PRODUCT QUALITY EVALUATION METHOD (PQEM): TO UNDERSTAND THE EVOLUTION OF QUALITY THROUGH THE ITERATIONS OF A SOFTWARE PRODUCT

Mariana Falco and Gabriela Robiolo

LIDTUA/CONICET, Engineering School, Universidad Austral, Pilar, Buenos Aires, Argentina

ABSTRACT

Promoting quality within the context of agile software development, it is extremely important as well as useful to improve not only the knowledge and decision-making of project managers, product owners, and quality assurance leaders but also to support the communication between teams. In this context, quality needs to be visible in a synthetic and intuitive way in order to facilitate the decision of accepting or rejecting each iteration within the software life cycle. This article introduces a novel solution called Product Quality Evaluation Method (PQEM) which can be used to evaluate a set of quality characteristics for each iteration within a software product life cycle. PQEM is based on the Goal-Question-Metric approach, the standard ISO/IEC 25010, and the extension made of testing coverage in order to obtain the quality coverage of each quality characteristic. The outcome of PQEM is a unique multidimensional value, that represents the quality level reached by each iteration of a product, as an aggregated measure. Even though a value it is not the regular idea of measuring quality, we believe that it can be useful to use this value to easily understand the quality level of each iteration. An illustrative example of the PQEM method was carried out with two iterations from a web and mobile application, within the healthcare environment. A single measure makes it possible to observe the evolution of the level of quality reached in the evolution of the product through the iterations.

KEYWORDS

Quality Characteristics, Product Quality Measurement, Coverage, Quality Attributes, PQEM

1. INTRODUCTION

Quality is key when evaluating the properties of a software product, and software metrics became an essential part to understanding whether the quality of the software corresponds to what the stakeholders needs [1]. The standard ISO/IEC 25010 characterized those needs with a set of quality characteristics: Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability, as well as the subset of sub-characteristics per each characteristic [2]. Considering the diverse stakeholders participating in software projects such as developers, managers, and end users, quality needs to be evaluated at different levels of detail.

Based on the above, several quality measures have been proposed to achieve the evaluation and measurement, but the practical application of these metrics is challenged, on the one hand, by the need to combine different metrics as recommended by distinct quality-model methods such as Goal-Question-Metric [3] and Factor-Criteria-Metric [4]; and on the other hand, by the need to reach insights in the quality of the entire software product, based on the metric values obtained for software elements such as methods and classes.

DOI: 10.5121/ijsea.2021.12501

Consequently, a meaningful quality assessment needs to blend the results of various methods to answer specific questions, combining for example cyclomatic complexity with test coverage [1]. As such, project managers and practitioners have different complications when they need to understand the product quality level, in a way that is easy, synthetic and intuitive to identify and extract the status related to each iteration within the software product. For example, when the project manager must make the decision to accept or reject the issues done within an iteration, evaluate the work of the developers, decide on a payment or negotiate a budget extension.

Based on these challenges, our first work was to define a method architecture evaluation method in order to analyse and measure the quality characteristics of a product architecture and its implementation [5]. Based on a deep analysis and feedback from colleagues, we developed a newer version of this method and we called it: Product Quality Evaluation Method (PQEM), which is a five-step method per iteration, whose main goal is to analyse, study, measure and assess the quality level of the different software iterations. PQEM produces a single value between 0 and 1 as the final outcome that represents the product quality level, which is basically the degree to which the software product covers/fulfils its quality attribute requirements [6].

The first step within PQEM is what we called the product setup, where the stakeholder defines the amount of expected iterations that constitutes the development process of the product, as well as the acceptance criteria for the expected quality level per iteration.

As mentioned earlier, PQEM is based on the Goal-Question-Metric approach [3] which is a main part of the method baseline, and through this approach is possible to define a set of goals (related to the quality characteristics), questions (our quality attribute requirements or QARs [6], and a set of quality measures or metrics, which allow to measure their fulfilment, are elicited for later aggregation. It is worth mentioning that the ISO/IEC 25010 [2] provided the set of quality characteristics and sub-characteristics as the main foundation to select what to measure according to the product domain and objectives.

The elicitation process is followed by: a) the measurement itself, b) the collection and synthesizing of the results that include the implementation of the extension of the testing coverage [7] as a quality coverage, and c) the final assessment of the product quality level obtained. Likewise, this process is repeated for each iteration within the product life-cycle; and this method can be applied to every development method that defines iterations, like agile methods; within academia and industry.

