PQEMw, Applying Weighted Sums to Software Quality Measurement

Mariana Falco[1] and Gabriela Robiolo[2]

[1] LIDTUA/CONICET, Facultad de Ingeniería, Universidad Austral
Pilar, Buenos Aires, Argentina
mfalco@austral.edu.ar
[2] LIDTUA, Facultad de Ingeniería, Universidad Austral
Pilar, Buenos Aires, Argentina
grobiolo@austral.edu.ar

Abstract. Product owners and quality leaders need to understand the quality level of a software product through its life cycle in order to achieve appropriate decision making to remain in the same iteration or to continue to the next one. Product Quality Evaluation Method (PQEM) evaluates and monitors the quality of a software product within each iteration. The present article introduces the extension of the method, $PQEM_{wi}$ to include the definition and calculation of weights for each quality characteristic measured. $PQEM_{wi}$ allows the stakeholder to set their point of view and importance of each quality characteristic, and we carried out an illustrative example of two apps, comparing the decision-making of the weights selection and the results of applying $PQEM_{wi}$.

Keywords: Quality measurement · Weighted sums · Software product.

1 Introduction

In Software Engineering, software evolves over time but there must be a balance between new developments and factors that worsen progress [1]. The changing and dynamic environments activated a scheme where the requirements associated with the release of software products are affected by a high demand from the stakeholders, to achieve quality products with short delivery times.

Although around the late seventies, the laws of software evolution were not widely accepted within the field, but they gradually became more important as they were able to provide inputs to understand the software process. Today, these eight laws are supported by the FEAST (Feedback, Evolution And Software Technology) hypothesis [2], which leads to point out the need to understand the bases of these laws to apply these concepts in the current field of software development.

A field in which the measurement plays a fundamental role in the development of efficient and effective software, where the adverse impact of low-level quality is way more significant in a dynamic era [3]. As a part of the software life cycle process, quality plays a vital role and after each iteration, it is necessary to

study the quality level achieved on that iteration before proceeding to the next one [4].

Even though there are some manual and automatic ways of analysing quality like SonarQube[3], there are no fully equipped tools to measure a set of quality characteristics. Based on these challenges, the authors have defined Product Quality Evaluation Method (PQEM) [5–7], which is a five-step method per iteration, whose main goal is to perform a thoughtful quality assessment of the different iterations within a software product, and that produces a single value between 0 and 1 as the final outcome that represents the product quality level, which is called Total Obtained Coverage (TOC_i).

PQEM is structured around the Representational Theory of Measurement [14], the Goal-Question-Metric approach [8], the set of quality characteristics defined by the standard ISO/IEC 25010 [18], the set of measures defined by the standard ISO/IEC 25023 [19], the extension of notion of coverage testing as a percentage scale concept to achieve the measurement, and the definition of acceptance criteria related to the expected quality level per each iteration.

Based on previous applications of PQEM, we discovered that the stakeholders might need to define a weighing of each quality attribute on the software product under analysis, as a way to specify the importance of each attribute in the domain. As such, the goal of the present article is to introduce the extension of PQEM, called $PQEM_w$ with the addition of weighting up each quality attribute trough the selection of a weight that is included as a part of an equation to obtain a quality value called TOC_{wi}. In other words, the addition of weights lets the stakeholder to specify their point of view based on the domain impact of each quality attribute.

This article is structured as follows: in Section 2, the related work will be addressed, while Section 3 will describe and characterize each of the steps within the Product Quality Evaluation Method (PQEM) as well as the extension of PQEM. Section 4 will include the characteristics and results of an illustrative example. Section 5 will describe the discussion, while Section 6 will portray the conclusions as well as the future work.

2 Related work

Weighted sums are used to quantify a number of software attributes, allowing to use a weighted sum to combine several, lower-level measures to build a single, upper-level measure that may quantify different aspects of a given attribute at the same time or even a single aspect of an attribute, based on a number of measures for it [25]. In this line, Hovorushchenko developed a method for evaluating the weights of software quality measures and indicators [10], based on ISO/IEC 25010:2011 set of quality characteristics, and they concluded that there are measures that there is a correlation of sub-characteristics and characteristics with measures.

