Software Quality Journal (2023) 31:721–773
https://doi.org/10.1007/s11219-022-09603-3

Matching terms of quality models and meta-models: toward a unified meta-model of OSS quality

Nebi Yılmaz[1,2] · **Ayça Kolukısa Tarhan**[1,3]

Accepted: 2 October 2022 / Published online: 17 December 2022

Abstract

Context In the last two decades, open-source software (OSS) has gained increasing attention due to its voluntary supporters, growing community, and ease of accessibility in cloud repositories. Standardization in OSS quality is of vital importance as a communication vehicle for stakeholders in identifying and selecting high-quality products. Thus, meta-models help to define a standardized language and enable to propose quality models that can be used to perform comparable measurements.
Objective Considering the lack of a comprehensive meta-model of OSS quality in the literature, there appears a need to see a more complete picture of OSS quality and to represent its concepts more formally. Therefore, in this study, it is aimed to develop a solid base for a comprehensive meta-model of OSS quality to create a common understanding among stakeholders.
Method A systematic way has been followed toward developing a common structure, defining a consistent terminology and, finally, providing a meta-model of OSS quality. In this context, (1) the common structure of the quality models for OSS has been investigated, (2) the terms of the general-purpose meta-models of software quality have been analyzed based on the international standards, and (3) the terms of the quality models for OSS have been mapped with the elements of these meta-models.
Results An initial meta-model of OSS quality, which employs a unified structure from the OSS quality models and eliminates the inconsistencies determined in the general-purpose meta-models of software quality, has been proposed and an implementation of this meta-model has been demonstrated.
Conclusion This initial meta-model of OSS quality with a standard terminology can be taken as a guide by researchers who will propose or revise their OSS quality models. It will allow developing multiple OSS quality models with homogenous structure and terms, and also enable comparing the evaluation results obtained by these models.

Keywords Software quality · Quality evaluation · Quality measurement · Quality metrics · Meta-model · Quality model · Open-source software · OSS

✉ Nebi Yılmaz
yilmaz@cs.hacettepe.edu.tr

Extended author information available on the last page of the article

1 Introduction

Open-source software (OSS) and its components are used as part of software that supports many activities of human life (Miguel et al., 2014; Tassone et al., 2018) and have attracted noteworthy attention in the last two decades (Adewumi et al., 2019; Rossi et al., 2012). Considering this increasing attention, evaluating the quality of OSS has become more crucial and essential (Maki-Asiala & Matinlassi, 2006). OSS data is accessible from cloud repositories in two aspects which are code-based (e.g., lines of code, number of bugs) and community-based (e.g., mailing list, number of developers). Because the data is scattered in a variety of databases and heterogeneous sources, defining and evaluating the quality of OSS are considered more challenging in comparison to its proprietary counterpart (i.e., commercial software). In other words, OSS has a dynamic and diverse nature, and accordingly, its quality is affected by various variables (e.g., the longevity of the project, the mailing list density). Therefore, defining situation-based procedures for determining evaluation criteria becomes a challenging task (Adewumi et al., 2019; Petrinja et al., 2010; Sung et al., 2007).

As modeling may facilitate controlling and understanding of complex concepts, several quality models for OSS have been developed by researchers such as SQO-OSS (Samoladas et al., 2008) and OSMM (Duijnhouwer & Widdows, 2003). Despite the existence of these models, they are not complete or not sufficient in some aspects. For example, they are not flexible enough to be applicable in all business domains (Adewumi et al., 1936), not applicable by external parties (Jean-Christophe & Alexandre, 2008), not cover all aspects of quality (Aversano & Tortorella, 2013; Ciolkowski & Soto, 2008), and not fair in quality validation (Adewumi et al., 2013), etc. The results of our recent SLR study (Yılmaz & Tarhan, 2022) on OSS quality models and some other studies (Hauge et al., 2009; Lenarduzzi et al., 2020; Stol & Babar, 2010; Thapar et al., 2012) have indicated that these models have some challenges of adoption in practice. In general, as identified by our SLR study (Yılmaz & Tarhan, 2022), the quality models for OSS have 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. Accordingly, the majority of the models consists of a wide variety of information in their bodies in various structures, as the base for specifying and evaluating OSS quality. 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 (Yılmaz & Tarhan, 2022). Considering that the results of such comparisons can be input to OSS selection and/or adoption by companies, the importance of having a standard in terminology and modeling to evaluate OSS quality becomes even more prominent.

In this regard, a comprehensive meta-model to guide stakeholders in specifying and evaluating OSS quality may create and enhance a common understanding in the community. However, the results of our SLR study on meta-models for software quality and its evaluation (SQiE) (Yılmaz & Tarhan, 2020) indicated that generally, there is a lack of meta-models targeted for OSS quality. Meta-models are defined as models of models with the rules needed to build specific models, so the models which are derived from the meta-models could have homogenous structures and common terms. In other words, meta-models are important because they enable to standardize model development (Garcia et al., 2007; Wagner et al., 2015). Standardization assists organizations to interoperate using engineering discipline with agreed and well-recognized practices and technologies (Garcia et al., 2006; Ramamoorthy et al., 1984). Accordingly, results obtained from different models for the same purpose can be compared.

In order to develop a comprehensive meta-model of OSS quality, first, the software quality meta-models in literature should be analyzed. Despite the fact that researchers have developed quality meta-models for different types of software (e.g., OSS (Mens et al., 2011), custom (Mohagheghi & Dehlen, 2008), and commercial-of-the-shelf (Wagner et al., 2012), concepts and terminology used in these meta-models differ among them (Garcia et al., 2009; Nistala et al., 2019; Yılmaz & Tarhan, 2020). Throughout this study, the reader should consider that the term "software quality meta-model (SQMM)" or "meta-model" correspond to the quality of all types of software, unless it is indicated that it corresponds to specific type of software (e.g., OSS). The same is also true for the term "software quality model." Specifically, the abbreviation for our initial meta-model developed within the scope of this study is indicated as OSS-QMM (Open Source Software Quality Meta Model). Inconsistencies among the terms of the SQMMs prevent standardization of software quality measurements so that results cannot be compared and the current state of the software products cannot be improved. For example, some terms are used interchangeably among the SQMMs (e.g., metric vs. measure) or the same terms can be named differently in different SQMMs (e.g., factor vs. measurable concept). In parallel to the efforts for international standardization such as ISO/IEC 15939 (2007), software measurement terminology continues to be defined, consolidated, and agreed upon. International standards that were proposed once upon a time (i.e., withdrawn or replaced) for software quality or measurement caused a lack of consensus on the terminology and concepts used in this field (Bertoa et al., 2006; Garcia et al., 2006; Ruiz et al., 2003). Consequently, inconsistencies and terminology conflicts appeared even between international standards (Garcia et al., 2006; Rout, 1999), and the inconsistencies in the terminology of them were reflected in the SQMMs.

To cope with the aforementioned challenges, in this study, the systematic research process has been followed toward proposing an initial meta-model of OSS quality. In this context, first of all, since inconsistencies among the terminology of international standards are reflected in the SQMMs, the terms of the SQMMs have been analyzed with respect to the terms of international standards for software quality and measurement. Then, the terms of the software quality models have been analyzed to elicit the inconsistencies among the terms of the quality models. Then, a matching has been performed between the terms of the quality models for OSS and the terms of the SQMMs. As a result of this systematic process, an initial meta-model has been proposed for defining and evaluating OSS quality. Finally, initial validation of the proposed meta-model has been performed over an example evaluation.

More specifically, the contributions of this study 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 initial OSS-QMM.
- The investigation on 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 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 initial OSS-QMM will allow to develop 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.
- Since the initial 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 initial OSS-QMM will have the chance of adopting a standard structure.

This study can be beneficial for two types of audiences: first, for OSS evaluators who may be confused by inconsistent terminology in the existing SQMMs; second, standard or meta-model developers for OSS quality, who will propose new meta-models or integrate consistent terms into the existing meta-models. Overall, this study has an effort to contribute to the provision of "coherent" and "consistent" terminology for OSS-specific meta-models.

The rest of this article is organized as follows: In Sect. 2, the background that forms the basis for the development process of the OSS-QMM is given. In Sect. 3, related studies in our scope are briefly discussed. In Sect. 4, the methodology followed in the development of the initial OSS-QMM is explained. In Sect. 5, the development process is elaborated, the OSS-QMM is presented, its initial validation is provided, and potential threats to its validity are discussed. Finally, in Sect. 5.6, conclusions and plans for future work are presented.

2 Background

A systematic way has been followed for the initial OSS-QMM developed within the scope of this study, as detailed in Sect. 4. In order to create a solid basis for the steps of this systematic process and to understand the structures used in this process, background analysis has been performed with the help of the literature in this section. It is essential to analyze the existing related quality models and meta-models in the literature in order to develop a meta-model fit to purpose (Othman & Beydoun, 2010; Wagner et al., 2012). In this context, first of all, the current situations and deficiencies of the existing SQMMs are analyzed in Sect. 2.1. This section has formed the infrastructure for the analyses performed in Sect. 5.1. Then, the quality models are addressed according to their structures, together with an analysis of which model was derived from which other, by considering basic and tailored models, in Sect. 2.2. These analyses have formed the basis for the process of obtaining the common structure of quality models, as explained in Sect. 5.2. Then, since the inconsistencies between the terms of the meta-models arise from the measurement standards and proposals, these sources are explained and analyzed in Sect. 2.3. This analysis is important to ensure that the matching process performed in Sect. 5.3 and the concepts of the OSS-QMM developed and explained in Sect. 5.4 are consistent. Finally, in Sect. 2.4, information is given about the MOF standard (2019) that has been used as a basis in our OSS-QMM development process.

2.1 Software quality meta-models (SQMMs), including OSS quality meta-models

In this section, the current situation of meta-models for OSS quality will be explored based on literature search. The SQMMs are important because they allow to standardize quality models and, thus, they create a common understanding between stakeholders for proper quality management throughout the entire life of a software product. Also, the SQMMs allow us to see a more complete picture of software quality and to represent concepts of software quality more formally (Wagner et al., 2015). Despite the fact that SQMMs are important for specifying and evaluating the quality of software products, research on SQMMs for OSS quality are not at the desired level in terms of mainly three deficiencies:

1. Few number of meta-models and their adoption in the community,
2. Lack of depth in their content, and
3. Inconsistent terminology among them.

The results of the SLR study (Yılmaz & Tarhan, 2020) verify the first two deficiencies that the number of meta-models for OSS quality is low and that their content remains shallow. Also, the results in that SLR study have triggered us to examine inconsistencies between the terms of the meta-models. Thus, we are currently reporting this research to verify the third deficiency, that is, the inconsistency in their terminologies. In this regard, inconsistencies, commonalities, and terminology conflicts in the SQMMs proposed for OSS and custom type of software have been analyzed and verified, as explained in detail in Sect. 5.1. In the SLR study (Yılmaz & Tarhan, 2020), a total of 28 meta-models were analyzed, and the results indicated that only two meta-models (Eghan et al., 2019; Mens et al., 2011) were developed for evaluating OSS quality and that their evaluation aspect was not widely explored. In the first study (Mens et al., 2011), only few elements were defined in the meta-model to evaluate the quality of evolving OSS systems. In the second study (Eghan et al., 2019), a meta-model was proposed to measure the quality of external OSS libraries in terms of trustworthiness. Therefore, it became apparent from the review results that has been no comprehensive meta-model proposed for OSS quality and the need to define one have remained open.

In the SLR study (Yılmaz & Tarhan, 2020), methods used to validate meta-models in the studies were also examined since the maturity of the meta-models is related to their validation. The results indicated that 13 of meta-models were validated by case studies, 6 by toy experiments, 4 by peer reviews by experts, and 1 by pilot project application. However, 5 studies did not explicitly mention the method of validation, and 1 did not use any validation method. It should be stated that a study might have used more than one validation method for its meta-model. Also, among all 28 meta-models analyzed in the SLR, the two meta-models proposed for OSS were both validated by designing case studies. An important finding out of these results was that a real-world case has not been used by the studies to validate their meta-models. Another important finding was that maturity of the meta-models and adoption of the meta-models in practice have remained limited.

2.2 Software quality models (SQMs), including 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 (Suman & Rohtak, 2014). This is not a current issue and studies have been

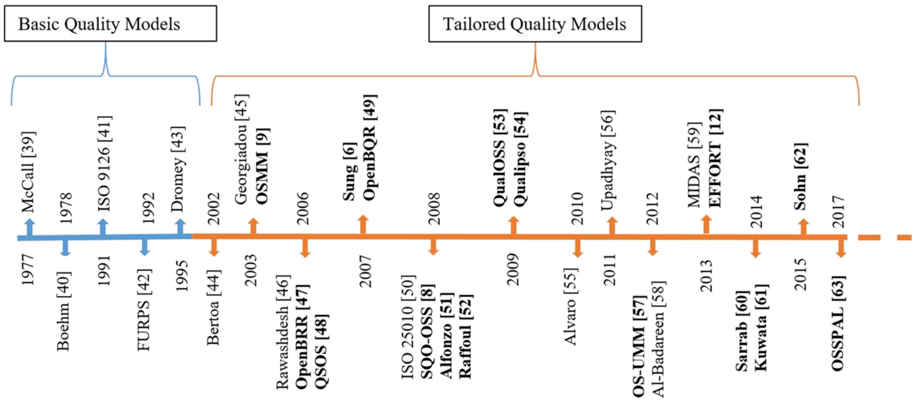

Basic Quality Models
Tailored Quality Models
1977 McCall [39]
1978 Boehm [40]
1991 ISO 9126 [41]
1992 FURPS [42]
1995 Dromey [43]
2002 Bertoa [44]
2003 Georgiadou [45] OSMM [9]
2006 Rawashdesh [46] OpenBRR [47] QSOS [48]
2007 Sung [6] OpenBQR [49]
2008 ISO 25010 [50] SQO-OSS [8] Alfonzo [51] Raffoul [52]
2009 QualOSS [53] Qualipso [54]
2010 Alvaro [55]
2011 Upadhyay [56]
2012 OS-UMM [57] Al-Badareen [58]
2013 MIDAS [59] EFFORT [12]
2014 Sarrab [60] Kuwata [61]
2015 Sohn [62]
2017 OSSPAL [63]

Fig. 1 Basic and tailored quality models over the years (quality models in bold text indicate OSS quality models)

conducted on software quality for years. As technology in the software industry is constantly evolving, expectations from 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 years is shown in Fig. 1. As seen from the figure, quality models are classified as basic quality models developed until 2001 and tailored quality models developed after this year (Miguel et al., 2014; Sadeghzadeh & Rashidi, 2017). Detailed information is given in Sect. 2.2.1.2 for both basic and tailored quality models, which is important to understand the structure and content of quality models proposed for OSS, prior to the matching process between terms of OSS quality models and SQMMs. Also, it is important in shaping the structure of an initial OSS-QMM quality developed within the scope of this study and in shaping the structure of SQMMs to be developed in the future.

