A QUALITY EVALUATION META-MODEL FOR OPEN SOURCE SOFTWARE

AÇIK KAYNAK YAZILIMLAR İÇİN KALİTE DEĞERLENDİRME ÜST MODELİ

NEBİ YILMAZ

ASSOC. PROF. DR. AYÇA KOLUKISA TARHAN

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a partial Fulfillment to the Requirements

for the Award of Degree of Doctor of Philosophy

in Computer Engineering

June 2023

ABSTRACT

A QUALITY EVALUATION META-MODEL FOR OPEN SOURCE SOFTWARE

Nebi YILMAZ

Doctor of Philosophy, Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ayça KOLUKISA TARHAN

June 2023, 224 pages

In recent years, Open Source Software (OSS) has gained an increasing attention due to its voluntary supporters, growing community, no vendor lock-in, low total cost of ownership, and ease of accessibility in cloud repositories. In turn, specifying and evaluating OSS quality has become a significant challenge for OSS adoption in organizations that are inclined to use them. Although many OSS quality models (OSS-QMs) have been proposed in literature, the dynamic and diverse nature of OSS has caused these models to be heterogeneous in terms of structure and content. This has adversely affected the standardization of evaluations and led to the evaluation results obtained from different OSS-QMs for the same purpose to be incomparable and sometimes unreliable. Standardization in OSS quality is of vital importance as a communication vehicle for stakeholders in identifying and selecting high-quality products. In this context, meta-modeling can help to define a standardized language and enable to propose quality models with comparable measurements. Therefore, in this thesis, a meta-model for OSS quality (OSS-QMM), which employs a unified structure from existing OSS-QMs and enables the derivation of homogeneous models, has been proposed. For this purpose, a systematic

and laborious effort has been spent via step-based meta-model creation process, including review-and-revise iterations. In order to validate the OSS-QMM, case study and expert opinion methods have been applied to answer three research questions (RQs) targeted to investigate results comparability, effectiveness, and practical applicability of using the meta-model. Multiple and embedded case study design has been employed for evaluating three real ERP systems, and 20 subject matter experts have been interviewed during the validation process. The results of multi-faceted empirical studies have indicated that the OSS-QMM have addressed solving problems in the OSS quality evaluation and its adoption with high degrees of confidence.

Keywords: Software quality, Quality model, Quality evaluation, Quality measurement, Meta-model, Open source software

ÖZET

AÇIK KAYNAK YAZILIMLAR İÇİN KALİTE DEĞERLENDİRME ÜST MODELİ

Nebi YILMAZ

Doktora, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Doç. Dr. Ayça KOLUKISA TARHAN

Haziran 2023, 224 sayfa

Son yıllarda Açık Kaynak Yazılım (AKY); gönüllü destekçileri, büyüyen topluluğu, satıcı firmaya bağımlılığının olmaması, toplam sahip olma maliyetinin düşüklüğü ve bulut depolarında kolayca erişilebilirliği nedeniyle artan bir ilgi görmüştür. Buna karşılık, AKY kalitesinin belirlenmesi ve değerlendirilmesi, bu yazılımları kullanmaya istekli olan kuruluşlarda AKY'nin benimsenmesi için önemli bir zorluk haline gelmiştir. Literatürde birçok AKY kalite modeli önerilmiş olsa da AKY'nin dinamik ve çeşitli doğası bu modellerin yapı ve içerik açısından heterojen olmasına neden olmuştur. Bu durum değerlendirmelerdeki standartlaşmayı olumsuz etkilemiş ve aynı amaç için farklı AKY kalite modellerinden elde edilen değerlendirme sonuçlarının karşılaştırılamaz ve bazen de güvenilmez olmasına yol açmıştır. AKY kalitesinde standartlaşma, yüksek kaliteli ürünlerin belirlenmesi ve seçilmesinde paydaşlar için bir iletişim aracı olarak hayati önem taşımaktadır. Bu bağlamda üst-modelleme, standartlaştırılmış bir dilin tanımlanmasına

yardımcı olabilir ve karşılaştırılabilen ölçümler üreten kalite modellerinin önerilmesini sağlayabilir. Bu nedenle, bu tez çalışmasında, mevcut AKY kalite modellerinden birleşik bir yapı kullanan ve homojen modellerin türetilmesini sağlayan; AKY kalitesi için bir üst-model önerilmiştir. Bu amaçla, gözden geçirme ve revize etme yinelemelerini içeren, adım-tabanlı üst-model oluşturma süreci takip edilerek sistematik ve zahmetli bir yol izlenmiştir. Üst-modeli doğrulamak için vaka çalışması ve uzman görüşü yöntemleri; değerlendirme sonuçlarının karşılaştırılabilir olduğunu, üst-modelin etkililiğini ve pratikte uygulanabilirliğini incelemeyi hedefleyen üç araştırma sorusunu yanıtlamak için uygulanmıştır. Üç gerçek ERP sistemini değerlendirmek için çoklu ve gömülü vaka çalışması tasarımı kullanılmış ve doğrulama sürecinde 20 konu uzmanıyla görüşülmüştür. Çok yönlü deneysel çalışmaların sonuçları, önerilen üst-modelin AKY kalitesini değerlendirmedeki ve benimsemedeki sorunları yüksek oranda çözdüğünü göstermiştir.

Anahtar Kelimeler: Yazılım kalite, Kalite modeli, Kalite değerlendirme, Kalite ölçme, Üst-model, Açık kaynak yazılım

ACKNOWLEDGEMENTS

I would like to give my warmest gratitude to my supervisor Assoc. Prof. Dr. Ayça Kolukısa Tarhan for invaluable guidance, wisdom, patience, and unlimited support since my initial graduate studies. Her expertise and encouragement helped me to complete this research and write this thesis. Also, I would like to thank the members of my thesis committee Assoc. Prof. Dr. Ali Seydi Keçeli, Assoc. Prof. Dr. Ömer Özgür Tanrıöver, Assoc. Prof. Dr. Adnan Özsoy, and Assist. Prof. Dr. Özden Özcan Top for providing helpful feedback and suggestions. Their insights and guidance were instrumental in helping to shape my research.

I would also like to thank my friends and colleagues Burçak Asal, Selma Dilek, Necva Bölücü, Merve Özdeş, Bahar Gezici, and Burcu Yalçıner for the pleasant time spent together in the office and in social settings.

I would also like to give special thanks to my wife Esra Sadi Yılmaz and my family as a whole for their continuous support and understanding when undertaking my research and writing my thesis. I could not keep my spirits and motivation high without their understanding and encouragement.

Finally, I would like to thank my daughter, who will be born in October, for making me the happiest father in the world.

CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS v

CONTENTS vi

FIGURES x

TABLES xiii

SYMBOLS AND ABBREVIATIONS xvi

1. INTRODUCTION 1

1.1. Definition of Problem 1
1.2. Proposed Solution 3
1.3. Methodology Followed 6
1.3.1. Literature Search-1 (Step-1) 7
1.3.2. Literature Search-2 (Step-2) 8
1.3.3. Mapping Process (Step-3) 9
1.3.4. The Proposed OSS-QMM (Step-4) 9
1.3.5. Validation in Real Context (Step-5) 10
1.4. Organization of the Thesis 11

2. BACKGROUND AND RELATED WORKS 13

2.1. Open Source Software (OSS) 13
2.1.1. General Information and History About OSS 13
2.1.2. Usage of OSS 15
2.1.3. Reasons for Preference 19
2.1.4. OSS Licenses 23
2.2. Software Quality Models and Meta-models 24
2.2.1. Analyzing the Current Situation of OSS Quality Models 25
2.2.2. Analyzing the Current Situation of OSS Quality Meta-models 30
2.3. Software Measurement Models/Standards 34
2.4. Meta-modeling 38

2.4.1. Basics of Modeling 38
2.4.2. Basics of Meta-modeling 39
2.4.3. Meta-Object Facility 43
2.4.4. Requirements/methods/criteria for Meta-model Validation 45

3. ELABORATION ON SOFTWARE QUALITY MODELS AND META-MODELS (STEP-1 AND STEP-2) 48

3.1. Software Quality Meta-models (Step-1) 48
3.1.1. Basic Characteristics of Meta-models 49
3.1.2. Software Quality Models Referenced in Developing Meta-models 51
3.1.3. Basic Characteristics of SQiE as Defined in Meta-models 52
3.1.4. Structure of Meta-models 54
3.1.5. Development of Meta-models 56
3.2. Software Quality Models (Step-2) 57
3.2.1 The Basic Characteristics of the OSS Quality Models 60
3.2.2. The Structure of the OSS Quality Models 61
3.2.3. The Degree of Guidance Provided by the QEMoF 63
3.2.4. The Basic Characteristics of QEMoF for Evaluating OSS 64
3.2.5. The Challenges in Developing the OSS Quality Models 65
3.2.6. The Evidence for the Usage of OSS Quality Models 67

4. MATCHING TERMS OF QUALITY MODELS AND META-MODELS (STEP-3) 70

4.1. Terms Analysis of Software Quality Meta-models 71
4.2. Detailed Analysis of the Structure and Content of Software Quality Models 76
4.2.1. Determination of QMs to be Taken as Reference in the Design of the OSS-QMM 77
4.2.2. Classification of Quality Models to be Taken as Reference 79
4.2.3. Structure Analysis of SQMs, Including OSS Quality Models 86
4.3. Mapping Process 88
4.3.1. Review of the Mapping Process by Experts (Step 3.1) 92
4.3.2. Performing the Mapping 93

5. OSS-QMM AND ITS DEVELOPMENT (STEP-4) 95

5.1. Development Process of the OSS-QMM 95
5.1.1. The Sub-steps of Development 97
5.2. Refinement Process of the OSS-QMM 107
5.2.1. Refinement with Subject Matter Experts (Step 4.4) 108
5.2.2. Refinement with the Validation Process 110
5.3. The Proposed OSS-QMM 113
5.3.1. Concepts of OSS-QMM in the Specification Category 115
5.3.2. Concepts of OSS-QMM in the Measurement Category 117
5.3.3. Concepts of OSS-QMM in the Evaluation Category 119
6. VALIDATION METHODS AND THEIR IMPLEMENTATION (STEP-5) 120
6.1. Case Studies 122
6.1.1. Determining the OSS Products 123
6.1.2. Determining Quality Characteristics and Sub-characteristics 124
6.1.3. Determining Measures and Measurable Concepts 125
6.1.4. Determining Evaluation Methods to Use in the Case Studies 128
6.1.5. Exploratory Study Applied as Part of Case Studies 132
6.1.6. Performing the Case Study-1 137
6.2. Expert Opinion Studies 142
6.2.1. Questionnaire Design and Execution 143
6.2.2. Part 1: Demonstrating Applicability of the OSS-QMM in Practice w.r.t Consistency in Evaluation Results and Effectiveness in Model Derivation 145
6.2.3. Part 2: Assessment of the OSS-QMM w.r.t Its Practical Applicability 150
7. DISCUSSION 153
7.1. RQ.1: Are Evaluation Results of the OSS-QMs Derived from the OSS-QMM Comparable? 153
7.2. RQ.2: Is the OSS-QMM Effective for Deriving the OSS-QMs? 156
7.3. RQ.3: Is the OSS-QMM Applicable in Practice? 160
7.4. Confidence in Validity and Potential Threats 162
8. CONCLUSION 166
REFERENCES 170

APPENDIX 189
APPENDIX-1 – List of Primary Studies Included in SLR Study 189
APPENDIX-2 – Development of the OSS-QMM Through Versions 191
APPENDIX-3 – The New Operationalized Quality Model Derived from OSS-QMM 195
APPENDIX-4 – Detailed Information about the Background of Experts 196
APPENDIX-5 – The Screenshots Obtained After the Semi-structured Interview conducted with Expert #10 via the Questionnaire 198
APPENDIX-6 – Mapping the Terms in Existing OSS-QMs (i.e., OSMM, OpenBRR, and SQO-OSS) to the Concepts of the OSS-QMM 200
CURRICULUM VIATE 201

FIGURES

Figure 1.1. The relationship between OSS-QMM and OSS-QMs 4
Figure 1.2. The development process of the OSS-QMM 7
Figure 2.1. Change in the number of developers of GitHub over the years 15
Figure 2.2. Distribution of: (a) respondents according to company size and (b) opinion of the respondent on the change in OSS usage compared to the previous year 16
Figure 2.3. Distribution of the reasons for users to prefer the OSS according to the survey result 19
Figure 2.4. Basic and tailored quality models over the years (OSS quality models are indicated in bold text) 25
Figure 2.5. Main relationships between the ISO/IEC standards of software quality and software measurement and their relationship with the CMMI model 35
Figure 2.6. The example use of a model over the inverters manufacturer 38
Figure 2.7. Meta-model that represents a language 40
Figure 2.8. Meta-model that represents abstract syntax modeling language for modeling inverter 40
Figure 2.9. The example representation of the mapping process between the concepts of inverter meta-model and inverter model 41
Figure 2.10. Abstraction levels of models and levels of modeling languages 42
Figure 2.11. The representation of the hierarchy of Meta-Object Facility (MOF) 44
Figure 2.12. Distribution of validation methods used for SQMM 46
Figure 3.1. Basic characteristics of meta-models (RQ1): (a) Percent distribution of main purpose, (b) Percent distribution of types of software products targeted, and (c) Percent distribution of whether meta-models are taken as the base for tool development 50
Figure 3.2. (a) Software quality model(s) taken as reference for meta-models, and the number of studies for (b) RQ2.2, (c) RQ 2.3, and (d) RQ 2.4 52
Figure 3.3. Basic characteristics of SQiE in meta-models (RQ3): (a) Percent distribution of subjective/objective evaluation, (b) Percent distribution of quantitative/qualitative evaluation, (c) Numeric distribution of evaluation result types, and (d) Percent distribution of structure of SQMMs 53

Figure 3.4. Percent distribution of basic characteristics of QEMoF for OSS: (a) whether represented formally (RQ 1.2), and (b) whether supported by a tool (RQ 1.4)..... 60
Figure 3.5. Number distribution of: (a) design structures of QEMoF, and (b) evaluation aspect of QEMoF 62
Figure 3.6. Number of studies for: (a) specification of the quality evaluation procedure, (b) demonstration of quality evaluation procedure by application 63
Figure 3.7. Distribution of: (a) subjective or objective evaluation supported (RQ 4.3), (b) quantitative or qualitative evaluation supported (RQ 4.4), (c) aggregation techniques used in QEMoF (RQ 4.7), and (d) how evaluation results are provided to users by QEMoF (RQ 4.11) 64
Figure 3.8. Distribution of studies with respect to author affiliation type..... 69
Figure 4.1. Percent distribution of sources (standards and proposals) that contribute to the terminology in SQMMs 73
Figure 4.2. The general structure of hierarchical SQMs..... 81
Figure 4.3. Classification w.r.t evaluation aspects and quality characteristics of: (a) OSS quality models, and (b) basic quality models 86
Figure 4.4. (a) An example of correct mapping, and (b) an example of incorrect mapping (there are unused terms and concepts)..... 90
Figure 4.5. The examples of the model development process considering the mapping process 91
Figure 5.1. Corresponding parts of Figure 1.2 for developing the OSS-QMM..... 97
Figure 5.2. Relationship between specification, measurement, and evaluation 101
Figure. 5.3. The process of sub-steps 4.1, 4.2 and 4.3 (indicated in green color) and their relationship with other steps 102
Figure 5.4. Types of relationships used in OSS-QMM 103
Figure 5.5. (a) Types of association relationships, and (b) an example usage of association relationship..... 104
Figure 5.6. (a) An example of an aggregation relationship, (b) an example of a composition relationship..... 105
Figure 5.7. The Venn diagram of the relationships between classes 105
Figure 5.8. An example of a generalization relationship 106
Figure 5.9. The refinement process of the OSS-QMM during the validation 111
Figure 5.10. The OSS-QMM 115

Figure 6.1. The validation process of OSS-QMM 121
Figure 6.2. Multiple-embedded case study design 123
Figure 6.3. Integrated AHP-TOPSIS method used for quality evaluation in the case studies.....130
Figure 6.4. Linear utility function according to the final quality scores of ERP products 142
Figure 7.1. Linear utility function according to the final quality score of each OSS product for case study 1-3 155

TABLES

Table 2.1. The cost of Microsoft solutions 22
Table 2.2. The cost of OSS solutions..... 22
Table 2.3. OSS solutions savings versus Microsoft solutions 22
Table 2.4. The OSS license types and their authorizations 24
Table 2.5. Comparison of the SLR studies by year, search string, number of primary studies, and research questions 26
Table 2.6. Comparison of meta-models for OSS..... 32
Table 2.7. Standards and proposals whose terminology is referenced by SQMMs 36
Table 3.1. The research question of SLR (Step-1) 49
Table 3.2. Concepts used in meta-models as entities with their frequencies..... 55
Table 3.3. Classification of challenges and the number of studies that have faced these challenges..... 56
Table 3.4. The research question of SLR (Step-2) 59
Table 3.5. List and frequency of challenges faced in developing QEMoF 66
Table 3.6 List of sources that: advance the QEMoF throughout the lifetime of their projects, expand the evaluations of the QEMoF, and provide evidence of practical use of the QEMoF..... 68
Table 4.1. List of concepts in SQMMs and analysis of their inconsistencies 74
Table 4.2. List of Evaluation Factors (EFs) with referenced RQs and scoring rules 78
Table 4.3. Classification of SQMs w.r.t structural, behavioral, and basic/tailored properties 80
Table 4.4. Description of community-related quality characteristics of OSS 85
Table 4.5. Structure comparison of SQMs (the first five are basic, and the last five are specific to OSS) 87
Table 4.6. Matching terms of quality models for OSS and concept of SQMMs w.r.t levels 93
Table 5.1. List of the relationship between concepts of OSS-QMM..... 106
Table 5.2. The versions of OSS-QMM with refinement performed and related reference 108

Table 6.1. List of RQs, description of RQs, and validation methods related to each RQ 120
Table 6.2. List of selected OSS products used in case studies.....124
Table 6.3. List of code-based measures with their description and measurable concepts associated with each measure.....127
Table 6.4. List of community-based measures with their equation and measurable concept associated with each measure.....128
Table 6.5. List of methods to use for the relevant concepts in the OSS-QMM 129
Table 6.6. Description of evaluation methods with their formulas used in case studies 130
Table 6.7. Background of industry experts participated in the case studies 133
Table 6.8. Parts of AHP process w.r.t. the Expert #10's judgement: **(a)** Pair-wise comparison, **(b)** Normalized decision matrix, and **(c)** Weight of sub-characteristics 130
Table 6.9. The weights for each sub-characteristics according to expert's judgements, the average of these weights, and the CR values 134
Table 6.10. Relationship between sub-characteristic and community-based measures 136
Table 6.11. The weights for each OSS aspect according to the expert's judgements and the average of these weights.....136
Table 6.12. **(a)** weights of sub-characteristics w.r.t importance, **(b)** weights of OSS aspects w.r.t importance, and **(c)** final weights for sub-characteristics in the case study-1.....138
Table 6.13. Final weights of sub-characteristics, impacts, measurable concepts (MC), measures associated with MC, values of measures, and decision matrix B.....139
Table 6.14. Normalized decision matrix R 140
Table 6.15. Weighted normalized decision matrix V.....141
Table 6.16. Values of Positive Ideal Solution (PIS) and Negative Ideal Solution (NIS) 141
Table 6.17. Separation measurement (S+ and S-), final quality evaluation score and rank values for ERP products.....141
Table 6.18. Background of experts (E1…E20) consulted during the validation process 144

Table 6.19. The answers to Q1, Q2, and Q3 obtained in Part 1 of semi-structured interview questionnaire 147
Table 6.20. The answers to Q4 obtained in Part 1 of semi-structured interview questionnaire 147
Table 6.21. Interpretation of 5-point Likert scale w.r.t its ranges 148
Table 6.22. The answers to Q5 and Q6 obtained in Part-1 of the semi-structured interview questionnaire 149
Table 6.23. The answers to Q1-12 obtained in Part-2 of the semi-structured interview questionnaire 151
Table 7.1. Evaluation results obtained from case studies and expert opinions 154
Table 7.2. Degree of confidence on the validity of the OSS-QMM with respect to empirical evaluation results 163

SYMBOLS AND ABBREVIATIONS

Symbols

A^+	Positive ideal solution
A^-	Negative ideal solution
S^+	Separation measures for positive ideal solution
S^-	Separation measures for negative ideal solution
$\sum$	Sigma (summation notation)
$[a_{ij}]_{mxn}$	Pair-wise comparison matrix A
$[b_{ij}]_{mxn}$	Decision matrix B
$[r_{ij}]_{mxn}$	Normalized decision matrix R
$[v_{ij}]_{mxn}$	Weighted normalized matrix V
$\in$	Belong to/is an element of
$\exists$	There exists
$\forall$	For all

Abbreviations

AHP	Analytic Hierarchy Process
BSI	Bug Severity Index
BSSR	Bug-solving Success Rate
CBO	Coupling Between Object classes
CC	Cyclomatic Complexity
CD	Commit Density
CI	Consistency Index
CMMI	Capability Maturity Model Integration
COTS	Commercial Off-The-Shelf

CR	Consistency Ratio
DD	Defect Density
DIT	Depth of Inheritance Tree
ED	Email Density
EF	Evaluation Factors
ERP	Enterprise Resource Planning
FRIS	Feature Request Implementation Success
FSM	Functional Size Measurement
GQM	Goal Question Metric
LCOM	Lack of Cohesion Of Methods
LOC	Lines Of Code
MC	Measurable Concept
MCDM	Multi-Criteria Decision Making
MDA	Model Driven Architecture
MDE	Model Driven Engineering
MOF	Meta-Object-Facility
MS	Micro Services
NC	Number of Contributors
ND	Number of Document
NIS	Negative Ideal Solution
NNL	Number of Nested Level
NOC	Number Of Children
NOS	Number Of Statement
NR	Number of Release
OMG	Object Management Group
OpenBRR	Open Business Readiness Rating

OSMM	Open Source Maturity Model
OSS	Open Source Software
OSS-QM	Open Source Quality Model
OSS-QMM	Open Source Quality Meta-model
PA	Product Age
PIS	Positive Ideal Solution
QEMoF	Quality Evaluation Models Or Frameworks
RFC	Response For a Class
RI	Random Index
SLR	Systematic Literature Review
SQiE	Software Quality and Its Evaluation
SQM	Software Quality Model
SQMM	Software Quality Meta-Model
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
UML	Unified Modeling Language
WMC	Weighted Methods per Class
WS	Web Services

1. INTRODUCTION

1.1. Definition of Problem

Open-source software (OSS) has a special copyright license allowing scrutiny of source code, free redistribution, unrestricted use, and the creation of derived works [1]. In the last two decades, OSS and its components have been used as part of the software that supports many activities of human life [2-4] and have attracted significant attention [5-6]. In the past, the primary motivation for using OSS was to reuse the existing code base by adapting it to the needs, which resulted in time and resource savings. However, in recent years, aside from these factors, the OSS has started to be seen as safe, reliable, and high quality, which has steadily increased the trend of using OSS and its components. The primary reason for this assumption is that it has undergone extensive testing by many developers from around the world and is thus considered error-free [7]. Since OSS solutions may be utilized for free and modified as needed, adopting them can be perceived as an easy solution. However, assuring the quality of the OSS is the biggest obstacle to its adoption. In other words, determining and assessing OSS quality have turned into a significant issue as the usage of OSS solutions by organizations has grown in popularity [8-9]. The increase in the use of OSS solutions by organizations and, accordingly, the poor quality of OSS products used in sensitive systems (e.g., real-time systems and control systems) may also cause disasters. In other words, it may cause permanent injury, loss of human life, dissatisfaction of the users, mission failure, financial loss or increase in the cost of maintenance [10].

Defining the word "quality" is a difficult task in the software engineering discipline [11-12]. Since the expectation of stakeholders (i.e., manager, user, tester, developer, customer, etc.) are different from software products or services, the meaning of "quality" varies among them. Even some international standards have defined software quality differently. For example, IEEE [13] defines software quality as follows: "*the degree to which a system, component, or process meets customer or user needs or expectations*". By contrast, the ISO 9001 standard [14] defines it as follows: "*the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs*". Aside from these differences in the definition, its abstractness and relativity have made the evaluation of software a challenging process. Despite the fact that determining and evaluating software quality is a difficult process, as mentioned above, it is a much more

challenging process in OSS products than its proprietary counterpart (i.e., commercial software). The reason is that they are different in considering objectives, planning, scheduling, task assignment, production, distributions, strategies, necessities, and documentation [15].

In the OSS projects, the source code is available for scrutiny, and historical data (i.e., number of defects, lines of code, etc.) has been stored in several repositories since the creation of the project. It means that several evaluation data are accessible from the code-based aspect (e.g., number of comments, lines of code, etc.) and community-based aspect (e.g., number of the developer, license type, mailing lists, etc.) in OSS projects. Unlike OSS, the other types of software (e.g., commercial off-the-shelf (COTS)) have limited, often private evaluation data. Therefore, determining and evaluating the quality of OSS projects is a challenging process since several heterogeneous data belonging to code-based and community-based aspects are scattered in various repositories and subject to evaluation. In other words, OSS has a dynamic and diverse nature, and accordingly, its quality is affected by many variables [16-18]. Since dealing with and aggregating all these heterogeneous data are difficult tasks, it is hard to develop a situation-based method for specifying the evaluation criteria for OSS projects [5][19-20]. Thus, performing quality evaluation for OSS is a complicated process for the reasons mentioned above.

Several well-designed, accepted and widely used software quality models (SQMs) have been proposed to evaluate and define software quality, such as ISO/IEC 9126 [21], Boehm [22], McCall [23], etc. Nevertheless, these quality models have primarily adapted to commercial software and ignored some specific properties of OSS products (e.g., community-based aspects) [2][24-26]. In other words, they do not provide sufficient support for evaluating OSS product quality [5][15]. To fill this gap, a variety of quality models or frameworks have been developed by practitioners and researchers to evaluate the quality of OSS, such as OSMM [27], OpenBRR [28], SQO-OSS [25], etc. However, results of some systematic literature reviews [24][29-30] and empirical studies [17][31] have concluded that there is little or no adoption of these OSS quality models or frameworks in practice. This is because these models have deficiencies in some aspects, as mentioned in the rest of this paragraph. For example, they are not applicable by external parties [9], not flexible enough to be applicable in all business domains [24], not fair in quality validation [26], and do not cover all aspects of quality [32-33]. Apart from them, the results of our latest SLR study [30] concluded that the quality models for OSS have

generally arisen from the needs of evaluators, such as organizations and software practitioners; and the variety in the needs and expectations of these evaluators has caused the structure of the developed models to be heterogeneous. Also, according to the results of a survey study [34], more than 71% of companies developed their own OSS quality models [35-36]. This survey indicated that quality models moved away from standardization and turned into individual models, supporting the results of our SLR study. The majority of the models consist of a wide variety of information in their bodies in various structures as the base for specifying and evaluating OSS quality [30][37]. All this diversity has led to the proliferation of individual heterogeneous OSS quality models [30][38]. As a consequence of this situation, results obtained by using different OSS quality models for the same product with the same purpose may diverge from each other, and it becomes impossible to have a common and consistent basis for comparing the OSS quality in the community [30]. It means that evaluation results may become incomparable and unreliable [30][35-36]. This negatively affects standardization, which is an important communication vehicle for companies when interoperating with others. Standardization assists organizations in interoperating using engineering discipline with agreed and well-recognized practices and technologies [39-40]. As a result, the reasons mentioned above have hampered the practical use and adoption of OSS quality models.

1.2. Proposed Solution

Due to the challenges described in Section 1.1, evaluators often choose OSS products on the recommendations of their colleagues without using a quality model [17][42]. In this context, in this thesis study, we have developed an Open Source Software Quality meta-model (OSS-QMM) that aims to eliminate the problems and address the challenges in the evaluation of OSS products. Our motivation has been to create a solid base to derive OSS quality models with homogeneous structure and common terms by using the proposed OSS-QMM and, accordingly, eliminate the standardization problem which is the most important criteria in measurement and evaluation. Prior to proposing the OSS-QMM, we conducted two separate SLR studies [30][37] analyzing OSS quality models and software quality meta-models. According to the common result of these SLR studies, the need for such a meta-model is emphasized. The results of these two SLR studies are explained in detail in Section 3.

In order to increase the traceability of the thesis content on the following pages, the reader should consider the following abbreviations. Throughout the thesis, the term "software

quality meta-model (**SQMM**) " or "meta-model" corresponds to the quality of all types of software, unless it is indicated that it corresponds to a specific type of software (e.g., OSS). The same is also true for the term "software quality model". That is, the term "software quality model (**SQM**)" or "quality model" corresponds to the quality of all types of software. An OSS quality model is indicated as **OSS-QM**. Specifically, the abbreviation for our meta-model developed within the scope of this thesis study is indicated as **OSS-QMM** (Open Source Software Quality Meta Model).

Meta-models are defined as "models of models with the rules needed to build specific models" [42-43], so the models which are derived from the meta-models have homogeneous structures and common terms [34][43]. In other words, meta-models are important because they enable standardization of model development [34][44]. In this context, as shown in Fig. 1.1, the OSS-QMM developed in this thesis will be an abstract form of OSS-QMs, and each OSS-QM will be an instance of the OSS-QMM. That is, new operationalized OSS-QMs and existing OSS-QMs with homogeneous structures and common terms will be derived (or instantiated) from the OSS-QMM. In this way, evaluation results obtained by using different OSS-QMs will be comparable and reliable. In this regard, the OSS-QMM aims to address the quality of OSS, consider all possible characteristics of the underlying areas, be flexible enough to apply for various needs of users with modifications or minor additions, meet the needs of all the interested parties, and serve as a standard reference for the evaluation of OSS products. After all, the OSS-QMM aims to facilitate formalization and helps standardizing the specification, measurement and evaluation of OSS products.

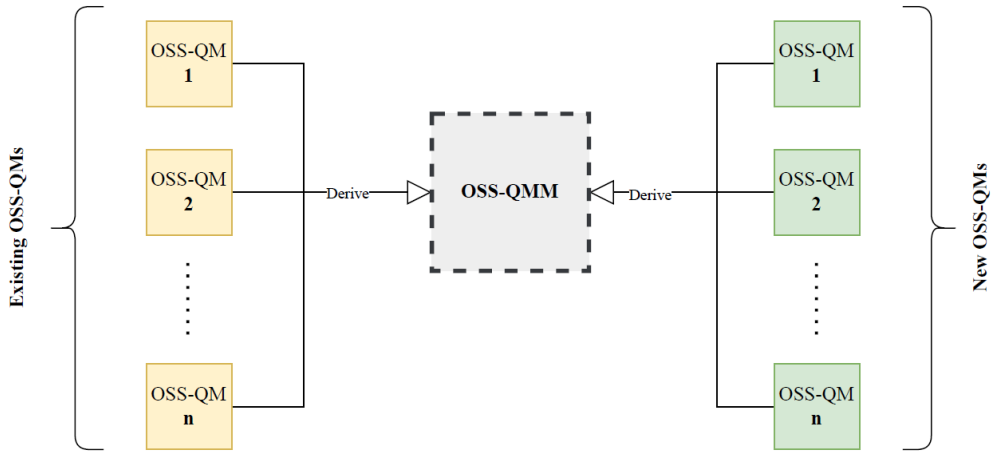


Figure 1.1. The relationship between OSS-QMM and OSS-QMs

More specifically, the work done in this thesis and the proposal of the OSS-QMM together with its validation studies contribute to the literature as follows;

- The content and structure of existing SQMMs in the literature are analyzed. This has helped us gather available information and build a solid foundation for developing a comprehensive OSS-QMM. Also, it has triggered us to eliminate the inconsistency in the concepts of the SQMMs (detailed in Section 3.1).
- The content and structure of existing OSS-QMs in the literature are analyzed. This has helped us gather available information and understand the deficiency of current OSS-QMs. Also, it has guided us to discover the common structure of OSS-QMs (detailed in Section 3.2).
- A comparative analysis of the concepts used in the SQMMs is presented. This has contributed to harmonizing the concepts of different SQMMs and also formed the basis for proposing the OSS-QMM (detailed in Section 4).
- An investigation on how the concepts used in the SQMMs are expressed in the referenced standards is carried out. This has led to eliminating inconsistencies in the international standards proposed so far (detailed in Section 4).
- The terminology used in the OSS-QMs is mapped with the terminology of the SQMMs. This has contributed to the resolution of the terminology conflicts between the OSS-QMs and the SQMMs (detailed in Section 4).
- Since the OSS-QMM is proposed considering the common structure of the important OSS quality models, similar quality models to be proposed in the future using the OSS-QMM will have the chance of adopting a standard structure (detailed in Section 4 and 5).
- Unlike the meta-models proposed in the literature, the OSS-QMM has been validated in a real-world context by employing multiple empirical research methods, i.e., expert opinion and multiple-case study (detailed in Section 6).
- It has been validated by the empirical studies that the OSS-QMM has concepts free of inconsistencies and enables the derivation of OSS quality models with homogenous structures and terms (detailed in Section 6).

- The validation results have indicated that the OSS-QMM has the potential to increase the adoption of the existing OSS quality models and contributes to the standardization of OSS quality evaluation (detailed in Section 6).
- The validation process has helped to understand the practical needs in industry by using expert opinions and to revise the OSS-QMM by taking these needs into account. That is, the OSS-QMM has been validated by stakeholders who are potential users of the OSS-QMM in practice (detailed in Section 6).
- Finally, the multiple case-study applied for quality evaluation of three open-source ERP products has become an example to guide non-experts in the use of the OSS-QMM (detailed in Section 6).

1.3. Methodology Followed

In this section, the methodology followed in developing the OSS-QMM and the organization of the thesis is presented. That is, the OSS-QMM development process is briefly explained with reference to the later sections. In this thesis, the ***step-based process for meta-model creation,*** which is described in Fig. 1.2, is followed. This process has been adapted from Beydoun et al. [45] and Othman et al. [46] and commonly used by many studies that proposed meta-models in literature, e.g., [47-48]. In this meta-model development process, necessary preparations are first performed to create a solid foundation that will enable the development of the targeted meta-model. Then, the initial meta-model is developed based on this solid foundation. Lastly, the final version of the meta-model is obtained with improvements during the validation phase.

An array of meta-modeling frameworks has been proposed by many researchers in information systems, e.g., [49-51]. Among them, we have adhered to a meta-modeling framework based on the "Meta-Object-Facility (MOF)" standard [49] offered by Object Management Group (OMG). This is because the MOF standard, which has proven itself in meta-model development, is widely used for meta-modeling in literature [52-53]. The details of the MOF standard are given in Section 2.4.2. As seen in Fig. 1.2, the development process of the OSS-QMM is iterative, with continuous refinement of new concepts. In other words, a systematic process consisting of five main steps is followed in the development of the OSS-QMM. In the remainder of this sub-section, the steps of this process are explained briefly. Details of each step are given in later sections.

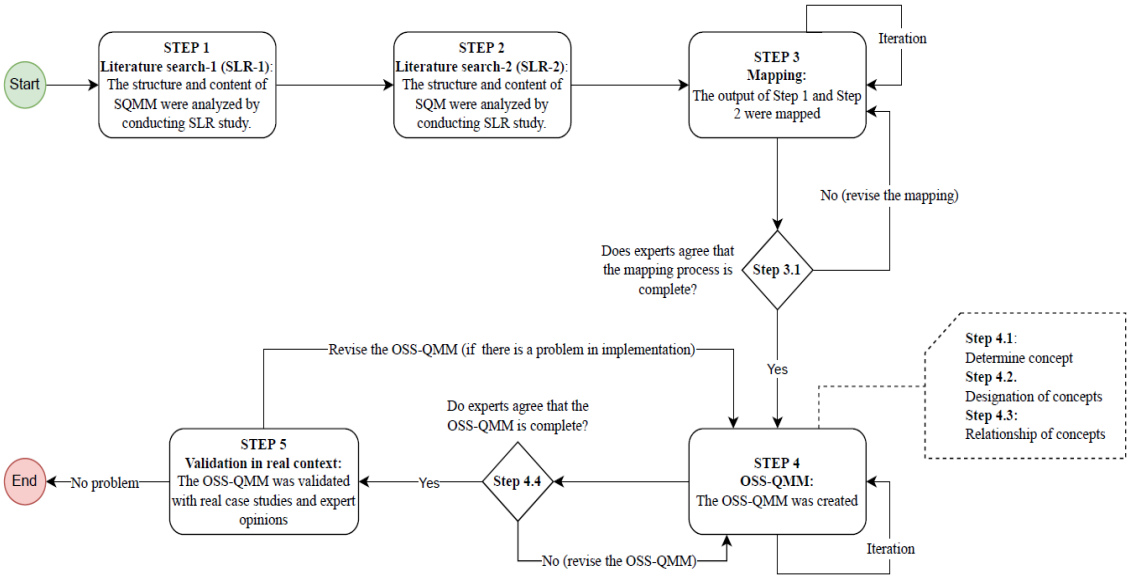


Figure 1.2. The development process of the OSS-QMM

1.3.1. Literature Search-1 (Step-1)

The structure and content of the existing quality meta-models in literature should be well understood in order to develop a complete and coherent Software Quality Meta-model (SQMM). This knowledge enhances our domain awareness, as mentioned in the initial step of the step-based meta-modeling process [45]. This step is valid for any meta-model creation process [46]. For this purpose, an SLR study [37] has been performed, and a total of 28 SQMMs have been analyzed. In this SLR study, the structure and content of the existing SQMMs, including meta-models proposed for OSS, have been analyzed.

In this context, we investigated these SQMMs from various aspects such as: basic characteristics, structure, content, referenced quality models, mapping process, data acquisition types, methods/techniques used in evaluation, research methods, challenges faced while developing the proposal, and validation methods. Therefore, this SLR study provided us with information about the structure, content and deficiencies of the SQMMs. Also, in this study, the concepts employed in the included meta-models with their frequency of use have been elicited, and this indicated that there are inconsistencies between the terms of the meta-models. Afterwards, a detailed analysis has been performed to eliminate these inconsistencies. The origins of the terms have been explored, and the terms have been analyzed based on international standards. As a result, the output of Step 1 formed the basis for shaping the content of the OSS-QMM. The details of Step-1 are explained in Section 3.1.

1.3.2. Literature Search-2 (Step-2)

This step covers gathering the knowledge sources to be used in the development process of the OSS-QMM, as in Step-1. In order to develop a complete and coherent SQMM, the structure and content of the existing quality models in the literature should be analyzed since the quality models are the instances of the SQMMs. In this context, we performed another SLR study [30] and analyzed a total of 36 quality evaluation models or frameworks (QEMoF) proposed for OSS. In this SLR study, we investigated these QEMoF from various aspects such as: basic characteristics, structure, technical details of the evaluation procedure, support for evolutionary evaluation, types of data collection, research methods, degree of required skills for evaluation, challenges faced while developing the proposal, evidence for practical use, and validation methods. The results from this SLR in this study are presented together with a number of evaluation factors, which can be used to compare the overall quality of the QEMoF to guide potential users, as the final output.

Although this SLR study examined the QEMoF from many aspects, one of the most important findings has been that there is little or no adoption of these models/frameworks in practice. Other secondary studies [24][29] also support this situation. As revealed in the SLR study [30], this is because quality models have moved away from standardization and turned into individual models. That is, OSS quality models vary in terms of the software aspects they evaluate, subjective and objective evaluations, quantitative and qualitative evaluations, the aggregation techniques, the level of user skills in using the model, the data type of the evaluation results provided to the user, etc. All this diversity has led to the proliferation of individual and heterogeneous quality models. Therefore, in our later study [54], a common structure of the OSS quality models has been aimed in order to eliminate this heterogeneity. In this context, a total of 10 quality models have been determined and analyzed, and the process of determining these quality models is explained in Section 2.2. The results of our second SLR study [30] revealed that most of the OSS quality models have a hierarchical structure. After the analysis of the 10 quality models, we observed that all software quality models are based on a common structure consisting of five levels (as explained in detail in Section 5.2). As a result, the output of Step-2 formed the basis for shaping the structure of the OSS-QMM. The details of Step-2 are explained in Section 3.2.

1.3.3. Mapping Process (Step-3)

As a result of the analysis performed in Step-1, the inconsistencies between the terminologies of the meta-models have been analyzed according to their meanings in the related meta-models and international standards. This analysis provided knowledge about the meanings of the terms to use in the OSS-QMM to be developed, in other words, its semantics. In Step-2, the structure of the quality models has been analyzed, and this analysis provided knowledge about the structure of the OSS-QMM to be developed, in other words, its syntax. Then, in our third study [54], the concepts of the quality meta-models have been matched to the terms of the quality models since models are defined as instances of meta-models according to the MOF architecture [49]. MOF structure is explained in Section 2.4.3. In this regard, a level-based matching process has been carried out in iterations, as already shown in Fig. 1.2. That is, during the matching process, a series of meetings have been held between this student and his supervisor. In this context, a total of 11 iterations has been performed through online meetings and emails. While performing the matching, the meanings of the terms and their intended uses have been taken into account. Also, in Step 3.1, this mapping process and its outputs have been reviewed by four subject matter experts on software quality models, as shown in Fig. 1.2. That is, it has been aimed to examine the mapping process by external parties other than the researchers involved in this thesis study. In this context, experts have been asked questions to review our mapping process. The mapping has been revised based on feedback from experts when necessary. As a result, the final version of the mapping has been obtained by considering the process described above. The background of the experts who reviewed the mapping, the questions asked to the experts, and details of the mapping process are explained in Section 4.3.

1.3.4. The Proposed OSS-QMM (Step-4)

In this step, the Open Source Software Quality Meta-model (OSS-QMM) has been proposed by considering the outputs of Steps 1, 2, and 3. The details of the OSS-QMM are explained in Section 5. The content of the OSS-QMM has been determined as the output of Step-1, and the structure of OSS-QMM has been determined as the output of Step-2. In Step-3, the meta-model concepts corresponding to each level of the quality models have been determined. Then, as seen in Fig. 1.2, Step-4 consists of four sub-steps, namely 4.1, 4.2, 4.3, and 4.4. In this regard, the concepts of the OSS-QMM have been

determined in Step 4.1, these concepts have been designated in Step 4.2, and the relationships between the concepts have been determined in Step 4.3. In Step 4.4, the OSS-QMM has been reviewed by four subject matter experts on software quality models, as shown in Fig. 1.2. That is, it has been aimed to examine the OSS-QMM by external parties other than the researchers involved in this thesis study. It should be noted that the experts consulted in this Step 4.4 and the ones in Step 3.1 have been the same group of experts. In this context, experts have been asked questions to review the OSS-QMM. The OSS-QMM has been revised based on feedback from experts when necessary.

As shown in Fig. 1.2, a review-and-revise process has been followed while carrying out the sub-steps. That is, a series of meetings have been held between this student and his supervisor for the purposes of reviewing the meta-model (in Step 4) and its development process (in Steps 1-3) and then revising the meta-model. In this context, a total of 15 iterations has been performed through online meetings and emails. Also, in these meetings, refinements have been made as a result of the activities mentioned in Step 4.4 and Step 5. Thus, the concluding decisions have been made as the result of a series of iterations. The background of the experts (in Step 4.4), the questions asked to these experts, and details of the OSS-QMM and its development process are explained in Section 5 in detail.

1.3.5. Validation in Real Context (Step-5)

In this step, the OSS-QMM has been implemented in practice and has been validated in a real context. Before validating the meta-model in a real context, an example implementation of the OSS-QMM has been demonstrated in an unreal OSS product with dummy evaluation data through a toy experiment [54]. In this example implementation, it has been understood that the OSS-QMM has been applicable in practice. Then, in Step 5, multi-faceted empirical research has been employed to validate the OSS-QMM in a real context, as details are given in Section 6. For this purpose, case study and expert opinion, which are the two most used empirical research methods for validating meta-models as reported in [37], have been used. In this context, three research questions (RQs) have been determined to investigate the validity of the OSS-QMM by using these two validation methods. Each RQ has been aimed at validating the OSS-QMM from different aspects such as results comparability, effectiveness in model derivation, and applicability in practice, respectively. More specifically, the three research questions are given below:

- **RQ.1**: Are the evaluation results of OSS quality models derived from the OSS-QMM comparable?
- **RQ.2**: Is the OSS-QMM effective for deriving the OSS quality models?
- **RQ.3**: Is the OSS-QMM applicable in practice?

For case study research, multiple-embedded case studies have been designed, in which a new OSS quality model has been derived and two important existing quality models (i.e., OSMM [27] and OpenBRR [28]) have been instantiated from the OSS-QMM. Three widely-used and open-source ERP systems (namely; Adempiere, Compiere, and Apache OFbiz) have been determined to evaluate quality for their maintainability.

In other words, the OSS-QMM has been validated by instantiating quality evaluation on real OSS products with real data, as specified in Fig. 1.2. This way, the applicability of the OSS-QMM has been demonstrated and also, it has been shown that the evaluation results obtained by using different OSS quality models, all derived from the OSS-QMM, are comparable. In the case studies, the integrated AHP-TOPSIS method has been used as an evaluation method, in accordance with the steps of using the OSS-QMM in quality evaluation. A total of 20 experts have taken into account the structure and the applicability of the OSS-QMM in a real-world setting while providing their opinions. In addition, they have shared their opinions about the meta-model by considering the OSS quality models that they have used in their own companies or are well-known in practice. Throughout this validation process, the meta-model has been reviewed to investigate whether there has been any unmatched concept in the OSS-QMM or the derived quality models, whether there has been a problem in the relationships between the concepts, and whether the meta-model could be applied successfully in practice. Then, the meta-model has been revised in case any of these situations have been encountered. Details of the validation process are explained in Section 6.

1.4. Organization of the Thesis

The rest of this thesis is organized as follows: In Section 2, the background that forms the basis for the development process of the OSS-QMM and an overview of related studies are given. In other words, general information about OSS, software quality models, meta-models, software measurement standards, meta-object facility standard, and validation methods for meta-modeling are presented. Section 3 presents details of analyzing meta-models (in Step-1) and quality models (in Step-2) that are taken as the basis for the

development of the OSS-QMM. In this context, analyzing concepts of meta-models and structure analysis of software quality models are presented. Section 4 elaborates the mapping process between concepts of meta-models and terms of quality models (in Step-3) that are also taken as the basis while developing the OSS-QMM. In Section 5, the Open Source Software Quality Meta-model (OSS-QMM) and its development process are presented in detail (in Step-4). In Section 6, empirical research methods employed for validating the OSS-QMM and their implementation are presented (in Step-5). In Section 7, the results obtained from the validation process are discussed. Finally, in Section 8, conclusions and plans for future work are presented.

2. BACKGROUND AND RELATED WORKS

This section introduces the general concepts that are used throughout the thesis and the importance of these concepts. Related work in quality modeling and meta-modeling are also presented. In this context, first, in Section 2.1, general information about OSS and its importance is given. In Section 2.2, the current status of the OSS quality models and meta-models in literature are presented together with related studies. In Section 2.3, international standards and their importance for the OSS-QMM developed in this thesis are presented. Finally, Section 2.4 focuses on the basics of meta-modeling.

2.1. Open Source Software (OSS)

In this section, general knowledge is given about the following issues: OSS and its history, usage of OSS, reasons for its preference, and OSS licenses. The purpose of this section is to explain the importance of OSS to the reader.

2.1.1. General Information and History About OSS

Open Source Software (OSS) is a type of software with a special copyright license allowing scrutiny of source code, free redistribution, unrestricted use, and the creation of derived works [1]. In addition, the OSS makes it possible to distribute the software to third parties for a fee or free of charge. In order for the software to be an OSS, it is not enough to allow scrutiny of source code and offer the software free of charge. According to the Open Source Initiative [55], the software must comply with the following criteria to be an OSS:

- "The software should be freely redistributed "
- "Providing the source code with the software product or providing the opportunity to obtain it free of charge"
- "The license should allow changes (derived works), but derived works should be distributed under the original software license"
- "The license of OSS should not discriminate against any person or group and should be valid for everyone"
- "No restrictions on the use of the software for a specific field of endeavor"

- "The license should not be specific to a particular product but to the software itself"
- "The license should not impose restrictions on other software distributed with the licensed software"
- "The license should be valid for all persons to whom the software reaches without any additional process"

Although the concept of OSS was mentioned in the artificial intelligence laboratories of universities such as Massachusetts Institute of Technology, Stanford University, Carnegie Mellon University, and California (Berkeley) University in the 1960s, the first official step was taken in the 1970s [56]. During these years, Richard Stallman, an American software developer, released an open-source code version of UNIX, which is an operating system with a large number of users [57]. This version, which offers everyone the opportunity to modify the source code freely, has attracted the attention of users.

The institutionalization of OSS was first started in 1983, again by Richard Stallman, by resisting the closing of the source code of the driver software for the Xerox printer in the laboratory of the Massachusetts Institute of Technology. Then, in 1989, the General Public License (GPL) was developed to determine the boundaries of OSS and put them on solid foundations. Another important development regarding OSS took place in 1998 [57]. The Netspace company lost market share to a large extent against Microsoft's web browser and Internet Explorer. Then, they made a critical decision in January 1998 to regain their lost position and opened the source codes of their web browser to the user. This move of the company was considered an important step for the development of OSSs [57]. The popularity of OSSs has increased day by day after these years and has increased remarkably, especially after 2010 [57][59-60]. In Fig. 2.1, the number of developers of GitHub is analyzed over the years according to GitHub's January Report [61]. It is easily seen in the figure that the interest in OSS has increased rapidly. As of 2023, the number of GitHub developers has exceeded 100 million [61]. These analyzes were obtained only for GitHub, and considering that there are many cloud repositories (e.g., source forge, Apache, etc.) other than GitHub, it can be seen how huge the popularity of OSS has reached.

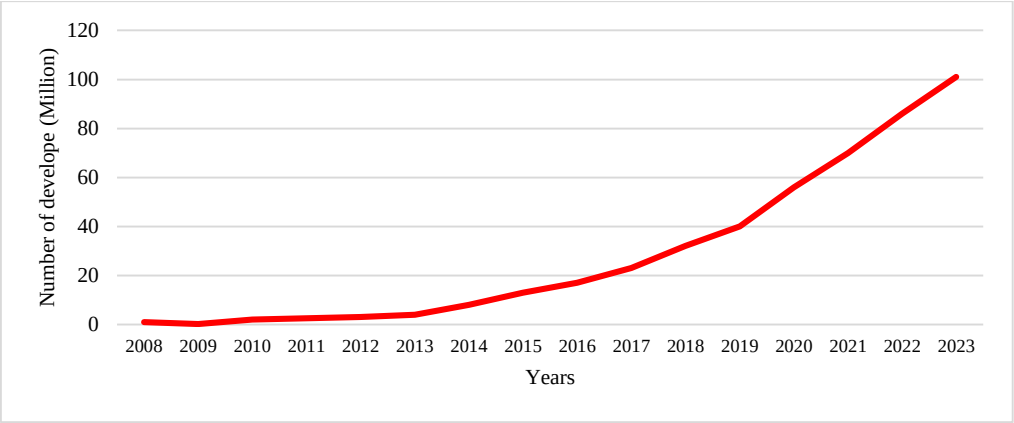
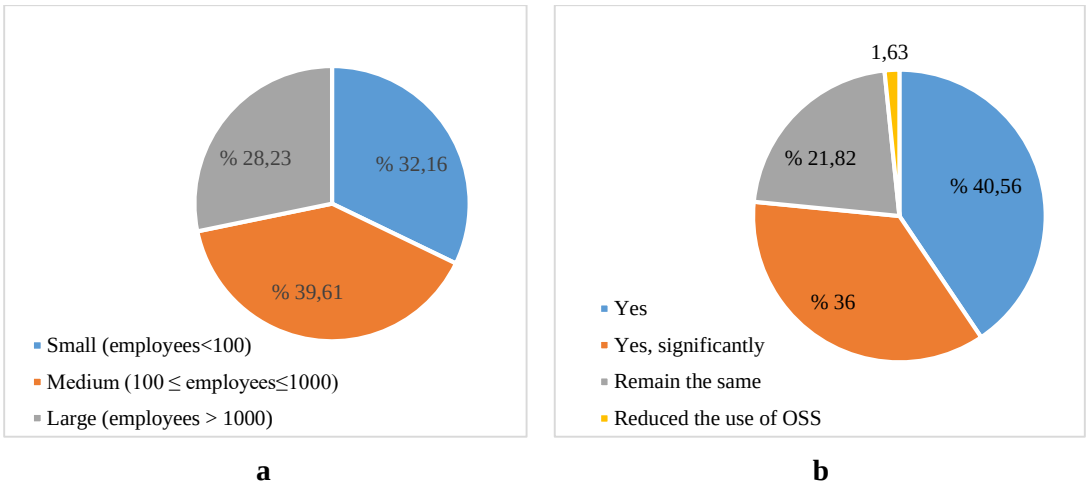


Figure 2.1. Change in the number of developers of GitHub over the years

2.1.2. Usage of OSS

The OSS has started to be preferred by the masses as they are perceived as high quality and reliable due to the scrutiny of many developers, and the users' dependence on vendor companies is removed. A survey was conducted by the open source initiative in 2022 to analyze the use of OSS in the market, and a report was published [62]. A total of 2660 respondents from all over the world participated in this survey. The distribution of these participants according to the size of the companies they represent is given in Fig. 2.2 (a). As seen in Fig., almost 65% of the participants represent medium and large companies. These participants were asked whether the use of OSS has increased in the companies they represent compared to the past year. As seen in Fig. 2.2 (b), 76% of the participants stated that OSS use increased compared to the previous year, and 21% of them stated that it remained the same. Only 1.63% of them declared that the use of OSS decreased compared to the previous year. These analyses made with the real data obtained directly from the participants in the market showed that the use of OSS is increasing day by day. In addition, the sectoral distribution of OSS usage was examined in the survey conducted by Synopsys, which is an American Design Automation (EDA) company [59]. As a result of this survey, it is seen that OSS use is mostly in public institutions by governments around the world. This is followed by the use of OSS in the field of healthcare and media. In parallel with the results of this survey, many governments and organizations have worked to increase the use of OSS. Europe Union governments and companies have already noticed the potential of Open by investing over €1 billion in open-source development in 2018 alone [63]. Also, they plan to invest over €95 billion in open-source development between 2021-2027 years [63]. In this context, examples from around the world are given in the following paragraphs.



a **b**

Figure 2.2. Distribution of: **(a)** respondents according to company size and **(b)** opinion of the respondent on the change in OSS usage compared to the previous year

European Commission: The European Commission has supported the use of OSS with various initiatives after the establishment of the Free Software Working Group [62]. It has supported many projects that support and spread OSSs within the scope of the 6th and 7th Framework Programs of the European Union. A cooperation working group was established to share experiences and good practices related to OSS with the support of the European Commission [63]. Well-prepared practices and studies have been published on the European Commission's website thanks to this working group. Even within the scope of the European Commission's 2020-2023 strategies, under the theme "Think Open", they have committed to promote the use of OSS not only in practical areas such as IT but also in areas where it can be strategic [64].

England: England first supported the use of OSS at the political level after 2004. They have not made the use of OSS directly obligatory in their own public institutions. However, they have followed encouraging methods to search for open-source alternatives to commercial software used in their public institutions. Then, with an action plan published in 2009 [65], the use of software in England was made compulsory in their public institutions. Detailed justifications were requested from the public institutions that preferred the use of commercial software, and only those who provided valid reasons were allowed. As a result, in a study by Aiven [66], 71% of England government employees reported using more OSS products in their public institutions compared to 5 years ago [67].

Netherlands: In the Netherlands, the planning for the use of OSS is followed by the Ministry of Finance. With an action plan (i.e., Open Standards and Open Source Software) published in 2003, the Netherlands encouraged the use of OSS in its public institutions. In 2007, a study was conducted within the framework of this action plan, and it was observed that public institutions started to adopt OSS rapidly. However, the Dutch Government found this insufficient and expanded the scope of the action plan to increase the use of OSS after 2007. The Dutch government launched open-source incentive action plans in 2017 and 2018, and finally published NL DIGITAAL, also referred to as the government data agenda, in 2019. In this regard, they have recommended the use of OSS in order to increase transparency with regard to the software that is used by the public administration [68]. For example, in 2019, Logius, the Dutch government digital service that is part of the Ministry of the Interior and Kingdom Relations, developed an OSS-based data exchange service [68]. This system has been used by both private and public organizations. As a result of these action plans on the use of OSS, more than 60% of Dutch government public organizations have adopted the use of OSS.

Malaysia: The Malaysia Public Sector Open Source Software Master Plan [69], consisting of three phases, was launched in 2004. Phase 1, called "Early Adoption", was completed in 2006, Phase 2, called "Accelerated Adoption", was completed in 2010, and Phase 3, called "Self-Reliance", started in 2011 and it continues [70]. After this action plan, the use of ABM in public institutions in Malaysia has greatly increased, and according to the reports of the Gartner company [71], the following gains have been achieved; 80% savings on license costs, 58% reduction in consulting and development work, 7% savings on software support services, and approximately 30% savings were achieved in total.

Turkey: The use of OSS in Turkey is one of the important issues addressed in the e-Transformation project launched in 2003 [57]. In this context, firstly, with action plan no. 7 in 2005, a report on the subject was published by the Information Society Department of the Ministry of Development with the participation of the relevant stakeholders. In this report, the basic features, history, usage areas and advantages of OSSs are included, and OSSs are examined from the legal and financial aspects [57].

The best example of OSS usage in Turkey is the PARDUS operating system, which was started to develop by TUBITAK-BILGEM in 2003 and released its first version in 2005 (Pardus 1.0) [72]. Thanks to the social effects of Pardus, versions that can also be used

by individual users have started to be published, and the awareness of OSSs has increased. In 2007, a new corporate version was published under the name "Pardus 2007", and the first institution to use it was the Ministry of National Defense Recruitment Office (ASAL) [58]. In 2007, ASAL started to use this operating system in a total of 625 servers and 4500 clients in all its institutions throughout Turkey. As of 2008, Radio and Television Supreme Council (RTÜK) started to use the Pardus operating system on nearly 100 computers in the Digital Recording, Archive and Analysis System (SKAAS). During these years, the Pardus operating system started to be used on 1700 computers at the General Directorate of Istanbul Water and Sewerage Administration (ISKI) [57].

The Pardus project, which is Turkey's most notable OSS project, was included in the Public Investment Programs of the Ministry of Development in 2008 and is still included in investment programs and action plans. In the final evaluation reports of the Information Society Strategy and Action Plan (2006-2010), it is stated that a contract was signed in 2009 for the OSS transformation in the Energy Market Regulatory Authority [57]. Also, in the final evaluation reports of the Information Society Strategy and Action Plan (2015-2018), it is stated that there are plans for the dissemination of OSS and Pardus in the public sector and for the development of the private sector ecosystem [58]. According to the plan, information, promotion and training activities will be carried out for public institutions in order to encourage the use of Pardus and OSS, and TÜBİTAK will provide free consultancy services and training to those institutions.

After all these efforts, many public institutions in Turkey are using OSS and Pardus operating systems, e.g., the Ministry of Justice, National Defense Department, Ministry of Environment, Urbanisation and Climate Change, Radio and Television Supreme Council, Energy Market Regulatory Authority, General Directorate of Security, Ministry of Education, Ministry of Health, Ministry of maritime transport and Communications, General Directorate of Land Registry and Cadastre, National Lottery Administration, Central Bank of the Republic of Turkey, General Directorate of Istanbul Water and Sewerage Administration, and various municipalities and universities [57]. For example, the Ministry of National Defense has planned to use it in 1 million 800 thousand computers until the end of 2023. In this context, it is aimed to generate an annual profit of 2.2 billion dollars from the operating system and other software licenses. Many OSS projects are carried out by TUBITAK-ULAKBIM apart from Pardus. For example, Octopus Integrated Cyber Security System, EnGerek Identity Management System,

LiderAhenk Central Management System, Etap Interactive Board Interface Project, ULAKBUS Integrated University System, LibreOffice, etc.

Apart from the countries mentioned above, many countries such as France, Germany, Spain, Mexico, Brazil, Korea and India have adopted the use of OSS in their public institutions and made it a part of their information society strategies [57].

2.1.3. Reasons for Preference

The OSS has enabled this ecosystem to be adopted as a preference rather than an option, thanks to the opportunities they provide [73]. This software has become a business model that reduces the costs of information systems, especially in public institutions, and increases information security [57][74]. In a survey conducted by the open source initiative in 2022 [62], OSS users (i.e., 2660 participants) are asked why they prefer the OSS in their organizations. The results of the survey are shown in Fig. 2.3. It is seen that the most frequent reason for users to use OSS is that this software has access to innovations and the latest technologies. This is because many developers around the world work with the OSS, and these developers are constantly intertwined with new technologies. As shown in the figure, the second most important reason is that the OSS does not have licensing costs, and the total cost of ownership is low. Also, more than 36% of participants indicated that they use the OSS to modernize their technology stack. Apart from them, as shown in Fig. 2.3, the reasons, such as offering many options for similar technologies and low vendor lock-in, are also important reasons for users to prefer the OSS.