[3] SonarQube, https://www.sonarqube.org/

The development of software applications involves concepts such as estimated quality level (used when the app is not yet available), planned quality level (obtained by studying the quality of same class software apps available on the market), and actual level of quality (must be set to ensure reaching the app's goals). In this line, the authors in [11] constructed an indicator for each evaluation criteria that chooses quality products, and two aggregated indicators which includes all quality characteristics, one is the aggregation using the sum operator while the other is the aggregation using the product operator. Their main variables are the following: the number of characteristics, the quality level and the weight associated to a characteristic.

Later on, two early models described quality using decomposition approach, one is from McCall [12] and the other from Boehm and others [13]. These models focus on the final product and identify key attributes of quality from user's perspective - called quality factors, which are normally high-level external attributes like reliability, usability, and maintainability; and they assume that the quality factors are still at too high a level to be measurable directly, and so they are further decomposed into quality subfactors [14]. Also, there are examples of fixed quality models.

These authors defined software quality in terms of external software attributes, regarding sub-attributes. Even though the use of weighted sums may not be mandatory in all of these quality models, they have been used to aggregate the measures associated with a sub-attribute to obtain a single value to measure the sub-attribute. Finally, a measure for the overall quality may be obtained as a weighted sum of the measures obtained for the attributes [15].

Likewise, PQEM method includes aggregated sums as a way to measure each quality characteristic which are composed by a subset of Quality Attribute Requirements (QARs), which lead to obtain a single measure per quality characteristic [6, 7]. Also, a measure for the overall quality is obtained as an aggregated sum of the measures obtained per each quality characteristic.

Literature shows approaches that address the measurement of different aspects within the life cycle of a software product. For example, a set of measures were defined to quantify the size of a product at the beginning of development based on the set of functional requirements, and are called Functional Size Measurement (FSM) methods. Function Points Analysis (FPA) is the first proposal for FSM, it provides a measure of the size in function points, and requires a manual analysis of informal documents such as software requirements and sizing of an application before it is developed.

In the process of identifying and weighting Basic Functional Components (BFC), the FP measurement is based on the measure of BFC and the subsequent weighting and aggregation of BFC measures [16]. Some studies have used weighted sums as ways of quantifying quality attributes, and so PQEM includes this idea of weighted sum as a way to combine measures which are part of each quality attributes analyzed on an iteration.

3 PQEM

PQEM [5] is a five-step method per iteration, whose main goal is to analyze, study, measure and assess the quality level of the different iterations within a software product. It produces a single value between 0 and 1 as the final outcome that represents the product quality level, basically the degree to which the software product fulfills its quality attribute requirements [17].

This method is structured around the quality model provided by the standard ISO/IEC 25010:2011 [18], which provides a basis to analyze a software product with respect to the following set of quality characteristics: Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability; and around the standard ISO/IEC 25023:2016 [19] that defines quality measures for quantitatively evaluating system and software product quality in terms of the previous characteristics. The five steps are described below.

- **Step 1**: *Product Setup* - definition of the amount of expected iterations of the software product, and the definition of expected quality level per iteration
- **Step 2**: *Elicitation of Quality Attributes Requirements (QARs)* - composed by the selection of quality characteristics, the specification of Quality Attribute Requirements (QARs), the elicitation of metrics, and the definition of acceptance criteria per QAR.
- **Step 3**: *Measure and Test each Quality Attribute Requirements (QARs)* - that involves measuring each question, running the defined quality measure, and describing whether or not the acceptance criteria were met (1 equals *passed*, 0 equals $failed$).
- **Step 4**: *Collect and Synthesize Results* - includes the implementation of the extension of the testing coverage [20] as a quality coverage
- **Step 5**: *Product Quality Level Assessment* - analyze the obtained quality level, and make a decision to move forward to the next iteration or not

It is worth mentioning that the steps described above are repeated for each iteration within the product life cycle. Also, each step can be done manually, while the measurement itself can be done through the aid of software in order to reach the coverage calculations and the quality level. This characteristic makes the method suitable to be applied in any software development method that defines iterations. When evaluating the progress of a project or a product, it addresses the logic of iterative methods, and when it comes to quality evolution, the product of each iteration is a new usable version of the product.