The primary motivation behind this study 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 (Yılmaz & Tarhan, 2022) to characterize the existing quality evaluation models or frameworks (QEMoF) for OSS and to examine comprehensively 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. 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 1. Also, due to their importance in the community, the OSS quality models examined within the scope of this study have been the subjects to the systematic studies (i.e., systematic mappings and systematic literature reviews) such as (Adewumi et al., 1936; Lenarduzzi et al., 2020) and to the comparison studies such as (Haaland et al., 2010; Jean-Christophe & Alexandre, 2008; Zahoor et al., 2017). These models fall into the "tailored" quality models category because of their construction based on the basic quality models (e.g., ISO/IEC 9126, 2001, Boehm et al., 1978) and due to their particular application domains (e.g., a specific quality characteristic of OSS product (Duijnhouwer & Widdows, 2003; Wasserman & Chan, 2006).

Table 1 Classification of software quality models w.r.t. structural and basic/tailored properties

Model/category	Structural		Basic and tailored		
	Hierarchical	Dynamic	Basic	Tailored	Based on (if tailored)
McCall	✓		✓		
Boehm	✓		✓		
Dromey	✓	✓	✓		
Furps	✓		✓		
ISO/IEC 9126	✓		✓		
OSMM	✓			✓	ISO/IEC 9126
QSOS	✓			✓	ISO/IEC 9126
OpenBRR	✓			✓	ISO/IEC 9126 and OSMM
SQO-OSS	✓			✓	ISO/IEC 9126 and OSMM
QualOSS	✓			✓	OSMM, OpenBRR and QSOS

Basic quality models, on the other hand, were 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. In the SLR study (Yılmaz & Tarhan, 2022), 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 the basic models given in Table 1. 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 1). 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 research, 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 were directly derived from ISO/IEC 9126, and there has been no model yet derived from ISO/IEC 25010. In the following sub-sections, the structure of the quality models will be classified before their structural analysis is carried out in Sect. 5.2. This way, a solid basis is formed for developing the OSS-QMM.

2.2.1 Classification of quality models

Many quality models have been proposed in literature, which serve 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 were 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. 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. Basic and tailored classification enables understanding the basics of the OSS quality models, examining of 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.

2.2.1.1 Structural classification In literature, each quality model is composed of a set of building blocks including quality objectives, factors, criteria, sub-criteria, and metrics (Sadeghzadeh & Rashidi, 2017). 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 (Wagner, 2008). In this context, quality models examined in this study 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 (Sadeghzadeh & Rashidi, 2017). 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. 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 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 the quality models with a hierarchical structure is included in Table 1.

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 such as (Dromey, 1995) focuses on the relationship between attributes and sub-attributes to provide flexibility in the evaluation of different software products (Al-Badareen et al., 2011). 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 for creating consistent quality models and focus on a variety of building blocks covering all quality engineering tasks (Sadeghzadeh & Rashidi, 2017). The only quality model that falls into this category is Dromey's quality model (Dromey, 1995). As also shown in Table 1, a quality model can fall into more than one category in structural classification.

2.2.1.2 Basic and tailored classification *Basic quality models* developed until 2001 are the models that mostly focus on comprehensive evaluation and that aim to evaluate software products from many aspects (Miguel et al., 2014). 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 mostly have been adopted to commercial software (e.g., COTS) and have overlooked some specific properties of OSS (e.g., community-based aspects) (Adewumi et al., 1936; Khatri & Singh, 2016). Thus, they do not provide sufficient support for assessing the quality of OSS (Adewumi et al., 2019; Miguel et al., 2014; Samoladas et al., 2008). Basic models are already shown in Fig. 1 and also listed in Table 1.

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 (Miguel et al., 2014). 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 (Miguel et al., 2014; Sadeghzadeh & Rashidi, 2017). For example, the MIDAS quality model proposed by Siemens (Siemens company, website url:https://www.siemens.com/global/en.html, xxxx) 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. 1 and also listed in Table 1. In addition, Table 1 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.

2.2.1.3 Classification according to evaluation aspect and the evaluated characteristic The quality models determined within the scope of this study are classified according to their evaluation aspects and key quality characteristics they possess, as shown in Fig. 2. The qual-

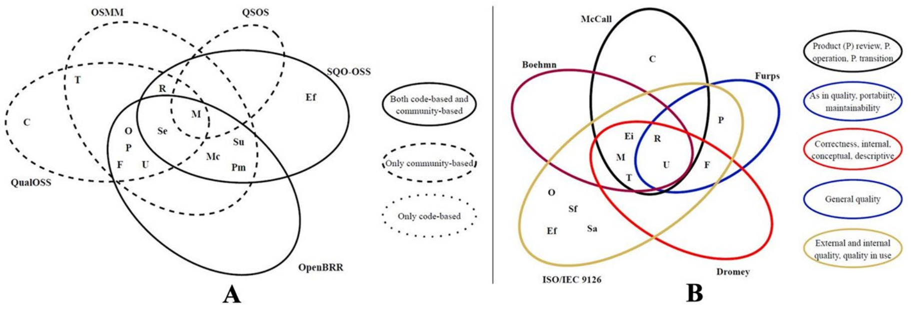

OSMM
QSOS
SQO-OSS
QualOSS
OpenBRR
C
T
R
M
O
Se
P
F
U
Su
Mc
Pm
Ef
Both code-based and community-based
Only community-based
Only code-based
A
McCall
Boehmn
Furps
C
P
Ei
R
M
U
T
F
O
Sf
Ef
Sa
ISO/IEC 9126
Dromey
Product (P) review, P. operation, P. transition
As in quality, portabiity, maintainability
Correctness, internal, conceptual, descriptive
General quality
External and internal quality, quality in use
B

Fig. 2 Classification w.r.t. evaluation aspects and quality characteristics of **A** OSS quality models and **B** basic quality models. M, Maintainability; R, reliability; F, functionality; P, performance; O, operability; Se, security; Ef, effectiveness; C, compatibility; T, transferability; U, usability; Ei, efficiency; Sa, satisfaction; Sf, safety; Mc, maintenance capacity; Su, sustainability; Pm, process maturity

ity characteristics used in the quality models are classified with respect to the quality characteristics of ISO/IEC 25010, which is the latest quality model. Also, some quality characteristics such as maintenance capacity, sustainability, and process maturity (Adewumi et al., 1936; Yılmaz & Tarhan, 2022) that belong to the community side of OSS are included. Abbreviations of the quality characteristics evaluated in each quality model are given just below the figure. As seen in Fig. 2A, 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 (Adewumi et al., 2019; Yılmaz & Tarhan, 2022). 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. 2B. 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. 2B, 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 (Adewumi et al., 2019; Yılmaz & Tarhan, 2022). 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 subsection 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.

2.3 Standards for software measurement

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. One of the primary goals of software engineering is to release high-quality software 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 to compare measurement results, with the pre-requisite that vocabulary in measurement standards or proposals are 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. The important organization and standardization bodies such as ISO, IEC, and IEEE have developed many international standards for software engineering. There is a large number of the international standards developed by only ISO for measuring software processes and products, as presented in Fig. 3.

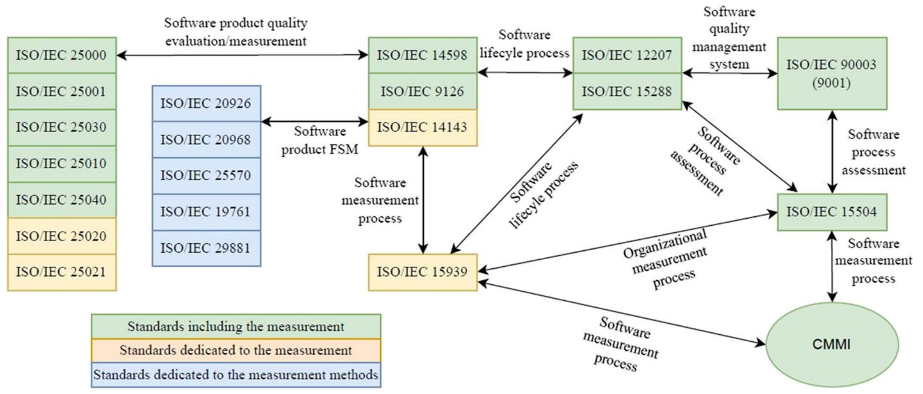

ISO/IEC 25000
ISO/IEC 25001
ISO/IEC 25030
ISO/IEC 25010
ISO/IEC 25040
ISO/IEC 25020
ISO/IEC 25021
Software product quality evaluation/measurement
ISO/IEC 20926
ISO/IEC 20968
ISO/IEC 25570
ISO/IEC 19761
ISO/IEC 29881
Software product FSM
ISO/IEC 14598
ISO/IEC 9126
ISO/IEC 14143
Software measurement process
ISO/IEC 15939
Software lifecyle process
ISO/IEC 12207
ISO/IEC 15288
Software quality management system
ISO/IEC 90003 (9001)
Software lifecyle process
Software process assessment
Software process assessment
ISO/IEC 15504
Organizational measurement process
Software measurement process
Software measurement process
CMMI
Standards including the measurement
Standards dedicated to the measurement
Standards dedicated to the measurement methods

Fig. 3 Main relationships between the ISO/IEC standards of software quality and software measurement, and their relation with the CMMI model, as adapted from Czarnacka (2009)

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 the 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 (Garcia et al., 2006). Inconsistencies, commonalities, and terminology conflicts in all these sources are reflected in SQMMs because they are created by adopting the terminology and concepts from the international standards.

The SLR study (Yılmaz & Tarhan, 2020) 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 meta-models. Complementary to that, in this study, it has been attempted to determine the international standards whose terminology is largely referenced by the SQMMs that have been proposed for OSS and custom type of software. In accordance with this purpose, ISO/IEC 14598 (Software Engineering-Product Evaluation) (1999), ISO/IEC 15939 (Software Engineering-Software Measurement Process) (2007), and VIM (International Vocabulary of Basic and General Terms in Metrology) (1993) are determined from ISO and IEC organizations. Also, IEEE 1061 Software Quality Metrics Methodology (1998) and IEEE 610.12 (1990) are determined from the IEEE organization. Descriptions of these international standards are overviewed in Table 2. The use of terminology of these standards in the SQMMs and terminology conflicts among the standards will be discussed in Sect. 5.1.

In addition to the standards mentioned above, some research proposals by Kitchenham et al. (2001), Briand et al. (2002), and Kim (1999), all related to software measurement, are included in this study since terminology of these proposals has been adopted by some SQMMs such as (Garcia et al., 2007). 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, as the descriptions of the international standards.

The international standards and research proposals related to software measurement and quality can be classified under three main categories according to particular topics they address (Garcia et al., 2006): software measures, measurement processes, and targets and goals, which are respectively denoted by C1, C2, and C3 in Table 2. The first category of

Table 2 Standards and proposals whose terminology is referenced by SQMMs

Standard/proposal	C1	C2	C3	Description
IEEE 610.12 (1990)	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 (1993)	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 (1998)	P	P	P	It is a standard that enables to obtain quality requirements, and also enables 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 (1999)	Y	P	N	It is a standard that enables to measure, assess, and evaluate the quality of software products. It enables to perceive the evaluation process from different points of view such as acquirers, developers, and evaluators
ISO/IEC 15939 (2007)	P	Y	P	It is a standard that enables 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 (1999)	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 et al. (2001)	N	N	Y	It is a conceptual model that covers the definition of many software measures and 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 et al. (2002)	N	N	Y	It is an approach that is based on the GQM approach (Solingen et al., 2002) 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, *C3* target and goals, *Y* yes, *N* no, *P* partially software measures (C1) focuses on main elements such as measures, unit of measurements, and scale in the definition of software metrics. The second category of measurement process (C2) focuses on the definition of terminology related to software measurement act such as measurement methods and measurement results. The third category of target and goals (C3) focuses on gathering concepts related to objectives and scope of the measurement process, such as attribute, measurable entity, and information need. The categories that the standards or proposals address are denoted in columns 2–4 in Table 2. 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 (Garcia et al., 2006). 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 and examined within the scope of this study, there are important models such as CMMI (Capability Maturity Model Integration) (2010) and standards such as ISO/IEC 12207 (Standard for Software Life Cycle Processes) (2008) and ISO/IEC 15504 (Standard for Software Process Assessments) (2004) 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. 3. It should be noted that the standards or models other than those listed in Table 2 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, as they mentioned them 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 (2007), and functional size measurement (FSM) standards (e.g., ISO/IEC 14143, 2012; ISO/IEC 19761, 2002) are totally aligned with VIM (1993) (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-object-facility (MOF) standard

The quality models were not proposed by adhering to a certain SQMM, and this negatively affects the comparison of evaluation results and, therefore, the possibility of standardization in measurement. As a solution to this problem, quality models need to be created in the language defined by SQMMs. Information systems researchers have proposed a variety of meta-modeling frameworks (e.g., OMG & MOF, 2019; Henderson-Sellers & Bulthuis, 1996) in the literature. In this study, we have followed the meta-modeling framework based on the Meta-Object-Facility (MOF) standard in developing our OSS-QMM.