The present article is based on a previous work from the authors [32], whose main goal was to present PQEM as well as the results of its application to the second iteration of an application. This extension will introduce and summarize the results of the application of PQEM to two iterations of a web and mobile app embedded in the healthcare domain.

This article is structured as follows: in Section II the related work will be addressed, while Section III will provide the description of the research method used. Section IV will describe and characterize each of the steps within the Product Quality Evaluation Method (PQEM). Section V will contain an illustrative application of the method with the results of two iterations from a web and mobile apps, while Section VI will address the discussion and threats to validity. Finally, Section VII will describe the conclusions and future work.

2. RELATED WORK

Both the definition of the architecture of a product and the specification of quality characteristics and QARs are decisions that should not be taken lightly because they have a high impact on the

state of the final product. Even though scenario-based architectural assessment techniques [8] are a well-established approach for performing structured evaluations of architectural designs, these techniques are not widely used in industry. A complete analysis was made in [5] with respect to the first iteration of the application presented here, and its comparison with other architecture-based methods.

E. Woods [9] created an architectural review method called Tiny Architectural Review Approach (TARA), which focuses on how well a particular architecture supports a set of key requirements, opposite of what most scenario-based methods like ATAM [8] do. TARA allows for the situation where the system has already been implemented, but PQEM can be applied while the software is in development. PQEM includes five steps per iteration, while TARA is defined with seven steps. Considering the seven steps in a TARA session, one of the main differences with PQEM is that it does not include metrics to analyse the quality characteristics, but in Step 3 they analyse system's production metrics.

TARA approaches only test coverage after running all automated test available, while PQEM extends this concept to analyse the coverage of all quality characteristics per iteration of a product, defining several equations to compute these values. Unlike TARA, PQEM reach to findings and conclusions per each iteration through the TOC quality level, which is a number between 0 and 1; and this number is able to show how close the implementation was to the defined acceptance criteria.

Later on, some authors agreed that managing the cost-effective evolution of industrial software systems represents a challenge based on their complexity and long lifetimes. As such, Koziolek et al. [10] applied several state-of-the-art approaches, to combine them into a holistic lightweight method called MORPHOSIS, which facilitates sustainable software architectures. Consequently, their main focus is sustainability, while our main target is to achieve a proper level of quality that will have impact not only in the sustainability but in the set of quality characteristics included. This method includes three phases: evolution scenario analysis, architecture enforcement and architecture-level metrics tracking.

In the first phase, the authors conducted an evolution scenario analysis according to an extended version of the ALMA method [11], from which they were able to perform a combined top-down, bottom-up scenario elicitation. This is a difference with our work, because apart from not being scenario-based, the elicitation process is not based on ALMA instead the Goal-Question-Metric approach is implemented. The second phase allows to treat the dependencies between module layers, and finally, within the third phase they have found several architecture-level code metrics that measured different aspect of sustainability. The set of metrics measure the quality of modularization of a non-object-oriented software system, and the authors employ the notion of API as the basis for the metrics [12].

They have used Goal Structuring Notation to break down maintainability according to ISO/IEC 25010. As such, they have not focused on the entire set of quality characteristics defined by the ISO/IEC, like PQEM does in the elicitation process. Several authors mentioned that it is important to comprehend the consequences of the decisions on the various software engineering artefacts, like code, test cases, deployments, among others; when analysing the impact of a change request.

As a summary, within PQEM the elicitation is based on the GQM method, to specify the needs of stakeholders in the form of goals, questions, metrics and acceptance criteria for each question. None of the studies proposed any form of synthesis of the analysis, as such, we introduce the definition and calculation of coverage values for each selected quality characteristics, and for the

entire product, which leads to the achievement of a multidimensional number as a summary value of the achieved quality level as final output. Finally, the focus of this method is oriented to the measurement of quality characteristics.

3. RESEARCH METHOD

Our research method is embedded within the design-science paradigm, which advocates the problem-solving perspective seeking to create innovations that define ideas, practices, capabilities, and products through analysis, design, implementation, and management; in order to achieve efficiency and effectiveness [13].

These authors defined a set of guidelines to assist the community to understand the requirements and necessities for effective design-science research. Considering that the paradigm comprises problem conceptualization, solution design, and validation, Runeson et al. [14] stated that it fits as a frame for empirical software engineering research with the goal of providing theoretical knowledge for practical solutions related to real-world software engineering challenges.