With the set of defined equations to compute the quality level, it is possible to obtain the quality level of each iteration from a software product. The following subsection describes the incorporation of weights to the PQEM method, to obtain an improved version of the equations, by letting the stakeholder to specify a set of weights to identify the priority of each quality characteristic over the others.

Furthermore, a newer logic was also added to PQEM that will allow the stakeholder to specify whether an attribute within the iteration is mandatory or

not. That is, when you define the list of quality attributes you can specify that a subset of them be mandatory, in other words: those QARs must have reached the expected coverage level for the iteration, if they do not reach it, the iteration cannot advance. So the acceptance criteria to be defined not only includes the acceptance criteria per quality attribute, but also whether it is mandatory or not.

3.1 PQEMw: PQEM with weights

With the previous definition of PQEM, all quality attributes are considered evenly, because the attributes have the same significance for the stakeholder. But the different domains lead to differences in valuation, and that some quality attributes are positioned over others. For example, in the medical field, Interoperability and Security are set to be priority over others, but that does not mean that the others are not important nor necessary, it only establishes an idea of hierarchy in a higher valuation for an attribute with respect to the other [21–24].

Based on the above ideas, it was thought to define a set of weights to include within the PQEM method, to reach those different level of significance, according to the requirements of the domain in which the software product is embedded. The original definition of the quality level (TOC_i) implied the sum of the coverage reached by each quality attribute, where the range of values that it can take is between 0 and 1, so the closer to 1 is the TOC_i level, the better the quality value of that iteration. The acceptance criteria is a number that can take values between 0 and 1, with a tendency to present values greater than 0.5; while they are defined by the stakeholder. The added Equations are shown as follows:

$$OCw_{qi} = O_vC_{qi} * W_{qi} \tag{1}$$

$$TOC_{wi} = \frac{\sum_{q=1}^{n} OCw_{qi}}{\sum_{q=1}^{n} OCw_{qi}max} \tag{2}$$

where:

- q identifies each quality characteristic
- i identifies each iteration
- n is the number of quality characteristics defined
- OC_{wi} is the obtained coverage with weights per quality characteristic within the iteration
- O_vC_{qi} is the overall coverage per quality characteristic per iteration
- W_{qi} is the weight per quality characteristic for each iteration which is an integer
- TOC_{wi} is the total obtained coverage of QARs per iteration
- $\sum_{q=1}^{n} OCw_{qi}max$ is the maximum possible value for the sum of coverage with weights when all the QARs passed within the iteration

To include these weights, the mathematical basis presented by S. Morasca [15] was used, who describes the use of weighted sums in the process of defining measures in Software Engineering. The author considers a given set of n real valued weights $(x_1, ..., x_n)$ and of n real valued weights $(w_1, ..., w_n)$, and define weighted sum ws as the sum of i from $1..n$ of $w_i * w_i$. For the case in which the sum of w_i is equal to 1, the weighted sums are said to be normalized.

Later on, once the coverage by quality characteristic and the quality value of the iteration have been calculated, the weight is now included, which will be multiplied by each coverage obtained, which will allow get a new quality value TOC_{wi}.

Traditionally, the most frequent way of defining weights is with coefficients, whose sum gives 1. In this case, it was not considered convenient to apply in PQEMw because, if a weight between 0 and 1 is taken, the multiplication between the obtained coverage for the quality characteristic and the defined weight, may result in a lower decimal value than the one originally obtained when calculating said coverage. This event may cause the stakeholder to misinterpret the chosen weight.

If, for example, the coverage value obtained for a quality characteristic was 0.23 and the weight is 0.11, the coverage result is 0.025, which is a number less than 0.23. These resulting values might confuse the stakeholder, when comparing quality results, thinking that the quality has dropped (by a decreasing value) when in reality that may not be the conclusion.

