applied system innovation

MDPI

Article

On the Use of Quality Models to Address Distinct Quality Views

Tamas Galli [1,*], **Francisco Chiclana** [1,2] and **Francois Siewe** [3]

1 Institute of Artificial Intelligence (IAI), Faculty of Computing, Engineering and Media, De Montfort University, Leicester LE1 9BH, UK; chiclana@dmu.ac.uk
2 Andalusian Research Institute on Data Science and Computational Intelligence (DaSCI), University of Granada, 18071 Granada, Spain
3 Software Technology Research Laboratory (STRL), Faculty of Computing, Engineering and Media, De Montfort University, Leicester LE1 9BH, UK; FSiewe@dmu.ac.uk
* Correspondence: tamas.galli@my365.dmu.ac.uk or tamas.galli@bcs.org

Abstract: Different software product quality models interpret different amounts of information, i.e., they can capture and address different manifestations of software quality. This characteristic can cause misleading statements and misunderstandings while explaining or comparing the results of software product quality assessments. A total of 23 previously identified distinct software product quality models are analysed on how they handle the abstract notion of quality, and a taxonomy on the quality manifestations that the individual software product quality models are able to capture is established. Quality models that are able to solely describe the quality manifestation of the source code are attractive due to their full automation potential through static code analysers, but their assessment results ignore a huge part of software product quality, which is the one that most impresses the end user. The manifestations of software product quality that address the behaviour of the software while it operates, or the perception of the end user with regard to the software in use, require human involvement in the quality assessment. The taxonomy contributes to interpreting the quality assessment results of different quality models by showing the possible quality manifestations that can be captured by the identified models; moreover, the taxonomy also provides assistance while selecting a quality model for a given project. The quality manifestations used for the quality measurement always need to be presented, otherwise the quality assessment results cannot be interpreted in an appropriate manner.

Keywords: software engineering; software product quality model; quality assessment; quality view; SQALE; ISO25010

check for updates

Citation: Galli, T.; Chiclana, F.; Siewe, F. On the Use of Quality Models to Address Distinct Quality Views. *Appl. Syst. Innov.* **2021**, *4*, 41. https://doi.org/10.3390/asi4030041

Academic Editor: Christos Douligeris

Received: 5 March 2021
Accepted: 28 June 2021
Published: 2 July 2021

1. Introduction

Software product quality models have undergone continuous development for more than 40 years, starting with their conception in the 1970s [1–4]. Assessing software products involves quality models of a wide range from simple hierarchic decomposition techniques to complex meta-models to cope with the abstract notion of software quality [3–5]. Nevertheless, the quality models applied and their concrete implementations differ significantly in terms of the ability to capture the manifestations of software quality. The differences in this ability of the identified software product quality models are introduced in the present paper to assist with the decision of whether a particular quality model is suitable for a specific task or whether the statement based on a specific software product quality model holds for software quality in general. Following the terminology of the ISO/IEC 9126 standard and its successor ISO/IEC 25010 [6,7], the quality manifestations the software product quality models are able to capture are called quality views in this paper, by which a unified terminology is also laid down. Creating a unified terminology is necessary, as some software product quality models, including ADEQUATE [8,9], FURPS+ [10–12], GEQUAMO [13], reflect the opinions of different stakeholders, which are directly or indirectly referred to as quality views in the model definition documents.

The ISO/IEC 9126 and ISO/IEC 25010 standards [6,7] distinguish three distinct views: (1) internal quality, (2) external quality, and (3) quality-in-use view [6,7,14]:

Internal quality view: The set of quality properties of the source code and documentation that are deemed to influence the behaviour and use of the software.

External quality view: The set of quality properties that determine how the software product behaves while it operates. These properties are usually measured when the software product assessed is operational and is being tested.

Quality-in-use view: The set of quality properties that determine how the user perceives software product quality, i.e., to what extent the objectives that the software is used for can be achieved.

The software product quality assessed by the three different views needs to have predictive power towards each other, called predictive validity [6,7]. Thus, the measured internal quality should predict the external quality, and the measured external quality should predict the quality-in-use [6,7]. Nevertheless, this validity does not necessarily hold since quality metrics and measures associated with the quality properties do not always have such a predictive power [14]. Software product quality model families explicitly or implicitly define which manifestations of quality they are able to address.

The term software product quality model family is coined for related quality models with the same concepts but with minor differences. While the terms software product quality model family and software product quality framework are used interchangeably here, a distinction is made between the product quality model and process quality model. The former deals with the quality properties of software products while the latter assesses the processes through which the software products come into existence [6,7]. Furthermore, the term software product quality model or quality model in short refers to any software product quality model or any software quality assessment approach that makes the use of a software product quality model possible.

This research considers quality models that aim to describe each known aspect of software product quality, i.e., models that aim for completeness and, therefore, do not focus solely on a limited scope of quality properties such as maintainability or security. Consequently, the identified software product quality models offer support for the assessment of the software product as a whole and allow to define quality targets that endeavour for completeness. To emphasise the aim of these models, the term "complete software product quality models" is used. Thus, partial quality models are not eligible when assessing a software product as a whole. In [5], 23 software product quality model families were evaluated from the point of view of their relevance for the industrial and scientific communities. These identified, complete software product quality models constitute the set of models analysed in the present study.