3.1. Problem Relevance

Quality plays a major role for end-users, because it is a confirmation of all requirements were designed and developed according to their needs [15,16]. A meaningful quality assessment needs to combine the results of various methods to answer specific questions, joining for example cyclomatic complexity with test coverage [1]; and also, the assessment needs to be able to define a model, broken down into different quality characteristics and sub-characteristics.

Project managers and practitioners have different difficulties when they need to understand the product quality level from every iteration and from the full product, and this understanding can occur when the project manager must make the decision to accept or reject the issues or tasks done as a part of an iteration or even the entire release [17], evaluate the work of the developers, decide on a payment or negotiate a budget extension. PQEM allows to monitor the evolution of the quality level within the product life cycle.

3.2. Design as an Artifact

Based on the previous challenges, the present paper introduces Product Quality Evaluation Method (PQEM) which is a five-step method per iteration, which concedes the managers and practitioners study, measure and understand the quality level of a software product. The final output is a numerical unique single value between 0 and 1, which represents the product quality level. This quality value can be thought as an aggregated measured because is obtained through the quality coverage of each quality characteristic measured, as well as a multidimensional value due to the fact that the quality level synthesize the quality level achieved by each quality characteristic or sub-characteristic.

The multidimensional component of the PQEM method can be understood as the degree to which the software product covers the set of quality attributes requirements [6]. It is worth mentioning that this process is repeated after each iteration, and even though is currently being performed with a spreadsheet, a tool that facilitates the application and use of PQEM is being developed. Consequently, what is lacking in systematization as far as the method is concerned, it is being put together on to achieve automation from the tool.

3.3. Design Evaluation

The evaluation of the method is carried out through an illustrative example within the academia, by applying it to a mobile application. We will conduct a case study within an industrial setting, in order to obtain information on measuring the acceptance and usefulness of the PQEM method by practitioners and companies.

3.4. Research Contributions

The main contributions of PQEM is six-fold:

i) we have built a product evaluation method that includes quality characteristics as defined by ISO/IEC 25010 [2];
ii) we have extended the use of testing coverage to define quality characteristics coverage, and product coverage;
iii) we have defined an aggregated measure that allows a fine-grained analysis of the results per quality characteristic;
iv) we have extended the acceptance criteria for functional and non-functional requirements;
v) we have synthesized the functional and non-functional requirements on a number that represents the quality level of a product;
vi) we defined a method that let the practitioners compared the obtained quality level per each iteration within the software life cycle.

3.5. Research Rigor

The method connects software quality evaluation and non-functional requirements, two areas of research that have a long history in contributing to each other's development. Also, the method is grounded on solid achievements from two disciplines, on the one hand, Goal-Question-Metric [3,17] has been empirically validated in many case studies, and demonstrated its worth in studies on requirements. On the other hand, the PQEM method has foundations on the standard ISO/IEC 25010 [2] which set the baseline of quality measurements and quality characteristics.

3.5. Design as a Search Process

We have defined the PQEM method, the validation has been done with small mobile and web applications, the design and development process of an automated tool based on PQEM has been started, we will seek to validate this tool with a case study on an industrial setting, and finally, we are defining an automated framework that will allow the practical implementation of PQEM, and it might also be used for the implementation of product quality measurement or quality model processes not restrictive to software.

Each of these steps generates useful feedback to improve and optimize what has been proposed. Considering that source code metrics are essential and they allow us to set grounds about the quality characteristics measured by the metrics, the automated tool will be able to integrate different automated tools that measure quality attributes in order to obtain a full quality analysis of each software product analysed; for example: SonarQube [18].

3.6. Communication of Research

Technology-oriented audiences are provided with sufficient detail to enable them to be able to replicate each step of the PQEM method. Also, the management-oriented audiences are able to understand whether the organizational resources should be committed to using the method within their specific organizational context. Our main goal is to promote the free software project as well as case studies in the industry.

4. PRODUCT QUALITY EVALUATION METHOD (PQEM)

This section describes the PQEM steps which are the following: **S1)** Product Setup, **S2)** Elicitation of QARs, **S3)** Measurement, **S4)** Collect Results, and **S5)** Assessment, as shown in Figure 1. It is worth mentioning that the first step (called product setup) should be performed only once, but steps 2 to 5 should be repeated per each iteration for any software product. The latter would lead to a fully functional software product or application to be deployed to customers.

