar - Seliya - O3b - Kim - Zimmermann - Whitehead Jr - Zeller - Lincke - Lundberg - Löwe - Zhou - Leung MARS model - Fenton - Ohlsson - Kanmani - Uthariaraj - Sankaranarayanan - Thambidurai - Khoshgoftaar - Allen - Hudepohl - Aud - Briand - Wüst - Daly - Porter - Yuan - Khoshgoftaar - Allen - Ganesan - Gondra - Singh - Kannoja - Malhotra - Jain - Brito e Abreu - Melo MOOD - Capability Maturity Model Integration (CMMi) v1.1
SLR-S86	18	<ul style="list-style-type: none"> - McCall - Boehm78 - ISO 9126 - ISO 25010 - Signore - QUINT2 - 2QCV3Q - Web-site Quality Evaluation Methodology (QEM) model - Web Q-Model - Web Application Quality Evaluation model (WAQE) - Dominic - Jati - Fuzz-Web model - Malak - Sahraoui - Badri - Badri Web quality model - Stefani - Xenos - Stavrinoudis - Singh - Malhotra - Gupta - Web-Application (WA) maintainability model - Oman - Hagemeister maintainability model - SATC model
SLR-S87	6	<ul style="list-style-type: none"> - SERVQUAL - Open Business Quality Rating (Open BQR) - Qualification and Selection of Open Source software (QSOS) - QualOSS - SQO-OSS - QualiPSo
SLR-S88	10	<ul style="list-style-type: none"> - McCall - Boehm78 - Ghezzi - Jazayeri - Mandrioli - FURPS - IEEE model for software maintenance - Dromey - ISO 9126 - Quality Model for Object-Oriented Design (QMOOD) - ISO 25010 - SATC model
SLR-S89	6	<ul style="list-style-type: none"> - Mc Call (FCM or RADC) - Boehm78 - ISO 9126 - ISO 25010

		<ul style="list-style-type: none">- FURPS- FURPS+
SLR-S90	48	<ul style="list-style-type: none">- ISO 9126- ISO 25010- SCube Quality Reference Model- IBM- Ran- W3C QoS- WSAF-QoS- Avizienis - Laprie - Randell - Landwehr- WebQ QoS- SemWebQ- Looker - Munro - Xu- QoSOnt- Web Services Modeling Ontology (WSMO) QoS- Jiang - Aagesen- Yeom - Yun - Min- Tsesmetzis - Roussaki - Papaioannou - Anagnostou- Garcia - Beatriz- Truong - Samborski - Fahringer- Ren - Chen - Chen - Song - Xiao- Web Service Modeling Ontology (WSMO)- Web Service Modeling Ontology (WSMO) v1.4- Yu - Radhakrishna - Pingali - Kolluri- Kang- Abramowicz - Hofman - Suryan - Zyskowski- Artaiam - Senivongse- onQoS-QL- BREIN QoS ontology- Chang - Lee- Al-Masri - Mahmoud- Tong - Cao - Zhang - Mou- WS-QoSOnto- Comuzzi - Pernici- Balfagih - Hassan- Li - Zhou- Reddy - Maralla - Thirumaran- Mohanty - Ravi - Patra- Yin - Yang - Fu - Chen- Pan - Baik- Zhang - Song- Lee - Yeom- Bocciarelli - D'Ambrogio- Qiu - Yu- Debnath - Martellotto - Daniele - Riesco - Montejano- Moser - Rosenberg - Dustdar- GESSI- WSQM- Phalnikar - Khutade- Nandanam - Rajmohan

SLR-S91	12	<ul style="list-style-type: none"> - Quality in Use Integrated Measurement (QUIM) model - Abran- Khelifi - Suryn - Seffah - Alonso-Ríos - Vázquez-García - Mosqueira-Rey - Moret-Bonillo - Boehm78 - Dix - Finlay - Abowd - Beale - McCall - FURPS - ISO 9126 - Nielsen - Preece - Benyon - Davies - Keller - Rogers - Shackel - Gupta - Ahlawat - Sagar
SLR-S92	16	<ul style="list-style-type: none"> - McCall - Boehm78 - ISO 9126 - FURPS - FURPS+ - CMM v1.0 (SEI) - Ghezzi - IEEE model for software maintenance - Dromey - SATC model - Quality Model for Object-Oriented Design (QMOOD) - Aspect-Oriented Software Quality Model (AOSQUAMO) - Sharma - Kumar - Grover - PQMOD - ISO 25010
SLR-S93	16	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - e-Government Services (E-GSQ) model - Communication between Agricultural Businesses and Government (CABAG) - SiteQual - SERVQUAL - E-S-SQUAL - E-RecS-SQUAL - Dagger - Sweeney - Johnson - eGovQual - Bhattacharya - Gulla - Gupta - Chutimaskul - Funilkul - Chongsuphajaisiddhi - Rababah - Hamtini - Harfoushi - Al-Shboul - Obiedat - Nawafleh - Website Evaluation Questionnaire (WEQ) - e-Government website evaluation tool (eGwet) - Hien
SLR-S94	13	<ul style="list-style-type: none"> - McCall - Boehm78 - FURPS - ISO 9126 - ISO 25010 - Bertoa - Vallecillo - GGeneric, multilayered and customisable QUALity MOdel (GEQUAMO) - Software Component Quality Model (CQM) v1.1 - Rawashdeh- Matakah - OpenSource Maturity Model (OSMM) Cap Gemini - Open Business Readiness Rating (OpenBRR) - SQO-OSS - QualOSS

Annexes

SLR-S95	16	<ul style="list-style-type: none"> - McCall - Boehm78 - FURPS - Dromey - ISO 9126 - ISO 25010 - Rawashdeh- Matakah - Behkamal - Kahani - Akbari - Software Quality STAR model - Stefani - Xenos - Stavrinoudis - Aspect-Oriented Software Quality Model (AOSQUAMO) - Bertoa - Vallecillo - Educational Software Quality Hierarchy Triandgle (ESHTri) model" - Software QUality In Development (SQUID) - Bianchi - Santos - Felizardo
SLR-S96	5	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - Software Component Quality Model (CQM) v1.0 - Software Component Quality Model (CQM) v1.1 - Bertoa - Vallecillo
SLR-S97	8	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - ColumbusQM - Hegedús - Software Quality Assessment Based on Lifecycle Expectation (SQALE) - Software QUALity Enhancement (SQUALE) - SIG maintainability model - QUAMOCO
SLR-S98	10	<ul style="list-style-type: none"> - McCall - Boehm78 - Dromey - ISO 9126 - ISO 25010 - Quality Model for Object-Oriented Design (QMOOD) - SIG maintainability model - SQO-OSS - Systemic Quality Model - Software Quality Model for Maintainability Analysis (SQMMA)
SLR-S99	9	<ul style="list-style-type: none"> - McCall - Boehm78 - Ghezzi - Jazayeri - Mandrioli - FURPS - IEEE model for software maintenance - Dromey - ISO 9126 - Quality Model for Object-Oriented Design (QMOOD) - ISO 25010
SLR-S100	7	<ul style="list-style-type: none"> - Mc Call (FCM or RADC) - Boehm78 - ISO 9126 - ISO 25010 - FURPS - FURPS+ - Basili (GQM)
SLR-S101	5	<ul style="list-style-type: none"> - McCall - Boehm78 - FURPS - ISO 9126 - Dromey

SLR-S102	2	- Lindland - Sindre - Solvberg - Roth - Ganser - Lichter - Rumpe Quality model for Models
SLR-S103	12	- McCall - Boehm78 - Shackel - FURPS - Nielsen - Software Usability Measurement Inventory (SUMI) quality model - ISO 9126 - Quality in Use Integrated Measurement (QUIM) model - Software Engineering Methodology (SEM) quality model - Preece - Benyon - Davies - Keller - Rogers - Alonso-Ríos - Vázquez-García - Mosqueira-Rey - Moret-Bonillo - Kabir - Rehman - Majumdar
SLR-S104	11	- McCall - Boehm78 - ISO 9126 - OpenSource Maturity Model (OSMM) Cap Gemini - Open Business Readiness Rating (OpenBRR) - Qualification and Selection of Open Source software (QSOS) - QualiPSo - SQO-OSS - QualOSS - Software Quality Assessment Based on Lifecycle Expectation (SQALE) - OSSMETER Quality model
SLR-S105	4	- ISO 9126 - ISO 25010 - MUSiC - QUIndicator quality model
SLR-S106	1	- Capability Maturity Model Integration (CMMi) v1.1
SLR-S107	4	- SIG maintainability model - Coleman - Ash - Lowther - Oman Maintainability Index - Oman - Hagemester maintainability model - Khoshgoftaar - Munson
SLR-S108	21	- Adline - Ramachandran - Ahmed - Al-Jamimi - Amasaki - Takagi - Mizuno - Kikuno - Azar - Vybihal - Guo - Lyu - Karunanithi - Whitley - Malaiya - Khoshgoftaar - Seliya - Khoshgoftaar - Allen- Halstead - Trio - Flass - Khoshgoftaar - Ganesan - Allen - Ross - Munikoti - Goel - Nandi - Khoshgoftaar - Allen - Jones- Hudepohl - 99a - Khoshgoftaar - Pandya - More - Khoshgoftaar - Seliya - Sundaresh - Khoshgoftaar - Shan - Allen - Mittal - Bhatia - Goswami - Pizzi - Summers - Pedrycz - Puri - Singh - Radliński - Rashid - Patnaik - Bhattacharjee - Wagner Activity-Based Quality Model (ABQM) - Xing - Guo - Lyu - Yuan - Khoshgoftaar - Allen - Ganesan

Annexes

SLR-S109	9	<ul style="list-style-type: none">- McCall- Boehm78- Ghezzi - Jazayeri - Mandrioli- FURPS- IEEE model for software maintenance- Dromey- ISO 9126- Quality Model for Object-Oriented Design (QMOOD)- ISO 25010
SLR-S110	8	<ul style="list-style-type: none">- ISO 25010- Bagheri - Gasevic- Duan - Kattepuray - Getahun - Elfakiz - Du- Etxeberria - Sagardui- Requirements-driven Quality Estimator (ReQuEst)- Zhang - Jarzabek - Yang- Zhang - Ye - Lin- CatalOg of measures for Feature modEl quality Evaluation (COFFEE)
SLR-S111	24	<ul style="list-style-type: none">- McCall- Dromey- ISO 9126- ISO 25010- Adewumi - Omoregbe - Misra - Fernandez- Capability Maturity Model Integration (CMMi) v1.1- OpenSource Maturity Model (OSMM) Cap Gemini- Qualification and Selection of Open Source software (QSOS)- Open Business Readiness Rating (OpenBRR)- Sung - Kim - Rhew- QualOSS- QualitPso Open Source Maturity Model (OSMM)- SQO-OSS- EFFORT- FLOSS-ITS' quality model- Software Quality Systemic Model (MOSCA)- FLOSS-ILS quality model- Chirila - Juratoni - Tudor - Crețu- Open-Source Usability Maturity Model (OS-UMM)- Adewumi - Omoregbe - Misra - Fernandez- Sudhaman - Thangavel- Sohn - Lee - Seong - Kim- Kuwata - Takeda - Miura Open-Source Software Community Maturity Model- Sarrab - Rehman
SLR-S112	10	<ul style="list-style-type: none">- Basili - Briand - Melo (QCM)- Briand - Wüst - Daly - Porter- ISO 25010- Gyimothy - Ferenc - Siket- Zhou - Leung MARS model- Zhou - Leung- Olague - Eitzkorn - Messimer - Delugach- Pai - Dugan- Corrêa - Lamb - Carro - Brisolara - Mattos physical properties prediction model- Pizzi
SLR-S113	9	<ul style="list-style-type: none">- McCall- Boehm78- FURPS- Dromey- ISO 9126- ISO 25010- WBA quality model (WBAQM)

- AL-Badareen - Selamat - Din - Jabar - Turaev
- Multi-Attribute Quality Model (MAQM)

SLR-S114	8	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - QUAMOCO - Software Quality Assessment Based on Lifecycle Expectation (SQALE) - ColumbusQM - Hegedús - Software QUALity Enhancement (SQUALE) - SIG maintainability model
SLR-S115	6	<ul style="list-style-type: none"> - ISO 9126 - ISO 9001 - SPICE (ISO / IEC 15504) - CMM v1.0 (SEI) - CMM v1.1 - Malcolm Baldrige Criteria for Performance Excellence
SLR-S116	25	<ul style="list-style-type: none"> - Boehm78 - ISO 9126 - ISO 25010 - QUAMOCO - CLOUDQUAL - SAfety Quality modEl (SAQE) - Structured Code Quality Assessment Method (SCQAM) - Athanasiou - Nugroho - Visser - Zaidman - Srivastava - Kumar CK-OO quality model - Embedded Systems software Quality Model (ESQM) - ColumbusQM - Software Quality Assessment Based on Lifecycle Expectation (SQALE) - PQMOD - 2D Software Quality Model Zhang - Li - Gao - Evaluation Method for Internal Software Quality (EMISQ) - SQO-OSS - Systemic Quality Model - Carvallo - Franch - Quality Model for Object-Oriented Design (QMOOD) - Method for Intensive Design Assessments (MIDAS) - SIG maintainability model - Challa - Paul - Dada - Nerella - Srivastava - Singh - Oo quality model for web applications (Oqmw) - Morisio - Stamelos - Tsoukias - Pedrycz - Peters - Ramanna

Annexes

SLR-S117	24	<ul style="list-style-type: none">- Boehm76- McCall- DRomey- ISO 9126- ISO 25010- question-nAire for Evaluation of QUALity in TElemedicine systems (AdEQUATE)- Sohn - Lee - Seong - Kim- Sharma - Dubey- Challa - Paul - Dada - Nerella - Srivastava - Singh- Lestari - Hendradjaya Learning Management Systems (LMS) quality model- Academic Information System Quality Instrument (AISQI)- El-Rayyes - Abu-Zaid SQA model for website- Communication between Agricultural Businesses and Government (CABAG)- 2Q2U v2- Challa - Paul - Dada - Nerella - Srivastava - Singh- Behkamal - Kahani - Akbari- Software Quality Systemic Model (MOSCA)- Chen - Lin - Wang - Chang- Ziemba - Papaj - Descours- ERP Systems Quality Model (ERPSQM)- Andrian - Hendradjaya - Sunindyo E-Government G2B quality model- Rochimah - Rahmani – Yuhana- Ladányi - Tóth - Ferenc - Keresztesi- Suwawi - Darwiyanto - Rochmani
SLR-S118	10	<ul style="list-style-type: none">- ISO 9126- ISO 25010- Qualixo model- Mc Call (FCM or RADC)- Quality Model for Object-Oriented Design (QMOOD)- Boehm76- Software QUALity Enhancement (SQUALE)- Oman - Hagemeister maintainability model- Zeephongsekul - Xia - Kumar Software-Reliability Growth Model- Garcés - Ampatzoglou - Avgeriou - Nakagawa Ambient Assisted Living (AAL) quality model
SLR-S119	6	<ul style="list-style-type: none">- ISO 9126- ISO 25010- Basili (GQM)- Capability Maturity Model Integration (CMMi) v1.1- Espinilla - Domínguez-Mayo - Escalona - Mejías - Ross - Staples Model-Driven Web Engineering (MDWE) Quality Model- Domínguez-Mayo - Escalona - Mejías - Torres

SLR-S120	24	<ul style="list-style-type: none"> - McCall - Boehm78 - ISO 9126 - ISO 25010 - COQUAMO - Quality Model for Object-Oriented Design (QMOOD) - Oman - Hagemeister maintainability model - Samsung s/w Component Quality evaluation Model (SCQM) - Jeong - Kim v1.0 - QM4MAS - Evaluation Method for Internal Software Quality (EMISQ) - Garcés - Ampatzoglou - Avgeriou - Nakagawa Ambient Assisted Living (AAL) quality model - Sherman - Wijnstra - Guessi - Nakagawa - Oquendo - Maldonado - Ahrens - Frey - Pfeiffer - Bertram - Oliveira - Guessi - Feitosa - Manteuffel - Galster - Oquendo - Nakagawa - Bianchi - Santos - Felizardo - Jeong - Park - Jeong - Embedded software component quality model (EQM) - Purhonen - Paulitsch - Ruess - Sorea - Åkerholm - Fredriksson - Sandström - Crnkovic - Tamrabet - Marir - Mokhati
SLR-S121	13	<ul style="list-style-type: none"> - Boehm76 - Basili (GQM) - ISO 9126 - ISO 25010 - Software QUALity Enhancement (SQUALE) - SPICE (ISO / IEC 15504) - CMM v1.0 (SEI) - Capability Maturity Model Integration (CMMi) v1.1 - Trillium - Software Quality Measurement and Assurance Technology (SQMAT) - Wagner - Deissenboeck meta-model - QUAMOCO - Software Quality, Architecture, Process (SQuAP)
SLR-S122	16	<ul style="list-style-type: none"> - ISO 9126 - McCall - Boehm78 - FURPS - Dromey - Ghezzi - Jazayeri - Mandrioli - WBA quality model (WBAQM) - WebMAC - WebCheck - Garcés - Ampatzoglou - Avgeriou - Nakagawa Ambient Assisted Living (AAL) quality model - Suradi, Nur Razia Mohd and Kahar, Saliyah and Jamaluddin, Nor Azliana Akmal - Wibowo - Erna - Hidayah - Tabassum - Bhatti - Asghar - Manzoor - Alam Quality model for XP process & product - Malik - Nasir - Javed Quality Model for Agile Application Development - Sarrab - Elbasir - Alnaeli - Wahdiniwati - Setiawan - Wahab
SLR-S123	3	<ul style="list-style-type: none"> - ISO 25010 - Bianchi - Santos - Felizardo - Gatica - Ponce - Noël - Astudillo

Annexes

SLR-S124	4	<ul style="list-style-type: none">- Boehm78- ISO 9126- Calderón - Ruiz- Abdellatif - McCollum - McMullan
SLR-S125	2	<ul style="list-style-type: none">- ISO 25010- Gordieiev - Kharchenko
SLR-S126	9	<ul style="list-style-type: none">- McCall- AL-Badareen - Selamat - Din - Jabar - Turaev- van Koten - Gray- Zhou - Leung MARS model- Baqais - Alshayeb - Baig- Jain - Tarwani - Chug- Li - Henry- SMP Learner- Jindal - Malhotra - Jain
SLR-S127	20	<ul style="list-style-type: none">- McCall- Boehm76- Boehm78- Dromey- FURPS- FURPS+- ISO 9126- ISO 25010- Software Component Quality Model (CQM) v1.0- Software Component Quality Model (CQM) v1.1- Bertoa - Vallecillo- Rawashdeh- Matakah- OpenSource Maturity Model (OSMM) Cap Gemini- GERIC, multilayered and customisable QUALity MODEL (GEQUAMO)- QUAMOCO- SQO-OSS- AL-Badareen - Selamat - Din - Jabar - Turaev- Open Business Readiness Rating (OpenBRR)- QualOSS- Method for Intensive Design Assessments (MIDAS)
SLR-S128	9	<ul style="list-style-type: none">- ISO 9126- Abdellatif - Sultan - Jabar - Abdullah- Web-site Quality Evaluation Methodology (QEM) model- Dubey - Gulati - Rana- E-Quality framework- Selim- hexagonal elearning assessment model (HELAM)- Chua - Dyson- User satisfaction model
SLR-S129	6	<ul style="list-style-type: none">- Boehm76- McCall- Dromey- FURPS- ISO 9126- ISO 25010
SLR-S130	5	<ul style="list-style-type: none">- Goel - Okumoto (NHPP)- Jelinski - Moranda- Littlewood - Verall- Park - Kim - Shin - Baik- Juneja - Juneja - Anand

SLR-S131	21	<ul style="list-style-type: none"> - McCall - Boehm76 - ISO 9126 - ISO 25010 - Dromey - Software QQuality In Development (SQUID) - Systemic Quality Model - Factor-Strategy Model - PQMOD - QUAMOCO - 2D Software Quality Model Zhang - Li - Gao - Aspect-Oriented Software Quality Model (AOSQUAMO) - Q'Facto 12 - Software QUALity Enhancement (SQUALE) - Lee - Lee - Technical Topic Classification (TTC) quality model - Service-Oriented Architecture (SOA) quality model - MoCQA - Mobile Software Quality Model - Lochmann - Goeb Unifying Model - Component Based Quality Models (CBQM)
SLR-S132	6	<ul style="list-style-type: none"> - ISO 25010 - CMM v1.0 (SEI) - Condori-Fernandez - Lago - Calero - Moraga - Bertoa - Penzenstadler - Femmer - Sustainability-quality model
SLR-S133	5	<ul style="list-style-type: none"> - ISO 9126 - ISO 25010 - Capability Maturity Model Integration (CMMi) v1.1 - Forouzani - Chiam - Forouzani - Um - Kim - Lee - In
SLR-S134	25	<ul style="list-style-type: none"> - Boehm78 - ISO 9126 - ISO 25010 - QUAMOCO - CLOUDQUAL - SAfety Quality modEl (SAQE) - Structured Code Quality Assessment Method (SCQAM) - Athanasiou - Nugroho - Visser - Zaidman - Srivastava - Kumar CK-OO quality model - Embedded Systems software Quality Model (ESQM) - ColumbusQM - Software Quality Assessment Based on Lifecycle Expectation (SQALE) - PQMOD - 2D Software Quality Model Zhang - Li - Gao - Evaluation Method for Internal Software Quality (EMISQ) - SQO-OSS - Systemic Quality Model - Carvallo - Franch - Quality Model for Object-Oriented Design (QMOOD) - Method for Intensive Design Assessments (MIDAS) - SIG maintainability model - Challa - Paul - Dada - Nerella - Srivastava - Singh - Oo quality model for web applications (Oqmw) - Morisio - Stamelos - Tsoukias - Pedrycz - Peters - Ramanna

Annexes

SLR-S135	7	<ul style="list-style-type: none">- McCall- Boehm78- ISO 9126- ISO 25010- Anwer - Adbellatif - Alshayeb - Anjum- SATC model- Gezici - Tarhan - Chouseinglou
SLR-S136	5	<ul style="list-style-type: none">- McCall- Boehm78- FURPS- ISO 9126- ISO 25010

Annex 6. List of the 492 quality models, from 1968 to 2019

TABLE 48 - THE 492 DISTINCT QUALITY MODELS, FROM 1968 TO 2019

Id	Year	Ref.	Citation	Authors	Name	Origin	DAP ²¹	Formalism	Scope	Views ²²	Domain	URL
QM-001	1968	[33]	96	Raymond J. Rubey & R. Dean Hartwick	Rubey - Hartwick		A	Hierarchical	Product	P & U	software program	https://dl.acm.org/citation.cfm?id=810631
QM-002	1968	[34]	1235	Shooman, M. L.	Shooman		Pr	Statistics	Product	U	software program	https://trove.nla.gov.au/version/12574319
QM-003	1972	[360]	2	Jelinski, J. & Moranda, P. B.	Jelinski - Moranda		Pr	Statistics	Product	U	software program	https://www.sciencedirect.com/science/article/pii/B9780122669507500281
QM-004	1972	[361]	136	Schick, G. J. & Wolverton, R. W.	Schick - Wolverton		Pr	Statistics	Product	U	software program	https://link.springer.com/chapter/10.1007/978-3-642-99746-4_30
QM-005	1972	[362]	198	Mills, H.	Mills		Pr	Capture-Recapture	Product	M	software program	https://ci.nii.ac.jp/naid/10007992140/
QM-006	1973	[363]	576	Littlewood, B. & Verall, J. L.	Littlewood - Verall		Pr	Bayesian Analysis	Product	U	software program	https://www.jstor.org/stable/2346781?seq=1
QM-007	1973	[150]	1746	Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., MacLeod, G. J. & Merritt, M. J.	Boehm73	Boehm73	A	Hierarchical	Product	P	software program	https://ui.adsabs.harvard.edu/abs/1973tsei.book.....B/abstract
QM-008	1975	[364]	774	Musa, J. D.	Musa		Pr	Statistics	Product	U	software program	https://ieeexplore.ieee.org/document/6312856
QM-009	1975	[365]	60	Funami, Y. & Halstead, M. H.	Funami - Halstead		Pr	Statistics	Product	M	software program	https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1092&context=cstech
QM-010	1976	[366]	6	Mohanty, S. N. & Adamowicz, M	Mohanty - Adamowicz		Pr	Statistics	Product	P	software program	https://www.oldcomputerbooks.com/pages/books/C811093/jerome-fox-polytechnic-press-of-the-polytechnic-institute-of-new-york/proceedings-of-the-symposium-on-computer-software-engineering-new-york-1976-microwave-research
QM-011	1976	[42]	995	Boehm, B. W., Brown, J. R. & Lipow, M.	Boehm76	Boehm73	A	Hierarchical	Product	P	software program	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.365.8420&rep=rep1&type=pdf https://dl.acm.org/doi/10.5555/800253.807736
QM-012	1977	[367]	90	Klobert, R. K.	Klobert		Pr	Statistics	Product	M	software program	https://dl.acm.org/doi/abs/10.1145/356778.356783
QM-013	1977	[41]	88	McCall, J.A., Richards, P.K. & Walters, G.F.	Mc Call (FCM or RADC)		D	Hierarchical	Product	P	software product	https://pdfs.semanticscholar.org/82a9/18fd83f1c0addb890ef313ff892807a10a11.pdf
QM-014	1977	[368]	18	Schneidewind, N. F.	Schneidewind77		Pr	Statistics	Product	M	software program	https://ieeexplore.ieee.org/abstract/document/1646445/
QM-015	1978	[128]	1662	Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M.,	Boehm78	Boehm 1976	A	Hierarchical	Product	P	software program	https://www.abebooks.fr/9780444851055/Characteristics-Software-Quality-TRW-series-0444851054/plp

²¹ D: Definition, A: Assessment, Pr: Prediction²² P: Product, U: User, M: Manufacturer

			MacLeod, G. J. & Merritt, M. J.									
QM-016	1979	[76]	2208	Goel, A. L. & Okumoto, K.	Goel - Okumoto (NHPP)		Pr	Stochastic	Product	U	software program	https://ieeexplore.ieee.org/abstract/document/5220566
QM-017	1981	[369]	1085	Henry, D. & Kafura, D.	Henry - Kafura		A	Statistics	Product	P	large scale system	https://ieeexplore.ieee.org/document/1702877
QM-018	1983	[370]	13	Bowen, T. P., Post, J. V., Tsai, J., Presson, P. E. Schmidt & Schmidt, R. L.	Bowen - Post - McCall Tsai - Presson - Schmidt		A	Hierarchical	Product	P	software for distributed system	https://apps.dtic.mil/docs/citations/ADA137956
QM-019	1984	[217]	1452	Basili, V. R. & Weiss, D. M.	Basili (GQM)		A	Meta-model	Any	P	software product, process & project	https://www.cs.umd.edu/~basili/publications/journals/J23.pdf
QM-020	1984	[58]	5222	N. Kano, N. Seraku, F. Takahashi, and S. Tsuji	Kano		A	Meta-model	Any	U	product	https://web.archive.org/web/20110813145926/http://ci.nii.ac.jp/Detail/detail.do?LOCALID=ART0003570680&lang=en
QM-021	1985	[371]	93	Bowen, T. P., Wigle, G. B. & Tsai, I. T.	Bowen - Wigle - Tsai		A	Hierarchical	Product	P	avionic equipment	https://apps.dtic.mil/sti/citations/ADA153988
QM-022	1985	[372]	254	Langberg, N. & Singpurwalla, D.	Langberg - Singpurwalla	Jelinski – Moranda, Littlewood - Verall, Goel - Okumoto	Pr	Bayesian Analysis	Product	U	Computer software	https://epubs.siam.org/doi/abs/10.1137/0906053
QM-023	1985	[373]	262	Shen, V. Y., Yu, T. -J., Thebaut, S.M. & Paulsen, L. R.	Shen - Yu - Thebaut - Paulsen		Pr	Regression analysis	Product	M	Software program (compiler, metric tool, database system)	https://ieeexplore.ieee.org/abstract/document/1702015
QM-024	1985	[374]	56	Sunazuka, Toshihiko and Azuma, Motoei and Yamagishi, Noriko	Software Quality Measurement and Assurance Technology (SQMAT)	Boehm76, McCall	A	Hierarchical	Product	M & P	software throughout software life cycle	https://dl.acm.org/doi/abs/10.5555/319568.319605
QM-025	1986	[375]	25	Nayak, T. K.	Nayak		Pr	Statistics	Product	M	software	https://ieeexplore.ieee.org/document/4335548
QM-026	1987	[104]	3259	Garvin, D. A.	Garvin Eight dimensions of Quality		D	Hierarchical	Product	P & U	product	https://hbr.org/1987/11/competing-on-the-eight-dimensions-of-quality
QM-027	1987	[376]	-	National Institute of Standards and Technology	Malcolm Baldrige Criteria for Performance Excellence		A	Hierarchical	Organization	M	Service quality, organizational performance quality act-1987	https://www.nist.gov/director/malcolm-baldrige-national-quality-improvement-performance-quality-act-1987
QM-028	1987	[85]	841	Grady, R. B. & Caswel, D. L.	FURPS		A	Hierarchical	Product	P & U	software & hardware	https://dl.acm.org/doi/book/10.5555/26775
QM-029	1987	[233]	-	ISO/TC 176/SC 2	ISO 9001 Quality systems		A	Hierarchical	Product & Process	M	Quality systems design/developmentml, production, installation, and servicing	https://www.iso.org/standard/16533.htm
QM-030	1987	[377]	77	Evans, J. & Marciniak, J.	Evans - Marciniak	McCall	A	Hierarchical	Product & Process	P	software	https://www.semanticscholar.org/paper/Software-quality-assurance-and-management-Evans-Marciniak/e57e370af035f3ac5f69a4a52d6062c2a7235907
QM-031	1987	[163]	1323	Gilb, T.	Gilb		A	Hierarchical	Product	P	software	https://dl.acm.org/doi/10.5555/59124

Annexes

QM-032	1987	[126]	152	Kitchenham, B	Constructive QUALity MOdel (COQUAMO)	COCOMO, McCall, Boehm78	A	Hierarchical	Product & Process	M & U	software	https://digital-library.theiet.org/content/journals/10.1049/sej.1987.0014
QM-033	1988	[153]	857	Humphrey, W. S.	Process Maturity Framework		A	Hierarchical	Process	M	software process maturity	https://ieeexplore.ieee.org/document/2014
QM-034	1988	[378]	159	Card, D. N. & Agresti, W. W.	Card - Agresti		A	Statistics	Product	P	software design	https://www.sciencedirect.com/science/article/pii/0164121288900210
QM-035	1988	[379]	43	Wake, S. & Henry, S.	Wake-Henry		Pr	Statistics	Product & Process	M	software system w/ components	https://ieeexplore.ieee.org/document/10191
QM-036	1988	[380]	302	Selby, R. W. & Porter, A. A.	Selby - Porter		Pr	Decision Tree- based	Product	M	software	https://ieeexplore.ieee.org/abstract/document/9061
QM-037	1988	[381]	428	Zeithaml, V. A., Berry, L. L. & Parasuraman, A.	SERVQUAL		A	Hierarchical	Service	U	service quality	https://books.google.fr/books?hl=en&lr=&id=Rt96wAig2oC&oi=fnd&pg=PA140&dq=%22SERVQUAL:+A+multiple-item+scale+for+measuring+consumer+perceptions+of+service+quality%22&ots=p5w6czBBCT&sig=poPxOYt8CzA6vda_Sc8LdZAWnhg&redir_esc=y#v=onepage&q=%22SERVQUAL%3A%20%20multiple-item%20scale%20for%20measuring%20consumer%20perceptions%20of%20service%20quality%22&f=false
QM-038	1989	[382]	137	Gaffney Jr, J. E. & Durek, T. A.	Gaffney - Durek		Pr	Statistics	Product	M	reusable software components	https://www.sciencedirect.com/science/article/abs/pii/0950584989900050
QM-039	1990	[166]	39	Shepperd, M.	Shepperd	based on GQM	A	Hierarchical	Product	M	software program	https://www.sciencedirect.com/science/article/abs/pii/095058499090065Y
QM-040	1990	[124]	317	Khoshgoftaar, T. M. & Munson, J. C.	Khoshgoftaar - Munson		Pr	Regression analysis	Product	U	telecommunication system software product	https://ieeexplore.ieee.org/abstract/document/46879
QM-041	1991	[383]	39	Henry, S. & Wake, S.	Henry - Wake	Wake-Henry	Pr	Statistics	Product	M	software system w/ components	https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.4360030302
QM-042	1991	[384]	50	Koltun, P. & Hudson, A.	Koltun - Hudson		A	Hierarchical	Process	M	software process w/ reuse maturity	https://scholar.google.com/scholar?q=Koltun%2C%20P.%2C%20Hudson%2C%20A.%3A%20A%20Reuse%20Maturity%20Model.%20In%3A%20WISIR4%204th%20Workshop%20on%20Institutionalizing%20Software%20Reuse%2C%20Center%20for%20Innovative%20Technology%2C%20Reston%2C%20Virginia%2C%20USA%20%28November%201991%29
QM-043	1991	[154]	101	Paulk, M., Curtis, B. & Chrissis, M. B. C.	CMM v1.0 (SEI)	Humprey's Process Maturity Framework	A	Hierarchical	Process	M	software process maturity	https://pdfs.semanticscholar.org/0b81/41e3eff5b8bdcc85c59a38546eaa35301648.pdf
QM-044	1991	[385]	53	Eriksson, A. & Törn, A.	SOLE		A	Hierarchical	Product & Process	M & P & U	Information System	https://digital-library.theiet.org/content/journals/10.1049/sej.1991.0018
QM-045	1991	[374]	-	ISO/IEC JTC 1/SCISO 9126	7 Software and systems engineering		A	Hierarchical	Product	P & U	Information System	https://www.iso.org/standard/16722.html
QM-046	1991	[387]	7622	Zeithaml, V. A., Berry, L. L. & Parasuraman, A.	SERVQUAL91	SERVQUAL	A	Hierarchical	Service	U	service quality	https://books.google.fr/books?hl=fr&lr=&id=Rt96wAig2oC&oi=fnd&pg=PA114&ots=pTq-dEulBN&sig=6hDGxGyajdrN5BYbRera1wYrTlk&redir_esc=y#v=onepage&q&f=false
QM-047	1991	[388]	69	Perry, W. E.	Perry		A	Hierarchical	Project & Product	M & P & U	software	https://dl.acm.org/doi/book/10.5555/110869

QM-048	1991	[389]	1	Dyson, K. A.	QUES model (RADC)	McCall	A	Hierarchical	Product	P & U software	https://archive.org/details/DTIC_ADA252976/page/n1/mode/2up
QM-049	1991	[390]	1914	Ghezzi, Carlo and Jazayeri, Mehdi and Mandrioli, Dino	Ghezzi - Jazayeri - Mandrioli		A	Hierarchical	Product	M & P software	https://scholar.google.com/scholar?hl=en&as_sdt=0%2C31&q=allintitle%3A+%22fundamentals+of+software+engineering%22&btnG=
QM-050	1991	[391]	1375	Shackel, Brian	Shackel		A	Hierarchical	Product	U software	https://dl.acm.org/doi/10.5555/117829.117833
QM-051	1992	[152]	1076	Grady, R. B.	FURPS+	FURPS	A	Hierarchical	Product	P & U software & hardware	https://dl.acm.org/doi/book/10.5555/140207
QM-052	1992	[392]	119	Agresti, W; W. & Agresti - Evanco, W. M.	Agresti - Evanco		Pr	Statistics	Product	M ADA subsystem software	https://www.computer.org/csdl/journal/ts/1992/11/e0988/13rRUx0gerq
QM-053	1992	[393]	99	Khoshgoftaar, T. M., Pandya, A. S. & More, H. B.	Khoshgoftaar - Pandya - More		Pr	Neural Network	Product	U large commercial software	https://ieeexplore.ieee.org/document/285855
QM-054	1992	[394]	282	Briand, L. C., Basili, V. R. & Thomas, W. M.	Briand - Basili - Thomas		Pr	Pattern-recognition	Any	M & P software system & U	https://ieeexplore.ieee.org/abstract/document/177363
QM-055	1992	[395]	22	Gillies, A.	Gillies		A	Hierarchical	Product	M & P software in & U commercial environments	https://link.springer.com/article/10.1007/BF01720924
QM-056	1992	[54]	972	Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K. & Wong, M.-Y.	Orthogonal Defect Classification (ODC)		A	Hierarchical	Process	M Software development process	https://www.computer.org/csdl/journal/ts/1992/11/e0943/13rUyYBliah
QM-057	1992	[396]	196	Stephen G. Eick; Clive R. Loader; M. David Long; Lawrence G. Votta; Scott Vander Wiel	Eick - Loader - Long - Votta - Wiel Capture-Recapture		Pr	Capture-Recapture	Product	M Telecommunication (AT&T)	https://dl.acm.org/doi/10.1145/143062.143090
QM-058	1992	[397]	307	Oman, Paul and Jack	Oman - Hagemester maintainability model		Pr	Hierarchical	Product	M software systems	https://ieeexplore.ieee.org/document/242525
QM-059	1992	[398]	337	Karunanithi, Nachimuthu and Whitley, Darrell and Malaiya, Yashwant K.	Karunanithi - Whitley - Malaiya		Pr	Neural Network	Product	M software	https://ieeexplore.ieee.org/abstract/document/143107
QM-060	1993	[399]	197	Stephen G. Eick; Clive R. Loader; Scott Vander Wile; Lawrence G. Votta	Eick - Loader - Votta Capture-Recapture		Pr	Capture-Recapture	Product	M Telecommunication (AT&T)	https://dl.acm.org/doi/pdf/10.1145/143062.143090
QM-061	1993	[400]	94	Zage, W. M. & Zage, D. M.	Zage		Pr	Hierarchical	Product	M Large scale software system	https://ieeexplore.ieee.org/document/219620
QM-062	1993	[38]	340	Paulk, M., Curtis, B., Chrissis, M. B. C. & Weber, C. V.	CMM v1.1	CMM V1.0	A	Hierarchical	Process	M software process maturity	https://resources.sei.cmu.edu/asset_files/TechnicalReport/1993_005_001_16211.pdf
QM-063	1993	[401]	60	Davis, T.	SPC Reuse Capability		A	Hierarchical	Process	M software process w/ reuse maturity	https://ieeexplore.ieee.org/document/291710