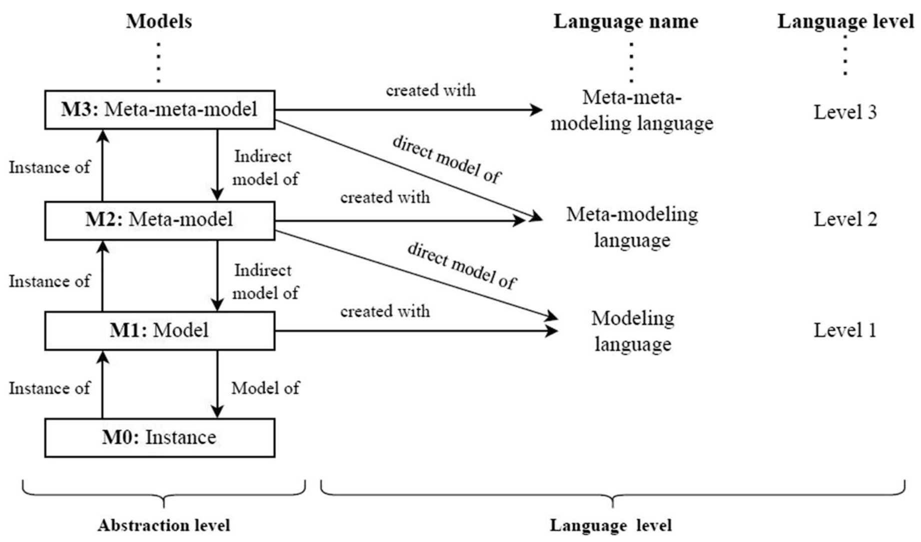


Fig. 4 Abstraction levels of models and levels of modeling languages (OMG & MOF, 2019; Karagiannis & Kühn, 2455)

The abstraction levels of the MOF architecture and the levels of modeling language are given in Fig. 4. The MOF standard has 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, the layer M_i contains an instance of the layer M_{i+1} and the layer M_{i+1} describes the layer M_i. 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)*. A modeling language consists of its *syntax*, *semantics*, and *notation*. 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 the semantic mapping, and the *notation* describes the visualization of a modeling language. A meta-model, therefore, allows the development of multiple models with the homogenous structure and common terms by using the same modeling language that it proposes. Within the scope of this research, in order to create a common OSS-QMM at layer M2 with the purpose of standardizing OSS quality measurements, the language defined at layer M3 should be used. Then, by using this OSS-QMM as a modeling language, OSS quality models with a common structure and terminology could be obtained as the instances of the meta-model.

3 Related work

Studies have been reported in literature on the consistency of terminologies of different standards in this field. However, to the best of our knowledge, our study is the first one to perform a matching process between the terms of the SQMMs and the terms of the quality models for OSS. Also, this is the first study that investigates inconsistency among the terms of the SQMMs. Since the terminology of the SQMMs is determined based on the measurement standards emerged in time, the studies that investigate harmonization of vocabulary used in these standards as well as the studies that investigate the terms of the meta-models are valuable for our study. Accordingly, the studies that reveal the inconsistency of terminology in different standards and also, our recent SLR study (Yılmaz & Tarhan, 2020) that addresses the terms of the meta-models for software quality and its evaluation, are summarized in this section.

García et al. (Garcia et al., 2006) conducted a study that addresses and analyzes inconsistent terminology, discrepancies, and gaps between software measurement proposals. For this purpose, a few international standards and articles were examined and inconsistent concepts and terminologies between them were investigated. In this context, a basic software measurement ontology was presented to help the consistency of terminology for standards and measurement proposals. The authors did not claim that this study would completely resolve the inconsistency between standards and software measurement proposals since it is a very challenging task. Considering the importance of standardization and harmonization, they stated that inconsistency between standards should be focused on, which supports the discussions in this field. In addition, the studies of Garcia et al. (2009) and (Bertoa et al., 2006) had a similar purpose as the study (Garcia et al., 2006) outlined above. Both also investigated inconsistent terminology between software measurement standards and proposed a software measurement ontology.

Barcellos and Almeida Falbo (2013) stated that there is a semantic interoperability problem in literature since it is a challenging task to jointly use different measurement-related standards. Thus, they aimed to support the semantic integration of software applications that support measurement (Barcellos & Almeida Falbo, 2013). In accordance with this purpose, they proposed a software measurement task ontology (SMTO) to establish a common conceptualization that considers the software measurement process. The process of software measurement is very diverse and complex, so this ontology considers only core activities such as measurement planning, execution, and analysis. The authors used this task ontology to harmonize the process of software measurement addressed in ISO/IEC 15939 (2007), PSM (Mcgarry et al., 2002), ISO/IEC/IEC 12207 (2008), and CMMI (2010). In addition, the study of Barcellos et al. (2010) had a similar purpose as the study (Barcellos & Almeida Falbo, 2013) outlined above, and also proposed a software measurement ontology to contribute to the semantic interoperability problem of standards.

Chirinos et al. (2005) stated that software measures generally are poorly defined in the industry and this causes the collected data to be invalid and incomparable. Therefore, it is not considered sufficient whether a definition is theoretically correct for just one measurement and also, everyone should understand what the measured values represent (Chirinos et al., 2005). In order to define software measures used in literature and to contribute to performing reliable and comparable measurements, the authors presented a data MOdel for Software MEasurement (MOSME) which is a conceptual model and describes a consistent measurement process. This model focused on the definition of terminology which is used in software measurement and it supported ISO/IEC 15939, ISO/IEC 14598, and also Goal Question Metric (GQM) (Solingen et al., 2002) approach.

Yilmaz and Tarhan (2020) conducted a systematic literature review (SLR) study to address in detail the structure and content of the meta-models developed for software quality and its evaluation (SQiE). The motivation for the SLR study was the fact that meta-models are important for standardization, harmonization, and consistency of software measurement. The study examined the meta-models from many aspects, but the most important contribution and motivation related to our current study are that it addressed the terms used in the meta-models and provided the frequency of use of these terms in the included meta-models. Therefore, the study triggered the investigation of the inconsistency of the terms in the meta-models and also in the standards since the terms of the meta-models were determined based on the measurement standards appeared in time.

As seen above, efforts have been put to eliminate the inconsistencies between different standards and to make the software measurements reliable and comparable. Since the SQMMs are of great importance for the standardization of the quality models, an effort has been spent to eliminate the inconsistencies between meta-models of OSS quality, unlike others, in this study. Therefore, this study is expected to have an important role that software quality meta-models proposed in the future will have consistent and coherent terminology and concepts.

4 Methodology

In this section, the methodology followed in developing the OSS-QMM is explained. The OSS-QMM is a set of constructs and their relationships identified as quality modeling language, which corresponds to the M2 layer (i.e., meta-model level) of the MOF architecture given in Fig. 4. In this regard, by considering the MOF architecture, an iterative, *step-based process for meta-model creation* has been adapted from Beydoun et al. (2009) and Othman et al. (2014), as shown in Fig. 5. This process has been used by several other studies in the literature such as (Al-Dhaqm et al., 2017; Othman & Beydoun, 2010). As seen from the figure, the development process of the OSS-QMM is iterative with a continuous refinement of new concepts. A systematic process is followed in development and each step of the process is explained in the following sub-sections briefly.

4.1 Literature search-1 (step 1)

In order to develop a complete SQMM, the structure and content of the existing quality meta-models in the literature should be well understood. In this context, an SLR study (Yılmaz & Tarhan, 2020) was performed and a total of 28 SQMMs were analyzed. This enhances our domain awareness as recommended in Beydoun et al. (2009) as an initial step for any meta-modeling process. This SLR study addressed the frequency of use of concepts

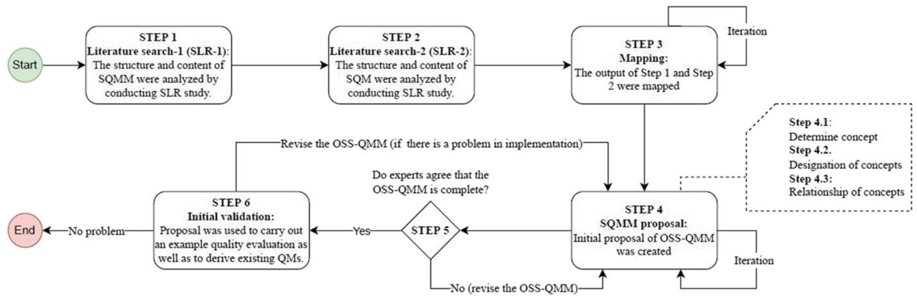

Start
STEP 1
Literature search-1 (SLR-1):
The structure and content of SQMM were analyzed by conducting SLR study.
STEP 2
Literature search-2 (SLR-2):
The structure and content of SQM were analyzed by conducting SLR study.
STEP 3
Mapping:
The output of Step 1 and Step 2 were mapped
Iteration
Step 4.1:
Determine concept
Step 4.2.
Designation of concepts
Step 4.3:
Relationship of concepts
Revise the OSS-QMM (if there is a problem in implementation)
Do experts agree that the OSS-QMM is complete?
STEP 6
Initial validation:
Proposal was used to carry out an example quality evaluation as well as to derive existing QMs.
STEP 5
Yes
STEP 4
SQMM proposal:
Initial proposal of OSS-QMM was created
Iteration
End
No problem
No (revise the OSS-QMM)

Fig. 5 Development methodology of the OSS-QMM

in the included meta-models and showed that there are inconsistencies between the terms of the meta-models. Therefore, in our current study, a detailed analysis has been performed to eliminate these inconsistencies. The origins of these terms have been explored and these terms have been analyzed based on international standards (as later shown in Table 3 in Sect. 5.1). As a result, the output of step 1 has formed the basis of shaping the content of our OSS-QMM.

4.2 Literature search-2 (step 2)

This step covers gathering the knowledge sources to be used, as in step 1. In order to develop a complete 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 have performed another SLR study (Yılmaz & Tarhan, 2022) and analyzed a total of 36 quality evaluation models or frameworks (QEMoF) proposed for OSS. This SLR study has examined the QEMoF from many aspects, but one of the most important findings is that there is little or no adoption of these models/frameworks in practice. Other secondary studies (Adewumi et al., 1936; Lenarduzzi et al., 2020) also support this situation. As revealed in the SLR study (Yılmaz & Tarhan, 2022), this is because quality models 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, the quantitative and qualitative evaluations, the aggregation techniques, the level of user skill in using the model, the data type of the evaluation results provided to the user, and so on. All this diversity has led to the proliferation of individual heterogeneous quality models. Therefore, in our current study, the OSS quality models are aimed to have a common structure in order to eliminate this heterogeneity. In this context, a total of 10 quality models have been determined, and the process of determining these quality models is explained in Sect. 2.2. The results of our second SLR study (Yılmaz & Tarhan, 2022) have revealed that most of the OSS quality models have a hierarchical structure, as already shown in Table 1. Therefore, we have observed that all the quality models are based on a common structure consisting of five levels (as later given in Table 4 in Sect. 5.2). As a result, the output of Step 2 has formed the basis of shaping the structure of our OSS-QMM.

4.3 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 has provided knowledge about the meanings of the terms to be used 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 concepts of the quality meta-models should be matched to the terms of the quality models since models are defined as instances of meta-models according to the MOF architecture (Facility and (MOF) 2019). In this regard, a level-based matching process has been carried out in iterations, as already shown in Fig. 5. That is, during the matching process, a series of meetings have been held between the authors of this article, and as a result of these iterations, the final version of the matching has been

Table 3 List of terms 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 14,598** **Kim** **IEEE 610.12**

Table 3 (continued)

Category of SQMM element		Properties of terms	Synonyms in different SQMMs	Synonyms in different standards	Source of terms	Standards defining terms differently
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**

Table 3 (continued)

Category of SQMM element		Properties of terms	Synonyms in different SQMMs	Synonyms in different standards	Source of terms	Standards defining terms differently
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 14,598:** 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**

Table 3 (continued)

Category of SQMM element		**Properties of terms**	**Synonyms in different SQMMs**	**Synonyms in different standards**	**Source of terms**	**Standards defining terms differently**
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				

Table 4 Structure comparison of software quality models (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

Table 5 Matching terms of quality models for OSS and terms of SQMMs with respect to 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)

The “*” symbol indicates that evaluation can be performed at any level with respect to the aggregations needs obtained (as later given in Table 5 in Sect. 5.3). While performing the matching, the meanings of the terms and their intended use have been taken into account.

4.4 Proposal of OSS-QMM (step 4)

In the fourth step, an initial proposal for a meta-model of OSS quality has been proposed by considering the outputs of steps 1, 2, and 3 (as later shown in Fig. 7 in Sect. 5.4). 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 model have been determined. Then, as seen in Fig. 5, the fourth step consists of three sub-steps, namely, steps 4.1, 4.2, and 4.3. It should be noted that in performing each sub-step, review-and-revise process has been followed. That is, a series of meetings has been held between authors of this article to review meta-model (i.e., step 4) and its development process (i.e., steps 1–3), and then to revise the meta-model. Also, in these meetings, refinements have been made as a result of the activities mentioned in steps 5 and 6. Thus, the final decisions have been taken as a result of a series of iterations.

First of all, in step 4.1, the candidate concepts to include in our OSS-QMM have been decided by considering the meaning of these concepts. After the review-and-revise process, a list of final concepts has been obtained. The first SLR study (Yılmaz & Tarhan, 2020) has indicated that the important meta-models (e.g., Garcia et al., 2007; Wagner et al., 2012) in the literature have a layered structure. That is, they group the terms in certain layers according to their content. Therefore, we have created three layers to group determined concepts in our OSS-QMM as *specification*, *measurement*, and *evaluation*. Also, this classification has made the matching process in step 3 more meaningful. In this context, in step 4.2, the determined concepts have been designated into one of these groups. After the review-and-revise process, the layer of each concept has been determined. Then, in step 4.3, the relationships between the concepts of the OSS-QMM have been determined. Association, specialization, and aggregation relationships have been used to link the concepts in the OSS-QMM. After the review-and-revise process, the final versions of the relationships between the concepts have been determined. For example, there is a *specialization* relationship between the concepts of OSS aspect and the concepts of code-based or community-based. As a result of these three steps including the iterations, the initial version of the OSS-QMM has been finalized.