Complete software product quality models usually encourage tailoring to specific project's needs, which is illustrated in Table 1 with variations within the same software product quality model families. The process of how abstract software product quality models can serve as a basis for a specific application domain (mobile application projects at Samsung) is demonstrated by Kim and Lee in [15]. In the specific tailoring process, Kim and Lee created a separate quality model starting from ISO/IEC 9126 [6] by removing the hierarchy, selecting the quality properties relevant for the specific context and added automation potential for measuring the internal quality view. The demonstrated process can also be applied to consider the specificities of particular application domains such as web, embedded, and IoT.

1.1. Research Question

Different software product quality models approach the abstraction of quality to capture its distinct manifestations in various ways, herein referred to as quality views following the terminology of the ISO/IEC 9126 [6] and ISO/IEC 25010 [7] standards. When assessing software products, it is of crucial importance to highlight the quality views used because the quality assessment results can only be interpreted with regard to this information in a fair manner to avoid misunderstandings and misinterpretation. For this reason, it is necessary to classify all the software product quality models identified in [5] to discover the quality views the models can address. Consequently, the research question in this study is formulated as follows:

RQ1: Which quality views can the identified 23 software product quality model families address?

1.2. Structure of the Study

Section 2 analyses previous studies in the field, Section 3 presents the research methods, Section 4 introduces the taxonomy of all software product quality model families identified in [5], while Section 5 presents the limitations of the conducted research, and Section 6 closes the paper with the concluding remarks.

2. Related Works

Galli et al. in [5] carried out a comprehensive systematic literature review as per [16], and developed a scoring scheme for measuring the relevance of software product quality model families for the industrial and scientific communities. In addition, an execution tracing quality study of software product quality models was reported in [3]. Other studies on software product quality models have been carried out by Hegeman in [17] (up to 2011) and by Ferenc et al. in [18] (up to 2014), respectively. However, these two studies are not systematic as per [16], do not measure the quality model families' relevance for the industrial and scientific communities, and include partial quality models that deal with a specific part of software product quality.

Software product quality models defined, tailored or referenced since 2000 are elicited in [5], which was complemented with an analysis of the developments and trends that emerged during this period of time. Criteria regarding whether a publication establishes a new software product quality model family or solely adjusts a quality model already in use while keeping the existing quality model's concepts are also introduced in detail in [5]. In addition, the relevance of the quality model families for the industrial and scientific communities was measured (1) on the extracted, structured and synthesised qualitative information from the software product quality model definitions, and (2) on the following quantitative indicators: (i) relevance score; (ii) quality score average; (iii) time range; and (iv) 12-month average Google Relative Search Index. The identified 23 software product quality model families reveal huge differences in terms of application and research.

To make the present study self-contained, a summary of the main contributions of the study [5] are given below:

(1) Search queries were developed and recorded for automatic document search;
(2) Searches were carried out in six computer science digital libraries, including IEEE and ACM;
(3) The automatic document search was complemented with manual and reference searches;
(4) All the documents returned by the searches were pre-analysed, the duplicates were removed, the non-primary sources were removed, the complete software product quality models were included in the investigation, but process quality models and partial quality models were excluded with the reason for exclusion recorded;
(5) Documents were assigned to the software product quality models they define, tailor or reference, which lead to the creation of related document clusters, i.e., to *software product quality model families*. Publications and model definitions stem from the academic and industrial domain, including EMISQ [19], SQUALE [20], FURPS [10,11], SQAE and ISO9126 combination [21], Quality Model of Kim and Lee [15] from companies such as Air France, Siemens, Qualixo, IBM, MITRE, and Samsung;

(6) Each document was analysed in-depth to extract the terminology and concepts of each software product quality model;
(7) Each document was assigned a score value for clarity of the publication and a score value for the actuality of the publication based on defined scoring criteria; the clarity score discriminates whether a quality model was published in a detailed mature state in one publication or in several publications by means of smaller increments;
(8) The quality score of a document is defined as the product of its clarity score and actuality score;
(9) For each cluster of documents, the following indicators are defined:
   (i) The relevance score as the sum of the quality scores of its documents;
   (ii) The quality score average;
   (iii) The publication time range; and
   (iv) The 12-month average of Google Trends called the Google Relative Search Index [22].

The first two indicators are related to research intensity, while the third one is related to the duration of research on a software product quality model family, none of which are normalised. The 12-month average Google Relative Search Index (GRSI) illustrates how far a software product quality model family is spread with everyday use cases. GRSI is computed from the normalised time series data of Google Trends. Cases when the GRSI indicator was not possible to apply as certain software product quality model families manifest homonyms, i.e., different things with the same name, were indicated with an "n.a." value in the ranking Table 1.

Table 1. Identified quality model families ranked by relevance scores [5].

Ranking	Model Family	Relevance Score	Quality Score Average	Publication Range from 2000	Google Relative Search Index
1	ISO25010 [7,23–29] [1]	130	16.25	[2011; 2018]	30.02
2	ISO9126 [6,30–37]	120	13.33	[2000; 2017]	53.06
3	SQALE [17,38–44]	107	13.38	[2009; 2016]	18.33
4	Quamoco [45–48]	90	22.5	[2012; 2015]	0
5	EMISQ [19,49,50]	38	12.67	[2008; 2011]	0
6	SQUALE [20,51–53]	36	9	[2012; 2015]	n.a.
7	ADEQUATE [8,9]	18	9	[2005; 2009]	n.a.
8	COQUALMO [54,55]	15	7.5	[2008; 2008]	0.21
=9	FURPS [10–12]	10	3.33	[2005; 2005]	20.56
=9	SQAE and ISO9126 combination [21]	10	10	[2004; 2004]	0
=9	Ulan et al. [56]	10	10	[2018; 2018]	n.a.
10	Kim and Lee [15]	9	9	[2009; 2009]	n.a.
11	GEQUAMO [13]	5	5	[2003; 2003]	0
12	McCall et al. [2,57]	1	0.5	[2002; 2002]	n.a.
=13	2D Model [58]	0	0	n.a.	n.a.
=13	Boehm et al. [1]	0	0	n.a.	n.a.
=13	Dromey [59]	0	0	n.a.	n.a.
=13	GQM [60]	0	0	n.a.	40.73
=13	IEEE Metrics Framework Reaffirmed in 2009 [61]	0	0	n.a.	0
=13	Metrics Framework for Mobile Apps [62]	0	0	n.a.	0
=13	SATC [63]	0	0	n.a.	n.a.
=13	SQAE [64]	0	0	n.a.	n.a.
=13	SQUID [14]	0	0	n.a.	n.a.