Annexes

QM-064	1993	[402]	217	Dorling, A.	SPICE (ISO / IEC 15504)	CMM ...	A	Hierarchical	Process	M	software process	https://www.sciencedirect.com/science/article/abs/pii/095058499390011Q
QM-065	1993	[403]	220	Briand, L. C., Brasili, V. R. & Hetmanski, C. J.	Briand - Brasili- Hetmanski		Pr	Optimized Set Reduction	Product	M	software components (ADA)	https://ieeexplore.ieee.org/document/256851
QM-066	1993	[399]	107	Stephen G. Eick; Clive R. Loader; Scott Vander Wiel; Lawrence G. Votta	Wiel - Votta Capture- Recapture		Pr	Capture- Recapture	Product	M	Telecommunication (AT&T)	https://ieeexplore.ieee.org/abstract/document/256852
QM-067	1993	[404]	43	IEEE Std 1219- 1993	IEEE model for software maintenance		A	Hierarchical	Product	M	software maintenance	https://ieeexplore.ieee.org/document/257623
QM-068	1993	[405]	6	Bootstrap project team	Bootstrap	CMM v1.0 (SEI), ISO 9001	A	Hierarchical	Process	M	software process maturity	https://ieeexplore.ieee.org/abstract/document/210613
QM-069	1993	[406]	22257	Nielsen, Jakob	Nielsen		D	Hierarchical	Product	U	software	https://books.google.fr/books?hl=en&lr=&id=95As2OF67f0C&oi=fnd&pg=PR9&dq=%22Usability+engineering%22&ots=3czyFqbqXv&sig=meU7r42kDFjZr4LXNALXYzBbQw&redir_esc=y#v=onepage&q=%22Usability%20Engineering%22&f=false
QM-070	1993	[407]	549	Preece, Jenny and Benyon, David and Davies, G. and Keller, L. and Rogers, Y.	Preece - Benyon - Davies - Keller - Rogers		A	Hierarchical	Product	U	software	https://dl.acm.org/doi/abs/10.5555/562852
QM-071	1993	[408]	1484	Li, Wei and Henry, Sallie	Li - Henry		Pr	Regression analysis	Product	M	object oriented software	https://www.sciencedirect.com/science/article/abs/pii/016412129390077B
QM-072	1994	[409]	94	Khoshgoftaar, T. M., Lanning, D. & Pandya, A. S.	Khoshgoftaar - Lanning - Pandya		Pr	Neural Network	Product	M	telecommunication software	https://ieeexplore.ieee.org/document/272878
QM-073	1994	[410]	73	European Foundation for Quality Management (EFQM)	EFQM Excellence		A	Hierarchical	Organizat ion	M	Service quality, organizational performance quality	https://www.researchgate.net/publication/238193450_European_Foundation_for_Quality_Management_Business_Excellence_Model
QM-074	1994	[411]	26	Evanco, W. M. & Agresti W; W.	Evanco - Agresti		Pr	Statistics	Product	M	software components from NASA space flight center	https://link.springer.com/article/10.1007/BF00426946
QM-075	1994	[412]	1148	Kettinger, W. J. & Lee, C. C.	USISF - SERVQUAL	SERVQUAL91	A	Hierarchical	Service	U	Information System service quality	https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5915.1994.tb01868.x
QM-076	1994	[264]	48	Khoshgoftaar, T. M. & Szabo, R. M.	Khoshgoftaar - Szabo94		Pr	Neural Network	Product	M	software program with modules	https://ieeexplore.ieee.org/abstract/document/336789
QM-077	1994	[413]	130	S. H. Kan; V. R. Basili; L. N. Shapiro	CUPRIMDSO		A & Pr	Hierarchical	Product	P	IBM Software	https://ieeexplore.ieee.org/abstract/document/5387351
QM-078	1994	[414]	568	Coleman, Don and Ash, Dan and Lowther, Bruce and Oman, Paul	Coleman - Ash - Lowther - Oman Maintainability Index		A	Regression analysis	Product	M	software	https://ieeexplore.ieee.org/abstract/document/303623
QM-079	1994	[415]	1156	Lindland, Odd Ivar and Sindre, Guttorm and Solvberg, Arne	Lindland - Sindre - Solvberg		A	Hierarchical	Product	M	conceptual models (i.e., collection of specification statements relevant to some problem)	https://ieeexplore.ieee.org/abstract/document/268955

QM-080	1994	[416]	70	Zeephongsekul, Zeephongsekul P and Xia, G and - Xia - Kumar Kumar, S	Software-Reliability Growth Model		Pr	Stochastic	Product	M	software		https://ieeexplore.ieee.org/abstract/document/326435
QM-081	1995	[417]	155	Khoshgoftaar, T. Khoshgoftaar - M. & Lanning, D. Lanning - L.	Khoshgoftaar - Lanning - Pandya		Pr	Neural Network	Product	M	telecommunication software		https://www.semanticscholar.org/paper/A-neural-network-approach-for-early-detection-of-in-Khoshgoftaar-Lanning/024971ad130b9ab833d4e1983f50504b91a542c8?citingPapersSort=is-influential#citing-papers
QM-082	1995	[49]	711	Dromey, R. G.	Dromey	ISO 9126	A	Hierarchical	Product	P & U	software		https://ieeexplore.ieee.org/document/345830
QM-083	1995	[418]	50	Schneidewind, N. F.	Schneidewind95		A & Pr	Discriminative power techniques	Product	P & U	Space Shuttle flight software		https://link.springer.com/article/10.1007/BF02249054
QM-084	1995	[419]	51	Troster, J. & Tian, J.	Troster - Tian		Pr	Decision Tree- based	Product	U	Large-scale legacy software system		https://link.springer.com/article/10.1007/BF02249047
QM-085	1995	[420]	624	Bevan, N.	MUSIC		A	Hierarchical	Product	U	Quality of use for software		https://link.springer.com/article/10.1007/BF00402715
QM-086	1995	[421]	65	Claes Wohlin; Per Runeson and Johan Brantestam	Wohlin - Runeson - Brantestam Capture-Recapture		Pr	Capture-Recapture	Product	M	software		https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.4370050403
QM-087	1995	[422]	10	April, Alain and Coallier, François	Trillium	ISO 9001, Malcolm Baldrige Criteria for Performance Excellence	A	Hierarchical	Product & Process	M	software component, software supplier		https://ieeexplore.ieee.org/abstract/document/525963
QM-088	1996	[423]	161	Hudepohl, J. P., Aud, S. J., Khoshgoftaar, T. M., Allen, E. B. & Mayrand, J.	EMERALD Test Targeting		Pr	Statistics	Product	M	very large telecommunication software		https://ieeexplore.ieee.org/abstract/document/536459
QM-089	1996	[424]	79	Azuma, M	INSTAC model	ISO/IEC 9126	A	Hierarchical	Product	P & U	Software		https://www.sciencedirect.com/science/article/pii/S0950584995010696
QM-090	1996	[425]	251	Khoshgoftaar, T. M., Allen, E. A., S. & Goel, N.	Khoshgoftaar - Allen - Kalaichelvan - Goel		Pr	Discriminant analysis	Product	M	very large telecommunication software		https://ieeexplore.ieee.org/abstract/document/476287 https://link.springer.com/article/10.1007%2FBF00125810
QM-091	1996	[221]	2095	Basili, V. R., Briand, L. C. & Melo, W. L.	Basili - Briand - Melo (QCM)		Pr	Logistic regression	Product	M	medium-size information system, Object-Oriented SW		https://ieeexplore.ieee.org/abstract/document/544352
QM-092	1996	[426]	87	Ebert, C.	Ebert		A & Pr	Fuzzy classification	Product	M & U	large telecommunication		https://www.sciencedirect.com/science/article/pii/S0957417496000486
QM-093	1996	[427]	210	Dujmovic, J. J.	Dujmovic LSP		A	Hierarchical	Product	U	Complex hardware & software systems		http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.4388
QM-094	1996	[428]	139	Levi, M. D. & Conrad, F. G.	Levi - Conrad		A	Hierarchical	Product	U	Web sites		https://dl.acm.org/doi/abs/10.1145/234813.234819
QM-095	1996	[429]	9	Martin, R. A. & Shafer, L. H.	MITRE Software Quality Assessment Exercise (SQAE)	Boehm, McCall, ISO 9126	A	Hierarchical	Product	M & P	Software		http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.9818

Annexes

Q M-096	1996 [430]	3	Software Analysis Team at Headquarters (HQ) AFOTEC	AFOTEC Maintainability		A	Hierarchical	Product	M	Software maintainability in Air Force	https://archive.org/details/DTIC_ADA324619/page/n3/mode/2up
Q M-097	1996 [431]	312	Park, R. E. & Goethert, W. B. & Florac, W. A.	GQIM - goal-driven model	GQM	A	Meta-model	Product	M & P & U	software product	https://apps.dtic.mil/docs/citations/ADA313946
Q M-098	1996 [432]	321	Brito e Abreu, F and Melo, Walcelio	Brito e Abreu - Melo MOOD		Pr	Regression analysis	Product	M	Object-oriented software and modules	https://ieeexplore.ieee.org/abstract/document/492446
Q M-099	1996 [433]	25	Harrison, R and Samaraweera, LG and Dobie, Mark R and Lewis, Paul H	Harrison - Samaraweera - Dobie - Lewis		Pr	Statistics	Product	M	Object-oriented program and software, C++ software	https://www.sciencedirect.com/science/article/abs/pii/0950584995010815
Q M-100	1997 [267]	64	Granja-Alvarez, JC & Barranco-Garcia, MJ	Granja-Alvarez - BarrancoGarcia Maintenance Cost	COCOMO	Pr	Statistics	Process	M	software process maintenance	https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291096-908X%28199705%299%3A3%3C161%3A%3AAID-SMR148%3E3.0.CO%3B2-8
Q M-101	1997 [434]	129	Gokhale, S. S. & Lyu, M. R.	Gokhale - Lyu		Pr	Regression tree	Product	M	software for medical imaging systems	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.46.122&rep=rep1&type=pdf
Q M-102	1997 [435]	38	Takahashi, R., Muraoka, Y. & Nakamura, Y.	Takahashi - Muraoka - Nakamura	Selby - Porter	Pr	Decision Tree- based	Product	M	medium-sized piece of software (85 thousand lines of source code; 562 samples)	https://ieeexplore.ieee.org/abstract/document/630869
Q M-103	1997 [144]	12	Dujmovic, J. J. & Bayucan, A.	Dujmovic - Bayucan	Dujmovic LSP	A	Hierarchical	Product	U	Windowed environment software	http://seas.com/downloadUNReg/sample_eval/SEAS_WE.pdf
Q M-104	1997 [436]	61	Schneidewind, N. F.	Schneidewind97	Schneidewind95	A & Pr	Discriminative power techniques	Product	P & U	Space Shuttle flight software	https://ieeexplore.ieee.org/abstract/document/630888
Q M-105	1997 [437]	60	Small, R. V.	WebMAC		A	Hierarchical	Product	U	Web sites from motivational perspective	https://eric.ed.gov/?q=Assessing+the+Motivational+Quality+of+World+Wide+Websites&ff1=subWorld+Wide+Web&id=ED407930
Q M-106	1997 [438]	46	Khoshgoftaar, T. M., Allen, E. B., Halstead, R., Trio, G. P. & Flass, R.	Khoshgoftaar - Allen - Halstead - Trio - Flass		Pr	Logistic regression	Product	M & U	Large tactical military software	https://ieeexplore.ieee.org/abstract/document/648056
Q M-107	1997 [439]	239	Wilson, W. H., Rosenberg, L. H. & Hyatt, L. E.	SATC model	McCall, ISO 9126	A	Hierarchical	Product	U	software requirements	https://dl.acm.org/doi/10.1145/253228.253258
Q M-108	1997 [127]	94	Kitchenham, B., Linkman, S., Pasquini, A. & Nanni, V.	Software Quality In Development (SQUID)	McCall, ISO 9126	D	Meta-model	Product	P & U	Software quality requirements, but also any	https://link.springer.com/article/10.1023/A:1018516103435
Q M-109	1997 [440]	84	T.M. Khoshgoftaar; K. Ganesan; E.B. Allen; F.D. Ross; R. Munikoti; N. Goel; A. Nandi	Khoshgoftaar - Ganesan - Allen - Ross - Munikoti - Goel - Nandi		Pr	Case-based reasoning (CBR)	Product	M	very large telecommunication software	https://ieeexplore.ieee.org/abstract/document/630845
Q M-110	1997 [441]	60	N. Ebrahimi	Ebrahimi Capture-Recapture		Pr	Capture-Recapture	Product	M	Telecommunication (AT&T)	https://ieeexplore.ieee.org/abstract/document/624308
Q M-111	1997 [442]	91	Kurt D. Welker, Paul W. Oman, Gerald G. Atkinson	Welker - Oman - Atkinson Maintainability index		Pr	Regression analysis	Product	M	software, electronic combat system, ada, c++ software	https://onlinelibrary.wiley.com/doi/abs/10.1002/(SICI)1096-908X(199705)9:3%3C127::AID-SMR149%3E3.0.CO;2-5

QM-112	1997	[443]	255	Khoshgoftaar, Taghi M and Allen, Edward B and Hudepohl, John P and Aud, Stephen J	Khoshgoftaar - Allen - Hudepohl - Aud		Pr	Neural Network	Product	M	very large telecommunication software	https://ieeexplore.ieee.org/abstract/document/595888
QM-113	1997	[444]	233	Sabata, Bikash and Chatterjee, Saurav and Davis, Michael and Sydir, Jaroslaw J and Lawrence, Thomas F	Sabata - Chatterjee - Davis - Sydir - Lawrence		D	Hierarchical	Service	P & U	distributed systems	https://ieeexplore.ieee.org/abstract/document/609931
QM-114	1998	[269]	13	Yokoyama, Y. & Kodaira, M.	SDCH COCOMO		Pr	Statistics	Product & Process	M	cost & quality estimation, productivity & quality evaluation	https://ieeexplore.ieee.org/abstract/document/671607
QM-115	1998	[268]	41	Khoshgoftaar, M., Allen, E. B., Naik, A., Jones, W. D. & Hudepohl, J.	Khoshgoftaar - Allen - Naik - Jones - Hudepohl		Pr	Classification Tree	Product	U	very large telecommunication software	https://ieeexplore.ieee.org/document/731598
QM-116	1998	[445]	62	Khoshgoftaar, M. & Allen, E. B.	Khoshgoftaar - Allen - Kalaichelvan - Goel		Pr	Discriminant analysis	Product	M	telecommunication software, military software	https://link.springer.com/article/10.1023/A:1009736205722
QM-117	1998	[446]	11	Schneidewind, N. F.	Schneidewind98		A & Pr	Statistics	Product & Process	U	Space Shuttle flight software but not domain specific	https://ieeexplore.ieee.org/abstract/document/731249
QM-118	1998	[447]	143	Kirakowski, J. & Cierlik, B.	WAMMI		A	Hierarchical	Product	U	web sites, including commercial websites (use case in the paper)	https://journals.sagepub.com/doi/abs/10.1177/154193129804200405?casa_token=j1lbtG4ceb8AAAAA%3AqwS8Y-pVxUF4ZuY7KxC1UpoGmcz7_Eg69LjxjVmZxHu85F5-fsdfccNhnvU6ea1d-jtud7UIJgOY&
QM-119	1998	[448]	86	Ohlsson, N., Zhao, M. & Helander, M.	Ohlsson - Zhao - Helander		Pr	Discriminant coordinates	Product	M	Telecommunication software system	https://link.springer.com/article/10.1023/B:SQJO.0000042059.16470.f0
QM-120	1998	[449]	61	L.C. Briand; K.E. Emam; B.G. Freimut	Briand - Emam - Freimut - Capture-Recapture		Pr	Capture-Recapture	Product	M	NASA/ GSFC describing functional specifications for satellite ground support software	https://ieeexplore.ieee.org/abstract/document/730766
QM-121	1998	[450]	93	Per Runeson; Claes Wohlin	Runeson - Wohlin - Capture-Recapture		Pr	Capture-Recapture	Product	M	software C code	https://link.springer.com/article/10.1023/A:1009728205264
QM-122	1998	[451]	7577	Dix, Alan and Finlay, Janet and Abowd, Gregory D and Beale, Russell	Dix - Finlay - Abowd - Beale		D	Hierarchical	Product	U	software	https://books.google.fr/books?hl=en&lr=&id=luQxui8GHDcC&oi=fnd&pg=PR14&dq=%22Human-Computer+Interaction%22&ots=I529BQzQVM&sig=RlI50_irtLUoNjH8cyKAS4cQ3VM&redir_esc=y#v=onepage&q=%22Human-Computer%20Interaction%22&f=false
QM-123	1998	[452]	12	Pedrycz, W; Peters, JF; Ramanna, S	Pedrycz - Peters - Ramanna	McCall	A	Fuzzy logic	Product	P	software	https://ieeexplore.ieee.org/abstract/document/686259
QM-124	1998	[453]	37	Veenendaal, EPWMV	Software Usability Measurement Inventory (SUMI) quality model	ISO 9126	A	Hierarchical	Product	U	software	https://www.semanticscholar.org/paper/Questionnaire-based-usability-testing-Veenendaal/03de9324b0130cfb5e5cfd50c15ab98b89dc9b31

Annexes

QM-125	1999 [197]	137	Khoshgoftaar, T. M., Allen, E. B., Jones, W. D. & Hudepohl, J.	Khoshgoftaar - Allen - Jones- Allen - Naik - Jones- Hudepohl	Pr	Classification Tree	Product	U	multiple release of very large telecommunication software	https://ieeexplore.ieee.org/abstract/document/809316
QM-126	1999 [454]	41	Jones, W. D., Hudepohl, J., Khoshgoftaar, M. & Allen, E. B.	Jones - Hudepohl - Allen - Naik - Jones- Hudepohl	Pr	Logistic regression	Product	U	very large telecommunication software	https://ieeexplore.ieee.org/abstract/document/756692
QM-127	1999 [455]	144	Khoshgoftaar, T. M. & Allen, E. B.	Khoshgoftaar - Jones- Hudepohl	Pr	Logistic regression	Product	M	Large tactical military software system	https://www.worldscientific.com/doi/abs/10.1142/S0218539399000292
QM-128	1999 [198]	121	Olsina, L., Godoy, D., Lafuente, G. & Rossi, G.	Web-site QEM ISO 9126	A	Hierarchical	Product	U	Academic website & web-site artifact	https://www.tandfonline.com/doi/abs/10.1080/13614569908914709
QM-129	1999 [456]	58	Khoshgoftaar, T. M., Allen, E. B., Jones, W. D. & Hudepohl, J.	Khoshgoftaar - Allen - Jones- Hudepohl - 99a	Pr	Classification Tree	Product	M & U	Very large legacy telecommunication software system	https://www.worldscientific.com/doi/abs/10.1142/S0218194099000309
QM-130	1999 [457]	14	Khoshgoftaar, T. M., Allen, E. B., Yuan, X., Jones, W. D. & Huderpohl, J. P.	Khoshgoftaar - Allen - Yuan - Jones - Huderpohl	Pr	Classification Tree	Product	M & U	very large legacy telecommunication software	https://ieeexplore.ieee.org/abstract/document/792634
QM-131	1999 [458]	357	Gehrken D. & Turban, E.	Gehrken - Turban	D	Hierarchical	Product	U	Web sites	https://ieeexplore.ieee.org/abstract/document/772943
QM-132	1999 [115]	83	Von Dran, G.M.; Zhang, P.; Small, R. V.	Von Dran - Kano - Small	D	Hierarchical	Product	U	web site from user perspective	https://aisel.aisnet.org/amcis1999/314/?utm_source=aisel.aisnet.org%2Famcis1999%2F314&utm_medium=PDF&utm_campaign=PDFCoverPages
QM-133	1999 [459]	96	Chulani, S. & Boehm, B.	COConstructive QUALity MOdel (COQUALMO)	Pr	Statistics	Product	M	Software with 3 modes: organic, semi-detached & embedded	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.6144&rep=rep1&type=pdf
QM-134	1999 [460]	46	Voas J.	Software quality certification triangle	D	Hierarchical	Product	M	COTS, certification	https://ieeexplore.ieee.org/document/776944
QM-135	1999 [461]	19	H. Petersson & C. Wohlin	Petersson - Wohlin Capture-Recapture99	Pr	Capture-Recapture	Product	M	software	https://ieeexplore.ieee.org/abstract/document/809317
QM-136	1999 [462]	84	Benlarbi, Saïda and Melo, Walcelio L	Benlarbi - Melo	Pr	Regression analysis	Product	M	C++ software, object-oriented software	https://dl.acm.org/doi/10.1145/302405.302652
QM-137	1999 [463]	70	El Emam, Khaled and Benlarbi, Saïda and Goel, Nishith and Rai, Shesh	El Emam - Benlarbi - Goel - Rai 99	Pr	Logistic regression	Product	M	Telecommunication system, C++ software, object-oriented software	https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.1923&rank=1&q=A%20validation%20of%20object-oriented%20metrics&osm=&ossid=
QM-138	1999 [464]	367	Tang, Mei-Huei and Kao, Ming-Hung and Chen, Mei-Hwa	Tang - Kao - Chen	Pr	Statistics	Product	M	object-oriented systems and software, industrial real-time system, HMI (Human Machine Interface) software, user interface-oriented program, communication-oriented program, a real time data logging process	https://ieeexplore.ieee.org/abstract/document/809745

QM-139	2000	[465]	46	Khoshgoftaar, T. Khoshgoftaar - Jones - M., Allen, E. B., Allen - Jones- Jones, W. D. & Hudepohl, J. 2000	Hudepohl - Khoshgoftaar - Allen	Pr	Logistic regression	Product	M & U	large legacy telecommunication software system	https://link.springer.com/article/10.1023/A:1018972607783
QM-140	2000	[466]	30	Khoshgoftaar, T. Khoshgoftaar - M., Shan, R. & Allen, E. B.	Khoshgoftaar - Allen - Jones- Hudepohl - 99b	Pr	Classification Tree	Product	M & U	very large telecommunication software	https://ieeexplore.ieee.org/abstract/document/885872
QM-141	2000	[467]	109	Zhang, P., Small, Zhang et al. R. V., Von Dran, Website G. M. & Barcellos, S.	Von Dran - Zhang - Small	A	Hierarchical	Product	U	web site quality from user perspective (satisfaction/disatification)	https://ieeexplore.ieee.org/abstract/document/926847
QM-142	2000	[468]	72	Khoshgoftaar, T. Khoshgoftaar - M. & Allen, E. B. Allen2000	Khoshgoftaar - Allen98	Pr	Discriminant analysis	Product	M & U	Telecommunication system software & military system software	https://ieeexplore.ieee.org/abstract/document/877340
QM-143	2000	[469]	10	H. Petersson & C. Wohlin	Petersson - Wohlin Capture-Recapture00	Pr	Capture-Recapture	Product	M	software	https://www.wohlin.eu/ease00.pdf
QM-144	2000	[470]	19	T Thelin, P Runeson	Thelin - Runeson Capture-Recapture	Pr	Capture-Recapture	Product	M	software	https://www.sciencedirect.com/science/article/pii/S0164121299001405
QM-145	2000	[471]	889	Briand, Lionel C.; Wüst, Jürgen; Daly, John W.; Porter, D Victor	Briand - Wüst - Daly - Porter	Pr	Logistic regression	Product	M	object-oriented software	https://www.sciencedirect.com/science/article/pii/S0164121299001028
QM-146	2000	[472]	4	Barnes, S.; Vidgen, R	WebQual SERVQUAL	A	Hierarchical	Product	U	websites	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.5463&rep=rep1&type=pdf
QM-147	2000	[473]	117	Muthanna, S.; Kontogiannis, Kostas; Ponnambalam, Kumaraswamy; Stacey, B	Muthanna - Kontogiannis - Ponnambalam - Stacey	Pr	Regression analysis	Product	M	industrial software systems	https://ieeexplore.ieee.org/abstract/document/891476
QM-148	2000	[474]	854	Fenton, Norman E. ; Ohlsson, Niclas	Fenton - Ohlsson	Pr	Bayesian Network	Product	M	industrial software systems, commercial software, telecommunication system	https://ieeexplore.ieee.org/abstract/document/879815
QM-149	2000	[475]	133	Yuan, Xiaohong; Yuan - Khoshgoftaar, Taghi M.; Allen, Edward B.; Ganesan, K	Khoshgoftaar - Allen - Ganesan	Pr	Fuzzy logic	Product	M	very large telecommunication software	https://ieeexplore.ieee.org/abstract/document/888052
QM-150	2000	[476]	45	Guo, Ping; Lyu, Michael R	Guo - Lyu	Pr	Statistics	Product	M	software modules	https://ieeexplore.ieee.org/abstract/document/883780
QM-151	2000	[477]	113	Benlarbi, Saida; El Emam, Khaled; Goel, Nishith; Rai, Shesh	Benlarbi - El Emam - Goel - Rai	Pr	Logistic regression	Product	M	Telecommunication system, C++ software, object-oriented software	https://ieeexplore.ieee.org/abstract/document/885858
QM-152	2000	[478]	72	Glasberg, Daniela; El Emam, Khaled; Melo, Walcelio; Madhavji, Nazim	Glasberg - El Emam - Melo - Madhavji	Pr	Logistic regression	Product	M	Commercial Java application, object-oriented software	https://citeseerx.ist.psu.edu/viewdoc/su mmmary?doi=10.1.1.20.4483&rank=1&q= Validating%20Object-oriented%20Design%20Metrics%20on%20a%20Commercial%20Java%20Application&osm=&osid=

Annexes

QM-153	2001	[199]	161	Zhang, P.; Von Dran, G.	Expanded Website Quality Model	Zhang et al. Website Quality Model, Kano	D	Hierarchical	Product	U	Web sites	https://ieeexplore.ieee.org/abstract/document/927050/
QM-154	2001	[479]	52	Liu, Y.; Khoshgoftaar, T. M.	Liu - Khoshgoftaar		Pr	Genetic Algorithm (GA)	Product	M & U	Very large C++ application	https://ieeexplore.ieee.org/abstract/document/966814
QM-155	2001	[480]	150	Balci, O.	Modeling & Simulation Application (MSA) certification model		A	Hierarchical	Any	M & P	Modeling and simulation applications	https://dl.acm.org/doi/abs/10.1145/508366.508369
QM-156	2001	[481]	148	Leung, H. K. N.	Intranet Application Quality Model	ISO/IEC 9126	A	Hierarchical	Product	U	Intranet applications	https://www.sciencedirect.com/science/article/pii/S0378720600000604
QM-157	2001	[482]	127	Fabbrini, F., Fusani, M., Gnesi, S. & Lami, G.	Natural Language Software Requirements Specification (NLSRS) quality model		A	Hierarchical	Product	U	software requirements	http://fmt.isti.cnr.it/WEBPAPER/P11RESFQ01.pdf
QM-158	2001	[483]	10	F Losavio, L Chirinos, M Pérez	Losavio - Chirinos - Perez	ISO 9126	A	Hierarchical	Product	U	interactive system	https://www.researchgate.net/profile/Francisca_Losavio/publication/249704579_Attribute-based-techniques-to-evaluate-architectural_styles_Case_study_for_interactive_systems/links/54ee3b2f0cf2e28308645d8e/Attribute-based-techniques-to-evaluate-architectural-styles-Case-study-for-interactive-systems.pdf
QM-159	2001	[484]	67	M. Kajko-Mattsson; S. Forssander; U. Olsson	CM3 maturity model	CMM	A	Hierarchical	Process	M	software process maturity	https://ieeexplore.ieee.org/abstract/document/919135
QM-160	2001	[485]	36	S Biffi, W Grossmann	Biffi - Grossmann Inspection-Reinspection		Pr	Capture-Recapture	Product	M	software	https://ieeexplore.ieee.org/abstract/document/919089
QM-161	2001	[486]	361	Olsina, Luis ; Lafuente, Guillermo ; Rossi, Gustavo	Web-site Quality Evaluation Methodology (QEM) model	ISO 9126	A	Hierarchical	Product	U	web sites	https://link.springer.com/chapter/10.1007/3-540-45144-7_26
QM-162	2001	[487]	469	El Emam, Khaled; Melo, Walcelio; Machado, Javam C	El Emam, - Melo - Machado		Pr	Logistic regression	Product	M	object-oriented software	https://www.sciencedirect.com/science/article/pii/S0164121200000868
QM-163	2001	[61]	26	Wijnstra, J Gerben	Wijnstra		D	Hierarchical	Product	P	embedded system, medical product	https://ieeexplore.ieee.org/abstract/document/927254
QM-164	2001	[488]	227	Briand, Lionel C.; Wüst, Jürgen; Lounis, Hakim	Briand - Wüst - Lounis		Pr	Logistic regression	Product	M	object-oriented systems, object-oriented software	https://link.springer.com/article/10.1023/A:1009815306478
QM-165	2001	[489]	500	El Emam, Khaled; Benlarbi, Saïda; Goel, Nishith; Rai, Shesh N.	El Emam - Benlarbi - Goel - Rai 01	El Emam - Goel - Rai 99	Pr	Logistic regression	Product	M	Telecommunication system, C++ software, object-oriented software	https://ieeexplore.ieee.org/abstract/document/935855
QM-166	2002	[219]	14	SEI CMMI Product Development Team	Capability Maturity Model Integration (CMMi) v1.1	CMM	A	Hierarchical	Process	M	software process maturity	https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6217

QM-167	2002 [218]	1160	Bansiya, J.; Davis, C. G.	Quality Model for Object-Oriented Design (QMOOD)	Dromey, ISO 9126	A	Hierarchical	Product	P	Object-oriented software	https://ieeexplore.ieee.org/abstract/document/979986
QM-168	2002 [274]	17	Mendoza, L. E., Grimán, A. C., Pérez, M. A. & Rojas, T.	Business Portal Development Environment (PBDE) quality model	ISO 9126	A	Hierarchical	Process	U	Internet Business Portal Development Environment	https://www.tandfonline.com/doi/abs/10.1201/1078/43200.19.2.20020228/35141.7?journalCode=uism20
QM-169	2002 [490]	131	Khoshgoftaar, M., Allen, E. B.; Deng, J.	T. Khoshgoftaar - Allen - Deng 2002	Khoshgoftaar - Allen - Jones-Hudepohl - 2000	Pr	Regression tree	Product	M & U	very large telecommunication system	https://ieeexplore.ieee.org/abstract/document/1044344
QM-170	2002 [73]	22	Khoshgoftaar, M., Cukic, B.; Seliya, N.	T. Khoshgoftaar - Cukic - Seliya 2002		Pr	Analogy-based classification	Product	M & U	Telecommunication embedded systems	https://www.worldscientific.com/doi/abs/10.1142/S0218194002000883
QM-171	2002 [491]	50	Pizzi, N. J., Summers, A. R.; Pedrycz, W.	Pizzi - Summers - Pedrycz		Pr	Neural Network	Product	M	Java software	https://ieeexplore.ieee.org/abstract/document/1007518
QM-172	2002 [492]	348	Briand, L. C., Melo, W. L.; Wust, J.	MARS model		Pr	Multivariate adaptive regression	Product	M & U	Midsized Java software	https://ieeexplore.ieee.org/abstract/document/1019484
QM-173	2002 [272]	14	Zhu, H., Zhang, Y., Huo, Q.; Greenwood, S.	HASARD model		D	Meta-model	Product	U	Information Systems, Web-based information systems	https://ieeexplore.ieee.org/abstract/document/1044544
QM-174	2002 [493]	51	So, S. S., Cha, S. D.; Kwon, Y. R.	So - Cha - Kwon		Pr	Fuzzy classification	Product	M	software	https://www.sciencedirect.com/science/article/pii/S0165011401001282
QM-175	2002 [494]	27	Ramler, R., Weippl, E., Winterer, M., Schwinger, W.; Altmann, J.	Ramler - Weippl - Winterer - Schwinger - Altmann	ISO 9126	A	Hierarchical	Product	P	Web application & site	http://www.schwinger.at/PUBLICATIONS/39_2002_ICWE_A_QualityDrivenApproach_to_Web_Testing.pdf
QM-176	2002 [495]	168	Khoshgoftaar, M.; Seliya, N	T. Khoshgoftaar - Seliya		Pr	Regression tree	Product	M & U	telecommunication system	https://ieeexplore.ieee.org/abstract/document/1011339
QM-177	2002 [496]	21	J Miller; F Macdonald; J Ferguson	Miller - Macdonald - Ferguson Capture-Recapture		Pr	Capture-Recapture	Product	M	software	https://link.springer.com/article/10.1023/A:1013112826330
QM-178	2002 [497]	16	F Padberg	Padberg Capture-Recapture		Pr	Capture-Recapture	Product	M	NASA, automatic teller machine	https://dl.acm.org/doi/abs/10.1145/581339.581350
QM-179	2002 [498]	201	M. Bertoa, A. Vallecillo	Bertoa - Vallecillo	ISO 9126	A	Hierarchical	Product	P	Component based systems (COTS) & Component-based Software Development (CBSD)	https://www.researchgate.net/profile/Manuel_Bertoa/publication/2921285_Quality_Attributes_for_COTS_Components/links/02bfe50d640ce3a7f0000000.pdf
QM-180	2002 [499]	34	Albuquerque, Adriano Bessa and Belchior, Arnaldo Dias	Fuzzy Model for Software Quality Evaluation (FMSQE)		A	Hierarchical	Product	P & U	e-commerce websites	http://archive.thewebconf.org/proceedings/www2002/poster/155.pdf
QM-181	2002 [500]	159	Schubert, Petra	Extended Web Assessment Method (EWAM)		A	Hierarchical	Product	U	e-commerce websites	https://www.tandfonline.com/doi/abs/10.1080/10864415.2002.11044262
QM-182	2002 [62]	3	Purhonen, Anu	Purhonen		D	Hierarchical	Product	M & P	DSP software, telecommunication system	https://link.springer.com/chapter/10.1007/3-540-47833-7_21

Annexes

QM-183	2002	[501]	40	Mani, Anbazhagan	IBM		A	Hierarchical	Service	P & U	web services, distributed systems	https://www.ibm.com/developerworks/library/ws-quality/
QM-184	2002	[502]	99	El Emam, Khaled; Benlarbi, Saïda; Goel, Nishith; Melo, Walcelio; Lounis, Hakim; Rai, Shesh N	El Emam - Benlarbi - Goel - Melo - Lounis - Rai		Pr	Logistic regression	Product	M	Telecommunication system, industrial C++ system, C++ software, commercial Java applications, object-oriented software	https://ieeexplore.ieee.org/abstract/document/1000452
QM-185	2003	[503]	58	Khoshgoftaar, T. Liu, Y.	Khoshgoftaar - Liu - Khoshgoftaar		Pr	Genetic Algorithm (GA)	Product	M & U	Embedded software systems to customize wireless telecommunication products	https://ieeexplore.ieee.org/abstract/document/1250214
QM-186	2003	[504]	8	Zeineddine, R.; Mansour, N.	Software Quality Model for Small Organizations (SQIMSO)	SPICE, CMM, SPIRE	A	Hierarchical	Process	M	Software process for small organizations	https://link.springer.com/chapter/10.1007/978-3-540-39737-3_127
QM-187	2003	[505]	30	Herrera, E. M.; Ramírez, R. A. T.	Self-diagnosis Ramirez model	CMM	A	Hierarchical	Process	M	software process for medium & small organizations	https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1681-4835.2003.tb00100.x
QM-188	2003	[506]	119	Quah, T.-S.; Thwin, M. M. T.	Quah - Thwin		Pr	Neural Network	Product	M	Object-oriented software	https://ieeexplore.ieee.org/document/1235412
QM-189	2003	[507]	23	Ruiz, J., Calero, C.; Piattini, M.	QUINT2	ISO 9126, Ramler - Weippl - Winterer - Schwinger - Altmann	A	Hierarchical	Product	P	Web application & site	https://link.springer.com/chapter/10.1007/3-540-45068-8_69
QM-190	2003	[229]	46	Georgiadou, E.	GENeric, multilayered and customisable QUALity MOdel (GEQUAMO)		A	Hierarchical	Product	M & P & U	Software	https://link.springer.com/article/10.1023/A:1025817312035
QM-191	2003	[508]	181	Ortega, M., Pérez, M.; Rojas, T.	Systemic Quality Model	Mc Call, ISO 9126, Dromey	A	Hierarchical	Product	P & U	Software	https://link.springer.com/article/10.1023/A:1025166710988
QM-192	2003	[50]	51	A Trendowicz, T Punter	Prometheus	Basili (GQM)	A	Hierarchical	Product	M	embedded systems, J231	https://www.researchgate.net/profile/Adam_Trendowicz/publication/228598402_T_Quality_Modeling_for_Software_Product_Lines/links/0a85e532b096fa5f8e00000/T-Quality-Modeling-for-Software-Product-Lines.pdf
QM-193	2003	[509]	68	Régis P. S. Simão; Arnaldo D. Belchior	FMSQE	ISO 9126	A	Hierarchical	Product	P & U	software components, COTS, Component Based Design (CBD)	https://link.springer.com/chapter/10.1007/978-3-540-45064-1_9
QM-194	2003	[510]	55	Duijnhouwer, Frans-Willem; Widdows, Chris	OpenSource Maturity Model (OSMM) Cap Gemini	ISO 9126	A	Hierarchical	Product	M & P	Free/Lbre open source software	https://scholar.google.com/scholar?q=Duijnhouwer%2C%20F.-W.%2C%20Widdows%2C%20C.%3A%20Capgemini%20Expert%20Letter%20Open%20Source%20Maturity%20Model%2C%20Capgemini%20282003%29
QM-195	2003	[511]	22	Stefani, Antonia; Xenos, Michalis; Stavrinoudis, Dimitris	Stefani - Xenos - Stavrinoudis	ISO 9126	A	Bayesian Network	Product	P	e-commerce systems (Business to Consumer (B2C) and Business-to-Business (B2B).)	https://ieeexplore.ieee.org/abstract/document/1227023
QM-196	2003	[512]	7	S Golubic	Golubic quality build-in based quality model	ISO 9126	A	Hierarchical	Product	M & P & U	object oriented software, continuous quality verification	https://ieeexplore.ieee.org/abstract/document/1215871

QIM-197	2003	[513]	76	Mich, Luisa ; Franch, Mariangela ; Gaio, Loris	2QCV3Q		A	Hierarchical	Product	M & U	web sites, banking, tourism, academic	https://www.researchgate.net/profile/Luisa_Mich/publication/24136748_Evaluating_and_designing_the_quality_of_web_Sites/links/00b4952c579eb10d8b000000.pdf
QIM-198	2003	[514]	85	Zhang, Hongyu; Jarzabek, Stan; Yang, Bo	Zhang - Jarzabek - Yang		Pr	Bayesian Network	Product	M & P & U	product lines, Computer Aided Dispatch (CAD)	https://link.springer.com/chapter/10.1007/3-540-45017-3_45
QIM-199	2003	[515]	103	Khoshgoftaar, Taghi M; Seliya, Naeem	Khoshgoftaar - Seliya - 03b		Pr	Case-based reasoning (CBR)	Product	M	large telecommunications software system	https://link.springer.com/article/10.1023/A:1025316301168
QIM-200	2003	[516]	644	Abran, Alain; Khelifi, Adel; Suryan, Witold; Seffah, Ahmed	Abran- Khelifi - ISO 9126 Suryan - Seffah		D	Hierarchical	Product	U	software	https://link.springer.com/article/10.1023/A:1025869312943
QIM-201	2003	[517]	1586	Ran, Shuping	Ran		A	Hierarchical	Service	P & U	web services, distributed systems	https://dl.acm.org/doi/abs/10.1145/844357.844360
QIM-202	2003	[518]	216	Lee, Kangchan; Jeon, JongHong; Lee, WonSeok; Jeong, Seong-Ho; Park, Sang-Won	W3C QoS	Ran	A	Hierarchical	Service	P & U	web services, distributed systems	http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/
QIM-203	2003	[519]	132	Patel, Chintan; Supekar, Kaustubh; Lee, Yuyung	WebQ QoS		A	Hierarchical	Service	P & U	web services, distributed systems	https://link.springer.com/chapter/10.1007/978-3-540-45227-0_80
QIM-204	2003	[520]	14	Moriso, Maurizio; Stamellos, Ioannis; Tsoukias, Alexis	Moriso - Stamellos - Tsoukias	CMM v1.1	A	Hierarchical	Product & Process	M	Software entities (software products or processes), management information system (MIS), Commercial Off The Shelf (COTS)	https://www.worldscientific.com/doi/abs/10.1142/S0218194003001433
QIM-205	2004	[277]	5	Oh, J., Park, D., Lee, B., Lee, J., Hong, E. & Wu, C.	Certification model	CMM, SPICE, ISO/IEC 9126	A	Meta-model	Product & Process	M & P & U	Commercial Off The Shelf	https://link.springer.com/chapter/10.1007/978-3-540-24675-6_17
QIM-206	2004	[521]	311	Lan Guo, Yan Ma, Bojan Cukic, Harshinder Singh	Guo - Ma - Cukic - Singh		Pr	Random forest	Product	M	Nasa software (data set from Nasa)	https://ieeexplore.ieee.org/abstract/document/1383136
QIM-207	2004	[522]	20	Navica Inc.	OpenSource Maturity Model (OSMM) Navica	ISO 9126	A	Hierarchical	Product	M & P	Free/Libre open source software	http://www.navicasoft.com/pages/osmm.htm
QIM-208	2004	[523]	32	Atos Origin	Qualification and Selection of Open Source software (QSOS)		A	Hierarchical	Product	M & P	Free/Libre open source software	http://www.qsos.org/
QIM-209	2004	[524]	63	Khosravi, Khashayar; Guéhéneuc, Yann-Gaël	Software Quality STAR model		A	Hierarchical	Product	M & P	software object oriented, design pattern	https://www.academia.edu/download/30799555/041021_Kashayar_Khosravi_Technical_Report.doc.pdf
QIM-210	2004	[525]	46	Di Lucca, Giuseppe A.; Fasolino, Anna Rita; Tramontana, Porfirio;	Web-Application (WA) maintainability model	Oman - Hagemeister maintainability	A	Hierarchical	Product	M	web-applications	https://ieeexplore.ieee.org/abstract/document/1281430