4.5 Review by SQM experts (step 5)

In this step, the practical applicability, structure, and content of the OSS-QMM has been reviewed by experts on software quality models, using the suggestions by Tanrıover and Bilgen (2011) and Kläs et al. (2010). That is, it has been aimed to examine the OSS-QMM by external parties other than the authors of the article. In this context, a total of four subject matter experts has been determined, two from industry and two from academia. In determining the experts, a prerequisite has been applied that an expert would have 7 years or more experience in the field of software quality and its modeling. The first and the second experts with industry background have had 8 and 10 years of software quality modeling experience, respectively. The other two experts with academic background have been researchers lecturing and consulting on information systems and software engineering for more than 11 years. The questions listed in Appendix 1 were prepared in order to obtain feedback from these experts. As seen in this appendix, these questions have been aimed at obtain feedback about the practical applicability, structure, and content of the proposed OSS-QMM. Then, a series of online meetings have been held to discuss and gather answer for each question with the experts, each one interviewed individually. In these meetings, development process has been presented to the experts (i.e., steps 1–3), as well as the OSS-QMM itself together with the concepts, their meanings and intended use, and the relationships between these concepts. As shown in Fig. 5, this step has progressed by the iterations of review-and-revise process, with respect to the suggestions of the experts about the OSS-QMM. That is, the meta-model (i.e., step 4) and its development process (i.e., steps 1–3) have been reviewed by the expert for each question in Appendix 1 and the OSS-QMM has been revised in the 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 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, and the OSS-QMM is understandable. Thus, the OSS-QMM have matured and taken its final form with the feedback obtained for each question.

4.6 Initial validation of OSS-QMM (step 6)

In this step, an example implementation of our OSS-QMM has been performed (as later detailed in Sect. 5.5). Also, examples of an operationalized OSS quality model and an existing OSS quality model have been derived from the OSS-QMM, as given in Appendices 2 and 3, respectively. Therefore, this step could be considered as the initial validation process of the OSS-QMM. During this process, the meta-model has been reviewed to investigate whether there was an unmatched concept in the OSS-QMM or the derived quality models, whether there was a problem in the relationships between the concepts, and whether it could be applied in practice. Then, the meta-model has been revised in case that one of these situations was encountered. In this step, the OSS-QMM has been refined by new concepts and relationships, in iterations again. That is, this step has contributed to both validation and improvement of our OSS-QMM.

5 Development of OSS quality meta-model (OSS-QMM)

The development steps of the OSS-QMM are elaborated throughout this section in accordance with the methodology presented in Sect. 4. Also, using the background provided in Sect. 2, detailed analysis is performed on the concepts of SQMMs as well as the common structure of SQMs. Then, the level-based matching process is explained in detail. Next, the initial version of OSS-QMM is presented and its initial validation is explored over an example. Finally, potential threats to validity are discussed.

5.1 Terms analysis of SQMMs

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 custom type of software are analyzed. This is aimed to explore inconsistent terminology in the SQMMs for future proposals as well as to help us determine the terminology of the initial OSS-QMM presented in Sect. 5.4.

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 (Yılmaz & Tarhan, 2020). As a result, a total of 20 meta-models, 2 for OSS quality and 18 for the quality of custom type of software, have been analyzed in this study. 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) 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. 6 shows the percent distributions of the sources. This figure has been created based on the number of international standards or proposals which were taken as sources for the terms used in the SQMMs. These terms are listed in Table 3. 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 3. Also, if a term does not have any source, it is considered as "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 (Garcia et al., 2006). 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

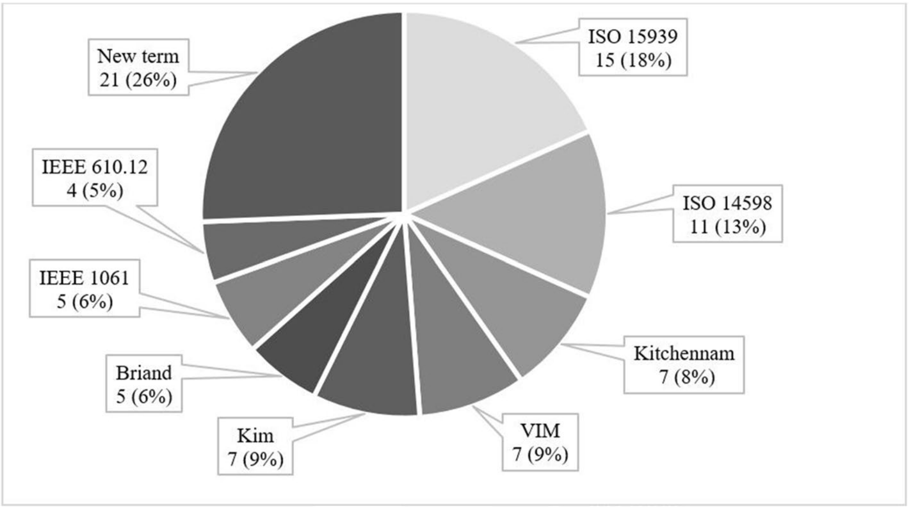


Fig. 6 Percent distribution of sources (standards and proposals) that contribute to the terminology in SQMMs

especially that the meta-models have not been validated by designing real-world cases, as previously mentioned in Sect. 2.1.

In Fig. 6, 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 were directly taken from the concepts of ISO/IEC 15939 that is 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 were not transferred from any standard or proposal. It is seen that the SQMMs employed more terms from ISO/IEC 15939 and ISO/IEC 14,598 since these are software measurement and software quality evaluation standards, respectively.

Despite the fact that there are studies conducted to investigate inconsistencies among the vocabulary of international standards as summarized in Sect. 3, no study has been found concerning the inconsistencies among the terms of the SQMMs. Therefore, Table 3 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 Yilmaz and Kolukısa Tarhan (2022). 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 second column, the terms are classified according to their properties that address how they were transferred from the sources. *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 third column of Table 3, 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 fourth 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 fifth column, the sources (standards or proposals) from which the SQMM terms in each category were adapted or adopted are listed. In the sixth and last column, the standards or proposals that differently described the SQMM terms in each category are listed. An important point here is that if a standard or proposal exists in the fifth column but not in the sixth column, it means that the term was used in the source but not defined in that source.

A term can have more than one definition by standards or proposals as also seen in Table 3. 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 3 shows that there is a lack of agreement even in the original sources to define a 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 were transferred from the sources directly (8 adopted terms) or with changes (9 adapted terms), and 21 (55%) of them were not transferred from any source (i.e., new term).

5.2 Structure analysis of SQMs, including OSS quality models

As we overview in Sect. 2.2, the quality models determined within the scope of this study are explained together with the reasons of 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 study, as listed in Table 4. 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 opportunity for OSS as well as form the basis of the OSS quality models. In Sect. 2.2.1.1, 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 1 are investigated and compared. This effort is important in shaping the structure of an initial OSS quality meta-model to develop within the scope of this study. It may also enhance the development of individual OSS quality models.

Considering the comparison of structures of the quality models listed in the table, 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, and designer.

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 and groups (product and application indicators). 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 the 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 (Sadeghzadeh & Rashidi, 2017). 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 quality measurement of the sub-attribute.

5.3 Matching between terms of OSS quality models and terms of SQMMs

Initial research has been conducted to investigate the structure of OSS quality models in Sect. 5.2, and it has been observed that all the quality models investigated have a common structure consisting of five levels as already given in Table 4. Also, in Sect. 5.1, the structure of the quality meta-models proposed for OSS and for custom type of software have been examined, and the terms used in these SQMMs have been categorized together with their synonyms and aggregations. In this way, it is aimed to understand the common structure of the quality models and to eliminate the inconsistencies among the terms of the SQMMs. In this section, 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 Table 5, since ideally, a quality model should be an instance of a meta-model. This matching has been performed in accordance with the MOF standard (Facility and (MOF) 2019). 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 3. 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 (ISO/IEC 19761, 2002; ISO/IEC 15939, 2007; ISO/IEC 25020, 2019) 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 (Nistala et al., 2019; Yılmaz & Tarhan, 2020), and a common output obtained from the definitions of these terms in the international standards have been taken as reference in this matching. Moreover, a series of meetings has been held between the authors of this article, following an iterative process, as already shown in Fig. 5. Considering all these sources, meta-model terms have been matched with the most appropriate terms in the levels of the OSS quality models.

In Table 5, 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. An 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 aggregations 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.

5.4 The OSS quality meta-model (OSS-QMM)

An initial proposal for a meta-model of OSS quality has been proposed, as shown in Fig. 7, by considering the outputs of the processes explained so far in this study. Common terminology for the initial meta-model proposal for OSS quality has been defined in order to help eliminate incompleteness and inconsistency available in the SQMMs. First of all, the terms of this initial meta-model have been determined by considering the most used terms of the quality meta-models proposed for OSS and custom type of software. Then, the level-based matching process, details of which are explained in Sect. 5.3, has been carried out between the terms of the OSS quality models and the terms of the meta-models. Finally, the initial proposal for a meta-model of OSS quality has emerged as a result of this matching process. The concepts used in the proposed meta-model have been demonstrated using color-coding in Fig. 7, with respect to the levels of terms in the OSS quality models as matched in Table 5. In addition, the categories (or stages, i.e., specification, measurement, and evaluation) of the terms used in the meta-model are also shown in the figure in accordance with the classification of terms given in Table 5. It should be reminded that during development of OSS-QMM, an iterative process has been followed. In this context, a series of meetings has been held between the authors of this article within the review-and-revise process, and refinements have been performed as a result of the issues identified during this process.

Detailed definitions of the terms used in the proposed meta-model are given in Yilmaz and Kolukısa Tarhan (2022), and definitions of the corresponding concepts will not be repeated here. Instead, the purposes that these concepts are used for and the relationships that exist between them in the meta-model are explained. Furthermore, in order to make the usage of the proposed meta-model more concrete, an example of an operationalized quality model for OSS (as an instance of the proposal) is given in Appendix 2. The intended uses of the terms included in this quality model are summarized in the following paragraphs.

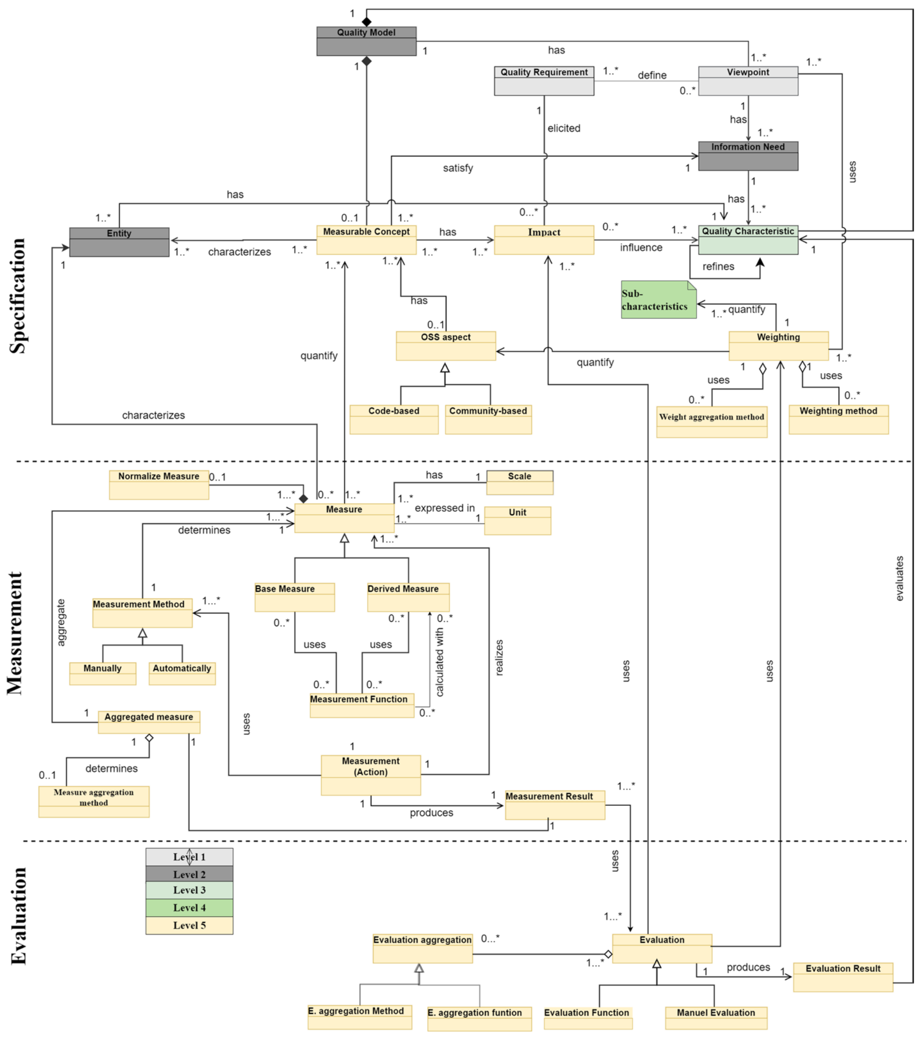


Fig. 7 Initial proposal for OSS-QMM

In the *specification* stage, a *quality model*, which is an instance of the proposed meta-model, is a set of measurable concepts and quality characteristics, and should have a viewpoint. The starting point of an OSS quality model is the *viewpoint.* Since stakeholders will perceive software quality from their points of view, the OSS quality model will be shaped with its content according to a specific viewpoint, e.g., of customer or developer. The OSS to be selected may be required for use as a component of a larger project and thus the software needs to be interoperable, or the OSS may be required for constant daily use by the adopter either in its original form or with modifications (Adewumi et al., 2019). Each stakeholder will have its own *information need* that is an insight necessary to manage objectives, goals, risks, and problems (e.g., request for calculation of e-mail density to evaluate maintainability, from the viewpoint of developer). An *entity* is an object to be characterized by measuring its attributes. The concept of entity is a part of a software product and is considered during the measurement process, e.g., source code (from code-based aspect) and contributor (from community-based aspect). An entity may have one or more properties that are of interest to meet an information need (ISO/IEC, 2007). An information need is related to *quality characteristics* that are used to control quality and to determine how well the software product or system satisfies the needs of its stakeholders. Quality characteristics are generally quite abstract concepts that cannot be measured directly (e.g., maintainability). Quality characteristics are decomposed into *sub-characteristics* (e.g., analyzability and testability) in quality models. However, sub-characteristics are still abstract to evaluate directly, so they can be considered as less abstract forms of quality characteristics. Therefore, sub-characteristics should be associated with *measurable concepts* which relate to one or more measures directly. The measurable concept covers the property of the OSS product that is associated with the quality of the product. It is defined in such a way that it is possible to talk about the extent to which it is present in the product. Also, it should satisfy the concept of information need. The measurable concept is characterized by an *OSS aspect* that can be specialized as *code-based* (e.g., the complexity of source code) or *community-based* (e.g., e-mail density by version, or bug solving success rate). There is an important point in the relationship between measurable concepts and sub-characteristics. That is, there can be a positive or negative relationship between them, called *impact*. This concept has a direct effect on evaluation results. For example, higher complexity of the code is undesirable for maintainability (indicating a negative effect), whereas higher bug solving success is desirable for maintainability (indicating a positive effect). To determine whether the effect is positive or negative, the need should be elicited from *quality requirements***.** Also, the importance of sub-characteristics and the OSS aspect may differ with respect to the viewpoint of an evaluator. For example, testability may be important from the viewpoint of test analyst, while modifiability may be important from the viewpoint of developer. Likewise, considering that the evaluator is a developer and will adapt the product according to his/her own needs, the code-based aspect may be more important. Therefore, the concept of *weighting* is employed to assign weights to sub-characteristics and OSS aspects using *weighting method* (e.g., AHP). Also, the importance of each sub-characteristic may be different for each OSS aspect, from a particular viewpoint. In this regard, the weights of the sub-characteristics should be distributed on the OSS aspects according to the importance of the aspects, by using the concept of *weight aggregation method.* The rationale behind this concept, for example, is that the evaluator is a developer and the code-based aspect is more important from this viewpoint. Therefore, the analyzability (as a sub-characteristic) of the source code (in code-based aspect) should have greater importance in this evaluation. In other words, the weight of analyzability on the code-based side should be greater than that on the community-based side.