Taking this as a basis, we now propose a weight associated with each quality characteristic that can take an integer value in a range defined by the stakeholder (for example: 1 to 5), always considering positive integer values. This value represents the highest level that can be considered when carrying out the quality analysis, and that will establish the range of possible values that the W_{qi} can take.

Following the idea that this weight can only take positive integer values, and considering the logic proposed by R. Likert in his method with evaluation scales with respect to a set of elements (which in our case would be the weights to choose from for each quality attribute), we are considering a range between 0 and n for the weights, where n is a positive integer value.

Even though, the stakeholder does not specify his level of agreement or disagreement with respect to a symmetric of statements, Likert's methodology works as a basis to analyze the idea of choosing a viable value of this positive integer that allows later to understand the level of quality achieved, based on the characterization of the domain in which the product to be analyzed is embedded.

It is important to understand that $\sum_{q=1}^{n} O_v C w_{qi} max$ allows to calculate the maximum value of coverage of the iteration, in an ideal condition; that is the one in which all quality attribute requirements (QARs) of each quality characteristic passed successfully. This result allows to simplify the TOC_{wi} value, allowing the stakeholder to have a similar value comparable to the previously TOC_i obtained.

That it is a whole number helps to make the analysis of the results uniform and understandable for the decision maker.

In the same way, the denominator in Equation (2) is greater than the denominator, and therefore, it will always be a number between 0 and 1. In the case that the numerator is 0, it implies a situation in which none of the QARs were successful. If the numerator is 1, then it is the ideal situation where all the QARs passed. These two extreme values lead to the justification that the possible range of values is between 0 and 1.

4 Illustrative example

The main goal of this section is to study the application of PQEMw to the quality analysis of two scenarios, in order to obtain the feasibility and viability of using weights as well as understanding how the weighting on each quality coverage obtained through PQEM impacts on the quality value. As such, we schematized the methodology for each scenario as follows: (1) applied the steps of PQEM to the scenario (scenario 1 will have the full procedure description), (2) obtain the quality level for the iteration, (3) define the set of weights based on the domain and the stakeholder's needs, (4) calculate an ideal run where all QARs passed, (5) obtain the results for the run with the obtained coverage values and their coverage weights, and (6) extract conclusions from comparisons.

4.1 Scenario 1: app within the healthcare field

In this case, we will consider the third iteration of an application embedded in the medical field, which ensures that the physical recovery of cardiac patients who are part of a cardiac rehabilitation program can take place in an environment outside hospitals. The application has a client-server architecture, with a web and mobile app, where the mobile version is used by the patients to monitor their heart rate and to keep track of their exercises, and the web version is used by the doctors to ensure a proper monitoring of each of their patients.

As shown in Table 1, the third iteration of the app were measured around 10 quality characteristics and 293 QARs, the quality value obtained was $TOC_3 = 0.90$, and so it passed the expected quality level for the iteration, which was defined at 0.80. Based on that the two previous iterations achieved a proper quality level, it is feasible to conclude the succession of iterations of the application, and it is not necessary to rerun the measurement. With respect to the selection of weights, and to carry out the coverage calculations, the value of weights will vary between 1 and 5, in order to obtain the TOC_{w3} value, which will provide the quality value of the iteration with the weights.

Ideal run with PQEMw A run was made with the application of PQEMw, considering that the coverage of each quality characteristic was reached by 100%, meaning that all the QARs for each quality characteristics managed to pass the acceptance criteria. The latter is the ideal state for an iteration, where every

single one of the QARs were present in the application. Table 1 shows the ideal situation where all quality attributes requirements are present within the application under study, and also we already included a selection of weights between 1 and 5, to analyze this case based on the stakeholders ideas and domain characteristics.

Table 1. Scenario 1: Ideal run when all $QARs$ passed.