Annexes

				Visaggio, Corrado Aaron								
QM-211	2004	[526]	292	Webb, Harold W.; Webb, Linda A.	SiteQual	SERVQUAL	A	Hierarchical	Product	U	B2C electronic commerce websites	https://www.emerald.com/insight/content/doi/10.1108/17410390410566724/full/html
QM-212	2004	[527]	808	Maximilien, E Michael; Singh, Munindar P	WSAF-QoS	Ran, W3C QoS	A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/1336749
QM-213	2004	[528]	6528	Avizienis, Algirdas; Laprie, J-C.; Randell, Brian; Landwehr, Carl	Avizienis - Laprie - Randell - Landwehr		A	Hierarchical	Product	M & P	software	https://ieeexplore.ieee.org/abstract/document/1335465
QM-214	2004	[529]	20	Patel, Chintan; Supekar, Kaustubh; Lee, Yugyung	SemWebQ	WebQ QoS	A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/1314773
QM-215	2004	[530]	17	Looker, Nik; Munro, Malcolm; Xu, Jie	Looker - Munro - Xu		A	Hierarchical	Service	P	web services, distributed systems	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.483.5007&rep=rep1&type=pdf
QM-216	2004	[531]	91	Marinescu, Radu; Ratiu, Daniel	Factor-Strategy Model	Boehm78, McCall (FCM)	A	Hierarchical	Product	M	software design	https://ieeexplore.ieee.org/abstract/document/1374319
QM-217	2004	[64]	18	Åkerholm, Mikael; Fredriksson, Johan; Sandström, Kristian; Crnkovic, Ivica	Åkerholm - Fredriksson - Sandström - Crnkovic		D	Hierarchical	Product	M & P	Component-based development, Component Based Software Engineering (CBSE), vehicular embedded systems (construction equipment vehicles, cars, train, heavy truck, marine engines, industrial robots)	http://www.es.mdh.se/pdf_publications/645.pdf
QM-218	2004	[532]	227	Chua, Bee Bee; Dyson, Laurel Evelyn	Chua - Dyson	ISO 9126	A	Hierarchical	Product	M & P & U	E-learning system, online learning	https://ascilite.org/conferences/perth04/procs/pdf/chua.pdf
QM-219	2005	[278]	31	Côté, M.-A., Suryan, W., Laporte, C. Y.; Martin, R. A.	Enhanced SQA	ISO/IEC 9126, SQA	A	Hierarchical	Product	M & P	Software	https://www.scopus.com/inward/record.uri?eid=2-s2.0-17444388547&doi=10.1007%2fs11219-004-5259-6&partnerID=40&md5=09ec05c221610b4c799ab6111def6568
QM-220	2005	[533]	109	Doerr, J., Kerkow, D. Koenig, T., Olsson, T.; Suzuki, T.	Experience-based NFR quality model	ISO 9126	A	Hierarchical	Product	P & U	Embedded software systems & information system	https://ieeexplore.ieee.org/abstract/document/1531057
QM-221	2005	[534]	88	Freimut, B. Denger, C.; Ketterer, M.	Defect Flow Model (DFM)	ODC	A	Hierarchical	Process	M	Software development process	https://ieeexplore.ieee.org/abstract/document/1509297
QM-222	2005	[148]	24	S. Khaddaj; G. Horgan	ADEQUATE	ISO 9126	A	Hierarchical	Product	M & P & U	software	https://www.researchgate.net/profile/G_Horgan/publication/26408254_A_Proposed_Adaptable_Quality_Model_for_Software_Quality_Assurance/links/55f457a108ae7a10cf88ec8e.pdf
QM-223	2005	[224]	231	Alain April, Jane Huffman Hayes, Alain Abran, Reiner Dumke	SMmm	CMM, CMMi,	A	Hierarchical	Process	M	software process maturity	https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.311

QM-224	2005	[535]	32	Kilsup Lee; Sung Lee - Lee Jong Lee	ISO 9126	A	Hierarchical	Product	P	software	https://ieeexplore.ieee.org/abstract/document/1434862
QM-225	2005	[536]	288	Mie Mie Thet Thwin; Tong-Seng Quah	Thwin - Quah Quah - Thwin	Pr	Neural Network	Product	M	object oriented software	https://www.sciencedirect.com/science/article/pii/S0164121204000871
QM-226	2005	[537]	13	Wasserman, A.; Pal, M.; Chan, C.	Open Business Readiness Rating (OpenBRR) ISO 9126, OpenSource Maturity Model (OSMM) Cap Gemini, OpenSource Maturity Model (OSMM) Navica	A	Hierarchical	Product	M & P	Free/Lbre open source software	http://www.openbrr.org/wiki/images/d/da/BRR_whitepaper_2005RFC1.pdf
QM-227	2005	[538]	119	Signore, Oreste	Signore	A	Hierarchical	Product	U	web sites	https://ieeexplore.ieee.org/abstract/document/1517978
QM-228	2005	[539]	70	Alvaro, Alexandre ; Santana de Almeida, Eduardo ; Romero de Lemos Meira, Silvio	Software Component Quality Model (CQM) v1.0 ISO 9126	A	Hierarchical	Product	M & P	Component-Based Software Engineering (CBSE), COTS	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.4703&rep=rep1&type=pdf
QM-229	2005	[540]	17	Chen, Chie-Bein; Lin, Chin-Tsai; Wang, Chun-Hsien; Chang, Che-Wei	Lin - Wang - Chang	A	Fuzzy logic	Product	M & U	digital video recorder software, component software	https://www.sciencedirect.com/science/article/pii/S0950584905000637
QM-230	2005	[541]	3332	Zeithaml, V. A.; Berry, L. L.; Parasuraman, A.	E-S-SQUAL SERVQUAL	A	Hierarchical	Service	U	e-service quality	https://journals.sagepub.com/doi/abs/10.1177/1094670504271156
QM-231	2005	[541]	1666	Zeithaml, V. A.; Berry, L. L.; Parasuraman, A.	E-RecS-SQUAL SERVQUAL	A	Hierarchical	Service	U	e-service quality	https://journals.sagepub.com/doi/abs/10.1177/1094670504271156
QM-232	2005	[542]	265	Dobson, Glen; Lock, Russell and Sommerville, Ian	QoSOnt WSAF-QoS	A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/1517730
QM-233	2005	[543]	162	De Bruijn, J.; Bussler, C.; Domingue, J.; Fensel, D.; Hepp, M.; Kifer, M.; König-Ries, B.; Kopecky, J.; Lara, R.; Oren, E.; Polleres, A.; Scicluna, J.; Stollberg, M.	Web Service Modeling Ontology (WSMO)	A	Hierarchical	Service	P & U	web services, distributed systems	https://www.w3.org/Submission/WSMO/
QM-234	2005	[544]	11	Amasaki, Sousuke; Takagi, Yasunari; Mizuno, Osamu; Kikuno, Tohru	Amasaki - Takagi - Mizuno - Kikuno, Tohru	Pr	Bayesian Network	Product	M	embedded system and software	https://search.ieice.org/bin/summary.php?id=e88-d_6_1134
QM-235	2005	[545]	154	Xing, Fei; Guo, Ping; Lyu, Michael R	Xing - Guo - Lyu	Pr	Classification	Product	M	medical systems	https://ieeexplore.ieee.org/abstract/document/1544736
QM-236	2005	[546]	8	Janakiram, D.; Rajasree, MS	Requirements-driven Quality Estimator (ReQuEst)	A	Hierarchical	Product	M & P	software product line, feature model	https://dl.acm.org/doi/abs/10.1145/1039174.1039194

Annexes

QM-237	2005	[547]	1057	Gyimothy, Tibor; Ferenc, Rudolf; Siket, Istvan	Gyimothy - Ferenc - Siket		Pr	Logistic regression	Product	M	object-oriented software, web and email software	https://ieeexplore.ieee.org/abstract/document/1542070
QM-238	2006	[282]	1	Wang, C. T., Lo, C. C. ; Jean, T. F.	Stack-based Markov (SBM) model		A	Stack-Based Markov	Product	P	Software	https://www.tandfonline.com/doi/abs/10.1080/10170660609509329
QM-239	2006	[125]	1	Khoshgoftaar, R. M.; Szabo, R. M.	Poisson Regression Model Fault	Khoshgoftaar - Szabo94, Lambert Zero-Inflated Poisson	Pr	Poisson regression	Product	U	large military telecommunications software system	https://www.worldscientific.com/doi/10.1142/9789812707147_0007
QM-240	2006	[201]	146	A. Rawashdeh; B. Matakah	Rawashdeh-Matakah	ISO 9126	A	Hierarchical	Product	M & P	COTS	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.8000&rep=rep1&type=pdf
QM-241	2006	[548]	203	C. van Koten; A.R. Gray	van Koten - Gray		Pr	Bayesian Network	Product	M	object oriented software	https://www.sciencedirect.com/science/article/pii/S0950584905000339
QM-242	2006	[549]	125	Miranda, Francisco Javier ; Cortés, Rosa ; Barriuso, Christina	Web Assessment Index (WAI)		A	Hierarchical	Product	U	electronic banking websites	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.3154&rep=rep1&type=pdf
QM-243	2006	[550]	63	Carvalho, Juan Pablo; Franch, Xavier	Carvalho - Franch	ISO 9126	A & Pr	Hierarchical	Product	M	COTS software	https://dl.acm.org/doi/abs/10.1145/1137702.1137706
QM-244	2006	[551]	106	Khoshgoftaar, Taghi M; Seliya, Naeem; Sundaresh, Nandini	Khoshgoftaar - Seliya - Sundaresh		Pr	Case-based reasoning (CBR)	Product	M	very large telecommunication software	https://link.springer.com/article/10.1007/s11219-006-7597-z
QM-245	2006	[552]	779	Seffah, Ahmed; Donyaaee, Mohammad; Kline, Rex B.; Padda, Harkirat K	Quality in Use Integrated Measurement (QUIM) model		A	Hierarchical	Product	U	interactive software systems, traditional GUIs-style applications, Web sites, mobile and PDA interactive services	https://link.springer.com/article/10.1007/s11219-006-7600-8
QM-246	2006	[553]	366	Wang, Xia; Vitvar, Tomas; Kerrigan, Mick; Toma, Ioan	Web Services Modeling Ontology (WSMO) QoS	Ran, W3C QoS, IBM, Web Service Modeling Ontology (WSMO)	A	Hierarchical	Service	P & U	web services, distributed systems	https://link.springer.com/chapter/10.1007/11948148_32
QM-247	2006	[554]	19	Jiang, Shanshan; Aagesen, Finn Arve	Jiang - Aagesen	WSAF-QoS	A	Hierarchical	Service	P	web services, distributed systems	https://link.springer.com/chapter/10.1007/11880905_14
QM-248	2006	[555]	47	Yeom, Gwyduk; Yun, Taewoong; Min, Dugki	Yeom - Yun - Min	Ran, IBM	A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/1611735
QM-249	2006	[556]	71	Tsesmetzis, Dimitrios T. ; Roussaki, Ioanna G. ; Papaioannou, Ioannis V. ; Anagnostou, Miltiades E	Tsesmetzis - Roussaki - Papaioannou - Anagnostou	W3C QoS, IBM	A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/1602261
QM-250	2006	[557]	54	Garcia, Diego Zuquim Guimarães ; de Toledo, Maria Beatriz Felgar	Garcia - Beatriz	Ran, WSAF-QoS	A	Hierarchical	Service	P	web services, distributed systems	https://dl.acm.org/doi/abs/10.1145/1186595.1186601

QIM-251	2006	[558]	75	Truong, Hong-Linh; Samborski-Robert; Fahringer, Thomas	Truong - Samborski - Fahringer	Avizienis - Laprie - Randell - Landwehr	A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/4031038
QIM-252	2006	[559]	429	Zhou, Yuming; Leung, Hareton	Zhou - Leung		Pr	Logistic regression	Product	M	object-oriented software	https://ieeexplore.ieee.org/abstract/document/1717471
QIM-253	2006	[560]	194	Kritikos, Kyriakos; Plexousakis, Dimitris	OWL-Q		D	Meta-model	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/4031170
QIM-254	2006	[561]	34	Mavromoustakos, Stephanos; Andreou, Andreas S	Web Application Quality Evaluation model (WAQE)	ISO 9126	A	Hierarchical	Product	P & U	web sites	https://www.inderscienceonline.com/doi/abs/10.1504/IJWET.2007.011529
QIM-255	2007	[562]	73	Andreas S. Andreou; Marios Tziakouris	Original software components quality model (OSQCM)	ISO 9126	A	Hierarchical	Product	P	original Component based systems (note: <i>this is not COTS</i>)	https://www.sciencedirect.com/science/article/pii/S0950584906000437
QIM-256	2007	[284]	52	Ormandjieva, S., Hussain, I.; Kosseim, L.	Ormandjieva - Hussain - Kosseim		A	Hierarchical	Product	U	software requirements	https://dl.acm.org/doi/abs/10.1145/1295074.1295082
QIM-257	2007	[287]	37	Sangeeta Neti; Hausi A. Muller	Neti - Muller	ISO 9126	A	Hierarchical	Product	P	self-healing systems	https://ieeexplore.ieee.org/abstract/document/4228606
QIM-258	2007	[563]	234	Yuming Zhou; Hareton Leung	Zhou - Leung	MARS model	Pr	Multivariate adaptive regression	Product	M	object oriented software	https://www.sciencedirect.com/science/article/pii/S0164121206003372
QIM-259	2007	[564]	73	Taibi, Davide ; Lavazza, Luigi ; Morasca, Sandro	Open Business Quality Rating (Open BQR)	Open Business Readiness Rating (OpenBRR)	A	Hierarchical	Product	M & P	Free/Libre open source software	https://link.springer.com/chapter/10.1007/978-0-387-72486-7_14
QIM-260	2007	[565]	34	Sibisi, Mbusi; Van Waveren, Cornelis Cristo	Mbusi - Van Waveren	ISO 9126	A	Hierarchical	Product	P	Software	https://ieeexplore.ieee.org/abstract/document/4401495
QIM-261	2007	[566]	142	Deissenboeck, Florian; Wagner, Stefan; Pizka, Markus and Teuchert, Stefan	Deissenboeck - Boehm78, Wagner - Pizka Wagner - Teuchert - Deissenboeck meta-model		A	Hierarchical	Product	M	embedded systems, matlab simuling models	https://ieeexplore.ieee.org/abstract/document/4362631
QIM-262	2007	[567]	84	Winter, Sebastian; Wagner, Stefan; Deissenboeck, Florian	Winter - Wagner - Deissenboeck	Mc Call (FCM or RADC), Wagner - Deissenboeck meta-model	A	Hierarchical	Product	U	software with user interface	https://link.springer.com/chapter/10.1007/978-3-540-92698-6_7
QIM-263	2007	[568]	386	Heitlager, Ilja; Kuipers, Tobias; Visser, Joost	SIG maintainability model	ISO 9126	A & Pr	Hierarchical	Product	M	software	https://ieeexplore.ieee.org/abstract/document/4335232
QIM-264	2007	[569]	581	Kim, Sunghun; Zimmermann, Thomas; Whitehead Jr, James; Zeller, Andreas	Kim - Zimmermann - Whitehead Jr - Zeller		Pr	Statistics	Product	M	open-source software	https://ieeexplore.ieee.org/abstract/document/4222610
QIM-265	2007	[570]	198	Kanmani, S.; Uthariaraj, V Rhymend; Sankaranarayanan, V.; Thambidurai, P	Kanmani - Uthariaraj - Sankaranarayanan - Thambidurai	Khoshgoftaar - Allen - Hudepohl - Aud, Briand - Wüst - Daly - Porter, Yuan - Khoshgoftaar - Allen - Ganesan	Pr	Neural Network	Product	M	Object-oriented software and modules	https://www.sciencedirect.com/science/article/pii/S0950584906001005

Annexes

QM-266	2007	[571]	24	Quirchmayr, Gerald; Funilkul, Suree; Chutimaskul, Wichian	e-Government Services (E-GSQ) model	ISO 9126	A	Hierarchical	Service	M	e-government services	https://sit.kmutt.ac.th/wichian/Paper/eGovServiceQualityModel.pdf
QM-267	2007	[572]	813	Dagger, Tracey S.; Sweeney, Jillian C.; Johnson, Lester W	Dagger - Sweeney - Johnson	SERVQUAL	A	Hierarchical	Service	U	health e-service quality	https://journals.sagepub.com/doi/10.1177/1094670507309594?icid=int.sj-abstract.similar-articles.3
QM-268	2007	[573]	128	Henriksson, Anders; Yi, Yiori; Frost, Belinda; Middleton, Michael	e-Government website evaluation tool (eGwet)		A	Hierarchical	Product	P	e-government websites	https://www.inderscienceonline.com/doi/abs/10.1504/EG.2007.013984
QM-269	2007	[574]	16	Ren, Kaijun; Chen, Jinjun; Chen, Tao; Song, Junqiang; Xiao, Nong	Ren - Chen - Chen - Song - Xiao		A	Hierarchical	Service	P	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/4438599
QM-270	2007	[575]	42	De Bruijn, J.; Bussler, C.; Domingue, J.; Fensel, D.; Hepp, M.; Kifer, M.; König-Ries, B.; Kopecky, J.; Lara, R.; Oren, E.; Polleres, A.; Scicluna, J.; Stollberg, M.	Web Service Modeling Ontology (WSMO) v1.4	Web Service Modeling Ontology (WSMO)	A	Hierarchical	Service	P & U	web services, distributed systems	http://www.wsmo.org/TR/d2/v1.4/D2v1-4_20070216.pdf
QM-271	2007	[576]	34	Yu, Weider D.; Radhakrishna, Rachana B.; Pingali, Sumana; Kolluri, Vijaya	Yu - Radhakrishna - Pingali - Kolluri		A	Hierarchical	Service	P & U	web services, distributed systems	https://journals.sagepub.com/doi/abs/10.1177/0037549707079228?
QM-272	2007	[577]	13	Kang, YunHee	Kang		A	Hierarchical	Product	P	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/4426288
QM-273	2007	[578]	96	Giallonardo, Ester; Zimeo, Eugenio	onQoS-QL	Ran, WSAF-QoS	A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/4273423
QM-274	2007	[579]	9	Lee, Youngkon; Yeom, Gwyduk	Lee - Yeom		A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/4296922
QM-275	2007	[580]	39	Sung, Won Jun; Kim, Ji Hyeok; Rhew, Sung Yul	Sung - Kim - Rhew	ISO 9126	A	Hierarchical	Product	M & P	open source software	https://ieeexplore.ieee.org/abstract/document/4460693
QM-276	2007	[581]	236	Pai, Ganesh J; Dugan, Joanne Bechta	Pai - Dugan		Pr	Bayesian Network	Product	M	object-oriented software	https://ieeexplore.ieee.org/abstract/document/4302779
QM-277	2007	[582]	47	Wagner, Stefan; Deissenboeck, Florian	Wagner - Deissenboeck meta-model		D	Meta-model	Product	M & P & U	software	https://ieeexplore.ieee.org/abstract/document/4273468
QM-278	2007	[583]	1278	Selim, Hassan M	Selim		A	Hierarchical	Product	U	E-learning system, website	https://www.sciencedirect.com/science/article/pii/S0360131505001338
QM-279	2008	[202]	263	Rüdiger Lincke, Jonas Lundberg; Welf Löwe	Lincke - Lundberg - Löwe	ISO 9126	A	Hierarchical	Product	M	air traffic management	https://dl.acm.org/doi/abs/10.1145/1390630.1390648
QM-280	2008	[288]	7	Li Zhang ; Lin Li ; Hui Gao	2D Software Quality Model Zhang - Li - Gao	McCall (FCM)	A & Pr	Hierarchical	Product	M	software	https://ieeexplore.ieee.org/abstract/document/5172787

QM-281	2008 [289]	2	A. A. Hamada; M. N. Moustafa; H. I. Shaheen	FUPRIMDSO	CUPRIMDSO	A & Pr	Hierarchical	Product	P	software	https://ieeexplore.ieee.org/abstract/document/4773015/
QM-282	2008 [290]	1	Xiuzhen Feng, Yijian Liu	Feng - Liu		A & Pr	Hierarchical	Product	U	information sharing software	https://ieeexplore.ieee.org/abstract/document/4722856
QM-283	2008 [291]	9	O. Alfonso ; K. Domínguez; L. Rivas; M. Pérez; L. Mendoza; M. Ortega	Software Quality Systemic Model (MOSCA)	ISO 9126, Dromey, Basili (GQM), Systemic quality model	A & Pr	Hierarchical	Product	M & P & U	Free/Libre open source software	https://ieeexplore.ieee.org/abstract/document/4483214
QM-284	2008 [584]	15	Foutse Khomh; Yann-Gaël Guéhéneuc	PQMOD	QMOOD, Dromey	Pr	Pattern-recognition	Product	M & P	object oriented system	https://dl.acm.org/doi/abs/10.1145/1753196.1753199
QM-285	2008 [585]	16	Haiguang Fang	Educational Software Quality Hierarchy Triangle (ESHTri) model	McCall, Boehm78, ISO 9126	A	Hierarchical	Product	M & P	Web-based learning	https://ieeexplore.ieee.org/abstract/document/5163782
QM-286	2008 [586]	101	Chang, Che-Wei; Wu, Cheng-Ru; Lin, Hung-Lung	Chang - Wu - Lin	ISO 9126	A	Fuzzy logic	Product	M & P	software	https://link.springer.com/article/10.1007/s11219-007-9035-2
QM-287	2008 [587]	94	Sharma, Arun; Kumar, Rajesh; Grover, PS	Sharma - Kumar - Grover	ISO 9126	A	Hierarchical	Product	M & P	software components and component-based systems (CBS)	https://dl.acm.org/doi/abs/10.1145/1449603.1449613
QM-288	2008 [39]	28	Choi, Yoonjung; Lee, Sungwook; Song, Houp; Park, Jingoo; Kim, SunHee	Samsung s/w Component Quality evaluation Model (SCQM)	ISO 9126	A	Hierarchical	Product	P	Software component, COTS (Commercial Off-The-Shelf)	https://ieeexplore.ieee.org/abstract/document/4493757
QM-289	2008 [588]	6	ELdesouky, Aly I.; Arafat, Hesham; Ramzey, Hazem	Web-site Quality Evaluation Method (QEM)	Web-site Quality Evaluation Methodology (QEM) model	A	Hierarchical	Product	U	web sites	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.476.4352&rep=rep1&type=pdf
QM-290	2008 [589]	84	Shim, Bingu; Choue, Siho; Kim, Suntae; Park, Sooyong	Service-Oriented Architecture (SOA) Design quality model	QMOOD	A	Hierarchical	Product	M & P	Service-Oriented Architecture (SOA) system	https://ieeexplore.ieee.org/abstract/document/4724572
QM-291	2008 [590]	1	CITY, CNR, FBK, INRIA, Lero, POLIMI, SZTAKI, TUW, UniDue, UPM, UStutt, Tilburg	SCube Quality Reference Model	ISO 9126	A	Hierarchical	Product	P & U	service-based application (SBA)	https://s-cube-network.eu/results/deliverables/wp-jra-1.3/Reference_Model_for_SBA.pdf
QM-292	2008 [222]	177	Samoladas, Ioannis; Gousios, Georgios; Spinellis, Diomidis; Stamelos, Ioannis	SQO-OSS	Basili (GQM), ISO 9126	A	Hierarchical	Product & Process	M & P	open source software, open source community process	https://link.springer.com/chapter/10.1007/978-0-387-09684-1_19
QM-293	2008 [591]	81	Stefani, Antonia; Xenos, Michalis	Stefani - Xenos	ISO 9126	A	Bayesian Network	Product	P & U	e-commerce systems (Business to Consumer (B2C) and Business-to-Business (B2B).)	https://link.springer.com/article/10.1007/s11219-007-9032-5#citeas
QM-294	2008 [592]	10	Heck, Petra; Eekelen, MCJD	LaQuSo software product certification model (LSPCM)		A & Pr	Hierarchical	Product	M	software certification	https://research.tue.nl/en/publications/laquso-software-product-certification-model-lspcm

Annexes

QM-295	2008 [593]	34	Plösch, Reinhold; Gruber, Harald; Hentschel, A.; Körner, Ch. Pomberger, Gustav; Schiffer, Stefan; Saft, Matthias; Storck, S	Evaluation Method for Internal Software Quality (EMISQ)	ISO 9126, SATC model, FURPS	A	Hierarchical	Product	M & P software	https://link.springer.com/article/10.1007/s11334-007-0039-7
QM-296	2008 [594]	11	Laval, Jannik ; Bergel, Alexandre ; Ducasse, Stéphane	Qualixo model		A	Hierarchical	Product	M & P software	https://hal.inria.fr/inria-00498482/
QM-297	2008 [595]	252	Gondra, Iker	Gondra		Pr	Neural Network	Product	M software, NASA's metric data program	https://www.sciencedirect.com/science/article/pii/S0164121207001240
QM-298	2008 [596]	13	Micali, F.; Cimino, S	Web Q-Model		A	Hierarchical	Product	M & U websites	https://www.semanticscholar.org/paper/Web-Q-Model%3A-a-new-approach-to-the-quality-Cimino-Sperone/041f5490aa7a86c4528b2a40358b66ded000223f
QM-299	2008 [597]	52	Chutimaskul, Wichian; Funilkul, Suree; Chongsuphajai siddhi ddhi, Vithida	Chutimaskul - Funilkul - Chongsuphajai siddhi	ISO 9126	A	Hierarchical	Service	U e-government services	https://dl.acm.org/doi/abs/10.1145/1509096.1509117
QM-300	2008 [598]	41	Abramowicz, Witold; Hofman, Radoslaw; Suryan, Witold; Zyskowski, Dominik	Abramowicz - Hofman, Suryan - Zyskowski	ISO 25010	A	Hierarchical	Product	M & P web services, IT & U solutions	https://www.researchgate.net/profile/Witold_Suryan/publication/44261676_SQuaRE_based_Web_Services_Quality_Model/links/0fcfd508ea67fd2f05000000/SQuaRE-based-Web-Services-Quality-Model.pdf
QM-301	2008 [599]	48	Artaiam, Natee; Senivongse, Twittie	Artaiam - Senivongse	IBM	A	Hierarchical	Service	P & U web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/4617464
QM-302	2008 [600]	41	Tran, Vuong Xuan	WS-QoSOnto	WSAF-QoS	A	Hierarchical	Service	P & U web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/4730492
QM-303	2008 [601]	15	Mittal, Harish; Bhatia, PK.; Goswami, Puneet	Mittal - Bhatia - Goswami		Pr	Fuzzy logic	Product	M software modules	https://www.researchgate.net/profile/Harish_Mittal/publication/267752698_Software_Quality_Assessment_Based_on_Fuzzy_Logic_Technique/links/5459b6ac0cf26d5090ad1098.pdf
QM-304	2008 [602], [603]	34	Etxeberría, Leire; Sagardui, Goiuria	Etxeberría - Sagardui		A	Hierarchical	Product	M & P software product line, feature model, arcade game maker	https://ieeexplore.ieee.org/abstract/document/4492407 https://link.springer.com/chapter/10.1007/978-3-540-68073-4_16
QM-305	2008 [604]	5	Raffoul, Eduardo; Domínguez, Kenyer; Pérez, María; Mendoza, Luis E.; Grimán, Anna C	FLOSS-ITS' quality model	Software Quality Systemic Model (MOSCA), ISO 9126, Dromey	A	Hierarchical	Product	P open source software, open source community process, issue tracking systems (ITS)	https://www.researchgate.net/profile/Anna_Griman/publication/228943503_Quality_model_for_the_selection_of_floss-based_issue_tracking_system/links/0046351472fc06abba000000/Quality-model-for-the-selection-of-floss-based-issue-tracking-system.pdf
QM-306	2008 [605]	53	Olague, Hector M.; Etkorn, Letha H.; Messimer, Sherri L.; Delugach, Harry S	Olague - Etkorn - Messimer - Delugach		Pr	Logistic regression	Product	M object-oriented software, iterative or agile development	https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.366

QM-307	2008	[67]	12	Sherman, Trudy Sherman				D	Hierarchical	Product	P	embedded system	https://link.springer.com/chapter/10.1007/978-1-4020-8741-7_95
QM-308	2008	[66]	11	Paulitsch, Michael, Ruess, Harald; Sorea, Maria	Paulitsch - Ruess - Sorea			D	Hierarchical	Product	P	avionic embedded system, ultra-critical embedded system in aerospace industry	https://link.springer.com/chapter/10.1007/978-3-540-88479-8_26
QM-309	2009	[606]	18	Klās, M., Nakao, H., Elberzhager, F., Münch, J.	Hybrid Defect Content and Effectiveness Early Prediction (HyDEEP)	COQUALMO		Pr	Statistics	Product	M	Software requirements Software	https://link.springer.com/article/10.1007/s10664-009-9112-1
QM-310	2009	[203]	160	Mohagheghi P., Dehlen V., Neple T.	Mohagheghi - Dehlen - Neple			A	Hierarchical	Product	M	model-based software	https://www.sciencedirect.com/science/article/pii/S0950584909000457
QM-311	2009	[295]	31	Brcina R., Bode S., Riebisch M.	Brcina - Bode - Riebisch	ISO 9126, Basili (GQM)		A	Hierarchical	Product	M	software	https://ieeexplore.ieee.org/abstract/document/4839246
QM-312	2009	[607]	22	Bergel, Alexandre ; Denier, Simon ; Ducasse, Stéphane ; Laval, Jannik ; Bellingard, Fabrice ; Vaillergues, Philippe ; Balmas, Françoise ; Mordal-Manet, Karine	Software QUALity Enhancement (SQUALE)	ISO 9126, ISO 25010, Qualixo model		A	Hierarchical	Product	M & P	software	https://ieeexplore.ieee.org/abstract/document/4812772
QM-313	2009	[608]	68	Petrinja, Etiel; Nambakam, Ranga; Sillitti, Alberto	QualitPso Open Source Maturity Model (OSMM)	CMM v1.0 (SEI)		A	Hierarchical	Product	M & P	Free/Libre open source software	https://ieeexplore.ieee.org/abstract/document/5071358
QM-314	2009	[609]	263	Behkamal, Behshid; Kahani, Mohsen; Akbari, Mohammad Kazem	Behkamal - Kahani - Akbari	ISO 9126		A	Hierarchical	Product	P & U	B2B software	https://www.sciencedirect.com/science/article/pii/S0950584908001109
QM-315	2009	[610]	42	Kumar, Avadhesh; Grover, PS.; Kumar, Rajesh	Aspect-Oriented Software Quality Model (AOSQUAMO)	ISO 9126		A	Fuzzy logic	Product	P	object oriented and component based	https://dl.acm.org/doi/abs/10.1145/1598732.1598736
QM-316	2009	[37]	14	Carvalho, Fernando; Meira, Silvio	Embedded software component quality model (EQM)	ISO 9126, ISO 25010		A	Hierarchical	Product	M & P	Embedded software components, COTS	https://ieeexplore.ieee.org/abstract/document/5090533
QM-317	2009	[611]	28	Srivastava, Praveen Ranjan; Kumar, Krishan	Srivastava - Kumar	McCall, ISO 9126, ISO 25010, CMMi		A	Statistics	Product	P	Java software, software	https://link.springer.com/chapter/10.1007/978-3-642-00405-6_19
QM-318	2009	[184]	5	Jamwal, Ranbireshwar S.; Jamwal, Deepshikha	Jamwal - Jamwal	ADEQUATE		A	Hierarchical	Product	M & P & U	very large information system	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.469.7953&rep=rep1&type=pdf
QM-319	2009	[612]	84	Wagner, Stefan	Wagner Activity-Based Quality Model (ABQM)			A & Pr	Bayesian Network	Product	M & P	software	https://www.sciencedirect.com/science/article/pii/S0950584910001175

Annexes

QM-320	2009	[613]	6	Moraga, Ma Ángeles; Calero, Coral; Garzás, Javier; Piattini, Mario	Portlet Quality Model (PtQM)	ISO 9126	A	Hierarchical	Product	M & P	Java portlet, Portlet component -based websites	https://www.sciencedirect.com/science/article/pii/S0920548908000573
QM-321	2009	[614]	44	Soto, Martín; Ciołkowski, Marcus	QualOSS	ISO 9126, Capability Maturity Model Integration (CMMi) v1.1, Open Business Readiness Rating (OpenBRR), Qualification and Selection of Open Source software (QSOS)	A	Hierarchical	Product & Process	M & P	open source software, sustainability, software process, open source community	https://ieeexplore.ieee.org/abstract/document/5314237
QM-322	2009	[615]	15	Plösch, Reinhold; Gruber, Harald; Körner, Christian; Pomberger, Gustav; Schiffer, Stefan	Technical Topic Classification (TTC) quality model		D	Meta-model	Product	M & P	Java, C#, and C++ software	https://www.academia.edu/download/46798845/A_Proposal_for_a_Quality_Model_Based_on_20160626-29291-16msrfw.pdf
QM-323	2009	[616]	84	Hussain, Azham; Kutar, Maria	Hussain - Kutar Basili (GQM)		A	Hierarchical	Product	U	Mobile devices, mobile application software	https://www.researchgate.net/profile/Azham_Hussain/publication/267368031_Useability_Metric_Framework_for_Mobile_Phone_Application/links/54b35d3e0cf2318f0f9541fd.pdf
QM-324	2009	[617]	42	Del Bianco, Vieri; Lavazza, Luigi; Morasca, Sandro; Taibi, Davide	QualiPSo	Basili (GQM), ISO 9126	A	Hierarchical	Product	M & U	open source software (OSS)	https://link.springer.com/chapter/10.1007/978-3-642-02032-2_18
QM-325	2009	[618]	116	Papadomichelaki, Xenia; Mentzas, Gregoris	Papadomichelake GovQuali		A	Hierarchical	Service	U	e-government services	https://link.springer.com/chapter/10.1007/978-3-642-03516-6_14
QM-326	2009	[619]	30	Frutos, Hénar Muñoz; Kotsiopoulos, Ioannis; Gonzalez, Luis Miguel Vaquero; Merino, Luis Rodero	BREIN QoS ontology	Ran, W3C QoS, Web Services Modeling Ontology (WSMO) - QoS, Ren - Chen - Chen - Song - Xiao, Tsesmetzis - Roussaki - Papaioannou - Anagnostou	A	Hierarchical	Service	P & U	web services, distributed systems	https://link.springer.com/chapter/10.1007/978-3-642-02121-3_42
QM-327	2009	[620]	18	Chang, Heejung; Lee, Kangsun	Chang - Lee		A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/5260426
QM-328	2009	[621]	14	Al-Masri, Eyhab; Mahmoud, Qusay H	Al-Masri - Mahmoud		A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/5346882
QM-329	2009	[622]	11	Hongxia Tong, Jian Cao, ShenSheng Zhang, Yujie Mou	Tong - Cao - Zhang - Mou		A	Fuzzy logic	Service	P	web services, distributed systems	https://www.emerald.com/insight/content/doi/10.1108/03684920910944236/full/html
QM-330	2009	[623]	161	Comuzzi, Marco; Pernici, Barbara	Comuzzi - Pernici	Ran, W3C QoS, IBM, WSQM (preliminary version)	A	Hierarchical	Service	P & U	web services, distributed systems	https://dl.acm.org/doi/abs/10.1145/1541822.1541825

QM-331	2009	[624]	49	Balfagih, Zain; Hassan, Mohd Fadzil	Balfagih - Hassan		A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/5077045
QM-332	2009	[625]	20	Li, Shuyu; Zhou, Juan	Li - Zhou Juan		A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/5366383
QM-333	2009	[626]	4	Reddy D, Kiran Kumar; Maralla, Karthiek; Thirumaran, M	Reddy - Maralla - Thirumaran		A	Hierarchical	Service	P & U	web services, distributed systems	https://dl.acm.org/doi/abs/10.1145/1517303.1517317
QM-334	2009	[627]	4	Marchetto, Alessandro; others	Oo quality model for web applications (Oqmw)		A & Pr	Classification	Product	M	Object-oriented software, web applications	http://jase.tku.edu.tw/articles/jase-200912-12-4-10
QM-335	2009	[628]	679	Ozkan, Sevgi; Koseler, Refika	Hexagonal eLearning Assessment Model (HELAM)		A	Hierarchical	Product	M & P & U	E-learning system, B-learning system (Blended)	https://www.sciencedirect.com/science/article/pii/S0360131509001584
QM-336	2009	[629]	23	Cappiello, Cinzia; Kritikos, Kyriakos; Metzger, Andreas; Parkin, Michael; Pernici, Barbara; Plebani, Pierluigi; Treiber, Martin	Cappiello - Kritikos - Metzger - Parkin - Pernici		A	Hierarchical	Service	P & U	web services, distributed systems	https://www.econstor.eu/obitstream/10419/58147/1/716006251.pdf#page=37
QM-337	2009	[630]	43	Mabrouk, Nebil Ben; Georgantas, Nikolaos; Issarny, Valérie	Mabrouk - Georgantas - Issarny	WSQM	D	Meta-model	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/5068817
QM-338	2009	[631]	146	Alonso-Ríos, David; Vázquez-García, Ana; Mosqueira-Rey, Eduardo; Moret-Moret-Bonillo, Vicente	Alonso-Ríos - Vázquez-García - Mosqueira-Rey - Moret-Moret-Bonillo		D	Hierarchical	Product	U	software	https://www.tandfonline.com/doi/abs/10.1080/10447310903025552
QM-339	2010	[296]	3	Qi Yu-dong; Zhu Ai-hong; Xie Xiao-fang; Yan Xiao-bin	Conceptual Model Quality	ISO 25010	A	Hierarchical	Product	M & U	software, model	https://ieeexplore.ieee.org/abstract/document/5622740
QM-340	2010	[297]	67	Letouzey, Jean-Louis; Coq, Thierry	Software Quality Assessment Based on Lifecycle Expectation (SQALE)	ISO 9126	A	Hierarchical	Product	M & P	ADA software	https://ieeexplore.ieee.org/abstract/document/5617180
QM-341	2010	[632]	29	Castillo, Isi; Losavio, Francisca; Matteo, Alfredo; Bøegh, Jørgen	REquirements, Aspects and Software Quality (REASQ) model	ISO 25010, ISO 9126, Dromey	A	Meta-model	Product	M & P	Object-oriented software development	http://www.jot.fm/issues/issue_2010_07/article4.pdf
QM-342	2010	[103]	18	Bawane, Neelam; Srikrishna, CV	Bawane - Srikrishna	ISO 9126	A	Hierarchical	Product	M & P & U	software	https://www.researchgate.net/profile/Azween_Abdullah/publication/47542466_Maximizing_Lifetime_of_Homogeneous_Wireless_Sensor_Network_through_Energy_Efficient_Clustering_Method/links/0c96053194f7e3d16c000000.pdf#page=66