In the *measurement* stage, after specifying the measurable concept, the concept should be quantified by *measures* to obtain concrete results. A measure can be a *base measure* (e.g., m_1: line of documented source code, and m_2: effort spent) or a *derived measure* (e.g., productivity represented by m_3: m_1/m_2) that is generated using a *measurement function.* Also, each measure should have a *scale* (e.g., integers from zero to infinite) and *unit* (e.g., defects, days, or lines). After measures are specified to quantify the measurable concept, *measurement method* determines how to assign value for a specified measure. The measurement method can be applied *manually* (e.g., counting measures from the website of OSS repository such as GitHub) or *automatically* (e.g., over static code analyzer tool as an instrument). Since measurement data are accessible from both code-based and community-based aspects of OSS, these data are scattered in a variety of databases and thus heterogeneous. Therefore, the concept of *normalized* *measure* is used to make measurement result meaningful and also comparable with those of different products. In case more than one measure is used to quantify a measurable concept, the normalized measures under the measurable concept are aggregated by *aggregated measure* using *measure aggregation method* (e.g., average). Then, a one-to-one relationship is obtained between measurable concept and measure. After the necessary infrastructure for the measurement process is determined, *measurement* is realized as an action and *measurement result* is produced as an output of the measurement performed.

In the *evaluation* stage, concepts are used to interpret the measurement results, and accordingly, to identify whether the quality of the OSS product fits the needs of the evaluator. The resulting values, impacts, and weights of sub-characteristics are used as inputs for *evaluation* which helps interpreting measurement results. All these inputs are aggregated using *evaluation aggregation* consisting of *evaluation aggregation method* (e.g., TOPSIS) and *evaluation aggregation function* (e.g., weighting aggregation function). The concept of evaluation can be specialized by *evaluation function* (e.g., linear utility function) or *manual evaluation* (e.g., expert opinion). After measurement results are interpreted by the concept of evaluation, an *evaluation result* between 0 to MaxPoint is obtained (e.g., range between 0 and 1). By this way, the quality of the OSS product concerning its specified quality characteristics is evaluated.

5.5 Initial validation of OSS-QMM

In this section, initial validation of the OSS-QMM is provided over an example evaluation. An operationalized OSS quality model has been derived from the proposed meta-model, as given in Appendix 2, and the stages of the quality evaluation using this model are explained. In this evaluation, an example is demonstrated to identify the OSS product that best meets the evaluator's needs among the alternatives. Quality evaluation scores are calculated for each alternative OSS product, since one of the best ways to understand the quality of a product is to compare it with those of possible alternatives. All these efforts are targeted to demonstrate the application of the OSS-QMM, allowing it to be visualized and better understood. In order to increase the traceability of the implementation of the quality model, this section also provides the techniques (i.e., methods, formulations, or functions) to use in the example evaluation, as explained in Sect. 5.5.1.

In addition, three existing OSS quality models (OpenBRR, OSMM, and SQO-OSS) have been derived from the OSS-QMM. In doing this, the terms of these models have been matched with the concepts of the OSS-QMM, as provided in Appendix 3. Since the purpose is to demonstrate that the existing models can be derived from the proposed meta-model, only the determined terms belonging to the levels of the models (i.e., characteristics, sub-characteristics, measure, etc.) have been used as examples in the matching process. That is, not all terms in each level of these models have been used. For example, maintainability is decomposed into four sub-characteristics in the OSMM, but only one of them (i.e., integration) is used in the matching process in the appendix. This applies to any level of matching. In addition, the concepts of the OSS-QMM listed in Table 6 (i.e., techniques) are not shown in the matching given in Appendix 10. since these concepts enable to perform some calculations by using some other concepts (impact, measure, OSS aspect, etc.) matched in the appendix.

Table 6 List of techniques to use for the relevant concepts in the OSS-QMM

Concept of OSS-QMM	Techniques to use in example evaluation
Weighting method	Integrated AHP-TOPSIS method (AHP, steps 1–7)
Weight aggregation method	Weighted distribution
Normalized measure	Integrated AHP-TOPSIS method (TOPSIS, step 2)
Measure aggregation method	Average of the measure
Measurement function	Some mathematical equations
Evaluation aggregation method	Integrated AHP-TOPSIS method (TOPSIS, steps 3–7)
Manual evaluation	Expert opinion

5.5.1 Techniques to use in example evaluation

In order to use the OSS-QMM in practice, some techniques should be determined for the concepts of the OSS-QMM (e.g., evaluation aggregation, weighting method) prior to quality evaluation. In this regard, the list of methods to use in the example evaluation is shown in Table 6. In the following sub-sections, information about these techniques is presented.

5.5.1.1 Integrated AHP-TOPSIS The OSS products have a diverse and dynamic nature, and their quality is affected by many variables. That is, a lot of quantitative and qualitative historical data can be accessed in both code-based and community-based aspects for the evaluation of the OSS products. Accordingly, aggregating heterogeneous data from different sources is a difficult process. Therefore, evaluation of OSS has been considered as a Multi-Criteria Decision Making (MCDM) problem (Jadhav & Sonar, 2011) to deal with these complex and diverse data sources. MCDM allows managing multiple complex and conflicting objectives to be used in the evaluation and accordingly assign quality scores to alternatives (Wang et al., 2013). A number of well-known MCDM methods exist such as Analytic Hierarchy Process (AHP), Analytic Network Process (ANP), Data Envelopment Analysis (DEA), DEMATEL, Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), VIKOR, and PROMETHEE. In the MCDM problem, the most widely used methods are AHP (Saaty, 1980) and TOPSIS (Hwang & Yoon, 1981), due to their strong mathematical background and systematic way of data collection (Ahmad & Laplante, 2013; Khondoker et al., 2014). They have been applied and validated in numerous multidisciplinary fields since their initial development such as engineering (Özcan et al., 2017), economics (Wang et al., 2020), and social science (Saaty & Sagir, 2015). While the AHP method is employed for weighting a set of alternatives by performing the pair-wise comparison, the TOPSIS method is employed for some determined criteria for weighting a set of alternatives.

In this study, in order to cope with the diverse and conflicting data sources of the OSS products, the integrated AHP-TOPSIS method shown in Fig. 8 has been used to perform quality evaluation following the concepts in the proposed meta-model. More specifically, AHP method is used to weight the criteria in the specification part (i.e., for code-based and community-based aspects), and TOPSIS is used to evaluate the alternatives by using these weights as inputs. The weights obtained from the AHP method, the results obtained from the concept of *impact*, and the measurement results obtained are

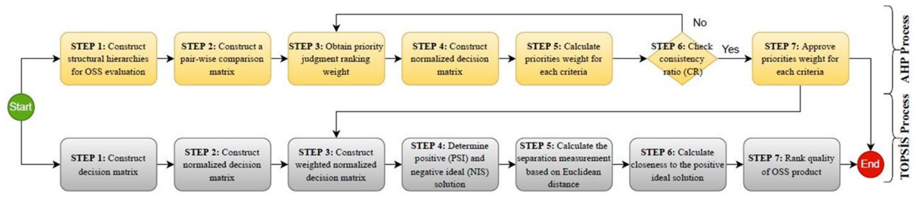

Start
STEP 1: Construct structural hierarchies for OSS evaluation
STEP 2: Construct a pair-wise comparison matrix
STEP 3: Obtain priority judgment ranking weight
STEP 4: Construct normalized decision matrix
STEP 5: Calculate priorities weight for each criteria
STEP 6: Check consistency ratio (CR)
No
Yes
STEP 7: Approve priorities weight for each criteria
AHP Process
STEP 1: Construct decision matrix
STEP 2: Construct normalized decision matrix
STEP 3: Construct weighted normalized decision matrix
STEP 4: Determine positive (PSI) and negative ideal (NIS) solution
STEP 5: Calculate the separation measurement based on Euclidean distance
STEP 6: Calculate closeness to the positive ideal solution
STEP 7: Rank quality of OSS product
End
TOPSIS Process

Fig. 8 Quality evaluation process of integrated AHP-TOPSIS

used as inputs for the TOPSIS method. Thus, the quality evaluation is performed using the strengths and validated aspects of both methods.

The steps of integrated AHP TOPSIS, which is the one of most important techniques used in initial validation, are explained above, and each step described in Fig. 8 consists of equations. Aside from integrated AHP-TOPSIS, other techniques given in Table 6 are also expressed by equations. The equations and brief descriptions of these techniques used in initial validation are given in Table 7. In the next sub-section, the application of these equations during initial validation is explained in detail.

5.5.2 Carrying out example evaluation

In this section, the use of the OSS-QMM (Fig. 7) to carry out OSS quality evaluation is demonstrated over an example. For this example, a quality model has been derived from the OSS-QMM, as shown in Appendix 2. It should be reminded that the concepts of the OSS-QMM are classified as specification, measurement, and evaluation. Respectively, in the following paragraphs, the meta-model concepts, which are shown in italics and bold to increase traceability, are explained in accordance with this classification.

In the *specification* part of the OSS-QMM, considering the quality model given in Appendix 2, assume for the example evaluation that three OSS products are selected as P1, P2 and P3; the developer as the ***viewpoint***; maintainability as the ***characteristic***; and analyzability (**A**), testability (**T**), and modifiability (**M**) as the ***sub-characteristics***. Also, assume that the ***measurable concepts*** (MC) for measuring sub-characteristics and the ***impacts*** of these measurable concepts on the sub-characteristic are determined using ***quality requirements.*** Apart from them, assume that the ***entity*** and ***information needs*** are determined. For example, on the code-based aspect considering analyzability, the measurable concept might be the "complexity of the source" code. Accordingly, the entity is the `"source code," information need is "calculation of source code complexity to evaluate maintainability," and impact is negative since "complexity of the source" affects the analyzability negatively. Then, the evaluator assigns weights to both the sub-characteristics and the OSS aspects (i.e., code-based and community-based) according to his/her perceived importance, by using the concept of ***weighting***. In this example, AHP method (steps 1–7) is used as the ***weighting method*** as already shown in Table 6. The steps of the AHP method are given in Fig. 8 and its application is explained in Sect. 5.5.1.1 (see Saaty, 1980, 2008; Işıklar & Büyüközkan, 2007 for details). The outputs obtained at each step of AHP method for the example evaluation are provided below.