[1] The original table in [5] contains an incorrect start year of ISO25010, which is corrected here.

Notice that all models except for FURPS [10] and GQM [60] have score values in agreement with the 12-month average GRSI values, which indicates that FURPS and GQM have more widespread application than shown by their relevance score. The ISO/IEC 9126 and ISO/IEC 25010 standards have very similar software product quality model concepts [6,7]. The 12-month average GRSI indicator shows more activity for the ISO/IEC 9126 standard [6] than for its successor ISO/IEC 25010 [7]; moreover, the last identified publication on the ISO/IEC 9126 standard appeared in 2017, years after the publication of the successor standard ISO/IEC 25010 [7]. The SQALE model [44] was ranked as third according to the relevance score, which mirrors the 12-month average GRSI if the two outliers, FURPS [10] and GQM [60], were excluded. In addition, SonarQube [65], a popular tool implementation of the abstract SQALE model [44], would suppress the 12-month average GRSI indicator for all listed quality model families to nearly zero [5,22], which means that the majority of the activities in the domain seem to be associated with such widespread implementation of the SQALE model [44]. Table 1 results are taken into account while filling in the columns "Research Interest" and "Widespread Use Cases" in the present study taxonomy in Table 2.

Table 2. Taxonomy: software product quality model families and their quality views.

ID	Relevance Rank	Name	Quality Views Considered	Predefined Quality Properties or Metrics Available	Research Interest	Widespread Use Cases	Also Process Related Properties
1	1	ISO25010 [7,23–29]	I, E, U	Yes	Yes	Yes	No
2	2	ISO9126 [6,30–37]	I, E, U	Yes	Yes	Yes	No
3	3	SQALE [17,38–44]	I	Yes	Yes	Yes	No
4	4	Quamoco [45–48]	I, E, U	Yes	No	No	No
5	5	EMISQ [19,49,50]	I	Yes	No	No	No
6	6	SQUALE [20,51–53]	I, E	Yes	No	No	Yes
7	7	ADEQUATE [8,9]	I, E, U	Yes	No	No	Yes
8	8	COQUALMO [54,55]	I, E	Yes	No	No	Yes
9	9	FURPS [10–12]	I, E, (U)	Yes	No	Yes	Yes
10	9	SQAE and ISO9126 combination [21]	I, E	Yes	No	No	No
11	9	Ulan et al. [56]	I	Yes	No	No	No
12	10	Kim and Lee [15]	I	Yes	No	No	No
13	11	GEQUAMO [13]	I, E, U	Yes	No	No	Yes
14	12	McCall et al. [2,57]	I, E, (U)	Yes	No	No	Yes
15	13	2D Model [58]	Undefined	No	No	No	Undefined
16	13	Boehm et al. [1]	I, E, (U)	Yes	No	No	No
17	13	Dromey [59]	I	Yes	No	No	No
18	13	GQM [60]	D	No	No	Yes	D
19	13	IEEE Metrics Framework Reaffirmed in 2009 [61]	Undefined	No	No	No	Undefined
20	13	Metrics Framework for Mobile Apps [62]	Undefined	Yes	No	No	Undefined
21	13	SATC [63]	I, E	Yes	No	No	Yes
22	13	SQAE [64]	I	Yes	No	No	No
23	13	SQUID [14]	D	D	No	No	D

3. Methods

The study applies qualitative research comprising the analysis of documents and tailored quality models to specific project needs. The documents forming the input of the present research were identified by a systematic literature review [16] in the scope of [5]. As the internal/external quality views and quality-in-use view are not explicitly defined in all the model descriptions, the quality property specifications and model concepts were investigated to obtain an insight as to whether a given quality model is able to address external or quality-in-use manifestations.

Quality models able to provide guidelines to the investigation of the source code or documentation, without exhibiting any quality property assessing the software's runtime behaviour or the perception of the end user, are designated as quality models with internal quality view only (denoted "I" in Table 2). Quality models that contain quality properties to assess the software's runtime behaviour are designated with the external quality view (denoted "E" in Table 2). In addition, if a quality model also shows up quality properties to reflect the end user's assessments in any form, then the quality-in-use view is also assigned (denoted "U" in Table 2). If a given quality model does not provide quality property specifications, but it contains concepts that allow to define or deal with existing quality property specifications and how to extend the model, then these concepts were examined to decide which quality views the quality model is able handle. Thus, quality models lacking explicit information about the quality views were classified based on the authors' interpretation of the published model definitions with regard to concepts, and quality properties. Cases where alternative interpretations to the authors' one may exist are indicated with brackets in Table 2 and with additional explanations in the next section. Quality models not providing enough information to perform a reasonable classification are classed "Undefined" in Table 2. There are also quality assessment approaches that allow the definition of a new quality model or the use of a different quality model from the one predefined. The quality views applied in such cases depend on the quality models implemented (denoted "D" in Table 2). The flow chart of the taxonomy process is depicted in Figure 1.