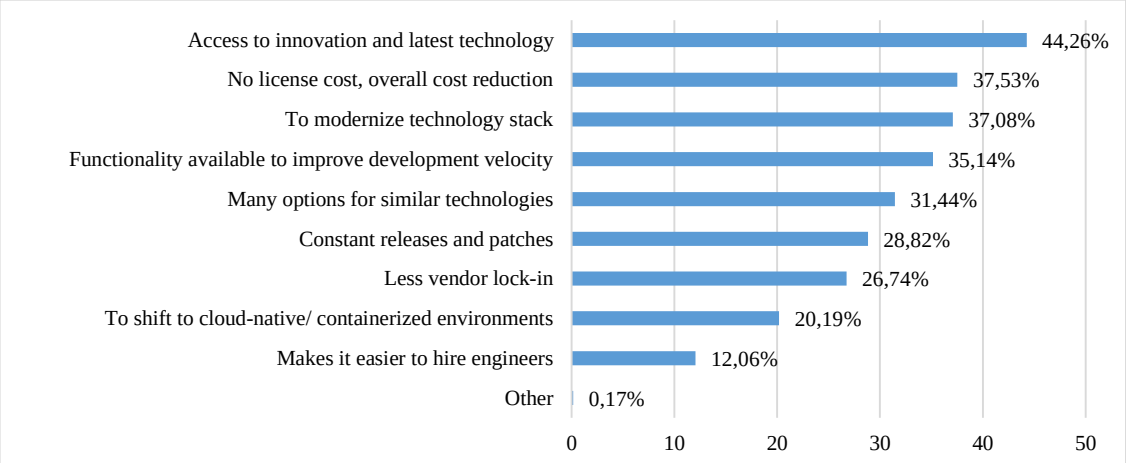


Figure 2.3. Distribution of the reasons for users to prefer the OSS according to the survey result [62].

Apart from this survey, many surveys (e.g., [59][75]) and studies (e.g., [57][74]) have been conducted in the literature related to the reasons for the preference of the OSS. For example, in a report jointly presented by Black Duck Software [59] and North Bridge Venture [76] organizations, it is emphasized that the reasons for preferring the OSS have changed over the years. According to the report, the primary reasons for users to prefer the OSS were that they have no vendor lock-in and no license cost. However, these reasons have changed in recent years, and users have started to prefer OSS software because they are of higher quality and reliable. In the following sub-sections, the most preferred reasons for OSS are explained according to the common results obtained from the studies and reports in the literature.

2.1.3.1. Access to Innovation and Latest Technologies

The OSS is under the follow-up of many OSS users (e.g., developers) from all over the world, and as a result of this follow-up, it constantly updates itself with new versions. In other words, after technological development and need, the OSS can be easily modified by experienced developers. This situation prevents this software from being behind the times. Therefore, OSSs are developed open to innovative ideas. The most important evidence of this is that huge companies such as Google, IBM, Yahoo, and Amazon have entrusted their important business operations to OSS solutions, especially the GNU/Linux operating system. Considering that these huge companies must constantly follow innovations and the latest technologies, it can be concluded that OSS is successful in this regard. Therefore, users who are constantly interacting with the OSS stated that the most important reason for preferring the OSS is that this software is constantly supported by new technologies.

2.1.3.2. Total Cost of Ownership

The OSS offers the opportunity to use the software freely without paying for the license. However, only the license costs of the software products are not taken into account in the calculation of the Total Cost of Ownership (TCO) of the software [77]. Personnel costs, cost of equipment requirements, training costs and opportunity costs should also be considered in calculating TCO. In addition, costs such as upgrading, technical support, and end-user costs should not be neglected. The initial acquisition cost of OSS is very low compared to Commercial off-the-shelf (COTS) software. It cannot be claimed that

OSS is completely free of cost. Like COTS software, there are support, maintenance, documentation and training costs for OSS.

In the study conducted by the Australian-based company Cybersource [78], it has analyzed the cost savings that can be made as a result of using OSS that provides similar functions instead of COTS software marketed by Microsoft in any fictional company. The analysis of the Cybersource company has revealed the possible amount of savings by using the license cost of the software packages. That is, the initial purchase prices were considered in this analysis. This analysis was performed in 2002, and therefore old versions and prices of the software were used. However, we have created Table 2.1 and Table 2.2 using current versions and prices of this software for a fictional company employing 50 persons. In this fictional company, it is assumed that standard office software, e-mail, intranet and internet services and database access are provided for each of the employees, as well as workstations for a limited number of experts and developers.

In Table 2.3, the results obtained in Table 2.1 and Table 2.2 are rearranged for two separate companies with 100 and 250 employees. As can be seen from Table 2.1, the initial purchase price of companies using COTS software is quite high. Moreover, as shown in Table 2.3, these costs increase as the size of the companies increases. However, the costs of companies using OSS are quite low, and these costs remain constant regardless of the size of the organization, as shown in Table 2.3. This situation is accepted as an indication that OSS has extremely high scalability.

The maintenance cost of any software package can be equal to the initial cost of ownership and often more than the initial cost of ownership. In addition to the initial cost of ownership, OSS also has significant advantages over COTS software in terms of upgrade, update and maintenance costs. One of the most important reasons for this is that the long-term pricing of the services mentioned above remains at the discretion of a single supplier in COTS software. However, even if technical support is paid for OSS, the market for these services is open to competition. Therefore, since there are many companies that will provide technical support to the software, another one can always be preferred instead of a technical support provider whose services are not liked or whose fees are exorbitant.

Table 2.1. The cost of Microsoft solutions [78]

Software	Number of licenses	Price
Norton Antivirus 360	50 copies	$2,799.5
MS Internet Information Server	2 copies	$0.00
MS Windows Server 2022	5 copies	$24,478.56
Sitecore Commerce Server	1 copy	$13,345.00
MS Forefront TMG	1 copy	$1,499.00
MS SQL Server	1 copy	$5,434.00
MS Exchange Standard Server 2019	1 copy	$299.00
Windows 11	50 copies	$9,950.00
MS Visual Studio Professional	3 copies	$3,597.00
MS Office Standard	50 copies	$13,200.00
Adobe Photoshop	2 copies	$1,399.98
Total		**$76,002.04**

Table 2.2. The cost of OSS solutions [78]

Software	Number of licenses	Price
GNU/Linux Distribution (Red Hat)	Only 1 copy required	$349.00
Apache Web	Provided with distribution	$0.00
Squid Proxy Server	Provided with distribution	$0.00
PostgreSQL Database	Provided with distribution	$0.00
Iptables Firewall	Provided with distribution	$0.00
Sendmail or Postfix (mail server)	Provided with distribution	$0.00
KDevelop (IDE)	Provided with distribution	$0.00
GIMP (graphics)	Provided with distribution	$0.00
OpenOffice (productivity suite)	Provided with distribution	$0.00
OSCommerce (e-commerce system)	Only 1 copy required	$0.00
Total		$349.00

Table 2.3. OSS solutions savings versus Microsoft solutions [78]

Size of companies	The solution of Microsoft	The solution of OSS	Amount of savings
Company A (50 employees)	$76,002.04	$349.00	$75,653.04
Company B (100 employees)	$152,004.08	$349.00	$151,655.08
Company C (250 employees)	$380,010.2	$349.00	$379,661.2

2.1.3.3. Reliability

Commercially available software products may have backdoors. A backdoor is a kind of method that ignores the normal security functioning of computer systems, thereby leaving the computer system open to unauthorized access and operations [79]. This is a very critical situation, especially in companies where information security is important. Considering that the source codes of the product used in COTS software are only accessed by the producer, it is not possible for users to know what the software product does in the background. No one can guarantee whether any part of the code that can harm the user,

steal their information and send it to other units is included in the software [80]. Companies that develop COTS software sometimes make statements that there is no backdoor in their software, and there are even companies that open some part of the source code of the software for examination [57]. However, these efforts are often not enough to remove the uneasiness about the issue. In summary, the source code of OSS products can be completely examined, but in traditional software, it is necessary to trust the statement of the software producer. For this reason, in institutions where information security is considered important, there is a greater tendency towards OSS products.

2.1.3.4. Vendor Lock-in

In COTS software, manufacturers oblige users to use their own software products with various agreements for a certain period of time; software other than the software products of that company cannot be used. Another problem is that the COTS software products provided by the manufacturers to the user only work in harmony with their own software products and do not match with the software products of the competitors. This situation restricts users. Thus, manufacturers aim to eliminate competition, set prices as they wish, and make users dependent on them. These and similar problems have been completely eliminated in OSS products. Since the source codes of the OSS products are accessible to the users, the vendor lock-in is eliminated, and the users can easily change the software products according to their own needs.

2.1.3.5. Software Quality

In recent years, OSS products have emerged as high-quality [57][74]. One of the most important reasons for this is that OSS products are scrutinized by software developers from all over the world and are free of errors [57][60]. Although the OSS is getting better in terms of product quality every passing day, it has not yet reached the level of commercial software in terms of documentation quality. The biggest reason for this is that OSS teams are strong in terms of code development, but they are not very meticulous about the preparation of help documentation.

2.1.4. OSS Licenses

An OSS license allows the source code, blueprint, or design to be used, modified, and/or shared under defined conditions and terms [81]. Considering the most specific feature of OSS is that its source code is open to users, license type is quite important as an essential property in terms of redistributing the source code. Before making modifications to the

OSS, it should be examined which type of changes the software license allows in order not to be subject to legal sanctions. There are over 80 OSS licenses in the literature, but they can be differentiated between permissive and restrictive (so-called copyleft) licenses. A permissive license (e.g., Berkeley Software Distribution [BSD]) provides more freedom for reuse, modification, and distribution [57]. A restrictive license (e.g., General Public License [GPL]) provides the same permission as a permissive license but requires that any derivative software be released under the same license as the original software [82]. That is, despite the fact that all OSS licenses allow using, distributing, changing, and redistributing the source code, the restrictions of license types are different. In Table 2.4, the most used OSS licenses in the literature are given with their authorizations. As seen in the Table, each type of OSS license has different permissions, conditions, and limitations that must be followed. The authorizations allowed by the licenses can be followed from the Table 2.4.

Table 2.4. The OSS license types and their authorizations [55][82]

	Type of license		**Permissions**					**Conditions**				**Limitations**	
License name / Authorizations	Permissive	Copyleft	Commercial use	Distribution	Modification	Private use	Patent use	License and Copyright Notice	State changes	Same license	Disclose source	Liability	Trademark use
Apache License 2.0	✓		✓	✓	✓	✓	✓	✓	✓			✓	✓
MIT license	✓		✓	✓	✓	✓		✓	✓			✓	
Berkeley Software Distribution	✓		✓	✓	✓	✓		✓	✓			✓	
Eclipse Public License	✓		✓	✓	✓	✓	✓	✓	✓			✓	
Microsoft Public License	✓		✓	✓	✓	✓		✓	✓			✓	
Mozilla Public License 2.0		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
GNU General Public License v3.0		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	
GNU Affero General Public License v3.0		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Open Software License 3.0		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

2.2. Software Quality Models and Meta-models

In this section, the current situation of OSS evaluation in literature before the OSS-QMM has been developed is summarized. In this context, we first discuss the current situation of OSS quality models and then the current situation of OSS meta-models.

2.2.1. Analyzing the Current Situation of OSS Quality Models

Software quality is vital for diverse types of organizations, so developing high-quality software in a cost-effective and timely manner has become a major challenge in software engineering [83]. This is not a current issue, and studies have been conducted on software quality for years. As technology in the software industry is constantly evolving, expectations of software quality are constantly changing. Therefore, an array of quality models is observed to measure and evaluate software quality, and the evolution of them over the years is shown in Fig. 2.4. As seen from the figure, quality models are classified as basic quality models developed until 2001 and tailored quality models developed after this year [2][84]. Detailed information is given in Section 4.2.2.4 for both basic and tailored quality models.

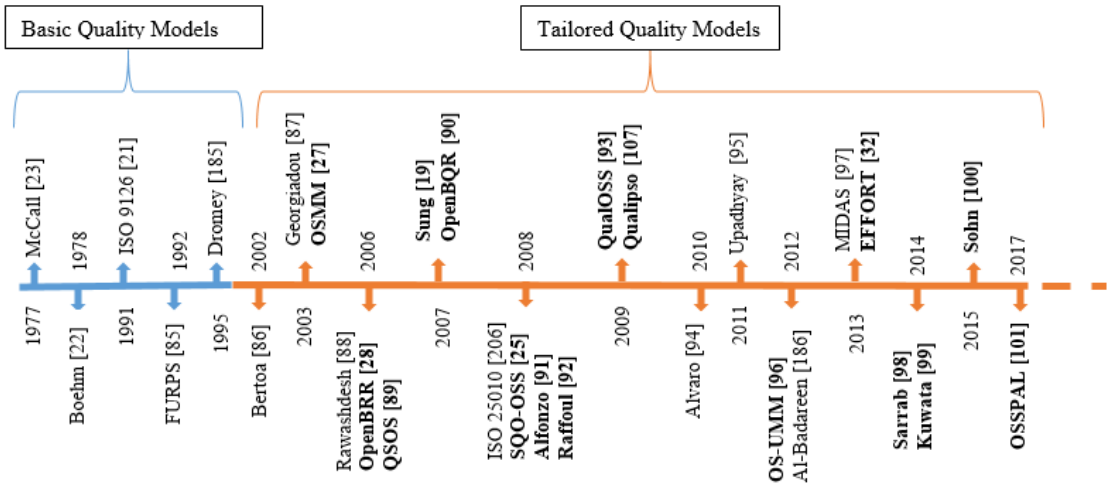


Figure 2.4. Basic and tailored quality models over the years (OSS quality models are indicated in bold text)

Many basic quality models, which are widely used, accepted, and well established, such as ISO 9126, McCall, and Boehm, have been proposed to evaluate software quality. After the emergence of OSS, it has been observed that these models do not provide sufficient support for assessing the quality of OSS [5][15][24-26]. This is because these models mostly have been adapted to commercial software (COTS) and have overlooked some specific properties of OSS, for example, community-based aspects. To fill this gap, researchers and practitioners have developed an array of quality evaluation models or frameworks (QEMoF) that are tailored for the quality evaluation of OSS, such as SQO-OSS, OSMM, OpenBRR, and QSOS.

As the number of OSS quality models increases, secondary studies have been conducted to see and analyze all of the OSS quality models developed in the literature, to obtain more detailed information about these models, and to see future suggestions discussed in these studies. In this context, a total of three SLR studies, one of which is within the scope of this thesis by us [30] and the other two are conducted by Adewumi et al. [24] and Lenarduzzi et al. [29], are revealed in the literature. In the following paragraphs, the contributions of our SLR [30] that differ from the other two SLR [24][29] will be discussed. In addition, the current status of the OSS quality models will be discussed according to the results obtained from these SLR studies. Apart from these SLR studies, there are comparison studies [20][102] and descriptive review studies [4][26] comparing the strengths and weaknesses of OSS quality models in the literature. However, they examined two or more OSS quality models comparatively from certain aspects instead of systematically investigating the existing studies in the literature by following formally defined SLR protocol. Therefore, since these studies do not provide an opportunity to analyze OSS quality models in depth from a wide perspective as in SLR studies, they are not included in the comparison below.

Table 2.5. Comparison of the SLR studies by year, search string, number of primary studies, and research questions.

Study /Year	**Search string**	**# of primary studies**					**RQs and Sub-RQ**
		Models	Frame -work	Survey	Lesson learned	**Total**	
Adewumi et al., 2016 [24]	*"(Open Source Software* **OR** *libre* **OR** *OSS* **OR** *FLOSS* **OR** *FOSS*) **AND** (*model* **OR** *quality model* **OR** *measurement model* **OR** *evaluation model*)*"*	19	-	-	-	19	Research questions (RQ1. key quality characteristics, RQ2. selection methods, RQ3. application domain)
Lenarduzzi et al., 2020 [29]	*"*(*evaluation* **OR** *selection* **OR** *adoption* **OR** *evaluation model** **OR** *selection model** **OR** *adoption model**) **AND** (*Open Source Software* **OR** *OSS* **OR** *FLOSS* **OR** *Libre Software* **OR** *Free Software*)*"*	35 (including evaluation, adoption and selection models)	-	20	5	60	Research questions (RQ1. factors mainly discussed, RQ2. common factors considered)
Our SLR, 2021 [30]	*"*(*quality*) **AND** (*evaluation model* **OR** *assessment model* **OR** *measurement model* **OR** *evaluation framework* **OR** *assessment framework* **OR** *measurement framework*) **AND** (*OSS* **OR** *FOS* **OR** *FLOSS* **OR** *open source software*)*"*	26	10	-	-	36	8 RQ and 23 Sub-RQ.

The SLR study conducted by Adewumi et al. [24]: They conducted a systematic literature review of the studies that proposed OSS quality models. A total of 19 quality evaluation models for OSS were selected after searching in scientific digital libraries, as shown in Table 2.5. After the studies were selected, the authors determined some criteria to assess the quality of these studies. The OSS quality models were classified by using three research questions with respect to quality characteristics, the methodology used for the assessment, and their domain of application. The SLR results indicated that half of the quality assessment models for OSS did not consider community-based criteria, although these criteria make OSS different from their proprietary counterparts. Also, the authors concluded that using hierarchical structures was found to be the most popular selection method in the existing OSS quality assessment models [24]. Moreover, an application domain, e.g., data-dominant, computation-dominant, and systems software, was determined by the majority (53%) of the existing evaluation models, such as the study conducted by Sohn et al. [100]. Aside from these OSS quality models, our SLR has also examined quality evaluation frameworks by including up-to-date studies. Within the scope of our study, 8 RQs and 23 sub-RQs were determined to examine QEMoF from wider aspects such as basic characteristics, structure, evaluation procedure, support for evolutionary evaluation, type of data collection, research method, degree of required skills for evaluation, research method, challenges faced while developing the proposal and validation method.

The SLR study conducted by Lenarduzzi et al. [29]: They also conducted a systematic literature review of studies that proposed quality models for the selection, evaluation, and adoption of OSS by focusing on criteria which affect the evaluation of OSS. A total of 60 primary studies, which consisted of 20 surveys, 5 lesson-learned studies, and 35 studies that proposed OSS evaluation models focusing on different technical aspects, were selected after searching in scientific digital libraries, as shown in Table 2.5. The number of models examined in their SLR study [29] (35) is higher than the one (26) in our SLR study because the main focus of the models in that SLR study is not "quality", as can be understood from its search string given in the Table. The search string in their SLR study [29] does not include the word "quality" but the terms "adoption model", "evaluation model", and "selection model". Therefore, in addition to quality evaluation models, the SLR study [29] also revealed different models for OSS, such as adoption cost models [103], risk models [104], etc. Also, that SLR revealed studies that do not propose models

and that are only related to software adoption, e.g., the studies [105], because of their search string. As seen from our search string in the Table, the main focus of this SLR study is "quality", and accordingly, we included the studies that proposed OSS models or frameworks that were only explicitly aimed at evaluating OSS quality. In addition, as stated in the inclusion criteria of the SLR study [29], the authors included some studies that were not peer-reviewed (e.g., blogs, forums, etc.), which were not included as the primary studies in our SLR study. As a result, the numbers of the studies differ between the pools of the two SLRs.

Lenarduzzi et al. [29] identified two research questions in order to: examine the factors that were mainly discussed by stakeholders during the selection process (RQ1) and the factors that were actually assessed by the available evaluation, selection, and adoption models (RQ2). Through the contributions of this study, users could get an overview of how evaluation, selection, or adoption models work and the common criteria used by these models. There are many differences between their SLR [29] and our SLR with respect to RQs. In their SLR [29], the first RQ has three sub-categories that investigate: the scope of the models (classified as quality, adoption and etc.), how they were built (classified as case studies, interviews, and experience), and how they work (classified by checklist and measurement). In our SLR, all proposals are only quality evaluation models or frameworks for OSS products, as different from the first sub-category of RQ1 in their SLR [29]. The other 2 sub-RQs in their SLR were determined to analyze the models from a higher perspective. However, in our SLR study, we elaborated the technical details of QEMoF, and the classification was further detailed. For example, how the models' work was classified as "measurement" in their SLR [29], while in our study [30], details were given such as how these measurements were made using which techniques and how the results were provided. In their SLR [29], RQ 2 examined the attributes, measures, and information that were evaluated in common in the quality models at a high level and without separating them. However, in our SLR, the quality attributes, the metrics, and the aspects of OSS used in evaluations were classified based on the attributes in the ISO 25010 quality model and other accepted studies in the literature. Apart from the RQs mentioned above, our SLR includes additional RQs that examine QEMoF from different aspects such as: tool support and formal representation, referenced quality models, degree of guidance provided for evaluation, support for evolutionary evaluation, type of data collection, degree of required skills for evaluation, research method, challenges faced

while developing the proposal, validation method, and type of application used for validation.

As a result, many inferences about OSS quality models have been obtained according to the common outputs of all these SLR studies. In addition to these common outputs, unique inferences related to OSS quality models were obtained in our SLR study. According to common outputs, it is observed that there is little or no adoption of OSS quality models in practice. This issue has been analyzed in depth in our SLR study. In this context, first of all, evidence for the practical use of the OSS quality models proposed in the primary studies was investigated. To access this evidence, all studies belonging to academic and grey literature citing the OSS quality models were searched. As a result of this research, only one study [106] was found that can be considered as evidence for the practical use of OSS quality models (for QualipSo [107]) in the industry. In this study, they have presented an overview of the usage of QualipSo for the evaluation of the quality of OSS and also SCRUM as a development methodology, for addressing the development needs of the Italian Army. Then, in our SLR study, the reasons for the little adoption of OSS quality models in practice were investigated in detail. The reasons are that OSS has a dynamic and diverse nature, and accordingly, its quality is affected by various data (e.g., the longevity of the project and the mailing list density). These data are scattered in a variety of databases and heterogeneous sources, and defining and evaluating the quality of OSS are considered challenging. In addition, the number of individual OSS quality models in the literature is high, and this has caused the structure of the developed quality models to be heterogeneous. Therefore, defining situation-based procedures for determining evaluation criteria becomes a challenging task. Accordingly, evaluation results obtained from different quality models for the same purpose can be incomparable and unreliable [30][35-36]. This negatively affects standardization, which is an important communication vehicle for companies when interoperating with others. For all these reasons and more, in our SLR, it was concluded that a software quality meta-model is needed to evaluate OSS quality. This is because of the ability of a meta-model to allow the development of multiple OSS quality models with a homogeneous structure and common terms using the same modeling language it proposes. Thus, it is aimed to provide standardization in software quality measurement and to compare the evaluation results obtained with different OSS quality models. Further results and their details from our SLR study are provided in Section 3.2.

2.2.2. Analyzing the Current Situation of OSS Quality Meta-models

In this section, the current situation of meta-models for OSS quality is explored based on a literature search. The SQMMs are important because "*they allow standardizing quality models, and thus, they create a common understanding between stakeholders for proper quality management throughout the entire life of a software product*" [34]. Also, the "*they allow us to see a complete picture of software quality and to represent concepts of software quality more formally*" [34]. In this context, we carried out an SLR study for the first step of the systematic process, as shown in Fig. 1.1. In the literature, only our systematic literature review (SLR) study [37] is revealed, and this study addresses the content and structure of the meta-models that were proposed for software quality and its evaluation (SQiE). The main motivation of this SLR study is that meta-models have an important place in the harmonization, standardization, and consistency of quality models. Within the scope of this study, a total of 28 studies were examined, and only 2 of these studies were classified as meta-models that allow evaluation of OSS quality.

In this SLR study, meta-models were analyzed from many aspects by using 7 RQ and 19 sub-RQ, such as basic characteristics, challenges faced in developing meta-models, means of data acquisition as defined in the meta-model, validation method, etc. However, the most important motivation and contribution related to our current study are that it provided information on how the structure and content of models are organized, and also addressed the terms used in the meta-models, and provided the frequency of use of these terms. Our SLR study has an important contribution in shaping the meta-model to be developed within the scope of this thesis. Also, this SLR study allowed us to explore the shortcomings of SQMMs. Despite the fact that SQMMs are important for specifying and evaluating the quality of software products, our SLR indicated that OSS-QMM are not at the desired level in terms of mainly four deficiencies;

1. Few numbers of OSS-QMM and their adoption in the community,
2. Lack of depth in their content,
3. Inconsistent concepts among them, and
4. Lack of validation in a real context.

In this thesis study, these deficiencies have been focused on, and a comprehensive OSS-QMM has been developed by eliminating these deficiencies. The other findings regarding existing SQMMs from our SLR study are given in detail in Section 3.1.

2.2.2.1. Studies Proposed the Meta-models for OSS

In our SLR study, a total of 28 studies were examined, and only 2 of these studies were classified as meta-models that allow evaluation of OSS quality. Therefore, in this section, these two SQMMs [108-109] that enable evaluating OSS quality from specific aspects and their differences from the OSS-QMM developed within the scope of the thesis will be explained. In fact, there are no proposed SQMMs to directly evaluate the quality of the OSS in the literature. However, there are studies that adapt existing SQMMs (not OSS-specific) to OSS quality evaluation. These SQMMs are classified as SQMMs that evaluate OSS quality in our SLR study [37] since they enable the evaluation of OSS products from certain aspects and use OSS products in the validation process.

Eghan et al. [108] adapted the existing SE-EQUAM ontology meta-model [110] (not OSS-specific) to evaluate the trustworthiness of open-source external libraries and APIs. Also, they take advantage of a unified ontological knowledge representation of different SE-related knowledge resources [108]. OSS libraries are beneficial in many ways, such as saving time and money, but OSS can have security risks. Therefore, it is aimed to measure the trustworthiness (i.e., security vulnerabilities and license violations) of open-source external libraries and APIs in this meta-model. They used adapted SQMM to evaluate OSS external libraries in terms of trustworthiness by designing a case study. Although Mens et al. [109] also did not propose new SQMMs for OSS quality, they adapted the existing MoCQA meta-model [112] (not OSS-specific) to measure the quality evolution of OSS ecosystems. Therefore, it is classified as an SQMM that evaluates OSS quality in the SLR study [37]. In this context, they instantiated the adapted MoCQA meta-model and, accordingly, developed the Customized Assessment Quality Model (CAQM). This quality model was used to evaluate the quality evolution of OSS ecosystems by designing a case study.

Table 2.6. Comparison of meta-models for OSS

Comparison criteria	Eghan et al. [108]	Mens et al. [109]	The OSS-QMM
Application domain	Specific	Specific	General
Number of concepts	5	15	35
Covering of OSS aspect	Partially	No	Yes
Covering the viewpoint of stakeholder	No	Partially	Yes
Consideration of the inconsistency of terms	No	No	Yes
Consideration of the common structure of quality models	No	No	Yes
Consideration of the mapping terms (Section 4.3)	No	No	Yes

These two meta-models [108-109] were not directly proposed to evaluate OSS quality, but they were obtained as a result of adapting existing meta-models to evaluate OSS from certain aspects. Therefore, although they are useful for the specific aspect of OSS quality, they are not comprehensive and have a deficiency in OSS evaluation from a broad perspective. Some of these deficiencies are listed in Table 2.6 comparatively. As shown in the Table, the application domain of these two adapted models is limited. That is, this adapted meta-model [108] was proposed to evaluate the trustworthiness of open-source external libraries and APIs. Another one [109] was proposed to evaluate the evolution of OSS ecosystem quality. However, the OSS-QMM has a flexible structure to evaluate the quality of OSS from desired perspectives. The number of concepts covered by meta-models supports this situation. The number of concepts in the OSS-QMM is saliently greater than the others, as shown in the Table. This indicates that the OSS-QMM enables an opportunity to evaluate OSS quality from a wider perspective, unlike SQMM with a specific application domain.

The unique feature that distinguishes the OSS from other types of software is that it stores many evaluation data belonging to the community-based aspect in various databases. The community-based aspect is not covered by this meta-model [109] and is partially covered by this model [108], as shown in Table 2.6. "Partially" means that the meta-model includes the data of both aspects (i.e., code-based and community-based) in the evaluation, but it did not specify to what extent the data belonging to each OSS aspect will affect the evaluation result. One of the most important reasons for this is that this meta-model [108] did not cover the viewpoint of an evaluator. For example, considering that the evaluator is a developer and will shape the OSS product according to his own needs, the code-based aspect may be more important for this evaluation. Thus, we added the concept of *weighting* to assign weight to OSS aspects using the concept of *the*

weighting method in the OSS-QMM. In this way, the evaluator can specify the effect of each aspect on the evaluation result. In addition, since this meta-model [108] has a specific application domain, it only includes data belonging to OSS aspects for evaluating the trustworthiness of open-source libraries and APIs.

Although Mens et al. include the concept of "viewpoint" in their meta-models, and consider it only in determining the quality factors. Therefore, this is considered as "partially" in Table 2.6. However, Eghan et al. did not consider viewpoints in their meta-model. Since the evaluator will shape or use the OSS products according to their own needs, the viewpoints of the evaluators are essential and should be considered in the meta-model. However, the meta-model should not allow heterogeneity in the structure of the OSS quality models to be derived while considering the viewpoint. Therefore, adhering to this rule, the "viewpoint" was taken into account in determining the importance of OSS aspects on quality evaluation, the quality characteristics to be evaluated, the importance of sub-attributes on quality evaluation, and the impact of the measurable concept on the quality in the OSS-QMM. These concepts mentioned here are explained in Section 5.3.

In addition to the concepts given above, we have added many concepts different from others, such as normalize measure, impact, evaluation aggregation, etc. (see Fig. 5.10) to the OSS-QMM. For example, the quality of OSS is affected by heterogeneous data from various sources. Therefore, according to SLR [30] results supporting this situation, the important challenge for OSS quality models is that OSS has a dynamic and diverse structure, thus aggregating heterogeneous data from different sources. In this context, we added a "normalize measure" that allows us to normalize each metric. For another example, we added the concept of "impact" because each heterogeneous measurable concept will have a different impact on OSS quality. Additively, the terms of OSS-QMM are classified as specification, measurement, and evaluation to make evaluation more meaningful. Meta-models in the literature ignore evaluation-related concepts that enable the interpretation of measurement results.

In this study, a systematic process was followed during the OSS-QMM development phase. In this context, first of all, inconsistencies between the terms of the SQMM proposed for the custom type of software and OSS were analyzed, as explained in Section 4.1. Then, the common structure of the OSS quality models was analyzed, as explained in Section 4.2.3. Then, terms of meta-model and terms of quality models were mapped. Finally, a level-based OSS-QMM was developed by considering the common structure

of quality models. The SQMMs developed in the literature, including the SQMMs developed by Eghan et al. and Mens et al., did not follow the aforementioned systematic process. Therefore, there may be inconsistencies between the structure and terms of OSS quality models derived using these SQMMs. This situation affects standardization negatively.

2.3. Software Measurement Models/Standards

In this section, general information about the measurement standards or proposals that were taken as the basis of the SQMMs is given, and some inferences are made about them. This information is important for analyses to be conducted in Section 4.1 to eliminate inconsistencies between concepts of SQMMs.

One of the main objectives of software engineering is to release high quality software product to the market. Software measurement is at the core of software engineering since improving the quality of software without measuring is impossible. In this context, a number of international standards and research proposals have been released to measure the quality of software. Standardization is essential for meaningful measurement since it enables comparison measurement results, with the prerequisite that vocabulary in measurement standards or proposals is consistent. The standards and research proposals to measure the quality of software have emerged in time sequence, and some have overwritten the others. Consequently, inconsistencies in their terminology have been reflected in various studies that proposed the SQMMs. Software measurement is an ongoing process, and approaches, methods, and terminologies of software measurement continue to be defined, consolidated and agreed. Important organizations and standardization bodies such as ISO, IEC, and IEEE have developed many international standards for software engineering. There are a large number of international standards developed by only ISO for measuring software processes and products, as presented in Fig. 2.5.

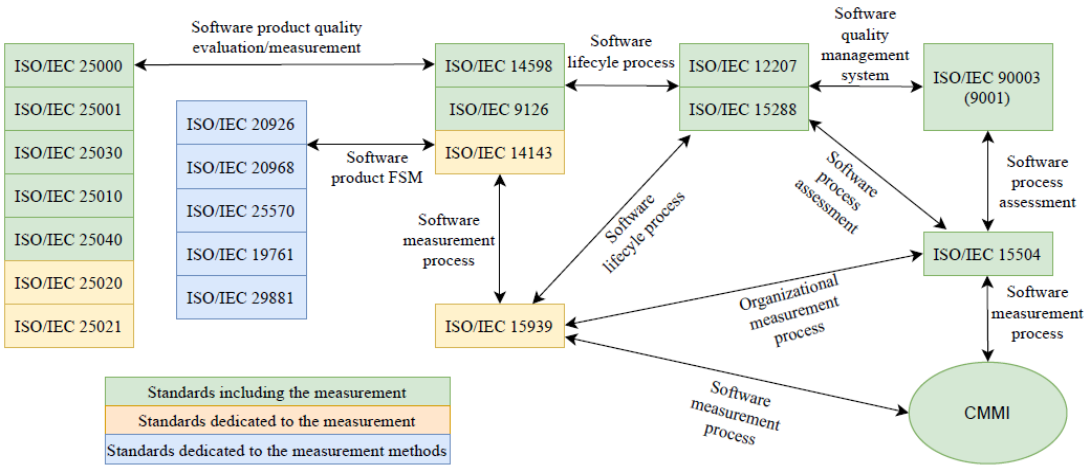


Figure 2.5. Main relationships between the ISO/IEC standards of software quality and software measurement, and their relationship with the CMMI model – as adapted from [112]

Considering also the international standards proposed by organizations other than ISO and the research proposals related to software measurement, it is not surprising that there is inconsistency in the concepts and terminology used in this field due to a large number of sources. Terminology conflicts and inconsistencies appear not only among the international standards of different organizations but also among those of the same organization [39]. Inconsistencies, commonalities, and terminology conflicts in all these sources are reflected in SQMMs because they are created by adopting the terminology and concepts from international standards.

The SLR study [37] that we performed to understand the structure and content of the meta-models for software quality and its evaluation (SQiE) has provided us with the opportunity to investigate the content and terminology of the SQMMs. Complementary to that, this study [54] has attempted to determine the international standards whose terminology is largely referenced by the SQMMs that have been proposed for OSS and a custom type of software. In accordance with this purpose, ISO/IEC 14598 (Software Engineering-Product Evaluation) [113], ISO/IEC 15939 (Software Engineering-Software Measurement Process) [114], and VIM (International Vocabulary of Basic and General Terms in Metrology) [115] are determined from ISO and IEC organizations. Also, IEEE 1061 (Software Quality Metrics Methodology) [116] and IEEE 610.12 (Standard Glossary of Software Engineering Terminology) [117] are determined by IEEE. Descriptions of these international standards are overviewed in Table 2.7. The use of the

terminology of these standards in the SQMMs and terminology conflicts among the standards will be discussed in Section 5.1.

In addition to the standards mentioned above, some research proposals by Kitchenham [118], Briand [119], and Kim [120], all related to software measurement, are included in this thesis as their terminology has been adopted by some SQMMs such as [44]. Thus, terminology conflicts among the research proposals related to software measurement are also addressed in this work. Descriptions of these proposals are included in Table 2.7, as the descriptions of the international standards.

Table 2.7. Standards and proposals whose terminology is referenced by SQMMs

Standard/ Proposal	C1	C2	C3	Description
IEEE 610.12 [117]	Y	N	N	It is a standard that is used as a glossary of Software Engineering terminology. This standard focuses on the definition of terms only, regardless of their relation to software measurement.
VIM [115]	Y	N	N	It is a standard that includes many terms of subjects related to software measurement. Although it is not focused on software mainly, it is used to define software measurement concepts in literature by many studies since terms are defined completely and in detail.
IEEE 1061 [116]	P	P	P	It is a standard that enables us to obtain quality requirements and also enables us to identify, implement, analyze and validate for quality measures of software. It can be used by all types of software in any phase of the software life cycle. It contains terms from all categories but not completely from each category.
ISO/IEC 14598 [113]	Y	P	N	It is a standard that enables us to measure, assess and evaluate the quality of software products. It enables us to perceive the evaluation process from different points of view, such as acquirers, developers, and evaluators.
ISO/IEC 15939 [114]	P	Y	P	It is a standard that enables us to define the approaches needed for identifying, defining, selecting, applying, and improving software measurement. It also defines some measurement terms that are commonly used in the software industry. It covers two main components as software measurement process and measurement information model. The software measurement process is established by the information needs of the organization. The measurement information model provides a relationship between information needs and measures. It describes how quality attributes are measured and how decision-making is performed by using indicators.
Kim [120]	N	N	Y	It is a measurement ontology that allows organizations to evaluate whether they comply with the ISO/IEC 9000 standard. It is not proposed primarily for software processes and products, but it covers many terminologies that can be used for measurement processes.
Kitchenham [118]	N	N	Y	It is a conceptual model that covers the definition of many software measures and the relationships among them. It consists of three components; first, generic components which define concepts; second, development model components that provide the link between measures and entities; and third, project domain components that present the metric values obtained from projects and link them to actual instances of the entities.
Briand [119]	N	N	Y	It is an approach that is based on the GQM approach [121] and defines measures of software product attributes. The primary goal of the approach is not to define the concepts but to represent their use in the GQM process.

***C1:** software measures, **C2:** measurement process, and **C3:** target and goals. ** **Y:** Yes, **N:** No and **P:** Partially

The international standards and research proposals related to software measurement and quality can be classified under three main categories according to the particular topics they address [39]: software measures, measurement processes, and targets and goals, which are respectively denoted by C1, C2, and C3 in Table 2.7. The first category of software measures (C1) focuses on main elements, such as measures, the unit of

measurements, scale, etc., in the definition of software metrics. The second category of the measurement process (C2) focuses on the definition of terminology related to software measurement act such as measurement methods, measurement results, etc. The third category of target and goals (C3) focuses on gathering concepts related to objectives and the scope of the measurement process, such as attributes, measurable entities, information needs, etc. The categories that the standards or proposals address are denoted in columns 2-4 in Table 2.7. In the Table, "Y" (yes) means that the standard or proposal covers the majority of the terms in that category, "P" (partially) means that the standard or proposal covers some of the terms in that category, and "N" (no) means that the standard or proposal does not contain any of the terms in that category. It is seen from the Table that there is no single standard or recommendation that completely covers all the categories of C1, C2, and C3 [39]. Here, it is important to note that the terminologies of standards or research proposals that focus on the same category are not homogeneous.

Apart from the standards and research proposals that are listed in Table 2.7 and examined within the scope of this study, there are important models such as CMMI (Capability Maturity Model Integration) [122] and standards such as ISO/IEC 12207 (Standard for Software Life Cycle Processes) [123] and ISO/IEC 15504 (Standard for Software Process Assessments) [124] which was lately revised by ISO/IEC 33000 series. The relationships between all these ISO/IEC standards with respect to covering software quality and measurement, and the relationships of these standards with CMMI are also represented in Fig. 2.5. It should be noted that the standards or models other than those listed in Table 2.7 have not been analyzed in this study although some standards were withdrawn, e.g., ISO/IEC 14598 was replaced by ISO/IEC 25040 later. One reason for this is that the SQMMs examined within the scope of this study have been created based on the standards or proposals investigated in Table 2.7, as they mentioned in their studies, as required by the years of publications. Therefore, the most important factor for a standard or proposal to be included in this study is that its terms have been adopted by the SQMMs. Another reason is that some of the standards or models not included in this study have been defined using the terms of the standards or proposals examined in this study and that some of them define only the terms specific to certain domains. For example, CMMI adopts the terminology of the ISO/IEC 15939 standard [114], and functional size measurement (FSM) standards (e.g., ISO/IEC 14143 [125] and ISO/IEC 19761 [126]) are totally aligned with VIM [115] (International Vocabulary of Basic and General Terms in

Metrology). Also, some standards contain terminology specific to the particular domain that is not adopted by the SQMMs concerned in this study. For example, ISO/IEC 15504 standard includes terminology such as "software process target" and "software process metric", which have been adopted to the process assessment domain.

2.4. Meta-modeling

In this section, the basics of meta-modeling are explained through examples to better understand the logic of meta-modeling. In this context, first, the basics of modeling are explained, as a model is an instance of a meta-model. Then the basics of meta-modeling are explained.

2.4.1. Basics of Modeling

In order to understand meta-models well, it is necessary to understand the basics of models first because meta-models are the abstract form of models. Models can be used in many areas and, therefore, have many definitions in the literature. For example, the most general definition was made by Benyon [127], *"A model is a representation of something, constructed and used for a particular purpose.*" Also, Sprinkle et al. [128] defined it as "a *powerful vehicle to explain the behavior, structure, and other features in all areas of engineering, in mathematics, or each of hard sciences"*. Also, it is defined as "*a description of a system, and it must be written with a well-defined language*" [44]. *Modeling* is the practice of creating a model. The person who creates a model is the *modeler.* The person who uses a model is the *interpreter.* The interpreter and modeler have a specific *purpose* for using and constructing the model. Every model has to represent *something* in the real world.

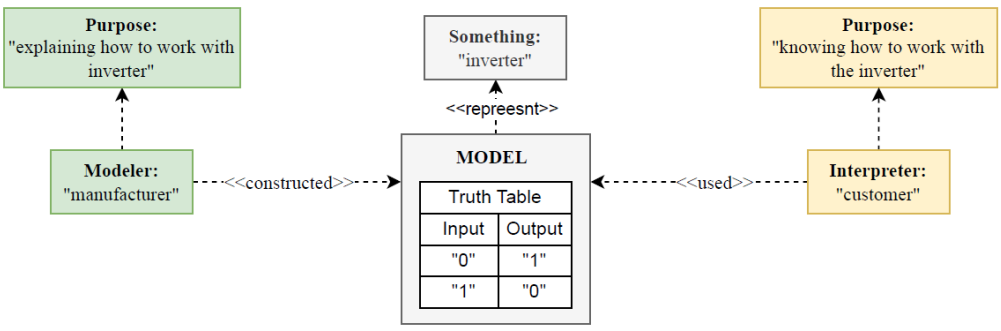


Figure 2.6. The example use of a model over the inverters manufacturer

The model alone has no meaning, and its meaning emerges with the situation and purpose in which the model is used. This is expressed by Stachowiak as the pragmatic feature of the model, as shown in Fig. 2.6 [129]. This is also the case for data. That is, the data itself has no meaning, but with an interpretation, the meaning behind it can be deduced and understood. In Fig. 2.6, the pragmatic features of the model are exemplified for better understanding. In this example, an inverter manufacturer would like to construct an inverter model. The truth table is a *model*, and it shows the case of the inverter. The inverter is *something* represented by a model, and the manufacturer is the *modeler* who constructs a model. It is the *purpose* of the modeler to explain the operation of the inverter. This model is used by a customer who has to understand how to operate the inverter. Here, the customer is the *interpreter*. It is the purpose of the interpreter to know the operation of the inverter. The visualized example in Fig. 2.6 explains the basics of the model, the process of its creation, and its meaning for the modeler and interpreter.

2.4.2. Basics of Meta-modeling

The "meta" is a word of Greek origin, meaning beyond or about, and is used to describe something. Meta-model is defined by Seidewitz [130] as "a model that represents a modeling language". Moreover, many definitions of the meta-model have been made in the literature, all of which have almost the same meaning. Some of these definitions are given below. A meta-model is;

- *"A model of a well-defined language [44]."*
- *"A model of models [34]."*
- *"A model that defines the language for expressing a model [131]".*
- *"A specification model for a class of system where each system in the class is itself a valid model expressed in a certain modeling language [132]".*

As can be understood from the definitions, meta-models are an abstract form of models. Therefore, multiple models can be derived from a meta-model. In Section 2.4.1, models are defined as a description of a system, and they must be written with a well-defined language. An important question arises in this context: "How do we define such a well-defined language?". At this point, the importance of "meta-modeling" is revealed. A meta-model must be expressed in a well-defined language, as with models, since a meta-

model is also a model. This well-designed language is called meta-language, as shown in Fig. 2.10.

The Backus Naur Form (BNF) meta-language has been used to describe the syntax of computing notation, such as programming languages, since the 1950s. In computer science, BNF is a meta-syntax notation format generally used to describe a programming language by the developer of programming languages. BNF was mostly developed for text-based languages, e.g., programming languages. Therefore, BNF can be used in modeling languages where the modeling language is expressed in text-based terms. However, modeling languages don't need and usually don't use text-based languages. Since they often use a graphical syntax such as UML, a different type of mechanism is needed to define language. Therefore, the ***meta-modeling*** mechanism has arisen to define graphical-based modeling language.

The language consists of 5 aspects: concrete syntax, abstract syntax, syntax mapping, semantic mapping, and semantic domain [133-134], as shown in Fig. 2.7. In this context, the abstract syntax is represented by the meta-models.

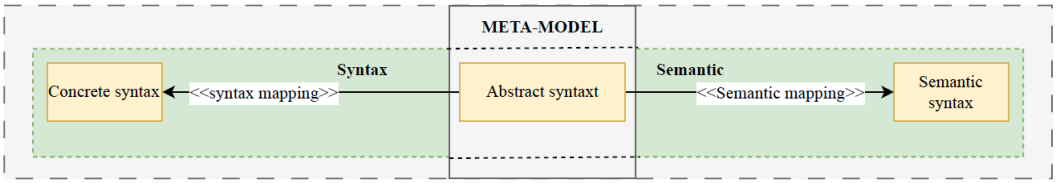


Figure 2.7. Meta-model that represents a language

Let's continue with the same example given above to explain this further. In Section 2.4.1, an example inverter model is constructed, as shown in Fig. 2.6. Assume that this inverter model will be used to develop a meta-model. This meta-model will represent the modeling language for modeling inverters and contains the abstract syntax of this modeling language, as shown in Fig. 2.8.

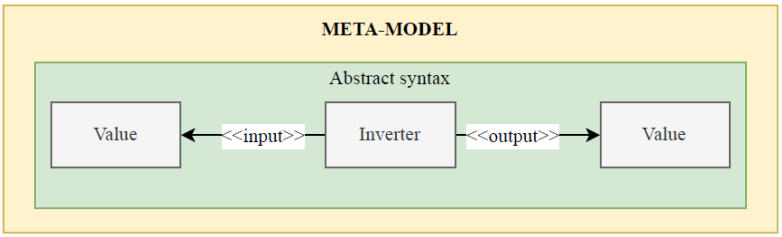


Figure 2.8. Meta-model that represents abstract syntax modeling language for modeling inverter

In meta-modeling, the concept of well-designed meta-models should map the concepts of models. Let's continue with the same example given above to explain this further. The example inverter model shown in Fig. 2.6 represents the semantic domain. Also, the example meta-model shown in Fig. 2.8 represents the abstract syntax. Therefore, a semantic mapping should be performed between the semantic domain and abstract syntax, as shown in Fig. 2.9. As shown in the figure, each concept of the meta-model is mapped to terms of models. This figure indicates that a model is an instance of a meta-model.

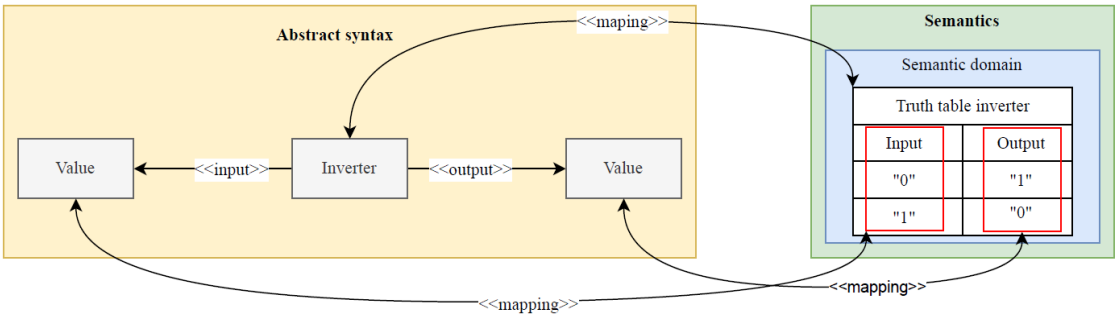


Figure 2.9. The example representation of the mapping process between the concepts of inverter meta-model and inverter model

2.4.2.1. Meta-model Hierarchy

Object Management Group (OMG), founded in 1989, has carried out laborious efforts to define and develop meta-models and has developed various standards (e.g., UML, MOF) in this context [134-135]. The hierarchy of modeling defined by OMG is important to understand the relationship between OMG standards. According to the OMG, the modeling structure consists of a four-layer architecture, as shown in Fig. 2.10. This layer is called: M0, M1, M2 and M3. These layers are explained with an example as follows:

Layer M0: Runtime Layer (The Instances):

This layer (M0) determines what will be modeled [136]. In other words, this layer contains the running system where the real data is present. For example, this data may belong to a customer or order, as shown in Fig. 2.10. These examples are the customer named "Harry Kane" living in "Manchester, UK" and the customer named "Tyler Adams" living in "Boston, USA." There can be many customers or orders like the ones in this example in the M0 layer. Assume that we are modeling a business; the examples of this layer (i.e., M0) will be elements in the business itself, such as the invoices, the actual people, or the products. Assuming that we are modeling a software, the examples of this layer (i.e., M0) will be the software representations of real-world items such as the

computerized version of the product information, the invoices or the orders, and the personnel data [137].

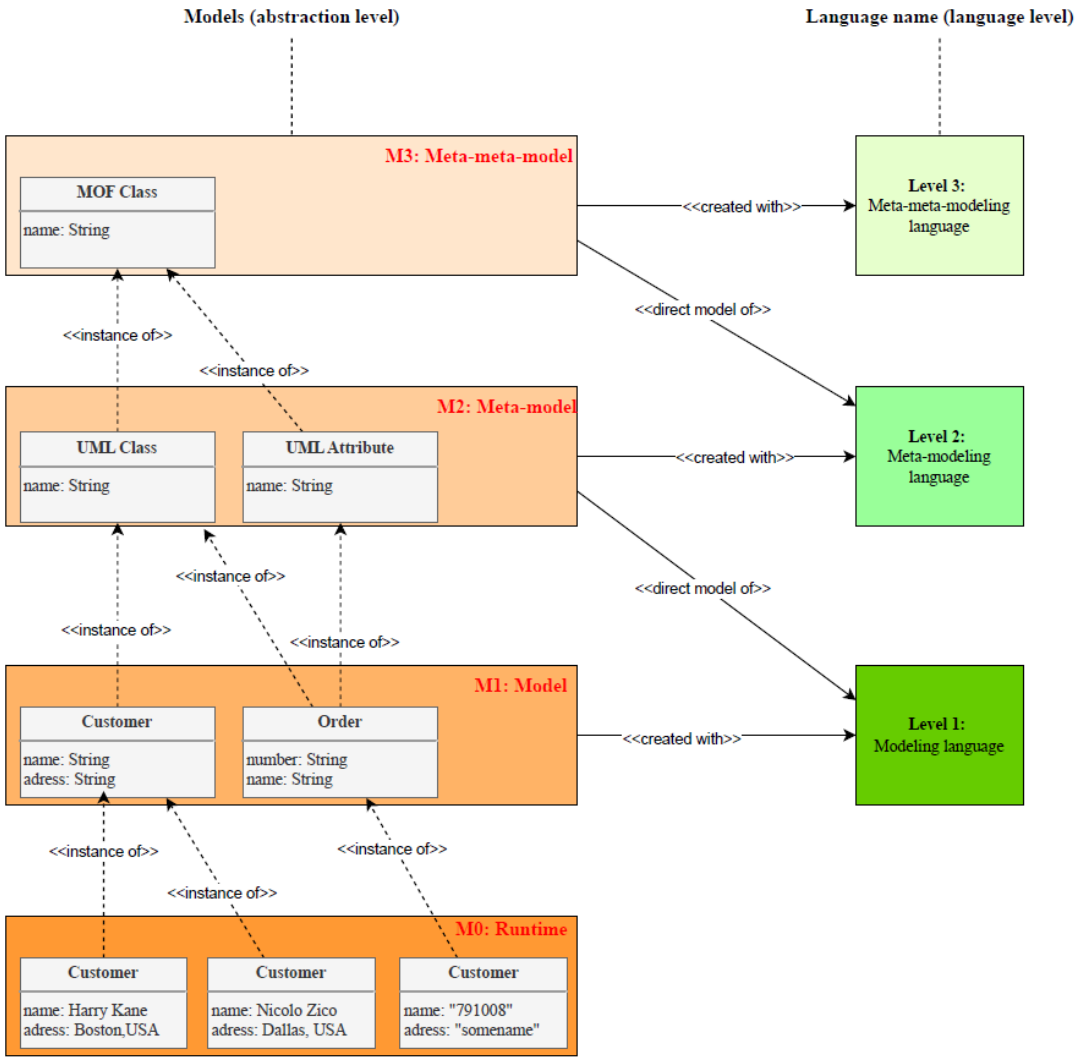


Figure 2.10. Abstraction levels of models and levels of modeling languages.

Layer M1: Model Layer:

This layer (M1) covers models such as a UML model of a software system [136]. For example, as shown in Fig. 2.10, in this layer (M1), the concept "customer" is defined by using its name, street, and city. Layer M0 and layer M1 have a definite relationship [136]. The elements at layer M1 is a classification or categorization of terms at layer M0. Similarly, a term at layer M0 should be an instance of a concept at layer M1. For example, as shown in Fig. 2.10, the customer named "Harry Kane" and "Tyler Adams" at layer M0 is an instance of the concept "Customer" at layer M1. The elements of layer M1 identify what the instances at layer M0 look like. More specifically, the UML model for the

customer class at the M1 layer describes what the instance of a customer at layer M0 looks like.

Layer M2: Meta-model Layer:

The terms at layer M1 (e.g., attributes, classes, etc.) are instances of concepts at layer M2. The concepts of layer M2 identify the terms of layer M1. In other words, the same relationship between layers M0 and M1 exists between layers M1 and M2. The elements at layer M2 is a classification or categorization of terms at layer M1. To explain more concretely, layer M2 contains a model, and this model is called a *meta-model*. Each UML model in layer M1 is an instance of the UML meta-model. That is, a running system corresponds with layer M0, a model of a running system corresponds with layer M1 (i.e., model), and a model of a running system corresponds with layer M2 (i.e., meta-model). To create a meta-model, it is necessary to define a language such as UML or CWM (Common Warehouse Meta-model)

Layer M3: Meta-meta-model Layer:

As in the previous layers, the concepts at layer M2 (i.e., meta-model) are instances of concepts at layer M3 (i.e., meta-meta-model). In other words, the same relationship between layers M0 and M1, and between layers M1 and M2 exists between layers M2 and M3. The concepts at layer M3 are a classification or categorization of concepts at layer M2. At this highest layer, M3, an abstract language such as UML or CWM is used to define the meta-model at layer M2. This is the MOF (meta-object facility) language, which is explained in Section 2.4.3 in detail. In summary, according to the information given at each level above, elements at any level are a subset of elements at a lower level, as shown in Fig. 2.10. For example, the elements belonging to layer M3 are a subset of elements belonging to layer M2.

2.4.3. Meta-Object Facility

The OMG, which is a computer industry standards consortium, was founded in 1989. It is a membership-driven, non-profit organization with members from 27 countries and over 230 organizations. The goal of OMG is to offer a solution to reduce cost and complexity and accelerate the release of new high-quality software products. In line with this purpose, they developed an architectural framework together with a comprehensive interface specification. The aim of these specifications is to obtain reusable, portable, interoperable software components, which are based on standard OO (object-oriented)

interfaces. In software engineering, the MDE (Model Driven Engineering) plays an important role in achieving these goals. The MDE focuses on developing and utilizing domain models that are an abstract representation of all the subjects relating to a particular issue. In this context, OMG has proposed the Model Driven Architecture (MDA) approach for carrying out the MDE paradigm. The OMG has proposed a set of standards to increase the productivity of MDA. Therefore, MDA is a set of standards, such as Meta-object facility (MOF), Unified Modeling Language (UML), and XML Metadata Interchange (XML). Among them, the core element of MDA is MOF among these standards [133][137]. Therefore, we decided to use the MOF standard to develop the OSS-QMM within the scope of the thesis.

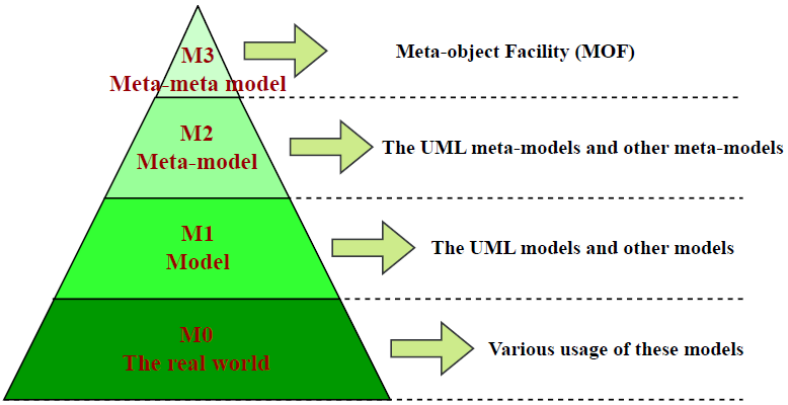


Figure 2.11. The representation of the hierarchy of Meta-Object Facility (MOF)

The MOF standard is used to define modeling languages. According to the OMG, the meta-model hierarchy consists of four layers, as shown in Fig. 2.11. Layer M3 is the top layer, so there is no other layer above this layer, and the MOF is located in this layer. This means that this layer (M3) can only consist of one meta-meta-model, which is the MOF. This layer (M3) is the abstract definition of meta-models at layer M2. For this abstract definition, the MOF uses the MOF language consisting of MOF class and MOF association to define meta-models. Since there is no other layer above layer M3, the MOF is defined using the MOF itself. Therefore, the MOF located in layer M3 contains the definition of UML or CWM and so enables the development of the UML or CWM meta-models at layer M2. Layer M2 can have multiple meta-models, and these meta-models are still abstract. Layer M2 is used to specify UML models at layer M1. Finally, layer M0 (runtime layer) contains objects instantiated from layer M1.

Many meta-models have been developed by OMG for different purposes using CORBA (Common Object Request Broker Architecture) and UML at the layer M2. The UML was

adopted as a standard by OMG in 1997 after its release, and this has made OMG the market leader. Considering this situation, all OMG meta-models were developed using the MOF meta-meta-model, which is a subset of the UML class meta-model [134][136]. This is because MOF-based meta-models have the expressive power of BNF (Backus–Naur Form). The power of MOF-based meta-models is the visual syntax used as a concrete syntax in representing the abstract syntax of the system to be modeled. Also, they use modular structures and have the advantage of standard visualization. Therefore, we developed the OSS-QMM developed within the scope of this thesis as a UML meta-model compatible with the MOF standard. In other words, since UML meta-models are instances of MOF meta-meta-models, the OSS-QMM is a MOF-based meta-model.

2.4.4. Requirements/methods/criteria for Meta-model Validation

In this section, the validation process of software quality meta-models developed in the literature will be analyzed. These analyses will be used in shaping the validation process of the OSS-QMM to be developed within the scope of this thesis. The validation process is one of the most crucial steps in the studies to ensure that the research is correct, clean, and valuable. The validation techniques are generally implemented after the completion of the SQMM development process. In this context, the OSS-QMM (Step 4) has been validated in Step 5, as shown in Fig. 1.2. While determining the validation methods at this validation step (Step 5), the methods widely used for software quality meta-model validation in the literature have been employed. In this regard, the validation methods employed in the validation process of the SQMM were examined in a research question (RQ) of the first SLR study [37]. A total of 28 meta-models were analyzed in this SLR study, and according to the results, the validation methods used in the literature are shown in Fig. 2.12. As shown in the figure, 13 meta-models were validated by designing case studies and six meta-models by performing toy experiments. Also, four studies conducted peer reviews and 1 study used a pilot project application to validate its meta-model. While five studies did not explicitly mention the method of validation for their proposals, only 1 study did not use a validation method. A study might have been validated with peer reviews along with a case study or toy experiment.