Table 7 Description of techniques with equations used in initial validation

Technique	Description	Equation	
AHP	The AHP method consists of the following steps. Please see (Saaty, 1980; Işıklar & Büyüközkan, 2007; Saaty, 2008) for details		
Step 1	First, structural hierarchies are created. The concepts of OSS aspect and quality characteristic provide this condition	No equation	
Step 2	A pair-wise comparison matrix A (size $n \times n$) is constructed to compare the criteria in pairs. Each OSS aspect and related sub-characteristics are as "criteria"	$A = [x_{ij}]_{n\times n}$ Matrix A is pair-wise comparison matrix	
Step 3	Pair-wise comparisons are performed by comparing relative importance of two selected criteria. The matrix A is filled by using the scale 1–9, as proposed by Saaty (Saaty, 1980) (see (Işıklar & Büyüközkan, 2007) for details)	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	$A_{norm} = [a_{ij}]_{n\times n} = {}^{x_{ij}} / _{\sum_{i=1}^{n} x_{ij}}$	**(1)**
Step 5	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	Consistency ratio (CR) is calculated to check consistency of decision-maker's judgment. First, Consistency Index (CI) is calculated, where $\lambda_{\max}$ is the eigenvalue (see (Saaty, 2008; Işıklar & Büyüközkan, 2007) 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 (Saaty, 2008; Işıklar & Büyüközkan, 2007) for values of RI according to n)	$\boldsymbol{CI} = \left(\lambda_{max} - n\right)/(n-1)$ $\boldsymbol{CR} = CI/RI$	**(3)** **(4)**
Step 7	Final weight of each criterion is approved	No equation	
TOPSIS	The final weight of each criterion obtained from AHP method is used as input to TOPSIS method. The TOPSIS method consists of the following steps (please see Hwang & Yoon, 1981; Hasnain et al., 2020; Işıklar & Büyüközkan, 2007 for details)		
Step 1	First, decision matrix B = $[b_{ij}]_{m\times n}$, 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 $= [b_{ij}]_{m\times n}$ Matrix B is decision matrix	
Step 2	Normalized decision matrix R = $[r_{ij}]_{m\times n}$ is constructed	$\boldsymbol{R} = [r_{ij}]_{m\times n} = b_{ij}/\sqrt{\sum_{i=1}^{m} b_{ij}^2}$ $i = 1, 2, 3 \dots m$; and $j = 1, 2, 3 \dots n$	**(5)**

Table 7 (continued)

Technique	Description	Equation	
Step 3	The final weights obtained from AHP method are multiplied by the values of the normalized decision matrix R. Thus, weighted normalized decision matrix V = $[v_{ij}]_{m\times n.}$ is obtained	$V = \left[v_{ij}\right]_{m\times n} = w_j \times r_{ij}$ $i = 1, 2, 3 \ldots m$; and $j = 1, 2, 3 \ldots n$	**(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. (7), 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	$A^+ = \{(max_i v_{ij} \| j \in J)(min_i v_{ij} \| j \in J^-)$ $\| i = 1, 2, 3, \ldots m\} = \{v_1^+, v_2^+, \ldots v_j^+, \ldots v_n^+\}$	**(7)**
		$A^- = \left\{\left(min_i v_{ij} \middle\| j \in J\right)\left(max_i v_{ij} \| j \in J^-\right)\right.$ $\| i = 1, 2, 3, \ldots m\} = \{v_1^-, v_2^-, \ldots v_j^-, \ldots v_n^-\}$	**(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. (7) and Eq. (8), respectively. At the end of step 5, two values, namely, S$^+$ and S$^-$, for each alternative has been counted; these two values represent the distance between each alternative and both the ideal and negative ideal	$S_i^+ = \sqrt{\sum_{j=1}^n \left(v_{ij} - v_j^+\right)^2}, i = \{1, 2, 3 \ldots m\}$	**(9)**
		$S_i^- = \sqrt{\sum_{j=1}^n \left(v_{ij} - v_j^-\right)^2}, i = \{1, 2, 3 \ldots m\}$	**(10)**
Step 6	In this process, the closeness of A$_i$ (ith alternative) to the ideal solution A^+ is defined, as shown in Eq. (7). $C_i^* = 1$ if and only if $A_i = A^+$; similarly, $C_i^* = 0$ if and only if $A_i = A^-$	$C_i^* = S_i^- / \left(S_i^- + S_i^+\right)$ $0 < C_i^* < 1, i = \{1, 2, 3 \ldots m\}$	**(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 distrib	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	$X_i = (w_i^a \times w_j^s) / \sum_{i=1}^n w_i^a$ $\sum_{i=1}^n w_i^a = 1$ (see Eq. (7)) $i = \{1or2\} j = \{1, 2, 3 \ldots m\}$	**(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 that is obtained from M1 and M2 by using following mathematical equation: M3 = M1/(M1 + M2). Therefore, this equation corresponds to the concept of measurement function in the OSS-QMM	It can be different kinds of equations	

Table 7 (continued)

Technique	Description	Equation
Average of the measures	In cases that multiple measures are associated with a measurable concept, these measures should be aggregated. In this aggregation process, the normalized measures (obtained in step 2 of TOPSIS) associated with a measurable concept are averaged Here, p is the number of alternative (OSS product), $m_{(k)}$ is new value of measures associated with a measurable concept for kth 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 measurable concept, respectively	$m_{(k)} = \sum_{j=1}^{n} r_{ij}/n$ $i = \{m \ldots n\} k = \{1, 2, 3 \ldots p\}$ **(13)**
Expert opinion	The evaluation results are manually interpreted according to the needs of the evaluator, after the results are obtained	No equation

Table 8 **A** Pairwise comparison results, **B** values of normalized decision matrix, and **C** weights of sub-characteristics w.r.t importance

(A) $A =$ ($n \times n$)

	A	T	M
A	1	2	5
T	1/2	1	4
M	1/5	1/4	1
Total	1.7	3.25	10

(B) $A_{norm} =$ ($n \times n$)

	A	T	M
A	0.588	0.620	0.500
T	0.294	0.31	0.400
M	0.118	0.08	0.100

(C)

	WEIGHT
A	**0.568**
T	**0.334**
M	**0.098**
CR	**0.0212**

A: Analyzability, **T**: Testability, **M**: Modifiability, and **CR**: Consistency ratio

- *Step 1*: Since maintainability is decomposed into three sub-characteristics and quality evaluation is performed in two OSS aspects, the necessary hierarchy for AHP implementation is provided.
- *Step 2*: The pairwise comparison matrix A (size $n \times n$) is constructed to compare criteria (e.g., OSS aspect and sub-characteristics) in pairs as shown in Table 8(A).
- *Step 3*: A scale from 1 to 9 is used for ratings in the AHP method (please see (Saaty, 1980) for details) for the evaluator's judgments in pairwise comparison. The matrix A is filled out as a result of the pairwise comparison is given in Table 8(A).
- *Step 4*: The normalized decision matrix A_{norm} is given in Table 8(B). To find the normalized values, the value in each cell is divided by the total value at the bottom of the column, as shown in Eq. (7).
- *Step 5*: The weights of the sub-characteristics with respect to their importance are calculated as shown in Table 8(C). These weights are calculated by using Eq. (7).
- *Step 6*: The Consistency Ratio (CR) value is calculated to measure the consistency of the judgments of decision-makers by using Eqs. (3, 4), as shown Table 8(C). The calculation of CR value is explained in detail in the studies (Işıklar & Büyüközkan, 2007; Saaty, 2008) and is not repeated here to save space.
- *Step 7*: The value of CR must be less than 0.1 for the result of AHP be accepted as consistent (Işıklar & Büyüközkan, 2007). If this is not the case, the pairwise comparison (step 3) should be revised. In this regard, since the CR value (0.0212) is less than 0.1 as shown in Table 8(C), the weights of sub-characteristics are consistent.

Assuming that the AHP method is applied according to the developer's viewpoint and at the end of the process, a weight of 0.650 is obtained for the code-based aspect and a weight of 0.350 is obtained for the community-based aspect, as shown in Table 9(B).

Table 9 **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

(A)

	WEIGHT
A	**0.568**
T	**0.334**
M	**0.098**

(B)

	WEIGHT
CODE-BASED	**0.650**
COMMUNITY-BASED	**0.350**

Apply Eq. (12)

(C)

OSS aspect	Code-based (0.650)			Community-based (0.350)		
Sub-characteristic	A (0.568)	T (0.334)	M (0.098)	A (0.568)	T (0.334)	M (0.098)
Final weight	0.369	0.217	0.063	0.198	0.116	0.034

A: Analyzability, **T**: Testability, **M**: Modifiability

Next, the final weight of each sub-characteristics affecting each OSS aspect is calculated as shown in Table 9(C) by using the concept of *weighting aggregation method*. While determining these weights, weighted distribution formula shown in Eq. (7) is used as the concept of the weighting aggregation method. The rationale behind this formula 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 aspects. For example, assume that the evaluator is a developer and the code-based aspect is more important for this viewpoint. Accordingly, the analyzability (i.e., sub-characteristic) of the source code (in code-based aspect) should be of greater importance for this evaluator. In other words, the weight of analyzability in the code-based side should be greater than that in the community-based side.

In the *measurement* part of the OSS-QMM, the concepts are used to quantify the quality of an OSS product via measures belonging to code-based and community-based aspects. Assume that the *measures (M)* are determined for each measurable concept as shown in Table 10. Assume that the type of the specified measures is *base measure* and no mathematical equation is used for a *derived measure*. Then, they are computed by using a *measurement method*. In this case, the values of these measures can be computed *automatically* or *manually*. Assume that all measures are computed manually for this example. After this stage, the TOPSIS method is used, which allows assigning weights to the OSS products at the end of the evaluation. The steps of the TOPSIS method are already shown in Fig. 8 and its application is explained in Sect. 5.5.1.1 (see Hwang & Yoon, 1981; Işıklar & Büyüközkan, 2007; Hasnain et al., 2020 for details). The outputs obtained at each step of the TOPSIS method for the example evaluation are provided in the items below.

- *Step 1*: After obtaining the values of code-based and community-based measures for the OSS products (i.e., P1, P2, and P3), the decision matrix $\mathrm{B} = [b_{ij}]_{m \times n}$ (TOPSIS, step 1) is constructed as represented in Table 10. In this matrix, m is the number of alternatives (i.e., OSS products) and n is the number of evaluation criteria (i.e., the measures).
- *Step 2*: Since *scales* and *units* for the measures can be heterogeneous, the values of the measures must first be normalized. In this context, they are normalized using the concept of *normalized measure*. In the normalization stage, Eq. (7) is applied to the decision matrix ($\mathrm{B} = [b_{ij}]_{m \times n}$) and as represented in Table 11(A), normalized decision matrix $\mathrm{R} = [r_{ij}]_{m \times n}$ is obtained (TOPSIS, step 2).

If a measurable concept (MC) is associated with more than one measure, these associated measures should be aggregated by using the concept of *aggregated measure*. In this regard, Eq. (7) is applied as the concept of *measure aggregation method* for the measures associated with MC1 and MC6 shown in Table 11(A). Then, *measurement* is performed as action, and *measurement results* are produced for each measurable concept before the evaluation phase.

In the *evaluation* part of the OSS-QMM, the concept of *evaluation* uses three inputs to initiate the evaluation: final weights of the sub-characteristics, impacts, and measurement results to start the evaluation. These three inputs should be aggregated by using the concept of *evaluation aggregation*. In the example evaluation, the concept of the *evaluation aggregation method* is used to aggregate these three inputs. That is, the integrated AHP-TOPSIS

Table 10 Final weights of sub-characteristics, impact of measurable concept (MC) on sub-characteristics, measures related to MC, measure values for each product, and decision matrix

Aspect	Code-based				Community-based				
Sub-characteristic	Analyzability (A)		Testability (T)	Modifiability (M)	Analyzability (A)	Testability (T)	Modifiability (M)		
Final weight	0.369		0.217	0.063	0.198	0.116	0.034		
Impact	Negative (−)		Negative (−)	Positive (+)	Positive (+)	Positive (+)	Positive (+)		
Measurable concept	MC1		MC2	MC3	MC4	MC5	MC6		
Measure	$M1_1$	$M1_2$	M2	M3	M4	M5	$M6_1$	$M6_2$	$M6_3$
Product 1	1.602	1.634	17.235	2.365	0.785	0.545	356	310	309
Product 2	2.287	1.975	20.654	2.125	0.556	0.523	211	287	327
Product 3	1.986	2.024	27.256	1.075	0.876	0.723	220	205	235

Table 11 **A** Normalized decision matrix, **B** weighted normalized decision matrix, and **C** value of PIS and NIS

(a)

Final Weight		0.369	0.217	0.063	0.198	0.116	0.034
Impact		-	-	+	+	+	+
MC		MC1	MC2	MC3	MC4	MC5	MC6
M		M1	M2	M3	M4	M5	M6
P1		0.483	0.450	0.705	0.603	0.521	0.676
P2	**R =**	0.636	0.539	0.633	0.427	0.500	0.568
P3		0.599	0.712	0.320	0.673	0.691	0.456

(b)

Impact		-	-	+	+	+	+
MC		MC1	MC2	MC3	MC4	MC5	MC6
M		M1	M2	M3	M4	M5	M6
P1		0.178	0.098	0.045	0.120	0.061	0.023
P2	**V =**	0.235	0.117	0.040	0.085	0.058	0.019
P3		0.221	0.154	0.020	0.134	0.081	0.016

(c)

v^+(PIS)	0.178	0.098	0.045	0.134	0.081	0.023
v^-(NIS)	0.235	0.154	0.020	0.085	0.058	0.016

MC: Measurable concept, **M**: measure, **P1**: Product 1, **P2**: Product 2, **P3**: Product 3, **PIS**: Positive ideal solution, and **NIS**: Negative ideal solution

method (TOPSIS, steps 3–7) is used as already specified in Table 6. The remaining steps of the TOPSIS are provided in the items below.

- *Step 3*: The final weight of each sub-characteristic (in Table 11A) is multiplied by each associated normalized measure by using Eq. (7), in order to obtain the weighted normalized matrix $V = [v_{ij}]_{m \times n}$ shown in Table 11(B).
- *Step 4*: Positive Ideal Solution (PIS) and Negative Ideal Solution (NIS) are determined (as shown in Table 11C) from the weighted normalized matrix by considering the impacts (i.e., (+) or (−)). The equations for calculating PIS and NIS are given in Eq. (7) and Eq. (7), respectively. If the impact is negative, the PIS value is the minimum of the normalized measure in the associated column as shown in Table 11(B); if it is positive, it is the maximum one, and vice versa for the NIS value.
- *Step 5*: Separation measurement based on Euclidian distance for PIS and NIS (i.e., S_i^+ and S_i^-) are calculated with Eq. (7) and Eq. (7), respectively, as shown in Table 12(A).
- *Step 6*: Each OSS product is assigned a final quality score using Eq. (7) in Table 12(B).
- *Step 7*: The OSS products are ranked according to their final quality score, indicating that a higher value corresponds to better quality.

The final quality score for each OSS product obtained from the concept of the evaluation aggregation method can be interpreted with the concepts of *evaluation function* or *manual evaluation*. In this example evaluation, expert opinion is used as the concept of manual evaluation. In this context, the evaluation results are interpreted by an expert. Considering the ranks of the products shown in Table 12 (B), the most preferable product in terms of maintainability is determined as P1, and it is followed by P3 and P2, respectively. Also, considering the final scores, the difference in quality between P1 and P3 is greater than the difference in quality between P2 and P3.3

Table 12 **A** Separation measurements for PIS and NIS, and **B** final weights of OSS products and ranks

(a)

	S_i^+	**S_i^-**
P1	0.024284	0.071388
P2	0.080561	0.042505
P3	0.075499	0.053774

(b)

	Final Product weight (Closeness to PIS)	**Rank of product**
P1	0.7170	1
P2	0.3032	3
P3	0.4696	2

P1: Product 1, **P2**: Product 2, **P3**: Product 3, **S_i^+** and **S_i^-** : Separation measurements

5.6 Discussion

In this study, an initial proposal for OSS-QMM has been made by considering the *step-based meta-modeling creation process* (Beydoun et al., 2009; Othman et al., 2014). Despite the fact that a systematic process has been followed for developing the OSS-QMM, several threats to its validity might have arisen regarding understandability and potentially overlooked concepts of the OSS-QMM. To cope with the former, all the concepts and steps used in the development of the OSS-QMM are explained in detail with their sources in this study, in order to increase the understandability of the meta-model. The concepts of the OSS-QMM are classified by specification, measurement, and evaluation, in order to make them more meaningful for the corresponding stakeholders. The OSS-QMM is aimed to be used by all users in need, e.g., requirements analysts, project leaders, developers, and quality assurance staff. However, it may be difficult for an evaluator without quality modeling and evaluation background to understand the OSS-QMM and use it in practice. To alleviate this threat, a guidance document on the use of the OSS-QMM will be developed.