Annexes

QM-343	2010	[633]	44	Alvaro, Alexandre ; Santana de Almeida, Eduardo ; Romero de Lemos Meira, Silvio	Software Component Quality Model (CQM) v1.1	ISO 9126, ISO 25010, Software Component Quality Model (CQM) v1.0	A	Hierarchical	Product	M & P	Component-Based Software Engineering (CBSE), COTS	https://dl.acm.org/doi/abs/10.1145/1668862.1668863
QM-344	2010	[102]	7	Kalaimagal, Sivamuni; Srinivasan, Rengaramanujam	Q'Facto 12	ISO 25010	A	Hierarchical	Product	M & P & U	Component based systems (COTS) & Component-based Software Development (CBSD)	https://dl.acm.org/doi/abs/10.1145/1734103.1734116
QM-345	2010	[634]	7	Kalaimagal, Sivamuni; Srinivasan, Rengaramanujam	Q'Facto 10	ISO 9126	A	Hierarchical	Product	M & P & U	Component based systems (COTS) & Component-based Software Development (CBSD)	https://www.researchgate.net/publication/251065355_Q'_FACTO_10-Commercial_off-the-shelf_component_quality_model_proposal
QM-346	2010	[635]	28	Malak, Ghazwa ; Sahraoui, Houari ; Badri, Linda ; Badri, Mourad	Malak - Sahraoui - Badri - Badri Web quality model	Basili (GQM), ISO 9126	A	Bayesian Network	Product	U	Web-based applications	https://dl.acm.org/doi/abs/10.1145/1806916.1806918
QM-347	2010	[636]	93	Lew, Philip ; Olsina, Luis ; Zhang, Li	2Q2U	ISO 25010	A	Hierarchical	Product	P & U	web-application	https://link.springer.com/chapter/10.1007/978-3-642-13911-6_15
QM-348	2010	[637]	42	Herrera, Mayte; Moraga, Ma Angeles; Caballero, Ismael; Calero, Coral	Quality in Use Model for Web Portals (QiUWeP)	ISO 25010	A	Hierarchical	Product	U	websites, web-portal	https://link.springer.com/chapter/10.1007/978-3-642-16985-4_9
QM-349	2010	[638]	6	Fan, Miao; Luo, Yi; Wu, Guoshi; Fu, Xiangling	Fan - Luo - Wu - Fu	ISO 9126	A & Pr	Fuzzy logic	Product	P	software	https://ieeexplore.ieee.org/abstract/document/5690372
QM-350	2010	[639]	24	Luckey, Markus; Baumann, Andrea; Méndez, Daniel; Wagner, Stefan	Extended Activity-Based Quality Model (ABQM)	Wagner Activity-Based Quality Model (ABQM), Deissenboeck - Wagner - Pizka - Teuchert - Girard maintainability 2D model	A	Meta-model	Product	M	embedded systems, Matlab simulating models	https://dl.acm.org/doi/abs/10.1145/1809101.1809101
QM-351	2010	[640]	97	Mohanty, Ramakanta; Ravi, Vadlamani; Patra, Manas Ranjan	Mohanty - Ravi - Patra		Pr	Classification	Service	M & P	web services, distributed systems	https://www.sciencedirect.com/science/article/pii/S0957417410001028
QM-352	2010	[641]	9	Yin, Baocai; Yang, Huirong; Fu, Pengbin; Chen, Xiaobo	Yin - Yang - Fu - Chen	ISO 9126, WSAF-QoS, Tsesmetzis - Roussaki - Papaioannou - Anagnostou, WS-QoSOnto	A	Hierarchical	Service	P & U	web services, distributed systems	https://link.springer.com/chapter/10.1007/978-3-642-15470-6_18
QM-353	2010	[642]	14	Pan, Zhedan; Baik, Jongmoon	Pan - Baik		A	Hierarchical	Service	P & U	web services, distributed systems	https://www.riverpublishers.com/journal/journal_articles/RP_Journal_1540-9589_943.pdf
QM-354	2010	[643]	33	Zhang, Shu; Song, Meina	Zhang - Song		A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/5440283

QM-355	2010 [644]	13	Corrêa, Ulisses Brisolara ; Lamb, Luis ; Carro, Luigi ; Brisolara, Lisane; Mattos, Júlio	Corrêa - Lamb - Carro - Brisolara - Mattos physical properties prediction model		Pr	Neural Network	Product	M	object-oriented software, embedded systems	https://ieeexplore.ieee.org/abstract/doc ument/5578300
QM-356	2010 [645]	16	Domínguez- Mayo, FJ.; Escalona, MJ.; Mejías, M.; Torres, AH	Domínguez- Mayo - Escalona - Mejías - Torres	ISO 9126	A	Hierarchical	Product	P & U	Model-Driven Web Engineering (MDWE)	https://ieeexplore.ieee.org/abstract/doc ument/5507323
QM-357	2011 [646]	1	Hwang, Sun Myung; Im, Soobin	Korean Software Process Quality Certification Model	SPICE (ISO / IEC 15504), Capability Maturity Model Integration (CMMi) v1.1	A	Hierarchical	Process	M	software certification	https://ieeexplore.ieee.org/abstract/doc ument/5954295
QM-358	2011 [23]	-	ISO/IEC JTC 1/SCISO/IEC/IEEE 7 Software and engineering technical committee	ISO 9126	ISO 9126	A	Hierarchical	Product	M & P & U	software product & in use	https://www.iso.org/standard/35733.ht ml
QM-359	2011 [300]	6	Coq T., Rosen J. P.	Software Quality Assessment Based on Lifecycle Expectation (SQALE) ADA	Software Quality Assessment Based on Lifecycle Expectation (SQALE)	A	Hierarchical	Product	M & P	ADA software	https://link.springer.com/chapter/10.100 7/978-3-642-21338-0_5
QM-360	2011 [647]	17	Upadhyay, Nitin; Despande, Bharat M.; Agrawal, Vishnu P	Upadhyay - Despande - Agrawal Software Component Quality Model (SCQM)	ISO 9126	A	Hierarchical	Product	M	Component based systems (COTS) & Component-based Software Development (CBSD)	https://link.springer.com/chapter/10.100 7/978-3-642-17857-3_40
QM-361	2011 [648]	18	Anas Bassam AL-Badareen, Mohd Hasan Selamat, Jamilah Din, Marzanah A. Jabar, Sherzod Turaev	AL-Badareen - Selamat - Din - Jabar - Turaev	ISO 9126	A	Hierarchical	Product	U	software	http://www.universitypress.org.uk/journ als/ami/20-649.pdf
QM-362	2011 [301]	16	Lochmann, Klaus; Heinemann, Lars	Lochmann - Heinemann		A	Hierarchical	Product	M & P	java software, open source	https://dl.acm.org/doi/abs/10.1145/198 5374.1985378
QM-363	2011 [302]	44	Nabil D., Mosad A., Hefny H.A.	WBA quality model (WBAQM)	ISO 9126	A	Hierarchical	Product	P & U	web-based application	https://www.sciencedirect.com/science/ article/pii/S1110866511000405
QM-364	2011 [187]	37	Polillo, Roberto	Polillo	ISO 25010	A	Hierarchical	Product	P & U	web [2.0] sites	https://link.springer.com/chapter/10.100 7/978-3-642-27997-3_25
QM-365	2011 [649]	30	Lochmann, Klaus; Goeb, Andreas	Lochmann - Goeb Unifying Model	ISO 25010, Wagner Activity- Based Quality Model (ABQM), Service- Oriented Architecture (SOA) Design quality model	D	Meta-model	Product	P & U	software	https://dl.acm.org/doi/abs/10.1145/202 4587.2024591

Annexes

QM-366	2011	[650]	127	Bakota, Tibor; Hegedűs, Péter; Körtvélyesi, Péter; Ferenc, Rudolf; Gyimóthy, Tibor	ColumbusQM		A & Pr	Statistics	Product	M	software	https://ieeexplore.ieee.org/abstract/document/6080791
QM-367	2011	[651]	19	Olsina, Luis ; Lew, Philip ; Dier, Alexander ; Rivera, Belen	2Q2U v2	2Q2U, ISO 25010	A	Hierarchical	Product	P & U	Web application & site, social web-applications	https://www.riverpublishers.com/journal/journal_articles/RP_Journal_1540-9589_1042.pdf
QM-368	2011	[652]	5	Murthy, PVR.; Kumar, Saravana; Sharma, Tushar; Rao, Kiron	Dynamic Analysis for Internal Software Quality (DAISQ) model	ISO 9126	A	Hierarchical	Product	M	software component or system, C++ software	https://ieeexplore.ieee.org/abstract/document/6032365
QM-369	2011	[649]	30	Goeb, Andreas; Lochmann, Klaus	Service-Oriented Architecture (SOA) quality model	QUAMOCO	A	Meta-model	Product	P	service oriented architecture, distributed system	https://dl.acm.org/doi/abs/10.1145/2024587.2024593
QM-370	2011	[653]	25	Dominic, PDD; Jati, Handaru	Dominic - Jati		A	Fuzzy logic	Product	M & P	air-lines websites	https://www.inderscienceonline.com/doi/abs/10.1504/IJBIR.2011.042451
QM-371	2011	[654]	14	Rekik, Rim; Kallel, Ilhem	Fuzz-Web model	ISO 9126	A	Fuzzy logic	Product	U	institutional websites	https://ieeexplore.ieee.org/abstract/document/6088194
QM-372	2011	[655]	13	Singh, Yogesh; Malhotra, Ruchika; Gupta, Poonam	Singh - Malhotra - Ruchika; Gupta		Pr	Logistic regression	Product	U	webpages, websites	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.214.223&rep=rep1&type=pdf#page=34
QM-373	2011	[656]	32	Bocciarelli, Paolo; D'Ambrogio, Andrea	Bocciarelli - D'Ambrogio		Pr	Hierarchical	Service	M & P & U	web services, distributed systems	https://link.springer.com/article/10.1007/s10270-010-0150-3
QM-374	2011	[657]	2	Qiu, Junping; Yu, Fan	Qiu - Yu		D	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/6006212
QM-375	2011	[658]	3	Debnath, Narayan; Martellotto, Paola; Daniele, Marcela; Riesco, Daniel; Montejano, Germán	Debnath - Martellotto - Daniele - Riesco - Montejano		A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/6060134
QM-376	2011	[659]	7	Mens, Tom; Leandro, Habra, Naji; Vanderose, Benoit; Kamseu, Flora	MoCQA		D	Meta-model	Product	M	evolving software-intensive systems	https://ieeexplore.ieee.org/abstract/document/5741344
QM-378	2011	[660]	59	Azar, Danielle; Vybihal, Joseph	Azar - Vybihal		Pr	Classification	Product	M	object-oriented software	https://www.sciencedirect.com/science/article/pii/S0950584910002144
QM-378	2011	[661]	9	Łukasz Radliński	Radliński	ISO 25010	Pr	Bayesian Network	Product	M & U	Information Systems	https://reshistorica.journals.umcs.pl/ai/article/view/3331
QM-379	2011	[662]	147	Bagheri, Ebrahim; Gasevic, Dragan	Bagheri - Gasevic		A	Hierarchical	Product	M & P	software product line, feature model	https://link.springer.com/article/10.1007/s11219-010-9127-2
QM-380	2011	[663]	113	Müller, Tristan	FLOSS-ILS quality model		A	Hierarchical	Product	M	open source software, open source community process, Integrated Library Systems (ILS)	https://www.emerald.com/insight/content/doi/10.1108/10650751111106573/full/html?journalCode=oclc

QM-381	2011	[664]	4	Chirila, Ciprian-Bogdan; Juratoni, Dana; Tudor, Dacian; Crețu, Vladimir	Chirila - Juratoni - Tudor - Crețu	ISO 9126	A	Hierarchical	Product	M	open source software, open source community process, static code analyzers	https://ieeexplore.ieee.org/abstract/document/5873026
QM-382	2011	[665]	58	Challa, Jagat Sesh; Paul, Arindam; Dada, Yogesh; Nerella, Singh Venkatesh; Srivastava, Praveen Ranjan; Singh, Ajit Pratap	Challa - Paul - Dada - Nerella - Srivastava -	ISO 9126	A	Fuzzy logic	Product	P	software, IT industry	https://www.researchgate.net/publication/228518831_Integrated_Software_Quality_Evaluation_A_Fuzzy_Multi-Criteria_Approach
QM-383	2011	[666]	5	Espinilla, M.; Domínguez-Mayo, F.J.; Escalona, M.J.; Mejías, M.; Ross, M.; Staples, G	Espinilla - Domínguez-Mayo - Escalona - Mejías - Ross - Staples	Domínguez-Mayo - Escalona - Mejías - Torres	A	Hierarchical	Product	P	Model-Driven Web Engineering (MDWE)	https://link.springer.com/chapter/10.1007/978-3-642-25661-5_85
QM-384	2011	[667]	73	Abdellatif, Majdi; Sultan, Abu Bakar Md.; Jabar, Marzanah A.; Abdullah, Rusli	Abdellatif - Sultan - Jabar - Abdullah	ISO 9126	A	Hierarchical	Product	P & U	E-learning system, Learning Management System (LMS), website	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.877.3797&rep=rep1&type=pdf
QM-385	2011	[668]	11	Um, Taehoon; Kim, Neunghoe; Lee, Donghyun; In, Hoh Peter	Um - Kim - Lee - In	ISO 9126	D	Hierarchical	Product	M & P	agile software development	https://ieeexplore.ieee.org/abstract/document/5954362
QM-386	2012	[304]	3	L. Cheikhi; A. Abran; J. Desharnais	ISBSG quality model	ISO 9126	A	Hierarchical	Product	P & U	software	https://ieeexplore.ieee.org/abstract/document/6389405
QM-387	2012	[160]	135	Wagner, Stefan; Lochmann, Klaus; Heinemann, Lars; Kläs, Michael; Trendowicz, Adam; Plösch, Reinhold; Seidi, Andreas; Goeb, Andreas; Streit, Jonathan	QUAMOCO	ISO 25010	A	Hierarchical	Product	P	Java and C# software	https://ieeexplore.ieee.org/abstract/document/6227106
QM-388	2012	[48]	18	Mayr, Alois; Plösch, Reinhold; Kläs, Michael; Lampasona, Constanza; Saft, Matthias	Embedded Systems software Quality Model (ESQM)	ISO 25010, QUAMOCO	A	Hierarchical	Product	P & U	embedded systems software	https://ieeexplore.ieee.org/document/6405376
QM-389	2012	[307]	5	Wan-Jiang, HAN; Tian-Bo, Lu	Wan-Jiang - Tian-Bo		A	Hierarchical	Product	M & P	communication system, embedded system	https://ieeexplore.ieee.org/abstract/document/6340726
QM-390	2012	[29]	2	Jeong, Hwa Young; Kim, Yoon Ho	Jeong - Kim v1.0		A	Hierarchical	Product	M & P	lightweight component software, embedded system	https://www.scientific.net/AMM.121-126.4907
QM-391	2012	[46]	21	Jeong, Hwa-Young; Kim, Yoon-Ho	Jeong - Kim v1.1	ISO 9126, Jeong - Kim v1.0	A	Hierarchical	Product	M & P	systems software	https://pdfs.semanticscholar.org/7ab2/64cb8b7e20beed57d9960d3b5c36d09e85cc.pdf
QM-392	2012	[669]	39	Franke, Dominik; Kowalewski,	Mobile Software Quality Model	McCall (FCM), Boehm78	A	Hierarchical	Product	P	mobile devices, mobile application software	https://ieeexplore.ieee.org/abstract/document/6319241

Annexes

												Stefan; Weise, Carsten
QM-393	2012	[670]	37	Yu, Liguo; Mishra, Alok	Yu - Mishra		A & Pr	Logistic regression	Product	M & P	commercial software	https://www.tandfonline.com/doi/abs/10.1080/16843703.2012.11673302
QM-394	2012	[671]	15	Singh, Brijendra; Kannoja, Suresh; Prasad	Singh - Kannoja		Pr	Neural Network	Product	P	C, C++ software, software components	https://www.scrip.org/html/4-9301401_19840.htm
QM-395	2012	[672]	115	Malhotra, Ruchika; Jain, Ankita	Malhotra - Jain	QMOOD, Brito e Abreu - Melo MOOD	Pr	Regression analysis	Product	M	Object-oriented software and modules	https://www.koreascience.or.kr/article/JAKO201222340312043.page
QM-396	2012	[673]	39	Dubey, Sanjay Kumar; Gulati, Anubha; Rana, Ajay	Dubey - Gulati - Rana		D	Hierarchical	Product	U	software	https://www.researchgate.net/publication/267364055_Integrated_Model_for_Software_Usability
QM-397	2012	[674]	145	Bhattacharya, Debjani; Gulla, Umesh; Gupta, MP	Bhattacharya - Gulla - Gupta		A	Hierarchical	Service	U	e-government services	https://www.emerald.com/insight/content/doi/10.1108/17410391211224408/full/html
QM-398	2012	[675]	84	Elling, Sanne; Lentz, Leo; de Jong, Menno; Van den Bergh, Huub	Website Evaluation Questionnaire (WEQ)		A	Hierarchical	Product	U	e-government websites	https://www.sciencedirect.com/science/article/pii/S0740624X12000342
QM-399	2012	[676]	46	Moser, Oliver; Rosenberg, Florian; Dustdar, Schahram	Moser - Rosenberg - Dustdar		A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/6231591
QM-400	2012	[677]	27	Cabrera, Oscar; Franch, Xavier	GESSE	ISO 9126	A	Hierarchical	Service	M & P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/6240444
QM-401	2012	[678]	1	OASIS Web Services Quality Model Technical Committee	WSQM	ISO 9126	A	Hierarchical	Service	M & P & U	web services, distributed systems	http://docs.oasis-open.org/wsqm/WS-Quality-Factors/v1.0/cos01/WS-Quality-Factors-v1.0-cos01.pdf
QM-402	2012	[679]	12	Phalnikar, Rashmi; Khutade, Pradnya A	Phalnikar - Khutade		D	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/6409157
QM-403	2012	[680]	27	Nadanam, Padmapriya; Rajmohan, R	Nadanam - Rajmohan	ISO 9126	A	Hierarchical	Service	P & U	web services, distributed systems	https://ieeexplore.ieee.org/abstract/document/6395991
QM-404	2012	[681]	13	Rashid, Ekbal; Patnaik, Srikanta; Bhattacharjee, Vandana	Rashid - Patnaik - Bhattacharjee		Pr	Case-based reasoning (CBR)	Product	M	operating systems	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.252.9529&rep=rep1&type=pdf
QM-405	2012	[682]	70	Raza, Arif; Capretz, Luiz Fernando; Ahmed, Faheem	Open-Source Usability Maturity Model (OS-UMM)		A	Hierarchical	Product	U	open source software, open source community process	https://www.sciencedirect.com/science/article/pii/S0747563212000209
QM-406	2012	[683]	9	El-Rayyes, Emad; Abu-Zaid, Ibrahim M	El-Rayyes - Abu-Zaid SQA model for website		D	Hierarchical	Product	M & P	website, server/client	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.6461&rep=rep1&type=pdf
QM-407	2012	[68]	31	Guessi, Milena; Nakagawa, Elisa; Yumi; Oquendo - Flavio; Maldonado, José Carlos	Guessi - Nakagawa - Maldonado		D	Hierarchical	Product	M & P	embedded system, software architecture	https://dl.acm.org/doi/abs/10.1145/2304656.2304661
QM-	2012	[684]	163	Masoumi, Davoud;	E-Quality framework		A	Hierarchical	Product	M & U	E-learning system, website	https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2729.2011.00440.x

QM-409	2012 [685]	29	Lindström, Berner Park, Jinhee; Kim, Hyeon-Jeong; Shin, Ju-Hwan; Baik, Jongmoon	Park - Kim - Shin - Baik	Goel - Okumoto (NHPP)	Pr	Stochastic	Product	M	embedded software	https://ieeexplore.ieee.org/abstract/document/6258310
QM-410	2013 [51]	10	Ahrens, Dirk; Frey, Andreas; Pfeiffer, Andreas; Bertram, Torsten	Ahrens - Frey - Pfeiffer - Bertram	ISO 9126	A	Hierarchical	Product	M & P	Embedded system, software architecture, automotive systems	https://link.springer.com/article/10.1007/s00450-011-0185-x
QM-411	2013 [47]	1	Jeong, Hwa-Young	Jeong	ISO 9126	D	Hierarchical	Product	M & P	Embedded system, embedded software, software as a service (SaaS)	https://ieeexplore.ieee.org/abstract/document/6685411
QM-412	2013 [686]	3	Calero, C.; Bertoa, MF	25010+S	ISO 25010	A	Hierarchical	Product	M & P & U	sustainable software product and software engineering	https://ieeexplore.ieee.org/abstract/document/6606421
QM-413	2013 [313]	4	Adewumi, Adewole; nd Omoregbe, Nicholas; Misra, Sanjay; Fernandez, Luis	Adewumi - Omoregbe - Misra - Fernandez	SQO-OSS	A	Hierarchical	Product & Process	M	open source web applications, repository software	https://ieeexplore.ieee.org/abstract/document/6755361
QM-414	2013 [314]	47	Li, Ke; Xiao, Junchao; Wang, Yongji; Wang, Qing	Li - Xiao - Wang - Wang		A	Regression analysis	Product	P	software from crowdsourcing platform	https://ieeexplore.ieee.org/abstract/document/6649922
QM-415	2013 [317]	18	Hegedús, Péter	Hegedús	ColumbusQM	A & Pr	Statistics	Product	M	C# software, IT systems	https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3835
QM-416	2013 [687]	3	Venkatasubram anyam, Radhika D.; Nayak, Snigdha	Dynamic Analysis Technical Model (DATM)	Dynamic Analysis for Internal Software Quality (DAISQ) model	A	Hierarchical	Product	M	Software code, c++, c#, java	http://www.lnse.org/papers/36-IE0007.pdf
QM-417	2013 [688]	9	Wang, Xiaojing; Ceberio, Martine; Virani, Shamsnaz; Garcia, Angel; Cummins, Jeremy	Wang - Ceberio - Virani - Garcia - Cummins	QMOOD	A & Pr	Fuzzy logic	Product	P	software	http://www.cs.utep.edu/vladik/2012/tr12-48.pdf
QM-418	2013 [689]	29	Baliyan, Niyati; Kumar, Sandeep	Baliyan - Kumar		A	Fuzzy logic	Service	U	software as a service (SaaS)	https://ieeexplore.ieee.org/abstract/document/6684439
QM-419	2013 [690]	32	Zahra, Sobia; Khalid, Asra; Javed, Ali	Mobile Application Quality Model	ISO 9126	A	Hierarchical	Product	P	mobile devices, mobile application software	https://www.researchgate.net/profile/Al_i_Javed2/publication/274048560_An_Efficient_and_Effective_New_Generation_Objective_Quality_Model_for_Mobile_Applications/links/56a8dc1d08aeea2a20497e7e.pdf
QM-420	2013 [691]	18	Ulman, M.; Vostrovský, V.; Tyrychtr, J.	Communication between Agricultural Businesses and Government (CABAG)	ISO 25010	A	Hierarchical	Service	U	e-government services, agricultural e-government services	https://ageconsearch.umn.edu/record/162303/
QM-421	2013 [692]	14	Rababah, Osama; Hamtini, Thair; Harfoushi, Osama; Al-	Rababah - Hamtini - Harfoushi - Al-	ISO 9126	A	Hierarchical	Product	P	e-gouvernement websites	https://www.scirp.org/journal/paperinformation.aspx?paperid=38972

Annexes

QM-422	2013	[693]	2	Dixit, DA	Shboul, Bashar; Obiedat - Obiedat, Ruba; Nawafleh, Sahem	Component Based Quality Models (CBQM)		D	Hierarchical	Product	M & P & U	Component based systems (COTS) & Component-based Software Development (CBSD)	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.831.43&rep=rep1&type=pdf
	2013	[694]	35	Ahmed, Moataz A.; Al-Jamimi, Hamdi A	Ahmed - Al-Jamimi			Pr	Fuzzy logic	Product	M	software	https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2013.0046
QM-424	2013	[695]	7	Duan, Yucong; Kattapury, Ajay; Getahun, Fekade; Elfakiz, Du; Wencai	Duan - Kattapury - Getahun - Elfakiz - Du			D	Meta-model	Product	M & P	software product line, feature model	https://ieeexplore.ieee.org/abstract/document/6630310
QM-425	2013	[696]	39	Aversano, Lerina; Tortorella, Maria	EFFORT	QualiPSo, Qualification and Selection of Open Source software (QSOS), Open Business Readiness Rating (OpenBRR)		A	Hierarchical	Product	M & P	open source software, open source community process, ERP systems	https://www.sciencedirect.com/science/article/pii/S0950584913000311
QM-426	2013	[697]	26	Pizzi, Nick J	Pizzi			Pr	Fuzzy classification	Product	M & P	software component, biomedical data analysis systems	https://www.sciencedirect.com/science/article/pii/S0020025513003319
QM-427	2013	[698]	5	Mayr, Alois; Plösch, Reinhold; Saft, Matthias	SAfety Quality modEl (SAQE)			A	Hierarchical	Product	M & P & U	object-oriented software, C/C++ systems	https://ieeexplore.ieee.org/abstract/document/6619487
QM-428	2013	[699]	22	Srivastava, Praveen Kumar; Ranjan, Krishan	Srivastava - Kumar CK-OO quality model			A & Pr	Hierarchical	Product	M	object-oriented software and design	https://ieeexplore.ieee.org/abstract/document/6526872
QM-429	2013	[700]	29	Samarthyam, Ganesh; Suryanarayana, Girish; Sharma, Tushar; Gupta, Shrinath	Method for Intensive Design Assessments (MIDAS)	QMOOD		A	Hierarchical	Product	M & P	Object-oriented software	https://ieeexplore.ieee.org/abstract/document/6606640
QM-430	2013	[69]	14	Oliveira, LBR; Guessi, M.; Feitosa, D.; Manteuffel, C.; Galster, M.; Oquendo, F.; Nakagawa, EY	Oliveira - Guessi - Feitosa - Manteuffel - Galster - Oquendo - Nakagawa			D	Hierarchical	Product	M & P	embedded system	https://www.rug.nl/research/portal/publications/an-investigation-on-quality-models-and-quality-attributes-for-embedded-systems(96345a91-b9e8-4d91-864a-8db11f17b597).html
QM-431	2013	[312]	36	alero, Coral; Moraga, M.; Bertoa, Manuel F	Calero - Moraga - Bertoa	ISO 25010		D	Hierarchical	Product	M & P	Sustainable software, Green IT	https://arxiv.org/abs/1309.1640
QM-432	2013	[701]	149	Penzenstadler, Birgit; Femmer, Henning	Penzenstadler - Femmer			D	Hierarchical	Product & Process	M & P	software engineering	https://dl.acm.org/doi/abs/10.1145/2451605.2451609
QM-433	2013	[702]	5	Alexander Roth, Andreas; Ganser, Horst; Lichter, Bernhard; Rumpe	Roth - Ganser - Lichter - Rumpe	Lindland - Sindre - Rumpe Quality model for Models		A	Hierarchical	Product	M	UML models libraries for software	https://arxiv.org/abs/1408.5707

QM-434	2014	[703]	5	Small, R. V. & Arnone, M. P.	WEBCHECK	WebMAC	A	Hierarchical	Product	U	Web sites from motivational perspective	https://eric.ed.gov/?q=Assessing+the+Motivational+Quality+of+World+Wide+Websites&id=EJ1040844
QM-435	2014	[324]	4	Yildiz, Ekrem; Bilgen, Semih; Tokdemir, Gul; Cagiltay, Nergiz E.; Erturan, Y Nasuh	Yildiz - Bilgen - ISO 25010		A	Hierarchical	Product	P	mobile devices, mobile application software, business to customer (B2C) mobile applications	https://link.springer.com/chapter/10.1007/978-3-319-10359-4_21
QM-436	2014	[325]	3	Ronchieri, Elisabetta ; Canaparo, Marco ; Salomoni, Davide	Ronchieri - Canaparo - Salomoni		Pr	Discriminant analysis	Product	M & P	Scientific computing infrastructures, distributed software	https://content.iospress.com/articles/journal-of-integrated-design-and-process-science/jid140016
QM-437	2014	[329]	33	Gupta, Deepak; Ahlawat, Anil; Sagar, Kalpna	Gupta - Ahlawat - Sagar		D	Hierarchical	Product	U	software	https://ieeexplore.ieee.org/abstract/document/7019810/
QM-438	2014	[704]	77	Hien, Nguyen Manh	Hien		D	Hierarchical	Service	U	e-government services	https://publications.waset.org/9997019/a-study-on-evaluation-of-e-government-service-quality
QM-439	2014	[705]	7	Masood, Zafar; Xuequn, Shang; Yousaf, Jamal	Software Engineering Methodology (SEM) quality model		A	Hierarchical	Product	U	software engineering	http://www.lnse.org/papers/127-IT3009.pdf
QM-440	2014	[706]	8	Adline, Agasta; Ramachandran, M.	Adline - Ramachandran		Pr	Genetic Algorithm (GA)	Product	M	software	https://www.ijareeie.com/upload/2014/apr14-special/49_ramachandraneaswari.pdf
QM-441	2014	[707]	11	Puri, Aditi; Singh, Harshpreet	Puri - Singh		Pr	Genetic Algorithm (GA)	Product	M	open-source software systems	https://www.researchgate.net/profile/Harshpreet_Singh/publication/276197625_Genetic_Algorithm_Based_Approach_for_Finding_Faulty_Modules_in_Open_Source_Software_Systems/links/559a30c408ae99aa62cc8bb3/Genetic-Algorithm-Based-Approach-for-Finding-Faulty-Modules-in-Open-Source-Software-Systems.pdf
QM-442	2014	[708]	36	Zhang, Guoheng; Ye, Huilin; Lin, Yuqing	Zhang - Ye - Lin		Pr	Hierarchical	Product	M & P	software product line, feature model	https://link.springer.com/article/10.1007/s11219-013-9197-z
QM-443	2014	[709]	15	Kuwata, Yoshitaka; Takeda, Kentaro; Miura, Hiroshi	Kuwata - Takeda - Miura Open-Source Software Community Maturity Model	CMM v1.1	A	Hierarchical	Product	M	open source software, open source community process	https://www.sciencedirect.com/science/article/pii/S1877050914012290
QM-444	2014	[327]	60	Sarrab, Mohamed; Rehman, Osama M Hussain	Sarrab - Rehman	SERVQUAL	A	Hierarchical	Product	M & P & U	open source software, open source community process, open source network tools, learning management systems.	https://www.sciencedirect.com/science/article/pii/S0965997813001798
QM-445	2014	[710]	120	Zheng, Xianrong; Martin, Patrick; Brohman, Kathryn; Da Xu, Li	CLOUDQUAL	SERQUAL	A	Hierarchical	Service	P & U	cloud services	https://ieeexplore.ieee.org/abstract/document/6740846

Annexes

QM-446	2014	[711]	6	Gupta, Shrinath; Singh, Himanshu Kumar; Venkatasubram anyam, Radhika D.; Uppili, Umesh	Structured Code Quality Assessment Method (SCQAM)				A	Hierarchical	Product	M	Software for Industry, Energy, Healthcare, and Infrastructure and Cities sectors	https://dl.acm.org/doi/abs/10.1145/2597008.2597806
QM-447	2014	[712]	86	Athanasiou, Dimitrios; Nugroho, Ariadi; Visser, Joost; Zaidman, Andy	Athanasiou - Nugroho - Visser - Zaidman	SIG maintainability model, ISO 9126			A	Hierarchical	Product	M & P	Open source software, software code	https://ieeexplore.ieee.org/abstract/document/6862882
QM-448	2014	[713]	12	Indah Le stari, Bayu Hendradjaya	Le stari - Hendradjaya Learning Management Systems (LMS) quality model	ISO 9126			A	Hierarchical	Product	P	Learning management systems	https://ieeexplore.ieee.org/abstract/document/7045251
QM-449	2014	[714]	13	Yuhana, Umi Laili ; Raharjo, Agus Budi ; Rochimah, Siti	Academic Information System Quality Instrument (AISQI)	ISO 9126, ISO 25010, WBA quality model (WBAQM)			A	Hierarchical	Product	M & P & U	Academic web-application (WBA), Academic information system	https://ieeexplore.ieee.org/abstract/document/7062684
QM-450	2014	[715]	14	Ziemba, Ewa ; Papaj, Tomasz ; Descours, Danuta	Ziemba - Papaj - Descours	ISO 25010			A	Hierarchical	Product	P	e-government portal, website	https://www.researchgate.net/profile/Yousef_Forti2/publication/283545930_The_Adoption_of_e-Government_in_Arab_Countries_The_Case_of_Libya/links/563e0c1e08ae45b5d28c428d/The-Adoption-of-e-Government-in-Arab-Countries-The-Case-of-Libya.pdf#page=268
QM-451	2014	[716]	3	Alrawashdeh, Thamer A.; Muhairat, Mohammad I.; Alqatawneh, Sokyna M	ERP Systems Quality Model (ERPSQM)	ISO 9126			A	Hierarchical	Product	P	Enterprise Resource Planning (ERP) systems	https://ieeexplore.ieee.org/abstract/document/6822174
QM-452	2014	[70]	2	Jeong, Hwa-Young; Park, Jong Hyuk; Jeong, Young-Sik	Jeong - Park - Jeong				A	Hierarchical	Product	M & P & U	secure embedded system, sensor network	https://journals.sagepub.com/doi/full/10.1155/2014/505242
QM-453	2014	[717]	7	Malik, M Usman; Nasir, Haseeb; Javed, Ali	Malik - Nasir - Javed Quality Model for Agile Application Development				D	Hierarchical	Product	M & P & U	Agile software development	https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.428.9146&rep=rep1&type=pdf
QM-454	2014	[718]	10	Baqais, Abdulrahman Ahmed Bobakr; Alshayeb, Mohammad; Baig, Zubair A	Baqais - Alshayeb - Baig				Pr	Neural Network	Product	M	software, object-oriented software, android	https://ro.ecu.edu.au/ecuworkspost2013/867/
QM-455	2015	[333]	10	Chawla, Mandeep K.; Chhabra, Indu	Software Quality Model for Maintainability Analysis (SQMMA)	ISO 9126, ISO 25010			A	Hierarchical	Product	M	software, Java based open source software	https://dl.acm.org/doi/abs/10.1145/2835043.2835062
QM-456	2015	[719]	17	Bezerra, Carla IM.; Andrade, Rossana MC.; Monteiro, José Maria S	CatalOg of measures for Feature model quality Evaluation (COFEE)	ISO 25010			A	Hierarchical	Product	M & P	software product line, feature model, mobile applications	https://link.springer.com/chapter/10.1007/978-3-319-14130-5_20

QM-457	2015	[720]	64	Sudhaman, Parthasarathy; Thangavel, Chandrakumar	Sudhaman - Thangavel		A	Data Envelopment Analysis	Project & Product	M	open source software, open source community process, ERP systems	https://www.sciencedirect.com/science/article/pii/S0263786314001689
QM-458	2015	[721]	10	Sohn, H-J.; Lee, M-G. Seong, B-M. Kim, J-B	Sohn - Lee - Seong - Kim	ISO 25010	A	Hierarchical	Product	M & P	open source software, open source community process, HTML5-based framework	https://scholarworks.bwise.kr/ssu/handle/2018.sw.ssu/9706
QM-459	2015	[722]	15	Alves, João Marcus; Wangenheim, C.; Lacerda, T.; Savaris, Alexandre; Wangenheim, A	question-nAire for Evaluation of QUALity in TELemedicine systems (AdEQUATE)	ISO 25010	A	Hierarchical	Product	U	Telemedicine systems	https://www.researchgate.net/profile/Alido_Von_Wangenheim2/publication/287596231_AdEQUATE_Software_Quality_Evaluation_Model_v10/links/5677c48e08ae125516ee3ace/AdEQUATE-Software-Quality-Evaluation-Model-v10.pdf
QM-460	2015	[723]	8	Ladányi, Gergely; Tóth, Zoltán; Ferenc, Rudolf; Keresztesi, Tibor	Ladányi - Tóth - Ferenc - Keresztesi	ISO 25010	A	Hierarchical	Product	M	IBM mainframe, RPG programming language	https://ieeexplore.ieee.org/abstract/document/7081819
QM-461	2015	[724]	14	Rochimah, Siti; Rahmani, Hanifa I.; Yuhana, Umi Laili	Rochimah - Rahmani - Yuhana	ISO 9126	A	Hierarchical	Product	U	Academic information system, administration module	https://ieeexplore.ieee.org/abstract/document/7220007
QM-462	2015	[725]	34	Suwawi, Dawam Dwi Jatmiko; Darwiyanto, Eko; Rochmani, Martiana	Suwawi - Darwiyanto - Rochmani	ISO 9126	A	Hierarchical	Product	U	academic website, web-portal, e-learning	https://ieeexplore.ieee.org/abstract/document/7231426
QM-463	2015	[71]	28	Bianchi, Thiago Santos; Soares; Felizardo, Katia Romero	Bianchi - Santos - Felizardo	ISO 25010	D	Hierarchical	Product	M & P	embedded system, system of system	https://ieeexplore.ieee.org/abstract/document/7179220
QM-464	2015	[726]	168	Calderón, Alejandro; Ruiz, Mercedes	Calderón - Ruiz		D	Hierarchical	Product	U	Video games, serious games, training system	https://www.sciencedirect.com/science/article/pii/S0360131515300166
QM-465	2015	[727]	15	Zhang, Wei; Huang, LiGuo; Ng, Vincent; Ge, Jidong	SMP Learner		Pr	Machine Learning (ML)	Product	M	software, code change history	https://link.springer.com/article/10.1007/s10515-014-0161-3
QM-466	2015	[728]	5	Jindal, Rajni; Malhotra, Ruchika; Jain, Abha	Jindal - Malhotra - Jain		Pr	Neural Network	Product	M	software, 'browser' application package of android operating system	https://ieeexplore.ieee.org/abstract/document/7359258
QM-467	2016	[336]	12	Kabir, Md Alamgir; Rehman, Muaan Ur; Majumdar, Shariful Islam	Kabir - Rehman - Majumdar	McCall, Boehm78, Shackel, FURPS, Nielsen, Software Usability Measurement Inventory (SUMI) quality model, ISO 9126, Quality in Use Integrated Measurement (QUIM) model, Software Engineering Methodology (SEM) quality model, Preece - Benyon - Davies	D	Hierarchical	Product	U	Point of Sale (POS) systems	https://ieeexplore.ieee.org/abstract/document/7883188

Annexes

											- Keller - Rogers, Alonso-Ríos - Vázquez-García - Mosqueira-Rey - Moret-Bonillo	
QM-468	2016	[337]	3	Di Ruscio, Davide ; Kolovos, Dimitrios S.; Korkontzelos, Yannis; Matragkas, Nicholas; Vinju, Jurgen	OSSMETER Quality model		A	Hierarchical	Product	M	Open Source Software (OSS)	https://ieeexplore.ieee.org/abstract/document/7814523/
QM-469	2016	[338]	4	Qian, Zhenzheng; Wan, Chengcheng; Chen, Yuting	QUIndicator quality model		A	Classification	Product	U	Free / Libre and Open Source Software (FLOSS)	https://ieeexplore.ieee.org/abstract/document/7515956
QM-470	2016	[56]	45	Garcés, Lina ; Ampatzoglou, Apostolos ; Avgeriou, Paris ; Nakagawa Yumi	Garcés - Ampatzoglou - Avgeriou - Nakagawa Ambient Assisted Living (AAL) quality model	ISO 25010	A	Hierarchical	Product	P & U	Ambient Assisted Living (AAL), embedded system and software	https://www.sciencedirect.com/science/article/abs/pii/S0950584916302932
QM-471	2016	[729]	4	Sharma, Chahat; Dubey, Sanjay Kumar	Sharma - Dubey	ISO 25010	A	Fuzzy logic	Product	P	object-oriented system	https://link.springer.com/chapter/10.1007/978-981-10-0451-3_9
QM-472	2016	[730]	9	Andrian, Rian; Hendradjaya, Bayu; Sunindyo Wikan D	Andrian - Hendradjaya - Sunindyo E-Government G2B quality model	ISO 9126	A	Hierarchical	Product	P	Enterprise Resource Planning (ERP) systems, Government to business (G2B)	https://ieeexplore.ieee.org/abstract/document/7571931
QM-473	2016	[731]	6	Marir, Toufik; Mokhati, Farid; Bouchlaghem-Seridi, Hassina; Acid, Youghourta; Bouzid, Maroua	QM4MAS	ISO 9126	A	Hierarchical	Product	P	Multi-agent systems (MASs), complex and distributed applications	https://www.inderscienceonline.com/doi/abs/10.1504/IJCAT.2016.080485
QM-474	2016	[732]	130	Sarrab, Mohamed; Elbasir, Mahmoud; Alnaeli, Saleh	Sarrab - Elbasir - Alnaeli		A	Hierarchical	Service	P & U	mobile learning services, mobile learning development	https://www.sciencedirect.com/science/article/pii/S0747563215301345
QM-475	2016	[733]	14	Jain, Ashu; Tarwani, Sandhya; Chug, Anuradha	Jain - Tarwani - Chug		Pr	Genetic Algorithm (GA)	Product	M	object oriented software	https://ieeexplore.ieee.org/abstract/document/7509314
QM-476	2016	[734]	1	Forouzani, Sepehr; Chiam, Yin Kia; Forouzani, Soroush	Forouzani - Chiam - Forouzani	ISO 25010, QMOOD	A	Hierarchical	Product	M & P	object oriented software	https://dl.acm.org/doi/abs/10.1145/3033288.3033316
QM-477	2017	[735]	3	Kumar, Nimish; Dadhich, Reena; Shastri, Aditya	Multi-Attribute Quality Model (MAQM)	ISO 9126	A	Hierarchical	Product	M & P & U	web-based application (WBA), object oriented	https://link.springer.com/article/10.1007/s13198-016-0512-5
QM-478	2017	[736]	2	Wibowo, Ripto Mukti; Erna, P Adhitya; Hidayah, Indriana	Wibowo - Erna - Hidayah	ISO 9126	A	Hierarchical	Product	U	Decision Support System (DSS) for recommendation of outstanding marketing officer, Information systems	https://ieeexplore.ieee.org/abstract/document/8304181