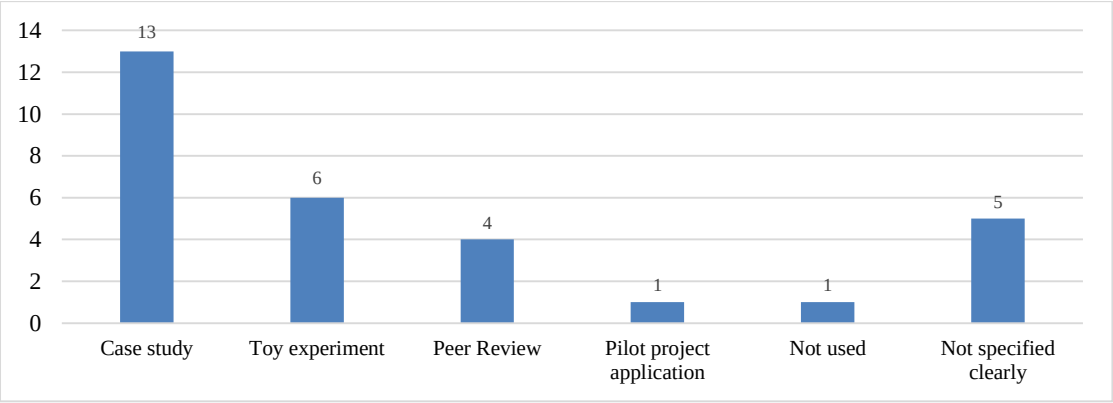


Figure 2.12. Distribution of validation methods used for SQMMs

As can be seen in Fig. 2.12, case studies, toy experiments, and peer reviews by experts are the most commonly used methods for validating meta-models. Therefore, these three validation methods have been used during the validation process of the OSS-QMM. In our study [54], which we have carried out as the output of Step 3, the application of the initial OSS-QMM was demonstrated in the unreal OSS products with dummy evaluation data through the designed toy experiment. In addition to the toy experiment, we have conducted case studies and expert opinions to validate the OSS-QMM in a real-world setting. Unlike the other meta-models, these two validation methods have been applied by considering the application of the OSS-QMM in the real context. Some additional efforts have been performed to apply these two validation methods in a real-world setting. In this context, during the designing of the case study, experts in the industry have been included in the evaluation process, and the evaluations have been performed with real data and OSS products according to the expert viewpoint. During collecting expert opinions, the real-world applicability of the OSS-QMM has been taken into account by experts. Also, they have expressed their opinions by considering OSS-QMs that they used in their own companies or are well known in practice. As a result, the most used validation methods in the literature are taken as a basis for the validation process of the OSS-QMM. Details of the validation process are explained in Section 6.

The purpose of the OSS-QMM is to enable the derivation of OSS quality models with a homogeneous structure and common concepts. Thus, they give quality models the opportunity to perform standard measurements and produce comparable results. For this purpose, a multiple-embedded case study was designed to see if the OSS-QMM fulfills this purpose, unlike the case studies applied in other meta-models. In this context, both a new operationalized OSS quality model and two existing OSS quality models are derived

from our meta-model. These derived quality models were applied in practice in accordance with our meta-model within the scope of the case study. These models were applied using data from real OSS repositories for the same type of OSS products, and the results were compared. Thus, it was monitored whether standard results were obtained as a result of applying the OSS-QMM in the real-world setting.

3. ELABORATION ON SOFTWARE QUALITY MODELS AND META-MODELS (STEP-1 AND STEP-2)

In this section, the findings obtained from the results of the SLR studies [30][37] carried out as the outputs of Step 1 and Step 2 within the scope of the thesis are presented. In the literature, SLR is defined as "*a tool for evaluating and interpreting available research related to a particular research hypothesis, topic area or phenomenon of interest*" [138-141]. Accordingly, in these SLR studies, software quality meta-models [37] and OSS quality models [30] were systematically analyzed in detail with the help of research questions from many aspects. As a result of these studies, empirical evidence and findings were obtained to contribute to further scientific studies about the SQMMs and OSS-QMs. However, this section does not present all the findings or evidence obtained. Rather, in this section, results that contribute and guide the OSS-QMM development are presented. Therefore, further details about the results of our SLR studies can be reached from these references [30][37]. In Section 3.1, the findings of the SLR study as the outputs for Step-1 are explained, and in Section 3.2, the findings of the SLR study as the output for Step-2 are explained.

3.1. Software Quality Meta-models (Step-1)

The meta-models are expected to combine the isolated views to achieve a complete picture of software quality and, in turn, to create a common understanding between stakeholders for proper quality management throughout the entire life of a software product. In order to examine comprehensively the content and structure of the meta-models proposed for software quality and its evaluation (SQiE) in scientific literature, a Systematic Literature Review (SLR) study was carried out, and its results are reported in this study [37]. In this regard, the most-known seven academic search engines (namely Google Scholar, ScienceDirect, Scopus, ACM, Web of Science, IEEE Xplore, and Springer) were used to survey the literature by using the following search string;

```
("Meta model" OR "Meta-model") AND ("software quality") AND
      ("evaluation" OR "assessment" OR "measurement")
```

Then, primary studies were determined for the SLR of the meta-models for SQiE. In this context, a total of 28 studies out of 114 initially selected and 6488 initially retrieved were identified for further analysis with respect to inclusion and exclusion criteria (See the SLR study for details of the criteria [37]). These primary studies were analyzed with

respect to a number of research questions (RQs) (7 RQ and 19 sub-RQs). A list of RQs is given in Table 3.1. The data of each primary study was extracted considering the RQs. The data extraction sheet can be reached by the following reference [142]. This section will not present all the findings or evidence obtained from the SLR study. That is, the findings of the RQs that contributed to the development process of the OSS-QMM will be categorized and synthesized in the following sub-sections. It should be noted that, to the best of our knowledge, this is the first SLR study conducted on the meta-models for SQiE.

Table 3.1. The research question of the SLR (Step-1)

RQ#		Description
RQ.1		**What are the basic characteristics of the meta-model proposed in the study?**
	RQ.1.1	What is the main purpose of the meta-model proposed? (e.g. generic or specific)
	RQ.1.2	Which type of software products are targeted for SQiE? (e.g. OSS, COTS, custom)
	RQ.1.3	Is the meta-model taken as the base for tool development in the study? (yes or no)
RQ.2		**Are there any software quality models taken as a reference for the proposal? If yes:**
	RQ.2.1	Which software quality model(s) are taken as a reference? (e.g. ISO 25000)
	RQ.2.2	Does the meta-model serve for SQiE with respect to all the models taken as a reference?
	RQ.2.3	Is the terminology of the software quality model(s) taken as a reference mapped to the terminology defined by the meta-model in the study?
	RQ.2.4	Is the structure of the software quality model(s) taken as a reference mapped to the structure of the meta-model in the study?
RQ.3		**What are the basic characteristics of SQiE as defined in the meta-model?**
	RQ.3.1	What methods/techniques are used as a reference for SQiE? (e.g. GQM)
	RQ.3.2	Does the meta-model support subjective or objective evaluation?
	RQ.3.3	Does the meta-model support qualitative or quantitative evaluation?
	RQ.3.4	Which data analytics methods are defined for SQiE in the meta-model? (e.g. statistical, machine learning, expert evaluation, fuzzy)
	RQ.3.5	How are the results of the evaluation provided to users? (e.g. single index, table, graphic)
	RQ.3.6	Does the meta-model support SQiE in a specific phase of software development? If yes, which phase is it? (e.g. requirements, coding, field-use)
	RQ.3.7	Does the meta-model support SQiE at a single point or throughout software evolution?
RQ.4		**How is the meta-model structured?**
	RQ.4.1	Is there a specific structure of the meta-model? If yes, what is it? (e.g. hierarchical)
	RQ.4.2	What are the entities defined in the meta-model?
	RQ.4.3	Is the meta-model structured to define/include new quality models in evaluation?
RQ.5		**What are the means of data acquisition as defined in the meta-model?** (e.g. manual entry, batch import, automatic transfer from other repositories)
RQ.6		**Has the meta-model been validated? If yes, what was the method of validation?** (e.g. case study, literature mapping, peer review)
RQ.7		**How was the meta-model developed?**
	RQ.7.1	Was there a research method employed for development? If yes, what was it?
	RQ.7.2	What were the challenges faced in developing the meta-model?

3.1.1. Basic Characteristics of Meta-models

In this section, RQs related to the basic characteristics of the meta-models in the SLR study are synthesized and explained. These basic characteristics allowed us to gather

evidence about the current situation of meta-models before developing our meta-model. In this context, according to the analysis of the results of the SLR study, 75% of the meta-models were proposed for general purposes, as shown in Fig. 3.1 (a). These introduce the fundamental concepts present in every single approach to fixed-quality models. They are abstract enough to be used in several software engineering activities: specification, design, development, certification, selection, etc. [143]. Fig. 3.1 (a) also shows that 25% of the meta-models were proposed for a specific purpose. These are either developed for a specific software type (e.g. web services) or proposed for a specific phase in the software development process. It has been seen that many quality meta-models have been proposed for general purposes but not the same for specific software, including OSS.

In addition, as shown in Fig. 3.1 (b), more than half of the meta-models were proposed to cover all types of software. Only 2 of them were proposed for open source software (OSS), 2 of them for commercial software (COTS), and 1 of them for microservices (MS) software. In addition, 5 meta-models were proposed for web services (WS). These results indicated a lack of meta-models for OSS. Therefore, within the scope of this thesis, it was decided to develop a comprehensive meta-model for OSS quality.

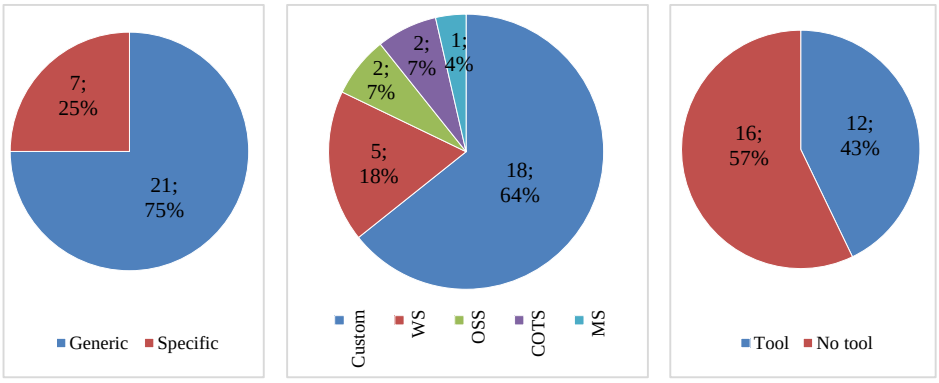


Figure 3.1. Basic characteristics of meta-models (RQ1): **(a)** Percent distribution of main purpose, **(b)** Percent distribution of types of software products targeted, and **(c)** Percent distribution of whether meta-models are taken as the base for tool development.

Another basic characteristic of meta-models is whether they have tool support. Time is a crucial factor in reducing software evaluation costs. Tools that allow automatic evaluation and eliminate manual effort have critical importance during software quality evaluation. As shown in Fig. 3.1 (c), less than half of the studies developed tools to reduce the effort spent on software quality evaluation. Since the main purpose of this thesis is to develop

a comprehensive quality meta-model for OSS that requires heavy effort, developing a tool that automates the use of this meta-model is planned as a future study.

3.1.2. Software Quality Models Referenced in Developing Meta-models

In this section, the software quality models referenced in developing software quality meta-models within the scope of the SLR study [37] are analyzed. These referenced software quality models guide the shaping of the structure and content of our meta-model. Existing software quality models were taken as reference in creating most of the meta-models. As shown in Fig. 3.2 (a), more than half of the proposed meta-models took ISO 9126 as a reference. Similarly, 6 of them took ISO 25010, 8 of them took McCall's model, 9 of them took Boehm's model, 6 of them took Dromey's model, 3 of them took IEEE 10610 as a reference, and 6 of them took other existing meta-models as reference. In 3 studies, the quality models referenced are not explicitly specified. It should also be noted that some meta-models took one or more quality models as references. Earlier meta-models were based on McCall and Boehm models, while later meta-models were based on ISO 9126.

As shown in Fig. 3.2 (b), 11 of the meta-models serve for SQiE with respect to all models taken as a reference, while 6 of them do not. The rest of them does not explicitly specify if they serve for SQiE with respect to all quality models taken as reference. Also, as shown in Fig. 3.2 (c), while 11 studies map the terminology of software quality models taken as a reference to the terminology of the meta-models they propose, 5 studies do not perform this mapping. Also, 7 studies do not explicitly specify whether they mapped the terminology of the quality models they referenced, and 5 studies make this mapping only partially. Moreover, as shown in Fig. 3.2 (d), 9 studies map the structure of the software quality model taken as a reference to the structure of the meta-models they propose, and 6 studies do not perform this mapping. In addition, 10 studies do not explicitly specify whether they map the structure of the quality models they referenced, and 3 studies make this mapping only partially.

As described above, meta-models have been proposed based on one or more quality models during development. In this respect, we developed the OSS-QMM based on well-designed and widely referenced quality models in the literature, as details are explained in Section 4.2. As can be seen from the results of the SLR study [37], the meta-models have deficiencies in carrying the characteristics of the quality models they reference. This

is especially valid for meta-models proposed for OSS quality. That is, as shown in Fig. 3.2 (b), no meta-model serves for software quality with respect to all the models taken as reference. The main reason for this is that meta-models often do not provide comprehensive terminology and structure matching during development. In this context, terminology and structure matching were performed with the quality models referenced during the development of the OSS-QMM. Therefore, the OSS-QMM serves for OSS quality with respect to all the OSS quality models taken as reference.

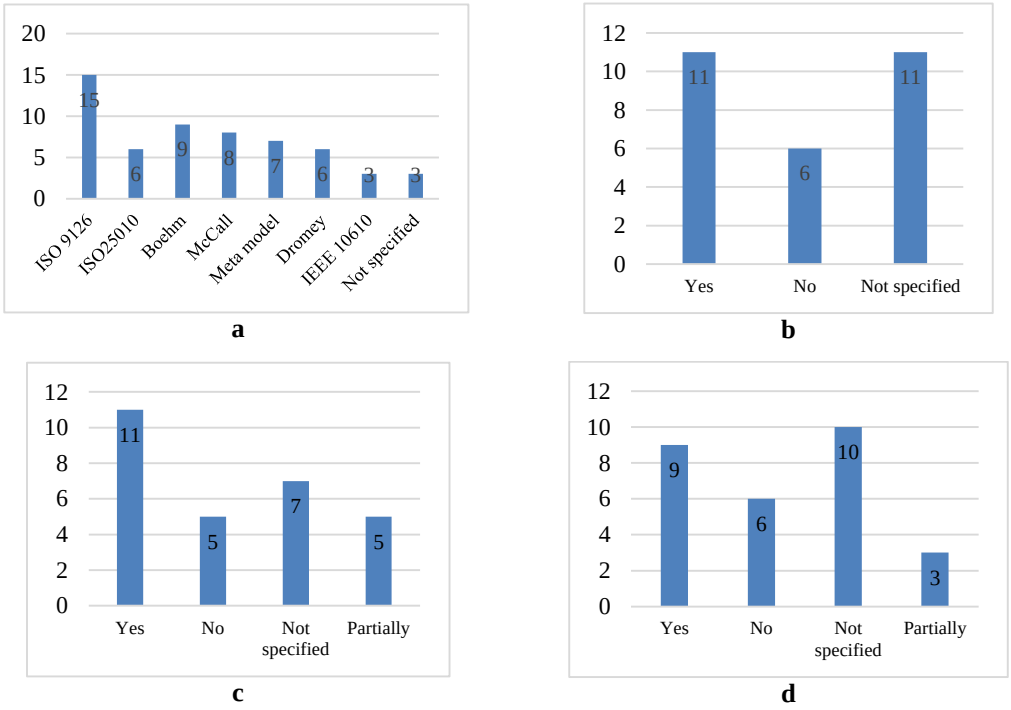


Figure 3.2. **(a)** Software quality model(s) taken as reference for meta-models, and the number of studies for **(b)** RQ2.2, **(c)** RQ2.3, and **(d)** RQ2.4.

3.1.3. Basic Characteristics of SQiE as Defined in Meta-models

In this section, RQs related to the basic characteristics of software quality and its evaluation (SQiE) as defined in software quality meta-models within the scope of the SLR study [37] are synthesized and explained. These basic characteristics of SQiE allowed us to shape the content of our meta-model. As shown in Fig. 3.3 (a), 14 meta-models evaluate quality objectively using only metric data. Only 1 study does not explicitly specify whether it uses metric data or user opinion in evaluation. The rest (46%) of studies make both objective and subjective evaluations considering both metric data and user opinions.

As shown in Fig. 3.3 (b), only 1 study gives qualitative results after evaluation, and 2 studies do not explicitly specify whether they provide quantitative or qualitative results. Also, 7 studies give both quantitative and qualitative results after evaluation. The rest of the studies (63%) provide quantitative results only. As shown in Fig. 3.3 (c), 9 studies provide evaluation results as an index in the range [0,1], and 2 studies provide evaluation results as an index in the range [0, max]. Also, 3 studies provide evaluation results in the Likert scale, and 4 studies provide results in graphical representation. However, 4 studies do not explicitly specify how the evaluation results are provided. Overall, many studies (64.2%) provide numerical values. It should be noted that one study might have one or more of the result types mentioned above.

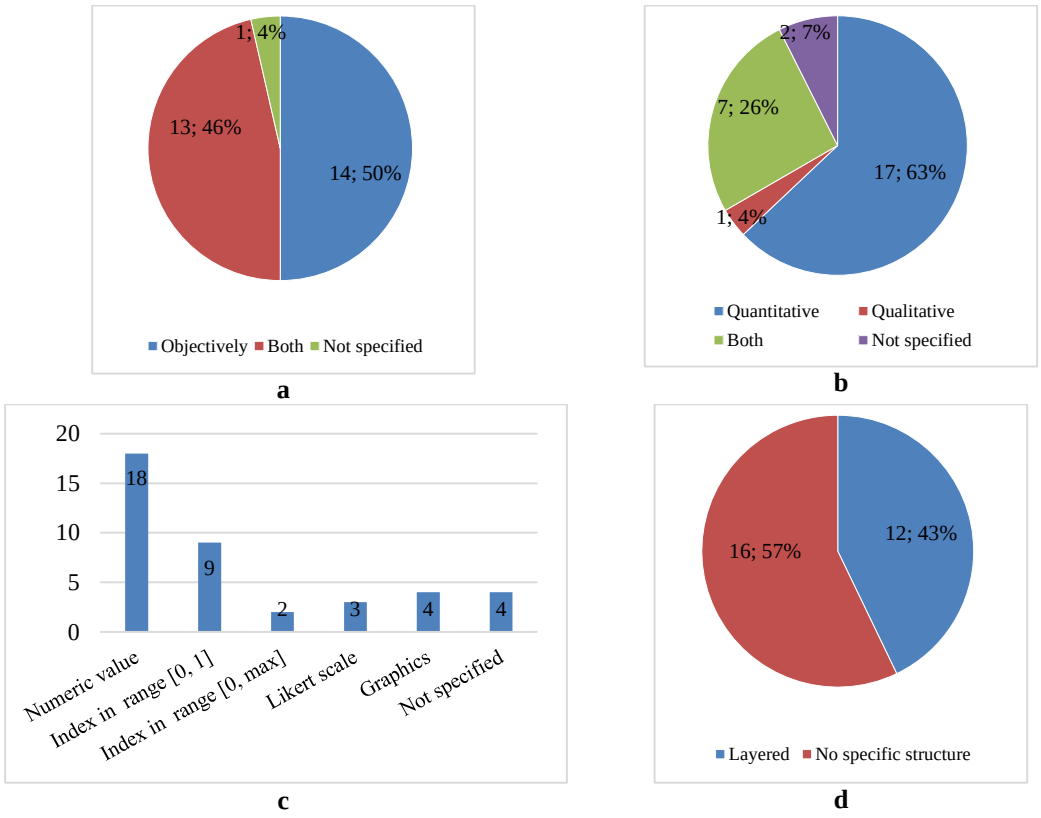


Figure 3.3. Basic characteristics of SQiE in meta-models (RQ3): **(a)** Percent distribution of subjective/objective evaluation, **(b)** Percent distribution of quantitative/qualitative evaluation, **(c)** Numeric distribution of evaluation result types, and **(d)** Percent distribution of structure of SQMMs

As can be seen, half of the meta-models evaluate software quality objectively, and almost the other half evaluate it both objectively and subjectively. As can be understood from these results, although objectivity is important in software quality evaluation, it is also important in subjectivity. This is because the meaning of the term "quality" is different

for stakeholders (i.e., customer, manager, tester, user, developer, etc.) since their expectations from software products or services are different. Thus, the OSS-QMM developed within the scope of this thesis allows the OSS to be evaluated both subjectively and objectively. That is, the OSS-QMM allows the evaluator to determine the importance of the quality characteristics to be evaluated and the OSS aspect (i.e., code-based and community-based aspects). For example, assume that the evaluator is a developer. According to this evaluator, the quality of the OSS product on the code side may be more important than the quality on the community side. Therefore, the OSS-QMM allows subjectivity in evaluation. However, the rest of the evaluation in the OSS-QMM allows a completely objective evaluation using metric data.

In addition, almost all of the meta-models proposed in the literature perform evaluation quantitatively, as shown in Fig. 3.3 (b). This finding shows that meta-models are generally aimed to produce quantitative values in order to see concrete results after evaluating software quality. In this context, the OSS-QMM enables the quantitative evaluation of OSS products. Then, as shown in Fig. 3.3 (c), the SLR study [37] analyzed how the evaluation results are provided to the user after these quantitative evaluations are performed. The purpose of the OSS-QMM is to derive OSS quality models that perform standardized evaluation by producing comparable results. As shown in Fig. 3.3 (c), although meta-models generally provide numeric values after evaluation, the fact that these values are in different scales and units can negatively affect standardization. In this context, the OSS-QMM developed within the scope of this thesis provides the evaluator with results in a certain number range (e.g., between 0 and 1) to obtain comparable results. Thus, standardization of evaluation results using different OSS quality models derived from the OSS-QMM is ensured.

3.1.4. Structure of Meta-models

In this section, RQs related to the structure and concepts of software quality meta-models in the SLR study [37] are synthesized and explained. These analyses guide the shaping of the structure and content of our meta-model. Most quality models have special structures such as hierarchical. Like quality models, considering the structure of meta-models, they also can have special structures. For instance, the meta-model in the study [144] consists of two layers for specification and evaluation. In another study [143], the meta-model consists of three layers for the fundamental, metric, and context. As shown in Fig 3.3 (d),

in the SLR study [37], it was observed that 12 studies propose meta-models having a layered structure and that the remaining (57%) studies do not have any specific structure.

In addition, the concepts of software quality meta-models were analyzed and categorized in the SLR study [37], as shown in Table 3.2. Meta-models are diagrams that contain concepts and relationships between these concepts. Terms for SQiE were sometimes defined by different concepts in meta-models, even though they were defined for the same purpose. For example, a meta-model [145] uses the concept of "quality attribute", and another meta-model [146] uses the concept of "quality characteristic" for the same purpose. These indicated that there is inconsistency among concepts of software quality models proposed in the literature. Also, the table indicated that "quality attribute" and "measure" are the most commonly used concepts, while "requirement", "scale" and "instrument" are the least frequently addressed concepts in the meta-models.

Table 3.2. Concepts used in meta-models as entities with their frequencies

Category	**Names of entities in different meta-models**	**#Freq.**
Data analysis	**Syn:** Analysis model / decision criteria / interpretation rule / analysis	7
Entity	**Syn:** Entity / component / quality artifact / measurable entity	18
Evaluation (E)	**Syn:** Evaluation / assessment model **Agg:** Formula / rule / E. result / E. aspect / E. impact	14
Instrument	**Syn:** Tool / instrument	5
Measure	**Syn:** Measure / metric **Agg:** Base measure / base metric / derived measure / derived metric	23
Measurement (M)	**Agg:** M. approach / measurable concept / M. method / M. function / M. data / M. result / value / indicator	13
Property	**Syn:** Property / quality aspect / quality dimension / quality type / feature	12
Quality attribute	**Syn:** Quality characteristic / quality attribute / quality factor / characteristic / attribute / factor / product factor **Agg:** Sub-characteristic / sub-factor / base attribute / direct attribute / derived attribute / indirect attribute	27
Quality goal	**Syn:** Quality goal / goal / quality target / purpose / target / objective	7
Quality model	**Syn:** Quality model	9
Requirement	**Syn:** Quality requirement / requirement / specification	3
Scale	**Syn:** Scale / type of scale / measurement scale	4
Unit	**Syn:** Unit / measurement unit / unit of measurement	7
View	**Syn:** Viewpoint / view / stakeholder	6

* **Syn** denotes synonymous concepts for a category, while **Agg** denotes sub-categories or aggregated concepts under a category.

As can be seen, the layered structure is frequently used in the structures of the software quality meta-models. This layered structure helps in categorizing the concepts in meta-models' design and in increasing the understandability of the meta-models. Therefore, while designing the OSS-QMM, we categorized the concepts of the OSS-QMM as specification, measurement and evaluation. Then, we designed the structure of the OSS-

QMM to consist of these three layers. Details of each layer are explained in Section 4. Also, in the SLR study [37], the RQ asked to find out the frequency of concepts used in the meta-models indicated inconsistency between the concepts, as shown in Table 3.2. This RQ triggered us to analyze in-depth the inconsistencies between the concepts of the meta-models. Therefore, in our other study [54], we analyzed these concepts in detail by getting the root of their meanings and origins. Details of these analyzes are given in Section 4.1. As a result of these analyses, the concepts were free of inconsistencies. Therefore, the OSS-QMM within the scope of this thesis was proposed by considering the concepts free of inconsistencies.

3.1.5. Development of Meta-models

In this section, the challenges encountered in the development of the meta-models examined in the SLR study [37] are discussed. In Table 3.3, the challenges faced in proposing the meta-models and the number of studies (with percent distribution) facing these challenges are given. As seen in the table, the poor interpretation of interdependencies and measurements was one of the most common challenges. Inconsistency among different terminologies was another important challenge. Only four studies reported the challenge of different expectations of stakeholders. These challenges were classified based on the suggestions of [147].

Table 3.3. Classification of challenges and the number of studies that have faced these challenges

Description of challenge	#study
C1: Inconsistency in terminology: "Most approaches that are not based on theoretical grounds, lack a definition for quality concepts that is precise and concise" [147].	**9** (24%)
C2: Partially defined: "Most quality models are outlined but not fully developed. All define measurable concepts, some of them also attributes, few of them include (most often partial) measures, and scarcely any defines decision criteria or indicators" [147].	**7** (18%)
C3: Lack of focus: "Most quality models provide an extensive (and mostly tangled) coverage of stakeholders and levels of abstraction" [147].	**7** (18%)
C4: Lack of clarity in interdependencies and measure interpretations: "In most quality models that are not based on theory, the degree of influence of individual internal quality factors on the quality in the use of the application, as well as their interdependencies, are not well established" [147]. Also, measure interpretations of some models are not clear.	**11** (30%)
C5: Different expectations of stakeholders: Stakeholders in the software process has different expectations from meta-models	**4** (10%)

In the background, according to the findings of the SLR study, these challenges arose, especially from the quality models taken as reference. Therefore, to cope with these challenges (i.e., C2 and C4) during the development of the OSS-QMM, we built it based

on well-designed and widely referenced quality models in the literature (see Section 4.2 for details). Also, to cope with challenge C1, we analyzed the concepts of the OSS-QMM in detail by getting the root of their meanings and origins. As a result of these analyses, the concepts were free of inconsistencies. According to the findings of the SLR study, the majority of the meta-models have been proposed for a custom type of software (i.e., all types of software) in the literature. Therefore, they do not provide a comprehensive evaluation for the specific type of software since their main focus is not clear, and they do not analyze in detail the requirements specific to particular software. Developing a meta-model directly for a certain software type requires knowledge and detailed analysis of that type of software. Since the main focus was determined as OSS in this thesis, we carried out an SLR study [30] that examined OSS quality models in detail to cope with challenge C3. Also, taking into account that stakeholders have different expectations from OSS quality (i.e., challenge C5), we added the concept of viewpoint to the OSS-QMM (see Fig. 5.10). Thus, we aimed to deal with C5 and enabled evaluations to be performed according to certain viewpoints. In summary, we followed a systematic way to develop our meta-model, as shown in Fig. 1.2. This systematic process helped us to overcome all the challenges encountered.

In addition to the analyzes discussed above, we analyzed validation methods in an RQ of our SLR study. All methods employed to validate meta-models are described in both the SLR study and Section 2.4.4. Therefore, it will not be repeated here. The findings we obtained from this RQ guided the validation process of the OSS-QMM developed within the scope of the thesis.

3.2. Software Quality Models (Step-2)

Software quality models are frameworks used to evaluate the quality of software and identify areas for improvement. They provide guidelines, best practices, and assessment criteria for software quality and can help organizations to continuously improve the software development process. The primary motivation behind this thesis was to enable the proposal of standardized quality models that can perform comparable measurements for OSS. In this context, meta-models are important because they can be used to standardize the quality models. To propose a comprehensive OSS-QMM, the structure of the existing OSS quality models must be well understood since the quality models are the instances of the meta-models.

In this regard, we have conducted an SLR study [30] to characterize the existing QEMoF for OSS and to comprehensively examine their content and structure for identifying the gap between theory and practice. While conducting the SLR, primary studies of QEMoF for OSS were determined by using the seven most-known scientific digital libraries (namely Google Scholar, Springer, ACM, ScienceDirect, IEEE Xplore, Scopus, Web of Science) to survey the scientific literature. In order to cover all relevant studies in literature, a search in scientific digital libraries was performed by using various combinations of search strings, and the following search string was determined as a result:

```
("quality") AND ("evaluation model" OR "assessment model"
                OR "measurement model" OR
   "evaluation framework" OR "assessment framework" OR
                "measurement framework") AND
  ("OSS" OR "FOSS" OR "FLOSS" OR "open source software")
```

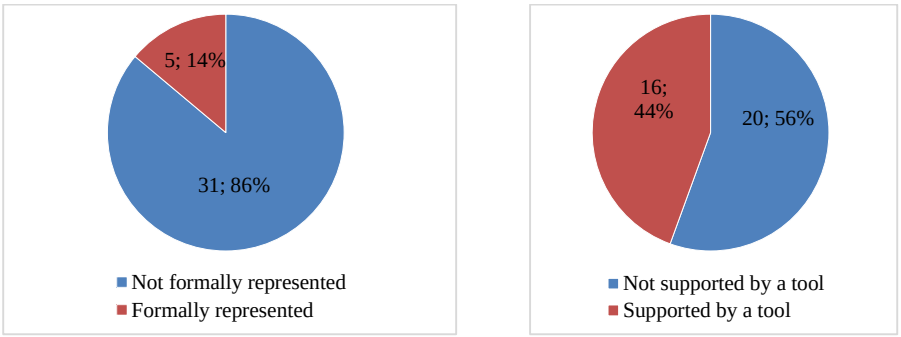
In this context, a total of 36 primary studies out of 142 initially selected and 13.146 initially retrieved were identified for further analysis with respect to inclusion and exclusion criteria. The final set of study pool is given in Appendix-1. The list of inclusion and exclusion criteria is given in the SLR study [30]. Then, in order to examine all aspects of these evaluation models or frameworks proposed for assessing OSS quality, the selected primary studies were reviewed and analyzed with respect to the research questions (RQs) that we raised. A list of RQs is given in Table 3.4. The data of each primary study was extracted considering the RQs, and the data extraction sheet can be reached by the following reference [152]. In this section, results from all RQs will not be presented. In other words, only the findings of the RQs that contributed to the development process of the OSS-QMM will be categorized and synthesized in the following sub-sections. Further results are given in the SLR study [30].

Table 3.4. The research question of SLR (Step-2)

RQ#		Research Question
RQ.1	What are the basic characteristics of the quality evaluation models or frameworks (QEMoF) proposed for OSS?	
	RQ.1.1	What is the main goal of the QEMoF proposed in the study? (e.g. generic goal or a specific goal)
	RQ.1.2	Is the QEMoF formally represented (e.g. by metamodeling) in the study? (yes or no)
	RQ.1.3	What is the contribution type of the study? (model, framework)
	RQ.1.4	Is the QEMoF supported by a tool in the study? (yes or no)
RQ.2	How is the QEMoF for OSS structured?	
	RQ.2.1	Which quality model is the QEMoF based on? (e.g. ISO 25000, Dromey)
	RQ.2.2	Is there a specific structure of the QEMoF? If yes, what is it? (e.g. hierarchical, layered)
	RQ.2.3	From what aspects can OSS be evaluated by the QEMoF? (e.g. product quality, quality in use, community-related)
RQ.3	What is the degree of guidance provided for the evaluation of OSS by the QEMoF?	
	RQ.3.1	Is the evaluation procedure of the QEMoF adequately described in the study? (yes, no, partially)
	RQ.3.2	Is a demonstration of the evaluation using the QEMoF provided in the study? (yes, no, partially)
RQ.4	What are the basic characteristics of QEMoF for evaluating OSS?	
	RQ.4.1	What methods/techniques are used as a reference for evaluation in the QEMoF? (e.g. GQM)
	RQ.4.2	Is the license type of the OSS product used as an evaluation criterion in the QEMoF? (yes, no)
	RQ.4.3	Does the QEMoF support subjective or objective evaluation? (e.g. user opinion, metric data)
	RQ.4.4	Does the QEMoF support qualitative or quantitative evaluation?
	RQ.4.5	What quality attributes are used for evaluation by the QEMoF? (e.g. maintainability, efficiency)
	RQ.4.6	What software metrics are used for evaluation by QEMoF?
	RQ.4.7	What are the aggregation methods used for evaluation by the QEMoF? (e.g. weighted arithmetic mean, overall sum)
	RQ.4.8	Does the QEMoF support evaluation at a single point in time or throughout the evolution of OSS?
	RQ.4.9	How is data collected for the evaluation of OSS in the proposed QEMoF? (e.g. automatically, manually)
	RQ.4.10	What is the required skill level of users who evaluate OSS using the QEMoF? (e.g. low, medium, high)
	RQ.4.11	How are the results of the evaluation provided to the user in the QEMoF? (e.g., ordinal scale, nominal scale)
RQ.5	How was the QEMoF for OSS developed?	
	RQ.5.1	Was there a research method employed for developing the QEMoF for OSS? If yes, what was it? (e.g. solution proposal, weak empirical study, strong empirical study)
	RQ.5.2	What were the challenges faced while developing the QEMoF for OSS?
RQ.6	Has the QEMoF for OSS been validated?	
	RQ.6.1	What was the method of validation? (e.g., case study, toy experiment, peer review, expert opinion, industrial validation)
	RQ.6.2	What type of OSS application was subject to validation?
RQ.7	What is the evidence for the usage of QEMoF for OSS?	
RQ.8	What is the overall quality of the QEMoF for OSS?	

3.2.1 The Basic Characteristics of the OSS Quality Models

In this section, RQs related to the basic characteristics of OSS software quality models or frameworks (QEMoF) analyzed in the SLR study [30] are synthesized and explained. These analyses helped us to find out whether there is a need for a meta-model in practice. In this context, it was investigated whether the QEMoFs are represented formally in the studies. In other words, it was investigated whether a representation containing an abstract relationship was presented between the concepts (rules and constructs) used in the QEMoF and whether the QEMoF was created based on this representation. Here, aside from representing the QEMoF formally, representing the relationships between a few concepts used in the QEMoF was also considered to classify the study as "formally represented". As shown in Fig. 3.4 (a), the vast majority (86%) of the models are not formally represented, and only 14% are represented as either a meta-model (3 studies) or a conceptual model (2 studies). Also, within the scope of the SLR study, it was investigated whether the QEMoFs are supported by a tool as a basic characteristic. As shown in Fig. 3.4 (b), 20 (56%) of the QEMoF are not supported by a tool, while the remaining 16 (44%) have such support.



a **b**

Figure 3.4. Percent distribution of basic characteristics of QEMoF for OSS: **(a)** whether represented formally (RQ1.2), and **(b)** whether supported by a tool (RQ1.4)

As seen above, the number of OSS quality models that are formally represented is very small. That is, the majority of OSS quality models are not derived from a particular meta-model. This indicates that the number of individual models proposed without adhering to a meta-model is increasing. This leads to heterogeneity in the structure and content of the proposed quality models, as stakeholders have different expectations of quality. In addition, the lack of sufficient tool support for these heterogeneous OSS quality models has a negative impact on their practical use. Because without tool support, it is difficult

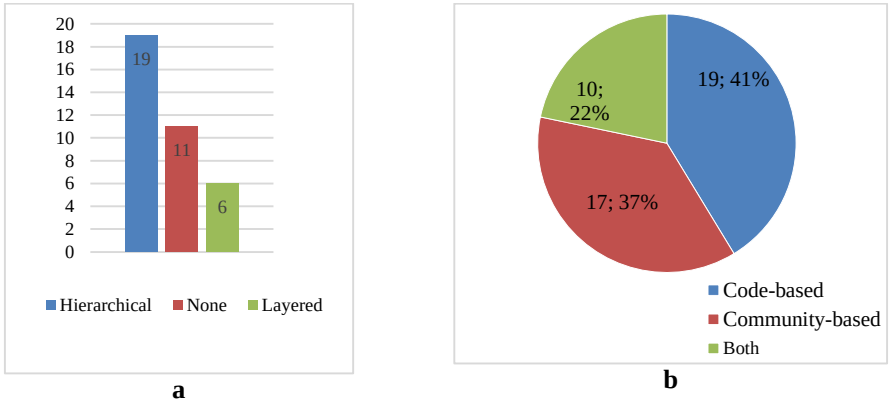
to understand and apply these individual models in practice. For the reasons described above, this thesis aims to develop a comprehensive meta-model for OSS quality. It is aimed that the quality models produced from this meta-model have a homogeneous structure and produce comparable results. Thus, it is aimed to transform OSS quality models from individual models to homogeneous models that allow evaluation according to the needs of the stakeholders.

3.2.2. The Structure of the OSS Quality Models

In this section, RQs related to the structure and content of OSS quality models in the SLR study [30] are synthesized and explained. In this context, the software quality models referenced in developing OSS quality models, and the structure and evaluation aspects of OSS quality models are synthesized and explained. These analyses guide the shaping of the structure of our meta-model since quality models are instances of meta-models. First of all, the software quality models referenced in developing OSS quality models are investigated. Software structure has changed in time with respect to manufactured or generated components. Therefore, quality models have also had to be constantly updated and evaluating software quality has faced new challenges. Models developed until the year 2000 are categorized as basic models (e.g. ISO 9126, McCall, or Boehm). Also, the tailored models developed after that year generally originated from the basic models and other tailored models [2][84]. OSS-specific models are grouped into tailored models [2]. Details of basic and tailored quality models are explained in Section 4.2. According to the results of the SLR study, both basic and tailored quality models are referenced in developing OSS quality models. Among these models, the most referenced quality models are explained in Section 4.2.2 (Table 4.3), together with the reasons for referencing them. Therefore, it is not repeated here.

In addition, the general structure of the QEMoF is investigated in the SLR study. This analysis is intended to help researchers understand the structure of the models proposed in the literature and to see the commonalities, if any. As shown in Fig. 3.5 (a), 19 (53%) of the models are structured as hierarchical. That is, they are generally structured from the characteristic features that cannot be measured directly at the top of the hierarchy and detailed into the characteristic features that can be measured directly at the bottom. Factor, sub-factor, and metric, in the order from top to bottom in the hierarchy, is such an example. Also, in the figure, 6 (17%) of the models are structured as layered. For

example, a model has three layers, namely basic, intermediate, and advance levels, in the study [27]. Also, 11 (30%) of the models do not have a specific structure.



a **b**

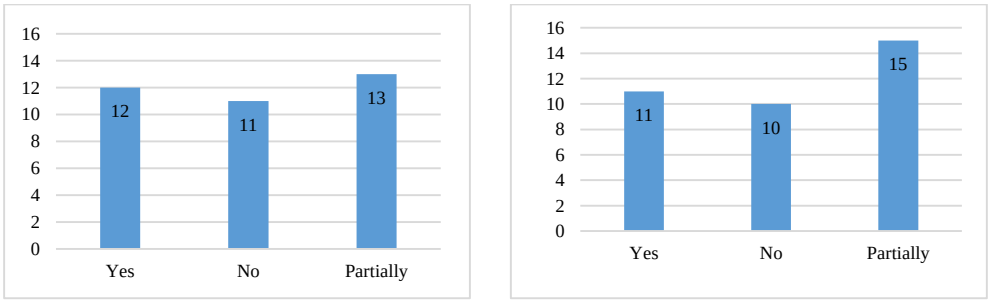
Figure 3.5. Number distribution of: **(a)** design structures of QEMoF, and **(b)** evaluation aspect of QEMoF

In addition, the evaluation aspects of the QEMoF are investigated in the SLR study. Unlike commercial software, numerous data is stored in OSS databases. This provides not only source code but also data from the community aspect, so a variety of data can be used for quality evaluation. Most significant aspects possessed by the existing QEMoF are investigated to provide guidance for future studies. Many basic and well-structured quality models exist, but they are mostly adopted for commercial products and overlook some unique properties of OSS, such as its community. As shown in Fig. 3.5 (b), 17 (37%) quality models consider community-based attributes, 19 (41%) of them consider code-based attributes, and 10 (22%) of them consider attributes from both aspects.

As described above, OSS quality models are usually developed with reference to well-designed basic (e.g., ISO 9126, McCall, or Boehm) and tailored models (e.g., OSMM, OpenBRR). Therefore, our meta-model developed in this thesis is based on these well-designed quality models since quality models are instances of meta-models. The referenced quality models are described in Section 4.2.2. In addition, the majority of the OSS quality models have a hierarchical structure and a few of them have a layered structure. Regarding this, we developed our meta-model by considering this hierarchical and layered structure. The details of the hierarchical and layered structure considered in our meta-model are described in Section 4. In addition, the main feature that distinguishes the OSS from commercial software is that OSS contains attributes belonging to the community-based aspect. Therefore, we developed our meta-model by considering this specific feature of OSS. That is, our meta-model allows for community-based evaluation, code-based evaluation, or evaluation in both aspects.

3.2.3. The Degree of Guidance Provided by the QEMoF

In this section, the degree of guidance provided for the evaluation of OSS by the QEMoF is synthesized and explained. These analyses helped us to find out the understandability and applicability of QEMoFs by external parties. In Fig. 3.6 (a), it is investigated whether the evaluation procedure of the QEMoF is adequately described in the studies. As shown in the figure, 12 of the studies describe their evaluation procedures adequately, while 13 of them describe their procedures only partially, and 11 of them do not describe a procedure at all. In Fig. 3.6 (b), it is investigated whether a demonstration of the evaluation using the QEMoF is provided in the studies. As shown in the figure, by using the proposed QEMoFs, 11 of the studies provide a demonstration of the quality evaluation adequately, 15 of them provide a demonstration of the evaluation inadequately, and 10 of them do not provide a demonstration at all. "Yes" means that the evaluation procedure/demonstration is adequately described, "No" means that the evaluation procedure/demonstration is not described, and "Partially" means that the evaluation procedure/demonstration is described to some degree but not sufficient for an application.



a **b**

Figure 3.6. Number of studies for: **(a)** specification of the quality evaluation procedure, **(b)** demonstration of quality evaluation procedure by application

As described above, the results indicate that there is a need to take care of the evaluation procedure and its demonstration in the studies in order to properly motivate and guide the potential users in evaluation. Because this negatively affects the practical use of the OSS quality models. In this context, in our recent study [54], each concept in our meta-model is given with its definition and intended use in the meta-model for better understanding by users. Also, a new operationalized OSS quality model and existing OSS quality models are derived from the OSS-QMM, and the derivation of each model is explained in detail [54]. Moreover, the practical application of these models is demonstrated step by step in harmony with our meta-model [54].

3.2.4. The Basic Characteristics of QEMoF for Evaluating OSS

In this section, RQs related to the basic characteristics of QEMoF for evaluating OSS are synthesized and explained. These analyses guide the shaping of the content of our meta-model since quality models are expected to be instances of a meta-model. As shown in Fig. 3.7 (a), 20 of the studies (56%) measure OSS quality subjectively by considering the viewpoint of the evaluator, while 16 (44%) of them use both subjective and objective measurements in evaluation. As it can be understood from these results, both objectivity and subjectivity are important in the evaluation of OSS. Therefore, the OSS quality models derived from our meta-model are intended to allow for both subjective and objective evaluation.

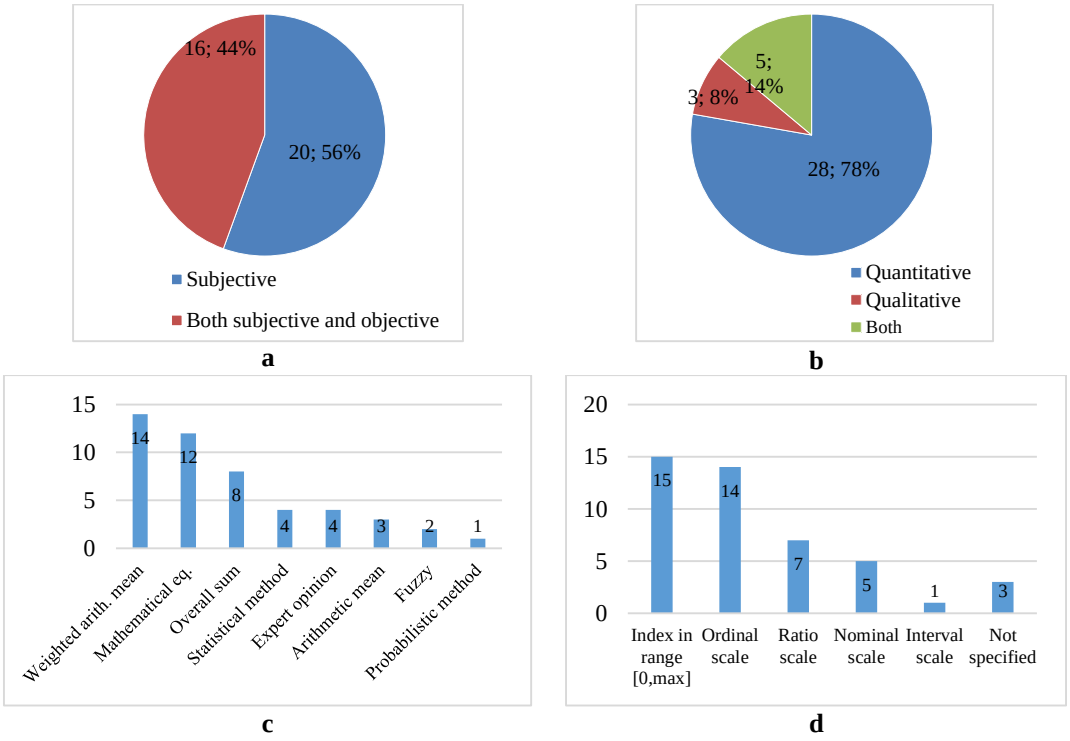


Figure 3.7. Distribution of: **(a)** subjective or objective evaluation supported (RQ 4.3), **(b)** quantitative or qualitative evaluation supported (RQ 4.4), **(c)** aggregation techniques used in QEMoF (RQ 4.7), and **(d)** how evaluation results are provided to users by QEMoF (RQ 4.11)

As shown in Fig. 3.7 (b), the majority (28 with 78%) of the studies support quantitative measurement with numeric values, while only 3 of them (8%) support qualitative measurement without using numeric values, and 5 of them (14%) support both types of measurement. This result indicated that OSS quality models generally perform a

quantitative evaluation to see concrete results. In this context, the OSS quality models derived from our meta-model are aimed to produce quantitative evaluation results.

As shown in Fig. 3.7 (c), the most commonly used technique is weighted arithmetic mean (in 14 studies), in which some measurement results contribute to the overall result more than others. The second most commonly used technique to aggregate the results is some mathematical equation (in 12 studies) developed by authors. This is followed by the overall sum aggregation technique (in 8 studies) in which the measurement results contribute equally to the overall result. Also, 4 studies employ statistical techniques, 4 use expert opinion, 3 use arithmetic mean, 2 use fuzzy logic, and 1 study employs probabilistic methods. For the studies (e.g., [15][32][149]) that do not clearly specify which aggregation technique they use, it was elicited from the applications of the QEMoF in these studies. It should be stated that a study may use more than one aggregation technique in its proposal. These results helped us determine the techniques to be used for some concepts of our meta-model (e.g., weighting method, Measurement function, etc.). The most commonly used techniques in Fig. 3.7 (c) were identified and used to implement our meta-model in practice. A list of these techniques is given in Section 6.

As shown in Fig. 3.7 (d), 15 of the studies provide their results as an index in the range [0, max], which enables the comparison between alternative OSS products. Also, 14 of the studies provide evaluation results on the ordinal scale, e.g., the study [25] presents its results as good, fair and poor. In addition, 7 studies provide evaluation results in ratio scales and 5 in the nominal scale, which uses the labeling of distinct classifications. Only 3 of the studies do not clearly state how they provide their results. It should be noted that a study may provide its results to users in more than one type of scale. The results indicated that OSS quality models generally provide results in different scales and units. These can affect standardization negatively. Therefore, the OSS quality models derived from the OSS-QMM are intended to provide an index in the range (e.g., between 0 and 1) that is mostly used by quality models as a result of the evaluation. Thus, standardization of evaluation results using different OSS quality models derived from the OSS-QMM is ensured.

3.2.5. The Challenges in Developing the OSS Quality Models

In this section, challenges faced, as stated in the studies while developing the QEMoF for OSS, are analyzed. These analyses are important to understand the importance of our

meta-model which allows us to derive comprehensive OSS quality models with a homogeneous structure. There are many challenges faced while developing OSS quality models, and the most common challenges reported in the studies are classified as shown in Table 3.5. The challenge faced with the highest frequency by the studies is C1 (in 17 studies) with a diverse and dynamic structure of OSS. Considering this structure, it is so challenging to design situation-based procedures to determine evaluation criteria in the studies. The challenge faced with the second highest frequency by the studies is C2 (in 14 studies), with difficulty in determining required metrics and data sources for evaluation. The OSS repositories provide many data sources in both code-based and community-based aspects, and many metrics are available for evaluation in these data sources. Therefore, it is also challenging to map various metrics existing in data sources to the properties of OSS to be evaluated.

Table 3.5. List and frequency of challenges faced in developing QEMoF

Description of challenge	**Frequency**
C1: The diverse and dynamic structure of OSS	17
C2: It is difficult to determine the required metrics and the data sources for evaluation	14
C3: Existing quality models are insufficient for evaluating OSS	12
C4: It is difficult to aggregate heterogeneous results from different data sources	9
C5: Different expectations of stakeholders from OSS products	8
C6: It is difficult to evaluate OSS from all aspects	5

The challenge with the third highest frequency is C3 (in 12 studies), indicating the insufficiency of the existing models for evaluating OSS. The QEMoF tailored for OSS generally originated from well-designed models such as ISO/IEC 9126. However, the studies face challenges in developing well-designed QEMoF for OSS since the referenced quality models are mostly adopted for commercial products, and they overlook some unique properties of OSS, such as its community. The challenge with the next highest frequency is C4 (in 9 studies) with the difficulty to aggregate heterogeneous results from different sources. Considering that OSS has various metrics for evaluation and, accordingly, heterogeneous measurement results, aggregating these results in QEMoF is also challenging. The next frequent challenge is C5 (in 8 studies) indicating different expectations of stakeholders from OSS products. Since the expectations of the users from the OSS quality are different, researchers face challenges in keeping up with all these expectations while developing QEMoF for OSS evaluation. Finally, the lowest frequency challenge is C6 (in 5 studies) with difficulty to evaluate OSS products from all aspects.

Since OSS is open to evaluation in many aspects, it is challenging to consider every single aspect in developing QEMoF. It should be stated that a study may report more than one challenge.

As seen above, developing a comprehensive quality model is a complex process due to the challenges mentioned in the table. In addition, the different expectations of users from quality have caused quality models to move away from standardization and turn into individual models. Therefore, the meta-model we have developed to overcome the challenges mentioned above helps us to derive comprehensive quality models that evaluate all aspects of software, deal with heterogeneous data, have a homogeneous structure and are shaped according to needs.

3.2.6. The Evidence for the Usage of OSS Quality Models

In this section, evidence for the practical use of the QEMoF is investigated. This evidence is crucial for understanding the lack of a meta-model in practice. To access this evidence, first of all, all studies citing QEMoF examined within the scope of this SLR were searched. Then, an additional search was conducted in the gray literature for the use of the QEMoF examined in this study. Only English sources and studies were taken into consideration while conducting this search. Thus, beyond the initial studies in which the QEMoF was proposed, expansion in their evaluation and evidence of practical use were investigated. Considering that technology is constantly evolving, it is important to conduct these searches because the opportunities in the years when the earlier QEMoF was proposed are not equal to the opportunities of today. As shown in Table 3.6, the results were obtained by examining the scientific literature for the studies citing the QEMoF and also the gray literature on the use of these QEMoF. It is noted under Table 3.6 that the sources belonging to the authors who proposed the QEMoF are marked with (*), while the sources belonging to the authors other than the ones who proposed the QEMoF are marked with (**). In this regard, information was obtained about only eight QEMoF listed in the first column of the table.

Table 3.6 List of sources that: advance the QEMoF throughout the lifetime of their projects, expand the evaluations of the QEMoF, and provide evidence of practical use of the QEMoF

Study	Sources that advance the QEMoF throughout the lifetime of their projects	Sources that expand the evaluations of the QEMoF	Sources that provide evidence of practical use of the QEMoF
OSMM	-	-	**Tawsopar et al. [150], **Akbari et al. [151]
QSOS	-	**Zarouk et al. [152]	**Laaziri et al. [153]
OpenBRR	-	*Wasserman et al. [154]	**Syahlie et al. [155], **Marinheiro et al. [156]
SQO-OSS	*Gousios et al. [157], *Gousios et al. [158], *Gousios et al [159]	*Spinellis et al. [160]	-
Qualipso	*Taibi et al. [161], **Xu et al. [162], **Petrinja et al. [163]	-	**Cotugno et al. [106], **Malanga et al. [164]
QualOSS	*Soto et al. [165], **Majchrowski et al. [166], *Deprez et al. [167]	**Cortazar et al. [168], **Cortazar et al. [169]	-
EFFORT	*Aversano et al. [170], *Aversano et al. [171], *Aversano et al. [172]	-	*Aversano et al. [173], *Aversano et al. [174]
OSSPal	-	-	**Leite et al. [175], **Marques et al. [176], **Calçada et al. [177], **Cruz et al. [178], **Paula et al. [179]

As seen in the second column of the table, some of the QEMoF, such as SQO-OSS, QualOSS, Qualipso, and EFFORT, were developed within the scope of a project. Since the projects usually progress step by step, more than one study has been found in the lifetime of the project related to the introduction and development process of a QEMoF. In addition, as mentioned above, considering the continuous development of technology, some sources have expanded the evaluation of the QEMoF over the years, as shown in the third column of the table.

As shown in the fourth column of Table 3.6, most of the QEMoF has been used by other authors to evaluate some OSS products in case studies conducted within their own studies. Although this situation is considered as evidence for the use of the QEMoF, it cannot be considered as strong evidence because all the authors of these articles had academic backgrounds and the products were evaluated in laboratory studies instead of real-world cases. In the literature, one study [106] was found that can be considered as evidence for the practical use of a QEMoF (for Qualipso) in the industry, and this study is marked with

(**) in the fourth column of the table. The authors of this study had academia-industry collaboration. In this study, they have presented an overview of the usage of Qualipso for the evaluation of the quality of OSS, and also SCRUM as a development methodology for addressing the development needs of the Italian Army.

As a result of this analysis, no strong evidence except this study [106] has been found in the gray literature regarding the use of the QEMoF by companies in order to evaluate the quality of OSS software used within their own development bodies. The main reason for the little adoption of proposed OSS quality models in practice is that the models have moved away from standardization and become individual models. In other words, the models are heterogeneous and produce incomparable results. The meta-model developed in this thesis is important in order to eliminate these undesirable situations and to facilitate the practical use of the models.

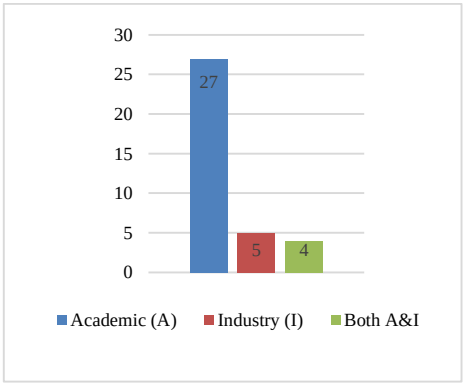


Figure 3.8 Distribution of studies with respect to author affiliation type

Apart from them, as shown in Fig. 3.8, the affiliation types of the developer of the 36 OSS quality models were examined. The results indicated that 27 studies (with 75%) were carried out with academia, 5 studies (with 14%) with industry practitioners, and the remaining 4 (11%) with industry-academia collaboration. As a general observation, the concentration of the proposed QEMoF in the academic domain indicates that the number of models/frameworks accepted and/or adopted in the industry is not at the desired level. This situation negatively affects the adoption of the OSS quality models in practice. In order to overcome this situation, the OSS-QMM development and validation process has been carried out in collaboration with the industry. In other words, the opinions of the subject matter experts have been considered during the development process of our meta-model, and it has been validated by referring to the expert opinions in the validation phase. This means that our meta-model has been developed taking into account practical needs.

4. MATCHING TERMS OF QUALITY MODELS AND META-MODELS (STEP-3)

In this thesis, the systematic research process has been followed to propose a meta-model for OSS quality. In this context, the ***step-based process for meta-model creation,*** which is described in Fig. 1.2, has been followed. The first two steps [30][37] and their outputs are described in Section 3. In this section, the analyses conducted in Step 3 and their outputs are presented. The main purpose of this section is to match the concepts of meta-models to the terms of quality models since quality models are defined as an instance of the meta-model according to MOF architecture [180]. For this matching process, a preliminary preparation has been performed with the two SLR studies [30][37] described in Section 3. In this context, meta-models have been first analyzed, and then quality models have been analyzed to create a suitable infrastructure for matching. These analyses have helped to highlight the shortcomings of quality meta-models and models. That is, these two SLR studies [30][37] have triggered further analyses of quality meta-models and models to address these shortcomings. Therefore, we have conducted another study [54] within the scope of the thesis to perform the following;

- The meanings of the terms used in SQMMs have been revealed by searching the referenced standards (i.e., the origin of the terms).
- A comparative analysis of concepts and terms used in SQMMs has been performed.
- The common structure of the quality models has been analyzed and revealed.
- The terminology in the quality models of OSS has been mapped with the terminology of the SQMMs.
- The meta-model has been developed taking into account all previous processes (subject of Section 5).
- The validation of the developed meta-model has been performed (subject of Section 6).

The contributions of the above-mentioned efforts to the literature can be listed as follows;

- Comparative analysis of concepts and terms used in SQMMs has contributed to harmonizing the terms and concepts of different SQMMs and also has formed the basis for proposing an OSS-QMM.
- The investigation of how the terms used in the SQMMs are expressed in the referenced standards has also contributed to eliminating inconsistencies in the international standards proposed so far.
- The examination of the common structure of quality models has contributed to a better understanding of the structure of the quality models in the literature.
- The terminology in the quality models of OSS has been mapped with the terminology of the SQMMs, which has contributed to the resolution of the terminology conflicts between the quality models of OSS and the SQMMs.
- The OSS-QMM will allow the development of OSS quality models that are flexible enough to apply in various business domains and that fulfill the needs of stakeholders, allow to obtain comparable results, cover various aspects of quality etc. (subject of Section 5).
- Since the OSS-QMM has been proposed considering the common structure of important OSS quality models, similar quality models to be proposed in the future using this OSS-QMM will have the chance of adopting a standard structure (subject of Section 5).

The rest of this section is organized as follows: In Section 4.1, the inconsistencies, commonalities, and terminology conflicts in the concepts of the SQMMs have been investigated by exploring the origin of these concepts. In Section 4.2, the structure and content of the determined quality models are analyzed in detail, taking into account the results of the SLR study explained in Section 3.2. In Section 4.3, a mapping process is performed between the terms of the quality models and the concepts of meta-models that are free of inconsistency.

4.1. Terms Analysis of Software Quality Meta-models

In this section, the concepts of the SQMMs are analyzed in detail before the concepts of the SQMMs are matched to the terms of the OSS quality models. An SLR study [37] has been carried out to analyze the structure and content of quality meta-models as described in Section 3.1. In an RQ of this study, the frequency of concepts used in the meta-models

has been analyzed. These analysis has indicated that there might be inconsistencies in the concepts of the meta-models. Therefore, this analysis has triggered us to investigate the concepts of the meta-models in detail before the matching process.

The SQMMs create a common understanding between stakeholders for proper quality management throughout the entire life of a software product. However, the terminology of the SQMMs must be consistent among themselves in order for the SQMMs to serve their purposes properly. Therefore, in this section, inconsistencies, commonalities, and terminology conflicts in the SQMMs proposed for OSS as well as the custom type of software, are analyzed. This is intended to explore inconsistent terminology in the SQMMs for future proposals as well as to help us determine the terminology of the OSS-QMM presented in Section 5.