As explained in Fig. 5, development process of the OSS-QMM is systematic and includes iterations. In this process, the development of the OSS-QMM has been carried out with the help of the related studies in literature. However, it is possible that there are concepts that have been overlooked or misused. To alleviate this threat, a series of meetings have been held between the authors of this article to discuss the matching process, concepts and their relationships, and the structure of the OSS-QMM. In addition, during the initial validation that includes example evaluation and derivation of existing OSS quality models, the practical applications (i.e., instantiations) of the OSS-QMM have been experienced. As a result, the shortcomings in the meta-model have been observed throughout these activities and revised as necessary. Moreover, external feedback from the SQM experts, two each from of industry and academia, has been received to strengthen the confidence in this initial validation of the OSS-QMM. Detailed information has been provided to the experts about the development process and content of the OSS-QMM. Although the experts have found the content and implementation of the OSS-QMM sufficient, we admit that it may be difficult to come up with a completely correct interpretation without applying it in practice. In this context, experts have been asked to use the OSS-QMM for deriving their own quality models (either an individual model used in their company or an existing model). However, this is an ongoing process which is left as an upcoming work. That is, although the OSS-QMM is built by providing a solid base, it is possible to make minor refinements to some concepts and, thus, the content of the meta-model may slightly change in the future.

6 Conclusion and future work

The motivation for this study was the lack of meta-models for OSS quality and the inconsistent terminology among the existing general-purpose SQMMs. Therefore, the main achievements of this study have been to eliminate the inconsistency in terminologies of the SQMMs, to make a matching between the terms of the SQMMs and the terms of the quality models for OSS, to provide an initial version of the OSS-QMM, and as a result, to provide the opportunity for developing OSS quality meta-models and models that enable comparable measurements.

In this context, first of all, meta-models have been examined in detail by performing an SLR study (Yılmaz & Tarhan, 2020), and thus deficiencies have been discovered in this domain and the current status of the meta-models for OSS quality has been analyzed. Then, another SLR study (Yılmaz & Tarhan, 2022) 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. Following that, 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, 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. Next, an initial version of the OSS-QMM has been proposed to help eliminate incompleteness and inconsistency in several SQMMs. During the development of the meta-model, an iterative process has been followed to refine the OSS-QMM. Finally, an example instantiation of the meta-model of OSS quality has been illustrated.

It is aimed that this initial version of the OSS-QMM motivates developing further OSS quality models with homogenous structure and common terms, by using a same modeling language defined by the proposed meta-model. Also, it is aimed that this meta-model contributes to the standardization of OSS quality measurements, which in turn will provide an important communication vehicle to companies in interoperating. However, the lack of consistency and completeness identified in the international standards and accordingly in the SQMMs may negatively affect this standardization. Therefore, in this study, a comparative analysis among the terms of the SQMMs has been carried out to identify inconsistent terminology in software quality measurement. Then, a common terminology provided by an initial OSS-QMM has been proposed to help eliminate incompleteness and inconsistency available in the SQMMs.

Nevertheless, it cannot be claimed that the effort spent in this study and the initially proposed meta-model 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. It can serve as a guide for evaluators who are confused by inconsistent terminology in the existing SQMMs or international standards, and also for meta-model developers who will propose new SQMMs or integrate consistent terms into the developed SQMMs.

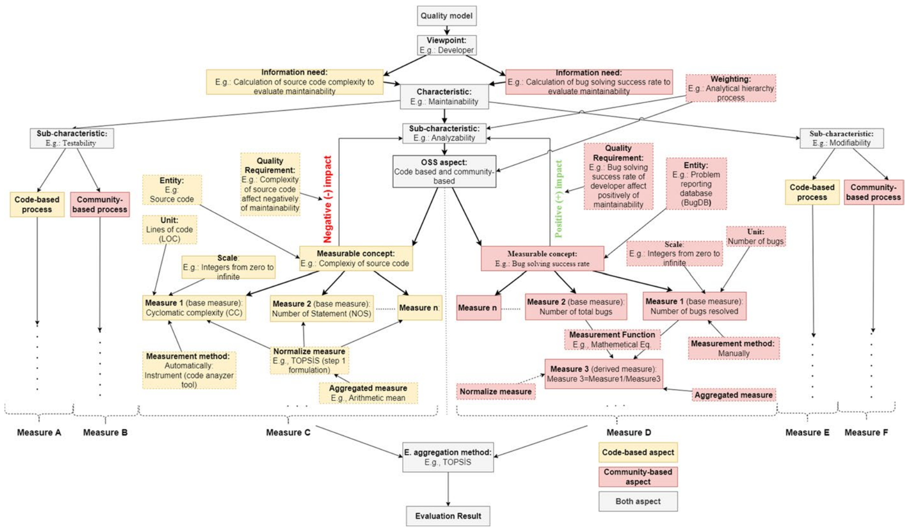
As future work, in the light of the efforts spent and the output produced in this study, we aim to elaborate further on this initial proposal of the OSS-QMM. Since the stage categories of the meta-model have processes of their own, we aim to detail each category by visualizing it with further UML diagrams. Also, it should be noted that the empirical validation of the OSS-QMM is currently in progress. In this context, the following activities are planned. First, we will apply the new operationalized quality model, which has been derived from the OSS-QMM and used in example evaluation in this study, to several OSS products in real-world cases. Then, we will instantiate the widely known OSS quality models from the proposed OSS-QMM and use them to evaluate the same OSS products in real-world cases. Consequently, we will compare the results of the operationalized quality model with those obtained by the widely known OSS quality models. Also, we plan that companies will derive their own quality models from the OSS-QMM and implement them in practice without our intervention. Furthermore, we will develop a guidance document on how to use the proposed meta-model so that it can be taken as a base by interested stakeholders. Finally, we aim to develop a tool to automate the use of the OSS-QMM.

Appendix 1 List of questions to obtain feedback from experts

	Feedback on the applicability of OSS-QMM in practice	**Purpose of question**
Q1	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 meta-model terms would you match?	It is asked to see the usefulness of the concepts in our meta-model and find out if there are any unused concepts
Q2	As a result of the matching you performed in the previous question (Q1), are there any unused concept of our OSS-QMM?	It is asked to see if there is a missing concept in our meta-model and get feedback from experts in this context
Q3	Do you agree that the terms of the example OSS quality model given in Appendix 2 (derived from OSS-QMM) and the terms of the OSS-QMM are compatible with each other?	Experts are asked to derive an OSS quality model using our meta-model. Then, it is asked to get feedback on the compatibility of the model they derived with the model we derived (Appendix 2)
	Feedback on the structure of OSS-QMM	
Q1	Do you agree that the mapping process is compatible with the (5-level) structure of the OSS-QMM?	It is asked to get feedback on the compatibility of the matching process with the structure of the meta-model
Q2	Do you agree that the classification of the OSS-QMM terms (under specification, measurement, and evaluation) are useful?	It was asked to get feedback on whether the classification process contributes to a better understanding of our meta-model and the coexistence of compatible concepts
Q3	Do you agree that the OSS quality models to be derived from the OSS-QMM will have homogeneous structure?	In order to benefit from the software quality modeling experiences of the experts, they are asked for their opinions on whether the structure of the models to be developed from our meta-model would be homogeneous
Q4	Do you agree that the OSS-QMM is understandable?	Experts are asked whether the developed meta-model was understandable and their feedback to make it more understandable
	Feedback on the content of OSS-QMM	

	Feedback on the applicability of OSS-QMM in practice	Purpose of question
Q1	Do you agree that the OSS-QMM is sufficiently general to describe any existing OSS quality model that you already know? (e.g., OSMM, SQO-OSS, and QualOSS)?	Experts are asked to derive an existing OSS quality model they know using our meta model. Then, it is asked whether there were deficiencies in our model
Q2	Do you agree that that relationship between concepts is compatible?	It is asked to get feedback on whether the relationships used between concepts were compatible with the concepts and the structure of our meta-model
Q3	Do you agree that the OSS-QMM is complete?	Considering the structure of the meta-model, its concepts, and the relationships among them, it was asked to get feedback on whether quality models derived from our meta-model address all aspects of OSS products (e.g., code-based and community-based aspects)

Appendix 2 The new operationalized quality model derived from OSS-QMM


Quality model
Viewpoint: E.g.: Developer
Information need: E.g.: Calculation of source code complexity to evaluate maintainability
Information need: E.g.: Calculation of bug solving success rate to evaluate maintainability
Weighting: E.g.: Analytical hierarchy process
Characteristic: E.g.: Maintainability
Sub-characteristic: E.g.: Testability
Sub-characteristic: E.g.: Analyzability
Sub-characteristic: E.g.: Modifiability
OSS aspect: Code based and community-based
Code-based process
Community-based process
Entity: E.g: Source code
Quality Requirement: E.g.: Complexity of source code affect negatively of maintainability
Negative (-) impact
Positive (+) impact
Quality Requirement: E.g.: Bug solving success rate of developer affect positively of maintainability
Entity: E.g.: Problem reporting database (BugDB)
Unit: Lines of code (LOC)
Scale: E.g.: Integers from zero to infinite
Measurable concept: E.g.: Complexiy of source code
Measurable concept: E.g.: Bug solving success rate
Unit: Number of bugs
Measure 1 (base measure): Cyclomatic complexity (CC)
Measure 2 (base measure): Number of Statement (NOS)
Measure n
Measure 2 (base measure): Number of total bugs
Measure 1 (base measure): Number of bugs resolved
Measurement Function E.g., Mathemetical Eq.
Measurement method: Manually
Measurement method: Automatically: Instrument (code anayzer tool)
Normalize measure E.g., TOPSİS (step 1 formulation)
Aggregated measure E.g., Arithmetic mean
Measure 3 (derived measure): Measure 3=Measure1/Measure3
Normalize measure
Aggregated measure
Measure A
Measure B
Measure C
Measure D
Measure E
Measure F
E. aggregation method: E.g., TOPSİS
Evaluation Result
Code-based aspect
Community-based aspect
Both aspect

Appendix 3 Matching concepts of OSS-QMM and existing OSS quality models (OSMM, OpenBRR, and SQO-OSS)

OSS-QMM concepts	Quality models terms				
Quality model	**OSMM**	**OpenBRR**		**SQO-OSS**	
Viewpoint	Developer	Developer		Developer	
OSS aspect	Community-based	Code-based	Community-based	Code-based	Community-based
Information need	Calculation of developer size to evaluate maintainability	Calculation of fault proneness 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	Maintainability
Sub-characteristic	Acceptance	Product quality	Product quality	Analyzability	Analyzability
Entity	Developer	Source code	Contributor	Source code	Contributor
Quality requirement	The large size of developer is desirable for maintainability	The low error proneness of the source code is desirable for maintainability	The productive developers are desirable for maintainability	The high comment frequency is desirable for maintainability	The large number of documents is desirable for maintainability
Impact	Positive	Negative	Positive	Positive	Positive
Measurable concepts	The size of developer	The fault proneness of source code	Productivity of contributors	Complexity of source code	Completeness of documentation
Measure	Number of developers (Base measure)	Defect density (Derived measure)	Number of releases (Base measure)	Weighted method per class (WMC) (Base measure)	Number of documents (Base measure)
Unit	Developer	Defects, lines	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 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	Automatically (e.g., Understand scitool, CKJM, Intellij IDEA, etc.)	Manually	Automatically (e.g., Understand scitool, CKJM, Intellij IDEA, etc.)	Manually

OSS-QMM concepts	Quality models terms				
Measurement function	There is no measurement function because it is a base measure	Number of defects/LOC	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

Acknowledgements This study was carried out as part of a PhD study pursued by the first author at the Graduate School of Science and Engineering of Hacettepe University.

Data Availability The data that support the findings of this study are openly available in Zenodo at the following URL: definition of the terminologies in the SQMM, Zenodo, https://doi.org/10.5281/zenodo.6367596.

Declarations

Conflict of interest The authors declare no competing interests.

References

Adewumi, A., Misra, S., & Omoregbe, N. (2013). A review of models for evaluating quality in open source software. *IERI Procedia, 4*, 88–92.

Adewumi, A., Misra S., Omoragbe, N., Crawford, B., & Soto, R. (2016). A systematic literature review of open source software quality assessment models. SpringerPlus, *5.1*, 1936.

Adewumi, A., Misra S., & Omoragbe, N. (2019). FOSSES: framework for open-source software evaluation and selection. *Software: Practice and Experience, 49.5*, 780–812.

Ahmad, N., & Laplante, P. A. (2013). A systematic approach to evaluating open source software. Howard C, ed. Strategic Adoption of Technological Innovations. Hershey, PA: IGI Global: pp 50–69.

Al-Badareen, A. B., Selamat, M. H., Jabar, M. A., Din, J., & Turaev, S. (2011). Software quality models: a comparative study. *International Conference on Software Engineering and Computer Systems,* (pp. 46–55). Springer, Berlin, Heidelberg.

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

Alfonzo, O., Domínguez, K., Rivas, L., Perez, M., Mendoza, L., & Ortega, M. (2008). Quality measurement model for analysis and design tools based on FLOSS. *19th Australian conference on software engineering*. Perth, (pp 26–28) Australia.

Alvaro, A., Almeida, E. S., & Meira, S. R. L. (2010). A software component quality framework. *ACM SIGSOFT Software Engineering Notes, 35*(1), 1–4.