QM-479	2017 [737]	8	Tabassum, Atika; Bhatti, Dr S Nazir; Asghar, A Rida; Manzoor, Iqra; Alam, Imtiaz	Tabassum - Bhatti - Asghar - Manzoor - Alam Quality model for XP process & product	ISO 9126, ISO 25010	D	Hierarchical	Product & Process	M & P & U	Agile software development, extreme programming	https://pdfs.semanticscholar.org/df9c/7cecf5b9ce9df823a21704e658a21ddabd51.pdf
QM-480	2017 [738]	32	Azainil, Ramadiani; Haryaka, Usfandi; Agus, Fahrul; Kridalaksana, Awang Harsa	User satisfaction model		A	Structural Equation Modeling (SEM)	Product	U	E-learning system, online learning	https://www.sciencedirect.com/science/article/pii/S1877050917321208
QM-481	2017 [739]	9	Anwer, Sajid; Adbellatif, Ahmad and Alshayeb, Mohammad; Anjum, Muhammad Shakeel	Anwer - Adbellatif - Alshayeb - Anjum		Pr	Logistic regression	Product	M	object oriented system and software, open-source software	https://ieeexplore.ieee.org/abstract/document/7918930
QM-482	2018 [191]	2	Gordieiev, Oleksandr and Kharchenko, Vyacheslav	Gordieiev - Kharchenko	ISO 25010	D	Hierarchical	Product	P	IT, service-based systems (internet of things, green IT, virtual reality, augmented reality, artificial intelligence, cloud computing, blockchain, web)	https://ieeexplore.ieee.org/abstract/document/8409162/
QM-483	2018 [6]	1	Tamrabet, Zouheyr and Marir, Toufik; Mokhati, Farid	Tamrabet - Marir - Mokhati		D	Hierarchical	Product	M & P	embedded system	https://www.igi-global.com/article/a-survey-on-quality-attributes-and-quality-models-for-embedded-software/204480
QM-484	2018 [349]	10	Russo, Daniel ; Ciancarini, Paolo; Falasconi, Tommaso; Tomasi, Massimo	Software Quality, Architecture, Process (SQuAP)	ISO 25010	D	Meta-model	Product & Process	M & P & U	Information systems, financial sector	https://dl.acm.org/doi/abs/10.1145/3230713
QM-485	2018 [350]	1	Wahdiniwaty, Rahma; Setiawan, Eko Budi; Wahab, Deden A	Wahdiniwaty - Setiawan - Wahab	McCall, Boehm78, Dromey, FURPS, ISO 9126	A	Hierarchical	Product	U	website, web-portal, e-commerce	https://ieeexplore.ieee.org/abstract/document/8696074
QM-486	2018 [740]	5	Suradi, Nur Razia Mohd; Kahar, Saliyah; Jamaluddin, Nor Azliana Akmal	Suradi - Kahar - Jamaluddin	McCall, Boehm78, Dromey, FURPS, ISO 9126	D	Hierarchical	Product	U	Higher Education Institution (HEI), Academic application, e-learning, e-course, web-portal	https://journal.utem.edu.my/index.php/jtec/article/view/4440
QM-487	2018 [351]	1	Gatica, Diego; Ponce, Francisco; Noël, René; Astudillo, Hernán	Gatica - Ponce - Noël - Astudillo		D	Hierarchical	Product	P	systems of systems, architecture, design	https://ieeexplore.ieee.org/abstract/document/8705229
QM-488	2018 [352]	20	Abdellatif, Abdelbaset Jamal ; McCollum, Barry ; McMullan, Paul	Abdellatif - McCollum - McMullan		A	Hierarchical	Product	U	serious game, education, programmed serious games, video-games, Robocode	https://ieeexplore.ieee.org/abstract/document/8340460

Annexes

QM-489	2018	[741]	37	Condori-Fernandez, Nelly; Lago, Patricia	Condori-Fernandez - Lago	ISO 25010	D	Hierarchical	Product	M & P	Software architecture, sustainable software	https://www.sciencedirect.com/science/article/pii/S0164121217302984
QM-490	2019	[77]	1	Juneja, Sapna; Juneja, Abhinav; Anand, Rohit	Juneja - Juneja Park - Kim - Shin - Baik		Pr	Stochastic	Product	M	embedded software, and systems that are mission-critical including aircrafts, automobile, nuclear power plants and various robotic medical applications	https://ieeexplore.ieee.org/abstract/document/8776814
QM-491	2019	[358]	3	Condori-Fernandez, Nelly; Lago, Patricia	Sustainability-quality model	Condori-Fernandez - Lago	D	Hierarchical	Product	M & P	software-intensive systems	https://ieeexplore.ieee.org/abstract/document/8877084
QM-492	2019	[205]	6	Gezici, Bahar; Tarhan, Ayca; Chouseinoglou, Oumout	Gezici - Tarhan - Chouseinoglou	ISO 25010	A	Hierarchical	Product	M & P & U	mobile applications, mobile software, open-source software (OSS), open-source market, object-oriented software	https://www.sciencedirect.com/science/article/pii/S0950584918301290

Annex 7. Project scorecard description

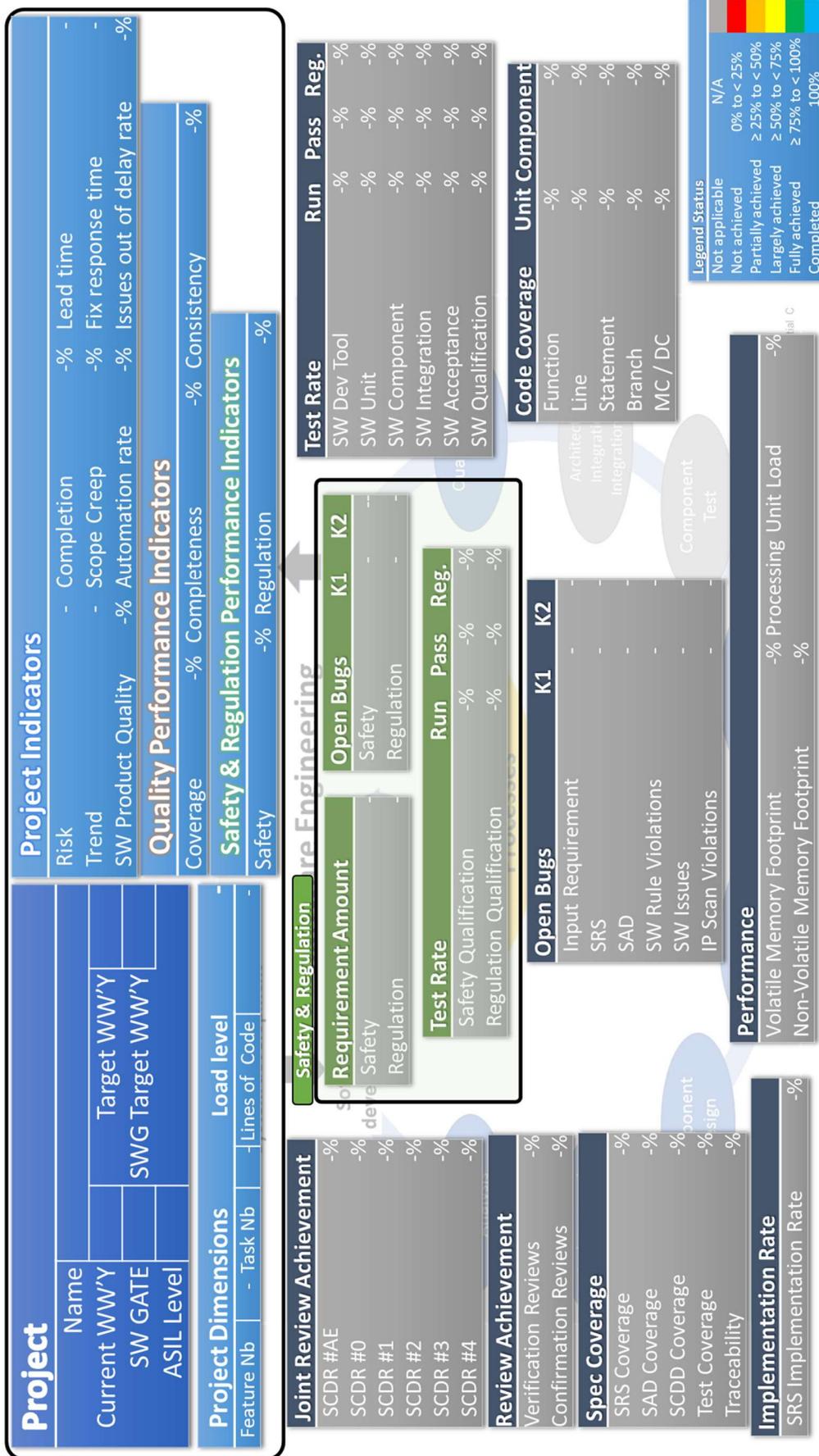


TABLE 49 - DEFINITIONS OF THE DIFFERENT ELEMENTS OF THE PROJECT SCORECARD

Scorecard Category	Name	Definition
Project	Name	Project name
	Current WW'Y	Current workweek and year under the form "ww'yy" (where "ww" and "yy" are respectively week and year with 2 digits). e.g.: week of 1st January 2019 is 01'19
	Target WW'Y	Current sprint or PI target workweek and year under the form "ww'yy" (where "ww" and "yy" are respectively week and year with 2 digits). e.g.: week of 1st January 2019 is 01'19
	SW GATE	Name of current targeted Software Gate. Possible values: SW_G1, SW_G2, SW_G2, SW_G3, SW_G4, SW_G5, SW_G6, SW_G7, SW_G8, SW_G9, SW_G10, SW_G11
	SWG Target WW'Y	Current SW Gate target workweek and year under the form "ww'yy" (where "ww" and "yy" are respectively week and year with 2 digits). e.g.: week of 1st January 2019 is 01'19
Project Dimensions	Load level	Load level of project: GM (Grand Mother), M (Mother), B (Brother) & C (Child)
	Feature Nb	Total number of features targeted by the project (including the ones already completed within the project scope)
	Task Nb	Total number of tasks, completed or not, within the project scope
	Lines of Code	Total number of lines of code done within the project scope
Project Indicators	Risk	Current global project risk level done by risk manager, or project manager. Possible values: low , med , high
	Trend	Current global project trend compares to previous week and done by project manager. Possible values: ↗, →, ↘
	Completion	Percentage of completed sprint or PI scope targeted by current target ; may be linked to current project management approach
	Scope Creep	Percentage of changes, continuous or uncontrolled growth in a project's scope (different from feature creep), at any point after the project begins. This metric can be achieved using variation metrics
	Lead time	Mean time of entire process crossing (e.g., from to requirement specification to code in use); may depend on project development methodology
	Fix response time	Mean time to fix defect (cycle time)
	Issues out of delay rate	Percentage of issues that are not addressed for more than a certain fixed delay: 15 days
	SW Product Quality	Consolidated percentage of SW product quality . This should be computed (weighted mean if weights are defined) based on proper quality model and all measured quality characteristics/sub-characteristics
	Automation rate	Cumulated percentage of spec (SRS, SAD) and code covered by automated tests
	Coverage	(1) At ECU level, Coverage reflects if a strategy has been decided or not (value=100% or 0%).

Annexes

Quality Performance Indicators		(2) At higher levels, Coverage reflects the ratio of underlying ECUs for which a strategy has been decided. (3) At lower levels, Coverage reflects the ratio of applicable and planned Gate criteria for current SW Gate
	Completeness	Ratio of evaluated outcomes (green, orange, red) or criteria with regards to expected outcomes or criteria
	Consistency	Ratio of non-failing outcomes (green, orange; i.e., good quality level, action in place or derogation of evaluated criteria) with regards to expected outcomes or evaluated criteria
Safety & Regulation Performance Indicators	Safety	Percentage of Safety related achieved outcomes
	Regulation	Percentage of Regulation related achieved outcomes
Spec Coverage	SRS Coverage	Percentage of STRComp covered by all Software Requirement Specifications (SRS)
	SAD Coverage	Percentage of SRS covered by all Software Architecture Designs (SAD)
	SCDD Coverage	Percentage of SAD covered by all Software Component Detailed Design (SCDD) (Software Unit)
	Test Coverage	Percentage of aggregated (i.e., can be either weighted or unweighted mean) "Percentage of Software Unit covered by Software Unit tests", "Percentage of SAD covered by Integration tests" and "Percentage of SRS covered by Qualification tests"
	Traceability	Percentage of End to End (SRS to qualification test) traceability
Implementation Rate	SRS Implementation Rate	Percentage of SRS implemented (Model & Code). Once implementation is completed, SCDR#2 can be done.
Review Achievement	Verification Reviews	Percentage of Completed Verification reviews (i.e., evaluated with review leader decision) over Planned Verification Reviews
	Confirmation Reviews	Percentage of Completed Confirmation reviews (i.e., evaluated with review leader decision) over Planned Confirmation Reviews
Joint Review Achievement	SCDR #AE	Percentage of SCDR (i.e., System Control Design Review) #AE reviews completed (i.e., evaluated with chairman decision) considering that SCDR number is directly linked to every approved software change request or package of change request according ECU development process
	SCDR #0	Percentage of SCDR #0 reviews completed (i.e., evaluated with chairman decision) considering that SCDR number is directly linked to every approved software change request or package of change request according ECU development process
	SCDR #1	Percentage of SCDR #1 reviews completed (i.e., evaluated with chairman decision) considering that SCDR number is directly linked to every approved software change request or package of change request according ECU development process
	SCDR #2	Percentage of SCDR #2 reviews completed (i.e., evaluated with chairman decision) considering that SCDR number is directly linked to every approved software change request or package of change request according ECU development process

	SCDR #3	Percentage of SCDR #3 reviews completed (i.e., evaluated with chairman decision) considering that SCDR number is directly linked to every approved software change request or package of change request according ECU development process
	SCDR #4	Percentage of SCDR #4 reviews completed (i.e., evaluated with chairman decision) considering that SCDR number is directly linked to every approved software change request or package of change request according ECU development process
Open Bugs ²³ [K1/K2] ²⁴	Input Requirement	Number of open Input requirement K1 bugs and K2 bugs. These bugs are directly linked to requirements input (i.e., STRComp) coming from system
	SRS	Number of open Software Requirements K1 bugs and K2 bugs. These bugs are directly linked to SRS
	SAD	Number of open Software Architecture K1 bugs and K2 bugs. These bugs are directly linked to SAD
	SW Rule Violations	Number of open Software Rule K1 bugs and K2 bugs. These bugs are directly linked to Software Rule violations detected when applying and executing MXAM on model or Static Analysis on source code (e.g., with QAC or CodeSonar tools).
	SW Issues	Number of open Software K1 bugs and Software K2 bugs not listed in the other categories (i.e., Software Input Requirement, Software Architecture, Software Rule Violations and Legal violations)
	IP Scan Violations	Number of open IP Scan K1 bugs and K2 bugs. These bugs are directly linked to IP Scan violations detected when applying and executing IP Plan.
Performance	Processing Unit load	Percentage of current Processing Unit (i.e., CPU, GPU, DSP, NPU) Load when running processing unit load test scenario. Assessment must be done accordingly to project target.
	Volatile Memory Footprint	Maximum volatile memory (i.e., RAM, Cache) footprint of the software with respect to available volatile memory. Assessment must be done accordingly to project target.
	Non-Volatile Memory Footprint	Maximum non-volatile memory (i.e., ROM, Flash) footprint of the software with respect to available non-volatile memory. Assessment must be done accordingly to project target.
Test Rate [Run/Pass/Reg.]	SW Dev Tool	Run rate: percentage of executed Software Development Tool tests over planned ²⁵ Software Development Tool tests, Pass rate: percentage of passed Software Development Tool tests over planned ²⁵ Software Development Tool tests,
	SW Unit	Run rate: percentage of executed Software Unit tests over planned ²⁵ Software Unit tests, Pass rate: percentage of passed Software Unit tests over planned ²⁵ Software Unit tests, Regression rate: percentage of currently failing Software Unit tests, which previously passed, over Software Unit tests that passed,
	SW Component	Run rate: percentage of executed Software Component tests over planned ²⁵ Software Component tests, Pass rate: percentage of passed Software Component tests over planned ²⁵ Software Component tests,

²³ Open bug.: other than closed and verified bug

²⁴ K1 and K2 represents bug criticality

²⁵ Planned: with regards to current Software Gate scope.

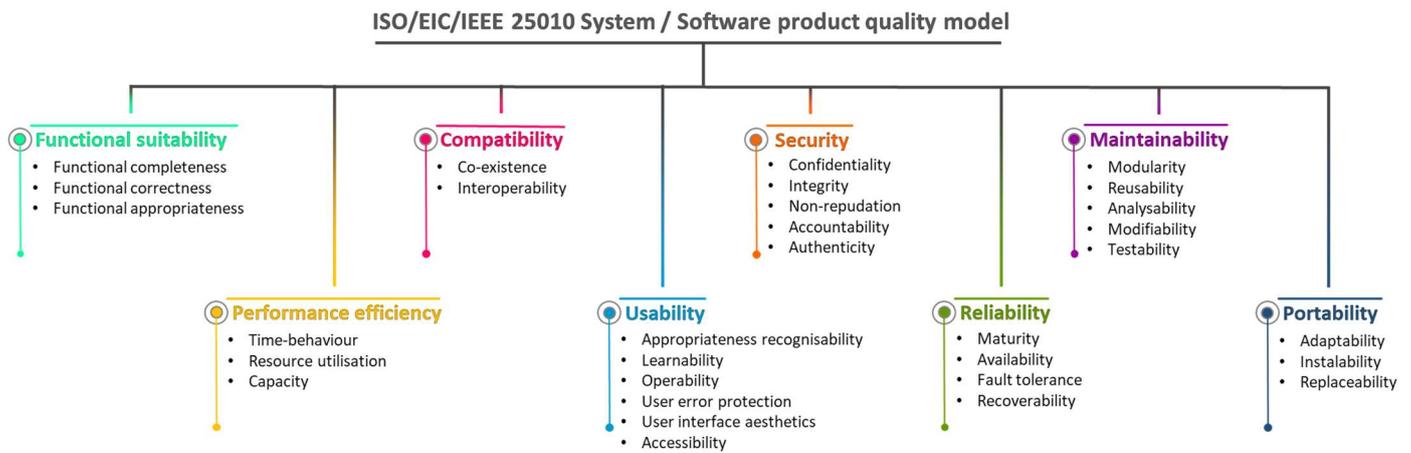
		Regression rate: percentage of currently failing Software Component tests, which previously passed, over Software Component tests that passed,
	SW Integration	Run rate: percentage of executed Software Integration tests over planned ²⁵ Software Integration tests, Pass rate: percentage of passed Software Integration tests over planned ²⁵ Software Integration tests, Regression rate: percentage of currently failing Software Integration tests, which previously passed, over Software Integration tests that passed,
	SW Acceptance	Run rate: percentage of executed Software Acceptance tests over planned ²⁵ Software Acceptance tests, Pass rate: percentage of passed Software Acceptance tests over planned ²⁵ Software Acceptance tests,
	SW Qualification	Run rate: percentage of executed Software Qualification tests over planned ²⁵ Software Qualification tests, Pass rate: percentage of passed Software Qualification tests over planned ²⁵ Software Qualification tests, Regression rate: percentage of currently failing Software Qualification tests, which previously passed, over Software Qualification tests that passed,
Code Coverage	Function	Unit: Percentage of functions covered by Software unit test executions. Metrics which can be obtained thanks to gcov / lcov tool, for example Component: Percentage of functions covered by Software component test executions. Metrics which can be obtained thanks to gcov / lcov tool, for example
	Line ²⁶	Unit: Percentage of code lines covered by Software unit test executions. Metrics which can be obtained thanks to gcov / lcov tool, for example Component: Percentage of code lines covered by Software component test executions. Metrics which can be obtained thanks to gcov / lcov tool, for example
	Statement Highly recommended for ASIL A, B	Unit: Percentage of statements covered by Software unit test executions. Component: Percentage of statements covered by Software component test executions.
	Branch Highly recommended for ASIL B, C, D	Unit: Percentage of branches covered by Software unit test executions. Component: Percentage of branches covered by Software component test executions.
	MC/DC (Modified Condition / Decision Changed) Highly recommended for ASIL D	Unit: Percentage of MC/DC covered by Software unit test executions. Component: Percentage of MC/DC covered by Software component test executions.
Requirement Amount	Safety	Number of Safety requirement with “Software Impact”
	Regulation	Number of Regulation requirement with “Software Impact”
Open Bugs ²³	Safety	Number of open Safety related K1 bugs and K2 bugs

²⁶ A line may contain zero (i.e., comment), one or more statements.

[K1/K2] ²⁴	Regulation	Number of open Regulation related K1 bugs and K2 bugs. Doesn't include Safety related bugs.
Test Rate [Run/Pass/Reg.]	Safety	<p>Run rate: percentage of executed Safety related software Qualification tests over planned²⁵ Safety related software Qualification tests,</p> <p>Pass rate: percentage of passed Safety related software Qualification tests over planned²⁵ Safety related Software Qualification tests,</p> <p>Regression rate: percentage of currently failing Safety related software Qualification tests, which previously passed, over Safety related software Qualification tests that passed</p>
	Regulation	<p>Run rate: percentage of executed Regulation related software Qualification tests over planned²⁵ Regulation related software Qualification tests,</p> <p>Pass rate: percentage of passed Regulation related software Qualification tests over planned²⁵ Regulation related software Qualification tests,</p> <p>Regression rate: percentage of currently failing Regulation related software Qualification tests, which previously passed, over Regulation related software Qualification tests that passed</p>

Annex 8. ISO/IEC/IEEE 25010 quality models [23]

- Definitions of the system / Software product quality model



o Quality characteristic definitions

Quality Characteristic	Definition
Functional Suitability	Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.
Performance Efficiency	Performance relative to the amount of resources used under stated conditions.
Compatibility	Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.
Usability	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.
Reliability	Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.
Security	Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.
Maintainability	Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.
Portability	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

o Quality sub-characteristic definitions

▪ *Functional Suitability*

Quality Sub-Characteristic	Definition
Functional Completeness	Degree to which the set of functions covers all the specified tasks and user objectives.
Functional Correctness	Degree to which a product or system provides the correct results with the needed degree of precision.

Quality Sub-Characteristic	Definition
----------------------------	------------

Functional Appropriateness	Degree to which the functions facilitate the accomplishment of specified tasks and objectives.
-----------------------------------	--

- *Performance Efficiency*

Quality Sub-Characteristic	Definition
----------------------------	------------

Time Behaviour	Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
-----------------------	--

Resource Utilization	Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
-----------------------------	---

Capacity	Degree to which the maximum limits of a product or system parameter meet requirements.
-----------------	--

- *Compatibility*

Quality Sub-Characteristic	Definition
----------------------------	------------

Co-existence	Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
---------------------	---

Interoperability	Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
-------------------------	---

- *Usability*

Quality Sub-Characteristic	Definition
----------------------------	------------

Appropriateness Recognizability	Degree to which users can recognize whether a product or system is appropriate for their needs.
--	---

Learnability	Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
---------------------	--

Operability	Degree to which a product or system has attributes that make it easy to operate and control.
--------------------	--

User Error Protection	Degree to which a system protects users against making errors.
------------------------------	--

User Interface Aesthetics	Degree to which a user interface enables pleasing and satisfying interaction for the user.
----------------------------------	--

Accessibility	Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.
----------------------	--

- *Reliability*

Quality Sub-Characteristic	Definition
----------------------------	------------

Maturity	Degree to which a system, product or component meets needs for reliability under normal operation.
-----------------	--

Quality Sub-Characteristic	Definition
Availability	Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
Fault Tolerance	Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
Recoverability	Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

- **Security**

Quality Sub-Characteristic	Definition
Confidentiality	Degree to which a product or system ensures that data are accessible only to those authorized to have access.
Integrity	Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.
Non-Repudiation	Degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.
Accountability	Degree to which the actions of an entity can be traced uniquely to the entity.
Authenticity	Degree to which the identity of a subject or resource can be proved to be the one claimed.

- **Maintainability**

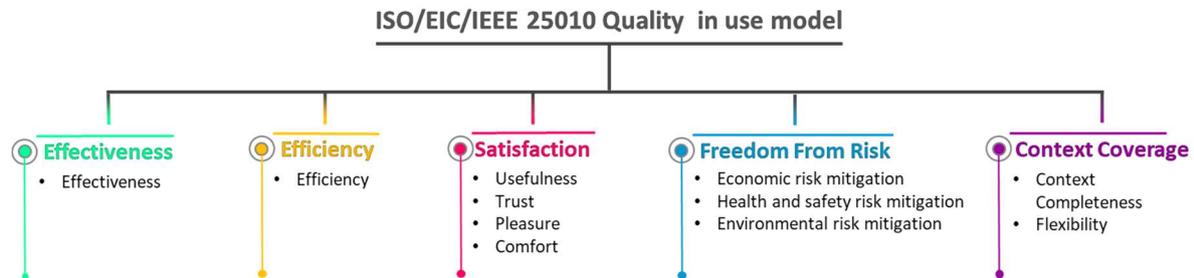
Quality Sub-Characteristic	Definition
Modularity	Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
Reusability	Degree to which an asset can be used in more than one system, or in building other assets.
Analysability	Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
Modifiability	Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
Testability	Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

- **Portability**

Quality Sub-Characteristic	Definition
Adaptability	Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
Installability	Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.

Quality Sub-Characteristic	Definition
Replaceability	Degree to which a product can replace another specified software product for the same purpose in the same environment.

- Definitions of the quality in use model



o Quality characteristic definitions

Quality Characteristic	Definition
Effectiveness	Accuracy and completeness with which users achieve specified goals.
Efficiency	Resources expended in relation to the accuracy and completeness with which users achieve goals.
Satisfaction	Degree to which user needs are satisfied when a product or system is used in a specified context of use.
Freedom from Risk	Degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment.
Context Coverage	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.

o Quality sub-characteristic definitions

- **Effectiveness**
<no further sub-characteristic>
- **Efficiency**
<no further sub-characteristic>
- **Satisfaction**

Quality Sub-Characteristic	Definition
Usefulness	Degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use.
Trust	Degree to which a user or other stakeholder has confidence that a product or system will behave as intended.
Pleasure	Degree to which a user obtains pleasure from fulfilling their personal needs.
Comfort	Degree to which the user is satisfied with physical comfort.

- *Freedom from Risk*

Quality Sub-Characteristic	Definition
Economic Risk Mitigation	Degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use.
Health and Safety Risk Mitigation	Degree to which a product or system mitigates the potential risk to people in the intended contexts of use.
Environmental Risk Mitigation	Degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment.

- *Context Coverage*

Quality Sub-Characteristic	Definition
Context Completeness	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use.
Flexibility	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.

Annex 9. Survey used against our real-world use case

Quality Characteristics to be considered for projects within HHN scope

The aim of this survey is to be able generate right quality model which qualifies SW product quality to a variety of HHN projects. The quality model is aligned with ISO/IEC 25010:2011, the current reference standard and highlighted by Automotive SPICE.
In the next questions we ask for your own judgment and vision.

Hi Yann, when you submit this form, the owner will be able to see your name and email address.

* Required

1. What is your role? *

- Architect
- IAQP
- PM
- Validation Lead

2. What is the project you are evaluating in this survey? *

- ADAS
- A-IV2
- FOTA
- IVC2

Next

Page 1 of 5

Quality Characteristics to be considered for projects within HHN scope

...

* Required

Software Quality Characteristic Ranking

ISO/IEC 25010:2011 provides 8 software product and 5 software quality in use quality characteristics. In the following questions, we asked you to rank how you consider the importance of each of these quality characteristics, in term of quality impact to your project.

The definitions of these quality characteristics are available in the wiki page:
<https://confluence.dt.renault.com/pages/viewpage.action?pageId=143785172>

3. Please, rank each of the following 8 software product quality characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important
Functional Suitability	<input type="radio"/>				
Performance Efficiency	<input type="radio"/>				
Compatibility	<input type="radio"/>				
Usability	<input type="radio"/>				
Reliability	<input type="radio"/>				
Security	<input type="radio"/>				
Maintainability	<input type="radio"/>				
Portability	<input type="radio"/>				

4. Please, rank each of the following 5 software quality in use characteristics *

	No Important at All	Somewhat Important	Important	Very Important	Extremely Important
Effectiveness	<input type="radio"/>				
Efficiency	<input type="radio"/>				
Satisfaction	<input type="radio"/>				
Freedom From Risk	<input type="radio"/>				
Context Coverage	<input type="radio"/>				

Back

Next

Page 2 of 5

Quality Characteristics to be considered for projects within HHN scope

...

* Required

Software Product Quality Sub-Characteristic Ranking

The 8 software product quality characteristics from ISO/IEC 25010:2011 are refined into 31 quality sub-characteristics. In the following questions, we asked you to rank how you consider the importance of each of these quality sub-characteristics, in term of quality impact to your project.

The definitions of these quality characteristics are available in the wiki page:

<https://confluence.dt.renault.com/pages/viewpage.action?pageId=143785172>

5. Please, rank each of the following 3 functional suitability quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Functional Completeness	<input type="radio"/>					
Functional Correctness	<input type="radio"/>					
Functional appropriateness	<input type="radio"/>					

6. Please, rank each of the following 3 performance efficiency quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Time-behavior	<input type="radio"/>					
Resource Utilization	<input type="radio"/>					
Capacity	<input type="radio"/>					

7. Please, rank each of the following 2 compatibility quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Co-existence	<input type="radio"/>					
Interoperability	<input type="radio"/>					

8. Please, rank each of the following 6 usability quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Appropriateness	<input type="radio"/>					
Recognizability	<input type="radio"/>					
Learnability	<input type="radio"/>					
Operability	<input type="radio"/>					
User Error Protection	<input type="radio"/>					
User Interface Aesthetics	<input type="radio"/>					
Accessibility	<input type="radio"/>					

9. Please, rank each of the following 4 reliability quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Maturity	<input type="radio"/>					
Availability	<input type="radio"/>					
Fault Tolerance	<input type="radio"/>					
Recoverability	<input type="radio"/>					

10. Please, rank each of the following 5 security quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Confidentiality	<input type="radio"/>					
Integrity	<input type="radio"/>					
Non-Repudiation	<input type="radio"/>					
Accountability	<input type="radio"/>					
Authenticity	<input type="radio"/>					

11. Please, rank each of the following 5 maintainability quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Modularity	<input type="radio"/>					
Reusability	<input type="radio"/>					
Analysability	<input type="radio"/>					
Modifiability	<input type="radio"/>					
Testability	<input type="radio"/>					

12. Please, rank each of the following 3 portability quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Adaptability	<input type="radio"/>					
Installability	<input type="radio"/>					
Replaceability	<input type="radio"/>					

Back

Next

Page 3 of 5

Quality Characteristics to be considered for projects within HHN scope



* Required

Software Quality in Use Sub-Characteristic Ranking

The 5 software product quality characteristics from ISO/IEC 25010:2011 are refined into 9 quality sub-characteristics. In the following questions, we asked you to rank how you consider the importance of each of these quality sub-characteristics, in term of quality impact to your project.

The definitions of these quality characteristics are available in the wiki page: <https://confluence.dt.renault.com/pages/viewpage.action?pageId=143785172>

13. Please, rank each of the following 4 satisfaction quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Usefulness	<input type="radio"/>					
Trust	<input type="radio"/>					
Pleasure	<input type="radio"/>					
Comfort	<input type="radio"/>					

14. Please, rank each of the following 3 freedom from risk quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Economic Risk Mitigation	<input type="radio"/>					
Health and Safety Risk Mitigation	<input type="radio"/>					
Environmental Risk Mitigation	<input type="radio"/>					

15. Please, rank each of the following 2 context coverage quality sub-characteristics *

	Not Important at All	Somewhat Important	Important	Very Important	Extremely Important	I don't know
Context Completeness	<input type="radio"/>					
Flexibility	<input type="radio"/>					

Back

Next

Page 4 of 5

Quality Sub-Characteristic Ranking

The previous quality characteristics and sub-characteristics are the ones recommended by the ISO/IEC 25010 standard. With this last question, you can suggest additional quality characteristics very or extremely important to take into account.

16. Please, provide any additional quality characteristic(s) (not listed above) that is(are) very or extremely important for your project.

Back

Submit

Page 5 of 5

This content is created by the owner of the form. The data you submit will be sent to the form owner. Microsoft is not responsible for the privacy or security practices of its customers, including those of this form owner. Never give out your password.

Powered by Microsoft Forms | [Privacy and cookies](#) | [Terms of use](#)

Annex 10. Survey results

Quality Characteristics to be considered for projects within HHN scope

<p>13 Responses</p>	<p>1692:20 Average time to complete</p>	<p>Active Status</p>
--------------------------------	--	---------------------------------

[View results](#)

Open in Excel

1. What is your role?

[More Details](#)

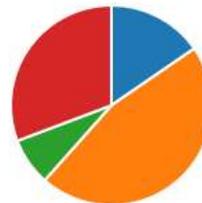
● Architect	0
● IAQP	4
● PM	8
● Validation Lead	1



2. What is the project you are evaluating in this survey?

[More Details](#)

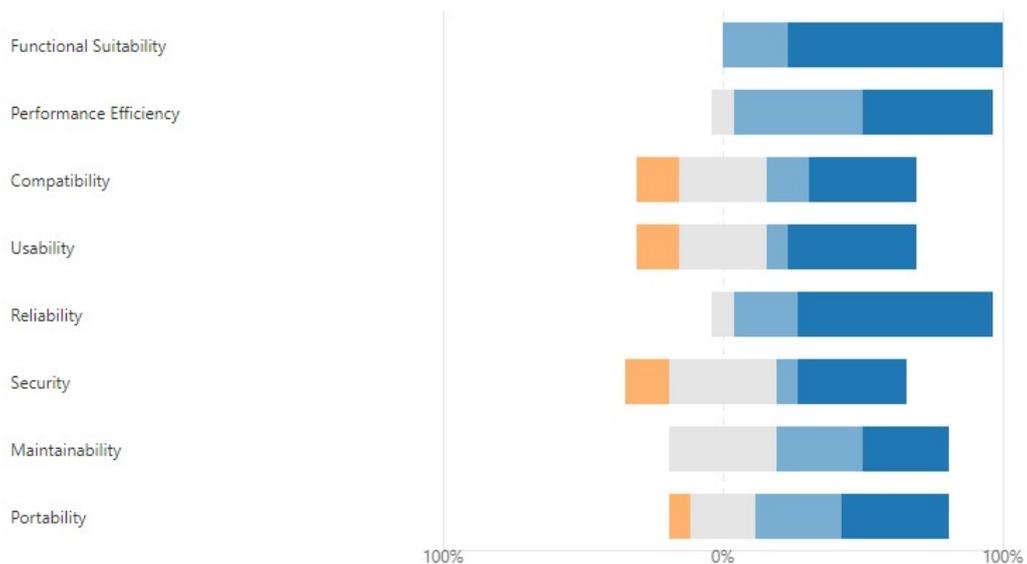
● ADAS	2
● A-IVI2	6
● FOTA	1
● IVC2	4



3. Please, rank each of the following 8 software product quality characteristics

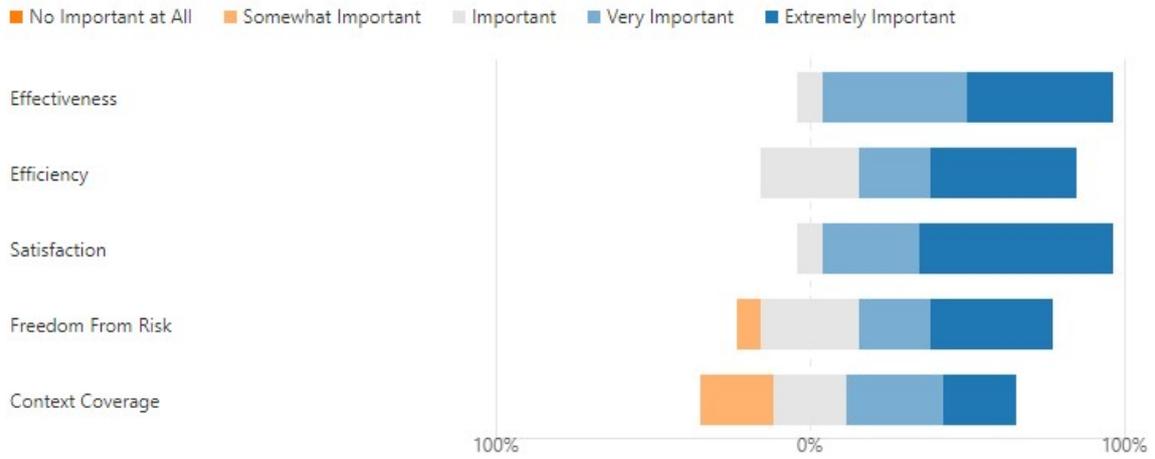
[More Details](#)

■ Not Important at All
 ■ Somewhat Important
 ■ Important
 ■ Very Important
 ■ Extremely Important



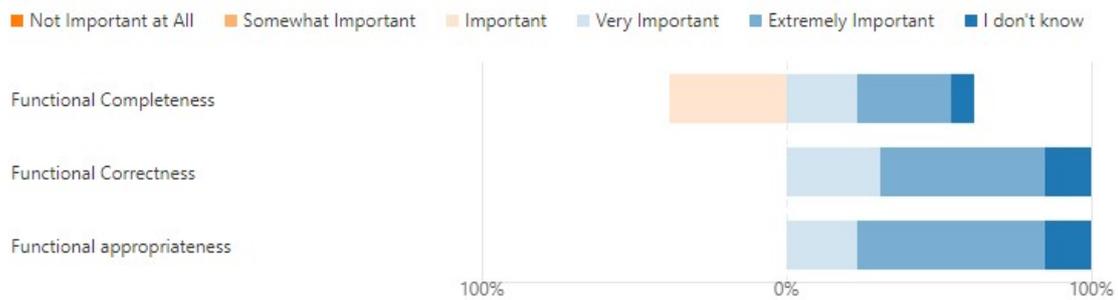
4. Please, rank each of the following 5 software quality in use characteristics

[More Details](#)



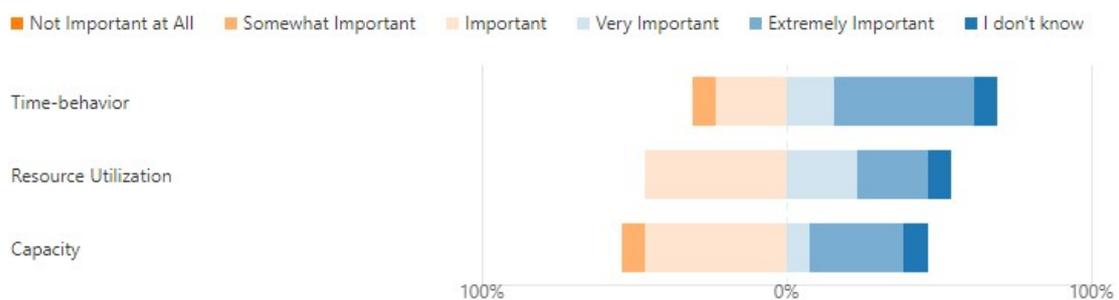
5. Please, rank each of the following 3 functional suitability quality sub-characteristics

[More Details](#)



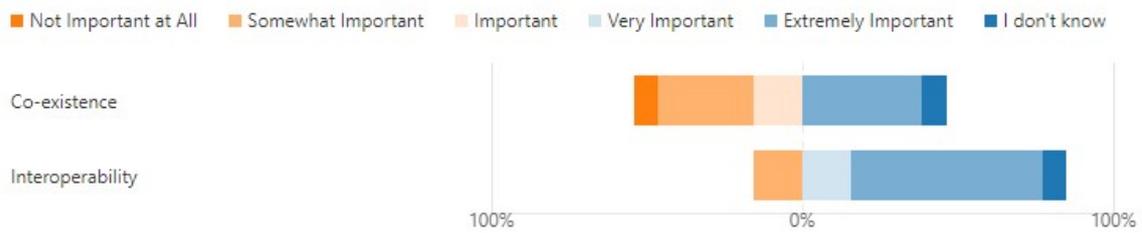
6. Please, rank each of the following 3 performance efficiency quality sub-characteristics