QC	$NQAR_{q3}$	$NpQAR_{q3}$	OC_{q3}	W	O_vC_{q3}
Availability	14	14	0.05	5	0.23
Fault-tolerance	4	4	0.01	4	0.05
Recoverability	7	7	0.02	3	0.07
Functional suitability	57	57	0.19	2	0.38
Interoperability	22	22	0.08	5	0.37
Modifiability	67	67	0.23	3	0.68
Performance efficiency	17	17	0.06	4	0.23
Security	30	30	0.10	5	0.51
Usability	64	64	0.22	3	0.65
Portability	11	11	0.04	2	0.07
Total	293	293	$TOC_3 = 0.90$	-	3.29/3.29=1

The total coverage value obtained in that ideal condition was 3.29, and it represents the highest possible value that can be computed when all the QARs passed and for that set of defined weights. Consequently, when computing 3.29/3.29, the resulting value is $TOC_{w3max} = 1$.

Regular run with PQEMw Based on this maximum value reached, we can now calculate and analyze the results of the application of PQEMw to the set of quality characteristics with the real coverage values obtained with the inclusion of the weights previously used. Table 2 describes this regular run with the same set of weights as Table 1.

Starting from the previous data, the TOC_{w3} value results in 0.88, which is calculated as the quotient between the total obtained coverage (2.92 - see Table 1) and the maximum value for the iteration (3.29 - see Table 2). In this case, the value obtained (0.88) is a value close to the maximum possible value on the scale and compared to the previous value of $TOC_3 = 0.90$, it can be said that there is similarity between them and therefore, the previously selected weights that multiply said individual coverages do not affect the resulting value and it is possible to understand said values in terms of quality.

Regular run with PQEMw - different set of weights TO properly understand, how the weighting on each quality coverage obtained through PQEMw impacts on the quality level value (TOC_3 and TOC_{w3}), we changed the set of

Table 2. Scenario 1: PQEMw results for the regular run.

QC	$NQAR_{q3}$	$NpQAR_{q3}$	OC_{q3}	W	O_vC_{q3}
Availability	14	14	0.05	5	0.24
Fault-tolerance	4	4	0.01	4	0.05
Recoverability	7	6	0.02	3	0.06
Functional suitability	57	50	0.19	2	0.34
Interoperability	22	12	0.08	5	0.20
Modifiability	67	60	0.23	3	0.61
Performance efficiency	17	16	0.06	4	0.22
Security	30	27	0.10	5	0.46
Usability	64	64	0.22	3	0.66
Portability	11	11	0.04	2	0.08
Total	QARs=293	Passed QARs=264	$TOC_3 = 0.90$	-	2.92/3.29=0.88

weights to let them take some value between 5 and 10, and re run the full procedure of coverage. The total coverage value obtained in that ideal condition was 7.85, and so, we calculated later the TOCw as shown in Table 3.

Table 3. Scenario 1: PQEMw results for the regular run with a different set of weights.

QC	$NQAR_{q3}$	$NpQAR_{q3}$	OC_{q3}	W	O_vC_{q3}
Availability	14	14	0.05	10	0.47
Fault-tolerance	4	4	0.01	9	0.12
Recoverability	7	6	0.02	8	0.16
Functional suitability	57	50	0.19	7	1.195
Interoperability	22	12	0.08	9	0.36
Modifiability	67	60	0.23	7	1.43
Performance efficiency	17	16	0.06	8	0.43
Security	30	27	0.10	10	0.92
Usability	64	64	0.22	8	1.74
Portability	11	11	0.04	5	0.18
Total	QARs=293	Passed QARs=264	$TOC_3 = 0.90$	-	7.05/7.85=0.89

By changing the weights, we tried to understand what happened with the coverage values. In this context, even though the coverage values are higher than the idea of a quality level between 0 and 1, the Equations defined allows a certain normalization of these values, letting the stakeholder or manager to easily understand the result. By comparing the final value, $TOC_3 = 0.90$ and $TOC_{w3} = 0.89$, we obtained a similar value between those quality values, and so, the weights can increase the general coverage value, but the final quality level can become a number between 0 and 1, providing the same idea of "how well performed the iteration".