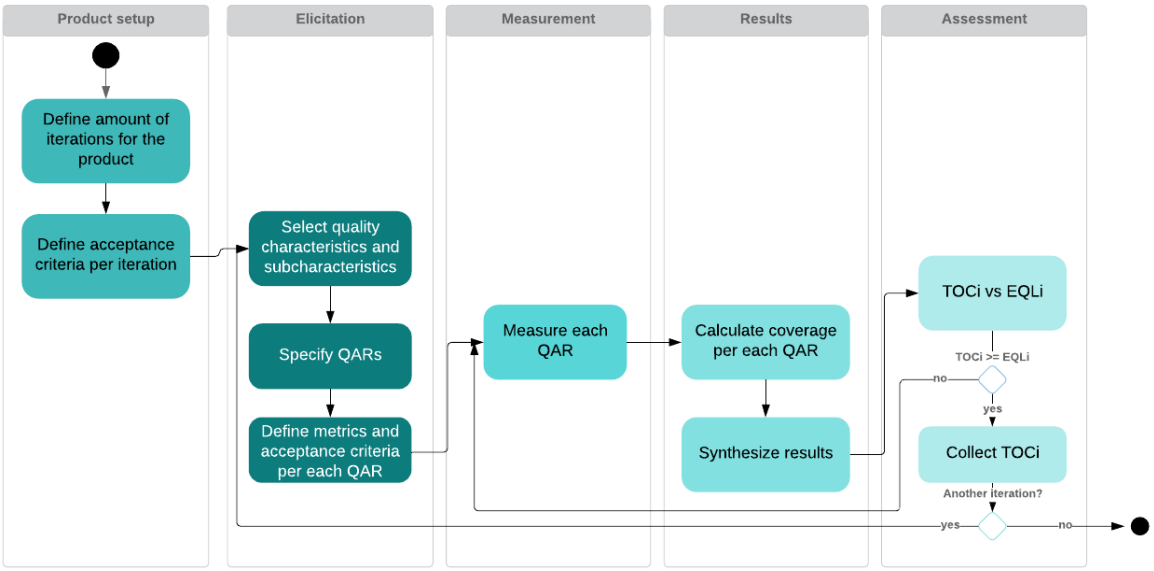

Figure 1. Describing PQEM through an activity diagram.

4.1. Step 1: Product Setup

The first step of PQEM includes the definition of the amount of iterations that the software product is expected to achieved, as well as the characterization and rationale of the acceptance criteria for the expected quality level per iteration EQL_i (as shown in Figure 1), which can be different and incremental from the first one to the last one [19]. The latter is a key point because the acceptance criteria advocates understanding how well the quality for each goal is achieved, allowing a glimpse of the entire product quality, within each iteration.

In this way, the acceptance criteria is defined by the stakeholders, and it is a positive number that can take any value between 0 and 1. For example, if you consider three iterations for a software product, then it can be defined an acceptance criteria of 0.70, 0.80, and 0.90, respectively per each of the three iterations. Based on the previous values, it is possible to see that in this example

it is expected to achieve an improvement in quality as the product grows in functionality. Later on, it is feasible to comprehend that 1 is the best and strictest value of the acceptance criteria, which means that all quality attributes requirements have passed the measurement; while 0 equates to all QARs did not pass. Another point in consideration is that PQEM is a five-step method, but per iteration only four steps are repeated (from Step 2 to Step 5).

4.2. Step 2: Elicitation of Quality Attributes Requirements (QARs)

Nowadays, system engineering is crucial in the industry, and requirements engineering is an important stage of that overall process, where a proper process can generate not only efficiently but rapidly new products [20]. We have conducted this elicitation process through the Goal-Question-Metric approach [3], and so **Step 2.1** will approach the conceptual level with the definition of goals (considering the structure defined in [17], composed of purpose, issue, object, and viewpoint); **Step 2.2** will include the operational level with the specification of the questions by goal, and finally, **Step 2.3** will specify the quantitative level, defining the metrics by question. Note that Step 2 should be validated by the stakeholders.

4.2.1. Step 2.1: Select quality characteristics and sub-characteristics

ISO/IEC 25010:2011 [2] describes the quality model as the cornerstone of a product quality evaluation system, and this standard determines which quality characteristics will be taken into account when evaluating a software product. The product quality model comprises the following quality characteristics: Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. As explained by Estdale & Georgiadou [21], the standard ISO/IEC 25010 provides a huge contribution to establish the delivery performance of different software processes. Regarding PQEM, all of the quality characteristics and their sub-characteristics can be selected by the stakeholders; and considering that each characteristic is mapped with a goal, it is defined by the purpose, issue, object and viewpoint per selected characteristic.