Since the meta-models for OSS quality are seldom, they are not likely to bring sufficient information to create the aforementioned background. Thus, meta-models proposed for the custom type of software have also been included in the analysis since they have been frequently taken as the base for OSS quality models and, thus, are related to OSS in some parts. In this context, meta-models proposed for the custom type of software have been identified from the primary studies of the SLR [37]. As specified in Section 3.1.1 (in Fig. 3.1 (b)), more than half of the meta-models (18) have been proposed to cover all types of software, and only 2 of them have been proposed for OSS. As a result, a total of 20 meta-models have been analyzed in this section. That is, meta-models proposed for other types of software (e.g., commercial-of-the-shelf software (COTS) or web services) have not been included in this analysis.

It is also necessary to analyze in detail the international standards or proposals (already listed in Table 2.7) as references to the terminology of the SQMMs and to understand any inconsistencies in terminology. The terms of the SQMMs have been generally derived from these standards or proposals, and Fig. 4.1 shows the percent distributions of the sources. This figure has been created based on the number of international standards or proposals which have been taken as sources for the terms used in the SQMMs. These terms are listed in Table 4.1. It should be noted that a term used in SQMMs can have more than one source, as shown in the column entitled "source of terms" in Table 4.1. Also, if a term does not have any source, it is considered a "new term" in the second column of the table. Terminology conflicts and inconsistencies are not only between the international standards of different organizations but also among those of the same

organization [39]. While there are inconsistent terms among the standards considered mature, it is perfectly normal to have inconsistent terms among the SQMMs that are not as mature as the standards, considering especially that the meta-models have not been validated by designing real-world cases, as previously mentioned in Section 2.4.4.

In Fig. 4.1, it is addressed to what extent the SQMMs use the concepts of the standards or proposals in their structure. As shown in the figure, among the standards and proposals; 15 (18%) of all the terms used in the SQMMs have been directly taken from the concepts of ISO/IEC 15939 that are followed by ISO/IEC 14598 with 11 terms (13%). Also, the SQMMs employed the least number of terms from IEEE 610.12 (with 5%) and then from IEEE 1061 (with 6%) and Briand (with 6%). Apart from these, a quarter (%26) of the terms are new, which have been not transferred from any standard or proposal. It is seen that the SQMMs employed more terms from ISO/IEC 15939 and ISO/IEC 14598 since these are software measurement and software quality evaluation standards, respectively.

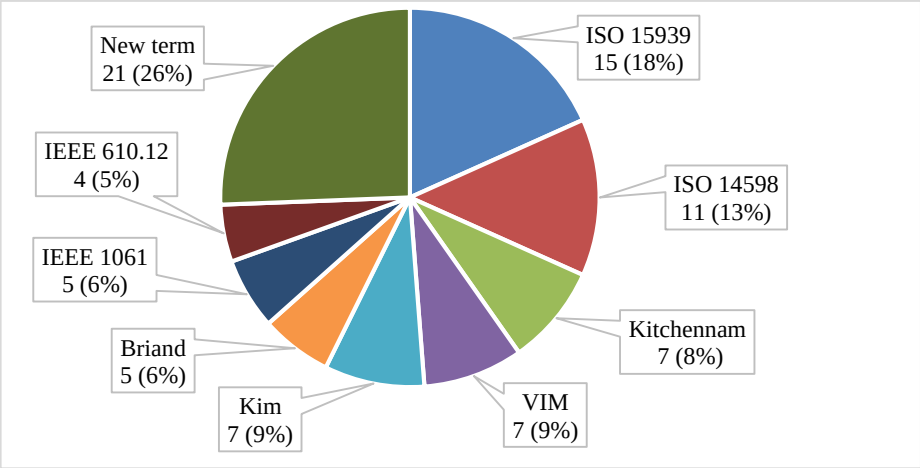


Figure 4.1. Percent distribution of sources (standards and proposals) that contribute to the terminology in SQMMs

Despite the fact that there are studies conducted to investigate inconsistencies among the vocabulary of international standards, no study has been found concerning the inconsistencies among the terms of the SQMMs. Therefore, Table 4.1 has been created to see all the terminology used in the SQMMs. In the first column of the table, the terms of the SQMMs are categorized according to the most frequently used ones among their synonyms. The definition of the terms in these standards or proposals is given in [181]. Also, in the first column, the aggregation (**Agg)** of each term in each category are listed to denote sub-categories or aggregated concepts under that category. In the 2nd column, the terms are classified according to their properties that address how they have been

transferred from the sources. The *adopted term*, *adapted term*, and *new term* are used for this classification. The terms taken directly from the sources (standards or proposals) without any changes, including their definitions, are classified as "adopted term". The terms borrowed from the sources either by changing their definitions or original names are classified as "adapted term". The terms not transferred from any source are classified as "new term". In the 3rd column of Table 4.1, synonymous terms used for each category in different SQMMs are listed. This column is important to see the inconsistencies among the terms of the different SQMMs. The 4th column lists the synonyms with which the SQMM elements in each category appear in different standards or proposals. This column is important to see the inconsistencies among the terms of the different standards or proposals. In the 5th column, the sources (standards or proposals) from which the SQMM terms in each category have been adapted or adopted are listed. In the 6th and last column, the standards or proposals that differently describe the SQMM terms in each category are listed. An important point here is that if a standard or proposal exists in the 5th column but not in the 6th column, it means that the term has been used in the source but not defined in that source.

Table 4.1. List of concepts in SQMMs and analysis of their inconsistencies

Category of SQMM Element		**Properties of Terms**	**Synonyms in Different SQMMs**	**Synonyms in Different Standards**	**Source of Terms**	**Standards defining terms differently**
Viewpoint		New term	View Quality goal			
Quality Requirement		New term				
Development Phase		New term				
Information Need		Adopted term	Need Purpose Target	**Briand:** Corporative objective **Kim:** Quality requirement	**ISO/IEC 15939** **Briand** **Kim**	**ISO/IEC 15939** **Kim**
Entity		Adopted term	Quality entity Entity type Measurable entity Artifact Component Entity class Software entity	**Kitchenham:** Project object occurrence	**ISO/IEC 15939** **Briand** **Kitchenham** **Kim** **IEEE 610.12**	**IEEE 610.12** **ISO/IEC 15939**
Agg.	*Derivation*	New term				
	Behavior	New term				
Quality model		Adapted term	Quality framework Quality	**Kitchenham:** Development model **Kim:** Enterprise quality model **IEEE 1061:** Metrics framework	**ISO/IEC 14598** **Kitchenham** **Kim** **IEEE 610.12**	**ISO/IEC 14598** **Kim** **IEEE 610.12**
Characteristic		Adapted term	Attribute Factor Property Quality-carrying property Fact Quality aspect Quality factor	**VIM:** Measurable quantity **Kitchenham:** Generic attribute **Kim:** Measured attribute	**ISO/IEC 14598** **VIM** **ISO/IEC 15939** **IEEE 610.12** **Briand** **Kitchenham** **Kim** **IEEE 1061**	**IEEE 1061** **IEEE 610.12** **ISO/IEC 15939** **ISO/IEC 14598** **VIM** **Kim**
Agg.	*Sub*-characteristic	New term	Sub-attribute Sub-factor Base attribute Derived attribute			

Measurable Concept		Adopted term			**ISO/IEC 15939**	**ISO/IEC 15939**
Measure		Adapted term	Metric	**Kitchenham:** Development model, Element measure type **IEEE 1061:** Metric **IEEE 610.12:** Metric **ISO/IEC 14598:** Metric	**ISO/IEC 14598** **IEEE 610.12** **Briand** **Kitchenham** **IEEE 1061**	**ISO/IEC 14598** **IEEE 1061** **IEEE 610.12**
Agg.	*Base measure*	Adapted term	Base metric	**IEEE 1061:** Direct metric **ISO/IEC 14598:** Direct measure **VIM:** Base quantity	**VIM** **ISO/IEC 15939** **ISO/IEC 14598** **IEEE 1061**	**VIM** **ISO/IEC 15939** **ISO/IEC 14598** **IEEE 1061**
	Derived measure	Adapted term	Derived metric Composed metric	**VIM:** Base quantity **ISO/IEC 14598:** Indirect measure	**VIM** **ISO/IEC 15939** **ISO/IEC 14598**	**VIM** **ISO/IEC 15939** **ISO/IEC 14598**
	Indicator	Adopted term			**ISO/IEC 15939** **ISO/IEC 14598**	**ISO/IEC 15939** **ISO/IEC 14598**
Measurement Approach		New term				
Agg.	*Measurement method*	Adopted term			**VIM** **ISO/IEC 15939**	**VIM** **ISO/IEC 15939**
	Measurement function	Adapted term			**ISO/IEC 15939**	**ISO/IEC 15939**
	Analyses model	Adapted term	Analysis decision		**ISO/IEC 15939**	**ISO/IEC 15939**
Measurement Results		Adapted term		**ISO/IEC 14598:** Measure **ISO/IEC 15939:** Measure **Kitchenham:** Recorded value **Kim:** Measurement point **IEEE 1061:** Metric value	**ISO/IEC 15939** **ISO/IEC 14598** **Briand** **Kitchenham** **Kim** **IEEE 1061**	**ISO/IEC 15939** **ISO/IEC 14598** **Kim** **IEEE 1061**
Measurement		Adapted term			**ISO/IEC 15939** **ISO/IEC 14598** **VIM** **Kim** **IEEE 1061**	**ISO/IEC 15939** **ISO/IEC 14598** **VIM** **Kim** **IEEE 1061**
Agg.	*Measurement data*	New term				
Decision Criteria		Adopted term		**ISO/IEC 14598:** Rating Level	**ISO/IEC 15939** **ISO/IEC 14598**	**ISO/IEC 15939** **ISO/IEC 14598**
Agg.	*Interpretation rule*	New term				
Instrument		New term	Tool			
Impact		New term				
Measurement Scale		Adopted term	Type of scale	**Kitchenham:** Generic scale range **VIM:** Reference-value scale	**ISO/IEC 14598** **VIM** **ISO/IEC 15939** **Kitchenham**	**ISO/IEC 14598** **VIM** **ISO/IEC 15939**
Unit of measurement		Adopted term	Unit	**Kitchenham:** Generic unit **ISO/IEC 14598:** Unit	**VIM** **ISO/IEC 15939** **ISO/IEC 14598** **Kim** **Kitchenham**	**VIM** **ISO/IEC 15939** **ISO/IEC 14598**
Evaluation		New term	Evaluation method Assessment model Assessment type			
Agg.	*Text evaluation*	New term				
	Manual evaluation	New term				
	Form-based evaluation	New term				
	Impact evaluation	New term				
	Quality aspect evaluation	New term				
Evaluation Result		New term	Quality aspect evaluation results			
Agg.	*Single-measure evaluation results*	New term				
	Multi-measure evaluation results	New term				
	Impact evaluation results	New term				

A term can have more than one definition by standards or proposals, as also seen in Table 4.1. For example, the most defined terms (with their frequencies) are: "characteristic" (6), "measurement" (5), "measurement results" (4), "base measure" (4), "measure" (3), "derived measure" (3), "quality model" (3), "scale" (3) and "unit of measurement" (3). These are the most essential terms of the measurement process, and the last column in Table 4.1 shows that there is a lack of agreement even in the original sources to define the same term. Also, it is observed that there are 24 cases of synonyms in the standards or proposals, which confirms the lack of consensus among them in terminology. Inconsistencies, commonalities, and terminology conflicts in all these standards or proposals are reflected in the SQMMs, and accordingly, there are 38 cases of synonyms for 15 terms in the SQMMs. In addition, it is observed that 17 (45%) of the terms have been transferred from the sources directly (8 adopted terms) or with changes (9 adapted terms), and 21 (55%) of them have been not transferred from any source (i.e., new term).

As a result, in this section, analyses are performed by going down to the origins of concepts in meta-models proposed for custom types of software. In this way, it is aimed to eliminate inconsistencies, commonalities, and conflicts between the concepts of the meta-models before the matching phase. These concepts, free of these inconsistencies, form the basis of the terminology of our meta-model that is developed within the scope of the thesis. Apart from them, these analyses provide guidance in eliminating the inconsistencies in international standards from the past to the present.

4.2. Detailed Analysis of the Structure and Content of Software Quality Models

In this section, OSS quality models and well-designed quality models referenced by these models are analyzed prior to the matching process between the terms of OSS quality models and SQMMs. As mentioned in Section 3.2, OSS quality models have been analyzed in the SLR study [30]. Based on the outputs of this SLR study, firstly, the most important OSS quality models and well-designed quality models referenced by these OSS models have been identified. Then, these quality models have been classified according to their structure, behavior and evaluation aspects. Finally, based on these classifications, the quality models have been structurally analyzed and fitted into a common structure. As it is well known, quality models are instances of meta-models. Therefore, these analyses are important in shaping the structure of the OSS-QMM developed within the scope of this thesis and in shaping the structure of SQMMs to be developed in the future.

Also, they are important to understand the common structure of OSS quality models in detail, prior to the matching process.

The rest of this section is organized as follows: In Section 4.2.1, the most important quality models taken as reference in the design of our meta-model are identified. In Section 4.2.2, these identified quality models are classified in detail prior to the matching. In Section 4.2.3, a structural analysis of quality models is performed, and these models are fitted into a common structure.

4.2.1. Determination of QMs to be Taken as Reference in the Design of the OSS-QMM

The primary motivation behind this thesis is to enable the proposal of standardized quality models that can perform comparable measurements for OSS. In this context, SQMMs are important because they may be used to standardize the quality models. To propose a comprehensive quality meta-model, the structure of existing quality models must be well understood since the quality models are the instances of the SQMMs. In this regard, we have conducted an SLR study [30] to characterize the existing quality evaluation models or frameworks (QEMoF) for OSS and to comprehensively examine their content and structure for identifying the gap between theory and practice. In this SLR study, the results are presented together with evaluation factors (EFs), which are used to assign a quality score for each OSS quality model. In another saying, in this SLR study, an RQ is asked to understand the overall quality of the QEMoF, based on a number of RQs answered so far as the evaluation criteria. In this context, among the RQs raised in this study, those aimed at evaluating the quality of the QEMoF have been determined. The list of all RQs is already given in Table 3.4 for this SLR study. The list of EFs used in the evaluation is given in Table 4.2, along with the scoring rules and the RQs taken as reference for the factors. While determining the scores, the EFs are weighted between the points 0-2 (2 if the EF is fully satisfied, 1 if partially satisfied, and 0 if not satisfied). The first author of the SLR study assigned a score for each EF, and then the second author performed a peer-review for the assigned scores. Then, the conflicts among the authors have been resolved by holding discussions.

The quality scores of each OSS quality model are given in the SLR study [30]. Among the OSS quality models with the highest scores in this study, the most cited and used five OSS quality models in the literature have been determined and listed in the last five rows

of Table 4.3. Also, due to their importance in the community, the OSS quality models examined within the scope of this thesis have been the subjects of systematic studies (i.e., systematic mappings and systematic literature reviews) such as [24][29] and the comparison studies such as [9][182-183]. These models fall into the "tailored" quality models category because of their construction based on the basic quality models (e.g., ISO/IEC 9126 [21] and Boehm [22]) and their particular application domains (e.g., a specific quality characteristic of OSS product [27-28]). Details of tailored quality models will be given in Section 4.2.2.3.

Table 4.2. List of Evaluation Factors (EFs) with referenced RQs and scoring rules

EF #	Source	Concern	Scoring Rules
EF-1	*RQ1.2*	*Formal representation*	If the QEMoF is represented formally, it is weighted as **2** points. If it is not, weighted as **0** point. There is no partial condition.
EF-2	*RQ1.4*	*Tool support*	If the QEMoF is supported by a tool, it is weighted as **2** points. If it is not, weighted as **0** point. There is no partial condition.
EF-3	*RQ2.1*	*Underlying standard or quality model*	If attributes of the QEMoF are mostly derived from a known standard/model (such as ISO 9126, ISO 25010 or CMMI), it is weighted as **2**. If they are not, weighted as **0**. If only a few of attributes are derived from a known standard, then it is weighted as **1**.
EF-4	*RQ2.3*	*OSS evaluation aspects*	If the QEMoF covers all evaluation aspects of OSS, it is weighted as **2** points. If it does not, weighted as **1**. Since all the QEMoF are assumed to evaluate the OSS from at least one aspect, 0 is not given as a weight.
EF-5	*RQ3.1*	*OSS evaluation procedure*	If the evaluation procedure of the QEMoF is adequately described, it is weighted as **2** points. If it is not, weighted as **0** points. If it is described at some degree but not in detail (i.e. only partially), then it is weighted as **1** point.
EF-6	*RQ3.2*	*OSS evaluation demonstration*	If a demonstration of the evaluation using the QEMoF is adequately provided, it is weighted as **2** points. If it is not, weighted as **0** point. If it is provided at some degree but not in detail, then it is weighted as **1** point.
EF-7	*RQ4.5*	*Quality evaluation scope*	If the QEMoF covers at least half of the quality characteristics, it is weighted as **2** points. If it does not, weighted as **1**. Since a QEMoF is considered to cover at least one quality characteristic, **0** is not given as a weight.
EF-8	*RQ4.9*	*Automation in data collection*	If data is collected automatically in the QEMoF, it is weighted as **2**. If data is collected manually, it is weighted as **0**. There is no partial condition.
EF-9	*RQ4.10*	*Skill level required for evaluation*	If the required skill level for using the QEMoF is low, it is weighted as **2**. If it is high, weighted as **0**. If the required skill level is medium, then it is weighted as **1**.

Basic quality models, on the other hand, have been mostly adopted for commercial products and therefore overlooked some specific properties of OSS. Nevertheless, the basic quality models, in addition to the OSS-specific ones, have also been analyzed in this research for several reasons: they have been widely studied in the literature, have provided partial evaluation for OSS quality, and have formed the basis of the OSS quality models thanks to their well-designed structure as specified in the SLR study [37]. In the

SLR study [37], the basic models that OSS quality models refer to have been investigated as well. The results have indicated that most of the OSS quality models are based on basic models. Among the basic quality models, based on the results of the SLR study [37], the models most commonly taken as basis by OSS quality models have been identified and listed in the last five rows of Table 4.3. Also, due to their importance, they are the most cited and used basic quality models in the literature. Consequently, a total of ten quality models, the first five being the basic quality models and the next five being the quality models tailored as specific to OSS, have been analyzed in this study (as already listed in Table 4.3). Among the basic models, ISO/IEC 9126 quality model was withdrawn and replaced by ISO/IEC 25010, which has many common quality characteristics with it. However, within the scope of this thesis, ISO/IEC 9126 has been included rather than ISO/IEC 25010 since the results of the SLR study have indicated that the majority of OSS quality models have been directly derived from ISO/IEC 9126, and there has been no model yet derived from ISO/IEC 25010.

4.2.2. Classification of Quality Models to be Taken as Reference

Many quality models have been proposed in the literature, which serves the same application domain and even the same type of software products. As such, it has become a challenging task to compare the results of the measurements performed by using these quality models. Likewise, there are many quality models in the literature for evaluating OSS. Therefore, before proposing further quality models for OSS, it is necessary to review and classify the quality models that have been proposed in the past and whose results cannot be currently compared. More specifically, developing a comprehensive OSS-QMM may support the development or revision of the OSS quality models with a standard structure, content, and terminology and, in turn, may reduce potential conflicts and confusion in future proposals. In this context, examining the structure of the previously proposed quality models in detail will support the validity, consistency, and comprehensiveness of the OSS-QMM to be developed. Accordingly, in this section, classification and analysis are performed before eliciting common structures of the quality models that have been proposed for OSS quality or taken as the basis for their development.

In this context, the ten quality models determined are classified in terms of their structure, behavior, application domain (i.e., basic and tailored), and evaluation aspect. Structural classification enables realizing the importance of the hierarchical structure used in quality

models and establishing a common structure of the quality models on this basis. This classification is important in shaping the structure of our meta-model. Behavioral classification enables to reveal the approaches of quality models to software quality. This classification is important in shaping the approach of our meta-model to OSS quality. This is because a comprehensive model should include concepts for both defining and assessing quality. Basic and tailored classification enables understanding the basics of the OSS quality models, examining well-designed quality models as well as OSS quality models and, thus, shaping the structure of the OSS-QMM. Classification according to the evaluation aspect enables understanding of the OSS aspects evaluated in OSS quality models and shaping the content of the OSS-QMM.

Table 4.3. Classification of SQMs w.r.t structural, behavioral, and basic/tailored properties

Models/ Category	**Structural**		**Behavioral**		**Basic and tailored**		
	Hierarchical	Dynamic	Definition	Assessment	Basic	Tailored	Based on (If tailored)
McCall	✓		✓		✓		
Boehm	✓		✓		✓		
Dromey	✓	✓	✓		✓		
Furps	✓		✓		✓		
ISO 9126	✓		✓		✓		
OSMM	✓		✓			✓	ISO 9126
QSOS	✓			✓		✓	ISO 9126
OpenBRR	✓		✓	✓		✓	ISO 9126 and OSMM
SQO-OSS	✓			✓		✓	ISO 9126 and OSMM
QualOSS	✓			✓		✓	OSMM, OpenBRR and QSOS

4.2.2.1. Structural Classification

In literature, each quality model is composed of a set of building blocks, including quality objectives, factors, criteria, sub-criteria, and metrics [84][184]. The names of these building blocks may vary in different models. For example, characteristic, attribute, or factor can be used interchangeably. The organization of these building blocks and their interactions with each other are examined as a structural classification [184]. In this context, quality models examined in this thesis are classified as having hierarchical or dynamic structures.

Hierarchical quality models are the models that build the quality of the software in a hierarchical structure of building blocks. The main purpose of this structure is to decompose the concept of quality into some quality attributes so that each attribute covers a certain aspect of product quality [84]. The general structure of hierarchical models is

shown in Fig. 4.2. In hierarchical quality models, quality attributes are generally quite abstract, so it is not possible to evaluate these attributes directly. Therefore, these are decomposed into less abstract forms known as sub-attributes, as shown in the figure. For example, in ISO/IEC 25010 quality model, the "maintainability" attribute, which can be defined as "the ease of change to the desired properties of the software after its delivery", is decomposed into five sub-attributes of modifiability, reusability, testability, analyzability, and modularity. Considering that these sub-attributes are still abstract and cannot be measured directly, it is necessary to associate each sub-attribute with a set of metrics that enable concrete measurements. Although these metrics provide concrete results within the quality models, interpretation of the results obtained is not easy as they are not fully covered in all the quality models. The list of quality models with a hierarchical structure is included in Table 4.3.

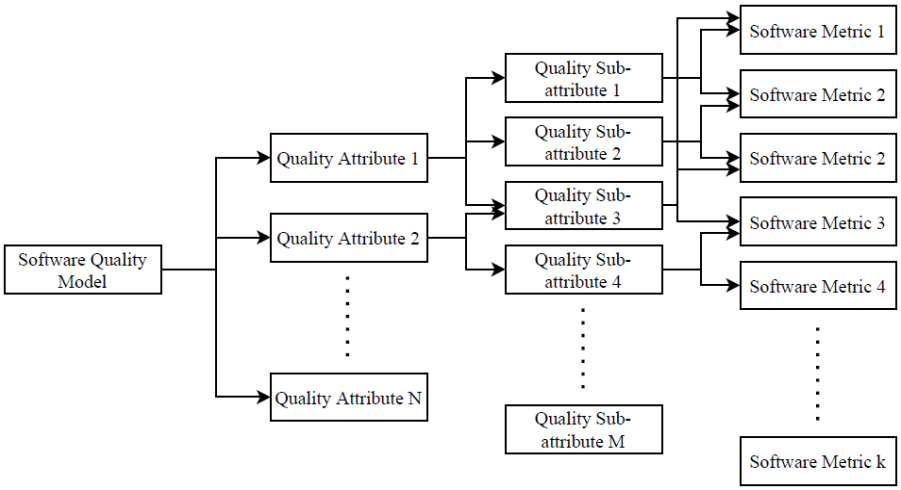


Figure 4.2. The general structure of hierarchical SQMs

Dynamic quality models state that the quality evaluation process of each software product is different and that dynamic development of quality attributes for the evaluation process is required. This type of models (e.g., proposed by Boehm et al. [185]) focuses on the relationship between attributes and sub-attributes to provide flexibility in the evaluation of different software products [186]. Although the dynamic quality models are not as comprehensive and abstract as the meta-models, they support the meta-modeling logic because they provide flexibility in the evaluation process. More specifically, the dynamic models concentrate on the relationship between building blocks such as attributes and sub-attributes, while meta-models are comprehensive enough to create consistent quality models and focus on a variety of building blocks covering all quality engineering tasks [84]. The only quality model that falls into this category is Dromey's quality model [187].

As also shown in Table 4.3, a quality model can fall into more than one category in structural classification.

4.2.2.2. Behavioral Classification

Although the common purpose of software quality models is to evaluate the quality of software products, it is a challenging process to compare these models with each other since they have diversity in approaches to defining or evaluating quality. Some of them are used for definition [21], and some of them are used for assessment [25] of software quality. Therefore, under behavioral classification, the quality models are considered as definition and assessment quality models, as also shown in Table 4.3. Despite the fact that definition and assessment are known as different types of activities, definition models and assessment models are dependent on each other. In other words, quality assessment of software products without an accurate definition of software quality is not possible. Definition quality models and assessment quality models are explained in detail in the following paragraphs.

Definition quality models, in the most general sense, are used mainly to describe or define quality [21][185]. They guide the use of the quality model for constructive quality assurance [187-188]. In order to have a high-quality system, the definition models should provide some direct suggestions and comprehensive definitions for different software development processes. In the requirement phase, all software requirements, as well as quality requirements and methods agreed with a customer about the concept of quality, are identified [189]. During the design and implementation phases, they are mainly considered a base for the identification of the designing and programming standards [185] to obtain a quality product [84]. Also, the majority of them focus on taxonomy and provide a guideline about the hierarchical decomposition of quality attributes. However, generally defined guidelines are not followed for decomposition and can be arbitrary in most of the definition models [145], which causes many critical problems. The ambiguous decomposition can cause overlapping between different quality attributes, which in turn causes redundancy due to multiple additions of the similar or same attributes, and makes positioning the quality attributes challenging [189]. Therefore, the guideline documents of the quality models, which provide communication with evaluators, play an important role. However, the rationales behind rules that are used to decompose building blocks are not generally explained in the guidelines, or the guidelines generally are not sufficiently detailed and concrete. In this context, considering the above shortcomings, it is

understood that there is a need within the quality models that explain the elements they use in their guidance documents in detail. The list of quality models that fall into this category is given in Table 4.3.

Assessment quality models are often considered as extensions of the definition models and are used to evaluate the quality of products that are characterized and defined by the definition models. This type of quality model is considered as the basis for the evaluation of quality, often by using some automatic analysis tools or by performing manual reviews. Therefore, assessment models monitor and control internal measures that can affect external properties [189]. Also, these models are often perceived to include mathematical models that aggregate software metrics to quality characteristics. In this way, they determine the values of quality characteristics. In this type of model, generally, each quality attribute is decomposed into sub-attributes, and each sub-attribute is mapped to some quality metrics for use in the assessment process by following the guidance provided by the definition models. Since these metrics often allow measurement directly, results are obtained in some scale within assessments by using measurement-based approaches. There is a considerable number of software metrics that have been proposed in the literature. All of these software metrics are difficult to cover in the quality models, and although some are defined, the quality models fail to give a detailed account of the impacts that specific metrics might have on software quality [188-189]. Also, because of the deficiency of clear semantics, the aggregation of metric values along the hierarchical levels may become problematic [188]. In addition, some of the metrics lack clear validation, which in turn threatens the validity of the measurement results. The list of quality models that fall into this category is given in Table 4.3. As shown in the table, a quality model can fall into more than one category in behavioral classification.

4.2.2.3. Basic and Tailored Classification

Basic quality models developed until 2001 are the models that mostly focus on a comprehensive evaluation and that aim to evaluate software products from many aspects [2]. This type of quality models is generally stand-alone, which means quality-related aspects determined by these models are based on their approach. Accordingly, a set of factors, criteria, and metrics are structured with the guidance of the determined aspects. Since basic models are the first known quality models, they are mostly considered as definition models that investigate meanings of quality for products aside from evaluating quality. Basic models form the basis of the tailored models, thanks to their well-designed

structure. However, these quality models have mostly been adopted to commercial software (e.g., COTS) and have overlooked some specific properties of OSS (e.g., community-based aspects) [15][27]. Thus, they do not provide sufficient support for assessing the quality of OSS [2][5][25]. Basic models are already shown in Fig. 2.4 and also listed in Table 4.3.

Tailored quality models developed after 2001 are mostly specific for a particular domain of application and focus on evaluating specific types of software products such as OSS [2]. In general, they are derived from the basic models by making some modifications to certain parts of the basic models. Tailored quality models have been proposed for the needs of organizations or software practitioners to perform a specialized evaluation on individual components [2][84]. For example, the MIDAS quality model proposed by Siemens [190] is used to design software products in their infrastructure in the industry. Consistent measurement results cannot be expected from such tailored models created within the needs of users unless these kinds of models are standardized. Like the basic models, tailored models are shown in Fig. 2.4 and also listed in Table 4.3. In addition, Table 4.3 shows in its rightmost column, which quality models are based on which other quality models. As also seen in that column, a tailored model could be derived from more than one basic or tailored model.

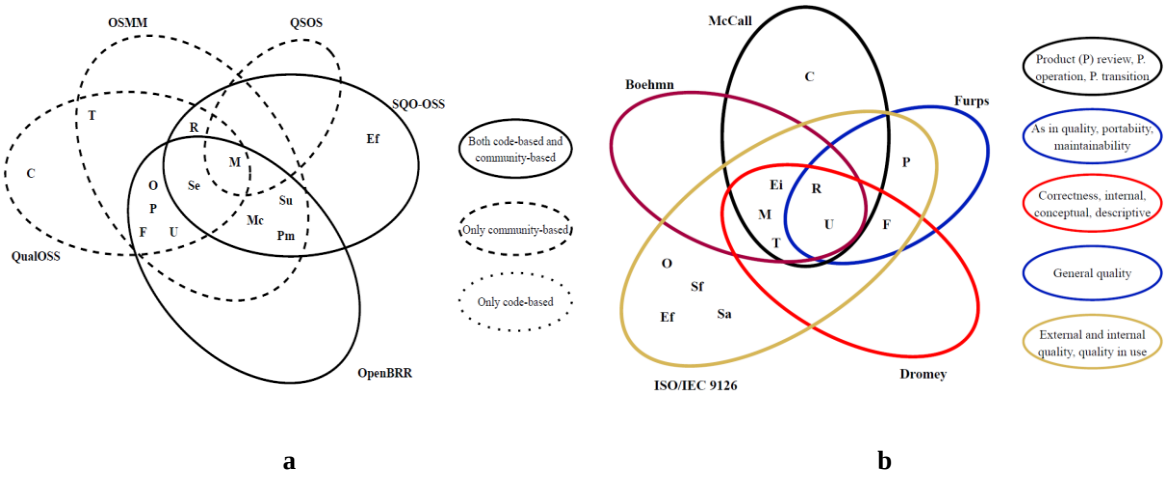
4.2.2.4. Classification According to the Evaluation Aspect and Evaluated Characteristics

The quality models determined within the scope of this thesis are classified according to their evaluation aspects and key quality characteristics they possess, as shown in Fig. 4.3. The quality characteristics used in the quality models are classified with respect to the quality characteristics of ISO/IEC 25010, which is the latest quality model. Apart from them, some quality characteristics such as maintenance capacity, sustainability, and process maturity [24][30] that belong to the community side of OSS are included. The definition of each community-related quality characteristic is given in Table 4.4. Also, abbreviations of the quality characteristics evaluated in each quality model are given just below the figure. As seen in Fig. 4.3 (a), in order to evaluate the quality characteristics, the OSS quality models allow measuring the code-based aspect, the community-based aspect, or both aspects of the OSS product. However, the situation is different in basic quality models since they do not provide sufficient support for evaluating the quality of OSS. This is because the basic quality models are mostly adapted for commercial

products and overlook some specific properties of OSS, e.g., community properties [5][30]. As it is not possible to obtain public information about the development details of the commercial software, the basic models group the quality characteristics considering the quality of the product output, as seen in Fig. 4.3 (b). For example, they group characteristics under "quality in use" considering quality when using the product, or under "product transition" considering adaptability of the product to new environments, etc. In Fig. 4.3 (b), these high-level characteristics covered by the models are specified using color-coding. Although the OSS quality models have been derived from the basic models, they fall apart from them at this point. That is, several types of data can be accessed with regard to the development details, such as code-based and community-based aspects, in OSS quality evaluation since the source code is open and historical data are stored in various cloud repositories (e.g., GitHub) belonging to the community [5][30]. Therefore, OSS quality models have modified their content to use all these relevant data for evaluation. We should note that the classification presented in this sub-section has served as a base for understanding the OSS aspects evaluated by the OSS quality models and for shaping the concepts of the OSS-QMM concerning these aspects.

Table 4.4. Description of community-related quality characteristics of OSS

Quality characteristics of the community side	Description
Maintenance capacity	It is the capability of a community to provide the resources needed for the maintainability of an OSS product over a period of time. It is mainly related to the number of contributors and to the amount of time that they are willing/able to contribute to the development effort. Data to monitor this effort can be obtained from databases such as mailing list, versioning logs, bug reports, and discussion forums.
Sustainability	It is the capability of a community to grow in terms of new contributors and regenerate by attracting and engaging new contributors to take the place of those leaving the community. It is also related to the heterogeneity of the community in addition to regeneration ability. For example, if a community of a project mainly consists of a particular company (i.e., non-heterogeneous) and the company withdraws its support, it is highly likely that the project will be stalled. As another example, if the same group of developers has been active for a long time as a result of monitoring the first and last contributions of the developers, this does not reflect a significant regeneration.
Process Maturity	It is the capability of a community to adopt and use standard practices in the development process, such as peer review of changes, planned releases, submission and review of changes, and provision of a test suite. That is, it is related to reaching development-specific goals (e.g., quality goals) in a consistent manner by following the determined process.



a **b**

Abbreviations: Maintainability **(M)**, Reliability (**R**), Functionality (**F**), Performance (**P**), Operability (**O**), Security **(Se),** Effectiveness **(Ef)**, Compatibility (**C**), Transferability (**T**), Usability **(U)**, Efficiency **(Ei**), Satisfaction **(Sa)**, Safety **(Sf)**, Maintenance Capacity (**Mc)**, Sustainability (**Su)**, Process Maturity (**Pm)**

Figure 4.3. Classification w.r.t evaluation aspects and quality characteristics of: **(a)** OSS quality models, and **(b)** basic quality models

4.2.3. Structure Analysis of SQMs, Including OSS Quality Models

The quality models determined within the scope of this thesis are explained together with the reasons for their inclusion. In this context, a total of 10 quality models, five being the basic quality models and five being the quality models tailored as specific to OSS, have been analyzed in this thesis, as shown in Table 4.5. As shown in the table, aside from OSS quality models, the structures of some cornerstone basic quality models have also been examined since they provide partial evaluation opportunities for OSS as well as form the basis of the OSS quality models. In Section 4.2.1.1, the structure of the determined quality models is classified before their structural analysis is performed. Therefore, in this section, guided by this classification, a structural analysis is carried out using the common hierarchical structure of the quality models. In other words, with the purpose of defining a common language and using it in proposing or revising the quality models in the future, the structures of the quality models listed in Table 4.3 are investigated and compared. This effort is important in shaping the structure of our meta-model to develop within the scope of this thesis. That is, this way, a solid basis is formed for developing the OSS-QMM. It may also enhance the development of individual OSS quality models.

Table 4.5. Structure comparison of SQMs (the first five are basic, and the last five are specific to OSS)

Level/ Model	McCall	Boehm	FURPS	Dromey	ISO/IEC 9126	OSMM	QSOS	OpenBRR	SQO-OSS	QualOSS
Level 1	View	View	View	View	View	View	View	View	View	View
Level 2	Major perspective	High-level characteristic	-	Product properties	Characteristic	Group	Top-level-criteria	-	Evaluation aspect	Evaluation aspect
Level 3	Factor	Intermediate-level characteristic	Characteristic	Quality attribute	Sub-characteristic	Indicator	Criteria	Characteristic	Quality attribute	Characteristic
Level 4	Criteria	Primitive characteristic	Sub-characteristic	Sub-attribute	Quality attribute	Sub-indicator	Sub-criteria	Sub-characteristic	Sub-attribute	Sub-characteristic
Level 5	Metric	Metric	Metric	Metric	Metric	Metric	Metric	Metric	Metric	Metric

Considering the comparison of structures of the quality models listed in Table 4.5, we realized that all the quality models are based on a common structure consisting of five levels. It has been observed that some quality models, from both basic and tailored models such as FURPS and OpenBRR, respectively, do not include model elements of Level 2 in their model structures. The levels of structure applied to all the quality models are explained below:

Level 1: Software quality is complex, multifaceted, and hard to define since the expectations of stakeholders are different from software products. Therefore, these stakeholders perceive software quality from their points of view. In this context, at Level 1, all quality models are shaped with their content according to a specific point of view, such as customer, manager, developer, tester, designer, etc.

Level 2: After determining the point of view that the software quality is evaluated from, it is determined which aspects of the product are evaluated. At Level 2, the evaluation aspects can have some synonymous words in the quality models, such as high-level characteristics, groups (product and application indicators), etc. Although each evaluation aspect has an impact on overall software product quality, most of the quality models focus on one or more aspects, such as community quality, quality in use, or service quality, rather than on evaluating overall product quality. Among the quality models, the only distinctions are FURPS and OpenBRR, which do not concentrate specifically on an evaluation aspect of the software quality.

Level 3: After determining the evaluation aspects of the software product, the quality attributes associated with these evaluation aspects are determined in the quality models examined. At Level 3, the quality attributes can have some synonymous words in the

quality models, such as factor, characteristic, criteria, or indicator. Quality attributes are measurable or testable concepts of software quality, and they are used to control quality and to determine how well the software product or system satisfies the needs of its stakeholders. However, despite the fact that quality attributes are defined as measurable concepts, they are generally quite abstract concepts that cannot be measured directly.

Level 4: After the quality attributes are associated with the evaluation aspects; at Level 4, these quality attributes are decomposed into sub-attributes in the quality models since the quality attributes remain abstract to evaluate directly. Quality sub-attributes can have synonymous words in the quality models, such as factor, sub-characteristic, sub-criteria, or primitive characteristic. Sub-attributes are defined for the quality attributes that represent a wide range of aspects of software use, in order to allow for valid measurements of compliance [84]. However, sub-attributes are still abstract to evaluate directly, so they can be considered as less abstract forms of quality attributes.

Level 5: After the sub-attributes are associated with the quality attributes; finally, at Level 5, sub-attributes are associated with software metrics that allow concrete measurements directly. Generally, the quality models use analysis tools to assign values to software metrics; however, the quality models of OSS use scoring criteria according to the rule sets defined in the models, especially for the metrics related to the community aspect. Since a quality sub-attribute is often associated with more than one software metric, the values obtained for all metrics are aggregated to obtain a single value for the quality measurement of the sub-attribute.

4.3. Mapping Process

Information systems researchers have proposed a variety of meta-modeling frameworks (e.g., [49-50]) in the literature. In this thesis, we have followed the meta-modeling framework based on the Meta-Object-Facility (MOF) standard in developing the OSS-QMM. As mentioned in Section 2.4.2, according to the basics of meta-models, the quality models are instances of a meta-model. This situation is explained in detail in the Meta-Object Facility (MOF) standard. For more information on the MOF standard, please see Section 2.4.3. The abstraction levels of the MOF architecture and the levels of modeling language are given in Fig. 2.10. The MOF standard has a four-layered architecture that includes, from the bottom to the up: M0 (run-time layer), M1 (model layer), M2 (meta-model layer), and M3 (meta-meta-model layer). In the MOF, layer M_i contains an instance

of layer $M_{i+1,}$ and layer M_{i+1} describes layer M_i, as shown in Fig. 2.10. That is, meta-models are defined as models of models, and a model is an instance of a meta-model. Accordingly, the model in level *(i)* is written in the modeling language described by the model in level *(i+1)*, as shown in Fig. 2.10. A modeling language consists of its *syntax* and *semantics*. The *syntax* describes the elements and rules for creating models and is described by grammar; the *semantics* describes the meaning of a modeling language and consists of a semantic domain and semantic mapping. As a result, according to MOF structure, the terms of the quality models of OSS (at layer M1) should be matched with the terms of the SQMMs (at layer M2).

In the matching process, each concept of a meta-model for a given domain should match one or more terms of the quality model for that domain. An example of the correct mapping process is shown in Fig. 4.4 (a). In addition, as shown in Fig. 4.4 (b), there should be no vocabulary that is not used in the concepts of the meta-model and the terms of the quality model. In this figure, term #3 of the quality model cannot be mapped to the SQMM and concept #2 of SQMM is unused. The matching process is exemplified in models for different application domains, as shown in Fig. 4.5. This figure supports Fig. 2.10. That is, each model is written in the modeling language described by the meta-model. Also, a meta-model determines the language concepts, the relationship between these concepts, matching rules and the transformation of model terms, in order to harmonize the domain's rules. In summary, considering Fig. 4.4 (a), at least a term in the quality model must match a concept in the meta-model. In this sense, it is reminiscent of surjective functions. That is, if the meta-model is represented as a domain and the quality model as a co-domain, every element of the function's co-domain is the image of *at least* one element of its domain [191]. Symbolically;

$$if\ f: X \to Y, \qquad then\ f\ is\ said\ to\ be\ surjective\ if$$

$$\forall y \in Y, \quad \exists x \in X, \quad f(x) = y$$

Here, $\boldsymbol{f}$ is a function, $\boldsymbol{X}$ is the domain (meta-model), $\boldsymbol{x}$ is an element of $\boldsymbol{X}$ (concept of meta-models), $\boldsymbol{Y}$ is the co-domain (quality model), and $\boldsymbol{y}$ is an element of $\boldsymbol{Y}$ (term of the quality model).

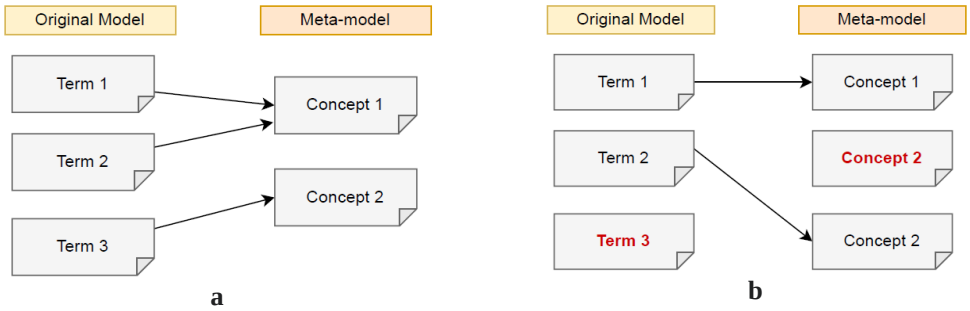


Figure 4.4. **(a)** An example of correct mapping, and **(b)** an example of incorrect mapping (there are unused terms and concepts)

In this context, a systematic way has been followed prior to the mapping process. As mentioned in Section 3.1 (Step-1), initial research has been conducted to examine the structure of the quality meta-models proposed for OSS and for the custom type of software. Considering the results of this research, the concepts used in these SQMMs have been categorized together with their synonyms and aggregations, as detailed in Section 3.1.4. As a result of the analysis performed in Step 1, the inconsistencies between the terminologies of the meta-models have been analyzed according to their meanings in the related meta-models and international standards. This analysis has provided knowledge about the meanings of the terms to be used in the OSS-QMM to be developed, in other words, its semantics. As mentioned in Section 3.2 (Step-2), research has been conducted to investigate the structure of OSS quality models. Considering the results of this research, it has been observed that all the quality models investigated have a common structure consisting of five levels, as already given in Table 4.5 (Section 4.2.3). In this way, the common structure of the quality models has been discovered, and the inconsistencies among the terms of the SQMMs have been eliminated. The analysis performed in Step 2 has provided knowledge about the structure of the OSS-QMM to be developed, in other words, its syntax.

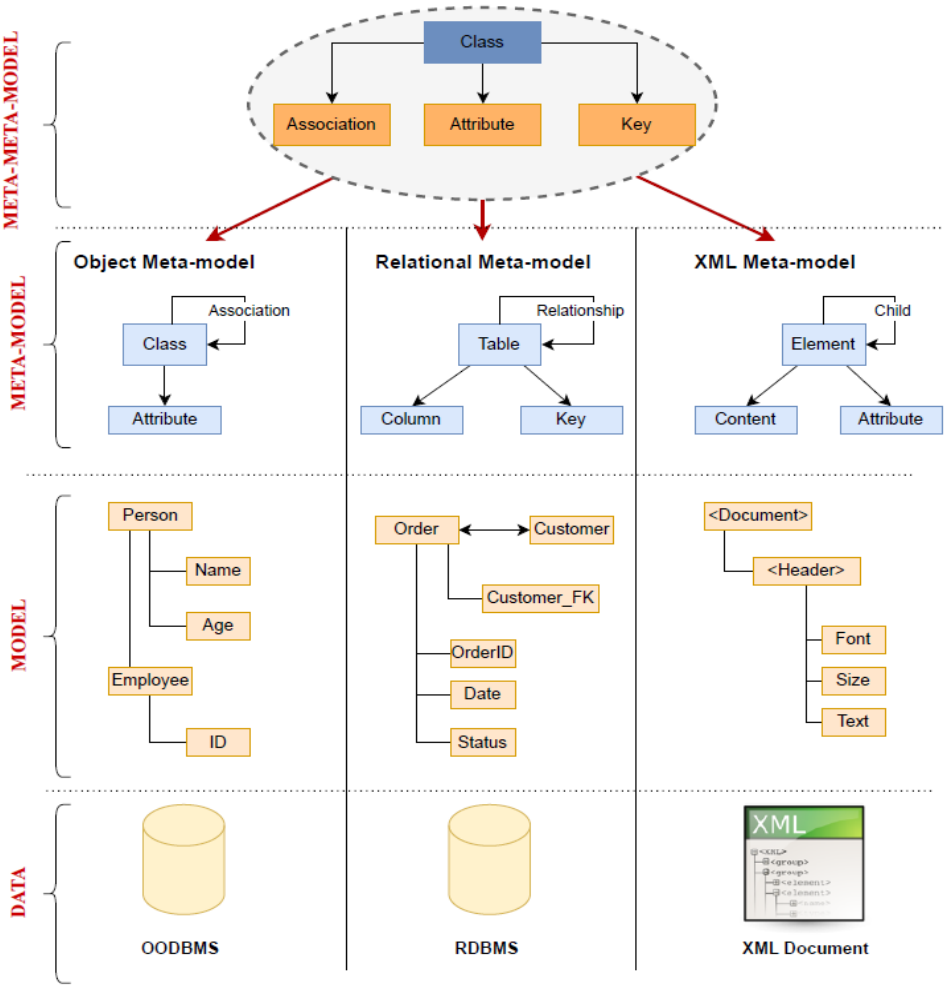


Figure 4.5. The examples of the model development process considering the mapping process

In this section (Step 3), then, the level-based matching process has been carried out between the terms of the OSS quality models and the terms of the SQMMs, as shown in Table 4.6, since ideally, a quality model should be an instance of a meta-model. This matching has been performed in accordance with the MOF standard [49]. That is, the terms of the quality models of OSS (at layer M1) are matched with the terms of the SQMMs (at layer M2) for the categories of SQMM elements specified in the first column of Table 4.1. This matching is an important step in that the OSS quality models to develop or revise will have a homogeneous structure and that the measurements performed using these quality models can be standardized. In this matching process, some accepted standards [114][126][192] have been taken as bases in the process of determining the terms of the SQMMs corresponding to each level. In addition, the intended usage of the terms in the SQMMs, the classification of the terms in some SQMMs [37-38], and a common output obtained from the definitions of these terms in the international standards

have been taken as reference in this matching. Moreover, the mapping process has been reviewed by the subject matter experts, as will be explained in Section 4.3.1 (Step 3.1).

4.3.1. Review of the Mapping Process by Experts (Step 3.1)

A mapping process has been carried out through a systematic process, taking into account the resources given above. Apart from them, a series of meetings have been held with subject matter experts, following an iterative process, as already shown in Fig. 1.2. In this regard, a total of four subject matter experts have been identified. Two of these experts have academic experience, and two of them have industrial experience. In determining these experts, it has been ensured that they had seven or more years of experience in the field of software quality and its evaluation. The following questions have been asked to the experts to obtain feedback;

- **Q1-** Do you think that the terms of the quality models are placed at the appropriate levels?
- **Q2-** Do you think the concepts of OSS quality meta-models are placed at the appropriate levels?
- **Q3-** Do you think the concepts of the OSS quality meta-model are placed in appropriate levels? (i.e., specification, measurement, and evaluation)

By considering these questions, a series of individual semi-structured interviews have been held with the experts to discuss the mapping process and obtain an answer for each question. Throughout these interviews, the meaning and intended use of each term in the quality model and each concept in the meta-model have been considered. Accordingly, the levels of each vocabulary in the matching process have been discussed with the experts, taking into account also their personal experiences. This step has progressed through the iterations of the review-and-revise process with respect to the suggestions of the experts, as shown in Fig. 1.2. In another saying, the meaning and intended use of each vocabulary has been reviewed by experts, and the mapping process has been revised in line with their suggestions. In this context, the review-and-revise process has continued until the experts have agreed that; each vocabulary has been placed at the appropriate level.

4.3.2. Performing the Mapping

Considering all these sources and expert opinions mentioned above, meta-model terms have been matched with the most appropriate terms in the levels of the OSS quality models. As a result of this process, the level-based mapping is given in Table 4.6. The table shows which concept in the meta-model have been mapped to which level in the quality model. It should be noted that the concepts belonging to each level in quality models are given in Table 4.5.

Table 4.6. Matching terms of OSS quality models and concept of SQMMs w.r.t levels

Level	Specification	Measurement	Evaluation
1	Development phase, Quality requirement, Viewpoint (**Syn:** View, quality goal)		*Evaluation aggregation**
2	Entity (**Syn:** Quality entity, Entity type, Measurable entity, Artifact, Component, Entity class, Software entity), Information need (**Syn:** Need, Purpose, Target), Quality model (**Syn:** Quality framework)		*Evaluation aggregation**
3	Characteristic (**Syn:** Attribute, Factor, Property, Quality-carrying property, Fact, Quality aspect, Quality factor)		*Evaluation aggregation**
4	Sub-characteristic (**Syn:** Sub-attribute, Sub-factor, Base attribute, Derived attribute)		*Evaluation aggregation**
5	Decision criteria, Impact, Measurable concept	Instrument (**Syn:** Tool), Measure (**Syn:** Metric), Measurement, Measurement approach, Measurement results, Measurement scale (**Syn:** Type of scale), Unit of measurement (**Syn:** Unit)	Evaluation (**Syn:** Evaluation method, Assessment model, Assessment type), Evaluation results (**Syn:** Quality aspect evaluation results)

In Table 4.6, the terms of the SQMMs for each level are categorized into three groups as *specification*, *measurement*, and *evaluation*, in order to make the matching process more systematic and comprehensible. These categories are identified in order not to put apples and pears in the same group. The terms grouped under "specification" are used to determine which aspect and what feature of the OSS product to measure. The viewpoint of stakeholders is taken into account in determining these characteristics. For example, the specification includes from which stakeholder viewpoint the product will be

evaluated, which entity of the OSS product will be measured, and which characteristics will be required to measure it. The terms under "measurement" are used to quantify some characteristics of an OSS product. Generally, some standardized tools are required for a consistent and meaningful measurement process. For example, software metrics such as lines of code (LOC), depth of inheritance tree (DIT), and cyclomatic complexity (CC) are measured, and some numerical values are obtained. These terms provide a solid base to perform an evaluation. Finally, the terms under "evaluation" are used to seek if the OSS is the best possible fit for the needs of evaluators by using measurement results. In other words, the terms classified under "evaluation" are used to interpret the numeric value obtained as a result of the measurement for any metric and to address whether this value is satisfactory or not. The important point is that although the terms related to the evaluation are usually considered at level 5, evaluation can be performed at any level with respect to the aggregation needs. However, measurement-related terms are matched to level 5 since measurement requires concrete data and takes place at the bottom-most level. This matching process is important in shaping the structure of the meta-model developed for OSS quality within the scope of this study as well as the ones to be developed in the future for OSS quality or other types of software quality. It is also useful to easily recognize the general abstract form of the quality models from the structure of the SQMMs.

5. OSS-QMM AND ITS DEVELOPMENT (STEP-4)

In this section, the OSS-QMM developed through a systematic process, which is the primary goal of this thesis, is described. Considering this systematic process, it can be easily understood that the meta-model developed is the result of a laborious process. In the following sub-sections, the phases of this laborious process are mentioned. In this context, firstly, the process of developing our meta-model is explained. Then, the refinement process of the proposed meta-model is discussed. Finally, the latest version of this developed OSS-QMM is presented.

5.1. Development Process of the OSS-QMM

In the process of developing the OSS-QMM, the ***step-based process for meta-model creation***, which is adapted from Beydoun et al. [44] and Othman et al. [45], has been followed. Details of this step-based process are given in Fig. 1.2. This process enables a systematic and well-founded meta-model to be proposed. Therefore, it has been widely used in the literature [54]. According to this process, the meta-model has been proposed based on the outputs of Steps 1, 2 and 3 and refined based on the outputs of Steps 4.1 and 5. The following paragraphs briefly summarize the OSS-QMM development process, details of which are given in Sections 2, 3, and 4.

The ***content*** of the OSS-QMM has been determined as the outputs of Step 1. As you will recall, in this step, the structure and content of existing SQMMs in the literature have been analyzed in depth. In this context, we carried out an SLR study [37] and analyzed 28 studies that proposed SQMMs. The SLR study [37] enhanced our domain awareness as recommended in [44] as an initial step for any meta-modeling process. In the SLR study [37], the frequency of the concepts in the SQMMs has been identified, and this indicated that there have been inconsistencies between the concepts of the SQMMs. This finding triggered us to investigate those inconsistencies. Therefore, inconsistencies, commonalities, and terminology conflicts in the concepts of the SQMMs have been investigated by exploring the origin of the concepts and then, the inconsistencies or conflicts have been eliminated. In this context, the international standards [113-117] or proposals [118-120], which have been taken as sources for the concepts used in the SQMMs, have been analyzed to obtain further evidence about the existence of these concepts. The usage purposes and meanings of the concepts in the SQMMs and their meanings in the international standards and proposals have been also analyzed. Finally,

the terms of the SQMMs have been categorized according to the most frequently used ones among their synonyms (see [181]). As a result of Step 1 (in Fig. 1.2), inconsistencies among the concepts of the SQMMs have been eliminated, which formed the basis for shaping the content of the OSS-QMM.

The ***structure*** of OSS-QMM has been determined as the output of Step 2. As you will recall, in this step, the content and structure of existing SQMs in the literature have been analyzed in detail. This is because the SQMs are the instances of the SQMMs according to the MOF standard [49]. Step 2 is about gathering information sources to be used in developing the OSS-QMM. In this regard, we carried out another SLR study [30] that examined 36 OSS quality evaluation models and frameworks (QEMoF). Based on this SLR study, a total of 10 well-designed and important quality models (i.e. five for OSS quality and five being basic models) have been determined. Details of these quality models and their determination are given in Section 4.2. These quality models have been investigated in depth to obtain a common structure of the OSS quality models and consequently to eliminate the heterogeneity in content and structure. It has been observed that all of these determined quality models have a common structure consisting of five levels. Details of the levels and the terms of the quality models at each level have been explained in Section 4.2.3. Finally, this 5-level structure formed the basis of shaping the structure of the OSS-QMM.

As a result of the analysis performed in Step 1, the inconsistencies between the terminologies of the meta-models have been analyzed according to their meanings in the related meta-models and international standards. This analysis has provided knowledge about the meanings of the terms to use in the OSS-QMM to be developed, in other words, its semantics. In Step 2, the structure of the quality models has been analyzed, and this analysis has provided knowledge about the structure of the OSS-QMM to be developed, in other words, its syntax. Then, the mapping process has been performed between the concepts of the meta-models and the terms of the quality models as the output of Step 3. A level-based mapping process has been carried out, as shown in Table 4.6. That is, the terms at each level of the quality models have been mapped to the concepts of the meta-models at that level. In the mapping process, several well-known standards [114][125][192], the classification of concepts in some studies [37-38], and the intended meaning of the concepts in the meta-models and international standards have been taken as references. In addition, in Step 3.1, a series of meetings have been held between the

subject matter experts during the mapping process, in iterations. By applying all these processes, the concepts of the meta-models have been mapped with the most appropriate levels in the hierarchy of the quality models. Finally, in order to make this mapping process more comprehensive and systematic, the concepts of the SQMMs have been classified under *specification*, *measurement*, and *evaluation*. The details of this classification and the efforts described above are explained in Section 4.3.

As a result, the OSS quality meta-model development process has begun, taking into account the steps (Steps 1, 2 and 3) mentioned above. First, an initial version of the OSS-QMM has been proposed by applying the sub-steps described in Section 5.1.1. Then, the refinement process has started for this proposed initial version. In this context, firstly, the expert opinion, the details of which are given in Section 5.2.1, has been utilized. Then, the validation process, the details of which are provided in Section 5.2.2, has been utilized. As a result of all these processes, the final version of the OSS-QMM presented has been proposed. Details are given in Section 5.3.

5.1.1. The Sub-steps of Development

The first three steps are summarized above. We have started Step 4 to propose the OSS quality meta-model after the completion of the first three steps. As seen in Fig. 5.1, the fourth step has consisted of three sub-steps, namely 4.1, 4.2, and 4.3. In order to increase the traceability, the relevant parts of Fig 1.2 are given in Fig. 5.1. It should be noted that in performing each sub-step, the review-and-revise process has been followed as shown in Fig. 5.1. In order words, a series of meetings have been held between the author of this thesis and subject matter experts to review the OSS-QMM. Then, the OSS-QMM has been revised according to the opinions of the subject matter experts. Details of these meetings are given in Section 5.2.1. Thus, the final decisions have been taken as the result of a series of iterations. In the following sub-sections, the sub-steps (i.e., Steps 4.1, 4.2, and 4.3) of Step 4 are explained.

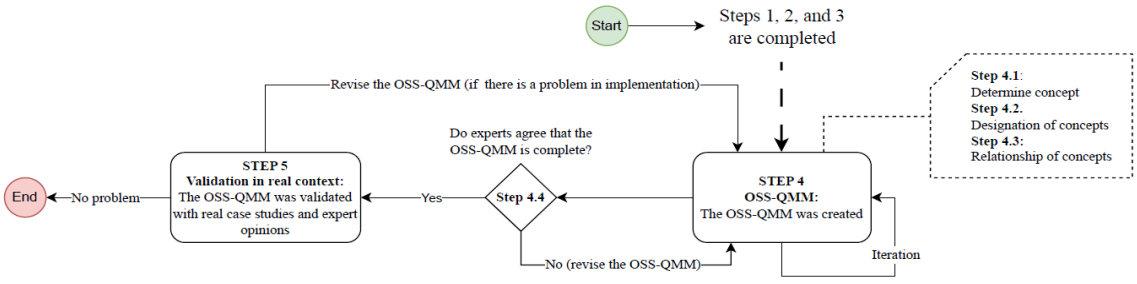


Figure 5.1. Corresponding parts of Figure 1.2 for developing the OSS-QMM

5.1.1.1. Determination of the Concepts

At this step of our OSS-QMM development process, the concepts to use in our meta-model have been determined. As shown in Fig 5.3, the determination of these concepts is a laborious process. In this context, first of all, the outputs of our first SLR study [37] have been used (Step-1). This SLR study has addressed the frequency of used concepts in the included meta-models and showed there are inconsistencies between the terms of the meta-models. Therefore, this result from the SLR study has triggered us to examine inconsistencies between the terms of the meta-models. In order to mitigate this deficiency, a study [54] has been conducted to analyze inconsistencies, commonalities, and terminology conflicts in the SQMM (Step-3). These analyses have been performed based on international standards and proposals since the concepts of the SQMMs have been generally derived from these standards or proposals. Details are given in Section 4.1. As a result of these analyses, concepts of the SQMMs have been categorized according to the most frequently used ones among their synonyms, and inconsistencies have been eliminated. The list of all concepts used in the SQMM is given in Table 4.1 with their origins. Also, the definition of the concepts in these standards or proposals is given in [181].