4.2 Scenario 2: e-commerce web app

In this case, we measured an e-commerce web app were the most critical qualities are those that impact the end user as well as the ones that should be considered for the company powering the e-commerce website, considering ten quality characteristics and 343 QARs. As such, the stakeholders schematized the priorities for the quality characteristics as the ones with the higher values of weights as shown in Tables 4 and 5.

Table 4. Scenario 2: Ideal run when all QARs passed.

QC	$NQAR_{q3}$	$NpQAR_{q1}$	O_vC_{q1}	W	O_vC_{q1}
Availability	19	19	0.055	5	0.277
Fault-Tolerance	12	12	0.035	4	0.140
Recoverability	30	30	0.087	2	0.175
Functional Suitability	45	45	0.131	3	0.394
Modifiability	35	35	0.102	3	0.306
Performance Efficiency	48	48	0.140	5	0.700
Testability	26	26	0.076	3	0.227
Security	41	41	0.120	5	0.598
Usability	64	64	0.187	5	0.933
Maintainability	23	23	0.067	5	0.18
Total	343	343	1	-	4.08/4.08=1

From the ideal run, we obtained $OvC_{wq1} = 4.08$, we reached to a $TOC_{w1} = 0.705$ quality level with weights. The quality level for this iteration is $TOC_1 = 0.708$, which is also similar to the $TOC_{w1} = 0.705$.

Table 5. Scenario 2: PQEMw results for the regular run.

QC	$NQAR_{q3}$	$NpQAR_{q1}$	O_vC_{q1}	W	O_vC_{q1}
Availability	19	14	0.041	5	0.23
Fault-Tolerance	12	10	0.029	4	0.12
Recoverability	30	24	0.07	2	0.16
Functional Suitability	45	30	0.087	3	0.30
Modifiability	35	27	0.079	3	0.23
Performance Efficiency	48	21	0.061	5	0.35
Testability	26	15	0.044	3	0.15
Security	41	34	0.099	5	0.58
Usability	64	57	0.166	5	0.97
Maintainability	23	11	0.032	5	0.18
Total	343	238	$TOC_1 = 0.708$	-	2.88/4.08=0.705

With the incorporation of weights to the method, one of the key ideas that are to be considered is the value to assign to each weight, and, regarding the

strategies developed by [15], we worked on an integration of measurer-defined weights and default weights; because in the first one the weights are chosen based on the goals of the application to which their quality level, and considering its domain, since the latter impacts what is the weighting and the mandatory quality characteristics; while the third strategy is the case when all the weights are 1, effectively considering that all the quality characteristics weigh the same.

Likewise, these weights should be representative of what the stakeholder consider as a priority with respect to the software product. As such, the highest values of chosen weights shows the importance described by the stakeholders with respect to each quality characteristic.

5 Discussion

Weighted sums continue to be used in software engineering as a means of quantifying a set of software attributes. The idea is to use a weighted sum to combine several lower-level measures to build a single higher-level measure, which quantifies different aspects of an attribute at the same time, and which you can use for both internal and external software attributes [14, 15].

The literature reports studies in which weighted sums have been used to specify unique numbers that allow evaluating a functionality, quality in general or other attributes, in order to understand which of them has a greater degree of functionality (or level of completeness), higher quality, among others.

In our case, we have used these weighted sums to work with quality characteristics and sub characteristics based on ISO/IEC 25010:2011, in order to obtain the quality level of a set of characteristics in each iteration of the life cycle of a software product. With PQEM it is possible to obtain the level of quality by means of TOC_i that represents a multidimensional value, obtained from the coverage of each quality characteristic analyzed; while TOC_{wi} summarizes the coverage multiplied by a weight defined by the stakeholder embedded in the domain.

One of the advantages of using weighted sums in the dimensional reduction, due to fact that they are able to transforms a n-dimensional problem into a unidimensional one, where a total order can always be found. In our case, we reduce into one number (TOC_i) the coverage of each quality characteristic, and this number allows to understand the quality performance of each iteration within the life cycle of the product. Simplicity is another advantage, because the defined Equations are quite simple to understand since they are only quotients between the amount of QARs passed and total.