Aversano, L., & Tortorella, M. (2013). Quality evaluation of floss projects: application to ERP systems. *Information and Software Technology*. *55.7*, 1260–1276.

Barcellos, M.P., de Almeida Falbo, R., & Dal Moro, R. (2010). A well-founded software measurement ontology. *FOIS,* (pp. 213–226).

Barcellos, M. P., & de Almeida Falbo, R. (2013). A software measurement task ontology. *Proceedings of the 28th Annual ACM Symposium on Applied Computing,* (pp. 311–318).

Bertoa, M., & Vallecillo, A. (2002). Quality attributes for COTS components*", I+D Computación*, Vol 1. *Nro, 2*, 128–144.

Bertoa, M. F., Vallecillo, A., & García, F. (2006). An ontology for software measurement. *Ontologies for Software Engineering and Software Technology,* (pp. 175–196). Springer, Berlin, Heidelberg.

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

Boehm, B. W., Brown, H., & Lipow, M. (1978). Quantitative evaluation of software quality. *Proceedings of the 2nd International Conference on Software Engineering,* (pp. 592–605).

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

Ciolkowski, M., & Soto, M. (2008). Towards a comprehensive approach for assessing open source projects. *Software Process and Product Measurement,* (pp. 316–330). Springer, Berlin, Heidelberg.

Chirinos, L., Losavio, F., & Bøegh, J. (2005). Characterizing a data model for software measurement. *Journal of Systems and Software, 74*(2), 207–226.

Czarnacka, C. B. (2009). The ISO/IEC Standards for the Software Processes and Products Measurement. (pp. 187–200).

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

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

Duijnhouwer, F. W., & Widdows, C. (2003). Capgemini Expert Letter Open Source Maturity Model, Capgemini, tinyurl.com/yxdbvjk6.

Eghan, E. E., Alqahtani, S.S., Forbes, C., & Rilling, J. (2019). API trustworthiness: an ontological approach for software library adoption. *Software Quality, Journal, 27.3*, 969–1014.

Garcia, F., Bertoa, M. F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., & Genero, M. (2006). Towards a consistent terminology for software measurement. *Information and Software Technology, 48*(8), 631–644.

Garcia, F., Ruiz, F., Calero, C., Bertoa, M. F., Vallecillo, A., Mora, B., & Piattini, M. (2009). Effective use of ontologies in software measurement. *Knowledge Eng. Review, 24*(1), 23–40.

Garcia, F., Serrano, M., Cruz-Lemus, J., Ruiz, F., Piattini, M., & ALARCOS Research Group. (2007). Managing software process measurement: a metamodel-based approach. *Information Sciences*, *177*(12), pp. 2570–2586.

Georgiadoui, E. (2003). GEQUAMO–a generic, multilayered, customizable software quality model. *Software Quality Journal, 11*(4), 313–323. https://doi.org/10.1023/A:1025817312035

Grady, R. B. (1992). *Practical software metrics for project management and process improvement*. Prentice Hall.

Haaland, K., Groven, A.K., Regnesentral, N., Glott, R., Tannenberg, A., & FreeCode, A.S. (2010). Free/libre open source quality models–a comparison between two approaches. *4th FLOS International Workshop on Free/Libre/Open Source Software,* (pp. 1–17).

Hauge, O., Osterlie, T., & Sorensen, C. F. (2009). An empirical study on selection of open source software–preliminary results. *ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*. IEEE.

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.

Henderson-Sellers, B., & Bulthuis, A. (1996). COMMA: Sample metamodels. *JOOP, 9*(7), 44–48.

Hwang, C. L., & Yoon, K. (1981). *Multiple attribute decision making: Methods and applications*. Springer-Verlag.

Işıklar, G., & Büyüközkan, G. (2007). Using a multi-criteria decision making approach to evaluate mobile phone alternatives. *Computer Standards & Interfaces, 29*(2), 265–274.

IEEE. (1998). Standard for a Software Quality Metrics Methodology. *IEEE Standards,* (pp. 1061–1998).

IEEE 610.12. (1990). IEEE Standard Glossary of Software Engineering Terminology.

ISO/IEC 9126–1. (2001). Software Engineering - Product Quality - Part 1: Quality Model, International Organization for Standardization, Geneva, Switzerland.

ISO/IEC 15939. (2007). Software engineering – software measurement process, second edition.

ISO/IEC 25010. (2008). Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) Quality Model and Guide. *International Organization for Standardization*. Geneva, Switzerland.

ISO/IEC 14598. (1999). Information Technology, Software Product Evaluation: Process for Developers. *Software Engineering*.

ISO/IEC 15504–1. (2004). Information technology – Process assessment – Concepts and vocabulary.

ISO/IEC 14143–6. (2012). Information technology, Software measurement, Functional size measurement.

ISO/IEC. ISO/IEC 12207. (2008). System and software engineering – Software life-cycle processes, second edition.

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

ISO/IEC 25020. (2019). Systems and software engineering, Systems and software Quality Requirements and Evaluation (SQuaRE), Quality measurement framework.

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

Jadhav, A. S., & Sonar, R. M. (2011). Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach. *Journal of Systems and Software, 84*(8), 1394–1407.

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

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

Karagiannis, D., & Kühn, H. (2002). Metamodelling platforms. *EC-Web, 2455,* p. 182.

Khondoker, R., Zaalouk, A., Marx, R., & Bayarou, K. (2014). Feature-based comparison and selection of software defined networking (SDN) controllers. *World Congress on Computer Applications and Information Systems (WCCAIS)*. Hammamet, Tunisia.

Kim, H. M. (1999). Representing and reasoning about quality using enterprise models. PhD thesis, Dept. Mechanical and Industrial Engineering, University of Toronto, Canad.

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

Kläs, M., Lampasona, C., Nunnenmacher, S., Wagner, S., Herrmannsdörfer, M., & Lochmann, K. (2010). How to evaluate meta-models for software quality. *Proceedings of the 20th International Workshop on Software Measurement*.

Kuwata, Y., Takeda, K., & Miura, H. (2014). A study on maturity model of open source software community to estimate the quality of products. *Procedia Computer Science, 35*, 1711–171.

Lenarduzzi, V., Taibi, D., Tosi, D., Lavazza, L., & Morasca, S. (2020). Open source software evaluation, selection, and adoption: a systematic literature review. *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE.

Maki-Asiala, P., & Matinlassi, M. (2006). Quality assurance of open source components: integrator point of view. *30th Annual International Computer Software and Applications Conference (COMPSAC'06),* (Vol 2). IEEE.

Mc Call, J. A., Richards, P. K., & Walters, G. F. (1977). Factors in Software Quality, Volumes I, II, and III. US Rome Air Development Center Reports, US Department of Commerce, USA.

Mcgarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., & Hall, F. (2002). Practical software measurement: objective information for decision makers. Addison Wesley.

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

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

Mohagheghi, P., & Dehlen, V. (2008). A metamodel for specifying quality models in model-driven engineering. *Proceedings of the Nordic Workshop on Model Driven Engineering*.

Nistala, P., Nori, K. V., & Reddy, R. (2019). Software quality models: a systematic mapping study. *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP),* (pp. 125–134). IEEE.

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

Othman, S. H., & Beydoun, G. (2010). Metamodelling approach to support disaster management knowledge sharing. *21st Australasian Conference on Information Systems.*

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

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

Özcan, E. C., Ünlüsoy, S., & Eren, T. (2017). A combined goal programming–AHP approach supported with TOPSIS for maintenance strategy selection in hydroelectric power plants. *Renewable and Sustainable Energy Reviews, 78*, 1410–1423.

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

Raffoul, E., Domínguez, K., Perez, M., Mendoza, L. E., & Griman, A. C. (2008). Quality model for the selection of FLOSS-based issue tracking system. *Proceedings of the IASTED international conference on software engineering,* Innsbruck, Austria.

Ramamoorthy, C. V., Prakash, A., Tsai, W. T., & Usuda, Y. (1984). Software engineering: Problems and perspectives. *Computer, 17*(10), 191–209.

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

Raza, A., Capretz, L. F., & Ahmed, F. (2012). An open source usability maturity model (OS-UMM). *Computers in Human Behavior, 28.4*, 1109–1121.

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

Rout, T. P. (1999). Consistency and conflict in terminology in software engineering standards. *Proceedings 4th IEEE International Software Engineering Standards Symposium and Forum (ISESS'99).*

Ruiz, F., Genero, M., García, F., Piattini, M., & Calero, C. (2003). A proposal of a software measurement ontology. *Conference on Computer Science and Operational Research, Buenos Aires, Argentina*.

Saaty, T. L. (1980). *The analytic hierarchy process: Planning, priority setting and resource allocation*. McGraw-Hill.

Saaty, T. L. (2008). Decision making with the analytic hierarchy process. *International Journal of Services Sciences, 1*(1), 83–98.

Saaty, T. L., & Sagir, M. (2015). Ranking countries more reliably in the Summer Olympics. *International Journal of the Analytic Hierarchy Process, 7*(3), 589–610.

Sadeghzadeh, H. M., & Rashidi, H. (2017). Software quality models: A comprehensive review and analysis. *Journal of Electrical and Computer Engineering Innovations (JECEI), 6*(1), 59–76.

Samarthyam, G., Suryanarayana, G., Sharma, T., & Gupta, S. (2013). MIDAS: a design quality assessment method for industrial software. *Software Engineering in Practice*, (pp 911–920). San Francisco, CA, USA.

Samoladas, I., Goussios G., & Spinellis, D. (2008). The SQO-OSS quality model: measurement based open source software evaluation. *IFIP international conference on open source systems*. Springer, Boston, MA.

Sarrab, M., & Rehman, O. M. H. (2014). Empirical study of open source software selection for adoption, based on software quality characteristics. *Advances in Engineering Software, 69*, 1–11.

Siemens company, https://www.siemens.com/global/en.html

Software Engineering Institute. (2010). CMMI for development, version 1.3, Technical Report CMU/SEI-2010-TR-033.

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

Soto, M., & Ciolkowski, M. (2009). The QualOSS open source assessment model measuring the performance of open source communities. *Proceedings of the 3rd International Symposium On Empirical Software Engineering and Measurement*.

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

Suman, M. W., & Rohtak, M. D. U. (2014). A comparative study of software quality models. *International Journal of Computer Science and Information Technologies, 5*(4), 5634–5638.

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

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

Tanrıöver, Ö. Ö., & Bilgen, S. (2011). A framework for reviewing domain specific conceptual models. *Computer Standards & Interfaces, 33*(5), 448–464.

Tassone, J., Xu, S., Wang, C., Chen, J., & Du, W. (2018). Quality assessment of open source software: a review. *IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)* (pp. 411–416). IEEE.

Thapar, S. S., Singh, P., & Rani, S. (2012). Challenges to development of standard software quality model. *International Journal of Computer Applications, 49*(10).

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

Van Solingen, R., Basili, V., Caldiera, G., & Rombach, H. D. (2002). Goal question metric (GQM) approach. *Encyclopedia of Software Engineering*.

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

Wagner, S., Lochmann, K., Heinemann, L., Kläs, M., Trendowicz, A., Plösch, R., & Streit, J. (2012). The Quamoco product quality modelling and assessment approach. *2012 34th International Conference on Software Engineering (ICSE),* (pp. 1133–1142). IEEE.

Wagner, S. (2008). Cost-optimisation of analytical software quality assurance: models, data, case studies. VDM Verlag.

Wang, X. F., Wang, J. Q., & Deng, S. Y. (2013). A method to dynamic stochastic multi-criteria decision making with log-normally distributed random variables. *The Scientific World Journal*.

Wang, Y., Xu, L., & Solangi, Y. A. (2020). Strategic renewable energy resources selection for Pakistan: Based on SWOT-Fuzzy AHP approach. *Sustainable Cities and Society, 52*, 101861.

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

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

Yılmaz, N., & Tarhan, A.K. (2022). Quality evaluation models or frameworks for open source software: a systematic literature review. *Journal of Software: Evolution and Process, 34*(6):e2458. https://doi.org/10.1002/smr.2458.

Yılmaz, N., & Tarhan, A. K. (2020). Meta-models for software quality and its evaluation: a systematic literature review. *International Workshop on Software Measurement and the 15th International Conference on Software Process and Product Measurement, Mexico.*

Yilmaz, N., & Kolukısa Tarhan, A. (2022). Definition of the terminologies in the SQMM. Zenodo. https://doi.org/10.5281/zenodo.6367596

Zahoor, A., Mehboob, K., & Natha, S. (2017). Comparison of open source maturity models. *Procedia Computer Science, 111*, 348–354.

Nebi Yılmaz is currently working as a Research Assistant at Computer Engineering Department of Hacettepe University. He works as a researcher in the area of software engineering for seven years. His major research interests are internal and external software quality, software evaluation and cost of quality. He received M. Sc. Degree in Computer Engineering from Hacettepe University, Ankara, Turkey in 2017. He is currently pursuing his Ph. D. degree from Ankara, Hacettepe University. You may contact his by yilmaz@cs.hacettepe.edu.tr

Ayça Kolukısa Tarhan is an Assoc. Prof. in Software Engineering and working as a researcher and practitioner in this area for twenty years. She has led or been involved in projects originating from industry-academia collaborations on software quality evaluation, software measurement, business process modeling, and system/software requirements elicitation. She had PhD in Information Systems from Informatics Institute of Middle East Technical University. She was an Adjunct Faculty for Software Management Program of the same institute between 2002-2006, and was a visiting researcher between 2013-2015 in Eindhoven University of Technology. Her research interests include internal and external software quality, software development methodologies, software measurement, process maturity, and process mining. She is a faculty with Computer Engineering Department of Hacettepe University in Ankara. You may contact her by atarhan@cs.hacettepe.edu.tr

Authors and Affiliations

Nebi Yılmaz[1,2] · **Ayça Kolukısa Tarhan**[1,3]

Ayça Kolukısa Tarhan
atarhan@cs.hacettepe.edu.tr

1 Software Engineering Research Group (HUSE), Hacettepe University, Ankara, Turkey

2 Hacettepe University Graduate School of Science and Engineering, Ankara, Turkey

3 Computer Engineering Department, Hacettepe University, Ankara, Turkey