Section 4 lays out the classification of the identified software product quality models and provides reasoning in each case where the model description does not explicitly define the use of the internal, external and quality-in-use views.

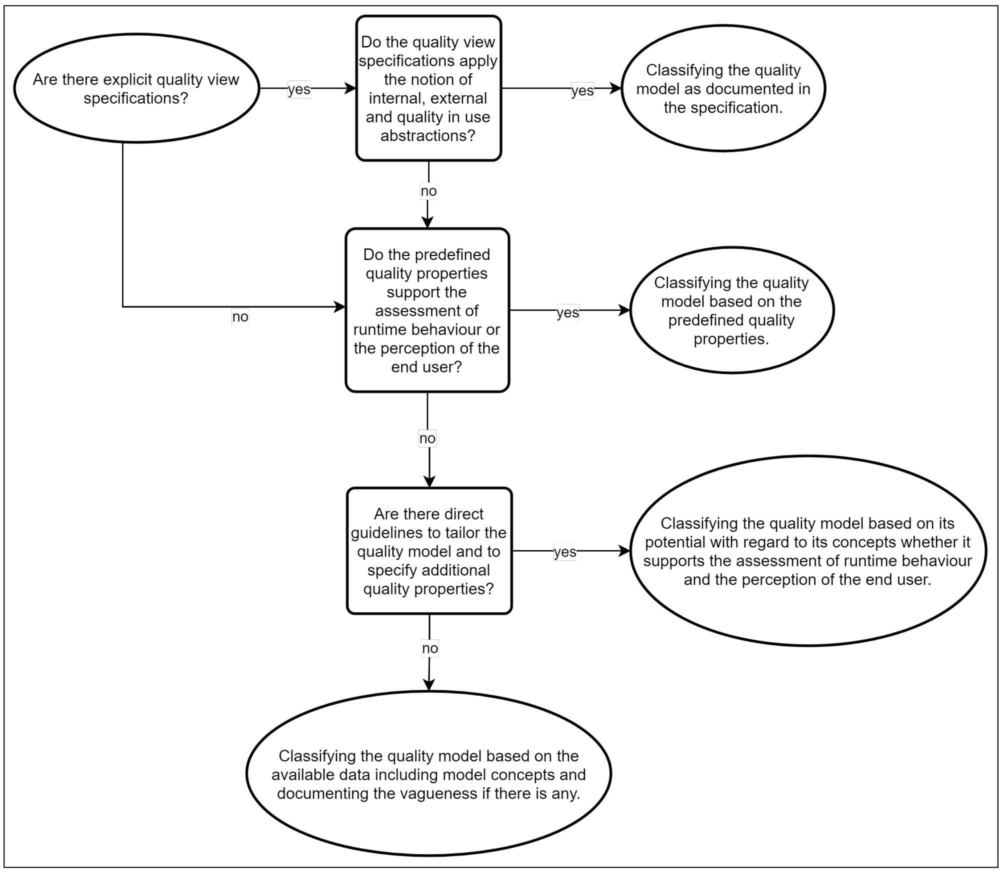
Are there explicit quality view specifications?
yes
Do the quality view specifications apply the notion of internal, external and quality in use abstractions?
yes
Classifying the quality model as documented in the specification.
no
Do the predefined quality properties support the assessment of runtime behaviour or the perception of the end user?
no
yes
Classifying the quality model based on the predefined quality properties.
no
Are there direct guidelines to tailor the quality model and to specify additional quality properties?
yes
Classifying the quality model based on its potential with regard to its concepts whether it supports the assessment of runtime behaviour and the perception of the end user.
no
Classifying the quality model based on the available data including model concepts and documenting the vagueness if there is any.

Figure 1. The flow chart of the quality model classification.

4. Results

This section presents the taxonomy of the software product quality model families from the point of view of quality manifestations they can capture. An important aspect of the quality models is their ability to measure the properties of software products; therefore, software product quality models are involved in the study; however, some quality models also exhibit properties to assess the process by which the software products come into existence. Such models are also marked in Table 2 to provide support for the selection of a potential quality model.

Quamoco, ADEQUATE, FURPS+, GEQUAMO, McCall et al.'s model, and Boehm et al.'s model are classified as quality models with internal, external and quality-in-use views in Table 2: the Quamoco quality model [46] encompasses an entity "utility" that describes the relative satisfaction of stakeholders with regard to a specific "factor" in the model's terminology; ADEQUATE [8,9] determines the key personnel and their views. FURPS+ [10–12] includes systematic approaches to collect architectural requirements involving the stakeholders to consider their opinions; GEQUAMO [13] encompasses quality views for users, sponsors and developers; McCall et al.'s model [2] contains usability and testability factors; Boehm et al.'s model [1] reflects quality characteristics for usability and human engineering, which could make it possible to consider the end user's view of quality.

SQUALE, COQUALMO and STAC models are classified as quality models with internal and external quality views in Table 2: SQUALE [20] possesses different metrics and practices related to distinct test levels; COQUALMO [54,55] comprises a defect introduction and removal sub-model, which assumes the software to be operational; the STAC model [63] defines metrics related to testing.