With the evaluation carried out as illustrative examples, we were able to establish that the use of weights in quality measurement allows defining a decision with a higher level of complexity, together with the original decision of whether the iteration passes or fails (based on that value TOC_i obtained).

At the individual coverage level, the inclusion of weights varies the relative value of each quality characteristic with respect to the total QARs. But this variation does not affect the value of total coverage for that iteration, which, when

implementing the quotient for the coverage in the ideal situation, normalizes that value of total coverage (meaning: the quality level) and a number similar to the level of quality obtained is obtained. In other words: that relative variation of coverage by the multiplication by the weights does not change the resulting quality level.

Consequently, the fact that the inclusion of weights does not vary the final decision (from two values that are similar to each other: TOC_i and TOC_{wi}), confirms the idea that weights expand the capacity for decision making because it amplifies the magnitude of the decision, based on the level of quality obtained and the relative importance of each of the quality characteristics.

In the same way, and considering that slight variation between the resulting quality values TOC_i and TOC_{wi}, it can be said that those quality characteristics whose weight is 5 (or the highest possible value within the range chosen for the set of weights) is mandatory. , which implies that for the iteration to pass, those quality characteristics must be fulfilled in order to advance the iteration.

Also, it is true that with just only one aggregate number, it is possible to lose information about the original measures which may be useful when making a decision. It is true that TOC_i and TOC_{wi} summarizes the quality performance of the application within the iteration, and that sums the coverages of each quality characteristic, it doesn't eliminate those original values, rather, it presents them in a different way, so going back a step can quickly obtain those coverage values.

Likewise, usually the final result of a weighted sum is a measure that does not have a measurement unit such as TOC_i or TOC_{wi}, although they have a range of possible values, they do not have an associated unit of measure, but rather those same values are represented in a trend graph of the quality level by iteration.

Another issue is the alleged objectivity on each selected weight, which in our case is fully dependent on the stakeholders point of view and the domain settings, which this level of subjectivity may affect the decision making of the project manager when he or she has to decide to move to the next iteration or not. This subjectivity may affect on not being able to obtain a one-size-fits-all measure that can be a fully sensible measure for an internal attribute or a useful measure for an external attribute in all domains.

Even though weighted sums are flexible enough to be applicable in many different cases, the excessive flexibility may be a problem when the definition of the weights defining a measure of its own means that it is not possible to generalize and compare measures in domains in which different versions are defined for a measure.

6 Conclusions

The quality of a system is extremely important, and software metrics became an essential part to understanding whether the quality of the software corresponds to what the stakeholders needs [26]. Those needs are characterized within the ISO/IEC 25010, as a set of quality characteristics and sub-characteristics [18].

Based on the previous definition of the Product Quality Evaluation Method (PQEM) [5], the present article introduced the extension of this method with the addition of weighting up each quality attribute trough the selection of a weight that is included within a set of equations in order to obtain a quality value called TOC_{wi}. In other words, the addition of weights lets the stakeholder to specify their point of view based on the domain impact of each quality attribute.

Also, we presented two illustrative examples to demonstrate its applicability. Knowing what to measure is a recurrent problem in a data-driven approaches, using GQM for identifying the quality attributes ensures that the assessment of the product is adapted to the organization applying the proposed method. To achieve the applicability, a quality model should not only be an assessment model but also a usable and intuitive guideline to increase quality [26].

The illustrative examples also shown the viability of setting weights values per each quality characteristic, and the quality analysis related to each run. The use of weights allows the stakeholder to specify their importance on the set of quality characteristics, based on their previous experience, and specific domain. As future work, we will conduct a validation within a business setting.