The custom type of software has been included in the analysis since they have provided partial evaluation for OSS quality. Then, the outputs of our second SLR study [30] have been used (Step-2). In this study, the requirements of these quality models have been analyzed in detail. Also, it has been observed that OSS quality models have a heterogeneous structure. Therefore, in another study [54], we have put effort to find commonalities in the structure of OSS quality models (Step-3). Then, it has been observed that important OSS quality models have a common structure consisting of five levels. Then, a series of meetings between the author of this thesis and his supervisor has been held, and accordingly, the initial concepts of the OSS-QMM related to OSS have been determined (Step-4.1). As can be seen in Fig. 5.3, in the process of determining these concepts, inconsistency-free concepts, the requirements and structure of the OSS quality models, and the common structure of OSS quality models have been taken into account. In addition to these inconsistency-free terms, new terms have been determined according to the requirements of the OSS quality models. These terms are grouped as "new term" in Table 4.1.

Then, the mapping process, which has an important contribution to the determination of the concepts, has been carried out, as shown in Fig. 5.3. In the mapping process, the terms of the determined quality models have been mapped to the determined concepts of the meta-models. Details of the mapping process are explained in Section 4.3 (please see this section). In the mapping process, as illustrated in Fig. 4.4, at least a term in the quality model must match a concept in the meta-model. That is, there should be no vocabulary that is not used in the concepts of the meta-model and the terms of the quality model. At this point, the answer to the following question has been sought: "*Are there any unmatched terms*?". As shown in Fig. 5.3, if the answer to this question is "Yes", the concepts are reviewed and then revised. If the answer to this question is "No", the process is continued with the designation of the concepts (i.e., Step-4.2) and then determining the relationships between the concepts (i.e., Step-4.3). These processes are described in Sections 5.1.1.2 and 5.1.1.3, respectively.

As shown in Fig. 5.3, then, the OSS-QMM has been refined in case there has been a problem with the outputs of the validation process (Step-4.4 and Step-5). Detailed information about the validation process is given in Section 6. During the validation process, as shown in Fig. 5.3, the answer to the following question has been sought: "*Is the OSS-QMM validated?*". As shown in the figure, if the answer to this question is "No", the concepts of OSS-QMM are reviewed and then revised. If the answer to this question is "Yes", the concept determination process is completed.

5.1.1.2. Designation of the Concepts

In the step-based meta-model creation process, it is necessary to designate the concepts after determining them. As can be seen in Fig. 5.3, a laborious process has been followed until the concepts of OSS-QMM have been identified. After the concepts have been determined, a mapping process has been carried out between these concepts of OSS-QMM and the determined quality models. If there is no problem in the mapping process, the determined concepts are designated. Also, as shown in Fig. 5.3, the determined concepts are revised in case of problems during the validation process. Accordingly, the designation of the concepts has been revised according to the added and removed concepts.

The results of our first SLR study [37] have indicated that the important meta-models (e.g., [44][140]) in literature have a layered structure. In other words, they have grouped

the terms in certain layers according to their content. Furthermore, meta-models developed using the "step-based meta-model creation process" in other application domains have also designated their concepts according to their needs. Because the designation of concepts is important for better management of concepts, understanding their differences and understanding their relationships with each other are necessary. For example, the meta-model developed for disaster management designated their concepts as: mitigation, preparedness, response or recovery [53]. Therefore, we have created three layers to designate our determined concepts in our OSS-QMM as: ***specification,*** ***measurement,*** and ***evaluation.*** Also, this classification has made the matching process in Step 3 more meaningful.

The concepts grouped under "specification" consist of the concepts defining the scope or objectives of the OSS evaluation process, as listed in Table 4.6. That is, these concepts describe which features of OSS products will be evaluated. Some example questions formed to determine the necessities to be evaluated belonging to the "specification" group are given in Fig. 5.2. For example, these concepts belonging to the specification are used to determine some OSS properties to be measured, such as the OSS aspect, quality characteristic to evaluate and quality requirements. It also contains a concept for determining the viewpoint to be taken into account in determining all these.

The terms under "measurement" are used to quantify some characteristics of an OSS product, as listed in Table 4.6. That is, in the measurement part, the numeric values are objectively assigned to the determined features of OSS in the specification part. Some example questions formed to assign a value to a characteristic of OSS are given in Fig. 5.2. For example, the concepts of the "measurement" stage consist of a set of measures, measurement functions, measurement methods and aggregating techniques. Generally, some standardized tools are required for a consistent and meaningful measurement process. For example, software metrics such as lines of code (LOC), depth of inheritance tree (DIT), and cyclomatic complexity (CC) are measured, and some numerical values are obtained. The concepts of measurement part provide a solid base to perform the evaluation.

The terms under "evaluation" are used to judge the numeric value obtained as a result of the measurement according to the needs of an evaluator. In another saying, these concepts are used to address whether these measurement results are satisfactory or not. Some example questions formed to interpret measurement results are given in Fig. 5.2. For

example, these concepts of evaluation part consist of terms for aggregating and interpreting measurement results. The interpretation determines whether the quality of the OSS product is at the desired level and whether it fits the needs of the evaluators.

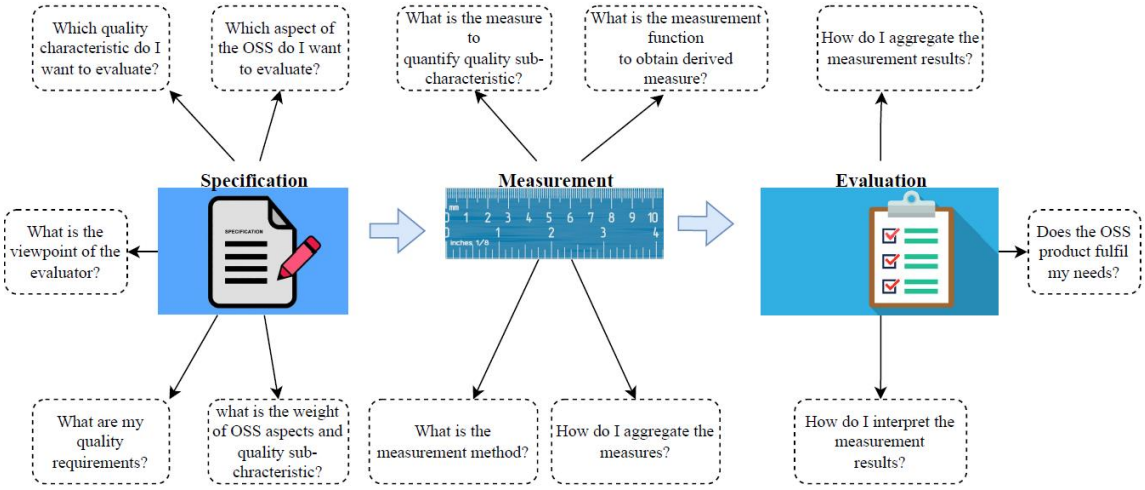


Figure 5.2. Relationship between specification, measurement, and evaluation

Among the concepts mentioned above, "measurement" and "evaluation" are often confused. Therefore, the difference between these two concepts is briefly discussed here. The main focus of "measurement" is being able to quantify something, such as performance or skills. However, the main focus of "evaluation" is to determine success or failure using data or information from "measurement". Therefore, while "measurement" consists of observation expressed numerically, observations in "evaluation" can be both quantitative and qualitative. Also, the content-oriented action is performed in "measurement", whereas the objective-oriented action is performed in "evaluation". Furthermore, "measurement" has a limited scope and requires less energy and time than "evaluation".

Apart from these, the determined concepts have been distributed into a 5-level hierarchical structure, as shown in Table 4.6. The results of our second SLR study [30] have indicated that the important OSS quality models (e.g., [44][144]) in literature have a hierarchical structure. This hierarchical structure is decomposed from abstract terms to more concrete terms, forming a 5-level structure: viewpoint, high level-characteristic, characteristic, sub-characteristic, and metric. Detailed information about these levels is given in Section 4.2.3. It is important to employ this hierarchical structure since quality models are instances of the meta-models.

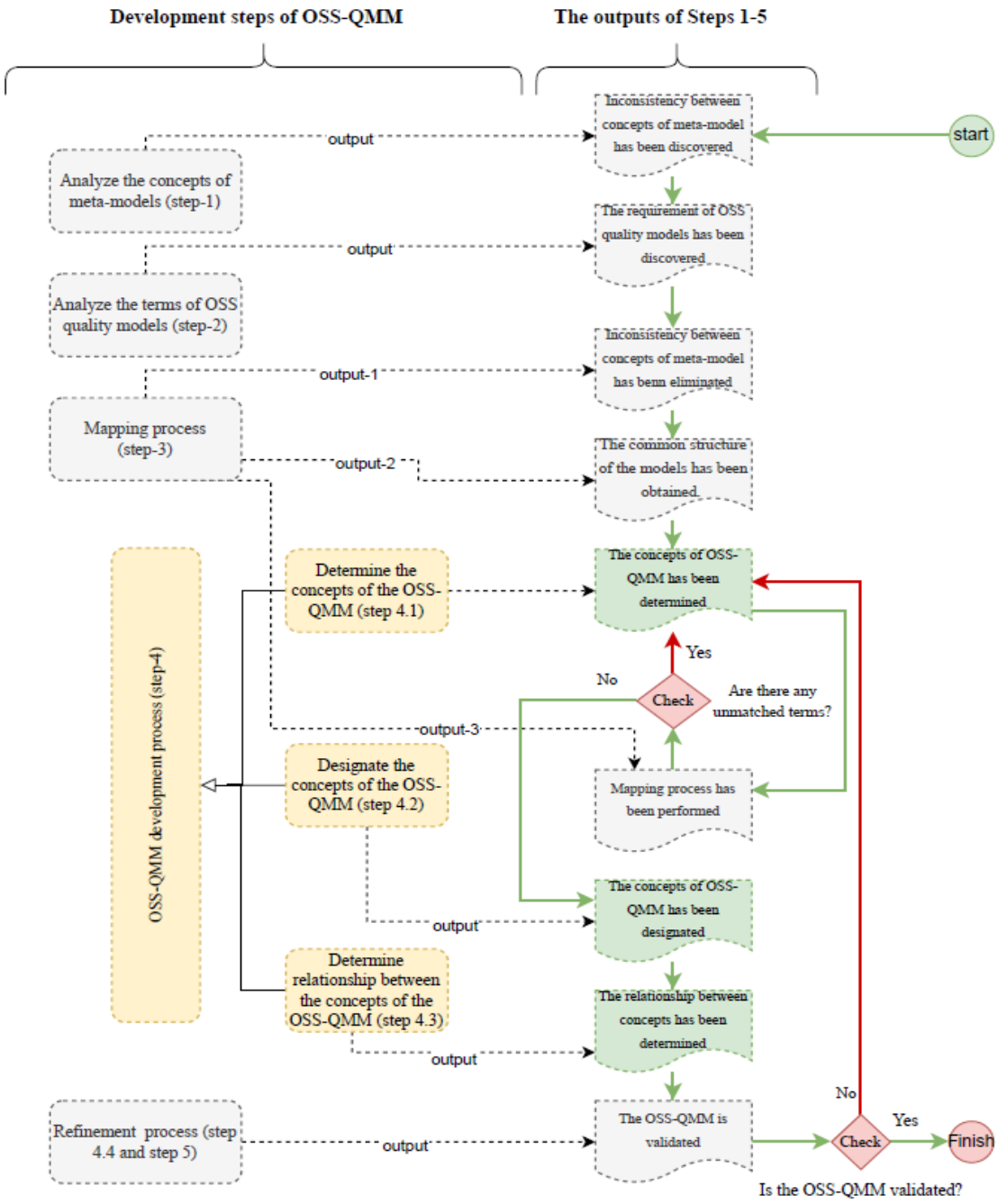


Figure. 5.3. The process of sub-steps 4.1, 4.2 and 4.3 (indicated in green color) and their relationship with other steps

5.1.1.3. Relationships of the Concepts

In the third sub-step of the meta-model creation (Step 4), it is necessary to determine the relationship between concepts after determining and designating them. As can be seen in Fig. 5.3, a laborious process has been followed until the concepts of OSS-QMM have been determined and designated. After the concepts have been determined, a mapping

process has been performed. If there is no problem in the mapping process, the determined concepts are designated, and then their relationship is determined. The important software quality meta-models [34][144][193] from our first SLR study [37] have been referenced to determine the relationship between the concepts. Furthermore, the meanings of concepts and their intended use in our OSS-QMM have been taken into account. Apart from them, the UML experience of the authors of this thesis has been utilized. The developed OSS-QMM has been then validated using multiple validation methods. As shown in Fig. 5.3, the determined concepts, their designation, and their relationship are revised in case of problems during the validation process. As the validation process involves expert opinion, the relationships are also reviewed by subject matter experts. Details on expert opinion will be explained in Section 6. After all, in Step 4.3, the final relationships between the concepts of the OSS-QMM have been determined after the review-and-revise process described above.

In this context, a total of four types of relationships, namely association, composition, aggregation, and generalization, have been used to link the concepts in the OSS-QMM. A graphical representation of these relationships is shown in Fig. 5.4. Also, the following paragraphs provide explanations of these relationships.

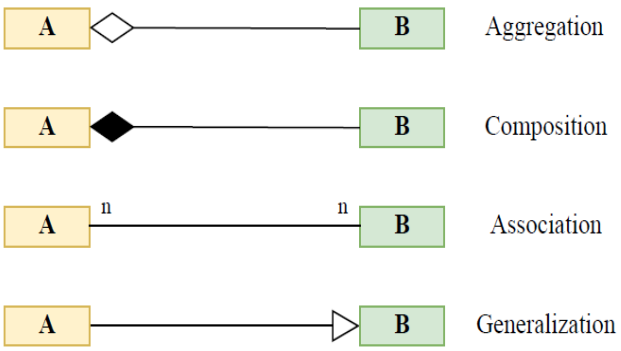


Figure 5.4. Types of relationships used in OSS-QMM

An ***association*** **relationship** is a structural relationship that links different concepts in OSS-QMM. In this relationship, a binary relationship is exhibited between concepts of OSS-QMM that represent an activity. Associations are characterized by a line between two concepts in a system to be developed. In Fig. 5.5 (a), the type of association relationship is represented. There are three types of association relationships: unidirectional, bidirectional and self/reflexive association. The navigation direction of the arrow specified its type. If the arrows are on both sides or no arrows, the association is known as a bidirectional association; if the arrow is on one side, the association is known

as a unidirectional association; and if a single concept is associated with itself, the association is known as a self/reflexive association. The example usage of association relationship is represented in Fig. 5.5 (b) for better understanding. As specified in the figure, the multiplicity of the relationship can be indicated by adding numbers (i.e., 1 / 0..1 / 1..* / *, etc.) to the entry and exit point of the line. For example, a student can associate with one or more instructors, as shown in the top example in Fig. 5.5 (b), or an instructor has one or more students in the middle example in Fig. 5.5 (b). Also, the behavior of a concept can be indicated by using some names. For example, one or more students can learn from one or more instructors, or one or more instructors can teach one or more students, as shown in the bottom example in Fig. 5.5 (b).

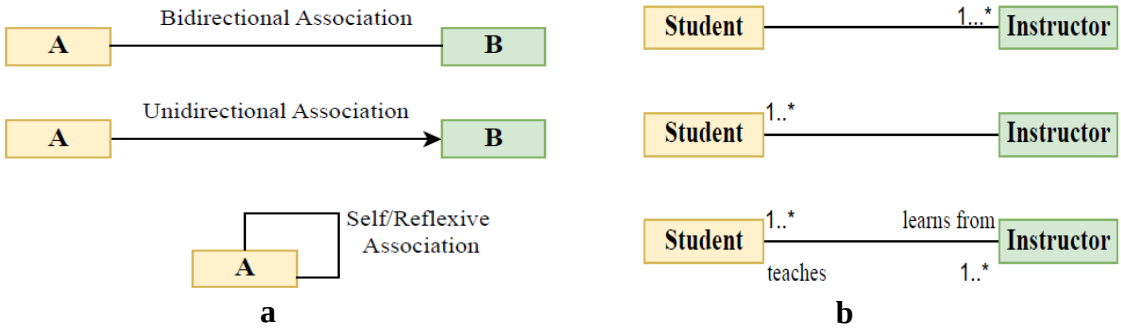
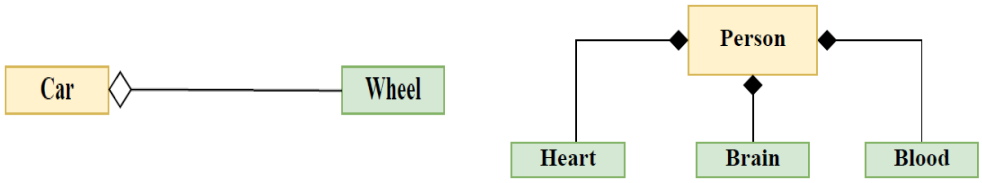


Figure 5.5. **(a)** Types of association relationships, and **(b)** an example usage of association relationship

The ***aggregation relationship*** is used for a more specific purpose compared to the association relationship. This relationship is considered as a part of the association relationship. That is, there is "has-a" relationship between objects. An empty diamond at one end of a straight line is used to symbolize this relationship. An example of an aggregation relationship is shown in Fig. 5.6 (a). As shown in Fig. 5.7, it is considered as a subset of the association relationship. In this type of relationship, the life cycles of objects (i.e., child and parent) are separate from each other. That is, the child object is independent of its parent in this relationship, as shown in Fig. 5.6 (a). In the example, a wheel is necessary for a car to move. However, the wheel may be used independently with any type of vehicle, including a bicycle, truck, or scooter. It is represented that there is an aggregation relationship since the wheel (child object) can exist independently of the car (parent object).



a **b**

Figure 5.6. **(a)** An example of an aggregation relationship, **(b)** an example of a composition relationship

In the ***composition relationship***, the life cycles of child and parent objects are interconnected. That is, the child object does not exist without its parent, which means that there is a strong relationship between objects. Therefore, there is "is-part-of" relationship between objects (i.e., child and parent). An example of an aggregation relationship is shown in Fig. 5.6 (b). As shown in Fig. 5.7, it is considered as a subset of the association relationship. A black diamond at one end of a straight line is used to symbolize this relationship, as shown in Fig. 5.6 (b). In the example (Fig. 5.6 (b)), there is a composition relationship between parent object (i.e., person) and child objects (i.e., brain, heart, and blood). This is because the brain, heart, and blood will all be wasted if the person is destroyed. In another saying, there is a composition relationship since the blood, heart and brain (child object) cannot exist independently of the person (parent object).

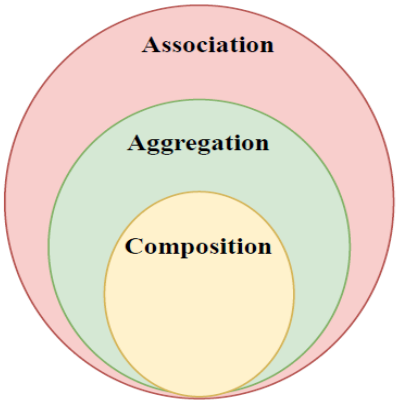


Figure 5.7. The Venn diagram of the relationships between classes

The ***generalization relationship*** is a directed relationship between two classifiers, namely superclass and subclass. The superclass (i.e., base class or parent) is a more general classifier, and the subclass (i.e., derived class or child) is a more specific classifier. This relationship put into practice the OO (i.e., Object-Oriented) concept called inheritance. The functionality superclass is inherited by the subclass, and the subclass can access and

update the superclass. Also, the subclass can add its functionality to itself in addition to the functionality of the superclass. Therefore, there is "is-a" relationship between the superclass and subclass. As shown in Fig. 5.4, this relationship is represented by a line from subclass to superclass with a hollow triangle that points to the superclass. An example of a generalization relationship is given in Fig. 5.8. In this example, there can be two types of bank accounts as saving and credit card accounts. As you can see, a superclass can have many subclasses, and also the subclass can have one or more superclass. These subclasses (i.e., saving account and credit card account) inherit some generalized functionality from the superclass (i.e., bank account), e.g., account balance and account number. Also, the subclass can add its functionality to itself, e.g., card verification value (CVV) number or expiry date.

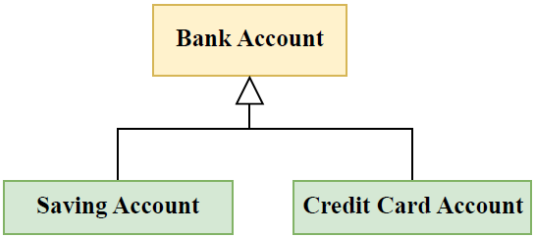


Figure 5.8. An example of a generalization relationship

The use of relationships detailed above between the concepts in the OSS-QMM is given in Table 5.1. The first column of the table shows the first concept of the relationship, the third column shows the second concept of the relationship, and the second column shows the type of relationship between these two concepts. The last column of the table shows the layer with the first and second concepts, respectively.

Table 5.1. List of the relationship between concepts of OSS-QMM

Concept 1	Relationship type	Concept 2	Layers of concept 1/concept 2 (group)
Q. model	Association	Viewpoint	Specification/Specification
Q. model	Aggregation	Q. characteristic	Specification/Specification
Q. model	Aggregation	Measurable concept	Specification/Specification
Q. requirement	Association	Viewpoint	Specification/Specification
Q. requirement	Association	Impact	Specification/Specification
Viewpoint	Association	Information need	Specification/Specification
Viewpoint	Association	Weighting	Specification/Specification
Information need	Association	Measurable concept	Specification/Specification
Information need	Association	Q. characteristic	Specification/Specification
Q. characteristic	Association	Entity	Specification/Specification
Q. characteristic	Association	Impact	Specification/Specification
Q. characteristic	Association	Evaluation results	Specification/Evaluation
Impact	Association	Measurable concept	Specification/Specification
Impact	Association	Evaluation	Specification/Evaluation

Measurable concept	Association	Entity	Specification/Specification
Measurable concept	Association	OSS aspect	Specification/Specification
Measurable concept	Association	Measure	Specification/Measurement
Entity	Association	Measure	Specification/Measurement
Weighting	Association	Sub-characteristic	Specification/Specification
Weighting	Composition	W. aggregation method	Specification/Specification
Weighting	Composition	Weighting method	Specification/Specification
Weighting	Association	OSS aspect	Specification/Specification
OSS aspect	Generalization	Code-based	Specification/Specification
OSS aspect	Generalization	Community-based	Specification/Specification
Measure	Aggregation	Normalize measure	Measurement/Measurement
Measure	Association	Scale	Measurement/Measurement
Measure	Association	Unit	Measurement/Measurement
Measure	Association	Measurement	Measurement/Measurement
Measure	Association	Measurement method	Measurement/Measurement
Measure	Association	Aggregated measure	Measurement/Measurement
Measure	Generalization	Base measure	Measurement/Measurement
Measure	Generalization	Derive measure	Measurement/Measurement
Base measure	Association	Measurement function	Measurement/Measurement
Derive measure	Association	Measurement function	Measurement/Measurement
Measurement method	Generalization	Manually	Measurement/Measurement
Measurement method	Generalization	Automatically	Measurement/Measurement
Aggregated measure	Composition	M. aggregation method	Measurement/Measurement
Aggregated measure	Association	Measurement result	Measurement/Measurement
Measurement	Association	Measurement method	Measurement/Measurement
Measurement	Association	Measurement result	Measurement/Measurement
Measurement result	Association	Evaluation	Measurement/Evaluation
Evaluation	Generalization	Evaluation function	Evaluation/Evaluation
Evaluation	Generalization	Manual evaluation	Evaluation/Evaluation
Evaluation	Association	Evaluation aggregation	Evaluation/Evaluation
Evaluation	Association	Evaluation result	Evaluation/Evaluation
Evaluation	Association	weighting	Evaluation/Specification
Evaluation aggregation	Generalization	E. aggregation method	Evaluation/Evaluation
Evaluation aggregation	Generalization	E. aggregation function	Evaluation/Evaluation

5.2. Refinement Process of the OSS-QMM

A literature-based approach has been followed in Steps 1 and 2, as shown in Fig. 1.2. Then, during the mapping process (Step 3), both expert opinions and literature have been utilized. Then, the Steps 4.1, 4.2, and 4.3 have been employed as explained in Section 5.1.1, and accordingly, the initial version of the OSS-QMM has been proposed. Afterwards, the steps (4.4 and 5) related to the refinement of the initial version of the OSS-QMM have been employed. In Step 4.4, a review-and-revise process has been performed by subject domain experts on the initial version of the OSS-QMM. In Step 5, multi-faceted empirical research has been employed to validate the OSS-QMM in practice. Details of the validation process are given in Section 6. In this context, in Section 5.2.1, the refinement process performed according to the output obtained from the review of the subject matter experts is explained. In Section 5.2.2, the refinement process carried

out according to the outputs obtained from the validation process is mentioned. Throughout these sub-sections, the refinement to the OSS-QMM is explained for each version of the OSS-QMM. All versions of the OSS-QMM are given in Appendix-2 and the initial version of the OSS-QMM (v1) is given in Appendix-2 (a). Also, all versions of the OSS-QMM and the refinement performed in each version are given in Table 5.2.

Table 5.2. The versions of OSS-QMM with refinement performed and related reference

Version	Refinements	Reference
Initial version (v1)	-	Appendix-2 (a)
Second version (v2)	The concept of *OSS aspect* and *quality requirement* have been added.	Appendix-2 (b)
Third version (v3)	The concepts of *weighting*, *weighting method*, and *evaluation aggregation* have been added.	Appendix-2 (c)
Fourth version (v4)	The concepts of *weighting aggregation method*, *aggregated measure*, and *measure aggregation method* have been added.	Appendix-2 (d)
Final version (v5)	The names of some relationships and the numbers of multiplicities in some relationships have been changed.	Fig. 5.10

5.2.1. Refinement with Subject Matter Experts (Step 4.4)

In this step, the development process, structure and content of the OSS-QMM have been reviewed by experts on software quality models, using the suggestions by Tanrıover et al. [194] and Kläs et al. [195]. That is, it has been aimed to examine the OSS-QMM by external parties other than the authors of the thesis. In this context, a total of four subject matter experts have been determined, two being from industry and two from academia. In determining the experts, a prerequisite has been applied that an expert would have seven years or more experience in the field of software quality and its modeling. The first and the second experts with industry backgrounds have had 8 and 10 years of software quality modeling experience, respectively. The other two experts with academic backgrounds have been researchers lecturing and consulting on information systems and software engineering for more than 11 years. As shown in Fig. 5.3, Steps 4.1, 4.2, and 4.3, respectively, are revised in case of problems during the review of these subject matter experts. The following questions have been prepared in order to obtain feedback from these experts;

- **Q1-** Do you think that the OSS-QMM development process is well-founded?
- **Q2-** Do you think the appreciated concepts are determined in the OSS-QMM? (Do you think that any unnecessary concept is used in our OSS-QMM)
- **Q3-** Do you think that the determined concepts are constructed appropriately? (designation – 5-level categorization)
- **Q4-** Do you agree that the relationship between concepts is compatible and appropriate?

As you can see, these questions have been aimed at obtaining feedback about the development process, structure and content of the proposed OSS-QMM. In Q1, experts have been asked about the quality of the OSS-QMM development process. It has been asked whether there has been anything missing in the development process. In this context, prior to asking the question, the development process of the OSS-QMM has been presented to the experts from Step-1 to Step-4. In Q2, considering the meaning and intended use of the concepts used in the OSS-QMM, it has been asked whether appropriate concepts have been determined. In other words, it has been asked whether there has been an overused or unused concept in the OSS-QMM. In deciding this, experts have been advised to take a well-designed OSS quality model they have been familiar with as a basis and proceed by mapping concepts. In Q3, the experts have been asked whether the identified concepts have been appropriately designed in our OSS-QMM. As it is known, a 5-level hierarchy and a 3-layer structure have been used in our OSS-QMM. In this context, it has been asked whether these structures are appropriate based on OSS quality models. In answering this question, experts have been advised to proceed based on the structures of well-designed OSS quality models they have been familiar with. In Q4, the experts have been asked whether the relationships between the determined concepts have been appropriate or compatible. In this context, experts reviewed the relationships, taking into account the meaning and intended use of the concepts in our OSS-QMM.

After the questions have been prepared and the experts have been determined, a series of online meetings have been held to discuss and gather answers for each question with the experts. Each expert has been interviewed individually. In these meetings, the development process has been presented in order to provide preliminary information to the experts (i.e., Steps 1-3). Also, the OSS-QMM itself, together with the concepts, their

meanings and intended use, and the relationships between these concepts, have been presented. As shown in Fig. 5.3, this step 4.4 has progressed by the iterations of the review-and-revise process with respect to the suggestions of the experts about the OSS-QMM. That is, the OSS-QMM and its development process have been reviewed by the expert for each question.

In this context, the consensus of the experts has been that the concept of *OSS aspect* was missing in the initial version (v1) of the OSS-QMM, although the concepts *code-based* and *community-based* were included, as shown in Appendix-2 (b). That is, the experts have given feedback that *code-based* and *community-based* should be child concepts and *OSS aspect* should be a parent concept above them. Experts have also stated that there should be a *quality requirement* concept between the concept of *impact* and the concept of *information need*. Because quality requirements should be known to determine the *impact* of *measurable concept* on *quality characteristic*.

Then, the OSS-QMM has been revised in line with the suggestions obtained from the experts. In this context, the review-and-revise process has continued until the experts have agreed that; the development process of the OSS-QMM is well-founded, the content of the OSS-QMM is sufficient to apply in practice, the structure and generality of the OSS-QMM is complete, the concepts and relationship between concepts are compatible. After these refinements, the second version of the OSS-QMM (v2) has been obtained. This version of the OSS-QMM is given in Appendix-2 (b). Thus, the OSS-QMM has matured and taken its new form with the feedback obtained for each question before the validation process.

5.2.2. Refinement with the Validation Process

In this section, the refinement process of the OSS-QMM developed in this thesis during the validation process is explained. As described in Section 5.2.1, a refinement process has been carried out with subject matter experts on the initial version of the OSS-QMM. After this, based on the validation of the OSS-QMM, a review-and-revise process has been applied to improve the OSS-QMM. The lifecycle of improving the OSS-QMM by maturing it through the validation process is shown in Fig. 5.9.

The initial version of the OSS-QMM has been proposed based on the previous steps (i.e., Steps 1-3). Then, initial validation of the OSS-QMM has been provided over an example evaluation, as shown in Fig. 5.9. In other words, the application of the initial OSS-QMM

has been demonstrated in the unreal OSS products with dummy evaluation data through the designed toy experiment. Detail of its implementation is given in our latest study [54]. In this evaluation, an example has been demonstrated to identify the OSS product that best met the evaluator's needs among the alternatives. Quality evaluation scores have been calculated for each alternative OSS product since one of the best ways to understand the quality of a product has been to compare it with those of possible alternatives. All these efforts have been targeted to demonstrate the application of the OSS-QMM, allowing it to be visualized and better understood. Thus, the applicability of the OSS-QMM has been monitored through an example implementation. As shown in Fig. 5.9, if there is a problem with the implementation of the initial validation, the OSS-QMM is reviewed and then revised. The review-and-revise process has continued until there is no problem with the implementation of the OSS-QMM.

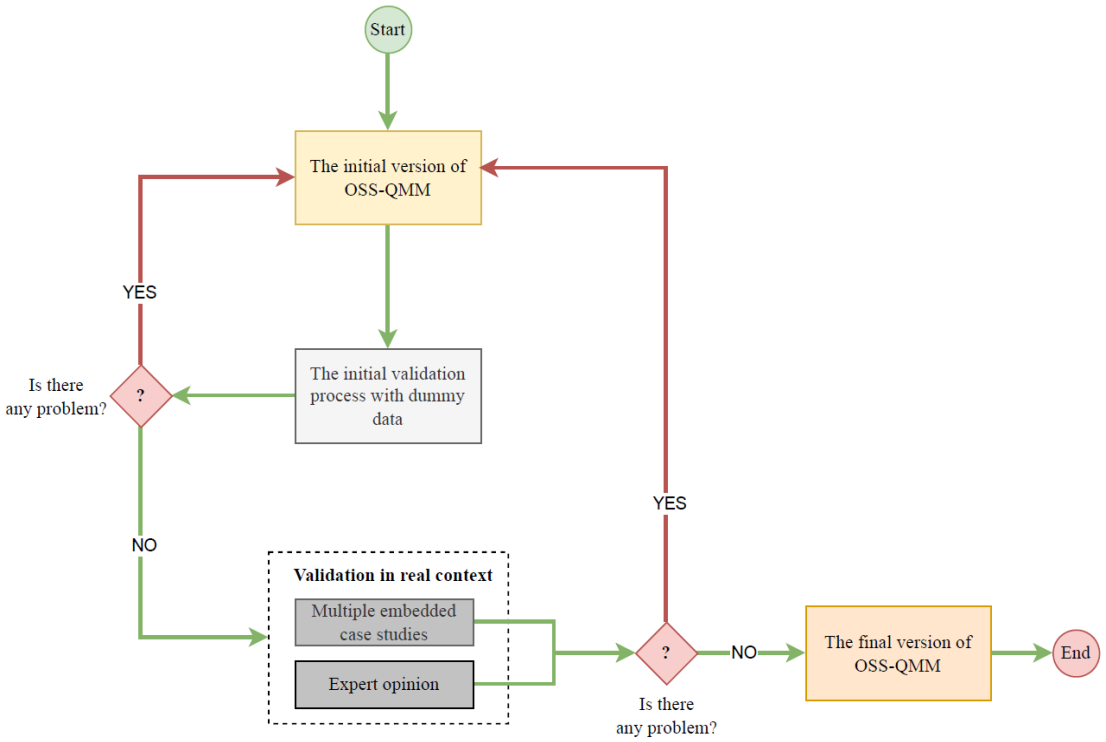


Figure 5.9. The refinement process of the OSS-QMM during the validation

In this context, after this initial validation, some refinements have been performed to the OSS-QMM as given in Table 5.2. That is, the concept of *weighting* and *weighting method* have been added to the OSS-QMM. During the initial validation process, we have realized that the importance of the OSS aspects and sub-attributes may be different with respect to the viewpoint of stakeholders. For example, the quality of code-based aspect may be more important according to a developer. It is also true for OSS aspects. Therefore, the

stakeholder (i.e., viewpoint) should assign the weight to them by using a *weighting method*. Also, we have realized that in the evaluation phase, three inputs (i.e., measurement result, impact, and weight of sub-characteristic) are required to interpret the measurement results. Therefore, we have added the concept of *evaluation aggregation* to the OSS-QMM. After these refinements, the third version (v3) of the OSS-QMM has been obtained and this version is given in Appendix-2 (c).

As shown in Fig. 5.9, if there is no problem in the implementation of the initial validation, validation of the OSS-QMM in a real-world setting has been performed. In this context, a multi-faced validation process has been followed to ensure that a suitable OSS-QMM has been developed to satisfy or fit the intended use in this study. That is, a validation process has been carried out to verify the applicability of the proposed OSS-QMM. In this context, two validation methods have been used; multiple-embedded case studies and expert opinion. In the multiple-embedded case studies, three case studies have been implemented in practice to demonstrate the applicability of our OSS-QMM. In this context, real evaluation data and OSS products have been used for quality evaluation. As shown in Fig. 5.9, if there has been a problem with the implementation of the case studies, the OSS-QMM has been reviewed and then revised. The review-and-revise process has continued until there is no problem with the implementation of the OSS-QMM in real setting.

In this context, after performing the case studies, some refinements have been performed to the OSS-QMM as given in Table 5.2. That is, the concepts of *weighting aggregation method*, *aggregated measure*, and *measure aggregation method* have been added to the OSS-QMM. During the case studies, we have realized that the concepts of *weighting aggregation method* needs to be used in order to obtain the final weights of sub-characteristics on OSS aspects, after assigning weights to OSS aspects and sub-characteristics. Because the weight of each sub-characteristic in each OSS aspect may be different for diverse viewpoints (see Section 5.3). Also, we have realized that a *measurable concept* can be associated with more than one *measure*. Therefore, these measures should be aggregated with the concept of *aggregated measure*. To perform aggregation, a *measure aggregation method* should be employed. Then, we have added 'satisfy' relationship between the concept of *measurable concept* and *information need* since determined measurable concept should satisfy the information need. After these

refinements, the fourth version (v4) of the OSS-QMM has been obtained and this version is given in Appendix-2 (d).

In the expert opinion part, validation has been performed by experts for the real-world application of the OSS-QMM. Details of all these validation processes are given in Section 6. Throughout this validation process in a real context, the practical application, structure, content, and levels of abstraction of the OSS-QMM have been observed. As shown in Fig. 5.9, if there is a problem with the implementation of the validation in the real setting, the OSS-QMM is reviewed and then revised as necessary. The review-and-revise process has continued until there are no problems in the implementation of the OSS-QMM and it has reached sufficient maturity, as implied in Fig. 5.9.

After receiving expert opinions, some refinements have been performed to the OSS-QMM as given in Table 5.2. That is, the names of some relationship and the numbers of multiplicities in some relationship have been changed. In this context, the relationship between the concept of *impact* and *quality requirement* has been changed as "elicited". Also the multiplicity of the relationship between *weighting* and *OSS aspect*, between *evaluation results* and *quality characteristic*, and between *quality model* and *quality characteristic* have been modified. As a result, the OSS-QMM has been refined with iterations again, and the fifth and the final version (v5) of the OSS-QMM has been obtained, as shown in the Fig. 5.10. That is, this step has contributed to both the validation and further refinement of the OSS-QMM.

5.3. The Proposed OSS-QMM

The final version of the OSS-QMM, which is shown in Fig. 5.10, has been proposed by following the ***step-based process for the meta-model creation*** process, as summarized in Section 1.3. In line with this process, a systematic process has been followed to propose the OSS-QMM. That is, the outputs of systematic development steps (i.e., Steps 1-3 shown in Fig. 1.2) have been considered in OSS-QMM development. In this context, first of all, the concepts of our OSS-QMM have been determined by considering the most used concepts of the quality meta-models proposed for OSS and the custom type of software obtained from our first SLR study [37] (Step-1). Then, the structure of our OSS-QMM is shaped by considering the common structure of OSS quality models obtained from the second SLR study [30] (Step-2). Then, the level-based matching process, details of which are explained in Section 5.3, has been carried out between the terms of the OSS quality

models and the terms of the meta-models (Step-3). Then, the initial proposal for a meta-model of OSS quality emerged as a result of this matching process. Then, the refinement process of OSS-QMM has been started, as detailed in Section 5.2.

In the refinement process, first of all, the review-and-revise process (i.e., iterative process) has been performed by four subject matter experts to review the content and structure of OSS-QMM, i.e., determined concept, their designation, and their relationships. Then, the initial validation process is implemented by using dummy data, as represented in Fig. 5.9. Next, the validation process is employed by using real evaluation data and OSS products in a real-world setting. All these validation processes are aimed to show the practical applicability of the OSS-QMM. During these validation processes, the review-and-revise (i.e., iterative process) process has been employed, and the final version of OSS-QMM has been developed, as represented in Fig. 5.10. That is, the OSS-QMM has been proposed by considering the concepts free of inconsistencies, the common 5-level structure of quality models, the mapping process, the review of subject matter experts, and the outputs of the validation process in an unreal and real-world setting.

According to the levels in the hierarchy of quality models, the concepts of the OSS-QMM have been demonstrated using color-coding as mapped in Table 4.6. Besides, in Fig. 5.10, the categories (or stages, i.e., specification, measurement, and evaluation) of the concepts in the OSS-QMM are shown. The definition of the concepts used in the OSS-QMM is given in [181] and will not be repeated here. Instead, we will explain the purpose of using and relationships of these concepts in the OSS-QMM supported by examples, according to their categories, in the following sub-sections. Furthermore, an example of an operationalized quality model for OSS, which is instantiated from the OSS-QMM, is given in Appendix-3. This operationalized quality model may help the reader to trace and make sense of the concepts in the OSS-QMM.

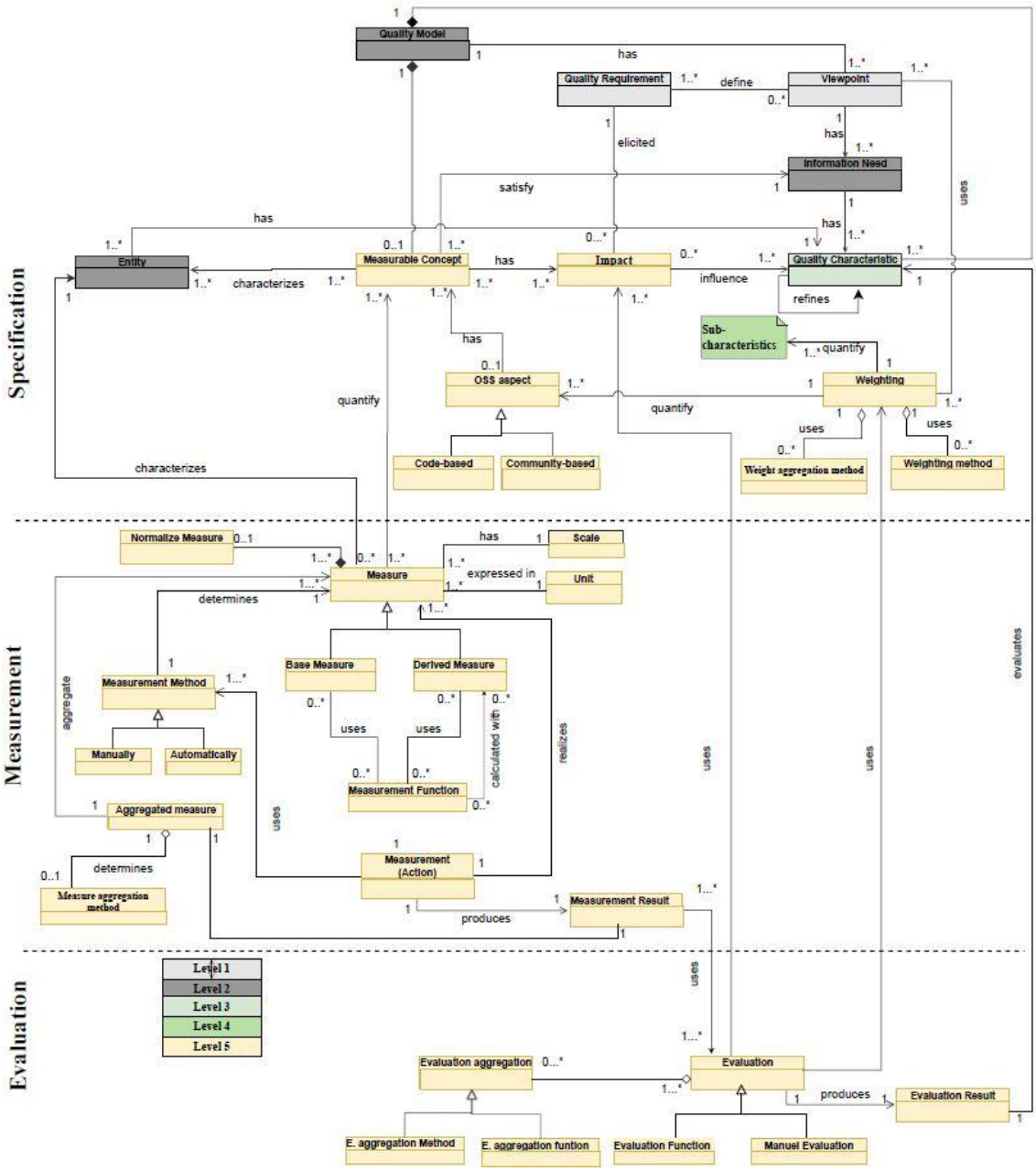


Figure 5.10. The OSS-QMM

5.3.1. Concepts of OSS-QMM in the Specification Category

In the specification category, concepts are used to establish the aspects, scope, and objectives of the OSS measurement process, as shown in Fig. 5.10. Detailed information about concepts belonging to this category is given in Section 4.1. The purpose of using and relationships of concepts in the OSS-QMM is explained in the following paragraph. The concept of OSS-QMM is shown in italics to increase traceability.

A *quality model* is a set of quality characteristics, entities, and measurable concepts characterized by measures. Also, it is an instance of the proposed OSS-QMM. The *viewpoint* is the starting point of an OSS quality model. The content of the OSS quality model should be shaped with respect to a specific point of view, as stakeholders can perceive OSS quality from different viewpoints, such as developer, customer, designer, etc. For instance, an OSS product as subject to quality evaluation may be required as component(s) of a larger project, and thus, the OSS product needs to be interoperable; or this product may be required for constant daily use by the adopter either in its original form or with modifications [5]. The needs of stakeholders can be different from the measurement process. Therefore, stakeholders can have their *information needs* as the insights necessary to manage goals, objectives, problems, and risks. The request for the calculation of defect density to evaluate maintainability is an example of an information need. An information need is related to ***the*** *quality characteristic* that is generally too abstract to measure directly. Quality characteristics are used to control quality and to determine how well the software product or system satisfies the needs of its stakeholders. Maintainability, usability, and reliability are examples of quality characteristics. In accordance with the structure of the quality models, quality characteristics decompose into *sub-characteristics* in the OSS-QMM. Quality sub-characteristics are used to control more specific aspects of quality. However, despite being considered a less abstract form of quality characteristics, they are still abstract to measure directly. For example, stability, testability, and analyzability are sub-characteristics of maintainability as a quality characteristic. Thus, a sub-characteristic is associated with *measurable concepts* which enable concrete measurements by directly relating to one or more measures. The measurable concept composes a property of the OSS product that is related to the quality of the product. They are always defined in such a way that it is possible to talk about the degree to which it is present in the product.

In addition to source code, diverse types of evaluation data belonging to the community-based aspect that is specific to OSS are stored in various databases. Therefore, measurable concepts consist of two *OSS aspects* that enable the complete measurement of OSS products. OSS aspect can be derived from *code-based* (e.g., comment frequency of source code) and *community-based* (e.g., the activeness of the contributors) aspects. In this regard, the concept of *entity* that is a part of the software product and considered during the measurement process should be defined. For example, if the measurable concept

belongs to the code-based aspect, the concept of entity can be "source code", or if the measurable concept belongs to the code-based aspect, the concept of entity can be "contributor". Thus, a measurable concept is usually defined together with an entity as follows: "name" of "entity", e.g., the activity of the developer and complexity of source code. A significant situation exists in the relationship between sub-characteristics and measurable concepts. Different measurable concepts can have a positive or negative effect on a sub-characteristic related to the quality of OSS. This relationship is established with the concept of *impact*, which has an effect on evaluation results. If there is a direct proportion between a measurable concept and a sub-characteristic, it is considered a positive impact. If there is an inverse proportion between them, it is considered a negative impact. For instance, higher complexity of source code (i.e., "measurable concept" belonging to code-based aspect) is undesirable for maintainability and then is considered as the negative impact. Conversely, the higher bug-solving success of the contributor (i.e., measurable concept belonging to community-based aspect) is desirable for maintainability and then is considered as the positive impact. To determine the value of impact (i.e., positive or negative), data is collected with the concept of *quality requirement* by considering the viewpoint of the stakeholder. In addition, since different viewpoints may perceive the OSS quality differently, the importance of the OSS aspects and sub-attributes may be different with respect to the viewpoint of stakeholders. Therefore, the concept of *weighting* is employed to assign weights for OSS aspects and sub-attributes by using a *weighting method* (e.g., AHP). For instance, it is essential from the viewpoint of the developer that the OSS product is easy to shape according to the needs. Therefore, the quality of the code-based aspect may be more important than the quality of the community-based aspect from this viewpoint. Also, considering sub-attributes of maintainability, modifiability (i.e., easy to modify) may be more critical than others for this viewpoint.

5.3.2. Concepts of OSS-QMM in the Measurement Category

In the measurement category, terms are used to quantify an OSS product's quality and provide a solid base to perform the evaluation. In other words, in the measurement part, the numeric values are objectively given to the defined properties of OSS in the specification part. These concepts can be followed from the OSS-QMM shown in Fig. 5.10. They are shown in italics to increase traceability in the following paragraphs.

In this category, after determining the measurable concept belonging to each OSS aspect, they should be quantified by using the concept of *measures* to obtain concrete values. The concept of measures can be derived from a concept of *base measure* or *derived measure*. For example, a base measure can be "m1: line of documented source code" or "m2: effort spent". Also, a derived measure is derived from a set of base measures using a *measurement function*. For example, a derived measure can be obtained using the measurement function (i.e., m3: m1/m2) with the two base measures given above. In this regard, a measure must be expressed in a *unit* (e.g., defects, days, lines) and have a *scale* (e.g., integers from zero to infinite). After measures are determined to quantify measurable concepts belonging to OSS aspects, the numeric values should be assigned for determined measures. In this context, the concept of *measurement method* is employed to determine how to quantify a determined measure. The measurement method can be performed in two ways, *manually* and *automatically*. For example, if the quantification of the code-based measure (e.g., depth of inheritance tree) is performed by a static or dynamic code analyzer tool (i.e., instrument), it means that the numeric value is automatically assigned to a measure. If the quantification of the community-based measure (e.g., number of contributors) is performed by counting from the website of OSS repositories (e.g., GitHub, Sourceforge), it means that the numeric value is manually assigned to a measure.

The quality of OSS is affected by many measurement data that are accessible from both code-based and community-based aspects. In addition, these data are scattered in a variety of databases and, accordingly, are heterogeneous. Regarding this, each measure must be normalized by using the concept of *normalized measure* to move the measured value to the same range before performing measurement action. Thus, the value of the measures can be compared with other measures and converted into meaningful forms. Measurable concepts can be associated with more than one measure, that is, a set of measures. Therefore, after normalizing each measure, it is necessary to aggregate the set of measures associated with a measurable concept. In this regard, the concept of *aggregated measure* is employed to aggregate a set of measures. To perform aggregation, a *measure aggregation method* should be used, such as calculating the weighted average of the measures. In this way, a one-to-one relationship is established between the measurable concept and the measure. After the infrastructure necessary for the measurement process is determined, the concept *measurement* is realized, and the measurement event is

performed as action. In the end, the *measurement result* is produced as an output of the measurement action for the measurable concept belonging to an OSS aspect. This process is repeated for measurable concepts belonging to each OSS aspect, and the measurement results are obtained before the evaluation process.

5.3.3. Concepts of OSS-QMM in the Evaluation Category

In the ***evaluation category,*** terms are used to interpret the measurement results, which provides a solid base to perform the evaluation. In other words, they are used to determine whether the quality of the OSS product is at the desired level and fits the needs of evaluators. These concepts can be followed from the OSS-QMM shown in Fig. 5.10. They are shown in italics to increase traceability in the following paragraphs.

In the evaluation process, data obtained from three concepts, namely measurement results, impact, and weights of sub-characteristics, are used as inputs. These inputs are used in the concepts of *evaluation* to produce an output allowing us to determine the quality of the OSS product. The concept of evaluation uses the concept of *evaluation aggregation* for aggregating these three inputs. The aggregation of the evaluation can be performed by using the concept of *evaluation aggregation method* (e.g., TOPSIS (see Section 6)) or *evaluation aggregation function* (e.g., average aggregation function). After these inputs are aggregated, the evaluation aggregation should be interpreted. The interpretation is performed using the concepts of *evaluation function* or *manual evaluation*. For example, an evaluation function such as the "linear utility function" can be used to interpret values in a specified range, or manual evaluation (e.g., by taking into account expert opinion) can be performed for interpretation. Then, after the inputs are aggregated and interpreted by the concepts of evaluation, the *evaluation result* is produced for an OSS product. For example, this evaluation result can be between 0 and MaxPoint (e.g., range between 0 and 1). In the end, this evaluation result is associated with the quality characteristic aimed to be evaluated at the beginning of the quality evaluation process. In this way, the quality of the OSS product concerning its specified quality characteristics is evaluated.

6. VALIDATION METHODS AND THEIR IMPLEMENTATION (STEP-5)

The validation process is one of the most crucial steps in the studies to ensure that the research is correct, clean, and valuable [196]. Therefore, a multi-faced validation process is followed to ensure that a suitable OSS-QMM is developed to satisfy or fit the intended use in this thesis. In other words, in this section, a validation process is decided and carried out to investigate the applicability of the proposed OSS-QMM. In this context, three Research Questions (RQs) were prepared, as seen in Table 6.1, in accordance with the stages of the developed OSS-QMM. The description and scope of each RQ are given in the second column of the table. Also, the methods used to validate each RQ are listed in the third column of the table. In this context, the validation of the proposal from both internal and external viewpoints are considered. The case studies are organized to evaluate validity from an internal (i.e. the authors') perspective, while expert opinions are taken to evaluate validity from an external perspective. Therefore, the RQs and the answers to them using these two methods address versatility in validating the OSS-QMM.

Table 6.1. List of RQs, description of RQs, and validation methods related to each RQ

RQs	Motivation (or Requirements)	Validation Methods and Purposes
RQ.1: Are the evaluation results of OSS quality models derived from the OSS-QMM comparable?	The evaluation results of the OSS quality models derived using the OSS-QMM should be comparable.	**1-Case study:** Demonstrate that the evaluation results of the derived OSS quality models are comparable. **2-Expert opinion** (Exploratory Study, Part-1 (Q1-3) (See the list of Questions (Qs) in [197])): The same OSS products used in the case studies are evaluated by the experts with their own OSS quality models to demonstrate that the results are consistent.
RQ.2: Is the OSS-QMM effective for deriving the OSS quality models?	The OSS-QMM should allow the derivation of new OSS quality models and should fit (or guide) the structure of the existing OSS quality models.	**1-Case study:** Demonstrate the effectiveness of the OSS-QMM in deriving the OSS quality models. **2-Expert opinion (**Exploratory Study, Part-1 (Q4-5)): Experts demonstrate the effectiveness of OSS-QMM by deriving their own OSS quality models.
RQ.3: Is the OSS-QMM applicable in practice?	The OSS quality models derived using OSS-QMM should be applied in practice. Also, experts should find it practical to use the OSS-QMM in deriving the OSS quality models.	**1-Case study:** Demonstrate the applicability of the OSS-QMs derived from the OSS-QMM in practice. **2-Expert opinion** (Exploratory Study, Part-2 (Q1-12)): Experts assess the OSS-QMM with respect to its practical applicability.

As already shown in Fig. 1, the validation process is performed iteratively by *review-and-revise* activities. This iterative process is also illustrated in Fig. 6.1, which details validation of the OSS-QMM in a real-world setting. In the figure, the beginning and end of the process are represented by circles, input (i.e., OSS-QMM) and output (i.e.,

validation of OSS-QMM) is represented by parallelograms, the validation activities (i.e., case studies and expert opinion) are represented by rectangles, the decision nodes (i.e., RQs) are represented by diamonds, and the parallel gateways are used to show the parallel flow of the process. The decision nodes in Fig. 6.1 correspond to the RQs defined in Table 6.1, and the answers to all three decision nodes should be "yes" (i.e., as required by parallel gateway) for the proposed OSS-QMM to be validated in the real-world setting.

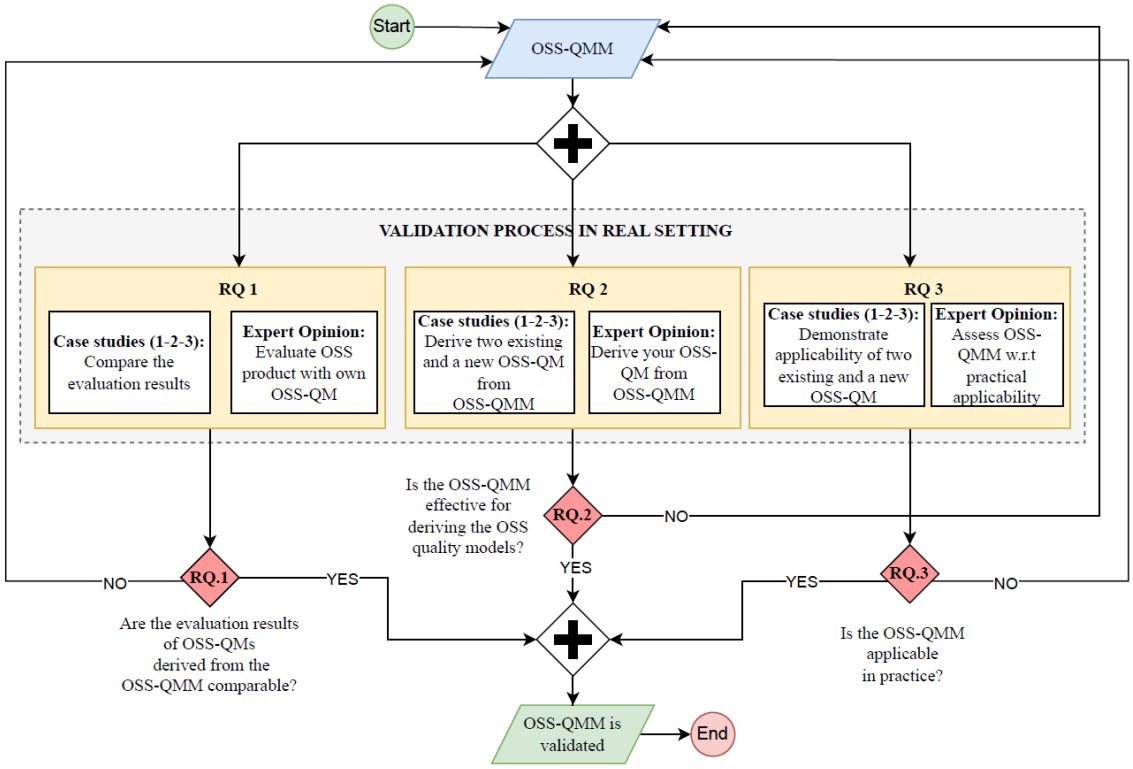


Figure 6.1. The validation process of OSS-QMM

For the first decision node on the side of the case studies (entitled RQ1), three cases are designed to demonstrate that the evaluation results are comparable. Then, on the side of the expert opinions, the activity of *evaluating OSS product with own OSS quality model* is carried out to answer RQ1. If the results of the three case studies and also the evaluation of the experts on the stated activity are comparable, the first decision node meets its requirements. Otherwise, the OSS-QMM needs to be reviewed and revised, and the answer to RQ1 is further investigated.

For the second decision node on the side of the case studies (entitled RQ2), three cases are designed to derive OSS quality models from the OSS-QMM. Then, on the side of the expert opinions, experts are expected to *derive an existing OSS quality model from the OSS-QMM*. This could be an OSS quality model they are familiar with from the literature

or the OSS quality model they use in their own companies. The most important criteria in this derivation process are that there should be no unmatched elements between their selected quality models and the OSS-QMM. If the vast majority of experts are able to successfully derive their quality models, the second decision node meets its requirements. Otherwise, the OSS-QMM needs to be reviewed and revised, and the answer to RQ2 is further investigated.

For the third decision node on the side of the case studies (entitled RQ3), three cases are designated to check the applicability of the OSS-QMM by applying the derived OSS quality models in practice. Then, on the side of the expert opinions, experts are expected to *assess the practical applicability of OSS-QMM* considering the validation activities carried out so far. If the vast majority of experts agree on the usability of OSS-QMM in practice, the third decision node meets its requirements. Otherwise, the OSS-QMM needs to be reviewed and revised, and the answer to RQ3 is further investigated.

As a result, if all three decision nodes fulfill their requirements, the OSS-QMM is considered as validated. In this regard, the validation methods listed in Table 6.1 are explained, and the applications of these methods are elaborated in the following sub-sections. Accordingly, the remainder of this section is organized as follow: In sub-section 6.1, firstly, the usage of the OSS-QMM for deriving the OSS quality models in the three case studies is demonstrated. Then, in Sub-section 6.2, the expert opinion studies are explained. After all, in Section 7, the results obtained by the two research methods are discussed in relation to the RQs.

6.1. Case Studies

The case study method has commonly been seen as a fruitful way to come up with hypotheses and generate theories [196-199] since it is an in-depth, detailed examination of a particular case (or cases) within a real-world context [200]. Also, the case study method is the most commonly used method for validating meta-models [29]. Therefore, in this section, the case studies are employed to investigate the validity of the proposed OSS-QMM. The case studies are used to answer each of the three RQ, as shown in Table 6.1. In this context, a multiple-embedded case study design is employed [201]. Accordingly; reliability (in evaluation results), effectiveness (in model derivation) and applicability (in practice) of the OSS-QMM are demonstrated by using three OSS quality models by the three case studies, as shown in Fig. 6.2.