Ulan et al.'s model [56], SQALE [17,38–44], EMISQ [19,49,50], Kim and Lee's model [15], Dromey's model [59], SQAE [64] are classified in Table 2 as quality models with internal quality view only: Ulan et al.'s model permits the use of any goal-question-metric approach;

its novelty lies in the complex mathematical computations used in a clustering approach, although the model published demonstrated internal quality properties only without direct guidelines for extension; SQALE also contains the testability quality property that is divided into two further quality properties, Unit Testing Testability and Integration-level Testability, that are based on the results of code analysers.

GQM [60] and SQUID [14] are rather general quality assessment approaches with no predefined quality models. These approaches assume the creation of a software product quality model or the use of an existing one, named content model in SQUID terminology, which is suitable to achieve the goals set in the given context. Consequently, the quality views they consider depend on the defined metrics and goals. In the published case study with the model definition [14], SQUID also encompasses metrics that can only be collected when the software is operational and metrics on usability that allow the expression of end users' quality perception. For these reasons, they are classified in Table 2 as quality assessment approaches with quality views that depend on the quality model used.

In addition to the quality views applied, further information is presented in the columns: "Predefined Quality Properties or Metrics Available" to describe whether the specified model defines quality properties with associated computation methods; "Research Interest" to indicate whether there are more publications related to the quality model and one in the last five years; "Widespread Use Cases" to provide the judgement based on the specified model's 12-month average GRSI indicator from Table 1; "Also Process Related Properties" to show whether the specified model is a hybrid model with quality properties that describe process quality.

From Table 2, it is clear that the identified software product quality model families apply different quality views to capture the abstract notion of quality. The capabilities of the listed software product quality models deviate significantly; many of them are unable to assess the external quality view and the quality-in-use view. The provided taxonomy makes it possible to interpret the quality assessment results. Based on quality assessments with internal quality view only, it is not possible to make valid statements regarding software product quality as a whole. This is because such models lack the capability to capture how the software behaves while it is used and how the end user perceives its quality. Quality assessment results always need to be interpreted with the quality model's capabilities, including quality views, which is especially important for project negotiations. In addition, Table 2 also contains relevant information to support the selection of a quality model for a specific project. In such cases, not only the model's quality views play an important role but the following criteria can also influence the decision: (1) the amount of associated research studies on the model, i.e., the research interest, (2) the model's public spread, and (3) whether the model has process related quality properties.

5. Limitations

The research conducted is qualitative in nature and rests on data extracted from the definition documents of the different quality models and from their variations, which depict how to tailor the identified models to specific needs. The analysis of the listed software product quality model families was done carefully, although the high amount of information implicates a risk of ignoring minor details. On the other hand, some quality models investigated did not provide enough information to allow their classification with regard to Table 2 taxonomy. These models were correspondingly marked in Table 2 and explained in Section 4.

Table 2 taxonomy lists the software product quality model families according to their relevance for the industrial and scientific communities, as reported in 2020 in [5]. This means that publications appeared afterwards and new applications of the listed quality models might have an influence on this order but not on the provided taxonomy of quality models with internal, external and quality-in-use manifestations.

6. Conclusions

Classification of software product quality models based on the amount of information they can interpret, i.e., the quality manifestations they can address, is an important step towards preventing misunderstandings and misleading statements regarding software product quality assessments. At present, the most comprehensive and easy-to-understand quality interpretations are defined by the ISO/IEC 9126 and ISO/IEC 25010 [6,7] standards. Both standards classify quality manifestations with three distinct views: internal, external and quality-in-use view. This was extended to all 23 software product quality models identified in 2020 in [5] and listed in Table 2 even if the definition documents use different terminologies or do not mention such views explicitly. Consequently, the current study places the existing publications related to software product quality modelling into a new context. Moreover, the research conducted stresses the requirement to present not only the quality assessment results but also the quality manifestations, i.e., the quality views, measured. The latter is a must for fair quality measurements and assessments.

The quality assessment results cannot be interpreted on their own without the knowledge of whether internal, external or quality-in-use quality manifestations were measured. Indicating and emphasising this characteristic of the software product quality assessment is a major contribution of the paper. The scope of software product quality that the individual quality models are able to assess has a direct impact on software development and maintenance costs. In conclusion, as explained in Section 4, a "good quality" statement made on the basis of a software product quality model that captures the above three views has a very different meaning to a "good quality" statement made on the basis of another software product quality model, which can only capture one individual view of the three defined ones. Consequently, it is never fair to make a statement with regard to the whole software product quality when only the internal quality view is assessed.

The software product quality models with full-automation potential can solely assess the internal view of quality, which can be captured by static code analysers. The software product quality models capable of handling all internal, external and quality-in-use views offer partial automation features so that the results of the measurement of the external and quality-in-use views could be considered. This is not an impediment but a broader range of quality manifestations to consider for approximating the reality better while assessing quality. Nevertheless, if such models are integrated into the development pipeline with the expectation of full automation, then their capabilities are restricted to the internal quality view only.

Author Contributions: Conceptualization, T.G.; methodology, T.G.; validation, F.C. and F.S.; formal analysis, T.G.; investigation, T.G.; resources, T.G.; data curation, T.G.; writing—original draft preparation, T.G.; writing—review and editing, F.C. and F.S.; visualization, T.G.; supervision, F.C. and F.S.; project administration, T.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Helsinki, and approved by the Institutional Review Board of the De Montfort University, ethical approval code: 1718/453, date of approval: 31 May 2018.

Informed Consent Statement: Not applicable.