4.2.2. Step 2.2: Specify Quality Attributes Requirements (QARs)

Bass et al. [6] explained that the requirements of a system originate from different sources and forms, like functional, quality attributes and constants. Regarding Step 2.1, the QARs for each of the quality characteristics are now specified by the stakeholder and the development team. These QARs are the questions defined for each of the goals, and for example, in the context of Availability a QAR can be defined as: *"Does the system allows to display trends of vital signs?* or for Interoperability: *"Does the system allows to capture and display data from wireless sensors?"* [22].

4.2.3. Step 2.3: Define Metrics and Acceptance Criteria per each Quality Attribute Requirement (QAR)

In this sub-step, the metrics are defined, and they will provide the necessary information to answer the questions defined in Step 2.2. By defining the limits and parameters of a user story or functionality, and determining when a story is complete and functioning as expected, it is possible to specify an acceptance criteria containing conditions that a software product must satisfy in order to be accepted by the stakeholder [23].

Acceptance criteria are also discussed when defining what requirements must be met in each incremental version of a software product [6]. In this context, we sought to extend these concepts for each of the quality measures in order to determine if this measured value was met or not,

addressing not only functionalities, but also quality characteristics. As such, the acceptance criteria possess a unique importance due to the fact that through its definition it is possible to objectively know whether each QAR is present or not, as well as obtain the TOC value based on the QARs coverage, which can be disaggregated per quality characteristic. Each quality measure requires some acceptance criteria in order to be useful and complete.

4.3. Step 3: Measure and Test each Quality Attribute Requirement (QAR)

This step involves the measurement of each question, executing the defined quality measure, and describing whether the acceptance criteria was met (1 as passed) or not (0 as failed). In the case of Usability, the measurement binds the responses of the number of users who were part of the Usability test. The final value of each test question will be obtained from the application of Equation (6) and (7), which promote the unification of the total number of answers per respondent, for each of the defined questions; allowing later to compute the measurement.

4.4. Step 4: Collect and Synthesize Results

Regarding the evaluation method, we have based our equations on the concept of testing coverage to derive coverage for the different quality characteristics, and the total coverage of QARs per iteration. Based on the foregoing, Equations (1) to (5) describe the calculations needed to compute the quality level of a software product on each iteration.

$$OC_{qi} = p_{qi} / r_{qi} \quad (1)$$

$$EC_{qi} = rq_i / T_i \quad (2)$$

$$O_vCq_i = pq_i / T_i \quad (3)$$

$$TEC_i = \Sigma_{(1 \text{ to } n)} EC_{qi} \quad (4)$$

$$TOC_i = \Sigma_{(1 \text{ to } n)} O_vC_{qi} \quad (5)$$

Where:

- *q* identified each quality characteristic,
- *i* identifies each iteration,
- *n* is the number of quality characteristics defined,
- OC_{qi} is the obtained coverage per quality characteristic for each iteration,
- p_{qi} is the number of passed QARs per quality characteristic for each iteration,
- r_{qi} is the number of QARs per quality characteristic for each iteration,
- EC_{qi} is the expected coverage per quality characteristic per iteration,
- T_i is the total number of QARs per iteration,
- OvC_{qi} is the overall coverage per quality characteristic per iteration,
- TEC_i is the total expected coverage per iteration (which its maximum value is 1),
- TOC_i is the total obtained coverage of QARs per iteration.

TOC_i is a multidimensional value, because it summarizes the obtained quality level of all quality characteristics and sub-characteristics. For each QAR corresponding to Usability, z answers will be obtained according to the number of participants that perform the Usability test. With respect to Usability, each QAR is analysed as follows:

A) It is necessary to unify the z answers from the Usability test that were different from 0 and 1 to become 0 or 1, for example those being a qualitative value like low, very low, medium or high can be unified defining a criterion that all those answers with low and very low will be considered as passed (1) and medium and high as failed (0).

B) Then, all of the values (0s and 1s) for each QAR are summarized, and it is obtained the number that represents the passed QARs.
C) Later on, the coverage per QAR is calculated with Equation (6), where *x* is each QAR.

$$UC_x = pa_x / re_x \quad (6)$$

Where:

- UC_x is the Usability coverage per QAR,
- pa_x is the number of passed answers per QAR,
- re_x is the number of respondents.