[More Details](#)



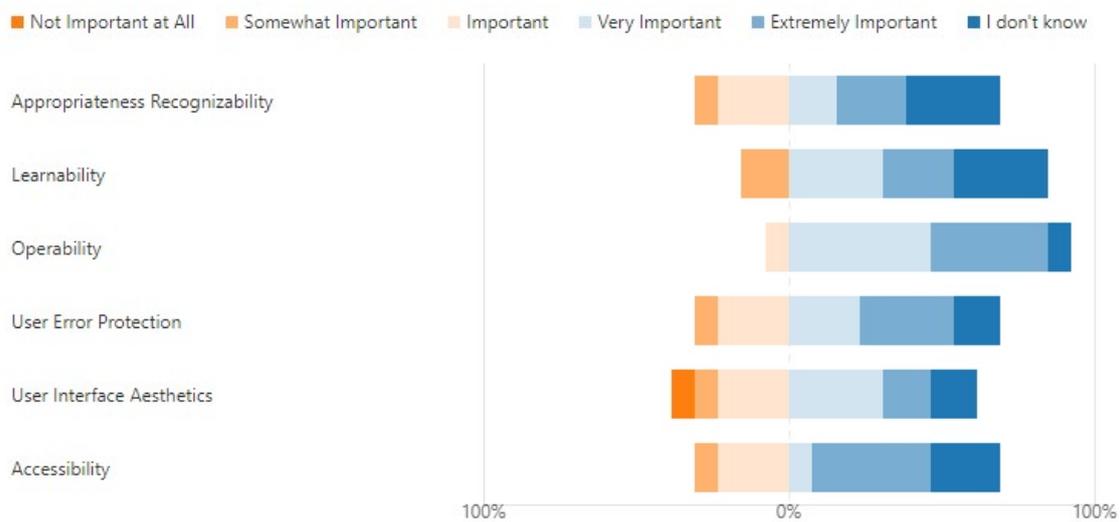
7. Please, rank each of the following 2 compatibility quality sub-characteristics

[More Details](#)



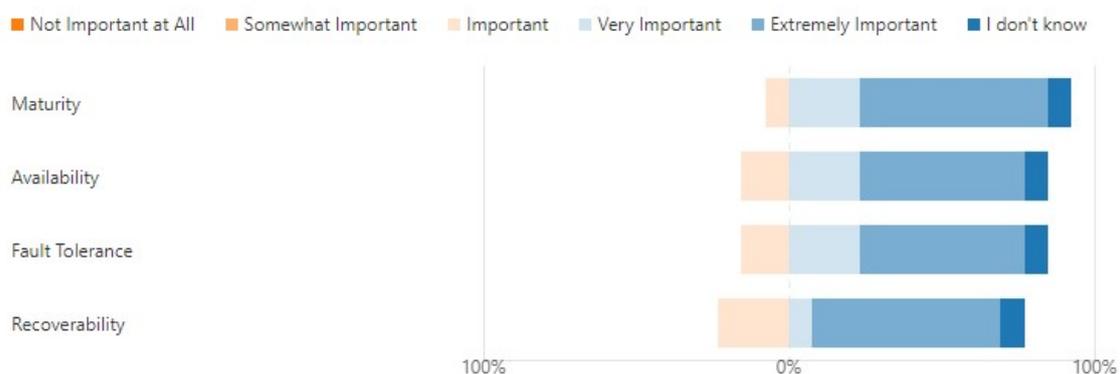
8. Please, rank each of the following 6 usability quality sub-characteristics

[More Details](#)



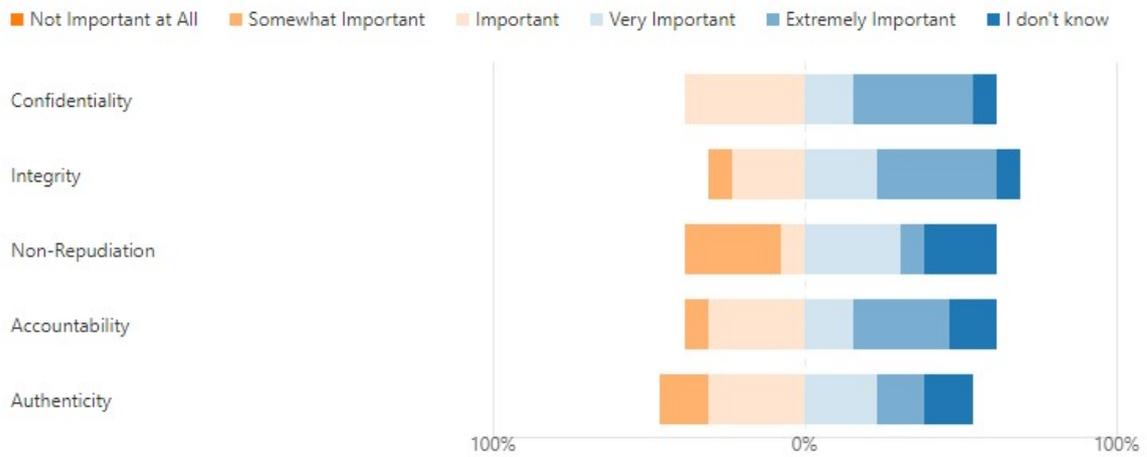
9. Please, rank each of the following 4 reliability quality sub-characteristics

[More Details](#)



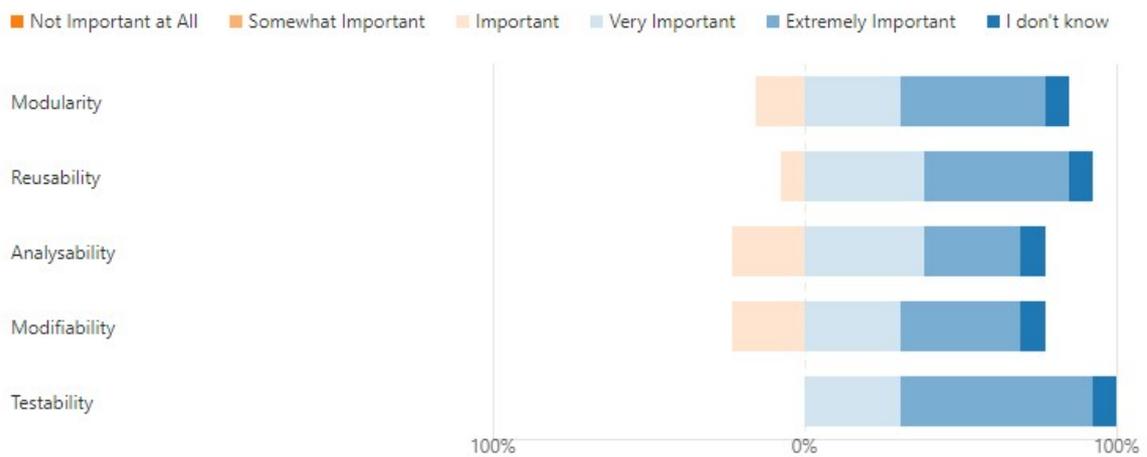
10. Please, rank each of the following 5 security quality sub-characteristics

[More Details](#)



11. Please, rank each of the following 5 maintainability quality sub-characteristics

[More Details](#)



12. Please, rank each of the following 3 portability quality sub-characteristics

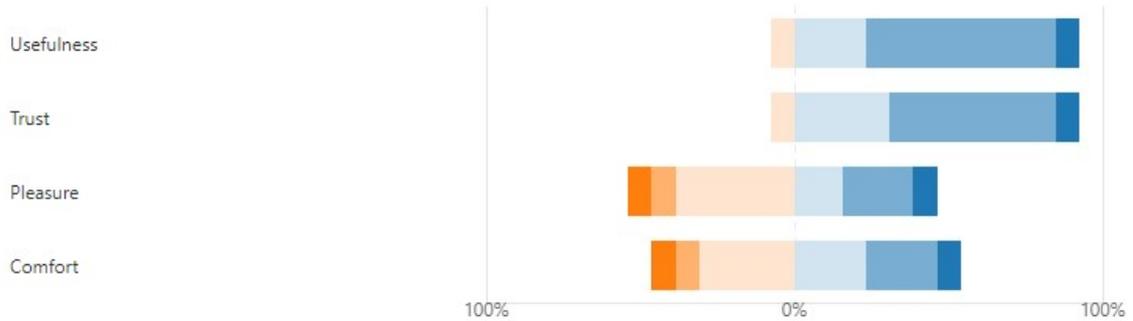
[More Details](#)



13. Please, rank each of the following 4 satisfaction quality sub-characteristics

[More Details](#)

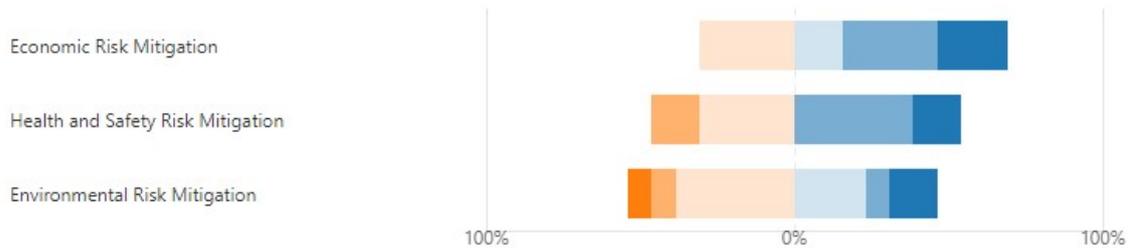
■ Not Important at All
 ■ Somewhat Important
 ■ Important
 ■ Very Important
 ■ Extremely Important
 ■ I don't know



14. Please, rank each of the following 3 freedom from risk quality sub-characteristics

[More Details](#)

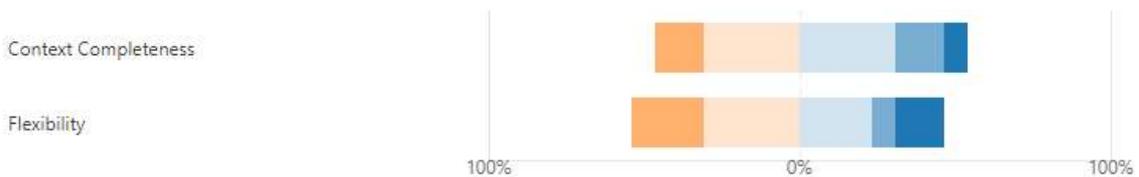
■ Not Important at All
 ■ Somewhat Important
 ■ Important
 ■ Very Important
 ■ Extremely Important
 ■ I don't know



15. Please, rank each of the following 2 context coverage quality sub-characteristics

[More Details](#)

■ Not Important at All
 ■ Somewhat Important
 ■ Important
 ■ Very Important
 ■ Extremely Important
 ■ I don't know



16. Please, provide any additional quality characteristic(s) (not listed above) that is(are) very or extremely important for your project.

[More Details](#)

3

Responses

Latest Responses

Annex 11. Automotive real-world use case analysis results: importance values, inclusion / exclusion decisions and weight factors

- Survey analysis results on IVI ECU embedded software quality characteristics and sub-characteristics

Perspective	Characteristics	First importance value	Final importance value	Final Decision	Σ importance valu	Weight factors
System / Software product quality	Functional Suitability	1.75	1.75	Included	9	0.194444 444
	Performance Efficiency	1	1	Included		0.111111 111
	Compatibility	0.75	0.8125	Excluded		
	Usability	1.5	1.5	Included		0.166666 667
	Reliability	2	2	Included		0.222222 222
	Security	0.5	0.93611111 1	Excluded		
	Maintainability	1.25	1.25	Included		0.138888 889
	Portability	1.5	1.5	Included		0.166666 667
Quality in use	Effectiveness	1.25	1.25	Included	4.5	0.277777 778
	Efficiency	1.25	1.25	Included		0.277777 778
	Satisfaction	2	2	Included		0.444444 444
	Freedom From Risk	0.75	0.8125	Excluded		
	Context Coverage	0.5	0.29166666 7	Excluded		
Characteristics	Sub-characteristics					
Functional Suitability	Functional Completeness	0.66666667	0.66666666 7	Excluded	3.16666667	
	Functional Correctness	1.66666667	1.66666666 7	Included		0.526315 789
	Functional Appropriateness	1.5	1.5	Included		0.473684 211
Performance Efficiency	Time-behavior	1.33333333	1.33333333 3	Included	3.66666667	0.363636 364
	Resource Utilization	1	1	Included		0.272727 273
	Capacity	1.33333333	1.33333333 3	Included		0.363636 364
Compatibility	Co-existence	0	0	Excluded	Excluded characteristic	
	Interoperability	1	1	Included		
	Appropriateness Recognizability	0	0	Excluded		
Usability	Learnability	0.5	0.5	Excluded	6	
	Operability	2	2	Included		0.333333 333
	User Error Protection	1.66666667	1.66666666 7	Included		0.277777 778
	User Interface Aesthetics	1.33333333	1.33333333 3	Included		0.222222 222
Reliability	Accessibility	1	1	Included	5	0.166666 667
	Maturity	1	1	Included		0.2
	Availability	1	1	Included		0.2
	Fault Tolerance	1.66666667	1.66666666 7	Included		0.333333 333

Security	Recoverability	1.333333333	1.333333333 3	Included	Excluded characteristic	0.266666667
	Confidentiality	1.666666667	1.666666667 7	Included		
	Integrity	1	1	Included		
	Non-Repudiation	0	0	Excluded		
	Accountability	0.333333333	0.333333333 3	Excluded		
	Authenticity	1	1	Included		
Maintainability	Modularity	1	1	Included	6.333333333	0.157894737
	Reusability	1	1	Included		0.157894737
	Analysability	1.333333333	1.333333333 3	Included		0.210526316
	Modifiability	1	1	Included		0.157894737
	Testability	2	2	Included		0.315789474
Portability	Adaptability	0.666666667	0.666666667 7	Excluded	2.333333333	0.428571429
	Installability	1	1	Included		0.571428571
	Replaceability	1.333333333	1.333333333 3	Included		
Satisfaction	Usefulness	1.333333333	1.333333333 3	Included	2.666666667	0.5
	Trust	1.333333333	1.333333333 3	Included		0.5
	Pleasure	0.333333333	0.333333333 3	Excluded		
	Comfort	0.666666667	0.666666667 7	Excluded		
Effectiveness	Effectiveness	1.25	1.25	Included	1.25	1
Freedom From Risk	Economic Risk Mitigation	0.5	0.5	Excluded	Excluded characteristic	
	Health and Safety Risk Mitigation	1	1	Included		
	Environmental Risk Mitigation	0	0	Excluded		
Efficiency	Efficiency	1.25	1.25	Included	1.25	1
Context Coverage	Context Completeness	0.666666667	0.666666667 7	Excluded	Excluded characteristic	
	Flexibility	0.666666667	0.666666667 7	Excluded		

- Survey analysis results on IVC ECU embedded software quality characteristics and sub-characteristics

Perspective	Characteristics	First importance value	Final importance value	Final Decision	\sum importance values	Weight factors
System / Software product quality	Functional Suitability	1.75	1.75	Included	7.048611111	0.248275862
	Performance Efficiency	1.25	1.25	Included		0.177339901
	Compatibility	1.5	1.5	Included		0.212807882
	Usability	0.25	0.659722222	Excluded		
	Reliability	1.5	1.5	Included		0.212807882
	Security	0.5	0.936111111	Excluded		
	Maintainability	0.25	0.9625	Excluded		
Quality in use	Portability	0.75	1.048611111	Included	2.75	0.148768473
	Effectiveness	1.5	1.5	Included		0.545454545
	Efficiency	0.75	0.75	Excluded		

Annexes

	Satisfaction	1.25	1.25	Included		0.454545455
	Freedom From Risk	0.75	0.993055556	Excluded		
	Context Coverage	0.5	0.166666667	Excluded		
Characteristics	Sub-characteristics					
Functional Suitability	Functional Completeness	1	1	Included		0.24
	Functional Correctness	1.5	1.5	Included	4.166666667	0.36
	Functional Appropriateness	1.666666667	1.666666667	Included		0.4
Performance Efficiency	Time-behavior	1	1	Included		1
	Resource Utilization	0.333333333	0.333333333	Excluded	1	
	Capacity	0	0	Excluded		
Compatibility	Co-existence	0.333333333	0.333333333	Excluded	2	
	Interoperability	2	2	Included		1
Usability	Appropriateness Recognizability	1	1	Included		
	Learnability	0.333333333	0.333333333	Excluded		
	Operability	1	1	Included		
	User Error Protection	0.333333333	0.333333333	Excluded		
	User Interface Aesthetics	0.666666667	-0.666666667	Excluded		
	Accessibility	0.333333333	0.333333333	Excluded		
Reliability	Maturity	1.666666667	1.666666667	Included		0.416666667
	Availability	1.333333333	1.333333333	Included	4	0.333333333
	Fault Tolerance	0.666666667	0.666666667	Excluded		
	Recoverability	1	1	Included		0.25
Security	Confidentiality	1.333333333	1.333333333	Included		
	Integrity	1.333333333	1.333333333	Included		
	Non-Repudiation	0	0	Excluded		
	Accountability	1	1	Included		
	Authenticity	0.333333333	0.333333333	Excluded		
Maintainability	Modularity	1.333333333	1.333333333	Included		
	Reusability	1.333333333	1.333333333	Included		
	Analysability	1	1	Included		
	Modifiability	1	1	Included		
	Testability	1.333333333	1.333333333	Included		
Portability	Adaptability	1.333333333	1.333333333	Included		1
	Installability	0.666666667	0.666666667	Excluded	1.333333333	
	Replaceability	0.333333333	0.333333333	Excluded		
Satisfaction	Usefulness	1.333333333	1.333333333	Included		0.444444444
	Trust	1.666666667	1.666666667	Included		0.555555556
	Pleasure	0.666666667	-0.666666667	Excluded	3	
	Comfort	0.666666667	-0.666666667	Excluded		
Effectiveness	Effectiveness	1.5	1.5	Included	1.5	1
Freedom From Risk	Economic Risk Mitigation	1	1	Included		
	Health and Safety Risk Mitigation	1.333333333	1.333333333	Included		

	Environmental Risk Mitigation	0.333333333	0.333333333	Excluded	
Efficiency	Efficiency	0.75	0.75	Excluded	Excluded characteristic
Context Coverage	Context Completeness	0.333333333	0.333333333	Excluded	Excluded characteristic
	Flexibility	0	0	Excluded	

- Survey analysis results on ADAS ECU embedded software quality characteristics and sub-characteristics

Perspective	Characteristics	First importance value	Final importance value	Final Decision	Σ importance values	Weight factors
System / Software product quality	Functional Suitability	1.75	1.75	Included	7.604166667	0.230136986
	Performance Efficiency	1.75	1.75	Included		0.230136986
	Compatibility	0.25	0.6875	Excluded		
	Usability	0.25	1.083333333	Included		0.142465753
	Reliability	1	1	Included		0.131506849
	Security	0.75	0.998611111	Excluded		
	Maintainability	1	1	Included		0.131506849
	Portability	0.75	1.020833333	Included		0.134246575
Quality in use	Effectiveness	1.25	1.25	Included	4.583333333	0.272727273
	Efficiency	1.25	1.25	Included		0.272727273
	Satisfaction	0.75	1.083333333	Included		0.236363636
	Freedom From Risk	1	1	Included		0.218181818
	Context Coverage	0	0.166666667	Excluded		
Characteristics	Sub-characteristics					
Functional Suitability	Functional Completeness	0.666666667	0.666666667	Excluded	3.166666667	
	Functional Correctness	1.666666667	1.666666667	Included		0.526315789
	Functional Appropriateness	1.5	1.5	Included		0.473684211
Performance Efficiency	Time-behavior	1.333333333	1.333333333	Included	3.666666667	0.363636364
	Resource Utilization	1	1	Included		0.272727273
	Capacity	1.333333333	1.333333333	Included		0.363636364
Compatibility	Co-existence	0	0	Excluded	Excluded characteristic	
	Interoperability	1	1	Included		
Usability	Appropriateness Recognizability	0	0	Excluded	6	
	Learnability	0.5	0.5	Excluded		
	Operability	2	2	Included		0.333333333
	User Error Protection	1.666666667	1.666666667	Included		0.277777778
	User Interface Aesthetics	1.333333333	1.333333333	Included		0.222222222
Reliability	Accessibility	1	1	Included	5	0.166666667
	Maturity	1	1	Included		0.2
	Availability	1	1	Included		0.2
	Fault Tolerance	1.666666667	1.666666667	Included		0.333333333
	Recoverability	1.333333333	1.333333333	Included		0.266666667
Security	Confidentiality	1.666666667	1.666666667	Included	Excluded characteristic	
	Integrity	1	1	Included		
	Non-Repudiation	0	0	Excluded		

Annexes

	Accountability	0.333333333	0.333333333	Excluded		
	Authenticity	1	1	Included		
	Modularity	1	1	Included		0.157894737
	Reusability	1	1	Included		0.157894737
Maintainability	Analysability	1.333333333	1.333333333	Included	6.333333333	0.210526316
	Modifiability	1	1	Included		0.157894737
	Testability	2	2	Included		0.315789474
	Adaptability	0.666666667	0.666666667	Excluded		
Portability	Installability	1	1	Included	2.333333333	0.428571429
	Replaceability	1.333333333	1.333333333	Included		0.571428571
	Usefulness	1.333333333	1.333333333	Included		0.5
Satisfaction	Trust	1.333333333	1.333333333	Included	2.666666667	0.5
	Pleasure	0.333333333	0.333333333	Excluded		
	Comfort	0.666666667	0.666666667	Excluded		
Effectiveness	Effectiveness	1.25	1.25	Included	1.25	1
	Economic Risk Mitigation	0.5	0.5	Excluded		
Freedom From Risk	Health and Safety Risk Mitigation	1	1	Included	1	1
	Environmental Risk Mitigation	0	0	Excluded		
Efficiency	Efficiency	1.25	1.25	Included	1.25	1
Context Coverage	Context Completeness	0.666666667	0.666666667	Excluded		
	Flexibility	0.666666667	0.666666667	Excluded		Excluded characteristic

- Survey analysis results on FOTA embedded software quality characteristics and sub-characteristics

Perspective	Characteristics	First importance value	Final importance value	Final Decision	\sum importance values	Weight factors
System / Software product quality	Functional Suitability	1.75	1.75	Included	7.604166667	0.230136986
	Performance Efficiency	1.75	1.75	Included		0.230136986
	Compatibility	0.25	0.6875	Excluded		
	Usability	0.25	1.083333333	Included		0.142465753
	Reliability	1	1	Included		0.131506849
	Security	0.75	0.998611111	Excluded		
	Maintainability	1	1	Included		0.131506849
	Portability	0.75	1.020833333	Included		0.134246575
Quality in use	Effectiveness	1.25	1.25	Included	4.583333333	0.272727273
	Efficiency	1.25	1.25	Included		0.272727273
	Satisfaction	0.75	1.083333333	Included		0.236363636
	Freedom From Risk	1	1	Included		0.218181818
	Context Coverage	0	0.166666667	Excluded		
Characteristics	Sub-characteristics					
Functional Suitability	Functional Completeness	0.666666667	0.666666667	Excluded	3.166666667	
	Functional Correctness	1.666666667	1.666666667	Included		0.526315789
	Functional Appropriateness	1.5	1.5	Included		0.473684211
	Time-behavior	1.333333333	1.333333333	Included		0.363636364

Performance Efficiency	Resource Utilization	1	1	Included		0.272727273
	Capacity	1.333333333	1.333333333	Included		0.363636364
Compatibility	Co-existence	0	0	Excluded	Excluded characteristic	
	Interoperability	1	1	Included		
Usability	Appropriateness Recognizability	0	0	Excluded		
	Learnability	0.5	0.5	Excluded		
	Operability	2	2	Included	6	0.333333333
	User Error Protection	1.666666667	1.666666667	Included		0.277777778
	User Interface Aesthetics	1.333333333	1.333333333	Included		0.222222222
Accessibility	1	1	Included	0.166666667		
Reliability	Maturity	1	1	Included	5	0.2
	Availability	1	1	Included		0.2
	Fault Tolerance	1.666666667	1.666666667	Included		0.333333333
	Recoverability	1.333333333	1.333333333	Included		0.266666667
Security	Confidentiality	1.666666667	1.666666667	Included	Excluded characteristic	
	Integrity	1	1	Included		
	Non-Repudiation	0	0	Excluded		
	Accountability	0.333333333	0.333333333	Excluded		
	Authenticity	1	1	Included		
Maintainability	Modularity	1	1	Included	6.333333333	0.157894737
	Reusability	1	1	Included		0.157894737
	Analysability	1.333333333	1.333333333	Included		0.210526316
	Modifiability	1	1	Included		0.157894737
	Testability	2	2	Included		0.315789474
Portability	Adaptability	0.666666667	0.666666667	Excluded	2.333333333	
	Installability	1	1	Included		0.428571429
	Replaceability	1.333333333	1.333333333	Included		0.571428571
Satisfaction	Usefulness	1.333333333	1.333333333	Included	2.666666667	0.5
	Trust	1.333333333	1.333333333	Included		0.5
	Pleasure	0.333333333	0.333333333	Excluded		
	Comfort	0.666666667	0.666666667	Excluded		
Effectiveness	Effectiveness	1.25	1.25	Included	1.25	1
Freedom From Risk	Economic Risk Mitigation	0.5	0.5	Excluded	1	
	Health and Safety Risk Mitigation	1	1	Included		1
	Environmental Risk Mitigation	0	0	Excluded		
Efficiency	Efficiency	1.25	1.25	Included	1.25	1
Context Coverage	Context Completeness	0.666666667	0.666666667	Excluded	Excluded characteristic	
	Flexibility	0.666666667	0.666666667	Excluded		

Annex 12. Basic set of measures to enable the real-world use case quality models

The purpose of this basic set of measures is to enable the use operations of the real-world use case quality models. This set should be reviewed, updated, and completed as needed, especially to integrate the temporal polymorphism behavior due to the evolution over the development life cycle stages. These measures are extracted from ISO/IEC 25023 [161] for system / software product quality measures, and from ISO/IEC 25022 [142] for quality in use measures. Further details are available in the standard documentation.

- System / Software product quality perspective

Characteristics	Sub-characteristics	Metrics	Formula	Formula details
Functional Suitability	Functional Completeness	Functional coverage	$X = 1 - \frac{A}{B}$	A = Number of functions missing B = Number of functions specified
	Functional Correctness	Functional Correctness	$X = 1 - \frac{A}{B}$	A = Number of functions that are incorrect B = Number of functions considered
	Functional Appropriateness	Functional appropriateness of usage objective	$X = 1 - \frac{A}{B}$	A = Number of functions missing or incorrect among those that are required for achieving a specific usage objective B = Number of functions required for achieving a specific usage objective
		Functional appropriateness of system	$X = \sum_{i=1}^n \frac{A_i}{n}$	A_i = Appropriateness score for usage objective i , that is, the measured value of Functional appropriateness of usage objective for i -th specific usage objective n = Number of usage objectives
Performance Efficiency	Time-behavior	Mean response time	$X = \sum_{i=1}^n \frac{A_i}{n}$	A_i = Time taken by the system to respond to a specific user task or system task at i -th measurement n = Number of responses measured
		Response time adequacy	$X = \frac{A}{B}$	A = Mean response time measured B = Target response time specified
	Time-behavior	Mean turnaround time	$X = \sum_{i=1}^n \frac{(B_i - A_i)}{n}$	A_i = Time of starting a job i B_i = Time of completing the job i n = Number of measurements
		Turnaround time adequacy	$X = \frac{A}{B}$	A = Mean turnaround time B = Target turnaround time specified
Performance Efficiency	Time-behavior	Mean throughput	$X = \sum_{i=1}^n \frac{(A_i)}{B_i}$	A_i = Number of jobs completed during the i -th observation time B_i = i -th observation time period n = Number of observations
		Mean processor utilization	$X = \sum_{i=1}^n \frac{(A_i)}{B_i}$	A_i = Processor time actually used to execute a given set of tasks in observation i B_i = Operation time to perform the tasks in observation i n = Number of observations
	Resource Utilization	Mean memory utilization	$X = \sum_{i=1}^n \frac{(A_i)}{B_i}$	A_i = Size of memory actually used to perform a given set of tasks for i -th sample processing B_i = Size of memory available to perform the tasks during i -th sample processing n = Number of samples processed
		Mean I/O devices utilization	$X = \sum_{i=1}^n \frac{(A_i)}{B_i}$	A_i = Duration of I/O device(s) busy time to perform a given set of tasks for i -th observation B_i = Duration of I/O operations to perform the tasks for i -th observation n = Number of observations

		Bandwidth utilization	$X = \frac{A}{B}$	A = Bandwidth of actual transmission measured over time to perform a given set of tasks B = Bandwidth capacity available to perform a given set of tasks
		Transaction processing capacity	$X = \frac{A}{B}$	A = Number of transactions completed during observation time B = Duration of observation
	Capacity	User access capacity	$X = \sum_{i=1}^n \frac{A_i}{n}$	A_i = Maximum number of users who can simultaneously access the system at i -th observation n = Number of observations
		User access increase adequacy	$X = \frac{A}{B}$	A = Number of users successfully added during observation time B = Duration of observation
		Data formats exchangeability	$X = \frac{A}{B}$	A = Number of data formats exchangeable with other software or systems B = Number of data formats specified to be exchangeable
Compatibility	Interoperability	Data exchange protocol sufficiency	$X = \frac{A}{B}$	A = Number of data exchange protocols supported B = Number of data exchange protocols specified to be supported
		External interface adequacy	$X = \frac{A}{B}$	A = Number of external interfaces that are functional B = Number of external interfaces specified
		Operational consistency	$X = 1 - \frac{A}{B}$	A = Number of specific interactive tasks that are performed inconsistently B = Number of specific interactive tasks that need to be consistent
		Message clarity	$X = \frac{A}{B}$	A = Number of messages that convey the right outcome or instructions to the user B = Number of messages implemented
		Functional customizability	$X = \frac{A}{B}$	A = Number of functions and operational procedures which can be customized for user's convenience B = Number of functions and operational procedures for which users could benefit from customization
Usability	Operability	User interface customizability	$X = \frac{A}{B}$	A = Number of user interface elements that can be customized B = Number of user interface elements that could benefit from customization
		Monitoring capability	$X = \frac{A}{B}$	A = Number of functions having state monitoring capability B = Number of functions that could benefit from monitoring capability
		Undo capability	$X = \frac{A}{B}$	A = Number of tasks that provide undo capability or prompt for re-confirmation B = Number of tasks for which users could benefit from having re-confirmation or undo capability
		Understandable categorization of information	$X = \frac{A}{B}$	A = Number of information structures that are familiar and convenient for the intended users B = Number of information structures used
		Appearance consistency	$X = 1 - \frac{A}{B}$	A = Number of user interfaces with similar items but with different appearances B = Number of user interfaces with similar items

Annexes

	Input device support	$X = \frac{A}{B}$	A = Number of tasks that can be initiated by all appropriate input modalities B = Number of tasks supported by the system
User Error Protection	Avoidance of user operation error	$X = \frac{A}{B}$	A = Number of user actions and inputs that are protected from causing any system malfunction B = Number of user actions and inputs that could be protected from causing any system malfunction
	User entry error correction	$X = \frac{A}{B}$	A = Number of entry errors for which the system provides a suggested correct value B = Number of entry errors detected
	User error recoverability	$X = \frac{A}{B}$	A = Number of user errors that are designed and tested to be recovered by the system B = Number of user errors which can occur during operation
User Interface Aesthetics	Appearance aesthetics of user interfaces	$X = \frac{A}{B}$	A = Number of display interfaces aesthetically pleasing to the users in appearance B = Number of display interfaces
Accessibility	Accessibility for users with disabilities	$X = \frac{A}{B}$	A = Number of functions successfully usable by the users with a specific disability B = Number of functions implemented
	Supported languages adequacy	$X = \frac{A}{B}$	A = Number of languages actually supported B = Number of languages needed to be supported
Maturity	Fault correction	$X = \frac{A}{B}$	A = Number of reliability-related faults corrected in design /coding/testing phase B = Number of reliability-related faults detected in design/coding/testing phase
	Mean time between failure (MTBF)	$X = \frac{A}{B}$	A = Operation time B = Number of system/software failures actually occurred
	Failure rate	$X = \frac{A}{B}$	A = Number of failures detected during observation time B = Duration of observation
	Test coverage	$X = \frac{A}{B}$	A = Number of system or software capabilities, operational scenarios or functions that are actually performed B = Number of system or software capabilities, operational scenarios or functions which are included in their associated test suites
Reliability	Availability	$X = \frac{A}{B}$	A = System operation time actually provided B = System operation time specified in the operation schedule
	Mean down time	$X = \frac{A}{B}$	A = Total down time B = Number of breakdowns observed
Fault Tolerance	Failure avoidance	$X = \frac{A}{B}$	A = Number of avoided critical and serious failure occurrences (based on test cases) B = Number of executed test cases of fault pattern (almost causing failure) during testing
	Redundancy of components	$X = \frac{A}{B}$	A = Number of system components redundantly installed B = Number of system components
	Mean fault notification time	$X = \sum_{i=1}^n \frac{(A_i)}{B_i} \frac{1}{n}$	A_i = Time at which the fault i is reported by the system B_i = Time at which fault i is detected n = Number of faults detected

Recoverability	Mean recovery time	$X = \frac{\sum_{i=1}^n A_i}{n}$	A_i = Total time to recover the downed software / system and re-initiate operation for each failure i n = Number of failures
	Backup data completeness	$X = \frac{A}{B}$	A = Number of data items actually backed up regularly B = Number of data items requiring backup for error recovery
Modularity	Coupling of components	$X = \frac{A}{B}$	A = Number of components which are implemented with no impact on others B = Number of specified components which are required to be independent
	Cyclomatic complexity adequacy	$X = 1 - \frac{A}{B}$	A = Number of software modules which have a cyclomatic complexity score that exceeds the specified threshold B = Number of software modules implemented
Reusability	Reusability of assets	$X = \frac{A}{B}$	A = Number of assets which are designed and implemented to be reusable B = Number of assets in a system
	Coding rules conformity	$X = \frac{A}{B}$	A = Number of software modules conforming to coding rules for a specific system B = Number of software modules implemented
Maintainability	System log completeness	$X = \frac{A}{B}$	A = Number of logs that are actually recorded in the system B = Number of logs for which audit trails are required during operation
	Diagnosis function effectiveness	$X = \frac{A}{B}$	A = Number of diagnostic functions useful for causal analysis B = Number of diagnostic functions implemented
	Diagnosis function sufficiency	$X = \frac{A}{B}$	A = Number of diagnostic functions implemented B = Number of diagnostic functions required
	Modification efficiency	$X = \frac{\sum_{i=1}^n \left(\frac{A_i}{B_i}\right)}{n}$	A_i = Total work time spent for making a specific type of modification i B_i = Expected time for making the specific type of modification i n = Number of modifications measured
Modifiability	Modification correctness	$X = 1 - \frac{A}{B}$	A = Number of modifications that caused an incident or failure within a defined period after being implemented B = Number of modifications implemented
	Modification capability	$X = \frac{A}{B}$	A = Number of items actually modified within a specified duration B = Number of items required to be modified within a specified duration
Testability	Test function completeness	$X = \frac{A}{B}$	A = Number of test functions implemented as specified B = Number of test functions required
	Autonomous testability	$X = \frac{A}{B}$	A = Number of tests that can be simulated by stub among the tests which depend on other systems B = Number of tests which depend on other systems

Annexes

	Test restartability	$X = \frac{A}{B}$	A = Number of cases in which maintainer can pause and restart executing test run at desired points to check step by step B = Number of cases in which executing test run can be paused
Adaptability	Hardware environmental adaptability	$X = 1 - \frac{A}{B}$	A = Number of functions which were not completed or results which were insufficient to meet requirements during testing B = Number of functions which were tested in different hardware environment
	System software environmental adaptability	$X = 1 - \frac{A}{B}$	A = Number of functions which were not completed or results which were insufficient to meet requirements during testing B = Number of functions which were tested in different system software environment
	Operational environment adaptability	$X = 1 - \frac{A}{B}$	A = Number of functions which were not completed or results which were insufficient to meet requirements during operational testing with user's environment B = Number of functions which were tested in different operational environment
	Installation time efficiency	$X = \sum_{i=1}^n \frac{\left(\frac{A_i}{B_i}\right)}{n}$	A_i = Total work time spent for making an installation i B_i = Expected time for making an installation i n = Number of installations measured
Portability	Ease of installation	$X = \frac{A}{B}$	A = Number of cases in which a user succeeds to customize the installation procedure B = Number of cases in which a user attempted to customize the installation procedure for user's convenience
	Usage similarity	$X = \frac{A}{B}$	A = Number of user functions which can be performed without any additional learning or workaround B = Number of user functions in the replaced software product
Replaceability	Product quality equivalence	$X = \frac{A}{B}$	A = Number of quality measures of the new product which are better or equal to the replaced product B = Number of quality measures of the replaced software product that are relevant
	Functional inclusiveness	$X = \frac{A}{B}$	A = Number of functions which produce similar results as before B = Number of functions which have to be used in the replaced software product
	Data reusability / import capability	$X = \frac{A}{B}$	A = Number of data which can be used continuously as before B = Number of data which are to be used continuously in the replaced software product
	Consequences of fatigue	$X = 1 - \frac{A}{B}$	A = Current performance B = Initial performance

- Quality in use perspective

Characteristics	Sub-characteristics	Metrics	Formula	Formula details
Satisfaction	Usefulness	Overall satisfaction	$X = \sum_{i=1}^n A_i$	A_i = Response to i - th question n = Number of questions in questionnaire
		Satisfaction with features	$X = \sum_{i=1}^n A_i$	A_i = Response to i - th question related to a specific feature n = Number of questions in questionnaire
		Discretionary usage	$X = \frac{A}{B}$	A = Number of users using a specific function, application or system B = Number of potential users who could have used the specific function, application, or system
		Feature utilization	$X = \frac{A}{B}$	A = Number of users using a particular feature B = Number of users in an identified set of users of the system
		Proportion of users complaining	$X = \frac{A}{B}$	A = Number of users complaining B = Number of users using the system
		Proportion of user complaints about a particular feature	$X = \frac{A}{B}$	A = Number of user complaints for a particular feature B = Total number of user complaints about features
		Trust	User trust	$X = A$
Effectiveness	Effectiveness	Tasks completed	$X = \frac{A}{B}$	A = Number of unique tasks completed B = Total number of unique tasks attempted
		Objectives achieved	$\left\{ X = 1 - \sum_{i=1}^n A_i \mid X \geq 0 \right\}$	A_i = Proportional value of the i - th missing or incorrect objective in the task output (maximum value = 1) n = Number of missing or incorrect objectives in the task output
		Errors in a task	$X = A$	A = Number of errors made by the user during a task
		Tasks with errors	$X = \frac{A}{B}$	A = Number of tasks with errors B = Total number of tasks
		Task error intensity	$X = \frac{A}{B}$	A = Number of users making an error B = Total number of users performing the task
Freedom From Risk	Health and Safety Risk Mitigation	User health reporting frequency	$X = \frac{A}{B}$	A = Number of users reporting health problems B = Total number of users
		User health and safety impact	$X = \left(\frac{1}{T_b} \right) \sum_{i=1}^n \left(\frac{T_{a_i}}{S_i} \right)$	T_{a_i} = Length of time for which the i - th person is affected S_i = Degree of significance of the impact on the i - th person T_b = Length of time from start of system in operation n = Number of affected people
		Safety of people affected by use of the system	$X = \frac{A}{B}$	A = Number of people put at hazard B = Total number of people who could be affected by use of the system
Efficiency	Efficiency	Task time	$X = T$	T = Task time
		Time efficiency	$X = \frac{A}{T}$	A = Number of objectives achieved T = Time
		Cost-effectiveness	$X = \frac{A}{B}$	A = Total cost of carrying out the task B = Number of objectives achieved
		Productive time ratio	$X = \frac{T_a}{T_b}$	T_a = Productive time = time taken to complete the task - time spent getting help or assistance - time taken recovering from errors - time taken