Data Availability Statement: Software product quality model definition documents and their adaptations were analysed in the scope of the study. The documents examined are dully cited in the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Boehm, B.W.; Brown, J.R.; Lipow, M. Quantitative Evaluation of Software Quality. In Proceedings of the 2nd International Conference on Software Engineering, San Francisco, CA, USA, 13–15 October 1976.
2. McCall, J.A.; Richards, P.K.; Walters, G.F. Factors in Software Quality, Concept and Definitions of Software Quality. 1977. Available online: http://www.dtic.mil/dtic/tr/fulltext/u2/a049014.pdf (accessed on 6 March 2018).
3. Galli, T. Fuzzy Logic Based Software Product Quality Model for Execution Tracing. Master's Thesis, Centre for Computational Intelligence, De Montfort University, Leicester, UK, 2013.
4. Galli, T.; Chiclana, F.; Carter, J.; Janicke, H. Towards Introducing Execution Tracing to Software Product Quality Frameworks. *Acta Polytech. Hung.* **2014**, *11*, 5–24. [CrossRef]
5. Galli, T.; Chiclana, F.; Siewe, F. Software Product Quality Models, Developments, Trends and Evaluation. *SN Comput. Sci.* **2020**. [CrossRef]
6. *ISO/IEC 9126-1:2001 Software Engineering—Product Quality—Part 1: Quality Model*; International Organization for Standardization: Geneva, Switzerland, 2001.
7. *ISO/IEC 25010:2011 Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models*; International Organization for Standardization: Geneva, Switzerland, 2011.
8. Khaddaj, S.; Horgan, G. A Proposed Adaptable Quality Model for Software Quality Assurance. *J. Comput. Sci.* **2005**, *1*, 482–487. [CrossRef]
9. Horgan, G.; Khaddaj, S. Use of an adaptable quality model approach in a production support environment. *J. Syst. Softw.* **2009**, *82*, 730–738. [CrossRef]
10. Grady, R.B.; Caswell, D.L. *Software Metrics: Establishing a Company-Wide Program*; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1987.
11. Grady, R.B. *Practical Software Metrics for Project Management and Process Improvement*; Prentice Hall: Upper Saddle River, NJ, USA, 1992.
12. Eeles, P. Capturing Architectural Requirements. 2005. Available online: https://www.ibm.com/developerworks/rational/library/4706-pdf.pdf (accessed on 19 April 2018).
13. Georgiadou, E. GEQUAMO—A Generic, Multilayered, Customisable, Software Quality Model. *Softw. Qual. J.* **2003**, *11*, 313–323. [CrossRef]
14. Kitchenham, B.; Linkman, S.; Pasquini, A.; Nanni, V. The SQUID approach to defining a quality model. *Softw. Qual. J.* **1997**, *6*, 211–233. [CrossRef]
15. Kim, C.; Lee, K. Software Quality Model for Consumer Electronics Product. In Proceedings of the 9th International Conference on Quality Software, Monte Porzio Catone, Italy, 23–25 June 2008; pp. 390–395.
16. Kitchenham, B.; Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; Technical Report; EBSE-2007-01; Software Engineering Group, School of Computer Science and Mathematics, Keele University: Keele, UK; Department of Computer Science, University of Durham: Durham, UK, 2007. Available online: https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf (accessed on 1 June 2017).
17. Hegeman, J.H. On the Quality of Quality Models. Master's Thesis, University Twente, Enskeard, The Netherlands, 2011.
18. Ferenc, R.; Hegedüs, P.; Gyimóthy, T. Software Product Quality Models, Chapter; In *Evolving Software Systems*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 65–100. [CrossRef]
19. Kothapalli, C.; Ganesh, S.G.; Singh, H.K.; Radhika, D.V.; Rajaram, T.; Ravikanth, K.; Gupta, S.; Rao, K. Continual monitoring of Code Quality. In Proceedings of the 4th India Software Engineering Conference 2011, ISEC'11, Thiruvananthapuram Kerala, India, 24–27 February 2011; pp. 175–184. [CrossRef]
20. Mordal-Manet, K.; Balmas, F.; Denier, S.; Ducasse, S.; Wertz, H.; Laval, J.; Bellingard, F.; Vaillergues, P. The Squale Model—A Practice-based Industrial Quality Model. 2009. Available online: https://hal.inria.fr/inria-00637364 (accessed on 6 March 2018).
21. Côté, M.A.; Suryn, W.; Martin, R.A.; Laporte, C.Y. Evolving a Corporate Software Quality Assessment Exercise: A Migration Path to ISO/IEC 9126. *Softw. Qual. Prof.* **2004**, *6*, 4–17.
22. Google. Google Search Trends for the Past 12 Months, Worldwide. 2020. Available online: https://trends.google.com/trends/explore (accessed on 30 January 2021).
23. Ouhbi, S.; Idri, A.; Fernández-Alemán, J.L.; Toval, A.; Benjelloun, H. Applying ISO/IEC 25010 on mobile personal health records. In Proceedings of the HEALTHINF 2015—8th International Conference on Health Informatics, Part of 8th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC, Lisbon, Portugal, 12–15 January 2015; SciTePress: Setúbal, Portugal, 2015; pp. 405–412, ISBN 978-989-758-068-0
24. Idri, A.; Bachiri, M.; Fernández-Alemán, J.L. A Framework for Evaluating the Software Product Quality of Pregnancy Monitoring Mobile Personal Health Records. *J. Med. Syst.* **2016**, *40*, 50. [CrossRef] [PubMed]
25. Forouzani, S.; Chiam, Y.K.; Forouzani, S. Method for assessing software quality using source code analysis. In *ACM International Conference Proceeding Series*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 166–170. [CrossRef]
26. Domínguez-Mayo, F.J.; Escalona, M.J.; Mejías, M.; Ross, M.; Staples, G. Quality evaluation for Model-Driven Web Engineering methodologies. *Inf. Softw. Technol.* **2012**, *54*, 1265–1282. [CrossRef]