If the value obtained with Equation (6) is lower than 0.5 then it is considered as failed, passed otherwise, obtaining the value p_{qi} for Usability; as the sum of the passed values.
D) Finally, once p_{qi} is calculated, it is possible to compute the coverage for Usability itself with Equation (1) and (3); and carry on with the calculations in order to obtain the TOC_i value, through Equation (5).

4.5. Step 5: Assessment of the Product Quality Level

It is possible to perform the analysis of the quality level obtained by means of Equation (5), and the comparison with EQL_i defined by Step 3. In this line, there are two possibilities: following Fig. 1, if TOC_i is bigger or the same to the EQL_i value, then it is possible to collect the TOC_i value. If there is another iteration, the elicitation step will begin. If there is no other iteration, then the process stops. Now, if TOC_i is lower to the EQL_i value, then a new measurement is needed in order to achieve at least the EQL_i value.

4.5.1. Step 5.1: Collect Measurement

Once, the previously defined Equations per each iteration are computed, it is necessary to collect the results to understand the product quality level. As such, Equation (6) allows to construct Equation (7) which contains the list of TOC_i values obtained by iteration $i = (1, 2, ..., y)$. It is worth mentioning that when TOC_i is equal to or bigger of EQL_i (Expected Quality Level) the collection is carried out.

$$TOC_{product} = \{TOC_1, TOC_2, ..., TOC_y\} \quad (7)$$

4.5.2. Step 5.2: Decision and Control

Once all the measurements from Step 4 are done, the TOC*i* value obtained for the current iteration will be compared to the expected quality level (EQL*i*). On the one hand, if the TOC*i* value is lower than the defined EQL*i* value, then it is necessary to return to the measurement step, which requires solving the QARs that are not existing in development. In this step, if there is another measurement, and it end up being equal or exceed the EQL_i defined for that iteration, it is possible to continue with the next one. On the other hand, if the obtained TOC value is greater than the EQL_i value, then the next iteration may begin, going back to Step 2.1.

5. AN ILLUSTRATIVE APPLICATION OF PQEM

We will present an illustrative application of the PQEM method, in order to help non-technical practitioners to understand the steps and the related costs to conduct the measurement.

5.1. Main Goal and Context

Our case study aims to address the implementation of the PQEM method to two iterations of a previously developed application called HeartCare [5], whose main goal is to ensure that the recovery of cardiac patients can take place in an environment outside hospitals.

The three iterations have three types of users and each uses the application in different environments.

- **Patients** will use the mobile application during their exercise sessions, made up of the sum of the assigned exercises. Each patient should use the application as many times as the doctor indicates, trying to achieve rehabilitation. The patient can be indoors or outdoors when doing the exercises.
- **Clinicians** use the app whenever they find it necessary to assign a patient a routine. The doctor is probably inside his office, at a desktop computer with the patient present. The time of use should not be very long, so as not to lengthen waits and time lost by the doctor. Therefore, the charging system must be easy to use.
- The **administrator**, for his part, will not use the application on a recurring basis since it is assumed that changes to equipment and doctors do not occur every day. Administrative staff will use the web application from their office on a computer.

HeartCare, the first iteration included a layered architecture includes a multiagent system and a heart-rate sensor (Polar H10) that helps the patient monitor their heart condition while he or she is in rest position, or while performing a physical exercise, through a mobile device with Android. The literature describes similar examples to HeartCare [24,25]. Also, HeartCare was analysed with PQEM, and obtained a quality level of 0.775 [5].

The second iteration is called Life +, which in its mobile version displays a series of routines, previously loaded by a medical group, which will be carried out by the patient during the course of their rehabilitation. The system collects data on the heart rate through a cardiac sensor administered by the patient, in order to obtain information on the physical state of the patient during his rehabilitation.

That information can be viewed by a medical group through a web interface to obtain a report on their health status and assign new routines. The architecture of the second iteration is based on the layered design principle, and takes into consideration the need to develop separate modules which can evolve independently, where one of these modules is responsible for managing the heart rate sensor.

Regarding the back-end language, an object-oriented paradigm was chosen, using the Java language. Spring Framework was selected because, according to the requirements of the Life + application, it was necessary to have a sufficiently fast reading and writing speed to support data flow from multiple sensors, to be able to maintain structured data, and it is the only one that has fast integration with swagger for automatic generation of documentation.