The first one of these quality models is a new operationalized OSS quality model which is derived from the OSS-QMM. A profile of this OSS quality model is shown in Appendix-3, visualizing it for a better understanding. In the first case study, the new OSS quality model is derived from the OSS-QMM and used for evaluating the quality of three open source ERP systems. The other two OSS quality models are the most used in the literature and also the most analyzed by the secondary studies (i.e., systematic mapping, systematic literature review, and comparison studies). Therefore, in the second and the third case studies, it is investigated if these two existing OSS quality models can be derived from the OSS-QMM and if they can be used for evaluating the three ERP systems, respectively.

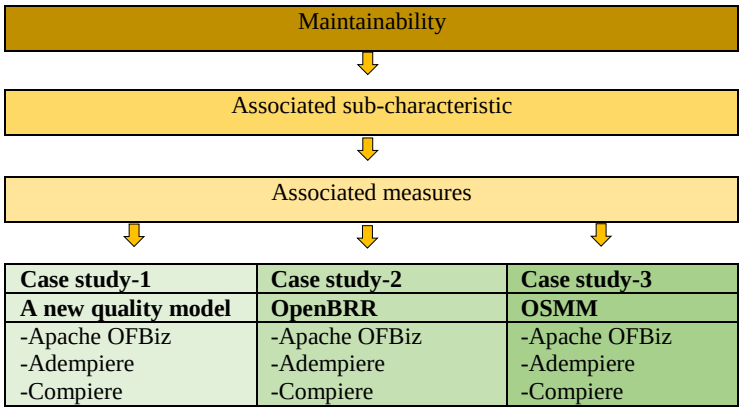


Figure 6.2. Multiple-embedded case study design

Before performing the case studies, preparations are carried out. In this regard, the OSS products to evaluate, the quality characteristic and the associated quality sub-characteristics to involve, the measures to quantify the quality sub-characteristics, and the methods (e.g., weighting method, evaluation aggregation method, etc.) to use in the evaluation are determined. In the following sub-sections, details about these preparations are given. Also, the enactment of the case studies is demonstrated within this section.

6.1.1. Determining the OSS Products

Numerous OSS products with different application domains have been developed in the market, and these products have taken their place in different cloud repositories, such as GitHub and SourceForge. Moreover, many OSS products that serve the same purpose, which can be alternatives to each other, are available in the market. Therefore, the evaluator must first determine the type of OSS product to evaluate, and then search for the alternative products that will fit the purpose. For the case studies, three open-source ERP (Enterprise Resource Planning) systems are selected since this product type (i.e.

ERP) is commonly needed by industry and the number of alternative products is vast. Companies use ERP systems to collect, manage, integrate, and store data from many business activities [202]. These systems are an essential asset of the companies because they help implement resource planning by integrating the processes needed to run the business within a single system [202-203]. They are increasingly gaining acceptance, especially by small and medium enterprises, for many reasons such that they generally impose no licensing cost [204] and that their source code is publicly available [205].

In this context, widely-used open source ERP systems were searched using the Internet search engines (Google, Yandex, Yahoo, etc.) and also the academic search engines (e.g. Google Scholar). From the search results, three open source ERP systems, which are written in Java programming language and alternatives to each other in terms of functionality, were identified as shown in Table 6.2. These ERP systems are Adempiere, Compiere, and Apache OFBiz. In addition to serving the same purpose, the fact that all these products are written in the same programming language, which supports the homogeneity factor in working systems, has an important place in selecting these products. In addition, the websites that provide the OSS products (Apache, SourceForge, Debian, Savannah, etc.) were searched, and it was verified that these ERP systems can be alternatives to each other in terms of popularity and functionality.

Table 6.2. List of selected OSS products used in case studies

Product Properties	**Apache OFBiz**	**Adempiere**	**Compiere**
Website	https://ofbiz.apache.org/	https://adempiere.org/	http://www.compiere.com/
Product type	Open-source ERP system	Open-source ERP system	Open-source ERP system
Programming language	Java	Java	Java
First release date	2009	2006	2000

6.1.2. Determining Quality Characteristics and Sub-characteristics

After the products to evaluate are determined, their quality characteristics to evaluate should also be determined. Since product quality is an abstract, complex and multi-faceted concept; it is decomposed into quality characteristics, each of which focuses on a specific concern such as reliability or maintainability. The product quality characteristics are used to elicit and specify the concerns of stakeholders regarding quality, and in turn, to determine how well the software product or system satisfies these concerns.

In this context, maintainability, one of the most concerned quality characteristics in general, was determined to specify and evaluate the quality of the three open source ERP systems in the three case studies. This is because maintainability plays an important role in the quality of the OSS products because of their sustainability by developer communities throughout the years. The SLR studies [24][29][30], which investigate OSS quality models, also support the importance of maintainability for OSS quality. The results of the most recent SLR study [30] indicated that maintainability is the most commonly evaluated quality characteristic by OSS quality models. Aside from this, maintainability is also the most measured quality attribute in well-designed quality models (e.g., ISO/IEC 25010 [206], McCall [21], Boehm [22], etc.).

Since maintainability is still abstract to measure as a quality characteristic, it is decomposed into sub-attributes to make it more concrete. The adapted, existing OSS quality models (i.e., OpenBRR and OSMM) define their own sub-characteristics and these were used in the case studies 2 and 3. Since a new operationalized OSS quality model was derived from the OSS-QMM in case study 1, its characteristics and sub-characteristics required to be determined specifically. To do this, the ISO/IEC 9126 quality model was taken as reference rather than its replacing counterpart ISO/IEC 25010. This was because the results of the SLR studies [24][29][30] indicate that the majority of the OSS quality models originated from this standard, and we searched for an alignment for the contexts of evaluation among the cases. The ISO/IEC 9126 standard provides a comprehensive specification and evaluation model for software product quality and explicitly addresses a product's user needs by allowing a common language for specifying user requirements by various stakeholders [207]. In this context, four sub-characteristic of maintainability – namely testability, analyzability, stability, and changeability – were determined for the case study 1.

6.1.3. Determining Measures and Measurable Concepts

Although the sub-characteristics are more concrete than the quality characteristics, they still remain abstract to measure directly as we specified in the OSS-QMM. That is, concrete measures are needed to quantify the sub-characteristics. The existing OSS quality models (i.e., OpenBRR and OSMM), which were used in the case studies 2 and 3, define their own set of measures and the scoring criteria. Therefore, the case studies 2 and 3 used the measures and the scores defined by OpenBRR and OSMM, respectively. In this way, it was investigated whether the existing OSS quality models comply and

work in harmony with the OSS-QMM. In this section, measures to use in the new, operationalized OSS quality model (in the case study 1) derived from the OSS-QMM are provided. This OSS quality model, which was developed within the scope of the case study 1, aimed to enable evaluations from both the code-based and the community-based aspects. Therefore, the measures for both OSS aspects required to be obtained.

In this context, firstly, code-based measures for evaluating maintainability were searched in the literature. It has been observed that there are many code-based measures used for this purpose. In the SLR study [208], measures to evaluate code maintainability are investigated, and the most popular and adopted measures among them are determined. This SLR study reports the total number of articles mentioning the measure, and calculates a score with respect to the frequency of use and the adoption of each measure in industry and academia. As a result of scoring, McCabe's cyclomatic complexity (CC), Chidamber and Kemerer (C&K) metric suite, and two others (i.e., number of statements (NOS) and number of nested levels (NNL)) were observed to have clearly higher scores than the remaining ones. Therefore, we used the CC, NNL, NOS measures and C&K metrics suite to evaluate code maintainability. In addition, the SLR study [30], in which we analyzed the OSS quality models, confirms that these measures are strongly adopted to evaluate maintainability of OSS. For the evaluations to be reasonable and consistent, it is important to note that the selected open source ERP systems are mostly written in Java and that the identified measures are valid for developments following an object-oriented approach. The list of the selected measures and their explanations are given in Table 6.3. *Measurable concepts* (MC) associated with each *measure* are also provided in this table.

Table 6.3. List of code-based measures with their description and measurable concepts associated with each measure [209-212]

Measurable concept (MC)	Measure	Description
MC1: Complexity of source code	**WMC:** Weighted methods per class	The degree of complexity and the number of methods in a class [213]. With the increasing number of methods, the code analyzability time will automatically increase.
	CC: Cyclomatic complexity	Measures the ratio of the flow of the program source code to follow independent paths from one another and is directly related to the complexity of the code. The high value of this metric is undesirable and will affect the source code analyzability.
	NNL: Number of nested levels	Measures the depth of nesting of the loops in a class, and a higher value of this metric reduces the testability and stability.
MC2: Comment frequency of source code	**NOS:** Number of statement	Measure the frequency of comments and explanations that will show us the way to reduce the complexity of software. It also facilitates the tracking and resolvability of the program.
MC3: Inheritance complexity degree of source code	**DIT:** Depth of inheritance tree	Measures the distance of a class to the root of the inheritance tree [214]. The high depth of the tree increases the complexity since it includes more classes and methods, indicating low changeability and the stability of the software product.
	NOC: Number of children	Measures the number of lower classes derived from a class. When the value of this metric is high, it indicates that the value of re-use is higher, more errors may occur [215], and a higher effort is required during testing [214].
MC4: Interaction Complexity (coupling) degree of source code	**CBO:** Coupling between object classes	Represents the number of classes coupled to a given class. This dependency is a dependency when some properties or methods in the class are used in other classes without inheritance between classes[215]. High levels of dependence between classes harm the modular design [214] and reduce changeability
	RFC: Response for a class	Measure the number of all the methods that can be triggered when calling methods of an object from one class to this object. Namely, the total number of written in a class and method called [214]. Software products with a lower RFC metric value can be better understood and tested.
MC5: Cohesion degree of source code	**LCOM:** Lack of cohesion of methods	Measures the degree of similarity of methods with each other [213]. Therefore, it is desirable to have low values of the metric.

After determining the measures belonging to the code-based aspect and the relations of these measures with the sub-attributes, secondly, the measures belonging to the community-based aspect required to be determined. In addition to the literature searches described above, additional effort was spent to obtain community-related measures. For this purpose, the websites of OSS repositories (e.g., GitHub, Sourceforge, Apache) were visited, and all possible measures to evaluate maintainability were determined. After this step, the relationships of the community-based measures with the sub-characteristic of maintainability were investigated. Although there are many community-based measures in the literature, no evidence was found on the relationship between the measures belonging to the community-based aspect and the maintainability sub-attributes. As mentioned in the SLR study [30], OSS quality models often used these measures to assess

the overall community quality. There are diverse types of data on the community-based side, and it is not easy to process them. In this context, we conducted exploratory research [216] both to understand that the determined community measures are suitable for evaluating maintainability and to associate these measures with the sub-characteristics. The details of this exploratory research are described in detail in Section 6.1.5.3. The list of the community-based measures obtained and the measurable concepts (MC) associated with these measures as a result of this process are given in Table 6.4. The description of each measure is provided in [217]. As specified in Table 6.4, some of the determined community measures are the derived ones. Therefore, in the last column of the table, the equations used for these derived measures and the base measures used in these equations are given.

Table 6.4. List of community-based measures with their equation and measurable concept associated with each measure

Measurable concept (MC)	Measure	Measurement functions (equation)
MC6: Difficulty degree of bug	***BSI:** Bug severity index	$\left(\frac{\#\ of\ blocker}{LOC}\times 9\right)+\left(\frac{\#\ of\ critical}{LOC}\times 7\right)+\left(\frac{\#\ of\ major}{LOC}\times 5\right)+\left(\frac{\#\ of\ minor}{LOC}\times 3\right)+\left(\frac{\#\ of\ trivial}{LOC}\times 1\right)$
MC7: Completeness of documentation	**ND:** Number of document	No equation *(it is a base measure)*
MC8: The activeness of the community	***CD:** Commit density	$\frac{(\#\ of\ commit)/(\#\ of\ developer)}{(kLOC)}$
	***ED:** Email density	$\frac{(\#\ of\ e-mail)/(\#\ of\ developer)}{(LOC)/(\#\ of\ release)}$
MC9: Size of contributor	**NC:** Number of contributors	No equation *(it is a base measure)*
MC10: Performance of contributor	***FRIS:** Feature request implementation success	$\frac{(\#\ of\ closed\ feature\ \ request)/(\#\ of\ total\ feature\ request)}{(kLOC)}$
	***BSSR:** Bug-solving success rate	$\frac{(\#\ of\ closed\ bug)/(\#\ of\ total\ bug)}{(kLOC)}$
MC11: Productivity of contributors	**NR:** Number of releases	No equation *(it is a base measure)*
MC12: Fault proneness of contributor	***DD:** Defect density	$\frac{\#\ of\ total\ defect}{(LOC)}$
MC13: Maturity of project	**PA:** Product age	No equation *(it is a base measure)*

6.1.4. Determining Evaluation Methods to Use in the Case Studies

In order to use the OSS-QMM in practice, some methods (with formulae or functions) should be selected and defined in OSS quality models as the instances of the concepts

(e.g., the weighting method and evaluation aggregation method) in the OSS-QMM. In this section, the methods that were used by the OSS quality model derived from the OSS-QMM (in the case study 1) and that were adapted for the existing OSS quality models (in the case studies 2 and 3) are introduced. The list of methods used in the case studies is shown in Table 6.5. The following sub-section describes the integrated AHP-TOPSIS method, which is essential for carrying out quality evaluations in the case studies.

Table 6.5. List of methods to use for the relevant concepts in the OSS-QMM

Concept of OSS-QMM	Methods used for case studies
Weighting method	Integrated AHP-TOPSIS method (AHP, Step 1-7)
Weight aggregation method	Weighted distribution
Normalize measure	Integrated AHP-TOPSIS method (TOPSIS, Step 2)
Measure aggregation method	Average of the measure
Measurement function	Some mathematical equation
Evaluation aggregation method	Integrated AHP-TOPSIS method (TOPSIS, Step 3-7)
Evaluation function	Linear utility function

6.1.4.1. Integrated AHP-TOPSIS

The OSS products have a dynamic and diverse nature, and their quality is affected by many variables. In other words, variety of qualitative and quantitative data is accessible from the code-based and the community-based aspects. The SLR study [30] investigated challenges faced while developing OSS quality models and evaluating OSS products. The results of the SLR study [30] indicated that, in addition to the diversity challenge mentioned above, stakeholders have different expectations from the OSS products. That is, each stakeholder will have different inputs for evaluation. In this context, processing and aggregating heterogeneous data from diverse sources considering the expectations of stakeholders is a complicated process. Since there are many criteria for evaluating software quality, it can be considered a Multi-Criteria Decision Making (MCDM) problem [218]. The MCDM is a powerful technique that allows managing multiple, complex and conflicting objectives in the evaluation. An array of MCDM methods exist, such as Analytic Hierarchy Process (AHP), Data Envelopment Analysis (DEA), Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), Analytic Network Process (ANP), etc. Among them, AHP and TOPSIS are the most widely used MCDM methods because of their strong mathematical background and systematic way of data collection [218-220]. They have been applied and validated in numerous multidisciplinary fields such as engineering [221], economics [222], social science [223],

etc. [224-225]. Therefore, the integrated AHP-TOPSIS method, having the steps shown in Fig. 6.3 [226], was used for evaluation in the case studies.

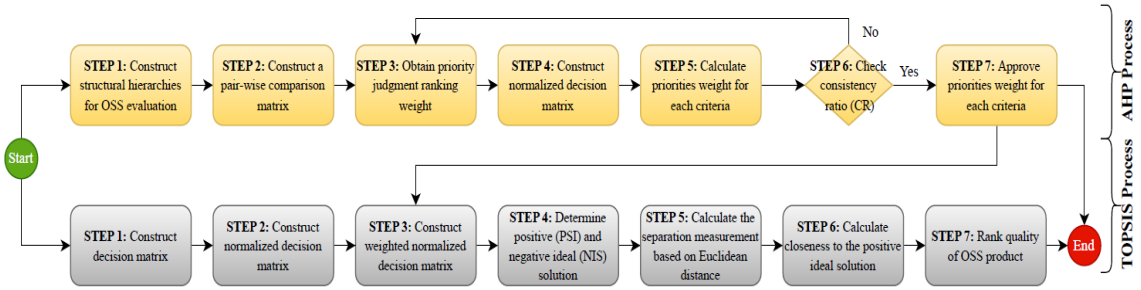


Figure 6.3. Integrated AHP-TOPSIS method used for quality evaluation in the case studies

AHP is very useful in involving several stakeholders with multiple conflicting criteria to arrive at a consensus, and TOPSIS is powerful in obtaining final scores for alternatives. A problem must be in a hierarchical structure to apply the AHP method. Since quality characteristics have a hierarchical structure, AHP was used to assign weights to them. The weights obtained from the AHP method, the value assigned to the concept of *impact*, and the actual measurement data were used as inputs to the TOPSIS method. The description and equations of the methods (in Table 6.5) used in the case studies are explained in Table 6.6.

Table 6.6. Description of evaluation methods with their formulas used in case studies

Technique	Description	Equation	
AHP	The AHP method consists of the following steps. Please see [227-228] for details.		
Step 1	Firstly, structural hierarchies are created. The concepts of OSS aspect and quality characteristics provide this condition.	No equation	
Step 2	A pair-wise comparison matrix A (size nxn) is constructed to compare the criteria in pairs. Each OSS aspect and related sub-characteristics are as "criteria".	$A = [x_{ij}]_{nxn}$ Matrix A is a pair-wise comparison matrix.	
Step 3	Pair-wise comparisons are performed by comparing the relative importance of two selected criteria. The matrix A is filled by using the scale 1-9, as proposed by Saaty [227] (see [228] for details).	A pairwise comparison is performed on matrix A, and the matrix is filled out.	
Step 4	The matrix A is normalized, and normalized pairwise comparison decision matrix A*norm* matrix is obtained. In this formula, each element of matrix A in a column is divided by the sum of the elements in the same column.	$A_{norm} = [a_{ij}]_{nxn} = {x_{ij}}/{\sum_{i=1}^{n} x_{ij}}$	**(1)**
Step 5	The final weight of each criterion is calculated.	$w_i = {\sum_{j=1}^{n} a_{ij}}/{n}$ *and* $\sum_{i=1}^{n} w_i = 1$ $i, j = 1,2, \dots n$	**(2)**
Step 6	The Consistency Ratio (CR) is calculated to check the consistency of the decision-maker's judgement. Firstly, the Consistency Index (CI) is calculated, where λ_{max} is the Eigenvalue (see [228-229] for details) corresponding to the matrix of pair-wise comparisons, and n is the number of criteria being compared. Then, CR is calculated. Here, Random Index (RI) is a value that depends on the number of criteria (n) (see [228-229] for values of RI according to n).	$\boldsymbol{CI} = (\lambda_{max} - n)/(n-1)$	**(3)**
		$\boldsymbol{CR} = CI/RI$	**(4)**
Step 7	The final weight of each criterion is approved.	No equation	

TOPSIS	The final weight of each criterion obtained from the AHP method is used as input to the TOPSIS method. The TOPSIS method consists of the following steps (please see [219][229-230] for details).		
Step 1	Firstly, decision matrix B= $[b_{ij}]_{mxn}$, where m is alternatives (i.e., OSS products) in the rows and n is evaluation criteria (i.e., measurable concepts) in the columns, is constructed.	$B = [b_{ij}]_{mxn}$ Matrix B is the decision matrix	
Step 2	Normalized decision matrix R= $[rij]_{mxn}$ is constructed.	$\boldsymbol{R} = [r_{ij}]_{mxn} = b_{ij} \Big/ \sqrt{\sum_{i=1}^{m} b_{ij}^2}$ $i = 1,2,3 \dots m$; and $j = 1,2,3 \dots n$	**(5)**
Step 3	The final weights obtained from the AHP method are multiplied by the values of the normalized decision matrix R. Thus, the weighted normalized decision matrix V= $[v_{ij}]_{mxn}$ is obtained.	$\boldsymbol{V} = \left[v_{ij}\right]_{mxn} = w_j \; x \; r_{ij}$ $i = 1,2,3 \dots m$; and $j = 1,2,3 \dots n$	**(6)**
Step 4	In this step, two artificial alternatives, A^+ (the positive ideal solution) and A^- (the negative ideal solution), are defined by Eq. (7) and Eq. (8), respectively. Here, J is the subset of {I = 1, 2, …, m}, which presents the concept of impact (positive impact) in the OSS-QMM, and J^- is the complement set of J.	$\boldsymbol{A^+} = \{(max_i v_{ij} \mid j \in J)(min_i v_{ij} \mid j \in J^-) \mid i = 1,2,3, \dots m\} = \{v_1^+, v_2^+, \dots v_j^+, \dots v_n^+\}$	**(7)**
		$\boldsymbol{A^-} = \{(min_i v_{ij} \mid j \in J)(max_i v_{ij} \mid j \in J^-) \mid i = 1,2,3, \dots m\} = \{v_1^-, v_2^-, \dots v_j^-, \dots v_n^-\}$	**(8)**
Step 5	In this step, separation measurement is performed by calculating the distance between each alternative in V and the ideal vector A^+ or the negative ideal A^- by using the Euclidean distance, which is given by Eq. (9) and Eq. (10), respectively. At the end of Step 5, two values, namely, S^+ and S^- for each alternative, have been counted. These two values represent the distance between each alternative and both the ideal and negative ideal.	$\boldsymbol{S_i^+} = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^+\right)^2}, \; i = \{1,2,3 \dots m\}$	**(9)**
		$\boldsymbol{S_i^-} = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2}, \; i = \{1,2,3 \dots m\}$	**(10)**
Step 6	In this process, the closeness of A_i (i^{th} alternative) to the ideal solution A^+ is defined, as shown in Eq. (11). $C_i^* = 1$ if and only if $A_i = A^+$; similarly, $C_i^* = 0$ if and only if $A_i = A^-$.	$\boldsymbol{C_i^*} = S_i^- / (S_i^- + S_i^+)$ $0 < C_i^* < 1, \quad i = \{1,2,3 \dots m\}$	**(11)**
Step 7	The set of alternatives A_i can now be ranked according to descending order of C_i^*, indicating that a higher value corresponds with better performance.	No equation	
Weighted distribution	The weight of each sub-characteristic for each OSS aspect can be different (these weights are calculated in the AHP process). Therefore, the final weight of each sub-characteristic as specific to the OSS aspect is calculated. Here, X_i is the final weight of a sub-characteristic for an OSS aspect, w_i^a are the weights of OSS aspects, w_j^s are the weights of OSS sub-characteristics, i is the number of OSS aspects (there are two OSS aspects), and m is the number of sub-characteristics.	$X_i = (w_i^a * w_j^s) \Big/ \sum_{i=1}^{n} w_i^a$ $\sum_{i=1}^{n} w_i^a = 1$ (see Eq. **(2)**) $i = \{1 \; or \; 2\} \quad j = \{1,2,3 \dots m\}$	**(12)**
Some math. equation	Some mathematical equations are used to obtain derived measures from the base measures in the concept of measurement function. For example, M1 and M2 are base measures, and M3 is a derived measure obtained from M1 and M2 using the following mathematical equation: M3=M1/(M1+M2). Therefore, this equation corresponds to the concept of measurement function in the OSS-QMM.	It can be a different kind of equations	
Average of the measures	In cases where multiple measures are associated with a measurable concept, these measures should be aggregated. The normalized measures (obtained in Step 2 of TOPSIS) associated with a measurable concept are averaged in this aggregation process. Here, p is the number of alternatives (OSS product), $m_{(k)}$ is a new value of measures associated with a measurable concept for k^{th} alternative, r_{ij} is a normalized measure (Step 2 of TOPSIS), and m, n are the first and the last indices of measures associated with a measurable concept, respectively.	$m_{(k)} = \sum_{j=1}^{n} r_{ij} / n$ $i = \{m \dots n\} \quad k = \{1,2,3 \dots p\}$	**(13)**
Linear utility function	The utility functions for each OSS product can be defined to operationalize the evaluation step. The higher the evaluation value of each of these OSS products and the best it is for software quality, the higher should be the associated utility. To reflect this, simple increasing linear utility functions can be selected with two thresholds, min and max, as shown in Fig. 6.4.	See Fig. 6.4	

6.1.5. Exploratory Study Applied as Part of Case Studies

A survey was conducted with industry experts to determine weights of sub-characteristics, relationships between community-based measures and sub-characteristics, and weights of OSS aspects as an initial part of the case studies. To do that, questions under five parts were prepared to ask to the experts [198]. The application of the survey in the case study 1 is shown in this study, and the applications of the survey in the other two case studies are shown in a supplementary document [231]. The part-5 of the survey is about performing the case studies 2 and 3, which is also shown in the supplementary document.

Prior to executing the survey, first, the participants working in well-known companies were identified and contacted via e-mail, and online meetings were arranged with each separately. Since the primary application area of the OSS-QMM is software industry, this survey allowed the implementation of the OSS-QMM from the viewpoints of the industry experts, and the information gathered guided the enactment of the case studies. Details about the survey is given in the following sub-sections, together with the parts considered in each case study.

6.1.5.1. Part-1: Background of the Participants (for the Case studies 1, 2, and 3)

This part of the survey includes questions aimed at obtaining information about the background of the participants. Thus, it was aimed to determine the concept of ***viewpoint*** in the OSS-QMM to be used in the case studies. In this context, the participants were expected to answer the following questions; positions in their company, periods of experience in each position, level of knowledge about OSS (to rate 1-5 in Likert scale), and experience in OSS quality evaluation (Yes or No). Although the survey was conducted with a total of 24 participants in different positions, the majority of the experts were in the developer position; so that the *developer* position was taken into consideration and determined as the *viewpoint* in the case studies. However, some additional criteria were sought in the experts in addition to being a developer, such that: the expert should work five or more years in the developer position, rate their OSS knowledge as 4 or 5, and have experience with OSS quality evaluation. Satisfying all these criteria, a total of 11 experts were determined, and some background information about these experts (E) is given in Table 6.7.

Table 6.7. Background of industry experts participated in the case studies

Background/Expert	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11
Experience in the developer position	5	9	5	6	7	6	5	10	6	5	7
Experience in other positions	PM	SA, SC	PM, TD, T	BDM, IM, PM	-	-	DS	-	SA	PM, T	SA, SE
Experience in OSS quality evaluation	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
OSS knowledge	4	5	5	5	4	5	4	4	5	5	4

Abbreviations: Project manager (PM), software architect (SA), security consultant (SC), technology director (TD), business development manager (BDM), innovation manager (IM), data scientist (DS), Tester (T), software engineer (SE)

6.1.5.2. Part-2: Weighting Sub-characteristics of the New OSS Quality Model (for Case Study 1)

The open source ERP systems were evaluated in terms of maintainability in the case studies as specified in Section 6.1. In this context, sub-characteristics of maintainability were determined as criteria (i.e., analyzability, changeability, stability, and testability) for the case study 1. These sub-characteristics were given weights (i.e., for the concept of ***weighting*** in the OSS-QMM) according to the degree of importance and considering a certain viewpoint. For the concept of the ***weighting method*** (i.e., assigning weights), the AHP method was used as specified in Table 6.5. The AHP process is implemented as explained in the following steps, as already given in Table 6.6. In addition, the process is demonstrated over the judgement of Expert #10 in Table 6.8, for a better understanding.

- **Step 1**: Maintainability was decomposed into four sub-characteristics (i.e., analyzability, changeability, stability, and testability); it means that the required hierarchy was provided (see Table 6.6).
- **Step 2**: A pair-wise comparison matrix A (size n*x*n) was constructed using sub-characteristics as criteria (see Table 6.6).
- **Step 3:** A total of 11 experts were asked to fill in the comparison matrix A using the scales (1-9) individually. The comparison matrix A was filled out as a result of the pairwise comparison of experts (see Table 6.8 (a)).
- **Step 4:** The decision matrix A_{norm} which was normalized using Eq. (1) was obtained (see Table 6.8 (b)).
- **Step 5:** Eq. (2) was applied to the matrix $A_{norm,}$ and the final weight of the priorities was obtained according to the judgment of each expert for each sub-

characteristic (see Table 6.9). The weights obtained for Expert #10 are shown in Table 6.8 (c).

- **Step 6:** Consistency Ratio (CR) was calculated using Eq. (3-4) to see if the judgments of the experts were consistent (see Table 6.9). The calculation of CR is explained in detail in the studies [228-229] and not repeated here to save space.
- **Step 7:** It was observed that CR values were less than 0.1, which means that the judgments of each expert were consistent [229]. If this was not the case, the pairwise comparison (in Step 3) should have been revised [229].

Table 6.8. Parts of AHP process w.r.t. the Expert #10's judgement: **(a)** Pair-wise comparison, **(b)** Normalized decision matrix, and **(c)** Weight of sub-characteristics

(a)

	A	C	T	S
A	1	1/3	5	4
C	3	1	7	6
T A=	1/5	1/7	1	1/4
S	1/4	1/6	4	1
Total	4.450	1.643	17.00	11.25

nxn

(b)

	A	C	T	S
A	0.225	0.20	0.294	0.356
C	0.674	0.61	0.412	0.533
T A_{norm}=	0.045	0.09	0.059	0.022
S	0.056	0.10	0.235	0.089

(c)

	Weight
A	0.2693
C	0.5569
T	0.0532
S	0.1204
CR	*0.0970*

Finally, the average value was calculated from the priority weights from all experts for each sub-characteristic, and final weights were obtained for the sub-characteristics in the case study 1. As seen in Table 6.9, the importance of changeability and analyzability was higher than the other two sub-characteristics. The most important reason for this might be that the developer was determined as the viewpoint in the evaluation, and it is essential for a developer to easily analyze and change the source code in the OSS.

Table 6.9. The weights for each sub-characteristics according to expert's judgements, the average of these weights, and the CR values

Sub-char. /Expert	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**Avg.**
Analyzability	0.1823	0.2047	0.5960	0.4392	0.1244	0.5998	0.5383	0.2212	0.1847	0.2693	0.4788	0.3490
Changeability	0.5011	0.5214	0.2282	0.2768	0.4213	0.2127	0.1794	0.4566	0.6038	0.5569	0.1615	0.3745
Testability	0.0678	0.0930	0.0436	0.0983	0.0855	0.1160	0.0819	0.1202	0.0713	0.0532	0.1059	0.0851
Stability	0.2487	0.1807	0.1321	0.1855	0.3686	0.0713	0.2002	0.2019	0.1401	0.1204	0.2536	0.1912
Consistency Ratio (CR)	*0.0590*	*0.0790*	*0.0510*	*0.0530*	*0.0290*	*0.0100*	*0.0070*	*0.0160*	*0.0230*	*0.0970*	*0.0530*	*0.0430*

6.1.5.3. Part-3: Understanding Relationships Between Community-based Measures and Sub-characteristics (for the Case study 1)

In the OSS projects, aside from the code-based data, diverse types of historical data belonging to the community-based aspect are stored in public repositories. This indicates that heterogeneous and scattered community-based data from different sources must be coped with while evaluating OSS quality. In this context, possible community-based measures that could be used to evaluate maintainability were determined for the case study 1 by following the process described in Section 6.1.3. However, while there are lots of evidence in the literature regarding the relationship between code-based measures and sub-characteristics of maintainability, there is little or no evidence for the relationship between community-based measures and these sub-characteristics. Therefore, this part of the survey was formed to understand whether the determined community measures were suitable for evaluating maintainability and also to associate these measures with the sub-characteristics.

In this context, experts were asked to associate possible measures with the sub-characteristics of maintainability, after having reminded that each measure might relate to one, more, or none of the sub-characteristics. While deciding on a relationship between a sub-characteristic and a community-based measure, it was required that seven or more experts agreed on the relationship. Accordingly, the community-based measures used to evaluate each sub-characteristic were identified and are marked by green color in Table 6.10. The measures that were determined as possible for evaluating the sub-characteristics but were not associated with any sub-characteristics by the experts are not given in the table. It should be reminded that the existing OSS quality models (i.e., OpenBRR and OSMM) used in the case studies 2 and 3 define their own sub-characteristics and related measures.

Table 6.10. Relationship between sub-characteristic and community-based measures

Sub-characteristic/ community-based measure	Defect density	E-mail density	Bug-solving success rate	Bug severity index	Number of contributors	Number of downloadable	Product age	Number of releases	Feature request implementation	Commit density
Analyzability	4	5	3	9	4	8	2	3	1	5
Changeability	1	4	5	0	9	2	5	7	8	8
Testability	8	2	9	8	2	0	1	2	1	2
Stability	7	8	5	5	2	3	9	8	5	4

6.1.5.4. Part 4: Weighting OSS-aspects (for the Case Studies 1, 2, and 3)

The OSS evaluation stage consists of the code-based and the community-based aspects, and the importance of these aspects may vary with respect to different viewpoints. These two aspects (i.e., specified as "criteria" in Table 6.6) should be given weights (i.e., for the concept of ***weighting*** in the OSS-QMM) according to the degree of importance with respect to a certain viewpoint. As explained in part-2 of the questionnaire, the process of the AHP method which is described in Section 6.1.5.2 was followed in assigning these weights. As different from the process given in Section 6.1.5.2, it was unnecessary to calculate the CR value since there were two criteria (i.e., OSS aspects) for pair-wise comparison. The weights obtained for each OSS aspect at the end of the AHP process according to the expert's judgments and the average of these weights are given in Table 6.11. These weights were used in the case studies 1, 2, and 3. As seen in the table, the importance of the code-based aspect was higher than that of the community-based aspect. This might be because the OSS has quality source code for the developer, as the main reason for the developer to use the OSS is to use the existing code base by changing it according to own needs.

Table 6.11. The weights for each OSS aspect according to the expert's judgements and the average of these weights

OSS aspects /Expert	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**Avg.**
Code-based aspect	0.8	0.75	0.875	0.6666	0.3333	0.8571	0.5	0.5	0.8333	0.5	0.3333	0.6317
Community-based aspect	0.2	0.25	0.125	0.3333	0.6666	0.1428	0.5	0.5	0.1666	0.5	0.6666	0.3682

6.1.6. Performing the Case Study-1

This section explains the usage of the OSS-QMM for creating the new, operationalized OSS quality model by the case study 1. The application of the OSS-QMM on the two existing OSS quality models are explained in [231] to save space. As shown in Fig. 3, the stages of using the OSS-QMM for instantiating quality models are classified as *specification*, *measurement*, and *evaluation*. To provide the traceability of the quality models instantiated in the case studies to the concepts of the OSS-QMM, the concepts are shown in **bold** and *italic*, and given in accordance with their stages classified.

In the *specification* stage, the required preparations were performed as described in Sections 6.1.1 – 6.1.5 before applying the case studies. In this context, three open source ERP products were determined as alternatives. In addition, the following concepts were determined: ***characteristics*** and ***sub-characteristics*** to evaluate, ***information needs***, ***entity***, ***measurable concepts*** (belonging ***code-based*** and ***community-based*** aspects), ***measures*** associated with these measurable concepts, and the methods (e.g., the concepts of ***weighting method***, ***measurement function***, etc.) as listed in Table 6.5. The values for these concepts are given with their explanation in Sections 6.1.1 – 6.1.5. Apart from these, the relationship of measurable concepts with sub-characteristics and their ***impacts*** on sub-characteristics were determined on the basis of ***quality requirements,*** as shown in Table 6.13. In addition, as explained in Section 6.1.5, survey was conducted to determine: the ***viewpoint*** to take into account in the measurement, the weights of the OSS aspects and the sub-characteristics according to the specified viewpoint, and the relations of the sub-characteristics with the community-based measures. As a result of this survey, the "developer" was determined as the viewpoint as stated in Section 6.1.5.1. Each OSS aspect and sub-characteristic were weighed by the pair-wise comparison according to the judgment of the developer's viewpoint. This corresponds to the concept of ***weighting*** in the OSS-QMM. Then, the weights for each sub-characteristic and OSS aspect were calculated with the AHP method (Steps 1-7), the application of which is shown in Section 6.1.4.1 with related equations. The AHP method corresponds to the concept of the ***weighting method*** in the OSS-QMM, as previously specified in Table 6.5.

Then, the final weights of the sub-characteristics affecting each OSS aspect were calculated, as shown in Table 6.12, using the weighted distribution method by using Eq. (12). This method corresponds to the concept of ***weighting aggregation method*** in the OSS-QMM, as specified Table 6.5**.** The rationale behind this concept is that from a

particular viewpoint, the importance of each sub-characteristic may be different in each OSS aspect. That is, the weights of the sub-characteristics should be distributed on the OSS aspects according to the importance of the OSS aspect. For example, assume that the evaluator is a developer, and the code-based aspect is more critical for this viewpoint. Therefore, the analyzability sub-characteristic of the source code in the code-based aspect should have a greater importance for this evaluator. In other words, the weight of analyzability on the code-based side should be greater than the one on the community-based side.

Table 6.12. **(a)** weights of sub-characteristics w.r.t importance, **(b)** weights of OSS aspects w.r.t importance, and **(c)** final weights for sub-characteristics in the case study-1

(a)

Sub-char.	Weight
Analyzability (A)	0.3490
Changeability (C)	0.3745
Testability (T)	0.0851
Stability (S)	0.1912

(b)

OSS-aspect	Weight
Code-based	0.6317
Community-based	0.3682

Apply Eq. (12)

(c)

OSS aspect	Code-based (0.6317)				Community-based (0.3682)			
Sub-characteristic	A (0.3490)	C (0.3745)	T (0.0851)	S (0.1912)	A (0.3490)	C (0.3745)	T (0.0851)	S (0.1912)
Final weight	0.2204	0.2366	0.0538	0.1208	0.1285	0.1379	0.0313	0.0704

In the *measurement* stage, the concepts are used to quantify the quality of an OSS product via measures belonging to code-based and community-based aspects. The general information about the determined code-based and community-based measures is provided in Section 6.1.3. As shown in Table 6.3, code-based measures are ***base measures*** which can be quantified directly. As specified in the OSS-QMM, a ***measurement method*** is determined to obtain values for these measures, and the values are obtained ***automatically***. Research has indicated that code analyzer tools are available such as MetricsReloaded (IntelliJ IDEA plugin) [232], CodeMetrics (IntelliJ IDEA plugin) [233], CKJM [234], and Understand Scitool [235] in the literature to automatically obtain the values of the measures. Among these tools, the Understand Scitool was selected due to its ease of use. Accordingly, the values of the determined measures were obtained at the class level from the source code of the open source ERP products. The values of the code-based measures for each ERP product are shown in Table 6.13. As indicated in Table 6.4, community-based measures consist of the base and derived measures. The ***measurement functions*** for computing these derived measures and the base measures used in these measurement functions are given in Table 6.4. For example, the defect density is a derived measure and calculated by dividing the total number of defects by the total number of

code lines. As specified in the OSS-QMM, a ***measurement method*** is determined to obtain values for the community measures, and the values are obtained ***manually***. The implementation of the AHP method has been demonstrated in the specification part. Here, the TOPSIS method is employed using the outputs of the AHP process. The steps of TOPSIS are represented in Fig. 6.3 and their equations are given in Table 6.6 (see [219][229-230] for details). The first two steps of the TOPSIS were employed in the measurement part for the concept of the ***normalized measure*** as follows.

- **Step 1**: The decision matrix B =$[b_{ij}]_{mxn}$ was constructed as shown in Table 6.13 after the measure values belonging to code-based and community-based aspect were obtained. In matrix B, ***m*** represents the number of ERP systems (i.e., alternatives), and ***n*** represents the number of measures (i.e., evaluation criteria) as specified in Table 6.6.
- **Step 2**: As seen in Table 6.13, as the values of the measure computed have heterogeneous ***scales*** and ***units***, they should be normalized using the concept of ***normalized measure.*** Therefore, the normalized decision matrix R =$[r_{ij}]_{mxn}$ was obtained, as represented in Table 6.14, by applying Eq. (5) to decision matrix B.

Table 6.13. Final weights of sub-characteristics, impacts, measurable concepts (MC), measures associated with MC, values of measures, and decision matrix B.

OSS aspect	Code-based aspect															..
Sub-charac.	Analyzability			Changeability				Testability				Stability				..
Final weight	0.2204			0.2366				0.0538				0.1208				..
Impact	⊖		⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖		⊖	..
M. concept	MC1		MC2	MC1	MC3	MC4	MC5	MC1		MC3	MC4	MC1	MC3		MC4	..
Measure	WMC	CC	NOS	NNL	DIT	CBO	LCOM	NNL	CC	NOC	RFC	NNL	NOC	DIT	CBO	..
Adempiere	4.5977	1.114	17.025	0.428	1.775	12.923	25.374	0.428	1.114	0.984	76.578	0.428	0.984	1.775	12.923	..
Compiere	7.315	2.18	25.81	0.696	2.084	12.116	25.021	0.696	2.18	0.781	95.359	0.696	0.781	2.084	12.116	..
Apache Ofbiz	6.478	1.392	19.094	0.563	1.587	11.706	16.799	0.563	1.392	0.434	11.014	0.563	0.434	1.587	11.706	..

	Community-based aspect													
…	Analyzability		Changeability				Testability			Stability				
…	0.1285		0.1379				0.0313			0.0704				
…	⊖	⊕	⊕	⊕	⊕	⊕	⊖	⊕	⊖	⊕	⊕	⊖	⊕	
…	MC6	MC7	MC8	MC9	MC10	MC11	MC6	MC10	MC12	MC8	MC11	MC12	MC13	
…	BSI	ND	CD	NC	FR	NR	BSI	BSSR	DD	ED	NR	DD	PA	
…	0.0044	5	0.0333	0.0144	0.0008	28	0.0044	0.0001	0.001	0.0012	28	0.001	16	
…	0.0365	4	0.3033	0.0191	0.0036	37	0.0365	0.0007	0.0063	0.0176	37	0.0063	22	
…	0.0136	5	0.4687	0.0248	0.0058	45	0.0136	0.0006	0.0032	0.0696	45	0.0032	13	

= Matrix B

If a measurable concept (MC) is associated with more than one measure, these associated measures should be aggregated using the concept of ***aggregated measure.*** In this context, the average of the measure's values was calculated using Eq. (13) within the scope of the ***measure aggregation method*** concept, as specified in Table 6.6. In this case study 1, the measures associated with the MC1 and MC3 were aggregated, as shown in Table 6.14.

Then, ***measurement*** was performed as action, and ***measurement results*** were produced for each measurable concept before the evaluation phase.

Table 6.14. Normalized decision matrix R

OSS aspect	Code-based aspect												
Sub-charac.	Analyzability		Changeability				Testability			Stability			…
Final weight	0.2204		0.2366				0.0538			0.1208			..
Impact	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	…
M. concept	MC1	MC2	MC1	MC3	MC4	MC5	MC1	MC3	MC4	MC1	MC3	MC4	…
Measure	Avg. (WMC. CC)	NOS	NNL	DIT	CBO	LCOM	Avg. (NNL. CC)	NOC	RFC	NNL	Avg. (NOC. DIT)	CBO	…
Adempiere	0.4106	0.4684	0.4313	0.5609	0.6086	0.6440	0.4134	0.7403	0.6236	0.4313	0.6506	0.6086	…
Compiere	0.7257	0.7102	0.7014	0.6586	0.5706	0.6351	0.7377	0.5876	0.7765	0.7014	0.6231	0.5706	…
Apache Ofbiz	0.5470	0.5254	0.5673	0.5015	0.5513	0.4264	0.5308	0.3265	0.0896	0.5673	0.4140	0.5513	…

	Community-based aspect												
…	Analyzability		Changeability				Testability			Stability			
…	0.1285		0.1379				0.0313			0.0704			
…	⊖	⊕	⊕	⊕	⊕	⊕	⊖	⊕	⊖	⊕	⊕	⊖	⊕
…	MC6	MC7	MC8	MC9	MC10	MC11	MC6	MC10	MC12	MC8	MC11	MC12	MC13
…	BSI	ND	CD	NC	FR	NR	BSI	BSSR	DD	ED	NR	DD	PA
…	0.1122	0.6154	0.0595	0.4179	0.1163	0.4331	0.1122	0.1078	0.1401	0.0167	0.4331	0.1401	0.5306
…	0.9311	0.4923	0.5423	0.5543	0.5237	0.5724	0.9311	0.7548	0.8827	0.2451	0.5724	0.8827	0.7296
…	0.3469	0.6154	0.8380	0.7197	0.8438	0.6961	0.3469	0.6469	0.4483	0.9693	0.6961	0.4483	0.4311

= Matrix R

In the *evaluation* stage, three inputs are needed for the concept of ***evaluation*** to start: measurement results, impacts, and final weights of sub-characteristics. The concept of ***evaluation aggregation*** should be used to aggregate these inputs. In this case study 1, the concept of the ***evaluation aggregation method*** was used to aggregate these three inputs. In other words, as specified in Table 6.5, we used the integrated AHP-TOPSIS method (TOPSIS, Steps 3-7) for this concept. Therefore, the steps 3-7 of TOPSIS were used for the evaluation as follows:

- **Step 3**: The weighted normalized matrix $V = [v_{ij}]_{mxn}$ was obtained, as shown in Table 6.15, by using Eq. (6). More clearly, the final weight of each sub-characteristic in Table 6.15 was multiplied by each associate normalized measure.
- **Step 4**: As shown in Table 6.16, the A^+ (PIS: positive ideal solution) and A^- (NIS: negative ideal solution) were calculated by Eq. (7) and Eq. (8), respectively. In other words, matrix V is used to calculate the impacts ((i.e., (+) or (-)). More clearly, if the impact is positive, the PIS value is the maximum of the normalized measure in the associated column, as shown in Table 6.15; if it is negative, it is the minimum one, and vice versa for NIS.
- **Step 5:** As shown in Table 6.17, separation measures based on Euclidian distance for PIS (i.e., S^+) and NIS (i.e., S^-) were calculated by Eq. (9) and Eq. (10), respectively. Please see the studies [229][230] for detail on its calculation.

- **Step 6:** The final quality score (i.e., the closeness of each alternative to the ideal solution (i.e., PIS)) was calculated by using Eq. (11), as shown in Table 6.17.
- **Step 7:** As shown in the last column of Table 6.17, each ERP system was ranked according to its final quality score, indicating that a higher value corresponds to better quality.

Table 6.15. Weighted normalized decision matrix V

OSS aspect	Code-based aspect												
Sub-charac.	Analyzability		Changeability				Testability			Stability			…
Final weight	0.2204		0.2366				0.0538			0.1208			..
Impact	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	…
M. concept	MC1	MC2	MC1	MC3	MC4	MC5	MC1	MC3	MC4	MC1	MC3	MC4	…
Measure	Avg. (WMC. CC)	NOS	NNL	DIT	CBO	LCOM	Avg. (NNL. CC)	NOC	RFC	NNL	Avg. (NOC. DIT)	CBO	…
Adempiere	0.0452	0.0516	0.0255	0.0331	0.0360	0.0380	0.0074	0.0132	0.0111	0.0173	0.0261	0.0245	…
Compiere	0.0799	0.0782	0.0414	0.0389	0.0337	0.0375	0.0132	0.0105	0.0139	0.0282	0.0250	0.0229	…
Apache Ofbiz	0.0602	0.0579	0.0335	0.0296	0.0326	0.0252	0.0095	0.0058	0.0016	0.0228	0.0166	0.0222	…

	Community-based aspect												
…	Analyzability		Changeability				Testability			Stability			
…	0.1285		0.1379				0.0313			0.0704			
…	⊖	⊕	⊕	⊕	⊕	⊕	⊖	⊕	⊖	⊕	⊕	⊖	⊕
…	MC6	MC7	MC8	MC9	MC10	MC11	MC6	MC10	MC12	MC8	MC11	MC12	MC13
…	BSI	ND	CD	NC	FR	NR	BSI	BSSR	DD	ED	NR	DD	PA
…	0.0072	0.0395	0.0020	0.0144	0.0040	0.0149	0.0011	0.0011	0.0014	0.0002	0.0076	0.0024	0.0093
…	0.0598	0.0316	0.0186	0.0191	0.0180	0.0197	0.0097	0.0078	0.0092	0.0043	0.0100	0.0155	0.0128
…	0.0222	0.0395	0.0288	0.0248	0.0290	0.0240	0.0036	0.0067	0.0046	0.0170	0.0122	0.0078	0.0075

= Matrix V

Table 6.16. Values of Positive Ideal Solution (PIS) and Negative Ideal Solution (NIS)

A+ (PIS)	0.0452	0.0782	0.0255	0.0296	0.0326	0.0252	0.0074	0.0058	0.0016	0.0173	0.0166	0.0222	…
A- (NIS)	0.0799	0.0516	0.0414	0.0389	0.0360	0.0380	0.0132	0.0132	0.0139	0.0282	0.0261	0.0245	…

…	0.0072	0.0395	0.0288	0.0248	0.0290	0.0240	0.0011	0.0078	0.0014	0.0170	0.0122	0.0024	0.0128
…	0.0598	0.0316	0.0020	0.0144	0.0040	0.0149	0.0097	0.0011	0.0092	0.0002	0.0076	0.0155	0.0075

Table 6.17. Separation measurement (S+ and S-), final quality evaluation score and rank values for ERP products

OSS Product	**S^+**	**S^-**	**Score**	**Rank**
Adempiere	0.0551	0.0692	0.5207	2
Compiere	0.0753	0.0366	0.3162	3
Apache OFBiz	0.0322	0.0675	0.6997	1

The final quality score for each ERP system obtained from the concept of the ***evaluation*** can be interpreted with the concepts of ***evaluation function*** or ***manual evaluation*** as specified in the OSS-QMM. For example, an expert opinion can be considered as the

concept of manual evaluation, and an expert can interpret the results according to the final quality scores or rank values. In this case study 1, as shown in Fig. 6.4, the linear utility function was used as the concept of the evaluation function to interpret the final quality scores. As seen from the figure, x_{max} was 1, and therefore, according to the formula $u = x/x_{max}$ the $u(x)$ value of each ERP system was the same as the final quality score of them. After all these processes, the evaluation results can be produced at this point by the concept of the ***evaluation result***. As seen from Fig. 6.4, Apache OFBiz was determined as the most preferable product in terms of maintainability, followed by Adempiere and Compiere, respectively. Also, considering that a>b in the figure, it is seen that the difference in quality between Compiere and Adempiere was greater than the difference in quality between Adempiere and Apache OFBiz.

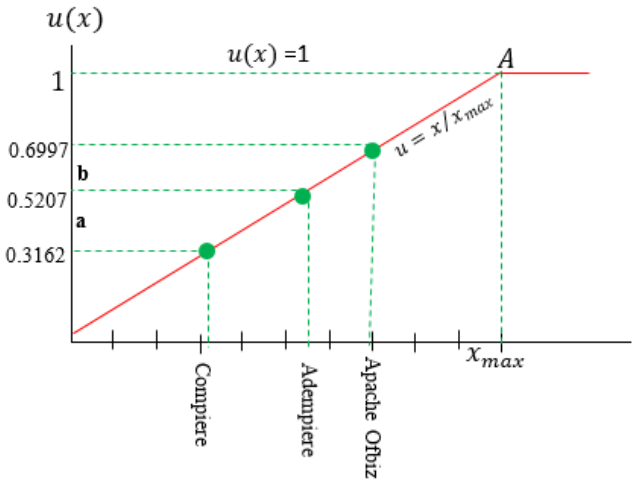


Figure 6.4. Linear utility function according to the final quality scores of ERP products

6.2. Expert Opinion Studies

The opinions of the experts in the market are crucial for the validation process of the OSS-QMM since the primary purpose of software engineering is to release high-quality software to the market. Similarly, since the primary purpose of the development of the OSS-QMM is to use it in the industry, it is important to validate it with expert opinions from the industry. Also, SLR studies [30][37] have concluded that expert opinion is one of the critical techniques used in the validation of models [5] or meta-models [236]. Therefore, semi-structured interviews, which are frequently used as a data collection technique in literature [237], were planned with domain experts. The semi-structured interview is a research method that supports the gathering and discussing of ideas from domain experts in line with a predetermined thematic framework [237]. In this context,

the domain experts working in important companies were determined, and online or face-to-face meetings were held with each separately. It should be noted that the experts consulted during the development stage of the OSS-QMM, the experts consulted in the implementation of the case studies, and the experts consulted during the validation process were different persons. This validation method (i.e., expert opinion) was used to answer the RQs in Table 6.1. In the following sub-sections, firstly, the questionnaire design and its execution are explained, and then the answers obtained from the experts are discussed.

6.2.1. Questionnaire Design and Execution

In this section, the effort spent in designing and executing the questionnaire is explained. To design the questionnaire, the following process has been followed; (*i*) we have used our experience on software quality modeling and open source software [30][54][57], (*ii*) we have considered the deficiency of OSS-QMs based on our SLR study [30], (*iii*) we have considered the fundamentals and intended use of OSS-QMM based on our SLR study [37], (*iv*) we have analyzed studies on validating the meta-models (e.g., [195]), and (*v*) we have analyzed accepted and well-known guidelines on conducting questionnaires [238]. As a result of this process, we have prepared a draft of the questionnaire.

To improve this draft, we have performed a series of meetings between this student and his supervisor. During these meetings, we have tried to improve the content and quality of the questionnaire. Then, we carried out pilot studies with two practitioners from industry and academia who have active research on software quality modeling. Accordingly, we have revised the questions to ensure that they are complete and consistent based on the feedback received. After this process, we have obtained a final version of the questionnaire as given in [197]. The questionnaire was available online via JotForm [239] between June 2022 and October 2022.

At the beginning of the questionnaire, information about the purpose (i.e., validation of a meta-model) and content (i.e., type of questions, how it will be applied, and time required) of the questionnaire have been given to the participants. Then, a pre-interview has been held with the participants, who have agreed to participate in the questionnaire, in order to obtain information about their backgrounds. If the background of a participant met the requirement given in Table 6.18, the opinion of this expert has been taken into account to investigate the validity of the OSS-QMM. In this context, a prerequisite has been applied

that experts should have seven years or more experience in the field of software quality and its modeling (e.g., software engineer (SE), software quality assurance (QA) engineer, QA manager, SE manager, lead QA engineer, QA tester, QA analyst, and IT consultant, etc.), regardless of other experiences. Also, the experts should rate their OSS and quality modeling knowledge as 4 or 5 on a 5-point Likert scale (5 indicating the highest degree of knowledge). Although 29 experts have been determined and pre-interviewed; after reviewing their experiences, 20 of them whose experiences are given in Table 6.18 have been selected after applying the prerequisite mentioned above. Detailed information about the background of all 29 experts is provided in Appendix-4. In addition to the information given in Table 6.18, the experts' current position, company size, the country where the companies are located, and interview duration are provided in this appendix. It should be noted that the durations of the interviews with the experts who were pre-interviewed and did not provide the necessary experience are not given in the appendix.

Table 6.18. Background of experts (E1…E20) consulted during the validation process

Background/Expert	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**E12**	**E13**	**E14**	**E15**	**E16**	**E17**	**E18**	**E19**	**E20**
Experience in software quality modeling	11	7	9	10	8	22	8	12	7	14	10	7	13	17	8	12	15	7	9	11
OSS knowledge	5	5	4	4	4	5	4	5	5	4	5	5	4	5	5	5	4	4	5	4
Experience in selecting the wrong software	Y	Y	N	Y	Y	Y	N	Y	Y	Y	N	N	N	Y	Y	Y	N	Y	Y	N
Quality modeling knowledge	5	5	4	5	4	5	5	4	4	5	4	5	5	5	4	5	5	4	4	5

In order to reach experts with the necessary background, we have shared the questionnaire with: (*i*) the personal connections of this student and his supervisor, (*ii*) industry-experienced authors of the papers proposing meta-models, (*iii*) some actively used mailing lists, (*iv*) engineers with necessary experience working for important companies, (*v*) people with experience in software quality and its modeling in the industry by searching the LinkedIn, and (*vi*) some actively used LinkedIn groups (e.g., quality assurance, quality modeling, open source software, etc.). As indicated in Table 6.18, after applying the abovementioned prerequisite, a total of 20 experts whose opinions to take into account in the validation process have been identified. Then, online or face-to-face meetings have been held with each separately. In these meetings, experts have been provided with a two-part questionnaire to investigate the validity of the RQs given in Table 6.1. The questionnaire has been provided to the experts before the meeting so that they could obtain preliminary information about the subject. In the interview meetings,

firstly, the development process of the OSS-QMM has been presented to the experts, as well as the OSS-QMM itself together with the concepts, their meanings and intended use, and the relationships between these concepts. Then, the questions in two parts have been asked to each expert individually. In addition, before each question, experts have been provided with the necessary information about the purpose of each question.

The Part-1 of the questionnaire has consisted of questions to validate RQ1 and RQ2 (see Table 6.1) in terms of the comparability of evaluation results and the effectiveness of model derivation. In Part-2, experts have assessed the practical applicability of the OSS-QMM considering its structure and content to validate RQ3 (see Table 6.1). These parts have consisted of some single-select multiple-choice questions (Part-1: Q1 and Q2), multi-select multiple-choice questions (Part-1: Q1.1, Q2.1, and Q4), 10-point Likert scale questions (Part-1: Q3) and 5-point Likert scale questions (Part 1: Q5, Q6; and Part 2: Q1, Q12). As an example, the answers obtained after the semi-structured interview conducted with Expert-10 are given in Appendix-5. In order to protect the confidentiality of the participants' personal information, each expert has been assigned a different code (e.g., E10). In addition, the answers obtained from all experts are given in this excel sheet [197]. The techniques used to analyze the data obtained after the completion of the semi-structured interviews together with the results obtained are described in Sections 6.2.2 and 6.2.3.

6.2.2. Part 1: Demonstrating Applicability of the OSS-QMM in Practice w.r.t Consistency in Evaluation Results and Effectiveness in Model Derivation

In this part of the semi-structured interview with the domain experts, evidence was gathered on the practical applicability of the OSS-QMM to answer RQ1 and RQ2 (in terms of evaluation results comparability and model derivation effectiveness, respectively). It is reminded that the questions prepared are given in [197]. The Part 1 of the questionnaire consisted of six main questions. The Q1, Q2, and Q3 were aimed at comparing the results of the case studies with the opinions of the experts and at investigating the deficiencies in the field of OSS quality evaluation. The Q4 and Q5 were aimed to analyze the effectiveness of the concepts of the OSS-QMM in addressing OSS quality modeling. Finally, the Q6 was aimed to investigate the potential usefulness of a web-based tool to automate the use of the OSS-QMM that is planned to be developed in future studies. The findings obtained in this part of the validation are interpreted in Section 7.

In Q1, the experts were asked about their experiences using open-source ERP systems, and in Q1.1, their experiences with three open-source ERP systems used in the case studies. Table 6.19 (Q1) shows that 16 experts (80%) had experiences with the open-source ERP systems. It is observed that 15 of them had experience with Apache OFBiz, 13 of them with Adempiere, and 10 of them with Compiere. These experiences could be about using the ERP systems, modifying them according to further needs, or integrating them into developed systems. It should also be noted that an expert might have experience with more than one ERP system. As seen from the table, the number of experts who had experiences with Apache OFBiz is higher than the others. This supports the results of the case studies and shows that Apache OFBiz was used more than others because it better met the needs.

In Q2, the experts were asked whether there was a quality model that they employed for software quality evaluation in their companies. Here, the quality model could be a method, framework, or technique. Table 6.19 shows that all experts had experiences in software quality evaluations using quality models. Then, in Q2.1, they were asked the types of software for which they used these quality models in evaluation. It should be noted that the experts could select one or more types of software. As seen in Table 6.19, 11 of the experts used the quality models in OSS quality evaluation, 10 of them in evaluating their developed software, 7 of them in evaluating commercial software, and 5 of them in evaluating other types of software (web services, microservices etc.). Despite the fact that almost all of the experts used OSS (from the answer to Q1), nearly half of them evaluated their quality using quality models (from the answer to Q2.1). The majority of the experts, including those who used quality models, usually decided on OSS quality based on wide acceptance rate of the products or the recommendations of their colleagues. This situation has also been noted in many studies in the literature [17][41]. However, 75% of the experts who used quality models in software selection had problems with OSS selection, as seen from Table 6.18. This observation confirms the SLR studies [24][30] claiming that there is little or no adoption of OSS quality models in practice. Also, the results of the SLR study [30] indicated that OSS quality models have moved away from standardization and turned into individual models. That is, there is a diversity in their structure, leading to the proliferation of individual heterogeneous OSS quality models. This situation, in turn, has caused the standardization problem and incomparable and

unreliable evaluation results obtained from different quality models for the same purpose [39][54].

Table 6.19. The answers to Q1, Q2, and Q3 obtained in Part 1 of semi-structured interview questionnaire (questions available in [197])

Experience in ERP systems				**Experience in quality models**				**Weighting the ERP systems**	
Q1		**Q1.1**		**Q2**		**Q2.1**		**Q3**	
Yes	16	Apache OFBiz	15	Yes	20	OSS	11	Apache OFBiz	7.75
		Adempiere	13			COTS	7	Adempiere	6.36
		Compiere	10			Developed by yourself	10	Compiere	5.70
No	4	-				Other	5		
				No	-	-			

In Q3, experts were asked to assign weights on a 10-point Likert scale to the ERP systems they had experience with, taking into account the answers to Q1.1. That is, if an expert did not have any experience with an ERP system, the weights given by that expert for that ERP system were not taken into account. Table 6.19 shows that Apache OFBiz met the needs better than the other two ERP systems, according to the averages of the weights obtained from the experts. As in Q1.1, this situation supports the results of the case studies performed in Section 6.1.