27. Idri, A.; Bachiri, M.; Fernandez-Aleman, J.L.; Toval, A. Experiment design of free pregnancy monitoring mobile personal health records quality evaluation. In Proceedings of the 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), Munich, Germany, 14–16 September 2016; pp. 1–6. [CrossRef]
28. Shen, P.; Ding, X.; Ren, W.; Yang, C. Research on Software Quality Assurance Based on Software Quality Standards and Technology Management. In Proceedings of the 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Busan, Korea, 27–29 June 2018; pp. 385–390. [CrossRef]
29. Liu, X.; Zhang, Y.; Yu, X.; Liu, Z. A Software Quality Quantifying Method Based on Preference and Benchmark Data. In Proceedings of the 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Busan, Korea, 27–29 June 2018; pp. 375–379. [CrossRef]
30. Kanellopoulos, Y.; Tjortjis, C.; Heitlager, I.; Visser, J. Interpretation of source code clusters in terms of the ISO/IEC-9126 maintainability characteristics. In Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, Athens, Greece, 1–4 April 2008; pp. 63–72. [CrossRef]
31. Vetro, A.; Zazworka, N.; Seaman, C.; Shull, F. Using the ISO/IEC 9126 product quality model to classify defects: A controlled experiment. In Proceedings of the 16th International Conference on Evaluation Assessment in Software Engineering (EASE 2012), Ciudad Real, Spain, 14–15 May 2012; pp. 187–196. [CrossRef]
32. Parthasarathy, S.; Sharma, S. Impact of customization over software quality in ERP projects: An empirical study. *Softw. Qual. J.* **2017**, *25*, 581–598. [CrossRef]
33. Li, Y.; Man, Z. A Fuzzy Comprehensive Quality Evaluation for the Digitizing Software of Ethnic Antiquarian Resources. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, 12–14 December 2008; Volume 5, pp. 1271–1274. [CrossRef]
34. Hu, W.; Loeffler, T.; Wegener, J. Quality model based on ISO/IEC 9126 for internal quality of MATLAB/Simulink/Stateflow models. In Proceedings of the 2012 IEEE International Conference on Industrial Technology, Athens, Greece, 19–21 March 2012; pp. 325–330. [CrossRef]
35. Liang, S.K.; Lien, C.T. Selecting the Optimal ERP Software by Combining the ISO 9126 Standard and Fuzzy AHP Approach. *Contemp. Manag. Res.* **2006**, *3*, 23. [CrossRef]
36. Correia, J.; Visser, J. Certification of Technical Quality of Software Products. In Proceedings of the International Workshop on Foundations and Techniques for Open Source Software Certification, 2008; pp. 35–51. Available online: http://wiki.di.uminho.pt/twiki/pub/Personal/Joost/PublicationList/CorreiaVisserOpenCert2008.pdf (accessed on 1 June 2018).
37. Andreou, A.S.; Tziakouris, M. A quality framework for developing and evaluating original software components. *Inf. Softw. Technol.* **2007**, *49*, 122–141. [CrossRef]
38. Letouzey, J.L.; Coq, T. The SQALE Analysis Model: An Analysis Model Compliant with the Representation Condition for Assessing the Quality of Software Source Code. In Proceedings of the 2010 Second International Conference on Advances in System Testing and Validation Lifecycle, Nice, France, 22–27 August 2010; pp. 43–48.
39. Letouzey, J.L. Managing Large Application Portfolio with Technical Debt Related Measures. In Proceedings of the Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), Berlin, Germany, 5–7 October 2016; p. 181. [CrossRef]
40. Letouzey, J.L. The SQALE method for evaluating Technical Debt. In Proceedings of the Third International Workshop on Managing Technical Debt (MTD), Zurich, Switzerland, 5 June 2012; pp. 31–36. [CrossRef]
41. Letouzey, J.; Coq, T. The SQALE Models for Assessing the Quality of Real Time Source Code. 2010. Available online: https://pdfs.semanticscholar.org/4dd3/a72d79eb2f62fe04410106dc9fcc27835ce5.pdf?ga=2.24224186.1861301954.1500303973-1157276278.1497961025 (accessed on 17 July 2017).
42. Letouzey, J.L.; Ilkiewicz, M. Managing Technical Debt with the SQALE Method. *IEEE Softw.* **2012**, *29*, 44–51. [CrossRef]
43. Letouzey, J.L.; Coq, T. The SQALE Quality and Analysis Models for Assessing the Quality of Ada Source Code. 2009. Available online: http://www.adalog.fr/publicat/sqale.pdf (accessed on 17 July 2017).
44. Letouzey, J.L. The SQALE Method for Managing Technical Debt, Definition Document V1.1. 2016. Available online: http://www.sqale.org/wp-content/uploads//08/SQALE-Method-EN-V1-1.pdf (accessed on 2 August 2017).
45. Gleirscher, M.; Golubitskiy, D.; Irlbeck, M.; Wagner, S. Introduction of static quality analysis in small- and medium-sized software enterprises: Experiences from technology transfer. *Softw. Qual. J.* **2014**, *22*, 499–542. [CrossRef]
46. Wagner, S.; Lochmann, K.; Heinemann, L.; As, M.K.; Trendowicz, A.; Plösch, R.; Seidl, A.; Goeb, A.; Streit, J. The Quamoco Product Quality Modelling and Assessment Approach. In Proceedings of the 34th International Conference on Software Engineering, Zurich, Switzerland, 2–9 June 2012; IEEE Press: Piscataway, NJ, USA, 2012; ICSE '12, pp. 1133–1142.
47. Wagner, S.; Lochmann, K.; Winter, S.; Deissenboeck, F.; Juergens, E.; Herrmannsdoerfer, M.; Heinemann, L.; Kläs, M.; Trendowicz, A.; Heidrich, J.; et al. The Quamoco Quality Meta-Model. October 2012. Available online: https://mediatum.ub.tum.de/attfile/1110600/hd2/incoming/2012-Jul/517198.pdf (accessed on 18 November 2017).
48. Wagner, S.; Goeb, A.; Heinemann, L.; Kläs, M.; Lampasona, C.; Lochmann, K.; Mayr, A.; Plösch, R.; Seidl, A.; Streit, J.; et al. Operationalised product quality models and assessment: The Quamoco approach. *Inf. Softw. Technol.* **2015**, *62*, 101–123. [CrossRef]