Annexes

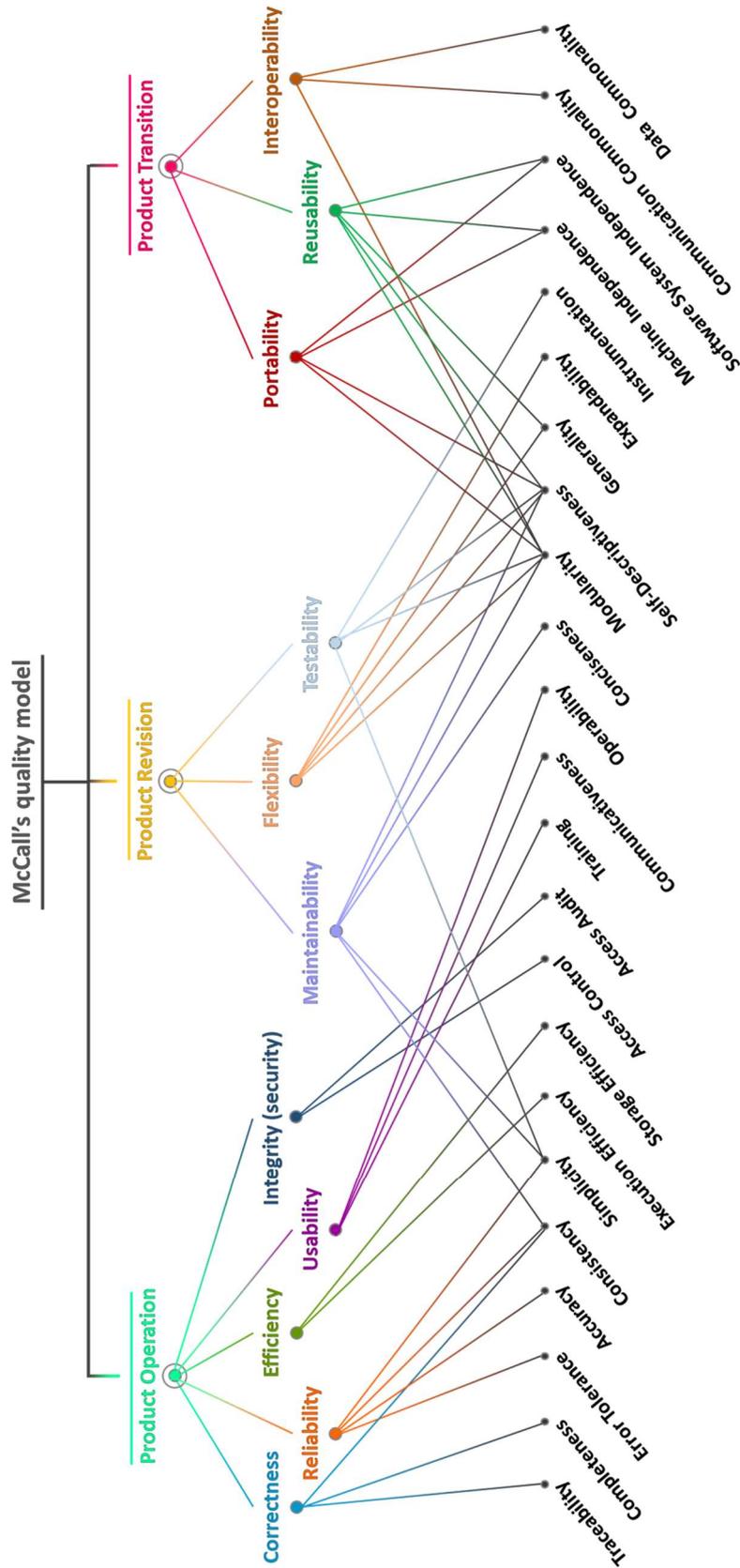
searching ineffectually
 T_b = Task time

Unnecessary actions	$X = \frac{A}{B}$	A = Number of actions actually that were not necessary to achieve the task B = Number of actions performed by the user
Consequences of fatigue	$X = 1 - \frac{A}{B}$	A = Current performance B = Initial performance

Terminology	Definition
Accessibility	Code possesses the characteristic of accessibility to the extent that it facilitates selective use of its parts. (Examples: variable dimensioned arrays, or not using absolute constants.) Accessibility is necessary for efficiency, testability, and human engineering.
Accountability	Code possesses the characteristic of accountability to the extent that its usage can be measured. This means that critical segments of code can be instrumented with probes to measure timing, whether specified branches are exercised, etc. Code used for probes is preferably invoked by conditional assembly techniques to eliminate the additional instruction words or added execution times when the measurements are not needed.
Accuracy	Code possesses the characteristic of accuracy to the extent that its outputs are sufficiently precise to satisfy their intended use. Necessary for reliability
Usability (As-is utility)	Code possesses the characteristic of usability to the extent that it is reliable, efficient, and human engineered.
Augmentability	Code possesses the characteristic of augmentability to the extent that it can easily accommodate expansion in component computational functions or data storage requirements. This is a necessary characteristic for modifiability.
Communicativeness	Code possesses the characteristic of communicativeness to the extent that it facilitates the specification of inputs and provides outputs whose form and content are easy to assimilate and useful. Communicativeness is necessary for testability and human engineering.
Completeness	Code possesses the characteristic of completeness to the extent that all its parts are present, and each part is fully developed.
Conciseness	Code possesses the characteristic of conciseness to the extent that excessive information is not present.
Consistency	Code possesses the characteristic of internal consistency to the extent that it contains uniform notation, terminology, and symbology within itself, and external consistency to the extent that the content is traceable to the requirements.
Device independence	Code possesses the characteristic of device independence to the extent it can be executed on computer hardware configurations other than its current one. Clearly, this characteristic is a necessary condition for portability.
Efficiency	Code possesses the characteristic of efficiency to the extent that it fulfills its purpose without waste of resources.
Human engineering	Code possesses the characteristic of human engineering to the extent that it fulfills its purpose without wasting the users' time and energy or degrading their morale. This characteristic implies accessibility, robustness, and communicativeness.
Legibility	Code possesses the characteristic of legibility to the extent that its function is easily discerned by reading the code. (Example: complex expressions have mnemonic variable names and parentheses even if unnecessary.) Legibility is necessary for understandability.
Maintainability	Code possesses the characteristic of maintainability to the extent that it facilitates updating to satisfy new requirements or to correct deficiencies.
Modifiability	Code possesses the characteristic of modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined. Note the higher level of abstractness of this characteristic as compared with augmentability.
Portability	Code possesses the characteristic of portability to the extent that it can be operated easily and well on computer configurations other than its current one.
Reliability	Code possesses the characteristic reliability to the extent that it can be expected to perform its intended functions satisfactorily
Robustness / Integrity	Code possesses the characteristic of robustness to the extent that it can continue to perform despite some violation of the assumptions in its specification.
Self-containedness	Code possesses the characteristic of self-containedness to the extent that it performs all its explicit and implicit functions within itself. Examples of implicit functions are initialization, input checking, diagnostics, etc.
Self-descriptiveness	Code possesses the characteristic of self-descriptiveness to the extent that it contains enough information for a reader to determine or verify its objectives, assumptions, constraints, inputs, outputs, components, and revision status. Commentary and traceability of previous changes by transforming previous versions of code into non-executable but present (or available by macro calls) code are some of the ways of providing this characteristic. Self-descriptiveness is necessary for both testability and understandability.
Structuredness	Code possesses the characteristic of structuredness to the extent that it possesses a definite pattern of organization of its interdependent parts.
Testability	Code possesses the characteristic of testability to the extent that it facilitates the establishment of verification criteria and supports evaluation of its performance.
Understandability	Code possesses the characteristic of understandability to the extent that its purpose is clear to the inspector.

- McCall's quality model [41] (1977)

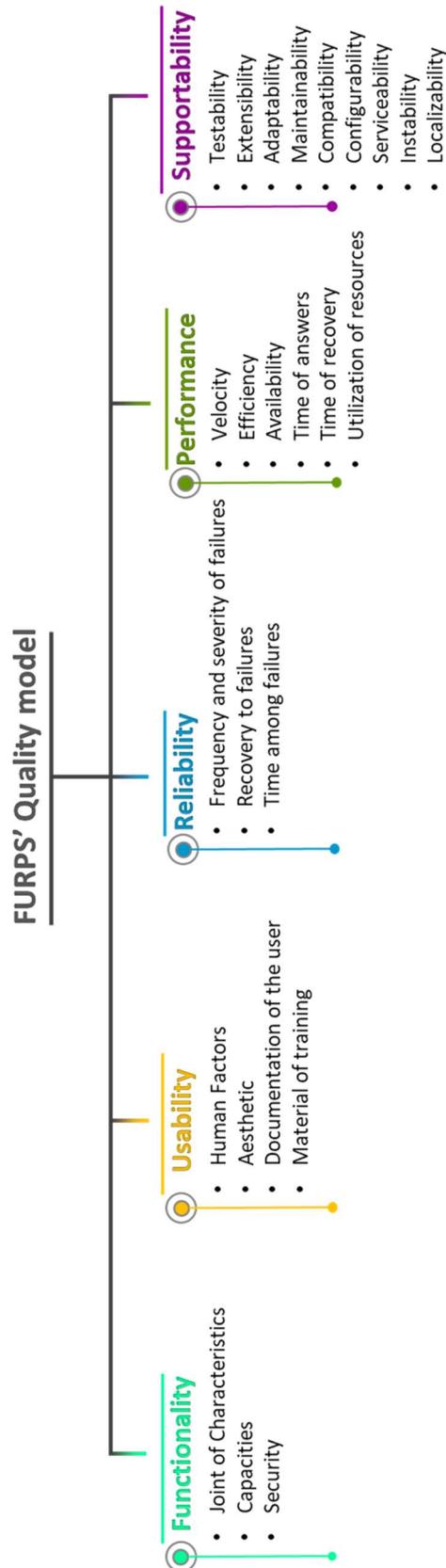
This model is composed of
 { 3 perspectives
 11 quality factors
 23 quality criteria



Terminology	Definition
Access audit	Those attributes of the software that provide for an audit of the access of software and data.
Access control	Those attributes of the software that provide for control of the access of software and data.
Accuracy	Those attributes of the software that provide the required precision in calculations and outputs.
Communication commonality	Those attributes of the software that Interoperability provide the use of standard protocols and Interface routines.
Communicativeness	Those attributes of the software that provide useful inputs and outputs which can be assimilated.
Completeness	Those attributes of the software that provide full implementation of the functions required.
Conciseness	Those attributes of the software that provide for implementation of a function with a minimum amount of code.
Consistency	Those attributes of the software that provide uniform design and implementation techniques and notation.
Correctness	Extent to which a program satisfies its specifications and fulfills the user's mission objectives.
Data commonality	Those attributes of the software that provide the use of standard data representations.
Efficiency	The amount of computing resources and code required by a program to perform a function.
Error Tolerance	Those attributes of the software that provide continuity of operation under nonnominal conditions.
Execution efficiency	Those attributes of the software that provide for minimum processing time.
Expandability	Those attributes of the software that provide for expansion of data storage requirements or computational functions.
Flexibility	Effort required to test a program to insure it performs its intended function.
Generality	Those attributes of the software that provide breadth to the functions performed.
Instrumentation	Those attributes of the software that provide for the measurement of usage or identification of errors.
Integrity (security)	Extent to which access to software or data by unauthorized persons can be controlled.
Interoperability	Effort required to couple one system with another.
Machine independence	Those attributes of the software that determine its dependency on the hardware system.
Maintainability	Effort required to locate and fix an error in an operational program.
Modularity	Those attributes of the software that provide a structure of highly independent modules.
Operability	Those attributes of the software that determine operation and procedures concerned with the operation of the software.
Portability	Effort required to transfer a program from one hardware configuration and/or software system environment to another.
Reliability	Extent to which a program can be expected to perform its intended function with required precision.
Reusability	Extent to which a program can be used in other applications - related to the packaging and scope of the functions that programs perform.
Self-Descriptiveness	Those attributes of the software that provide explanation of the implementation of a function.
Simplicity	Those attributes of the software that provide implementation of functions in the most understandable manner. (Usually avoidance of practices which increase complexity.)
Software system independence	Those attributes of the software that determine its dependency on the software environment (operating systems, utilities, input/output routines, etc.)
Storage efficiency	Those attributes of the software that provide for minimum storage requirements during operation.
Testability	Effort required to modify an operational program.
Traceability	Those attributes of the software that provide a thread from the requirements to the implementation with respect to the specific development and operational environment.
Training	Those attributes of the software that provide transition from current operation or initial familiarization.
Usability	Effort required to learn, operate, prepare input, and interpret output of a program.

- FURPS quality model [85] (1987)

This model is composed of { 5 Components
25 Sub-components

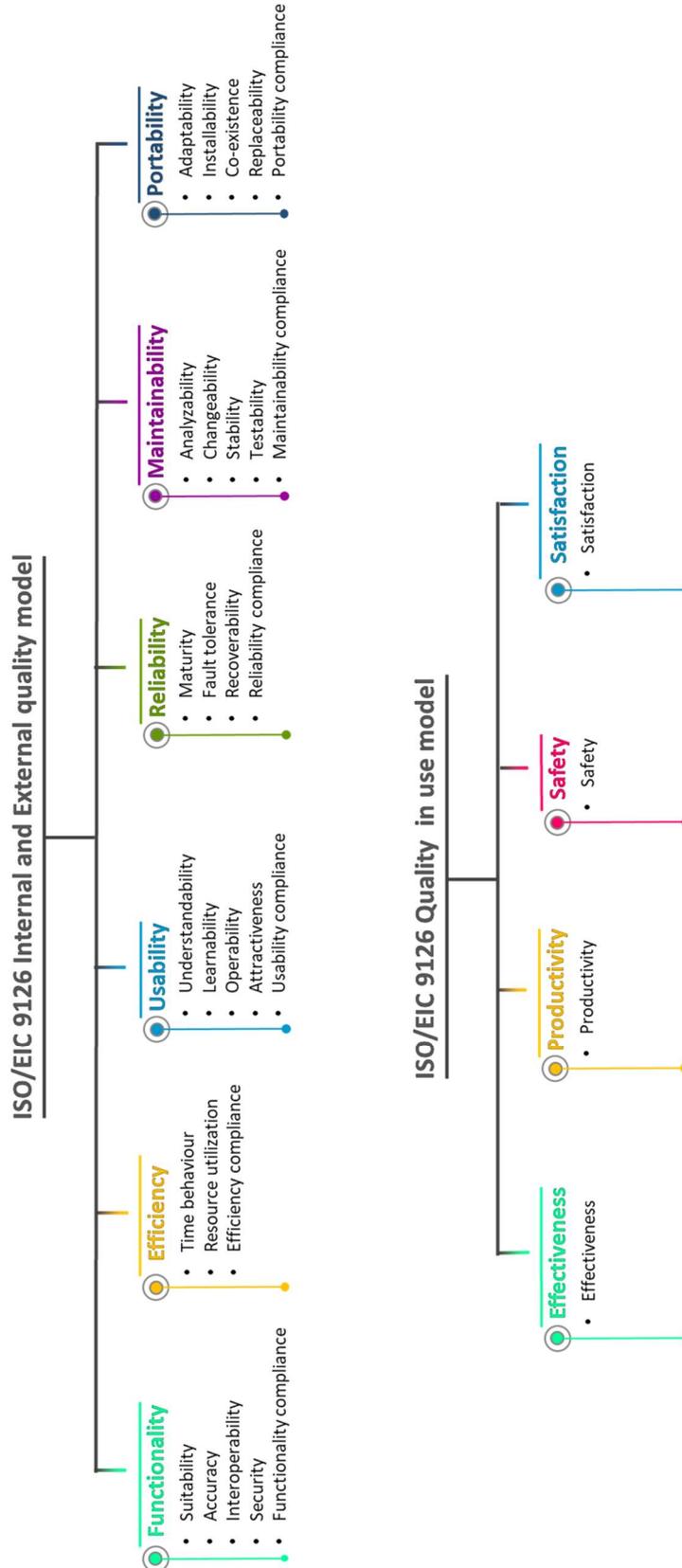


This model uses same terminology than Boehm's and McCall's quality models

- ISO/IEC 9126 quality model [24] (1991)

This model is composed of

- 2 quality perspectives
- 10 quality characteristics
- 31 quality sub-characteristics

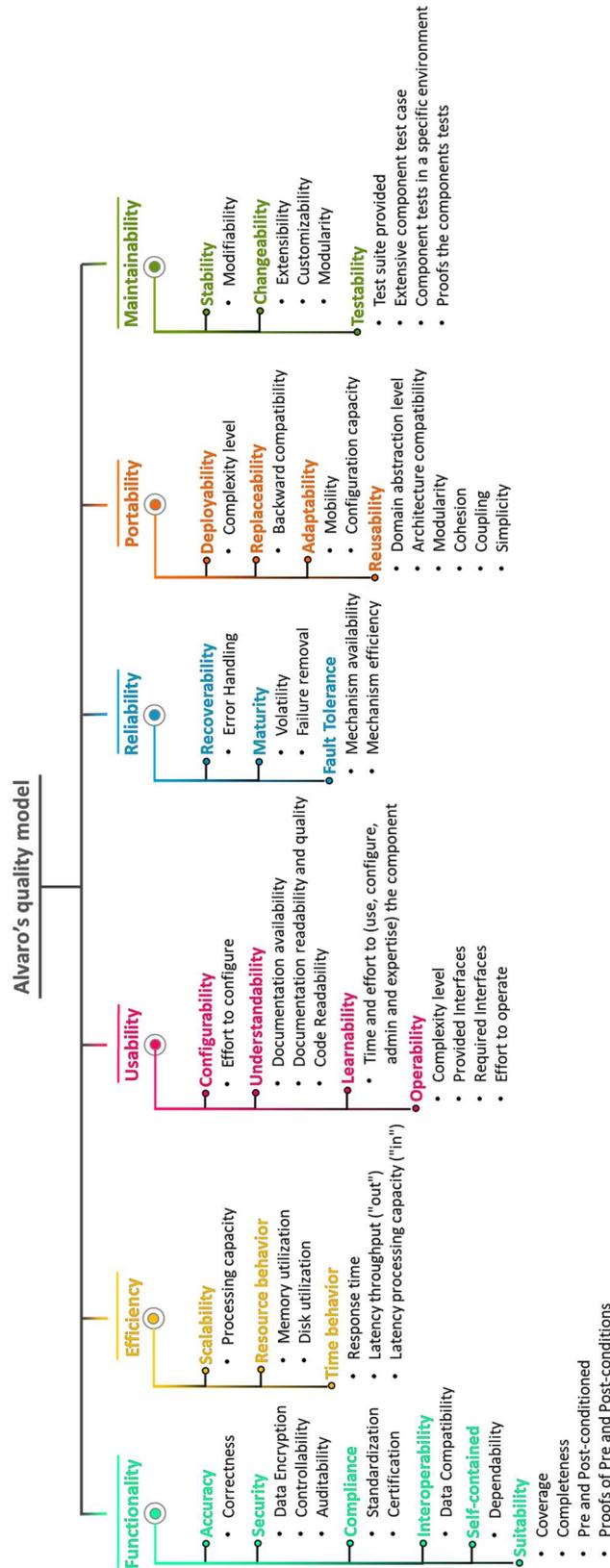


Terminology	Definition
Accuracy	The capability of the software product to provide the right or agreed results or effects with the needed degree of precision.
Adaptability	The capability of the software product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered.
Analyzability	The capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.
Attractiveness	The capability of the software product to be attractive to the user.
Changeability	The capability of the software product to enable a specified modification to be implemented.
Co-existence	The capability of the software product to co-exist with other independent software in a common environment sharing common resources.
Effectiveness	The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.
Efficiency	The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions.
Efficiency compliance	The capability of the software product to adhere to standards or conventions relating to efficiency.
Fault tolerance	The capability of the software product to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
Functionality	The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.
Functionality compliance	The capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality.
Installability	The capability of the software product to be installed in a specified environment.
Interoperability	The capability of the software product to interact with one or more specified systems.
Learnability	The capability of the software product to enable the user to learn its application.
Maintainability	The capability of the software product to be modified. Modifications may include corrections, improvements, or adaptation of the software to changes in environment, and in requirements and functional specifications.
Maintainability compliance	The capability of the software product to adhere to standards or conventions relating to maintainability.
Maturity	The capability of the software product to avoid failure as a result of faults in the software.
Operability	The capability of the software product to enable the user to operate and control it.
Portability	The capability of the software product to be transferred from one environment to another.
Portability compliance	The capability of the software product to adhere to standards or conventions relating to portability.
Productivity	The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.
Recoverability	The capability of the software product to re-establish a specified level of performance and recover the data directly affected in the case of a failure.
Reliability	The capability of the software product to maintain a specified level of performance when used under specified conditions.
Reliability compliance	The capability of the software product to adhere to standards, conventions or regulations relating to reliability.
Replaceability	The capability of the software product to be used in place of another specified software product for the same purpose in the same environment.
Resource utilization	The capability of the software product to use appropriate amounts and types of resources when the software performs its function under stated conditions.
Safety	The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property, or the environment in a specified context of use.
Satisfaction	The capability of the software product to satisfy users in a specified context of use.
Security	The capability of the software product to protect information and data so that unauthorized persons or systems cannot read or modify them, and authorized persons or systems are not denied access to them.
Stability	The capability of the software product to avoid unexpected effects from modifications of the software.
Suitability	The capability of the software product to provide an appropriate set of functions for specified tasks and user objectives.
Testability	The capability of the software product to enable modified software to be validated.

Time behaviour	The capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.
Understandability	The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.
Usability	The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.
Usability compliance	The capability of the software product to adhere to standards, conventions, style guides or regulations relating to usability.

- Alvaro quality model [36] (2010)

This model is composed of { 6 quality characteristics, 23 quality sub-characteristics, 48 attributes }

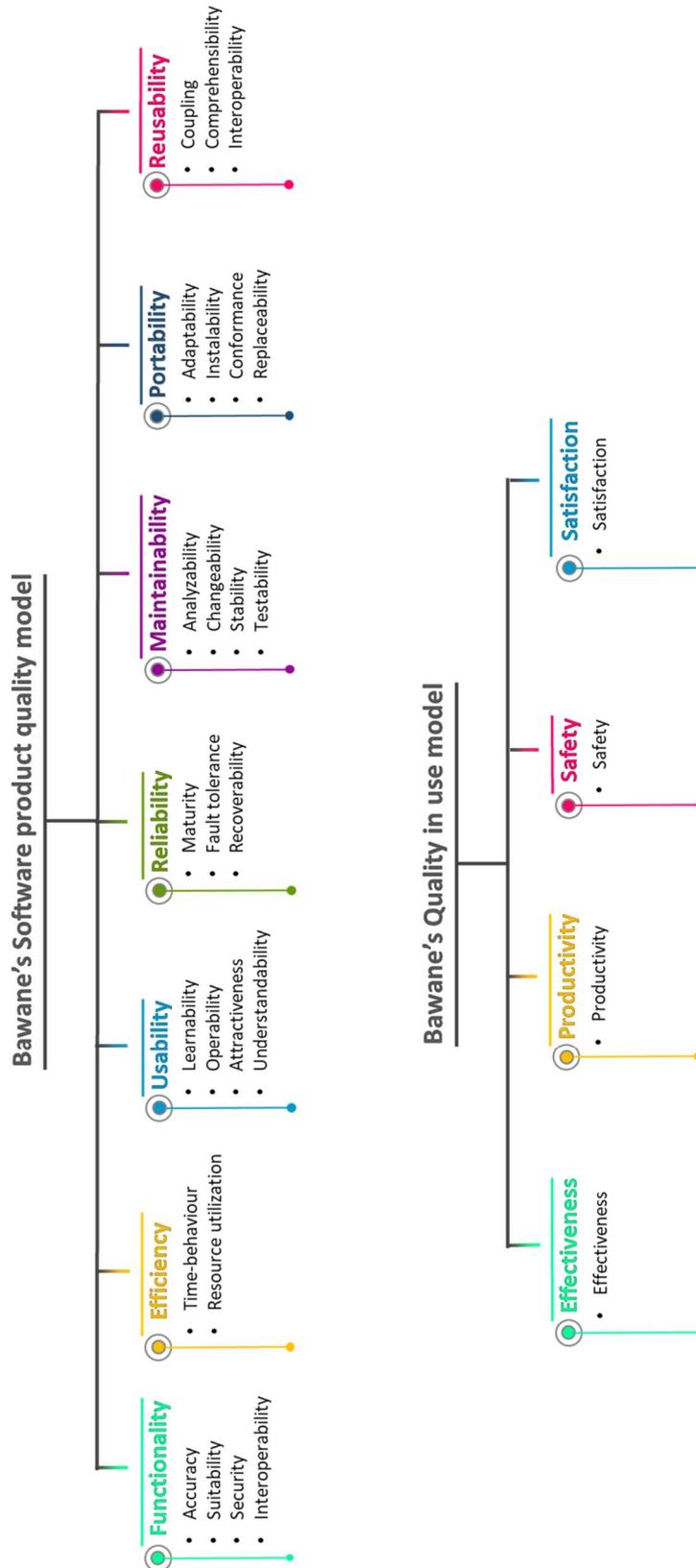


This model uses terminology from ISO/IEC 9126, and ISO/IEC/IEEE 25010

- Bawane quality model [103] (2010)

This model is composed of

- 2 quality perspectives
- 11 quality characteristics
- 28 quality sub-characteristics

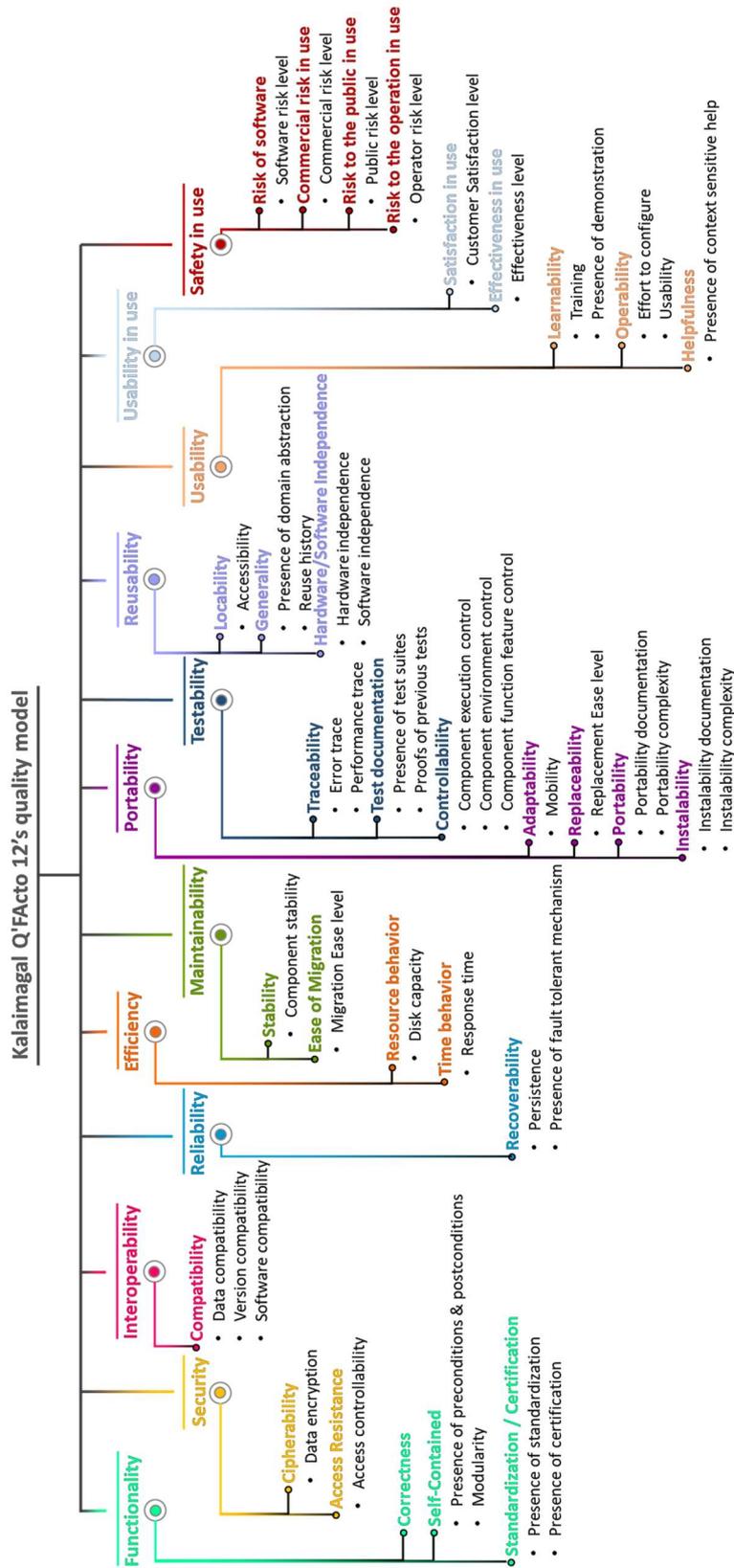


This model uses terminology from ISO/IEC 9126

- Kalaimagal's Q'Facto 12 quality model [102] (2010)

This model is composed of

- 12 quality factors
- 30 quality criteria
- 44 quality measures

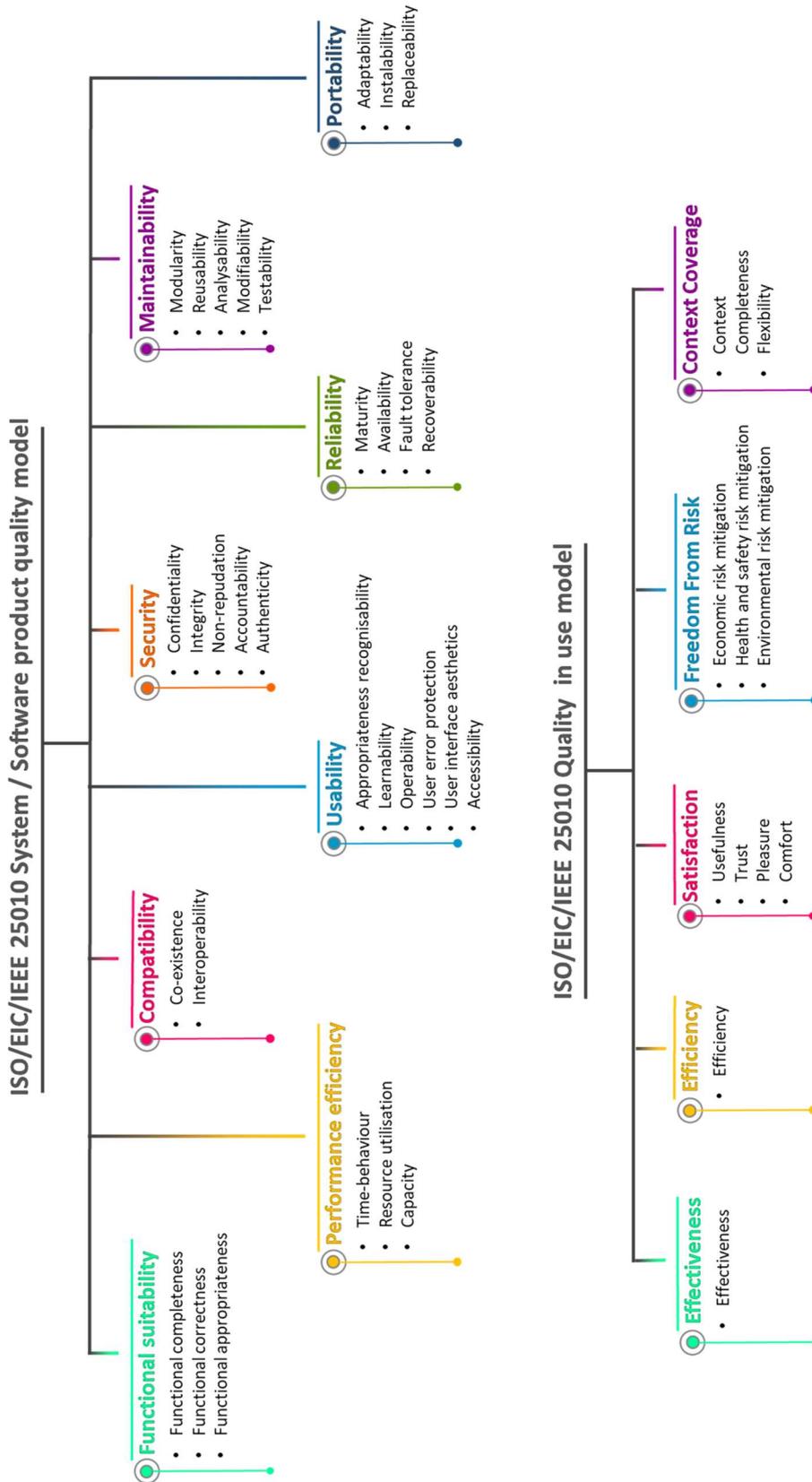


This model uses terminology from ISO/IEC/IEEE 25010

- ISO/IEC/IEEE 25010 quality model [23] (2011)

This model is composed of

- 2 quality perspectives
- 13 quality characteristics
- 42 quality sub-characteristics



Terminology	Definition
Accessibility	degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.
Accountability	degree to which the actions of an entity can be traced uniquely to the entity.
Adaptability	degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
Analyzability	degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
Appropriateness	degree to which users can recognize whether a product or system is appropriate for their needs.
Recognizability	
Authenticity	degree to which the identity of a subject or resource can be proved to be the one claimed.
Availability	degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
Capacity	degree to which the maximum limits of a product or system parameter meet requirements.
Co-existence	degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
Comfort	degree to which the user is satisfied with physical comfort.
Compatibility	degree to which a product, system or component can exchange information with other products, systems, or components, and/or perform its required functions, while sharing the same hardware or software environment.
Confidentiality	degree to which a product or system ensures that data are accessible only to those authorized to have access.
Context Completeness	degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use.
Context Coverage	degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.
Economic Risk Mitigation	degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use.
Effectiveness	accuracy and completeness with which users achieve specified goals.
Efficiency	resources expended in relation to the accuracy and completeness with which users achieve goals.
Environmental Risk Mitigation	degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment.
Fault Tolerance	degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
Flexibility	degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.
Freedom from Risk	degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment.
Functional Appropriateness	degree to which the functions facilitate the accomplishment of specified tasks and objectives.
Functional Completeness	degree to which the set of functions covers all the specified tasks and user objectives.
Functional Correctness	degree to which a product or system provides the correct results with the needed degree of precision.
Functional Suitability	degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.
Health and Safety Risk Mitigation	degree to which a product or system mitigates the potential risk to people in the intended contexts of use.
Installability	degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.
Integrity	degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.
Interoperability	degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Learnability	degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
Maintainability	degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.
Maturity	degree to which a system, product or component meets needs for reliability under normal operation.
Modifiability	degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
Modularity	degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
Non-Repudiation	degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.
Operability	degree to which a product or system has attributes that make it easy to operate and control.
Performance Efficiency	performance relative to the amount of resources used under stated conditions.
Pleasure	degree to which a user obtains pleasure from fulfilling their personal needs.
Portability	degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.
Reliability	degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.
Replaceability	degree to which a product can replace another specified software product for the same purpose in the same environment.
Resource Utilization	degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
Reusability	degree to which an asset can be used in more than one system, or in building other assets.
Satisfaction	degree to which user needs are satisfied when a product or system is used in a specified context of use.
Security	degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.
Testability	degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.
Time Behaviour	degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
Trust	degree to which a user or other stakeholder has confidence that a product or system will behave as intended.
Usability	degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.
Usefulness	degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use.
User Error Protection	degree to which a system protects users against making errors.
User Interface Aesthetics	degree to which a user interface enables pleasing and satisfying interaction for the user.

Annex 14. Variations of quality characteristic genes

These gene and site variations are determined by gathering the quality characteristics and sub-characteristics from the quality models identified in the first step of the meta-model construction. This regrouping is done based on lexical and semantic such as in Motogna *et al.* study [94]. An example around “Portability” gene is shown in TABLE 29 and TABLE 30. The complete gene and site enumeration with their related possible variations and corresponding likelihood is given by TABLE 50 and TABLE 51.

In TABLE 50, each gene is present only in one quality model. Thus, their sites contain only one possible variation in each gene context. Nevertheless, we considered that a gene can be part of another gene even resulting from several quality models and then seen also as a site. Consequently, that gene can have variations. For instance, “Changeability” gene is resulting from only one single quality model. However, as a site “Changeability” can be retrieved in “Maintainability” gene.