Table 6.20. The answers to Q4 obtained in Part 1 of semi-structured interview questionnaire (questions available in [197])

Concepts of OSS-QMM	# of expert (hesitant)	%	5-point L. scale (mean)	Decision	Concepts of OSS-QMM	# of expert (hesitant)	%	5-point L. scale (mean)	Decision
Q. model	15 (1)	78.9	4.15	Agree	Weighting (agg: w. method)	17 (-)	85	4.4	Str. agree
Viewpoint	16 (1)	84.2	4.36	Str. agree	Measures (agg: base and derived measure, measurement function)	20 (-)	100	5.0	Str. agree
Q. requirements	20 (-)	100	5.0	Str. agree	Unit	20 (-)	100	5.0	Str. agree
Information needs	17 (-)	85	4.4	Str. agree	Scale	20 (-)	100	5.0	Str. agree
Entity	15 (1)	78.9	4.15	Agree	M. method (agg: manually, automatically)	20 (-)	100	5.0	Str. agree
Q. characteristics	20 (-)	100	5.0	Str. agree	Normalize measure	16 (1)	84.2	4.36	Str. agree
Q. sub-characteristics	20 (-)	100	5.0	Str. agree	Aggregated Measure (agg: measure aggregation method)	16 (1)	84.2	4.36	Str. agree
M. concepts	16 (-)	80	4.2	Str. agree	Measurement	20 (-)	100	5.0	Str. agree
OSS aspects	15 (1)	79.8	4.15	Agree	Measurement result	20 (-)	100	5.0	Str. agree
Code-based	15 (1)	79.8	4.15	Agree	Evaluation (agg: evaluation aggregation, manual evaluation)	14 (2)	77.7	4.1	Agree
Community-based	15 (1)	79.8	4.15	Agree	E. aggregation (agg: evaluation agg. method, evaluation agg. function)	14 (2)	77.7	4.1	Agree
Impact	14 (2)	77.7	4.1	Agree	Evaluation results	14 (2)	77.7	4.1	Agree
					Avg.	-	87.94	4.5	Str. agree

In Q4, it was aimed to analyze the extent to which the concepts used in the OSS-QMM matched the terms of the OSS quality models according to the experts' points of view. The OSS quality model used for the matching process might be a quality model that the experts used in their company; or if it was not, it might be the OSS quality model they knew from the literature. To respond this question, each expert matched the terms of their OSS quality models with the concepts of the OSS-QMM. In this matching process, the experts were asked to derive the OSS quality models from the OSS-QMM. The number of experts who matched the concepts of the OSS-QMM with the terms of the OSS quality models is given in Table 6.20 per OSS-QMM concept. Some experts were hesitant to match some terms in the OSS quality models to the concepts in the OSS-QMM. The number of hesitant experts for each concept is also given in parentheses in the second column of the table. Also, the matching percentages of the concepts (obtained without including the opinions of the hesitant experts) are given in the table. For example, if 15 experts agreed that a concept was matched and 1 expert was hesitant, then the matching percentage of that concept would be 78.9 %, as 15 out of 19 experts agreed on the match. Afterwards, as shown in the fourth column of Table 6.20, matching percentages were moved to a Likert scale of 1-5. In the moving process, 0% corresponded to 1; 25% to 2; 50% to 3; 75% to 4; and 100 % to 5 [240]. The range values were taken from the studies [240-241] in literature to interpret the 5-point Likert scale, as shown in Table 6.21. According to the values in the table, decisions were made for each concept in the OSS-QMM considering the expert opinions. As a result of the average of the answers given by the experts during the matching process, it was observed that the experts "agree" or "strongly agree" on the inclusion of the concepts in the OSS-QMM. It should be noted that the expert judgement of "agree" or "strongly agree" indicated that the judgement was at an acceptable level according to Likert scale equivalent given in Table 6.21.

Table 6.21. Interpretation of 5-point Likert scale w.r.t its ranges [240-241]

Mean score range	Likert scale equivalent
1.00 - 1.79	Strongly disagree
1.80 - 2.59	Disagree
2.60 - 3.39	Neutral
3.40 - 4.19	Agree
4.20 - 5.00	Strongly agree

In Q5, the experts were asked to what extent the terms of the OSS quality model derived from the OSS-QMM (given in Appendix-3) were compatible with the concepts of the OSS-QMM. Experts were already familiar with the matching process of the terms as they performed it in answering Q4. Also, this derived OSS quality model was explained to the experts before expressing their judgments for Q5 in order to provide a stable basis for their assessments. Then, in this question, experts were asked to rate the compatibility of the quality model with the OSS-QMM using a 5-point Likert scale. The score given by each expert can be accessed from [197]. The mean and median of the scores obtained from the 20 experts for Q5 are given in Table 6.22. The range values in Table 6.21 were used to interpret the compatibility in line with mean values of the expert opinions. As shown in Table 6.22, the experts "strongly agree" on the compatibility between terms of the quality model and the OSS-QMM.

Table 6.22. The answers to Q5 and Q6 obtained in Part-1 of the semi-structured interview questionnaire (questions available in [197])

Question	Description of Question	Mean value of expert opinion	Median value of expert opinion	Decision
Q5	Compatibility of the terms of the derived QM and the OSS-QMM	4.60	5	Strongly agree
Q6	The degree of necessity of the tool to be developed in the future	4.75	5	Strongly agree

In order to automate the usage of the OSS-QMM and thus increase the likelihood of its adoption in industry, it is planned to develop a tool that enables the derivation of standard OSS quality models in future studies. Therefore, in Q6, the experts were asked how useful such a tool would be for their company. Then, in this question, experts were asked to rate the usefulness of the tool using a 5-point Likert scale. One can access the score given by each expert from [197]. The mean and median of the scores obtained from the 20 experts for Q6 are given in Table 6.22. The range values given in Table 6.21 were used to interpret the usefulness of the tool in line with expert opinions. As shown in Table 6.22, the experts "strongly agree" that the developed tool will increase the use of the OSS-QMM and its adoption in industry.

6.2.3. Part 2: Assessment of the OSS-QMM w.r.t Its Practical Applicability

In this second part of the semi-structured interview, the experts assessed the practical applicability of the OSS-QMM considering its structure and content. During this assessment, they took into account the outputs obtained so far (i.e., from the case studies and then expert opinions (in Part 1)) in the validation process. In this regard, the implementation of the case studies and their results were presented to the experts before answering the questions in this part. The Part 2 of the questionnaire consisted of a total of 12 questions, and a list of them is given in [197]. The answer to each question was provided by the experts in a 5-point Likert scale. Before the experts answered the questions, preliminary information was given to them about the content of the questions. However, there were questions that some of the experts felt they did not had enough experience to answer. In this case, the experts did not assess the OSS-QMM and the answers of the experts for such questions were not taken into account. In this context, the number of questions, the number of experts whose answers were counted per question, the brief descriptions of the questions, the mean and median values obtained from the expert judgements, and the interpretation of the mean values by considering Table 6.21 are given in Table 6.23. The findings obtained in this part of the validation are interpreted in Section 7.

In Q1, the experts were asked to what extent the OSS-QMM was generic or abstract for deriving an existing (e.g., OSMM, QualOSS) or a new OSS quality model. That is, this question asked about the ability of the OSS-QMM to cover these quality models. To answer this question, the experts took into account an OSS quality model that they used in their companies or existed in literature. In other words, they took into account the model derivation process they carried out in Q4 of the Part-1. As seen from Table 6.23, experts "strongly agree" that the OSS-QMM was sufficiently generic or abstract. In Q2, the experts were asked about the compatibility of the concepts of the OSS-QMM as they were mapped to its 5-level structure, which is indicated by the color codes in Fig. 3. As shown in Table 6.23, experts "strongly agree" that the 5-level structure and the mapping process is compatible. In Q3, the experts were asked about the usefulness of the 3-stage structure (i.e,. specification, measurement, evaluation) shown in Fig. 3 for understanding and applying the OSS-QMM. As seen from Table 6.23, experts "strongly agree" that separating the terms by their categories is useful in OSS-QMM. In Q4, the experts were asked about the completeness of the OSS-QMM. The quality of OSS products is affected

by many variables from the source code and community side, and scattered data from different sources. The experts were asked to what extent the OSS-QMM allowed them to obtain quality models incorporating this data according to their needs. As shown in Table 6.23, experts "strongly agree" with the capability of OSS-QMM in this regard.

Table 6.23. The answers to Q1-12 obtained in Part-2 of the semi-structured interview questionnaire (questions available in [197])

Question	# of experts	Description of Question	Mean value of expert opinion	Median value of expert opinion	Decision
Q1	19	Generality of OSS-QMM	4.473	5	Str. agree
Q2	19	Compatibility of the 5-level structure with the mapping process	4.578	5	Str. agree
Q3	20	The usefulness of classification (i.e., specification, measurement, and evaluation) of QMM-OSS concepts	4.600	5	Str. agree
Q4	17	Completeness of the OSS-QMM	4.235	4	Str. agree
Q5	18	Completeness of the QM given Appendix-3	4.222	4	Str. agree
Q6	18	The homogeneity level of QMs to be derived from OSS-QMM	4.388	4.5	Str. agree
Q7	19	The degree of flexibility of the OSS-QMM in deriving QM	4.631	5	Str. agree
Q8	17	The degree of intervention provided by the OSS-QMM to stakeholders in the derivation of QMs	4.529	5	Str. agree
Q9	18	The extent to which OSS-QMM can cope with heterogeneous data of OSS products	4.222	4	Str. agree
Q10	20	Understandability of the OSS-QMM	3.700	4	Agree
Q11	20	Understandability of the derived QM given Appendix-3	3.950	4	Agree
Q12	20	The ease of deriving quality models from OSS-QMM	3.600	4	Agree
Avg.		Average of all answers	4.261	4.458	Str. agree

In Q5, based on the previous question, the experts were asked about the completeness of the OSS quality model (Appendix-3) derived from the OSS-QMM. As shown in Table 6.23, experts "strongly agree" with the completeness of the quality model. In Q6, experts were asked about the degree of homogeneity of the structure of the OSS quality models to be derived from the OSS-QMM considering the abstraction level of the OSS-QMM discussed in Q1. As seen from Table 6.23, experts "strongly agree" on the homogeneity of the quality models to be derived. In Q7, the experts were asked about the degree to which the OSS-QMM provides flexibility in the structure or content of the derived quality

models (existing or to be derived), provided that homogeneity is maintained. As shown in Table 6.23, experts "strongly agree" that the meta-model will provide flexibility in the derived models. In Q8, the experts were asked about the degree of intervention provided by the OSS-QMM to stakeholders in the derivation of the quality models. As seen from Table 6.23, experts "strongly agree" that the OSS-QMM allows stakeholders to intervene in the measurement based evaluation process in the derived quality models.

In Q9, the experts were asked to what extent the OSS-QMM could cope with heterogeneous data of OSS products. Accessing several types of heterogeneous evaluation data from the code and community side is possible. Thus, in this question, the ability of the OSS-QMM to process these complex data and perform accurate evaluations was investigated. As shown in Table 6.23, experts "agree" on the capability of OSS-QMM in this regard. In Q10, experts were asked about the understandability of the OSS-QMM, considering its structure and content. That is, it was investigated to what extent an external-party QM expert could understand the OSS-QMM without any guidance document. As seen from Table 6.23, experts "agree" on the understandability of OSS-QMM. In Q11, the experts were asked the same question as the previous one for the derived quality model in Appendix-3. The purpose was to investigate the understandability of the derived quality model to external parties. As shown in Table 6.23, experts "agree" on the understandability of the OSS-QMM. In Q12, the experts were asked about the difficulty of deriving quality models from the OSS-QMM. As seen from Table 6.23, experts "agree" on the understandability of OSS-QMM.

7. DISCUSSION

In this section, the answers to the RQs listed in Table 6.1 are discussed in light of the results obtained in Section 6. As seen from the table, two validation methods were used for each RQ. As mentioned in Section 6, the OSS-QMM should fulfill the validation conditions for the three RQs. That is, it was necessary to obtain successful results from the validation methods under each RQ. The following sub-sections are structured as follows: In Section 7.1, the results of validation methods related to RQ.1 are discussed. In this regard, it is discussed whether the results of the OSS-QMs derived from our OSS-QMM are comparable. In Section 7.2, the results of validation methods related to RQ.2 are discussed. In this regard, the effectiveness of the OSS-QMM in deriving new and existing OSS-QMs is discussed. In Section 7.3, the results of validation methods related to RQ.3 are discussed. In this regard, the results of the expert assessment of the practical applicability of the OSS-QMM are discussed. In Section 7.4, the degree of confidence in the validation process and potential threats to the validity of our thesis are discussed. As explained in the following sub-sections, successful results were obtained for each validation method, and accordingly, the OSS-QMM was validated. The results of these validation methods are grouped on the basis of RQs, and compared and discussed in the following sub-sections.

7.1. RQ.1: Are Evaluation Results of the OSS-QMs Derived from the OSS-QMM Comparable?

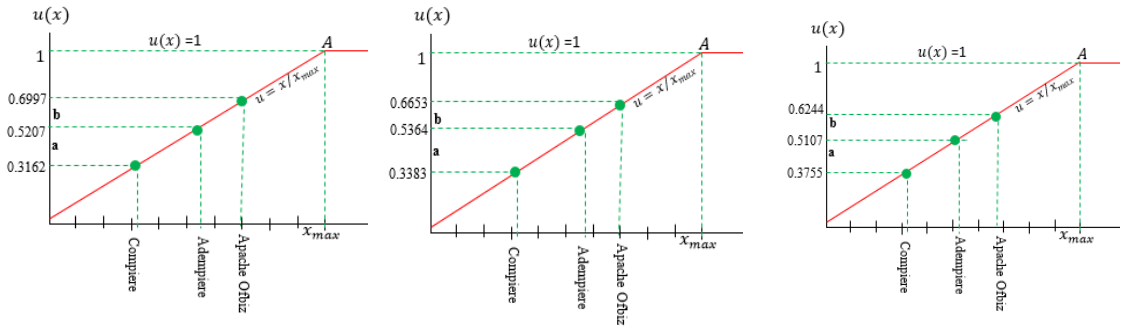
In RQ.1, it was investigated whether the results of the quality models derived from the OSS-QMM were comparable. As seen in Fig. 6.2, multiple-embedded case studies were designed as the validation method to answer this RQ. In this context, a new operationalized, and two existing OSS quality models were derived from the OSS-QMM. These quality models were applied to three real-world cases with open-source ERP systems and real data obtained from cloud repositories. As shown in Table 7.1, the evaluation results were obtained from each case study separately. The same OSS products were used in the case studies to demonstrate that the evaluation results were comparable. Although different types of measures were used in each OSS quality model and evaluations were performed in different OSS aspects, comparable results were obtained. In other saying, evaluations performed using an OSS quality model created results comparable with those from the evaluation using a different OSS quality model by

another person. Also, as shown in Fig. 7.1, the evaluation results for the case studies were shown on linear utility functions, separately. Given that a>b in all figures, aside from the comparable results, even the quality difference between Compiere and Adempiere, or between Adempiere and Apache OFBiz was similar. This further increased the consistency in the comparable results.

Table 7.1. Evaluation results obtained from case studies and expert opinions

	Case studies			**Expert option**		
Products/Results	**Case study-1 (New OSS-QM)**	**Case study-2 (OpenBRR)**	**Case study-3 (OSMM)**	**RQ1.1 (expert opinion)**	**RQ3 (expert opinion)**	**EFFORT [32]**
Apache OFBiz	0.6997	0.6653	0.6244	15	7.75	3.97
Adempiere	0.5207	0.5364	0.5107	13	6.36	2.93
Compiere	0.3162	0.3383	0.3755	10	5.70	2.83

The comparable results are essential for the standardization and reliability of the measurement. Standardization in OSS quality is of vital importance as a communication vehicle for stakeholders in identifying and selecting high-quality products [39]. As revealed in the SLR study [30], the OSS quality models have moved away from standardization and turned into individual quality models. This has highlighted the need for a comprehensive OSS-QMM. During the development of this OSS-QMM, a systematic process was followed to enable the derivation of homogeneous OSS quality models and standard measurements. In this context, a rigorous effort has been performed to eliminate inconsistencies and terminology conflicts between the vocabularies of the OSS-QMMs and the quality models, as detailed in our latest study [54]. An important step has been taken towards standardization with the OSS-QMM developed as a result of this effort, as also observed from the results of the case studies.



Case study-1 **Case study-2** **Case study-3**

Figure 7.1. Linear utility function according to the final quality score of each OSS product for case studies 1-3

As the other validation method, expert opinion was employed to support the accuracy of the results obtained from the case studies, as already shown in Table 6.1. In this context, in the first part of the semi-structured interview, questions (i.e., Q1 and Q3) were asked to the experts to confirm the results of the case studies. In this context, in Q1, the experts were asked whether they had experience with open-source ERP systems. In Q1.1, if they had experience, they were asked which open-source ERP systems they had experience with. As seen from Table 6.19, 16 out of 20 experts had experience with ERP systems. Based on the experiences of these 16 experts, the most used open Source ERP systems in practice were identified as Apache OFBiz, followed by Adempire and then Compiere. Therefore, it was concluded that Apache OFBiz, as the most used ERP system, was more useful and met the needs better than the others. These results indicated that the judgments of the experts supported the results of the case studies. Moreover, in Q3, the experts were asked to assign weights (on a 10-point Likert scale) to the ERP systems in terms of maintainability (i.e. 1-10, 10 indicating the highest degree of maintainability). In order for the judgment of the experts to be taken into account, it was required that they should have experiences with the products they weighted. As seen from Table 6.19, Apache OFBiz got the highest weight, followed by Adempire and Compiere, respectively. These weights also supported the results of the case studies. As a result, as shown in Table 7.1, the results of the case studies and the judgments of the experts indicated that the OSS-QMM enabled the derivation of OSS quality models that would produce comparable results.

Apart from the validation methods, taking into account the OSS quality models studied in our second SLR study [30], we analyzed the quality models validated by case studies.

Among these studies, we came across an OSS quality model called EFFORT [32], which evaluates open-source ERP systems. Within the ERP systems, Apache OFBiz, Adempiere and Compiere had also been evaluated in terms of a number of quality characteristics, including maintainability. The results obtained for the three open-source ERP systems in terms of maintainability using the EFFORT quality model [32] are also given in Table 7.1. According to the results, Apache OFBiz was observed to be the best product in terms of maintainability. It should be noted that the evaluations using this quality model were performed independent of the OSS-QMM (i.e. the EFFORT model had not been derived from the OSS-QMM). Nevertheless, the results obtained using this quality model were given to demonstrate the accuracy of the results obtained by our studies.

7.2. RQ.2: Is the OSS-QMM Effective for Deriving the OSS-QMs?

The RQ.2 investigated whether the developed OSS-QMM allowed the derivation of new OSS quality models and whether it fits the structure of existing OSS quality models. As seen from Table 6.1, case studies and expert opinions were employed as the validation methods for answering this RQ.

In the case study method, a multiple-embedded case design was employed, as already shown in Fig. 6.2. In this context, one new operationalized and two existing quality models were derived from the OSS-QMM. As provided in Appendix-6, the terms of these models were matched with the concepts of the OSS-QMM. Since the aim was to represent the abstraction levels in the OSS-QMM, only the determined terms belonging to the levels of the models (i.e., characteristics, sub-characteristics, measure, etc.) were used as examples in the matching process. That is, not all terms in each level of these models were used in the matching. For example, maintainability is decomposed into four sub-characteristics in the OSMM, but only one of them (i.e., integration) was used in the matching process in the appendix. This applied to any level in the matching. In addition, the concepts of the OSS-QMM listed in Table 6.5 (i.e., techniques) were not shown in the matching given in Appendix-6, since these concepts enabled to perform some calculations by using some other concepts (impact, measure, OSS aspect, etc.) matched in the appendix. For example, the concept of *weight aggregation method* was not shown in the matching, as different quality models might use different formulations or methods for this concept. As shown in Fig. 4.4, there should be no unmatched terms left in models or meta-models in the matching process. Accordingly, the OSS-QMM covered all the OSS quality models used in the case studies, and there were no terms left unmatched. As a result, it

was observed that the proposed OSS-QMM was effective to derive a new or existing OSS quality model.

In the expert opinion method, the experts with experience in the field of OSS quality and its modeling were consulted to gather more information on the effectiveness of the OSS-QMM. As the details are given in Section 6.2.1, semi-structured interviews were conducted with the domain experts. Within this scope, it was aimed to gather evidence for the generality of the OSS-QMM through some questions (i.e., Q4 and Q5) in the first part of these interviews. In Q4, the experts were asked to match the terms of the OSS-QMM with those of the OSS quality models they used in their companies or knew from the literature. This question is important to understand whether the OSS-QMM was found to be generic enough to apply in practice. In this context, as seen in the last row of Table 6.20, the average of the experts' judgements was obtained as "strongly agree" about the presence of concepts in the OSS-QMM. On the other hand, the average of the experts' judgements was obtained as "agree" for the presence of some concepts (e.g., impact and evaluation) in the OSS-QMM, as shown in the same table. Nevertheless, it should be remembered that the judgement of "agree" indicates an acceptable level according to the mean values given for the interpretation of the 5-point Likert scale in Table 6.21 [240-241].

In the semi-structured interviews with the experts, it was investigated the reasons why the experts were not "strongly agree" (i.e., they "agree") about the existence of these terms in the OSS-QMM. In this regard, as seen from Table 6.20, according to the average of the expert opinion ratings, "agree" was concluded for the existence of the "*quality model*" concept in the OSS-QMM. The experts who did not use the concept of "*quality model*" in the matching process explained the reason for this as *using a framework or method instead of using a quality model to evaluate OSS*. These experts also declared that this concept should be included in the OSS-QMM.

Moreover, these experts stated that the specifications were not formally expressed for measurement because a formal quality model was not used for OSS quality in their evaluations. In this regard, as seen in Table 6.20, according to the average value of the expert opinion ratings, "agree" was concluded for the existence of the "*entity*" and "*impact*" concepts in the OSS-QMM. More clearly, companies that developed commercial off-the-shelf (COTS) software performed measurements by following formal procedures using formal quality models to ensure product quality. However, sometimes

this was not the case with the OSS products. In other words, sometimes internal assessments were performed, ignoring formal procedures in evaluating the quality of OSS to use or integrate into another software. Also, considering that individual OSS quality models in companies were developed without depending on a comprehensive meta-model, it was likely that the specifications were not formally determined. Sometimes evaluators even chose OSS products on the recommendations of their colleagues without using quality models. Hence, it was usually OSS users who suffered from the wrong software selection, as analyzed in Table 6.18. In summary, some experts expressed that the OSS quality models they used in the matching process did not include terms related to "*entity*" and "*impact*" concepts. Nevertheless, these experts also declared that such concepts should be included in the OSS-QMM.

Some of the OSS quality models in literature or the OSS quality models that experts used in their companies enable to perform either "*code-based*" or "*community-based*" measurements. Therefore, some experts did not use both of these concepts together in the matching process, as specified in Table 6.20. That is, as seen from the table, according to the average value of the expert opinion ratings, "agree" was concluded for the existence of these concepts in the OSS-QMM. However, a comprehensive quality model is needed to deal with both code-based and community-based data simultaneously [24][30]. Based on this need, the OSS-QMM provided the opportunity to evaluate these two aspects of OSS quality, both simultaneously and separately. The experts who did not use these concepts together in the matching declared, however, that these concepts should be included in the OSS-QMM. In other words, they expressed that the quality models they used in the matching process performed either code-based or community-based evaluation. However, as seen in Table 6.20, the majority of experts used both of these concepts in the matching.

As seen in Fig. 5.10, OSS evaluation in the OSS-QMM is grouped into three main stages: *specification*, *measurement*, and *evaluation*. The concepts belonging to the *measurement* group are all about the numbers and being able to quantify the quality of OSS, whereas the concepts belonging to the *evaluation* group are all about interpreting these numbers to judge the quality of OSS. Since some experts performed result-oriented individual measurements, they did not use the concepts belonging to the *evaluation* group in the OSS-QMM after obtaining the measurement values. In other words, the quality models they used in the matching process did not include the terms regarding the *evaluation* stage.

Therefore, as shown in Table 6.20, according to the average value of the expert opinion ratings, "agree" was concluded for the existence of the concepts that reside in the *evaluation* stage of the OSS-QMM. However, the concepts belonging to the *evaluation* group are essential in terms of revealing the consistency of the measurement, so these concepts were included in the OSS-QMM. Accordingly, the experts who did not use these concepts in the matching process declared that these concepts should be included in the OSS-QMM. Also, the majority of the experts agreed on the existence of these concepts in the OSS-QMM, as shown in Table 6.20.

As a result, experts generally "strongly agree" on the generality of the OSS-QMM. Despite the experts "agree" (not "strongly agree") on some issues, this situation is considered normal considering the nature of the OSS and the general perception about the OSS as explained above. Moreover, it is satisfactory that the participants "agree" on the subject according to the mean values obtained. It is important to note that the experts used different OSS quality models in the matching process, and yet the OSS-QMM was found effective. In any case, the diversity in the structure of the OSS quality models used by the experts was the most important indicator of why some experts did not use some terms in the matching process. Notwithstanding, the experts "agree" on average about the existence of these concepts, and the 5-level structure of the OSS-QMM served well to abstract and group the concepts in related levels.

In Q5, the experts were asked about the compatibility between the terms of the derived quality model (given in Appendix-3) and the concepts of the OSS-QMM. The experts, as the external parties, performed the matching process between the derived quality model and the OSS-QMM, without any intervention. They were already familiar with the matching of the terms since they previously performed it in answering Q4. At the end of this matching, the experts indicated that they "strongly agree" on the compatibility between the terms of the derived quality model and the concepts of the developed OSS-QMM. Consequently, it was observed that the OSS-QMM developed was effective in deriving a new or existing OSS quality model, as also discussed partially in the previous paragraphs.

7.3. RQ.3: Is the OSS-QMM Applicable in Practice?

In RQ.3, it was investigated whether the developed OSS-QMM was applicable in practice. As mentioned in Table 6.1, case studies and expert opinions were employed as the validation methods in answering this RQ.

In the case study method, a multiple-embedded case design was employed, as shown in Fig. 6.2. In this context, the OSS-QMM was implemented in practice by deriving a new operationalized and two existing OSS quality models. Then, evaluations were performed to demonstrate the practical applicability of the derived quality models using real open-source ERP systems, as detailed in Section 6.1. These case studies were conducted with real data provided by open-source repositories. The evaluations were carried out using the structure and content provided by the quality models in accordance with the OSS-QMM. Therefore, the case studies indicated that the OSS-QMM could be applied in practice on real OSS products with real data. The practical application of the OSS-QMM and the results obtained are explained in detail in Section 6.1.

In the expert opinion method, the experts with experience in the field of OSS quality and its modeling were consulted to gather more information on the applicability of the OSS-QMM in practice. As the details are given in Section 6.2.2, semi-structured interviews were conducted with the domain experts within this scope. In this context, 12 questions were asked to the experts in the second part of the interviews. The experts were expected to answer these questions considering the practical applicability of the OSS-QMM. In this context, first of all, the practical application of the OSS-QMM in the case studies and the information needed for applying the OSS-QMM were explained to the experts. Furthermore, the experts observed the practical applicability of the OSS-QMM by deriving their selected OSS quality models from the OSS-QMM in Part 1 of the semi-structured interviews. In this regard, in Part 2 of the semi-structured interviews, the experts were expected to assess the applicability of the OSS-QMM by considering the validation process performed so far in order to answer RQ3.

As seen in the last row of Table 6.23, based on the average value of the experts' judgements, it was observed that the experts "strongly agree" on the application of the OSS-QMM in practice. One of the most important reasons underlying this is that the OSS-QMM has been developed by following a systematic process. In this context, meta-models and OSS quality models were investigated in detail by conducting SLR studies.

This way, deficiencies in OSS meta-models and quality models were revealed, and effort was spent to eliminate these deficiencies. Furthermore, during the development process of the OSS-QMM, we studied with domain experts and obtained feedback from them on the improvement of the OSS-QMM. In this context, subject matter experts were consulted during the development and validation processes, as shown in Fig. 1.2. That is, the proposed OSS-QMM was developed by consulting experts from both industry and academia, following an iterative review-and-revise process. Thus, the development of the OSS-QMM has been based on solid foundations, and the OSS-QMM has been improved with continuous feedback until the final version has been obtained.

As a result, subject matter experts validated the practical applicability of the OSS-QMM in the assessments that they performed in Part 2 of the semi-structured interview. In this regard, the systematic and iterative development process mentioned above was an important factor for the experts to "strongly agree" on the validity of the OSS-QMM for practical applicability. It should be noted that the experts consulted at the development stage of the OSS-QMM and the experts consulted at the validation stage were different persons. Due to the nature of the OSS, its evaluation includes a variety of heterogeneous data from both code-based and community-based aspects, making the evaluation a challenging process [5]. At this point, considering the judgements of the experts, it was important that the experts reached the conclusion that the OSS-QMM was complete and it could deal with the heterogeneous data.

As shown in Table 6.23, however, the experts "agree" (i.e., not "strongly agree") on the understandability of the developed OSS-QMM and the new OSS quality model derived from the OSS-QMM (and given in Appendix-3). Also, they "agree" on the difficulty of deriving a new OSS quality model from the OSS-QMM. Considering the common opinions of the experts, the most important reason for this was that it would be difficult for the employees who did not have software quality modeling experience to understand the OSS-QMM and derive an OSS quality model from it. The experts stated that they made these inferences based on the impressions that they obtained in their companies.

The experts also stated that they would agree on the understandability of the OSS-QMM if there was a guarantee that only people with experience in this field would use it. They also indicated that this situation would not change the fact that the applicability of the OSS-QMM in practice was at a good level. Therefore, given the understandability of the OSS-QMM to non-experts in this field, the next goal can be developing a web-based,

open-source tool that automates the use of the OSS-QMM for deriving OSS quality models. In addition, it is planned to develop a guidance document for using the OSS-QMM to instantiate the OSS quality models. This way, it is planned that, even if employees do not have sufficient software quality modeling experience, they will be able to develop their OSS quality models and perform quality evaluations according to their needs.

7.4. Confidence in Validity and Potential Threats

In this section, firstly, all the results obtained throughout the validation process are provided collectively, along with the degrees of confidence they provide for validation. Then, we discuss potential threats to the validity of our study and the actions taken to mitigate these threats.

The degrees of confidence in the validity of the OSS-QMM, based on the empirical evaluation results for each RQ, are given in Table 7.2. As seen from the table, case studies were conducted to answer the three RQs asked for validating the OSS-QMM. In this context, the confidence degree for having comparable results was determined as "very high" (for RQ1). In the case studies, a new OSS quality model and two existing OSS quality models were derived from the OSS-QMM (as given in Appendix-6). Considering the matched concepts and the compatibility of the OSS-QMM with the derived OSS quality models, the degree of confidence in validating the effectiveness of the OSS-QMM in model derivation was determined as "very high" (for RQ2). Moreover, the successful application of these derived OSS quality models in practice showed that the confidence degree in validating the applicability of the OSS-QMM in practice was "very high", as shown in Table 7.2 (regarding RQ3).

As the other validation method, the Part 1 of the expert opinion study was used to validate RQ1 and RQ2. In this context, first, the experts evaluated the same OSS products (i.e., the three ERP systems) used in the case studies in order to check the accuracy of the results obtained from the case studies. According to the opinions of the experts, the degree of confidence in obtaining comparable results from the OSS quality models derived from the OSS-QMM was determined as "very high" (for RQ1). In order to answer RQ2 (via Part 1 of the expert opinion study), each expert derived an OSS quality model of his/her concern from the OSS-QMM, and then performed a mapping between the terms of this OSS quality model and the concepts of the OSS-QMM. Considering the values obtained

as a result of these mappings by the experts (given in Table 6.20), the confidence degree in validating the effectiveness of the OSS-QMM in deriving new OSS quality models was determined as "very high", as shown in Table 7.2. Since the applicability of the OSS-QMM was not validated in Part 1 of the expert opinion study (for RQ3), it is not specified (labelled as "*not applicable*") in Table 7.2.

Even though Part 2 of the expert opinion study was mainly designed to validate RQ3, it partially contained questions about validating RQ2. The degree of confidence in validating the effectiveness of the OSS-QMM in deriving new OSS quality models was determined as "very high" in the case studies and also Part 1 of the expert opinion study. However, it was determined as "high" in Part 2 of the expert opinion study since the understandability of the OSS-QMM was not at a desired level according to the expert opinions (regarding RQ2). Moreover, considering the average values obtained as a result of the expert opinions (as given in Table 6.23), the degree of confidence in validating the applicability of the OSS-QMM in practice was determined as "very high" (for RQ3), as specified in Table 7.2.

Table 7.2. Degree of confidence on the validity of the OSS-QMM with respect to empirical evaluation results

RQ# – Motivation	**Case studies**	**Expert Opinion – Part 1**	**Expert Opinion – Part 2**
RQ.1 – Comparability of results by OSS-QMs	**Very High** (from Fig. 7.1)	**Very High** (from Table 6.19)	*Not applicable*
RQ.2 – Effectiveness of OSS-QMM	**Very High** (from Appendix -6)	**Very High** (from Table 6.20, Table 6.22)	*Partially applicable* **High** (from Table 6.23)
RQ.3 – Applicability of OSS-QMM in practice	**Very High** (from Table 6.88-6.17)	*Not applicable*	**Very High** (from Table 6.23)

Potential threats to the validity of this study can be analyzed in four main categories, which are *internal*, *external*, *construct*, and *conclusion* validity, as adapted from Wohlin et al. [242]. Although a systematic process was followed to develop and validate the OSS-QMM, some potential threats might have arisen regarding validity.

Researcher bias can be considered as the main *internal* threat to the validity of this study. Researchers of this study have played an important role while determining the first set of concepts and an initial version of the relationships between these concepts. There might be concepts that were overlooked or not used. To mitigate this threat, however, a laborious

effort was spent. In this context, firstly, two separate SLR studies [30][37] were conducted to investigate the SQMMs and the OSS quality models, respectively. The results of these SLR studies guided the researchers in determining the concepts of the OSS-QMM, details of which are given in Section 5.1.1. Nevertheless, the concepts identified might still have contained subjectivity. Therefore, an iterative review-and-revise process was followed with subject matter experts in order to validate the identified concepts and their relationships, as detailed in Section 5.2.1 (Step 5), using the guidelines by Kläs et al. [195] and Tanrıöver et al. [194]. Moreover, the OSS-QMM was validated in a real-world setting (in Step 6), and the concepts and their relationships in the OSS-QMM were revised in case of any opportunities observed during the validation process.

Throughout the validation process, experts were asked questions for various purposes using surveys. The low quality or understandability of these questions included in these surveys could be considered as the main threat to the *construct* validity in this study. To mitigate this threat, firstly, in the SLR study [37], we analyzed the problems in proposing meta-models and prepared questions considering to address these problems. In addition, studies on the validation of meta-models (e.g., [195]) were examined, and questions were prepared in line with this purpose in order to increase the quality of the questions. To mitigate the threat of not understanding the questions by experts, online or face-to-face semi-structured interviews were held, and necessary information was provided to the experts on the points that were not very clear. In addition, during these interviews, it was ensured that the experts answered each question carefully, and also, the questions matured with the advice of the experts.

The quality of the RQs (given in Table 6.1) underlying the validation aspects and the validation methods were important factors affecting the *conclusion* validity of the study. To mitigate this threat, the fundamentals and practical uses of the SQMMs were well explored through the SLR study [37] in order to ensure the quality of the RQs. In this way, RQs were created to validate the intended uses of the OSS-QMM. In addition, validation methods common in literature for validating meta-models were determined [37]. In synthesizing the results obtained, the methods provided in the literature (given in Table 6.21) were used without the intervention of the authors of this study.

External validity deals with the generalization of the findings of our study. Our main concern is to develop a quality meta-model for OSS. Therefore, the OSS-QMM developed in this study cannot be generalized beyond this context, and thus, does not

guarantee accurate evaluation for other types of software (i.e., COTS). Moreover, external validity can be strengthened by deriving other OSS quality models and using them in further real contexts by interested researchers and/or practitioners, which might also provide opportunities for improving the OSS-QMM in future studies.

8. CONCLUSION

The motivation for this thesis has been the lack of meta-models for OSS quality and the inconsistent terminology among the existing general-purpose SQMMs. Aside from these, existing OSS-QMs produce incomparable and unreliable evaluation results. This is because the dynamic and diverse nature of OSS has caused the OSS-QMs to be heterogeneous in terms of structure and content. Therefore, there is little or no adoption of existing OSS-QMs in practice. In this context, in this thesis, a comprehensive OSS-QMM has been proposed to enable the derivation of OSS-QMs having homogenous structure and terms, contribute to the standardization of OSS quality evaluation, and increase the adoption of OSS-QMs in practice.

For this purpose, a systematic and laborious effort has been spent via the step-based meta-model creation process, including review-and-revise iterations. In this context, in Step-1, meta-models have been examined in detail by performing an SLR study [37], and thus deficiencies have been discovered in this domain, and the current status of the meta-models for OSS quality has been analyzed. Then, in Step-2, another SLR study [30] has been performed to analyze the OSS quality models. Considering the output of this SLR, the common structure of the tailored quality models which were proposed for OSS and that of the basic quality models which provide partial evaluation for OSS have been analyzed. Consequently, it has been observed that these quality models have a common structure consisting of five levels. Then, in Step-3, inconsistencies and terminology conflicts between international standards and proposals have been identified and analyzed since these standards or proposals are the basis for the SQMMs. Then, the terminologies of the SQMMs have been analyzed, and how inconsistencies and terminology conflicts in standards or proposals are reflected in the SQMMs have been discussed. It has been observed that the SQMMs cover more terms from ISO/IEC 15939 than the others since other sources identify concepts for only certain application domains and purposes. The synonyms of the terms in different SQMMs have been listed, and the terms have been categorized according to the most used ones among their synonyms. The aggregations of the terms under each category have also been listed. Consequently, it has been observed that 38 cases of synonymity exist for 15 terms in the SQMMs, and this situation has confirmed that there are inconsistencies among the terms of different SQMMs. Next, the terms at each level of the OSS quality models and the terms of the SQMMs in each category have been matched since quality models are assumed to be the instances of the

SQMMs. As a result of all these processes, the infrastructure for developing consistent meta-models of OSS quality has been established.

Then, in Step-4, a comprehensive OSS-QMM has been proposed by following an iterative process to refine the OSS-QMM. Therefore, in summary, the main achievements of this OSS-QMM are to;

- eliminate the inconsistency in terminologies of the SQMMs,
- make a matching between the concepts of the SQMMs and the terms of the OSS-QMs,
- enable common understanding among stakeholders,
- enable the derivation of OSS-QMs having homogenous structure and terms,
- represent concepts of OSS quality more formally,
- provide the opportunity for deriving new or existing OSS-QMs that enable comparable measurements,
- contribute to the standardization of OSS quality evaluation, which in turn will provide an important communication vehicle to companies in interoperating, and
- provide the opportunity to increase adoption of OSS-QMs in practice.

Finally, in Step-5, the OSS-QMM developed initially has been validated by using multi-faced methods and to obtain the final version of the OSS-QMM by review-and-revise process applied during this validation. We have considered the outputs of the previous SLR studies that analyzed meta-models in determining the validation techniques and, accordingly, determined the most used methods to apply in validating the OSS-QMM. For this purpose, three research questions (RQs) have been determined to investigate the validity of the OSS-QMM by using case study and expert opinion methods. Each RQ has aimed at validating the OSS-QMM from different aspects such as results comparability, effectiveness, and applicability.

First, three case studies have been conducted to validate the following issues;

- A new OSS quality model and two existing OSS quality models have been derived from the OSS-QMM.

- These derived OSS quality models have been applied in practice to evaluate real OSS products.
- Evaluation results obtained as a result of the practical applications of these OSS quality models have been found as comparable.

Then, semi-structured interviews have been held with 20 domain experts. Since the main application area of the OSS-QMM is the software industry, the opinions of software quality experts from the industry are very important for the validity of the OSS-QMM. To obtain relevant feedback, a questionnaire consisting of two parts has been prepared prior to conducting the interviews. It has been aimed to gather evidence about the applicability of the OSS-QMM with respect to evaluation results comparability and model derivation effectiveness in Part 1; and to collect evidence for the practical applicability of the OSS-QMM considering its structure and content in Part 2.

The feedback from the domain experts have mainly validated the following issues;

- Evaluation results from different OSS quality models derived from the OSS-QMM have been found as comparable.
- The concepts of the OSS-QMM have matched well with the terms of the OSS quality models used in practice.
- The generality and completeness of the OSS-QMM have been found as sufficient.
- The structure of the OSS-QMM (i.e., the 5-level structure, classification of its concepts and their relationships) has found as appropriate.
- The OSS-QMM has provided flexibility to the users in deriving the OSS quality models and also enabled the OSS quality models to be homogeneous.

The results of multi-faceted empirical studies have indicated that the OSS-QMM addressed solving problems in the OSS quality evaluation and its adoption with high degrees of confidence. Nevertheless, it cannot be claimed that the effort spent in this thesis and the proposed OSS-QMM will solve all the problems related to consistency and harmonization in a way that will be accepted by all parties in the OSS community.

Still, it can serve as a guide for;

- OSS quality specification and evaluation by forming the basis of discussions to solve the problems with standardization,

- evaluators who want to develop a new OSS-QM,
- evaluators who are confused about the heterogeneity of existing OSS-QMs,
- evaluators who are confused by inconsistent terminology in the existing SQMMs or international standards,
- for meta-model developers who will propose new SQMMs or integrate consistent concepts into the developed SQMMs.

REFERENCES

[1] Androutsellis-Theotokis, S., Spinellis, D., Kechagia, M. and Gousios, G.: Open source software: A survey from 10,000 feet. Foundations and Trends in Technology, Information and Operations Management, 4(3-4), pp.187-347. (2011).

[2] Miguel, J.P., Mauricio, D. and Rodríguez, G.: A review of software quality models for the evaluation of software products. In: arXiv preprint arXiv:1412.2977, (2014).

[3] Miloudi, C., Cheikhi, L., Abran, A., and Idri, A.: Maintenance effort estimation for open source software: Current trends, The 31st International Workshop on Software Measurement (IWSM), (2022).

[4] Tassone, J., Xu, S., Wang, C., Chen, J. and Du, W.: Quality Assessment of Open Source Software: A Review. IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS) (pp. 411-416). IEEE. (2018).

[5] Adewumi, A., Misra S. and Omoragbe, N.: FOSSES: Framework for open-source software evaluation and selection. In: Software: Practice and Experience 49.5, 780-812, (2019).

[6] Rossi, B., Russo, B. and Succi, G.: Adoption of free/libre open source software in public organizations: factors of impact. Information Technology & People. (2012).

[7] Midha, V. and Palvia, P.: Factors affecting the success of Open Source Software. Journal of Systems and Software, 85(4), pp.895-905. (2012).

[8] Wang, Q., Zhang, W., Jiang, J. and Li, L.: A Reliability Automatic Assessment Framework of Open Source Software Based on JIRA. In: Proceedings of the 2020 9th International Conference on Software and Computer Applications, (2020).

[9] Jean-Christophe D., and Alexandre, S.: Comparing assessment methodologies for free/open source software: OpenBRR and QSOS. In: International Conference on Product Focused Software Process Improvement. Springer, Berlin, Heidelberg, (2008).

[10] Al-Qutaish, R. E.: Quality models in software engineering literature: an analytical and comparative study. In: Journal of American Science. 6(3): 166-75, (2010).

[11] Laporte, C. Y., and April, A.: Software quality assurance, John Wiley and Sons, (2018).

[12] O'Regan, G.: Introduction to software quality. Springer, (2014).

[13] IEEE Std. 610.12: Standard Glossary of Software Engineering Terminology. The Institute of Electrical and Electronics Engineers, New York, NY, USA, (1990).

[14] ISO. ISO/IEC 14598-1: Software product evaluation - Part 1: General overview. International Organization for Standardization, Geneva, Switzerland, (1999).

[15] Khatri, S.K. and Singh, I.: Evaluation of open source software and improving its quality. In: 5th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO). IEEE, (2016).

[16] Azadeh A., Nazari-Shirkouhi S., Samadi H., Nazari-Shirkouhi A.: An integrated fuzzy group decision making approach for evaluation and selection of best simulation software packages. Int J Ind Syst Eng., 18(2):256-282, (2014).

[17] Hauge, O., Osterlie, T. and Sorensen, C.F.: An empirical study on selection of Open Source Software-Preliminary results. In: ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. IEEE, (2009).

[18] Parker K. R.: Selecting software tools for IS/IT curricula. Educ Inf Technol., 15(4):255-275, (2010).

[19] Sung, W.J, Kim, J.H. and Rhew, S.Y.: A quality model for open source software selection. In: Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007). IEEE, (2007).

[20] Petrinja, E., Sillitti, A. and Succi. G.: Comparing OpenBRR, QSOS, and OMM assessment models. In: IFIP International Conference on Open Source Systems. Springer, Berlin, Heidelberg, (2010).

[21] ISO/IEC TR 9126, Software engineering–Product quality – Part 1,2,3. 2002-03-15.

[22] Boehm, B. W., Brown, J. R., Lipow, M.: Quantitative evaluation of software quality. In: Proceedings of the 2nd Int Conf on Software Eng, Oct 13, pp. 592-605, IEEE, 1976.

[23] McCall, J.A., Richards, P. K., Walters, G.F.: Factors in Software Quality. Volume I. Concepts and Definitions of Software Quality, Fort Belvoir, VA: Defense Tech Info Center, 1977.

[24] Adewumi, A., Misra S. and Omoragbe, N., Crawford, B. and Soto, R.: A systematic literature review of open source software quality assessment models. In: SpringerPlus 5.1, 1936, (2016).

[25] Samoladas, I., Gousios G., Spinellis, D. and Stamelos, I.: The SQO-OSS quality model: measurement based open source software evaluation. In: IFIP international conference on open source systems. Springer, Boston, MA, (2008).

[26] Adewumi, A., Misra, S. and Omoregbe, N.: A review of models for evaluating quality in open source software." IERI Procedia 4, 88-92, (2013).

[27] Duijnhouwer, F.W., Widdows, C.: Capgemini Expert Letter Open Source Maturity Model, Capgemini, url: tinyurl.com/yxdbvjk6, (2003).

[28] Wasserman, M.P., Chan, C.: Business Readiness Rating Project, BRR Whitepaper RFC 1, url: tinyurl.com/y5srd5sq, (2006).

[29] Lenarduzzi, V., Taibi, D., Tosi, D., Lavazza, L and Morasca, S.: Open Source Software Evaluation, Selection, and Adoption: a Systematic Literature Review. In: 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, (2020).

[30] Yılmaz, N., and Tarhan, A.K.: Quality evaluation models or frameworks for open source software: A systematic literature review. J Softw Evol Proc. 2022;34(6):e2458. doi:10.1002/smr.2458, (2022).

[31] Stol, K.J., and Babar, M.A.: Challenges in using open source software in product development: a review of the literature. In: Proceedings of the 3rd international workshop on emerging trends in free/libre/open source software research and development, (2010).

[32] Aversano, L., and Tortorella, M.: Quality evaluation of floss projects: Application to ERP systems. In: Information and Software Technology 55.7, 1260-1276, (2013).

[33] Ciolkowski, M., and Soto, M.: Towards a comprehensive approach for assessing open source projects. In Software Process and Product Measurement (pp. 316-330). Springer, Berlin, Heidelberg. (2008)

[34] Wagner, S., Goeb, A., Heinemann, L., Kläs, M., Lampasona, C., Lochmann, K., Mayr, A., Plösch, R., Seidl, A., Streit, J. and Trendowicz, A.: Operationalised product quality models and assessment: The Quamoco approach. Information and Software Technology, 62, pp.101-123. (2015).

[35] Stefan Wagner, Klaus Lochmann, Sebastian Winter, Andreas Goeb, Michael Klaes, Quality models in practice: a preliminary analysis, in: Proc. 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM'09), IEEE, 2009, pp. 464–467.

[36] Stefan Wagner, Klaus Lochmann, Sebastian Winter, Andreas Goeb, Michael Kläs, Sabine Nunnenmacher, Software quality in practice. survey results, Technical Report TUM-I129, Technische Universität München, 2012.

[37] Yılmaz, N., and Tarhan, A.K.: Meta-models for Software Quality and Its Evaluation: A Systematic Literature Review. In: International Workshop on Software Measurement and the 15th International Conference on Software Process and Product Measurement, Mexico, (2020).

[38] Nistala, P., Nori, K.V. and Reddy, R.: Software quality models: A systematic mapping study. In 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP) (pp. 125-134). IEEE. (2019).

[39] Garcia, F., Bertoa, M.F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M. and Genero, M.: Towards a consistent terminology for software measurement. Information and Software Technology, 48(8), pp.631-644. (2006).

[40] Ramamoorthy, C.V., Prakash, A., Tsai, W.T. and Usuda, Y.: Software engineering: Problems and perspectives. Computer, 17(10), pp.191-209. (1984).

[41] Li, J., Conradi, R., Bunse, C., Torchiano, M., Slyngstad, O.P.N. and Morisio, M.: Development with off-the-shelf components: 10 facts. IEEE software, 26(2), pp.80-87, (2009).

[42] Kleppe, A., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture Practice and Promise. Addison Wesley (2003)

[43] OMG: Model Driven Architecture (MDA). Object Management Group. (2001), OMG document ormsc/2001-07-01.

[44] Garcia, F., Serrano, M., Cruz-Lemus, J., Ruiz, F., Piattini, M. and ALARCOS Research Group.: Managing software process measurement: A metamodel-based approach. Information Sciences, 177(12), pp.2570-2586, (2007).

[45] Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J. J., Pavon, J., and Gonzalez-Perez, C.: FAML: a generic metamodel for MAS development. IEEE Transactions on Software Engineering, 35(6), 841-863, (2009).

[46] Othman, S. H., Beydoun, G., and Sugumaran, V.: Development and validation of a Disaster Management Metamodel (DMM). Information Processing & Management, 50(2), 235-271, (2014).

[47] Othman, S. H., and Beydoun, G.: Metamodelling approach to support disaster management knowledge sharing, In: 21st Australasian Conference on Information Systems, (2010).

[48] Al-Dhaqm, A., Razak, S., Othman, S. H., Ngadi, A., Ahmed, M. N., and Ali Mohammed, A.: Development and validation of a database forensic metamodel (DBFM). PloS one, 12(2), e0170793, (2017).

[49] Object Management Group (OMG), Meta Object Facility (MOF). Core Specification Version 2.5.1. October 2019. https://www.omg.org/spec/MOF/2.5.1/PDF

[50] Henderson-Sellers, B., and Bulthuis, A.: COMMA: Sample metamodels. JOOP, 9(7), 44-48, (1996)

[51] Rossi, M., Ramesh, B., Lyytinen, K., & Tolvanen, J. P.: Managing evolutionary method engineering by method rationale. Journal of the association for information systems, 5(9), 12, (2004).

[52] Karagiannis, D. and Kühn, H.: Metamodelling platforms. In EC-Web (Vol. 2455, p. 182). (2002).

[53] Othman, S. H., and Beydoun, G.: A disaster management metamodel (DMM) validated. In Knowledge Management and Acquisition for Smart Systems and Services: 11th International Workshop, PKAW 2010, Daegue, Korea, August 20-September 3, Proceedings 11 (pp. 111-125). Springer Berlin Heidelberg, (2010).

[54] Yılmaz, N., and Tarhan, A. K.: Matching terms of quality models and meta-models: toward a unified meta-model of OSS quality. Software Quality Journal, 1-53, (2022).

[55] The Open Source Definition | Open Source Initiative. URL: https://opensource.org/docs/osd, [Las visited: 11-Mayıs-2023].

[56] Özdaş, M. R.: Kamuda Açık Kaynak Kodlu Yazılım Kullanımı," no. T.C kalknma bakanlığı bilgi toplum dairesi, 2012.

[57] Yılmaz, N., and Tarhan, A.K.: A two-dimensional method for evaluating maintainability and reliability of open source software. Journal of the Faculty of Engineering and Architecture of Gazi University, 34(4), (2019).

[58] Çavuş, M. F., and Kurt, H. S.: Kamu kurumlarında açık kaynak kodlu yazılımların kullanımı. Siyaset, Ekonomi ve Yönetim Araştırmaları Dergisi, 5(3), 39-49, (2017).

[59] Black Duck Software Composition Alanysis (SCA), URL: https://www.synopsys.com/, [Las visited: 11-Mayıs-2023].

[60] Yılmaz, N., and Dinçer, K.: Açık Kaynak Yazılım Seçimi için İki Boyutlu Değerlendirme Metodu, UYMS, (2016).

[61] GitHub's January Report, URL: https://octoverse.github.com/, [Las visited: 11-Mayıs-2023].

[62] The 2022 state of Open Source Report, URL: https://www.openlogic.com/resources/2022-open-source-report, [Las visited: 11-Mayıs-2023].

[63] OpenForum Europe (ofe), URL: Open Source Study - OpenForum Europe, [Las visited: 11-Mayıs-2023].

[64] Open Source Software Stratagey (European Commission), URL: https://commission.europa.eu/about-european-commission/departments-and-

executive-agencies/informatics/open-source-software-strategy_en, [Las visited: 11-Mayıs-2023].

[65] Open source, open standards and re-use: government action plan - GOV.UK. URL: https://www.gov.uk/government/publications/open-source-open-standards-and-re-usegovernment-action-plan, [Las visited: 11-Mayıs-2023].

[66] Aiven and the Open Source Community, URL: https://aiven.io/open-source, [Las visited: 11-Mayıs-2023].

[67] Open Access Government, URL: https://www.openaccessgovernment.org/why-the-government-is-backing-open-source-software/140839/, [Las visited: 11-Mayıs-2023].

[68] Open Source Software Country Intelligence Report (Netherlands), URL: https://joinup.ec.europa.eu/sites/default/files/inline-files/OSS%20Country%20Intelligence%20Report_NL.pdf, [Las visited: 11-Mayıs-2023].

[69] The Malaysian Public Sector Open Source Software Master Plan, URL: https://digitallibrary.un.org/record/594324, [Las visited: 11-Mayıs-2023].

[70] Jusoh, Y., Chamili, K., Yahaya, J. H., and Pa, N. C.: The selection criteria of open source software adoption in Malaysia. International Journal of Advancements in Computing Technology, 4(21), 278-287, (2012).

[71] Technology Research | Gartner Inc. URL: http://www.gartner.com/technology/home.jsp, [Las visited: 11-Mayıs-2023].

[72] Pardus. URL: http://www.pardus.org.tr/, [Las visited: 11-Mayıs-2023].

[73] Zhao, Y., Liang, R., Chen, X., & Zou, J.: Evaluation indicators for open-source software: a review. Cybersecurity, 4(1), 1-24, (2021).

[74] Gezici, B., Özdemir, N., Yılmaz, N., Coşkun, E., Tarhan, A., and Chouseinoglou, O.: Quality and success in open source software: A systematic mapping. In 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 363-370). IEEE, (2019).

[75] The State of Enterprise Open Source: A Red Hat report, URL: https://www.redhat.com/en/resources/state-of-enterprise-open-source-report-2022, [Las visited: 11-Mayıs-2023].

[76] North Bridge. URL: Available: http://www.northbridge.com/, [Las visited: 11-Mayıs-2023].

[77] Ellram, L. M.: Total cost of ownership. Handbuch Industrielles Beschaffungsmanagement: Internationale Konzepte—Innovative Instrumente—Aktuelle Praxisbeispiele, 659-671, (2002).

[78] CyberSource. URL: Linux vs. Microsoft TCO Comparison (lwn.net), [Las visited: 11-Mayıs-2023].

[79] Thomas, S. L., and Francillon, A.: Backdoors: Definition, deniability and detection. In Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21 (pp. 92-113). Springer International Publishing, (2018).

[80] Schuster, F., and Holz, T.: Towards reducing the attack surface of software backdoors. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 851-862), (2013).

[81] Rosen, L.: Open source licensing. Software Freedom and Intellectual Property Law, (2005).

[82] Lerner, J., and Tirole, J.: The scope of open source licensing. Journal of Law, Economics, and Organization, 21(1), 20-56, (2005).

[83] Suman, M.W. and Rohtak, M.D.U.: A comparative study of software quality models. International Journal of Computer Science and Information Technologies, 5(4), pp.5634-5638. (2014).

[84] Sadeghzadeh H. M., and Rashidi, H.: Software quality models: A comprehensive review and analysis. Journal of Electrical and Computer Engineering Innovations (JECEI), 6(1), pp.59-76. (2017).

[85] Grady, R. B.: *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, Englewood Cliffs, NJ, USA, (1992).

[86] Bertoa, M., and Vallecillo A.: Quality Attributes for COTS Components*," I+D Computación*, Vol 1, Nro 2, 128-144, (2002).

[87] Georgiadoui, E.: GEQUAMO-A Generic, Multilayered, Customizable Software Quality model," *Software Quality Journal*, 11, 4, 313-323. DOI=10.1023/A:1025817312035, (2003).

[88] Rawashdeh, A., and Matalka, B.: A New Software Quality Model for Evaluating COTS Components," *Journal of Computer Science* 2 (4): 373-381, (2006).

[89] Orijin, A.: Method for qualification and selection of open source software (QSOS) version 2.0, url: *http://www.qsos.org/*, (2006).

[90] Taibi, D., Lavazza, L., and Morasca, S..: OpenBQR: A framework for the assessment of OSS. In: IFIP International Conference on Open Source Systems. Springer, Boston, MA, (2007).

[91] Alfonzo, O., Domínguez, K., Rivas, L., Perez, M., Mendoza, L., and Ortega, M.: Quality measurement model for analysis and design tools based on FLOSS. In:

19th Australian conference on software engineering, Perth, Australia,26–28, (2008)

[92] Raffoul, E., Domínguez, K., Perez, M., Mendoza, L. E., and Griman, A. C.: Quality model for the selection of FLOSS-based Issue tracking system. In: Proceedings of the IASTED international conference on software engineering, Innsbruck, Austria, (2008)

[93] Soto, M., and Ciolkowski, M.: The QualOSS open source assessment model measuring the performance of open source communities. In: Proceedings of the 3rd international symposium on empirical software engineering and measurement, (2009).

[94] Alvaro, A., Almeida, E. S., and Meira, S. R. L.: A Software Component Quality Framework," *ACM SIGSOFT SEN 35, 1*, 1-4, (2010).

[95] Upadhyay, N., Despande, B. M., and Agrawal, V. P.: Towards a software component quality model. In *International Conference on Computer Science and Information Technology* (pp. 398-412). Springer, Berlin, Heidelberg, (2011).

[96] Raza, A., Capretz, L. F., and Ahmed, F.: An open source usability maturity model (OS-UMM). In: Computers in Human Behavior 28.4, 1109-1121, (2012).

[97] Samarthyam, G., Suryanarayana, G., Sharma, T., and Gupta, S.: MIDAS: A Design Quality Assessment Method for Industrial Software, *Software Engineering in Practice*, San Francisco, CA, USA, pp 911-920, (2013).

[98] Sarrab, M., and Rehman, O. M. H.: Empirical study of open source software selection for adoption, based on software quality characteristics. In: Advances in Engineering Software 69, 1-11, (2014).

[99] Kuwata, Y., Takeda, K., and Miura, H.: A study on maturity model of open source software community to estimate the quality of products. In: Procedia Computer Science 35, 1711-1717, (2014).

[100] Sohn H., Lee M.G., Seong B.M., Kim J.B.: Quality evaluation criteria based on open source mobile HTML5 UI framework for development of cross-platform. In: International Journal of Software Engineering and Its Applications 9.6, 1-12, (2015).

[101] Wasserman, A. I., Guo, X., McMillian, B., Qian, K., Wei, M. Y., and Xu, Q.: OSSpal: finding and evaluating open source software. In: IFIP International Conference on Open Source Systems. Springer, Cham, (2017).

[102] Haaland, K., Groven, A.K. and Regnesentral, N.: Free/libre open source quality models-a comparison between two approaches. In: 4th FLOS International Workshop on Free/Libre/Open Source Software, (2010).

[103] Russo, B. and Succi, G.: A cost model of open source software adoption. International Journal of Open Source Software and Processes (IJOSSP), 1(3), pp.60-82, (2009).

[104] Silic, M. and Back, A.: The influence of risk factors in decision-making process for open source software adoption. International Journal of Information Technology & Decision Making, 15(01), pp.151-185, (2016).