References

1. Lehman, M. M., Belady, L. A. (Eds.). (1985). Program evolution: processes of software change. Academic Press Professional, Inc..
2. Lehman, M. M., Ramil, J. F. (2001). Rules and tools for software evolution planning and management. Annals of software engineering, 11(1), 15-44.
3. Kan, S. H. (2003). Metrics and models in software quality engineering. Addison-Wesley Professional.
4. Sneed, H. M., Merey, A. (1985). Automated software quality assurance. IEEE Transactions on Software Engineering, (9), 909-916.
5. Falco, M., Robiolo, G. (2019, November). A Unique Value that Synthesizes the Quality Level of a Product Architecture: Outcome of a Quality Attributes Requirements Evaluation Method. In International Conference on Product-Focused Software Process Improvement (pp. 649-660). Springer, Cham.
6. Falco, M., Scott, E., Robiolo, G. (2020, December). Overview of an Automated Framework to Measure and Track the Quality Level of a Product. In 2020 IEEE Congreso Bienal de Argentina (ARGENCON) (pp. 1-7). IEEE.
7. Falco, M., Robiolo, G. (2021, July). Product Quality Evaluation Method (PQEM): A Comprehensive Approach for the Software Product Life Cycle. In 7th International Conference on Software Engineering (SOFT).
8. Basili, V. R. (1992). Software modeling and measurement: the Goal/Question/Metric paradigm.
9. Morasca, S. (2004, September). On the definition and use of aggregate indices for nominal, ordinal, and other scales. In 10th International Symposium on Software Metrics, 2004. Proceedings. (pp. 46-57). IEEE.
10. Hovorushchenko, T. (2017). Method of Evaluating the Weights of Software Quality Measures and Indicators. Application and Theory of Computer Technology, 2(2), 16-25.
11. Ivan, I., Zamfiroiu, A., Doinea, M., Despa, M. L. (2015). Assigning weights for quality software metrics aggregation. Procedia Computer Science, 55, 586-592.

12. McCall, J. A. (2002). Quality factors. encyclopedia of Software Engineering.
13. Boehm, BW and Brown, JR and Kaspar, H and Lipow, M and Macleod, GJ and Merrit, MJ. (1978) Characteristics of Software Quality. North-Holland.
14. Fenton, N., Bieman, J. (2014). Software metrics: a rigorous and practical approach. CRC press.
15. Morasca, S. (2010, May). On the use of weighted sums in the definition of measures. In Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics (pp. 8-15).
16. Lavazza, L., Morasca, S., Robiolo, G. (2013). Towards a simplified definition of Function Points. Information and Software Technology, 55(10), 1796-1809.
17. Bass, L., Clements, P., Kazman, R. (2003). Software architecture in practice. Addison-Wesley Professional.
18. ISO/IEC 25010, System and Software quality models, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010
19. ISO/IEC 25023 - Measurement of system and software product quality, https://www.iso.org/standard/35747.html
20. Chilenski, J. J., Miller, S. P. (1994). Applicability of modified condition/decision coverage to software testing. Software Engineering Journal, 9(5), 193-200.
21. Gritzalis, D. A. (1998). Enhancing security and improving interoperability in healthcare information systems. Medical Informatics, 23(4), 309-323.
22. Hovenga, E. J. S. (2008). Importance of achieving semantic interoperability for national health information systems. Texto Contexto-Enfermagem, 17, 158-167.
23. Benson, T., Grieve, G. (2016). Principles of health interoperability: SNOMED CT, HL7 and FHIR. Springer.
24. Blobel, B. (2020, September). The value of domain information models for achieving interoperability. In Phealth 2020: Proceedings of the 17th International Conference on Wearable Micro and Nano Technologies for Personalized Health (Vol. 273, p. 75). IOS Press.
25. Morasca, S. (2004, September). On the definition and use of aggregate indices for nominal, ordinal, and other scales. In 10th International Symposium on Software Metrics, 2004. Proceedings. (pp. 46-57). IEEE.
26. Mordal, K., Anquetil, N., Laval, J., Serebrenik, A., Vasilescu, B., Ducasse, S. (2013). Software quality metrics aggregation in industry. Journal of Software: Evolution and Process, 25(10), 1117-1135.