TABLE 50 - LIST OF THE 27 SINGLE-QUALITY MODEL GENES WITH THEIR POSSIBLE VARIATIONS

Gene ID	Type	Quality Characteristic Name	quality characteristic variations: name = probability (calculation detail)			
			Variation 1	Variation 2	Variation 3	Variation 4
A01	Gene	Adaptability	Adaptability = 100%			
	site 1	Mobility	Mobility = 100%			
	site 2	Configuration capacity	Configuration capacity = 100%			
A02	Gene	Changeability	Changeability = 50% (3/6)	Modifiability = 33.33% (2/6)	Ease of Migration = 16.67% (1/6)	
	site 1	Extensibility	Extensibility = 100%			
	site 2	Customizability	Customizability = 100%			
	site 3	Modularity	Modularity = 100%			
A03	Gene	Context coverage	Context coverage = 100%			
	site 1	context completeness	context completeness = 100%			
	site 2	Flexibility	Flexibility = 100%			
A04	Gene	Controllability	Controllability = 100%			
	site 1	Component execution control	Component execution control = 100%			
	site 2	Component environment control	Component environment control = 100%			
	site 3	Component function feature control	Component function feature control = 100%			
A05	Gene	Correctness	Correctness = 100%			
	site 1	Traceability	Traceability = 100%			
	site 2	Consistency	Consistency = 100%			
	site 3	Completeness	Completeness = 100%			
A06	Gene	Freedom from risk	Freedom from risk = 100%			
	site 1	Economic risk mitigation	Economic risk mitigation = 100%			
	site 2	Health and safety risk Mitigation	Health and safety risk mitigation = 100%			
	site 3	Environmental risk mitigation	Environmental risk mitigation = 100%			
A07	Gene	Flexibility	Flexibility = 100%			
	site 1	Modularity	Modularity = 100%			
	site 2	Generality	Generality = 100%			
	site 3	Expandability	Expandability = 100%			
	site 4	Self-Descriptiveness	Self-Descriptiveness = 100%			
A08	Gene	General utility	General utility = 100%			
	site 1	Portability	Portability = 100%			
	site 2	Usability (As-is utility)	Usability (As-is utility) = 100%			

	site 3	Maintainability	Maintainability = 100%			
A09	Gene	Generality	Generality = 100%			
	site 1	Presence of domain abstraction	Presence of domain abstraction = 100%			
	site 2	Reuse history	Reuse history = 100%			
A10	Gene	Hardware/Software Independence	Hardware/Software Independence = 100%			
	site 1	Hardware independence	Hardware independence = 100%			
	site 2	Software independence	Software independence = 100%			
A11	Gene	Human engineering	Human engineering = 25% (1/4)	Human Factors = 25% (1/4)	Communicativeness = 25% (1/4)	Accessibility = 25% (1/4)
	site 1	Robustness / Integrity	Robustness / Integrity = 100%			
	site 2	Accessibility	Accessibility = 100%			
	site 3	Communicativeness	Communicativeness = 100%			
A12	Gene	Instalability	Instalability = 80% (4/5)	Deployability = 20% (1/5)		
	site 1	Instalability documentation	Instalability documentation = 100%			
	site 2	Instalability complexity	Instalability complexity = 100%			
A13	Gene	Learnability	Learnability = 71.4% (5/7)	Training = 14.3% (1/7)	Material of Training = 14.3% (1/7)	
	site 1	Training	Training = 100%			
	site 2	Presence of demonstration	Presence of demonstration = 100%			
A14	Gene	Maturity	Maturity = 100%			
	site 1	Volatility	Volatility = 100%			
	site 2	Failure removal	Failure removal = 100%			
A15	Gene	Modifiability	Changeability = 50% (3/6)	Modifiability = 33.33% (2/6)	Ease of Migration = 16.67 (1/6)	
	site 1	Structuredness	Structuredness = 100%			
	site 2	Augmentability	Augmentability = 100%			
A16	Gene	Product operation	Product operation = 100%			
	site 1	Correctness	Correctness = 100%			
	site 2	Reliability	Reliability = 100%			
	site 3	Efficiency	Efficiency = 100%			
	site 4	Integrity (security)	Integrity (security) = 100%			
	site 5	Usability	Usability = 100%			
A17	Gene	Product revision	Product revision = 100%			
	site 1	Maintainability	Maintainability = 100%			
	site 2	Flexibility	Flexibility = 100%			
	site 3	Testability	Testability = 100%			
A18	Gene	Product transition	Product transition = 100%			
	site 1	Portability	Portability = 100%			
	site 2	Reusability	Reusability = 100%			
	site 3	Interoperability	Interoperability = 100%			
A19	Gene	Resource behavior	Resource utilization = 66.67% (4/6)	Resource behavior = 16.67% (1/6)	Utilization of resources = 16.67% (1/6)	
	site 1	Memory utilization	Memory utilization = 100%			
	site 2	Disk utilization	Disk utilization = 100%			
A20	Gene	Safety in use	Safety in use = 100%			
	site 1	Risk of software	Risk of software = 100%			
	site 2	Commercial risk in use	Commercial risk in use = 100%			
	site 3	Risk to the operation in use	Risk to the operation in use = 100%			
	site 4	Risk to the public in use	Risk to the public in use = 100%			

Annexes

A21	Gene	Satisfaction	Satisfaction = 75% (3/4)	Satisfaction in use = 25% (1/4)
	site 1	Usefulness	Usefulness = 100%	
	site 2	Trust	Trust = 100%	
	site 3	Pleasure	Pleasure = 100%	
	site 4	Comfort	Comfort = 100%	
A22	Gene	Self-Contained	Self-contained = 100%	
	site 1	Presence of precondition & postconditions	Presence of precondition & postconditions = 100%	
	site 2	Modularity	Modularity = 100%	
A23	Gene	Suitability	Suitability = 75% (3/4)	Functional appropriateness = 25% (1/4)
	site 1	Coverage	Coverage = 100%	
	site 2	Completeness	Completeness = 100%	
	site 3	Pre- and Post-conditioned	Pre- and Post-conditioned = 100%	
	site 4	Proofs of Pre- and Post-conditions	Proofs of Pre- and Post-conditions = 100%	
A24	Gene	Supportability	Supportability = 100%	
	site 1	Testability	Testability = 100%	
	site 2	Extensibility	Extensibility = 100%	
	site 3	Adaptability	Adaptability = 100%	
	site 4	Maintainability	Maintainability = 100%	
	site 5	Compatibility	Compatibility = 100%	
	site 6	Configurability	Configurability = 100%	
	site 7	Serviceability	Serviceability = 100%	
	site 8	Instability	Instability = 100%	
	site 9	Localizability	Localizability = 100%	
A25	Gene	Test documentation	Test documentation = 100%	
	site 1	Presence of test suites	Presence of test suites = 100%	
	site 2	Proofs of previous tests	Proofs of previous tests = 100%	
A26	Gene	Time behavior	Time behavior = 100%	
	site 1	Response time	Response time = 100%	
	site 2	Latency throughput ("out")	Latency throughput ("out") = 100%	
	site 3	Latency processing capacity ("in")	Latency processing capacity ("in") = 100%	
A27	Gene	Traceability	Traceability = 100%	
	site 1	Error trace	Error trace = 100%	
	site 2	Performance trace	Performance trace = 100%	

TABLE 51 - LIST OF THE 16 MULTI-QUALITY MODELS GENES WITH THEIR POSSIBLE VARIATIONS

Gene ID	Type	Quality Characteristic Name	quality characteristic variations: name = probability (calculation detail)			
			Variation 1	Variation 2	Variation 3	Variation 4
B01	Gene	Efficiency	Efficiency = 75% (6/8)	Performance = 12.5% (1/8)	Performance efficiency = 12.5% (1/8)	
	site 1	Time-behavior	Time-behavior = 83.33% (5/6)	Velocity = 16.67% (1/6)		
	site 2	Resource utilization	Resource utilization = 66.67% (4/6)	Resource behavior = 16.67% (1/6)	Utilization of resources = 16.67% (1/6)	
	site 3	Storage efficiency	Storage efficiency = 50% (1/2)	Resource behavior = 50% (1/2)		
	site 4	Execution efficiency	Execution efficiency = 50% (1/2)	Efficiency = 50% (1/2)		
	site 5	Efficiency Compliance	Efficiency Compliance = 50% (1/2)	Device efficiency = 50% (1/2)		
	site 6	Capacity	Capacity = 100%			

	site 7	Accountability	Accountability = 100%	
	site 8	Accessibility	Accessibility = 100%	
	site 9	Scalability	Scalability = 100%	
	site 10	Availability	Availability = 100%	
	site 11	Time of answers	Time of answers = 100%	
	site 12	Time of recovery	Time of recovery = 100%	
B02	Gene	Fault tolerance	Fault tolerance = 50% (1/2)	Recoverability = 50% (1/2)
	site 1	Mechanism availability	Mechanism availability = 50% (1/2)	Presence of fault tolerant mechanism = 50% (1/2)
	site 2	Mechanism efficiency	Mechanism efficiency = 100%	
	site 3	Persistence	Persistence = 100%	
B03	Gene	Functionality	Functionality = 83.33% (5/6)	Functional suitability = 16.67% (1/6)
	site 1	Accuracy	Accuracy = 60% (3/5)	Functional correctness = 20% Correctness = 20% (1/5) (1/5)
	site 2	Security	Security = 100%	
	site 3	Suitability	Suitability = 75% (3/4)	Functional appropriateness = 25% (1/4)
	site 4	Interoperability	Interoperability = 100%	
	site 5	Compliance	Compliance = 0.3333 (1/3)	Functionality Compliance = 0.3333 (1/3) Standardization / Certification = 0.3333 (1/3)
	site 6	Self-contained	Self-contained = 100%	
	site 7	Functional Completeness	Functional completeness = 50% (1/2)	Joint of Characteristics = 50% (1/2)
	site 8	Capacities	Capacities = 100%	
B04	Gene	Interoperability	Interoperability = 66.67% (2/3)	Compatibility = 16.67% (1/3)
	site 1	Data commonality	Data commonality = 50% (1/2)	Data compatibility = 50% (1/2)
	site 2	Communication commonality	Communication commonality = 50% (1/2)	Interoperability = 50% (1/2)
	site 3	Co-existence	Co-existence = 50% (1/2)	Software compatibility = 50% (1/2)
	site 4	Modularity	Modularity = 100%	
	site 5	Version compatibility	Version compatibility = 100%	
B05	Gene	Maintainability	Maintainability = 100%	
	site 1	Changeability	Changeability = 50% (3/6)	Modifiability = 33.33% (2/6) Ease of Migration = 16.67 (1/6)
	site 2	Testability	Testability = 100%	
	site 3	Analyzability	Analyzability = 60% (3/5)	Understandability = 20% (1/5) Simplicity = 20% (1/5)
	site 4	Stability	Stability = 100%	
	site 5	Modularity	Modularity = 100%	
	site 6	Reusability	Reusability = 100%	
	site 7	Consistency	Consistency = 100%	
	site 8	Conciseness	Conciseness = 100%	
	site 9	Self-Descriptiveness	Self-Descriptiveness = 100%	
	site 10	Maintainability Compliance	Maintainability Compliance = 100%	
B06	Gene	Operability	Operability = 100%	
	site 1	Effort to operate	Effort to operate = 100%	
	site 2	Complexity level	Complexity level = 100%	
	site 3	Provided Interfaces	Provided Interfaces = 100%	
	site 4	Required Interfaces	Required Interfaces = 100%	
	site 5	Effort to configure	Effort to configure = 100%	
B07	Gene	Portability	Portability = 100%	

Annexes

site 1	Adaptability	Adaptability = 100%			
site 2	Replaceability	Replaceability = 100%			
site 3	Instalability	Instalability = 80% (4/5)	Deployability = 20% (1/5)		
site 4	Conformance	Conformance = 50% (1/2)	Portability Compliance = 50% (1/2)		
site 5	Modularity	Modularity = 50% (1/2)	Self-containedness = 50% (1/2)		
site 6	Machine independence	Machine independence = 50% (1/2)	Device independence = 50% (1/2)		
site 7	Self-Descriptiveness	Self-Descriptiveness = 50% (1/2)	Portability documentation = 25% (1/2 * 50%)	Portability complexity = 25% (1/2 * 50%)	
site 8	Software system independence	Software system independence = 100%			
site 9	Co-existence	Co-existence = 100%			
site 10	Reusability	Reusability = 100%			
B08	Gene	Product in use	Product in use = 50% (2/4)	Usability in use = 25% (1/4)	Quality in use = 25% (1/4)
site 1		Effectiveness	Effectiveness = 75% (3/4)	Effectiveness in use = 25% (1/4)	
site 2		Satisfaction	Satisfaction = 75% (3/4)	Satisfaction in use = 25% (1/4)	
site 3		Productivity	Productivity = 66.67% (2/3)	Efficiency = 33.33% (1/3)	
site 4		Safety	Safety = 100%		
site 5		Freedom from risk	Freedom from risk = 100%		
site 6		Context coverage	Context coverage = 100%		
B09	Gene	Reliability	Reliability = 100%		
site 1		Fault tolerance	Fault tolerance = 57.1% (4/7)	Error Tolerance = 14.3% (1/7)	Recovery to failures = 14.3% (1/7)
site 2		Recoverability	Recoverability = 100%		Robustness / Integrity = 14.3% (1/7)
site 3		Maturity	Maturity = 100%		
site 4		Accuracy	Accuracy = 100%		
site 5		Consistency	Consistency = 100%		
site 6		Availability	Availability = 50% (1/2)	Time among failures = 50% (1/2)	
site 7		Reliability Compliance	Reliability Compliance = 50% (1/2)	Completeness = 50% (1/2)	
site 8		Simplicity	Simplicity = 50% (1/2)	Self-containedness = 50% (1/2)	
site 9		Frequency and severity of failures	Frequency and severity = 100%		
B10	Gene	Reusability	Reusability = 100%		
site 1		Self-Descriptiveness	Self-Descriptiveness = 33.33% (1/3)	Simplicity = 33.33% (1/3)	Comprehensibility = 33.33% (1/3)
site 2		Generality	Generality = 100%		
site 3		Modularity	Modularity = 100%		
site 4		Software independence	Software independence = 50% (1/2)	Software system independence = 50% (1/2)	
site 5		Hardware independence	Hardware independence = 50% (1/2)	Machine independence = 50% (1/2)	
site 6		Coupling	Coupling = 100%		
site 7		Domain abstraction level	Domain abstraction level = 100%		
site 8		Architecture compatibility	Architecture compatibility = 100%		
site 9		Cohesion	Cohesion = 100%		
site 10		Locability	Locability = 100%		
site 11		Interoperability	Interoperability = 100%		
B11	Gene	Security	Security = 75% (3/4)	Integrity (security) = 25% (1/4)	
site 1		Confidentiality	Confidentiality = 25% (1/4)	Access control = 25% (1/4)	Controllability = 25% (1/4)
site 2		Auditability	Auditability = 50% (1/2)	Access Audit = 50% (1/2)	Access Resistance = 25% (1/4)

	site 3	Cipherability	Cipherability = 50% (1/2)	Data Encryption = 50% (1/2)
	site 4	Integrity	Integrity = 100%	
	site 5	Non-repudiation	Non-repudiation = 100%	
	site 6	Accountability	Accountability = 100%	
	site 7	Authenticity	Authenticity = 100%	
B12	Gene	Software product	Software product = 66.67% (2/3)	System / Software product = 33.33% (1/3)
	site 1	Functionality	Functionality = 66.67% (2/3)	Functional suitability = 33.33% (1/3)
	site 2	Usability	Usability = 100%	
	site 3	Reliability	Reliability = 100%	
	site 4	Efficiency	Efficiency = 66.67% (2/3)	Performance efficiency = 33.33% (1/3)
	site 5	Maintainability	Maintainability = 100%	
	site 6	Portability	Portability = 100%	
	site 7	Compatibility	Compatibility = 100%	
	site 8	Security	Security = 100%	
	site 9	Reusability	Reusability = 100%	
B13	Gene	Compliance	Compliance = 50% (1/2)	Standardization / Certification = 50% (1/2)
	site 1	Standardization	Standardization = 50% (1/2)	Presence of standardization = 50% (1/2)
	site 2	Certification	Certification = 50% (1/2)	Presence of certification = 50% (1/2)
B14	Gene	Testability	Testability = 100%	
	site 1	Self-descriptiveness	Self-descriptiveness = 100%	
	site 2	Accountability	Accountability = 100%	
	site 3	Accessibility	Accessibility = 100%	
	site 4	Communicativeness	Communicativeness = 100%	
	site 5	Structuredness	Structuredness = 100%	
	site 6	Simplicity	Simplicity = 100%	
	site 7	Modularity	Modularity = 100%	
	site 8	Instrumentation	Instrumentation = 100%	
	site 9	Test suite provided	Test suite provided = 100%	
	site 10	Extensive component test case	Extensive component test case = 100%	
	site 11	Component tests in a specific environment	Component tests in a specific environment = 100%	
	site 12	Proofs the components tests	Proofs the components tests = 100%	
	site 13	Test documentation	Test documentation = 100%	
	site 14	Controllability	Controllability = 100%	
	site 15	Traceability	Traceability = 100%	
B15	Gene	Understandability	understandability = 100%	
	site 1	Self-descriptiveness	Self-descriptiveness = 50% (1/2)	Code Readability = 50% (1/2)
	site 2	Consistency	Consistency = 100%	
	site 3	Structuredness	Structuredness = 100%	
	site 4	Conciseness	Conciseness = 100%	
	site 5	Legibility	Legibility = 100%	
	site 6	Documentation availability	Documentation availability = 100%	
	site 7	Documentation readability and quality	Documentation readability and quality = 100%	
B16	Gene	Usability	Usability = 87.5% (7/8)	Usability (As-is utility) = 12.5% (1/8)
	site 1	Learnability	Learnability = 71.4% (5/7)	Training = 14.3% (1/7) Material of Training = 14.3% (1/7)

Annexes

site 2	Operability	Operability = 100%			
site 3	Understandability	understandability = 50% (3/6)	Appropriateness recognizability = 16.67% (1/6)	Helpfulness = 16.67% (1/6)	Documentation of the user = 16.67% (1/6)
site 4	Attractiveness	Attractiveness = 50% (2/4)	Aesthetic = 25% (1/4)	User interface aesthetics = 25% (1/4)	
site 5	Human engineering	Human engineering = 25% (1/4)	Human Factors = 25% (1/4)	Communicativeness = 25% (1/4)	Accessibility = 25% (1/4)
site 6	Reliability	Reliability = 50% (1/2)	Accessibility = 50% (1/2)		
site 7	Usability compliance	Usability compliance = 100%			
site 8	Efficiency	Efficiency = 100%			
site 9	Configurability	Configurability = 100%			

Chapter X. Synthèse de la Thèse en Français

L'objectif de la thèse nous a conduit à définir un cadre théorique pour la supervision et le pilotage des processus d'ingénierie et de développement de produits par la qualité.

Ce travail a donné lieu à de multiples contributions, partiellement valorisées sous la forme de deux articles présentés en conférences internationales (cf. Argotti et al. [167], [230]). Sa contribution majeure au corpus des connaissances scientifique est l'étude exhaustive et approfondie des modèles qualité existants dans la littérature prérequis incontournable pour aller plus loin dans des propositions conceptuelles et méthodologiques consolidées.

1. Synthèse Générale

Dans le Chapitre I, nous posons la problématique de recherche de thèse, « *Etude des éléments essentiels de la Qualimétrie appliquée au développement de logiciels embarqués* », que nous avons reformulée en « *renforcer et unifier la définition, l'évaluation, le contrôle ou la prédiction de la qualité des logiciels embarqués* », au regard de notre contexte industriel. Nous avons entamé son analyse en identifiant les quatre questions de recherche suivantes :

Question de recherche 1	La Qualimétrie, en tant que science de la quantification de la qualité, est-elle la bonne approche et quels sont les éléments essentiels de la qualité et de la Qualimétrie ?
Question de recherche 2	Considérant l'ensemble des modèles qualité pour le logiciel, comment identifier et décider quel modèle qualité est le plus approprié pour le logiciel embarqué ?
Question de recherche 3	Comment opérationnaliser un modèle qualité pour un produit logiciel ?
Question de recherche 4	Peut-on avoir un modèle qualité de référence unique pour les produits logiciels ?

Nous avons ensuite approfondi cette analyse dans le Chapitre II. Nous avons remarqué que le contexte industriel automobile, conjointement avec le véhicule en tant que système complexe, le modèle de développement avec les fournisseurs et les exigences actuelles en matière de normes et de réglementations, augmentent la complexité globale de notre problématique. Par conséquent, dans ce contexte, il est essentiel de disposer d'une méthode unifiée, opérationnelle et appropriée pour définir, évaluer, contrôler ou prévoir la qualité des logiciels embarqués.

Afin de vérifier si une telle solution unifiée, opérationnelle et appropriée pour la qualité des logiciels embarqués existe déjà, nous avons effectué une analyse documentaire exploratoire sur « *la manière dont la modélisation de la qualité est appliquée aux logiciels embarqués* ». Nous avons constaté l'existence d'une myriade de systèmes et de logiciels embarqués possibles, chacun d'eux ayant ses propres spécificités, caractéristiques de qualité et éventuellement une diversité de modèles qualité. Il est donc apparu qu'il n'y avait pas encore de solution juste et unique à notre question.

En conséquence, nous avons affiné ces quatre questions de recherche en 15 sous-questions de recherche, puis nous avons détaillé notre méthodologie de recherche au Chapitre III. Dans ce chapitre, également, nous avons expliqué le réalignement de notre méthodologie de recherche en soulignant non seulement les difficultés dans la sélection d'un modèle qualité approprié pour les logiciels intégrés, mais aussi les conséquences d'une telle sélection en écartant de nombreuses contributions précieuses.

Ensuite, nous avons abordé ces questions de recherche dans les Chapitres de IV à VIII.

Question de recherche 1	La Qualimétrie, en tant que science de la quantification de la qualité, est-elle la bonne approche et quels sont les éléments essentiels de la qualité et de la Qualimétrie ?
--------------------------------	---

Ainsi, dans le Chapitre IV, nous avons exploré l'essence de la qualité (i.e., sous-question de recherche **1a**), et la modélisation de la qualité en particulier dans le domaine des logiciels (i.e., sous-question de recherche **1b**). Nous

avons défini, clarifié ces connaissances et les concepts associés (par exemple, la qualité perçue, les perspectives de qualité, les dimensions et les caractéristiques de la qualité, le modèle qualité, les mesures, l'échelle), et conclu cette exploration par la construction de la première chronologie des contributions clés à la modélisation de la qualité des logiciels, allant de 1965 avec la première apparition du concept de génie logiciel [32] à 2015 avec Azgaldov *et al.* et l'ABC de la Qualimétrie [113]. Ensuite, nous avons cherché la Qualimétrie, reconnaissant que nous avons une bonne compréhension de la jeune science de la quantification de la qualité, avant de démontrer que la bonne approche pour nos besoins est la Qualimétrie (i.e., sous-question de recherche **1c**).

Au cours de cette enquête, nous avons constaté que la Qualimétrie était souvent mal comprise et nous commençons donc à contribuer à cette science en la vulgarisant, en résumant ses concepts majeurs sous une vue synthétique : la « *Maison de la Qualimétrie* » et ses 6 piliers. Nous avons également remarqué qu'il était possible d'unifier la diversité et l'évolution temporelle dans la modélisation de la qualité (i.e., sous-question de recherche **1d**) en trouvant notre inspiration dans la génétique, et donc en introduisant le concept de polymorphisme (c'est-à-dire le polymorphisme ad hoc, universel et temporel) dans la modélisation de la qualité. Pour compléter cette contribution, nous avons proposé et prouvé que l'utilisation d'une formule basée sur la diversité génétique [86] était plus appropriée pour comparer des modèles qualité ensemble (i.e., sous-question de recherche **2c**) que la distance de Hamming, par exemple, et nous avons proposé un nouveau processus de mesure cadencé avec le cycle de vie du système et du logiciel pour intégrer le polymorphisme temporel.

Question de recherche 2

Considérant l'ensemble des modèles qualité pour le logiciel, comment identifier et décider quel modèle qualité est le plus approprié pour le logiciel embarqué ?

L'objectif du Chapitre V était de plonger en profondeur dans la littérature pour récupérer les modèles qualité des logiciels existants, puis de déterminer quel modèle qualité pourrait être sélectionné pour répondre à nos besoins avec les logiciels intégrés (i.e., question de recherche **2**). Pour mener à bien cette entreprise, nous avons procédé à un examen systématique de la littérature, dans le cadre duquel nous avons identifié et analysé 136 documents d'étude publiés au cours d'une période allant de 1979 à 2019. Le résultat de cette revue, combiné à l'approche en « *boule de neige* », telle que décrite par Wohlin [215], [216] et qui consiste à exploiter chaque document référencé comme source supplémentaire de documents d'étude, a été la récupération de 492 modèles qualité de logiciels de 1968 à 2019 (i.e., sous-question de recherche **2a**). Cette liste de modèles qualité des logiciels est une contribution unique puisqu'elle représente une collection dix fois supérieure au maximum que nous avons trouvé dans les articles publiés : Oriol *et al* [12] ont énuméré 48 modèles qualité liés aux services web. A Noter que dans Kläs *et al* [97], les auteurs affirment avoir fourni une classification pour environ 80 modèles qualité, mais nous n'avons pas réussi à retrouver cette liste de modèles qualité, même dans les articles référencés de cette étude ou dans la publication des auteurs.

Fort des résultats de la revue systématique de la littérature et parlant de classification, notre contribution suivante a été de proposer l'utilisation de la cladistique comme méthode de classification pour les modèles qualité des logiciels (i.e., la sous-question de recherche **2b**). Pour cette raison, le schéma de classification a été constitué de 20 éléments de classification des modèles qualité des logiciels organisés en cinq thèmes (i.e., id, bibliographique, définition, portée et structure), puis décliné en cladistique des modèles qualité des logiciels : homologie (i.e., similarité liée à une ascendance commune) et taxons (i.e., entités conceptuelles).

Bien que nous ayons commencé à utiliser un sous-ensemble de ces taxons pour classer ces 492 modèles qualité, ils ont suffi pour réussir à représenter un paysage de modèles qualité logicielle. Nous avons constaté que ces modèles ont été conçus principalement pour l'évaluation de la qualité, puis pour la prédiction, ils sont généralement hiérarchisés, sauf pour la prédiction, où le formalisme statistique ou implicite est mieux adapté, avec un champ d'application souvent mis sur le produit, et une perspective de qualité également répartie entre les perspectives du fabricant, de l'utilisateur et du produit. En outre, notre contribution sur le paysage des modèles qualité des logiciels a rectifié le postulat de Thapar *et al.* [11] concernant l'évolution des modèles qualité (c'est-à-dire les modèles qualité de base avant 2000 et les modèles qualité adaptés depuis 2000). En effet, nous avons montré que cette évolution s'articule autour de trois périodes : jusqu'en 1990, nous avons la période des modèles qualité de base, de 1990 à 2003, la période de transition, et depuis 2003, nous sommes dans la période de personnalisation des modèles qualité.

Enfin, la conclusion de ce chapitre a confirmé l'inadéquation d'un modèle qualité de référence unique couvrant tous les cas de produits logiciels (i.e., sous-question de recherche **4a**), et a suggéré la sélection, ainsi que la personnalisation, de la dernière norme de modèle qualité, ISO / IEC / IEEE 25010, pour générer un modèle approprié pour les logiciels embarqués dans le domaine automobile (i.e., sous-question de recherche **2d**)

Question de recherche 3

Comment opérationnaliser un modèle qualité pour un produit logiciel ?

Au Chapitre VI, notre objectif était d'étudier le passage de la théorie du modèle qualité à la pratique, et plus particulièrement l'opérationnalisation du modèle qualité (i.e., question de recherche **3**). Cet aspect opérationnel est essentiel pour développer et déployer le modèle qualité par rapport à un cas réel d'utilisation de mots, ou pour réussir à reproduire et à tirer profit des études de modèles qualité.

Ainsi, au cours de notre étude, nous avons identifié une liste de 16 défis ou problèmes distincts qui empêchent le développement et l'utilisation de modèles qualité des logiciels (i.e., sous-question de recherche **3a**), et nous avons ensuite réussi à associer des solutions pratiques (i.e., des solutions liées à des expériences, des situations réelles ou des actions qu'il est possible de reproduire, réutiliser ou déployer) à chacun de ces 16 défis (i.e., sous-question de recherche **3b**).

La synthèse consolidée de l'identification et de la résolution de ces problèmes a été réalisée grâce à la proposition de deux processus complémentaires (i.e., sous-question de recherche **3c**) :

- Le processus « en 6 étapes » se concentre sur le développement opérationnel du modèle qualité, avec un algorithme d'analyse basé sur des enquêtes, les Kappa de Fleiss et de Cohen ; cet algorithme, utilisé pour la construction du modèle qualité, prend en compte la contrainte, le point de vue des parties prenantes et permet de déterminer automatiquement les facteurs de pondération des caractéristiques et sous-caractéristiques qualité.
- Le processus « *Thermomètre de la Qualité* » se concentre sur l'utilisation opérationnelle du modèle qualité ; il comprend donc le processus « en 6 étapes », car l'une des premières étapes de l'utilisation du modèle qualité concerne le développement du modèle qualité.

Les parties innovantes de ces deux contributions au processus sont l'encapsulation transparente des solutions pratiques et l'utilisation du concept de polymorphisme.

Outre les réflexions et les propositions relatives à la transition de la théorie à la pratique, le Chapitre VII reflète la mise en pratique de nos conclusions et contributions par rapport à notre cas d'utilisation réel : les logiciels intégrés pour l'industrie automobile (i.e., la sous-question de recherche **3d**).

Nous avons donc décidé de les appliquer à un sous-ensemble de logiciels embarqués pour véhicules (i.e., trois unités de contrôle électronique - IVI, IVC, ADAS - avec leur propre logiciel embarqué et une fonctionnalité logicielle embarquée transversale - FOTA). Le résultat a été la création de trois modèles qualité polymorphes distincts avec leurs facteurs de poids respectifs. Nous avons remarqué dans le résultat l'existence de deux niveaux d'héritage du modèle qualité polymorphe, et un modèle qualité commun pour ADAS et FOTA. En outre, toutes les étapes de construction, détaillées dans ce chapitre, peuvent servir de lignes directrices éprouvées pour effectuer une modélisation de la qualité par rapport à n'importe quel logiciel ou système.

Enfin, ce succès dans le développement opérationnel d'un modèle qualité pour un cas réel d'utilisation de mots dans le domaine automobile, nous a non seulement permis de répondre aux besoins de l'entreprise, mais a également démontré le bien-fondé et la pertinence de nos conclusions et contributions.

Question de recherche 4

Peut-on avoir un modèle qualité de référence unique pour les produits logiciels ?

En tant que chapitre subsidiaire puisque nous avons déjà répondu à la demande de la société, l'objectif du Chapitre VIII était d'aller plus loin dans l'exploration d'un modèle de référence de la qualité des logiciels. En effet, le Chapitre V conclut sur l'inadéquation d'un modèle qualité de référence unique couvrant tous les cas de produits logiciels et, en conséquence, nous avons pu élaborer plutôt un méta-modèle qualité, rassemblant les

connaissances des modèles qualité existants, qui pourrait être utilisé comme base pour développer un nouveau modèle qualité.

En continuant avec la génétique, nous avons constaté qu'un certain niveau d'analogie pouvait être atteint entre les séquences d'ADN et les caractéristiques de qualité et la séquence des sous-caractéristiques. En outre, comme dans les séquences d'ADN, des variations des caractéristiques de qualité et des sous-caractéristiques peuvent exister avec un certain niveau de probabilité, ce qui rappelle le concept de polymorphisme. Ainsi, la base de notre contribution au méta-modèle qualité reposait sur cette analogie qui a également été reprise dans une ontologie du méta-modèle.

Après la conception détaillée de l'algorithme de construction du méta-modèle (i.e., sous-question de recherche **4b**), nous avons sélectionné un ensemble de modèles qualité de logiciels existants pour lancer la création de la première version du méta-modèle (i.e., sous-question de recherche **4c**). Le résultat de cette contribution unique et finale est le génome de la qualité logicielle composé de 7 chromatides : utilité générale, fonctionnement du produit, révision du produit, transition du produit, supportabilité, produit en cours d'utilisation et produit logiciel.

Les travaux de recherche et les réalisations de la thèse sont également résumés dans la synthèse globale réalisée à travers la Figure 103.

2. Perspectives de recherche

Cette étude complète et approfondie sur les modèles qualité des logiciels est le début d'un voyage passionnant mais trépidant dans le domaine de la Qualimétrie. Les perspectives de recherche qui en résultent éclairent les premières directions que ce voyage devrait prendre. Elles sont au nombre de trois, à savoir : la valorisation, la consolidation et l'exploration.

- **Perspectives de valorisation** : l'intention derrière ces perspectives est non seulement de partager plus largement nos résultats et contributions de recherche avec la communauté académique et industrielle, mais aussi d'améliorer notre proposition par le retour d'expérience et la mesure de l'efficacité, ainsi que favoriser son appropriation et adoption, à travers une stratégie de déploiement en entreprise accompagné par de la formation et un outillage. Ce type de perspective se situe principalement entre le court et le moyen terme.

Ainsi, une façon de réaliser le partage de l'information est de s'appuyer sur les médias littéraires. Nous avons prévu de rassembler les résultats et contributions suivants dans plusieurs documents de recherche :

- o La revue systématique de la littérature avec un « *effet boule de neige* » aboutissant à une liste unique de 492 modèles qualité des logiciels,
- o La classification de modèle qualité des logiciels basée sur la cladistique,
- o Le paysage du modèle qualité du logiciel et la correction du postulat de Thapar *et al*,
- o De la théorie à l'analyse de la pratique : 16 défis d'opérationnalisation et leurs solutions pratiques,
- o Les processus de développement et d'utilisation de modèle qualité, résumant des solutions pratiques et illustrées par un exemple tiré du secteur automobile,
- o Le modèle qualité polymorphe en pratique,
- o Le méta-modèle du génome du modèle qualité logicielle, y compris l'algorithme de construction, et les 7 chromatides du premier résultat du méta-modèle.

En outre, et comme nous l'avons déjà évoqué dans le Chapitre V.5, un outil de portail en ligne doit être créé pour diffuser le partage de la collection de 492 modèles qualité des logiciels et permettre la collaboration pour leur utilisation, leur achèvement et leur maintenance. L'objectif est de permettre à la communauté universitaire et industrielle de collaborer sur cette collection et d'éviter que cette liste ne devienne obsolète dans les années à venir.

Une autre façon de valoriser les résultats de nos recherches est d'industrialiser, de mettre à l'échelle et de déployer les contributions de la thèse par rapport à des systèmes de production réels. Cependant, si nous mesurons la maturité technologique de nos réalisations de thèse en utilisant l'échelle du niveau de préparation technologique (TRL) [261], nous atteignons actuellement le niveau 4, c'est-à-dire que la technologie a été validée en laboratoire, tandis que l'industrialisation, la mise à l'échelle et le déploiement

signifient un niveau 9. Nous espérons pouvoir utiliser le TRL pour nous guider sur la voie de la préparation technologique.

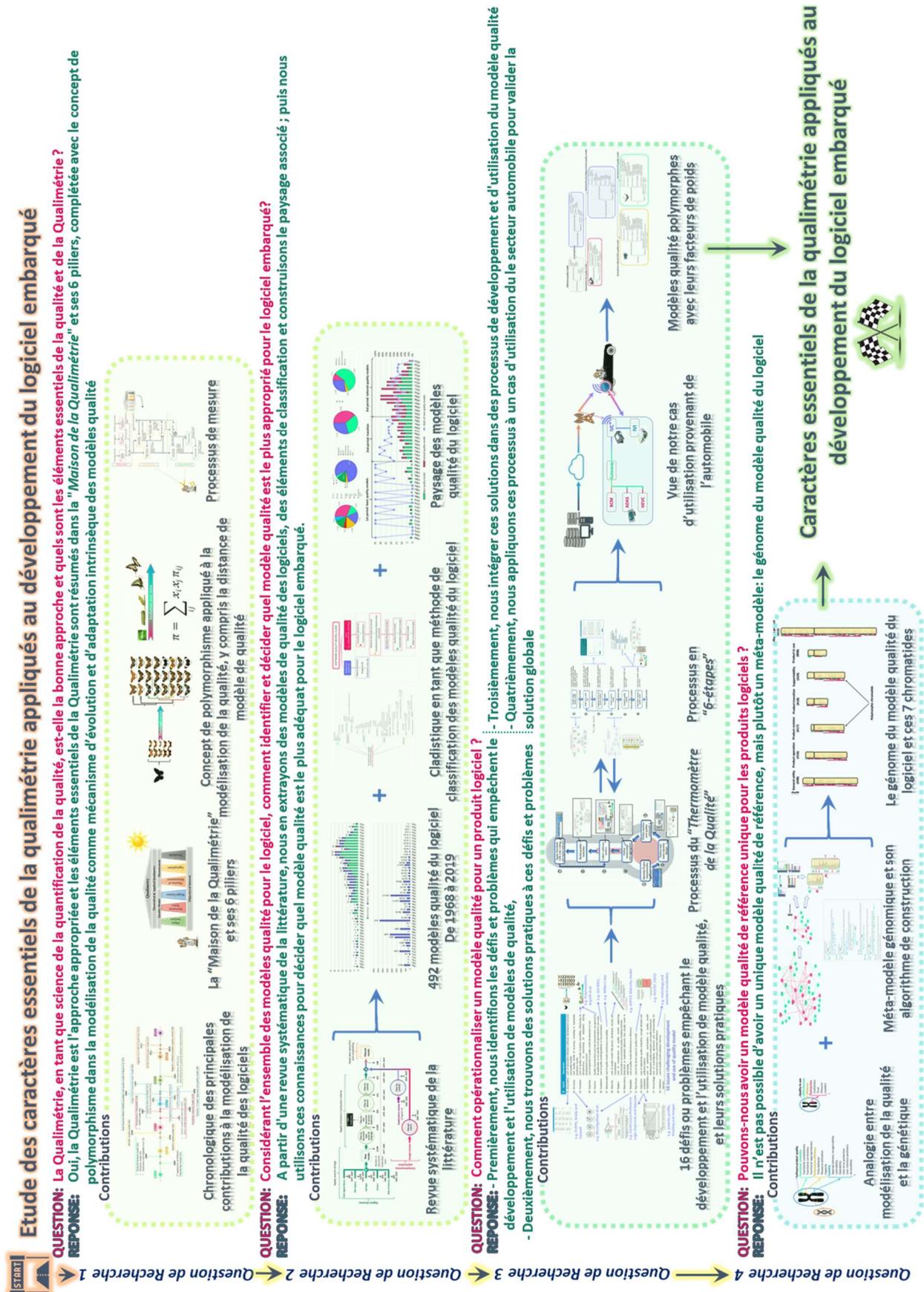


Figure 103 - Synthèse générale des travaux de recherche et des réalisations de la thèse

- **Perspectives de consolidation** : ce deuxième type de perspectives de recherche nous amène à poursuivre la consolidation de nos résultats et contributions actuels, tant du point de vue de la recherche que du développement. Ce type de perspective de recherche s'inscrit au moins à moyen terme. Les paragraphes suivants décrivent brièvement les principales orientations de la recherche et du développement.

En ce qui concerne la collecte et la classification des modèles qualité, la consolidation signifie que nous devons créer l'outil et le modèle de données adéquats pour recueillir, stocker et classer correctement au moins les 492 modèles qualité trouvés. Ensuite, nous devrions être en mesure d'effectuer une classification complète basée sur la cladistique et de tirer tous les bénéfices de ces contributions connexes.

Pour le méta-modèle, cette perspective de recherche indique que nous devons d'abord achever la mise en place d'un outil pour la construction automatique du méta-modèle, et ensuite, intégrer davantage de modèles qualité dans le méta-modèle au cours de sa construction. L'un des résultats attendus liés à l'intégration d'un plus grand nombre de modèles qualité dans le méta-modèle est de renforcer la convergence des caractéristiques et sous-caractéristiques de qualité les plus pertinentes et les plus importantes. Une autre amélioration du méta-modèle sera l'intégration de mesures.

Au sujet de la perspective de consolidation du développement du modèle qualité, nous visons à utiliser le méta-modèle comme modèle qualité référencé, à encourager la réutilisation du modèle qualité par le biais du polymorphisme, et à construire un outil pour automatiser le processus « *en 6-étapes* » pour le développement du modèle qualité, y compris la prise en compte des métriques. En parallèle, la modélisation pratique de la qualité pour un système complexe entier tel qu'un véhicule entier devrait être traitée.

De même, pour consolider l'utilisation du modèle qualité avec le processus « *Thermomètre de Qualité* », l'aspect outillage doit être abordé et devrait couvrir au moins : l'utilisation transparente du modèle qualité polymorphe, le déploiement opérationnel automatisé et l'exécution des modèles qualité et de leurs métriques, le tableau de bord en ligne et la carte de pointage avec capacité d'exploration de données pour permettre la prédiction et la prescription.

- **Perspectives d'exploration** : les objectifs de ces perspectives de recherche sont d'explorer et d'étudier des problèmes ouverts liés à la Qualimétrie, au modèle qualité ou à la modélisation. Par conséquent, la base temporelle correspondante pour ce travail de recherche est à long terme. Un certain nombre de problèmes ouverts ont déjà été saisis.

Le premier problème concerne l'évaluation, ou la prévision, de la valeur apportée par le développement et l'utilisation de modèles qualité. C'est une question récurrente qui vient souvent des chefs d'entreprise pour accepter le coût de l'activité de Qualimetry. Cependant, cette question reste sans réponse malgré les quelques études de recherche telles que Khoshgoftaar *et al.* [173] qui ont lancé la construction d'un modèle coûts-avantages de l'activité de modélisation de la qualité basé sur l'hypothèse de la réutilisation du modèle qualité sur plusieurs versions de logiciels, Porta [262] avec une enquête sur le modèle d'analyse coûts-avantages pour l'assurance qualité, ou la qualité intégrale composée de la qualité et de la rentabilité (voir Chapitre IV).

Un deuxième type de problème concerne la définition formelle et la généralisation de seuils pour évaluer, contrôler ou prédire objectivement qu'un niveau de qualité d'un produit, par exemple, est bon. Malheureusement, nous n'avons généralement pas de seuil d'acceptation, de référence ou d'objectif universel (c'est-à-dire convenu d'un commun accord). Une façon de contourner ce problème consiste à définir une cible, ou un seuil d'acceptation, sur la base de résultats antérieurs obtenus avec un produit identique, comme une version précédente du logiciel. Ainsi, on élimine le problème en ne considérant que les progrès par rapport aux réalisations antérieures. Néanmoins, le problème initial reste intact même s'il y a peu de tentatives industrielles mineures comme les 15 plages d'acceptation des mesures du code source du HIS automobile [263].

La troisième catégorie de problèmes ici est la généralisation de la modélisation de la trajectoire de qualité et de sa vitesse. Un problème parallèle concerne la discontinuité qui peut exister entre le modèle d'évaluation et le modèle de prédiction avec une portée et une perspective de qualité identiques mais avec des formalismes de modèle distincts.

Enfin, la dernière catégorie de problèmes englobe la modélisation du domaine d'intérêt contemporain de la qualité et où certaines études de recherche ont été lancées mais n'ont pas encore résolu le problème. Nous pouvons citer par exemple :

- L'écologie et la durabilité des logiciels,
- Le vieillissement et l'obsolescence des logiciels,
- Les données de qualité pour les systèmes et services logiciels connectés.