49. Plösch, R.; Gruber, H.; Hentschel, A.; Körner, C.; Pomberger, G.; Schiffer, S.; Saft, M.; Storck, S. The EMISQ method and its tool support-expert-based evaluation of internal software quality. *Innov. Syst. Softw. Eng.* **2008**, *4*, 3–15. [CrossRef]
50. Plösch, R.; Gruber, H.; Körner, C.; Saft, M. A Method for Continuous Code Quality Management Using Static Analysis. In Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology, Porto, Portugal, 29 September–2 October 2010; pp. 370–375. [CrossRef]
51. Laval, J.; Bergel, A.; Ducasse, S. Assessing the Quality of Your Software with MoQam. 2008. Available online: https://hal.inria.fr/inria-00498482 (accessed on 6 March 2018).
52. Balmas, F.; Bellingard, F.; Denier, S.; Ducasse, S.; Franchet, B.; Laval, J.; Mordal-Manet, K.; Vaillergues, P. Practices in the Squale Quality Model (Squale Deliverable 1.3). October 2010. Available online: http://www.squale.org/quality-models-site/research-deliverables/WP1.3Practices-in-the-Squale-Quality-Modelv2.pdf (accessed on 16 November 2017).
53. INRIA RMoD, Paris 8, Qualixo. Technical Model for Remediation (Workpackage 2.2). 2010. Available online: http://www.squale.org/quality-models-site/research-deliverables/WP2.2Technical-Model-for-Remediationv1.pdf (accessed on 16 November 2017).
54. Boehm, B.; Chulani, S. *Modeling Software Defect Introduction and Removal—COQUALMO (Constructive QUALity Model)*; USC—Center for Software Engineering: Los Angeles, CA, USA, 1999.
55. Madachy, R.; Boehm, B. Assessing Quality Processes with ODC COQUALMO. In *Making Globally Distributed Software Development a Success Story*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5007, pp. 198–209. [CrossRef]
56. Ulan, M.; Hönel, S.; Martins, R.M.; Ericsson, M.; Löwe, W.; Wingkvist, A.; Kerren, A. Quality Models Inside Out: Interactive Visualization of Software Metrics by Means of Joint Probabilities. In Proceedings of the 2018 IEEE Working Conference on Software Visualization (VISSOFT), Madrid, Spain, 18–19 September 2018; pp. 65–75. [CrossRef]
57. Benedicenti, L.; Wang, V.W.; Paranjape, R. A quality assessment model for Java code. In Proceedings of the Canadian Conference on Electrical and Computer Engineering, Winnipeg, MB, Canada, 12–15 May 2002; pp. 687–690. [CrossRef]
58. Zhang, L.; Li, L.; Gao, H. 2-D Software Quality Model and Case Study in Software Flexibility Research. In Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control and Automation, Vienna, Austria, 10–12 December 2008; IEEE Computer Society: Washington, DC, USA, 2008; CIMCA '08, pp. 1147–1152. [CrossRef]
59. Dromey, R. A Model for Software Product Quality. *IEEE Trans. Softw. Eng.* **1995**, *21*, 146–162. [CrossRef]
60. Van Solingen, R.; Berghout, E. *The Goal/Question/Metric Method a Practical Guide for Quality Improvement of Software Development*; McGraw Hill Publishing: London, UK, 1999.
61. IEEE Computer Society. *IEEE Stdandard 1061–1998: IEEE Standard for a Software Quality Metrics Methodology*; IEEE: Piscataway, NJ, USA, 1998.
62. Franke, D.; Weise, C. Providing a software quality framework for testing of mobile applications. In Proceedings of the 4th IEEE International Conference on Software Testing, Verification, and Validation, ICST 2011, Berlin, Germany, 21–25 March 2011; pp. 431–434. [CrossRef]
63. Hyatt, L.E.; Rosenberg, L.H. A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality. In Proceedings of the Product Assurance Symposium and Software Product Assurance Workshop, EAS SP-377, Noordwijk, The Netherlands, 19–21 March 1996; European Space Agency: Paris, France, 1996.
64. Martin, R.A.; Shafer, L.H. Providing a Framework for effective software quality assessment—A first step in automating assessments. In Proceedings of the First Annual Software Engineering and Economics Conference, McLean, VA, USA, 2–3 April 1996.
65. SonarSource. SonarQube. 2017. Available online: https://www.sonarqube.org (accessed on 16 February 2017).