[105] Heili, J. and Assar, S.: An empirical enquiry into the adoption of Open Source Software by individual users in France. In IADIS 2009: International Conference Information Systems, February 25-27, Barcelona, Spain, (2009).

[106] Cotugno, F.R. and Messina, A.: Adapting scrum to the Italian army: methods and (open) tools. In IFIP International Conference on Open Source Systems (pp. 61-69). Springer, Berlin, Heidelberg, (2014).

[107] Del Bianco, V., Lavazza, L., Morasca, S., and Taibi, D.: Quality of open source software: The QualiPSo Trustworthiness Model. In: IFIP International Conference on Open Source Systems. Springer, Berlin, Heidelberg, (2009).

[108] Eghan, E. E., Alqahtani, S.S., Forbes, C. and Rilling, J.: API trustworthiness: an ontological approach for software library adoption. In: Software Quality Journal 27.3, 969-1014, (2019).

[109] Mens, T., Doctors, L., Habra, N., Vanderose, B. and Kamseu, F.: Qualgen: Modeling and analysing the quality of evolving software systems. In: 15th European Conference on Software Maintenance and Reengineering. IEEE, (2011).

[110] Hmood, A., Keivanloo, I., and Rilling, J.: SE-EQUAM-an evolvable quality metamodel. In 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops (pp. 334-339), IEEE, (2012).

[111] Vanderose, B., Habra, N., & Kamseu, F. (2010). Towards a model-centric quality assessment. In Proceedings of the 20th International Workshop on Software Measurement (IWSM 2010): Conference on Software Process and Product Measurement (Stuttgart Nov 2010).

[112] Czarnacka, C. B.: The ISO/IEC Standards for the Software Processes and Products Measurement. (pp. 187-200). (2009).

[113] ISO/IEC 14598, Information Technology, Software Product Evaluation: Process for Developers. Software Engineering. 1999

[114] ISO/IEC 15939: Software Engineering – Software Measurement Process, Second edition, 2007.

[115] ISO, International Standard ISO VIM, International Vocabulary of Basic and General Terms in Metrology, International Standards Organization, Geneva, Switzerland, second edition, 1993.

[116] IEEE Standard for a Software Quality Metrics Methodology, in IEEE Std 1061-1998, vol., no., pp.i-, 31 Dec. 1998.

[117] IEEE 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology

[118] Kitchenham, B., Hughes, R.T., Linkman, S.G.: Modeling software measurement data, IEEE Transactions on Software Engineering 27 (9), 788–804. (2001).

[119] L. Briand, S. Morasca, V. Basili, An operational process for goal driven definition of measures, IEEE Transactions on Software Engineering 28 (12), 1106–1125, (2002).

[120] Kim, H.M.: Representing and Reasoning about Quality using Enterprise Models, PhD thesis, Dept. Mechanical and Industrial Engineering, University of Toronto, Canada, 1999.

[121] Van Solingen, R., Basili, V., Caldiera, G. and Rombach, H.D.: Goal question metric (GQM) approach. Encyclopedia of software engineering. (2002).

[122] Software Engineering Institute. CMMI for Development, Version 1.3, Technical Report CMU/SEI-2010-TR-033, (2010).

[123] ISO/IEC. ISO/IEC 12207: System and software engineering – Software life-cycle processes, Second edition, 2008.

[124] ISO/IEC 15504-1, Information Technology – Process Assessment – Concepts and Vocabulary, 2004.

[125] ISO/IEC 14143-6:2012 Information technology, Software measurement, Functional size measurement

[126] ISO/IEC 19761, Software Engineering COSMIC-FFP, A functional size measurement method. International Organization for Standardization ISO, Geneva, (2002).

[127] Benyon, D., Information and Data Modelling, second edition, McGraw-Hill, Wokingham, (1997).

[128] Sprinkle, J., Rumpe, B., Vangheluwe, H., and Karsai, G.: Metamodelling: State of the Art and Research Challenges. In Model-Based Engineering of Embedded Real-Time Systems: International Dagstuhl Workshop, Dagstuhl Castle, Germany, November 4-9, 2007. Revised Selected Papers (pp. 57-76), Springer Berlin Heidelberg, (2010).

[129] Stachowiak, H., Allgemeine Modelltheorie. Springer-Verlag, Wien and New York, (1973).

[130] Seidewitz, E., What Models Mean, IEEE Computer Society, (2003).

[131] Kurpjuweit, S., and Winter, R.: based Meta Model Engineering. In EMISA (Vol. 143, p. 2007), (2007).

[132] Guizzardi, G.: Conceptualizations, Modeling Languages, and (Meta) Models. In Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference, DB&IS'2006 (Vol. 155, p. 18). IOS Press, (2007).

[133] Kleppe, A. G., Warmer, J. B., and Bast, W.: MDA explained: the model driven architecture: practice and promise. Addison-Wesley Professional, (2003).

[134] Olivé, A.: Conceptual modeling of information systems. Springer Science & Business Media, (2007).

[135] Object Management Group (OMG), URL: https://www.omg.org/, [Las visited: 11-Mayıs-2023].

[136] Soley, R. M., Stone, C. M., Object Management Architecture Guide, Revision 3.0, 1995

[137] Overbeek, J. F.: Meta Object Facility (MOF): investigation of the state of the art (Master's thesis, University of Twente), (2006).

[138] Kitchenham, B. and Charters, S.: Guidelines for performing systematic literature reviews in software. In: Engineering Technical Report EBSE-2007-01, (2007).

[139] Wohlin, C.: Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In International Conference on Evaluation and Assessment in Software Engineering, ser. EASE '14, pp. 38:1–38:10, (2014).

[140] Kitchenham, B. and Brereton, O.: A systematic review of systematic review process research in software engineering. In: Information & Software Technology, vol. 55, no. 12, pp. 2049–2075, (2013).

[141] Keele,S.: Guidelines for performing Systematic Literature Reviews in Software Engineering, Version 2.3, EBSE-2007-01.

[142] Yilmaz, N., and Tarhan, A.K.: Data extraction sheet of SLR 1, URL: https://tinyurl.com/ybz2ybky

[143] Burgués, X., Franch, X., Ribó, J.M.: A MOF-compliant approach to software quality modeling. In: International Conference on Conceptual Modeling, Oct 24, pp. 176-191, Springer, Berlin, Heidelberg, 2005.

[144] Wagner, S., Lochmann, K., Heinemann, L., Kläs, M., Trendowicz, A., Plösch, R., Seidi, A., Goeb, A., Streit, J.: The quamoco product quality modelling and assessment approach. In: 34th International Conference on Software Engineering, pp. 1133-1142, IEEE, 2012.

[145] Deissenboeck, F., Wagner, S., Pizka, M., Teuchert, S., and Girard, J. F.: An activity-based quality model for maintainability. In 2007 IEEE International Conference on Software Maintenance (pp. 184-193), IEEE, (2007).

[146] Casola, V., Fasolino, A. R., Mazzocca, N., and Tramontana, P.: An ahp-based framework for quality and security evaluation. In 2009 International Conference on Computational Science and Engineering (Vol. 3, pp. 405-411), IEEE, (2009).

[147] Cachero, C., Calero, C., Poels, G.: Metamodeling the quality of the web development process' intermediate artifacts, In: International Conference on Web Engineering, pp. 74-89, Springer, Berlin, Heidelberg, Jul 16, 2007.

[148] Yilmaz, N., and Tarhan, A.K.: Data extraction sheet of SLR 2, URL: tinyurl.com/yz368ubg

[149] Adewumi, A., Omoregbe, N., Misra, S. and Fernandez, L.: Quantitative quality model for evaluating open source web applications: case study of repository software. In IEEE 16th International Conference on Computational Science and Engineering (pp. 1207-1213), (2013).

[150] Tawsopar, K. and Mekhabunchakij, K.: An evaluation of open source e-learning systems incorporated with OSMM. In Proceedings of the 6th International Conference on e-Business, Bangkok, Thailand, (2007).

[151] Akbari, M. and Peikar, S.R.H.: Evaluation of free/open source software using OSMM model case study: WebGIS and spatial database. Advances in Computer Science: an International Journal, 3(5), pp.34-43, (2014).

[152] Zarouk, M.Y., Restivo, F. and Khaldi, M.: Student-Centered Learning Environment for Self-Regulated Project-Based Learning in Higher Education: A Qualification/Selection Study. Learning through Inquiry in Higher Education: Current Research and Future Challenges (2018).

[153] Laaziri, M., Benmoussa, K., Khoulji, S. and Kerkeb, M.L.: A Comparative study of PHP frameworks performance. Procedia Manufacturing, 32, pp.864-871, (2019).

[154] Wasserman, T. and Das, A.: Using FLOSSmole data in determining business readiness ratings, In: Workshop on public data about software development. (2007).

[155] Syahlie, K., Eng, K.I., Lim, C. and Galinium, M.: Hypervisors assessment in education industry: Using OpenBRR methodology. In Proceedings of International Conference on Information, Communication Technology and System (ICTS) 2014 (pp. 303-308), (2014).

[156] Marinheiro, A. and Bernardino, J.: Openbrr evaluation of an open source bi suite. In Proceedings of the international c* conference on computer science and software engineering (pp. 134-135), (2013).

[157] Gousios, G., Karakoidas, V., Stroggylos, K., Louridas, p., Vlachos, V., and Spinellis, D.: Software Quality Assessment of Open Source Software, In: 11th Panhellenic Conference in Informatics, (2007).

[158] Gousios, G. and Spinellis, D.: Alitheia core: An extensible software quality monitoring platform. In IEEE 31st International Conference on Software Engineering (pp. 579-582), (2009).

[159] Gousios, G. and Spinellis, D.: A platform for software engineering research. In 6th IEEE International Working Conference on Mining Software Repositories (pp. 31-40), (2009).

[160] Spinellis, D., Gousios, G., Karakoidas, V., Louridas, P., Adams, P.J., Samoladas, I. and Stamelos, I.: Evaluating the quality of open source software. Electronic Notes in Theoretical Computer Science, 233, pp.5-28, (2009).

[161] Taibi, D., Del Bianco, V., Dalle Carbonare, D., Lavazza, L. and Morasca, S.: Towards the evaluation of OSS trustworthiness: Lessons learned from the observation of relevant OSS projects. In IFIP International Conference on Open Source Systems (pp. 389-395), (2008).

[162] Xu, H., Wang, B. and Zhang, H.: Research on the Experimental Framework of OpenSource Maturity Model. In International Conference on Computational Intelligence and Software Engineering (pp. 1-4). IEEE, (2010).

[163] Petrinja, E. and Succi, G.: Assessing the open source development processes using OMM. In: Advances in Software Engineering, (2012).

[164] Malanga, K.N.: Evaluation of Open Source Software with QualiPSO OMM: a case for Bungeni and AT4AM for All, In: Free and Open Source Software Conference (2015).

[165] Soto, M. and Ciolkowski, M.: The QualOSS process evaluation: initial experiences with assessing open source processes. In European Conference on Software Process Improvement (pp. 105-116). Springer, Berlin, Heidelberg, (2009).

[166] Majchrowski, A. and Deprez, J.C.: An operational approach for selecting open source components in a software development project. In European Conference on Software Process Improvement (pp. 176-188). Springer, Berlin, Heidelberg, (2008).

[167] Deprez, J.C., Monfils, F.F., Ciolkowski, M. and Soto, M.: Defining software evolvability from a free/open-source software. In Third International IEEE Workshop on Software Evolvability 2007 (pp. 29-35), (2007).

[168] Izquierdo-Cortazar, D., Gonzalez-Barahona, J.M., Duenas, S. and Robles, G.: Towards automated quality models for software development communities: The

QualOSS and FLOSSMetrics case. In Seventh International Conference on the Quality of Information and Communications Technology (pp. 364-369), (2010).

[169] Izquierdo-Cortazar, D., González-Barahona, J.M., Robles, G., Deprez, J.C. and Auvray, V.: Floss communities: Analyzing evolvability and robustness from an industrial perspective. In IFIP International Conference on Open Source Systems (pp. 336-341), (2010).

[170] Aversano, L., Pennino, I. and Tortorella, M.: Evaluating the Quality of Free/Open Source Projects. In: ENASE (pp. 186-191), (2010).

[171] Aversano, L. and Tortorella, M.: Evaluating the quality of free/Open source systems: A case study. In: International Conference on Enterprise Information Systems (pp. 119-134). Springer, Berlin, Heidelberg, (2010).

[172] Aversano, L., Guardabascio, D. and Tortorella, M.: Analysing the Quality Evolution of Open Source Software Projects. In: International Conference on Software Quality (pp. 117-129). Springer, Cham, (2017).

[173] Aversano, L. and Tortorella, M.: Applying EFFORT for evaluating CRM open source systems. In: International Conference on Product Focused Software Process Improvement (pp. 202-216). Springer, Berlin, Heidelberg, (2011).

[174] Aversano, L., Pennino, I. and Tortorella, M.: Evaluating the Quality of Free/Open Source ERP Systems. In: ICEIS (1) (pp. 75-83), (2010).

[175] Leite, N., Pedrosa, I. and Bernardino, J.: Open Source Business Intelligence Platforms' Assessment using OSSpal Methodology. In ICETE (1) (pp. 356-362), (2018).

[176] Marques, J.F. and Bernardino, J.: Evaluation of Asana, Odoo, and ProjectLibre Project Management Tools using the OSSpal Methodology. In: KEOD (pp. 397-403), (2019).

[177] Calçada, A. and Bernardino, J.: Evaluation of Couchbase, CouchDB and MongoDB using OSSpal. In: KDIR (pp. 427-433), (2019).

[178] Cruz, S. and Bernardino, J.: Project Management Tools Assessment with OSSpal. In: KEOD (pp. 390-396), (2019).

[179] De Paula, H.C. and Bernardino, J.: An Application of OSSpal for the Assessment of Open Source Project Management Tools. In: WEBIST (pp. 411-417), (2019).

[180] Poernomo, I.: The meta-object facility typed. In Proceedings of the 2006 ACM symposium on Applied computing (pp. 1845-1849), (2006).

[181] Yilmaz,. N, and Kolukısa Tarhan, A.: Definition of the terminologies in the SQMM [Data set]. Zenodo. https://doi.org/10.5281/zenodo.6367596 (2022).

[182] Zahoor, A., Mehboob, K. and Natha, S.: Comparison of open source maturity models. In: Procedia computer science 111, 348-354, (2017).

[183] Stol, K.J., and Babar, M.A.: "A comparison framework for open source software evaluation methods", In: International Conference on Open Source Systems. Springer, Berlin, Heidelberg, (2010).

[184] Wagner, S.: Cost-Optimization of analytical software quality assurance: models, data, case studies, VDM Verlag, (2008).

[185] Dromey, R. G.: A model for software product quality, IEEE Transactions on Software Engineering, 21(2): 146-162, 1995.

[186] Al-Badareen, A.B., Selamat, M.H., Jabar, M.A., Din, J. and Turaev, S.: Software quality models: A comparative study. In International Conference on Software Engineering and Computer Systems (pp. 46-55). Springer, Berlin, Heidelberg. 2011.

[187] Yan, M., Xia, X., Zhang, X., Xu, L. and Yang, D.: A systematic mapping study of quality assessment models for software products. In 2017 International Conference on Software Analysis, Testing and Evolution (SATE) (pp. 63-71). IEEE. (2017).

[188] Deissenboeck, F., Juergens, E., Lochmann, K. and Wagner, S.: Software quality models: Purposes, usage scenarios and requirements. In 2009 ICSE workshop on software quality (pp. 9-14). IEEE. (2009).

[189] Kitchenham, B. and Pfleeger, S.L.: Software quality: the elusive target [special issues section]. IEEE software, 13(1), pp.12-21. 1996.

[190] Siemens company, website url: https://www.siemens.com/global/en.html

[191] Pinter, C. C.: A book of abstract algebra. Courier Corporation, (2010).

[192] ISO/IEC 25020:2019 Systems and software engineering, Systems and software Quality Requirements and Evaluation (SQuaRE), Quality measurement framework

[193] García, F., Ruiz, F., Calero, C., Bertoa, M. F., Vallecillo, A., Mora, B., & Piattini, M. (2009). Effective use of ontologies in software measurement. The Knowledge Engineering Review, 24(1), 23-40.

[194] Tanrıöver, Ö. Ö., and Bilgen, S.: A framework for reviewing domain specific conceptual models. Computer Standards & Interfaces, 33(5), 448-464, (2011).

[195] Kläs, M., Lampasona, C., Nunnenmacher, S., Wagner, S., Herrmannsdörfer, M., and Lochmann, K.: How to evaluate meta-models for software quality. In Proceedings of the 20th International Workshop on Software Measurement, (2010).

[196] Kitchenham, B., Pfleeger, S. L., & Fenton, N. (1995). Towards a framework for software measurement validation. IEEE Transactions on software Engineering, 21(12), 929-944

[197] List of questions and expert opinion (Step-5), URL: https://tinyurl.com/2ow7ayua

[198] Flyvbjerg, B. (2006). Five misunderstandings about case-study research. Qualitative inquiry, 12(2), 219-245

[199] Gerring, J. (2016). Case study research: Principles and practices. Cambridge university press

[200] Feagin, J. R., Orum, A. M., & Sjoberg, G. (Eds.). (1991). A case for the case study. UNC Press Books.

[201] Yin, R. K.:The case study anthology. Sage. (2004).

[202] Wang, S., & Wang, H. (2014). A Survey of Open Source Enterprise Resource Planning (ERP) Systems. International Journal of Business & Information, 9(1).

[203] Fougatsaro, V. (2009). A study of open source ERP systems

[204] Carvalho, R. A. D. (2006). Issues on evaluating free/open source ERP systems. In Research and practical issues of enterprise information systems (pp. 667-675). Springer, Boston, MA

[205] Kim, H., & Boldyreff, C. (2005). Open source ERP for SMEs.

[206] ISO/IEC 25010.: Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) Quality Model and guide. International Organization for Standardization, Geneva, Switzerland, (2008).

[207] Al-Kilidar, H., Cox, K. and Kitchenham, B.: The use and usefulness of the ISO/IEC 9126 quality standard. In: International Symposium on Empirical Software Engineering, IEEE, (2005).

[208] Ardito, L., Coppola, R., Barbato, L., & Verga, D. (2020). A tool-based perspective on software code maintainability metrics: a systematic literature review. Scientific Programming, 2020.

[209] Bakar, A. D., Sultan, A. B. M., Zulzalil, H., & Din, J. (2012). Review on 'Maintainability'Metrics in Open Source Software.

[210] Chawla, M. K., & Chhabra, I. (2015, October). Sqmma: Software quality model for maintainability analysis. In Proceedings of the 8th Annual ACM India Conference (pp. 9-17).

[211] Dagpinar, M., & Jahnke, J. H. (2003, November). Predicting maintainability with object-oriented metrics-an empirical comparison. In 10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings. (pp. 155-155). IEEE Computer Society.

[212] Dubey, S. K., & Rana, A. (2011). Assessment of maintainability metrics for object-oriented software system. ACM SIGSOFT Software Engineering Notes, 36(5), 1-7.

[213] M. Hanefi Calp, N. Arici, B. Enstitüsü, G. Üniversitesi, and T. Ankara, "Nesne Yönelimli Tasarım Metrikleri ve Kalite Özellikleriyle İlişkisi," Politek. Derg. J. Polytech. Cilt Digit. Object Identifier, vol. 14141, no.10, pp. 9–14, 2011.

[214] F. B. U. Erdemir, U. Tekin, "Nesneye Dayalı Yazılım Metrikleri ve Yazılım Kalitesi," Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu, 2008.

[215] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Trans. Softw. Eng., vol. 20, no. 6, pp. 476–493, 1994.

[216] Stebbins, R. A.: Exploratory research in the social sciences (Vol. 48). Sage, (2001)

[217] Yilmaz, N.: List of community-based measures and their descriptions. Zenodo. https://doi.org/10.5281/zenodo.8008179

[218] Triantaphyllou, E.: Multi-criteria decision making methods. In: Multi-criteria decision making methods: A comparative study. Springer, Boston, MA, 5-21, (2000).

[219] Hasnain, S., Ali, M. K., Akhter, J., Ahmed, B., & Abbas, N. (2020). Selection of an industrial boiler for a soda-ash production plant using analytical hierarchy process and TOPSIS approaches. Case Studies in Thermal Engineering, 19, 100636.

[220] Zaidan, A. A., Zaidan, B. B., Al-Haiqi, A., Kiah, M. L. M., Hussain, M., & Abdulnabi, M. (2015). Evaluation and selection of open-source EMR software packages based on integrated AHP and TOPSIS. Journal of biomedical informatics, 53, 390-404.

[221] Dweiri, F., Kumar, S., Khan, S. A., and Jain, V.: Designing an integrated AHP based decision support system for supplier selection in automotive industry. Expert Systems with Applications, 62, 273-283, (2016).

[222] Caputo, A. C., Pelagagge, P. M., and Salini, P.: AHP-based methodology for selecting safety devices of industrial machinery. Safety science, 53, 202-218, (2013).

[223] Saaty, T. L., and Sagir, M.: Ranking countries more reliably in the summer olympics. International Journal of the Analytic Hierarchy Process, 7(3), 589-610, (2015).

[224] Deshamukhya, T., and Ray, A.: Selection of cutting fluid for green manufacturing using analytical hierarchy process (AHP): a case study. International Journal of Mechanical Engineering and Robotics Research, 3(1), 173-182, (2014).

[225] Khashei-Siuki, A., & Sharifan, H. (2020). Comparison of AHP and FAHP methods in determining suitable areas for drinking water harvesting in Birjand aquifer. Iran. Groundwater for Sustainable Development, 10, 100328

[226] Hanine, M., Boutkhoum, O., Tikniouine, A., & Agouti, T. (2016). Application of an integrated multi-criteria decision making AHP-TOPSIS methodology for ETL software selection. SpringerPlus, 5(1), 1-17.

[227] Saaty TL. The analytic hierarchy process: planning, priority setting and resource allocation. New York: McGraw-Hill; (1980).

[228] Saaty, T. L.: Decision making with the analytic hierarchy process. International journal of services sciences, 1(1), 83-98, (2008).

[229] Işıklar, G., and Büyüközkan, G.: Using a multi-criteria decision making approach to evaluate mobile phone alternatives. Computer Standards & Interfaces, 29(2), 265-274, (2007).

[230] Hwang CL, Yoon K. Multiple attribute decision making: methods and applications. New York: Springer-Verlag; 1981.

[231] Yilmaz, N., and Kolukısa Tarhan, A.: Supplementary document of the thesis. Zenodo. https://doi.org/10.5281/zenodo.7986369 (2023).

[232] Online URL: https://plugins.jetbrains.com/plugin/93-metricsreloaded

[233] Online URL: https://plugins.jetbrains.com/plugin/12159-codemetrics

[234] D. Spinellis and M. Jureczko.(May 2011). Metric Description [Online] Available: http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/

[235] Scitools | Build Notes. URL: https://scitools.com/buildnotes/

[236] Goeb, A. (2013). A Meta Model for Software Architecture Conformance and Quality Assessment. Electronic Communications of the EASST, 60

[237] Newcomer, K. E., Hatry, H. P., & Wholey, J. S. (2015). Conducting semi-structured interviews. Handbook of practical program evaluation, 492, 492.

[238] Molléri, J. S., Petersen, K., and Mendes, E.: Survey guidelines in software engineering: An annotated review. In Proceedings of the 10th ACM/IEEE

international symposium on empirical software engineering and measurement (pp. 1-6), (2016).

[239] Jotform, URL: https://www.jotform.com/, [Las visited: 11-Mayıs-2023].

[240] Joshi, A., Kale, S., Chandel, S., & Pal, D. K. (2015). Likert scale: Explored and explained. British journal of applied science & technology, 7(4), 396.

[241] Salem, I. E. B. (2015). Transformational leadership: Relationship to job stress and job burnout in five-star hotels. Tourism and Hospitality Research, 15(4), 240-253

[242] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B.,Wesslén, A.: Experimentation in software engineering: An introduction. Springer, (2012).

APPENDIX

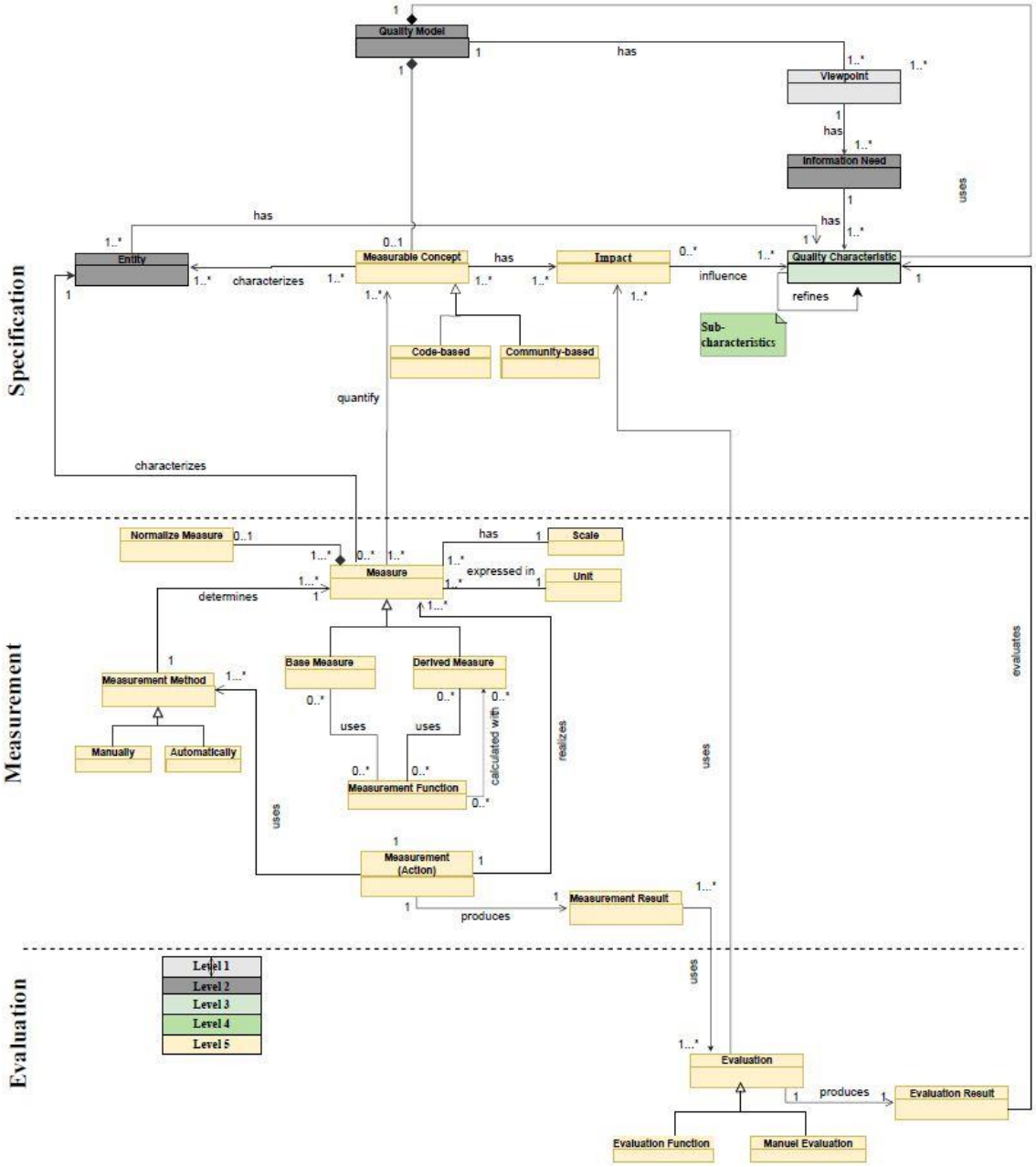
APPENDIX-1 – List of Primary Studies Included in SLR Study [30]

R1	Duijnhouwer, F.W., Widdows, C.: Capgemini Expert Letter Open Source Maturity Model, URL: tinyurl.com/yxdbvjk6, Capgemini, (2003)
R2	Polancic, G., Horvat R.V., and Rozman, T..: Comparative assessment of open source software using easy accessible data. In: 26th International Conference on Information Technology Interfaces, IEEE, (2004).
R3	Koponen, T.: Evaluation framework for open source software maintenance. In: International Conference on Software Engineering Advances (ICSEA'06), IEEE, (2006).
R4	Semeteys, R.: Method for Qualification and Selection of Open Source software (QSOS), version 1.6." URL: tinyurl.com/y2phllex, (2006).
R5	Wasserman, M.P., Chan, C.: Business Readiness Rating Project, BRR Whitepaper RFC 1, URL: tinyurl.com/y5srd5sq, (2006).
R6	Sung, W.J, Kim, J.H. and Rhew, S.Y.: A quality model for open source software selection. In: *Sixth International Conference on Advanced Language Processing and Web Information Technology*. IEEE, (2007).
R7	Taibi, D., Lavazza, L., and Morasca, S..: OpenBQR: A framework for the assessment of OSS. In: IFIP International Conference on Open Source Systems. Springer, Boston, MA, (2007).
R8	Raffoul, E., Domínguez, K., Perez, M., Mendoza, L. E. and Griman, A. C.: Quality model for the selection of FLOSS-based Issue tracking system. In: Proceedings of the IASTED international conference on software engineering, Innsbruck, Austria. Vol. 12. (2008).
R9	Samoladas, I., Gousios G., Spinellis, D. and Stamelos, I.: The SQO-OSS quality model: measurement based open source software evaluation. In: *IFIP international conference on open source systems*. Springer, Boston, MA, (2008).
R10	Ciolkowski, M., and Soto, M.: Towards a comprehensive approach for assessing open source projects. In: Software Process and Product Measurement. Springer, Berlin, Heidelberg, 316-330, (2008).
R11	Alfonzo, O., Domínguez, K., Rivas, L., Pérez, M., Mendoza, L. and Ortega, M.: Quality measurement model for analysis and design tools based on FLOSS. In 19th Australian Conference on Software Engineering, (pp. 258-268), IEEE, (2008)
R12	Petrinja, E., Nambakam, R., and Sillitti, A.: Introducing the opensource maturity model. In: ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. IEEE, (2009).
R13	Del Bianco, V., Lavazza, L., Morasca, S., and Taibi, D.: Quality of open source software: The QualiPSo Trustworthiness Model. In: IFIP International Conference on Open Source Systems. Springer, Berlin, Heidelberg, (2009).
R14	Soto, M., and Ciolkowski, M.: The QualOSS open source assessment model measuring the performance of open source communities. In: 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE, (2009).
R15	Del Bianco, V., Lavazza, L., Morasca, S., Taibi, D., and Tosi, D.: The QualiSPo approach to OSS product quality evaluation. In: Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. (2010).
R16	Müller, T.: How to choose a free and open source integrated library system. In: OCLC Systems & Services: International digital library perspectives, (2011).
R17	Chirila, C. B., Juratoni, D., Tudor, D., and Cretu, V.: Towards a software quality assessment model based on open-source statical code analyzers. In: 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI). IEEE, (2011).

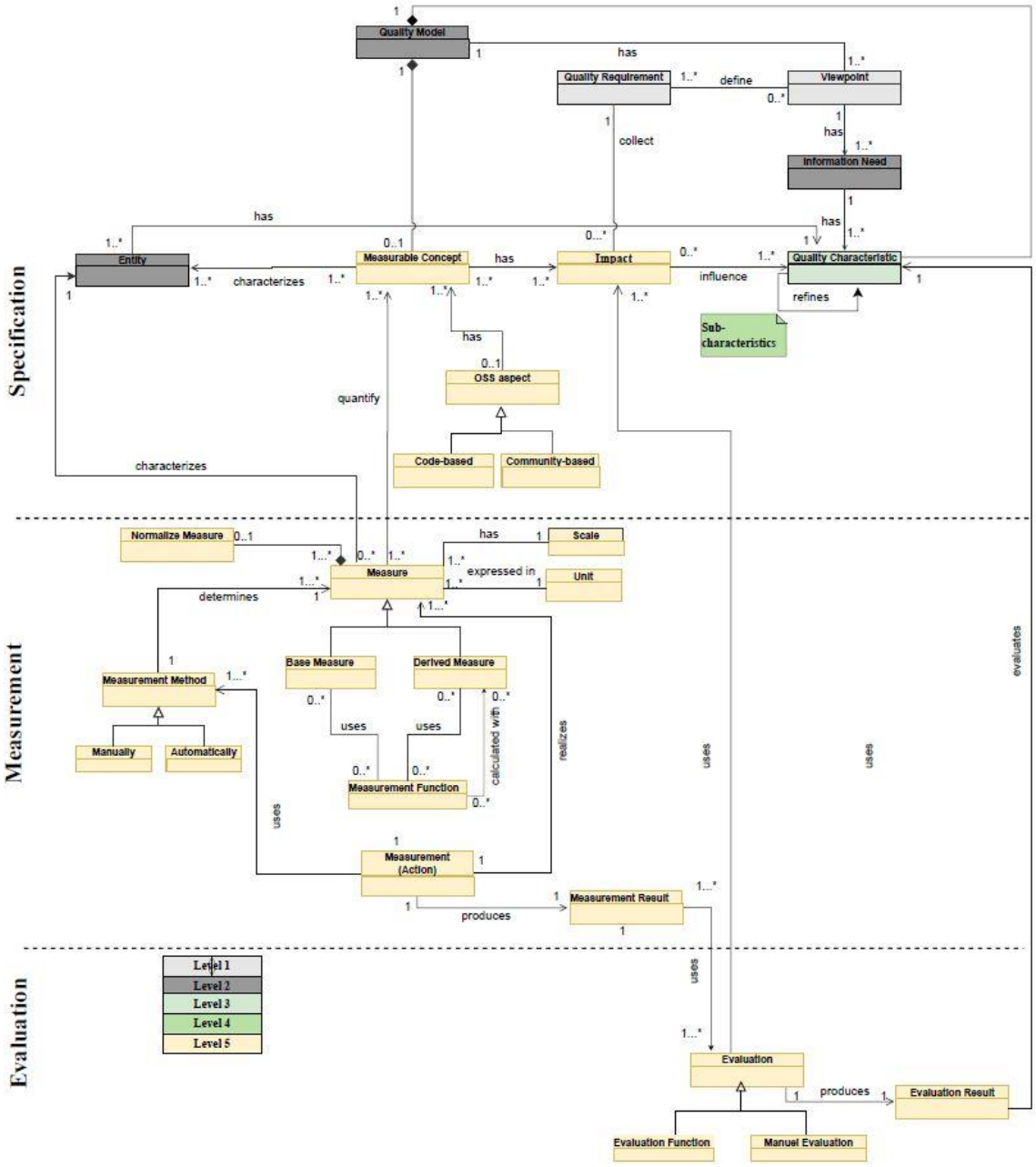
R18	Lee, W., Lee, J.K. and Baik, J.: Software reliability prediction for open source software adoption systems based on early lifecycle measurements. In 35th Annual Computer Software and Applications Conference (pp. 366-371). IEEE. (2011)
R19	Lavazza, L., Morasca, S., Taibi, D. and Tosi, D.: OP2A: how to improve the quality of the web portal of open source software products. In International Conference on Web Information Systems and Technologies (pp. 149-162). Springer, Berlin, Heidelberg. (2011)
R20	Raza, A., Capretz, L. F., and Ahmed, F.: An open source usability maturity model (OS-UMM). In: Computers in Human Behavior 28.4, 1109-1121, (2012).
R21	Adewumi, A., Omoregbe, N., Misra, S., and Fernandez, L.: Quantitative quality model for evaluating open source web applications: case study of repository software. In: IEEE 16th International Conference on Computational Science and Engineering. IEEE, (2013).
R22	Aversano, L., and Tortorella, M.: Quality evaluation of floss projects: Application to ERP systems. In: Information and Software Technology 55.7, 1260-1276, (2013).
R23	Sarrab, M., and Rehman, O. M. H.: Empirical study of open source software selection for adoption, based on software quality characteristics. In: Advances in Engineering Software 69, 1-11, (2014).
R24	Roy, J., Contini, C., Brodeur, F., Diouf, N., and Suryn, D. W.: Method for the evaluation of open source software quality from an IT untrained user perspective. In: Proceedings of the International C* Conference on Computer Science & Software Engineering, (2014).
R25	Kuwata, Y., Takeda, K., and Miura, H.: A study on maturity model of open source software community to estimate the quality of products. In: Procedia Computer Science 35, 1711-1717, (2014).
R26	Houaich, Y. A., and Belaissaoui, M.: Measuring the maturity of open source software. In: 6th International Conference on Information Systems and Economic Intelligence (SIIE). IEEE, (2015).
R27	Aversano, L., and Tortorella, M.: Analysing the reliability of open source software projects. In: 10th International Joint Conference on Software Technologies, Vol. 1. IEEE, (2015).
R28	Okamura, H. and Dohi, T.: Towards comprehensive software reliability evaluation in open source software. In 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE) (pp. 121-129). IEEE. (2015).
R29	Khatri, S. K., and Singh, I.: Evaluation of open source software and improving its quality. In: 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), IEEE, (2016).
R30	Di Ruscio, D., Kolovos, D. S., Korkontzelos, Y., Matragkas, N., Vinju, J.: Supporting custom quality models to analyse and compare open-source software. In: 10th International Conference on the Quality of Information and Communications Technology, IEEE, (2016).
R31	Wasserman, A. I., Guo, X., McMillian, B., Qian, K., Wei, M. Y., and Xu, Q.: OSSpal: finding and evaluating open source software. In: IFIP International Conference on Open Source Systems. Springer, Cham, (2017).
R32	Vijaya, P., Chander, S., and Raju, G.: Usqo-Foss quality model: utilization based software quality observatory for evaluation of free and open source software. In: Free and Open Source Software Conference, (2017).
R33	Eghan, E. E., Algahtani, S. S., Forbes, C., and Rilling, J.: API trustworthiness: an ontological approach for software library adoption. In: Software Quality Journal 27.3, 969-1014, (2019).
R34	Adewumi, A., Misra, S., Omoregbe, N., and Sanz, L. F.: FOSSES: Framework for open-source software evaluation and selection. In: Software: Practice and Experience 49.5, 780-812, (2019).
R35	Razzaq, S., and Xie, M.: Understanding the Surviving Bugsin Open Source Software through the Community Perspective: Using Bayesian Analysis. In: Amity International Conference on Artificial Intelligence, IEEE, (2019).
R36	Wang, Q., Zhang, W., Jiang, J., and Li, L.: A Reliability Automatic Assessment Framework of Open Source Software Based on JIRA. In: Proceedings of the 2020 9th International Conference on Software and Computer Applications. (2020).

APPENDIX-2 – Development of the OSS-QMM Through Versions

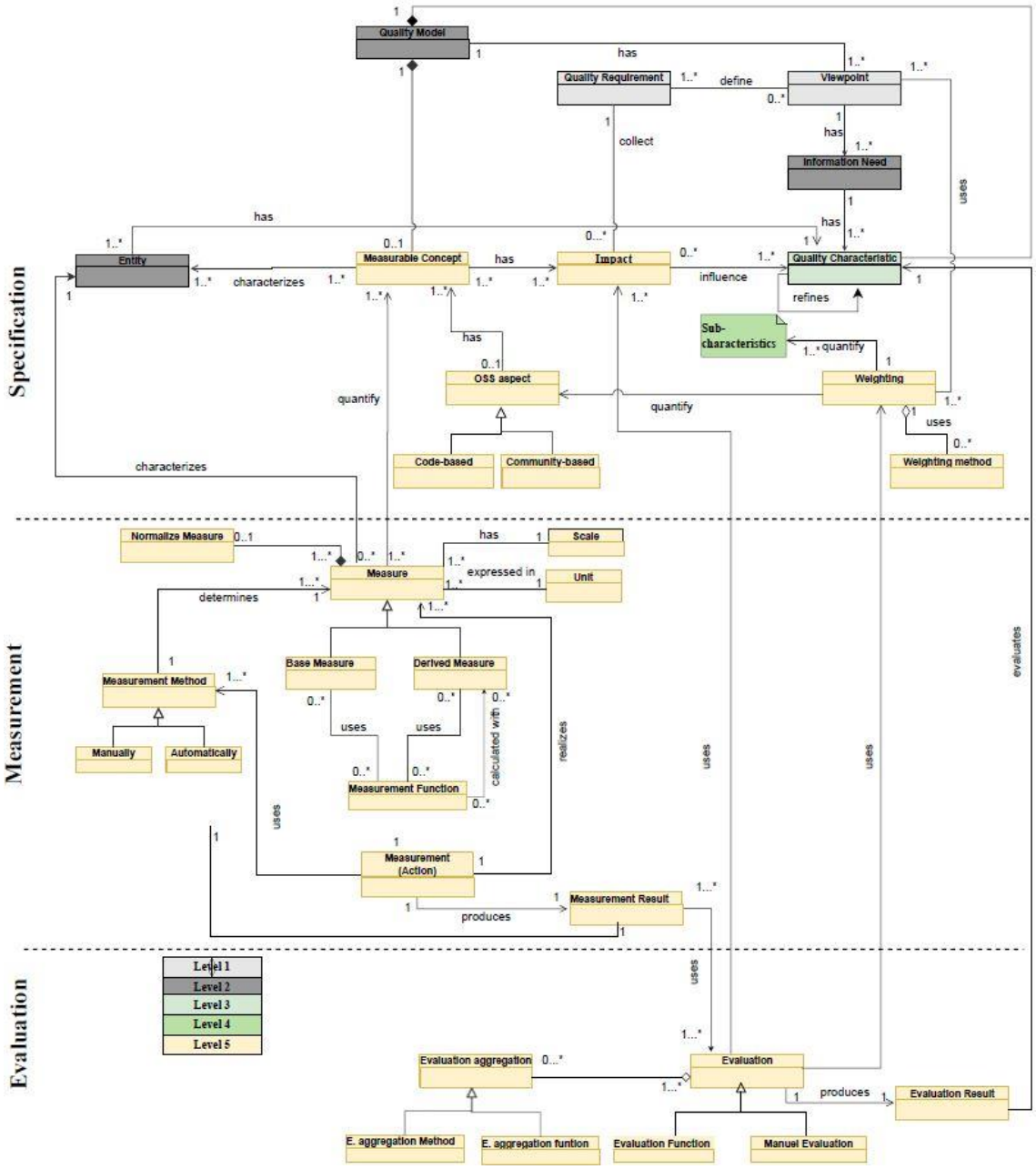
(a) The Initial Version (v1) of the OSS-QMM



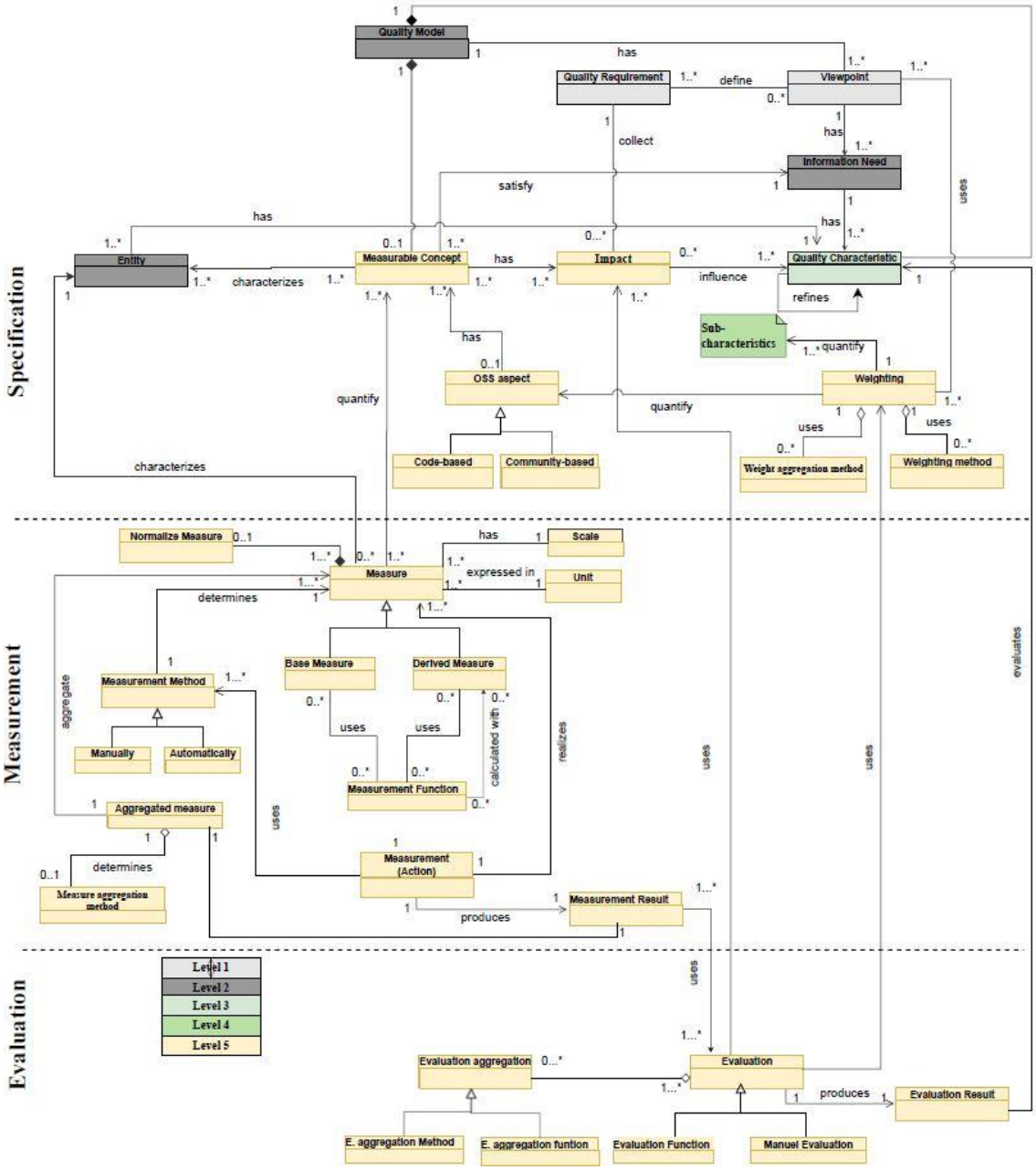
(b) The Second Version (v.2) of the OSS-QMM



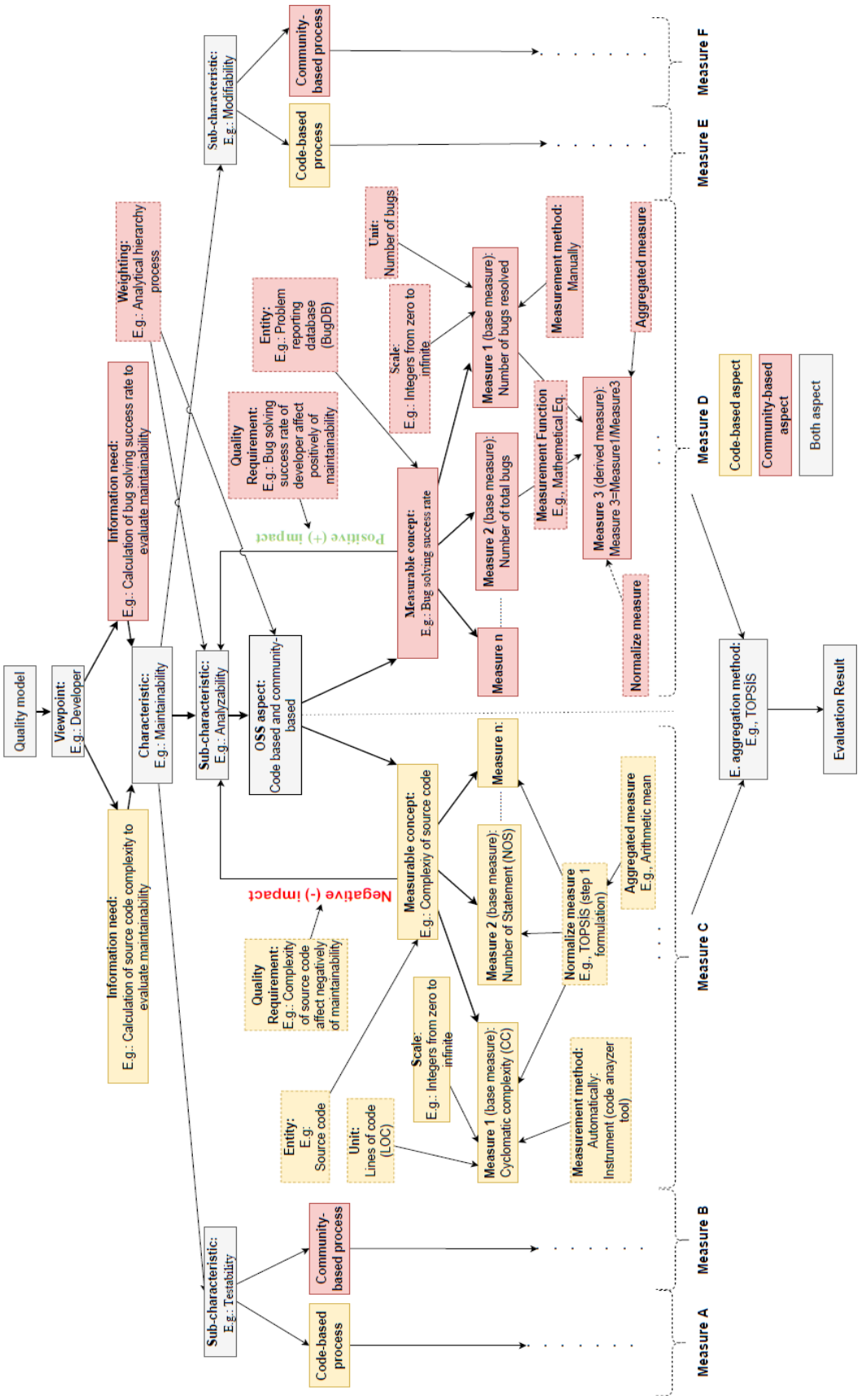
(c) The Third Version (v.3) of the OSS-QMM



(d) The Fourth Version (v.4) of the OSS-QMM



APPENDIX-3 – The New Operationalized Quality Model Derived from OSS-QMM



APPENDIX-4 – Detailed Information about the Background of Experts

Expert	Experience in quality modeling	Current Position	OSS knowledge (years)	Modeling knowledge (years)	Company size	Country	Interview time	Interview type	Include/ exclude
E1	11	SD	5	5	Very large	Turkey	77 min.	Online	Include
E2	7	SQAE	5	5	Medium	A.B.D	83 min.	Online	Include
E3	9	QAT and QAA	4	4	Large	Turkey	68 min.	Face-to-face	Include
E4	10	SQAE	4	5	Very large	Turkey	55 min.	Online	Include
E5	8	ITMC	4	4	Large	Canada	60 min.	Online	Include
E6	22	CO	5	5	Very large	Norway	73 min.	Face-to-face	Include
E7	8	SE and SQAE	4	5	Very large	Turkey	60 min.	Online	Include
E8	12	PM	5	4	Large	Turkey	84 min.	Online	Include
E9	7	QAM	5	4	Medium	Turkey	81 min.	Online	Include
E10	14	QAM	4	5	Very large	Turkey	74 min.	Online	Include
E11	10	QAM	5	4	Very large	England	64 min.	Online	Include
E12	7	SSE	5	5	Very large	Turkey	56 min.	Face-to-face	Include
E13	13	LQAE	4	5	Medium	Turkey	75 min.	Online	Include
E14	17	QAT	5	5	Very large	Canada	67 min.	Online	Include
E15	8	SEM	5	4	Large	Turkey	80 min.	Face-to-face	Include
E16	12	QAT	5	5	Very large	A.B.D	67 min.	Online	Include
E17	15	QAM	4	5	Very large	Turkey	59 min.	Online	Include
E18	7	SQAE	4	4	Large	Turkey	51 min.	Face-to-face	Include
E19	9	LQAE	5	4	Large	Germany	67 min.	Online	Include
E20	11	QAA	4	5	Very large	Netherlands	72 min.	Online	Include
E21	7	QAT	3	3	Medium	Turkey	-	Online	Exclude
E22	6	QAA	4	3	Large	Turkey	-	Online	Exclude
E23	8	SE	4	3	Medium	Germany	-	Online	Exclude
E24	10	BDM	3	3	Very large	A.B.D	-	Online	Exclude
E25	7	SA	3	2	Large	Turkey	-	Online	Exclude
E26	8	SC	3	4	Large	Turkey	-	Online	Exclude
E27	4	SE	4	3	Medium	Turkey	-	Online	Exclude
E28	6	QAT	4	4	Medium	Turkey	-	Online	Exclude
E29	5	SQAE	4	4	Very large	Turkey	-	Online	Exclude

Abbreviation: Software Director (SD), Software Quality Assurance Engineer (**SQAE**), Quality Assurance Tester (**QAT**), Quality Assurance Analyst (**QAA**), IT management consultant (**ITMC**), Company owner (**CO**), Software Engineer (**SE**), Project Manager (**PM**), Quality Assurance Manager (**QAM**), Senior Software Engineer (**SSE**), Lead quality assurance engineer (**LQAE**), Software Engineering Manager (**SEM**), Quality Assurance Manager (**QAM**), Business Development Manager (**BDM**), Software Architect (**SA**), and Security Consultant (**SC**).

Information: Very large company (1000+ employees), Large company (200-999 employees), Medium (50-199 employees), Small (10-46 employees), and Very small (1-9 employees)

APPENDIX-5 – The Screenshots Obtained After the Semi-structured Interview conducted with Expert #10 via the Questionnaire

(a) The output for Part-1: Demonstrating Applicability of the OSS-QMM in Practice w.r.t Consistency in Evaluation Results and Effectiveness in Model Derivation

Q1-Do you have previous experience in using Open source ERP systems?

- (●) Yes
- ( ) No

Q1.1-Which of the following Open source ERP systems do you have experience with? (you may select one, or more than one)

- [x] Apache OfBiz
- [x] Adempiere
- [ ] Compiere

Q2-Do you employ any method/technique (e.g., quality model) in choosing the software to use in your company?

- (●) Yes
- ( ) No

Q2.1 What type of software can this method/techniquebe applied to? (you may select more than one)

- [ ] Open SourceSoftware (OSS)
- [x] Commercial off-the-shelf (COTS)
- [ ] Developed by yourself
- [ ] Other

Q3-If you evaluated the following open source ERP systems to use in your company with your own techniques, what would the weights be? (Consider that these ERP systems are assumed to be easy to maintain. Therefore, giving a higher weight means that the product is more preferable in terms of maintainability.)

	1	2	3	4	5	6	7	8	9	10
Apache OFBiz	○	○	○	○	○	○	○	●	○	○
Adempiere	○	○	○	○	○	○	●	○	○	○
Compiere	○	○	○	○	○	○	○	○	○	○

While answering **Q4** and **Q5**, please open the following link in a new page: OSS-QMM: (Link)

Q4-Suppose that you would match the terms of the quality model used in your own company for software evaluation, with the terms of the OSS-QMM. Which of the following meta-model terms would you match? ("Agg" denotes sub-categories or aggregated concepts under that category.)

- [x] Quality model
- [x] Information need
- [x] Entity
- [x] Quality sub-characteristics
- [x] OSS aspect
- [x] Community-based
- [x] Weighting (Agg:Weighting method)
- [x] Unit
- [x] Measurement method (Agg: manually, automatically)
- [x] Aggregated measure (Agg: measure aggregation method)
- [x] Measurement result
- [ ] Evaluation aggregation (Agg: evaluation aggregation method, evaluation aggregation function)
- [x] Viewpoint
- [x] Quality requirements
- [x] Quality characteristics
- [x] Measurable concepts
- [x] Code-based
- [ ] Impact
- [x] Measures (Agg: base measure, derived measure, measurement function)
- [x] Scale
- [x] Normalized measure
- [x] Measurement
- [ ] Evaluation (Agg: evaluation function, manual evaluation)
- [ ] Evaluation result

While answering **Q5**, please open the following link in a new page: An example OSS qualit model (Link)

Q5-Do you think that the terms of the example OSS quality model (derived from OSS-QMM) and the terms of the OSS-QMM are compatible with each other?

1 2 3 4 **5**

Strongly disagree — Strongly agree

Q6-If a tool was developed to automate the use of the developed OSS-QMM and enable the derivation of standard OSS quality models,would you use such a tool?

1 2 3 4 **5**

Strongly disagree — Strongly agree

(b) The output for Part-2: Assessment of the OSS-QMM w.r.t Its Practical Applicability

While answering **Q1**, please open the following links in the new pages: OSS-QMM (Link-1), and structure of selected OSS quality models (Link-2)

Q1-Do you think the OSS-QMM is sufficiently general to describe an existing OSS quality model (e.g., OSMM, SQO-OSS, and QualOSS)?

1 2 3 **4** 5
Strongly disagree Strongly agree

While answering **Q2**, please open the following link in a new page: 5-level structure of software quality models: (Link)

Q2-Do you think the mapping process is compatible with the (5-level) structure of the SQMMs?

1 2 3 4 **5**
Strongly disagree Strongly agree

Q3-Do you think the classification of the OSS-QMM terms (under specification, measurement, and evaluation) are useful?

1 2 3 **4** 5
Strongly disagree Strongly agree

Q4-Do you think the OSS-QMM is complete? (In other words, do you think the OSS-QMM enables to derive OSS quality models that address all aspects of OSS products -e.g., code-based and community-based aspects?)

1 2 3 **4** 5
Strongly disagree Strongly agree

While answering **Q5**, please open the following link in a new page: An example OSS quality model (Link)

Q5-Do you think the example OSS quality model (derived from OSS-QMM) is complete? (In other words, do you think the example OSS quality model can be used to evaluate all aspects of OSS products – e.g., code-based and community-based aspects?)

1 2 3 4 **5**
Strongly disagree Strongly agree

Q6-Do you think the OSS quality models to be derived from the OSS-QMM will have homogeneous structure?

1 2 3 4 **5**
Strongly disagree Strongly agree

Q7-Do you think the OSS-QMM provides flexibility inderiving the OSS quality models?

1 2 3 4 **5**
Strongly disagree Strongly agree

Q8-Do you think the stakeholders can intervene in the quality model to be derived to fulfill needs?

1 2 3 4 **5**
Strongly disagree Strongly agree

Q9-Do you think the OSS-QMM can be useful in coping with the heterogeneous data of the OSS products?

1 2 3 4 **5**
Strongly disagree Strongly agree

Q10-Do you think the OSS-QMM is understandable?

1 2 3 4 **5**
Strongly disagree Strongly agree

Q11-Do you think the OSS quality model (derived from OSS-QMM) is understandable?

1 2 3 **4** 5
Strongly disagree Strongly agree

Q12-Do you think deriving OSS quality models from the OSS-QMM will be difficult – if yes, what is the level of difficulty?

1 2 3 4 **5**
Strongly disagree Strongly agree

APPENDIX-6 – Mapping the Terms in Existing OSS-QMs (i.e., OSMM, OpenBRR, and SQO-OSS) to the Concepts of the OSS-QMM

OSS-QMM concepts	**Terms in OSS quality models**			
Quality model	**OSMM**	**OpenBRR**	**SQO-OSS**	
Viewpoint	Developer	Developer	Developer	
OSS aspect	Community-based	Community-based	Code-based	Community-based
Information need	Calculation of developer size to evaluate maintainability	Calculation of developer productivity to evaluate maintainability	Calculation of comment frequency to evaluate maintainability	Calculation of documentation quality to evaluate maintainability
Characteristic	Maintainability	Maintainability	Maintainability	Maintainability
Sub-characteristic	Acceptance	Product quality	Analyzability	Analyzability
Entity	Developer	Contributor	Source code	Contributor
Quality requirement	The large size of developer is desirable for maintainability.	The productive developers are desirable for maintainability.	The high comment frequency is desirable for maintainability.	The large number of document is desirable for maintainability.
Impact	Positive	Positive	Positive	Positive
Measurable concepts	The size of developer	Productivity of contributors	Complexity of source code	Completeness of documentation
Measure	Number of developer (Base measure)	Number of release (Base measure)	Weighted method per class (WMC) (Base measure)	Number of documents (Base measure)
Unit	Developer	Release	Methods	Documents
Scale	Integer from zero to five (The score (1-5) is assigned w.r.t. rules given in OSMM)	Integer from zero to three (The score (1-3) is assigned w.r.t. rules given in OpenBRR)	Integer from zero to infinity	Integer from zero to infinity
Measurement method	Manually	Manually	Automatically (e.g., Understand scitool, CKJM, Intellij IDEA, etc.)	Manually
Measurement function	There is no measurement function because it is a base measure.	There is no measurement function because it is a base measure.	There is no measurement function because it is a base measure.	There is no measurement function because it is a base measure.