Then, the technology chosen for the mobile front-end was React Native, since it allowed the integration with Android and IOs required, while for the web version Angular was selected in order to achieve modularization, the separation of the behaviour, the complete separation of the back-end, and also, being developed by Google, facilitates the resolution of problems that may arise in the future. Also, Firebase Cloud Messaging was chosen to send notifications from the backend to the mobile application, which is developed by Google and easily integrated into the Spring Framework.

Finally, BraveHeart is the third iteration where the web version was done with Angular, while the mobile version was used React Native to be able to measure the heart rate via Bluetooth (in this case, an Android Wear). As far as databases are concerned, PostgreSQL was used.

5.1.1. Step 1: Product Setup

With respect to the application, three iterations were defined [5]. And, according to the stakeholder, the first iteration called HeartCare were set with a 0.70 acceptance criteria, the second one with 0.70 as well, and the third one and final was set with an acceptance criteria of 0.85.

5.1.2. Step 2: Elicitation of Quality Attributes Requirements (QARs)

The quality characteristics, questions, metrics and acceptance criteria as well as the results are stored in a structured artifact (spreadsheet), as shown in Table 1. The ID column allows to identify and quickly group each row by quality characteristic, followed by the QAR, the metric and the acceptance criteria. Finally, the Result column contains the result of the measurement made per row for all QARs (1 passed, 0 failed) from the second iteration.

Table 1. Artifact to store data. Example for Fault-Tolerance from the second iteration of the application.

ID	Quality characteristic	Question	Metric	Acceptance criteria	Result
13	Fault-tolerance	Are the amount of crashes under control?	Number of crashes	Number should be less than 10	Passed
14	Fault-tolerance	Are the amount of hangs under control?	Number of hangs	Number should be less than 10	Passed
15	Fault-tolerance	Are the amount of functionality incorrect responses under control?	Number of functionality incorrect responses	Number should be less than 15	Passed
16	Fault-tolerance	Are the amount of updates requiring restart under control?	Number of updates requiring restart under control	Number should be less than 4	Passed
17	Fault-tolerance	Are the amount of incompatibility errors under control?	Number of incompatibility errors	Number should be less than 4	Passed

5.1.2.1. Step 2.1: Select quality characteristics and sub-characteristics

Based on the needs of stakeholders, the following characteristics and sub-characteristics from ISO/IEC 25010 [2] have been selected for second iteration: Availability, Fault-Tolerance, Recoverability, Functional Suitability, Interoperability, Modifiability, Security, Usability, and Portability. In addition to the above quality characteristics, the third iteration also included Performance Efficiency.

In this context, only one goal will be presented to achieve the traceability of the steps, but it is convenient to emphasize that the specific goals of all the quality characteristics have been specified. Instantiating the GQM approach, the goal for Reliability is analyse the delivered product and development process for the purpose of understanding, with respect to reliability and its causes, from the viewpoint of the project manager and user, in the context of the second and third iteration. It is important to mention that the following subsections will use Fault-Tolerance as the quality sub-characteristic to show each step of PQEM, which is part of Reliability.

5.1.2.2. Step 2.2: Specify Quality Attributes Requirements (QARs)

Considering the goal, one of the questions that arises for Fault-Tolerance is: *Are the amount of crashes under control?* as shown in Fig. 2. The full set of QARs by quality characteristic leads to obtain the list of aspects that need to be study in the software product. It is worth mentioning that, based on the needs of the stakeholders, the second iteration included 258 QARs while the number of QARs in the third iteration grew to 293.

5.1.2.3. Step 2.3: Define Metrics and Acceptance Criteria per each Quality Attribute Requirement (QAR)

Based on the previous QAR and following Fig. 2, it is necessary to define the metric and the acceptance criteria; consequently, it is possible to explicit the following metric: *Number of crashes*, and acceptance criteria: *Less than 10 crashes*.

5.1.3. Step 3: Measure and Test each Quality Attribute Requirement (QAR)

In order to fulfil the Result column in Fig. 2, an analysis was carried out to identify whether each QAR were part or not in the iteration under measure. For example, the row with ID 17 shown in Fig. 2 ask whether the amount of incompatibility errors is under control in the application. As such, some tests were carried out to count the incompatibility errors within the web version, the mobile version and the sensor. Only one compatibility error was found, and so this QAR was set as passed. This same procedure was performed for all the QARs.

5.1.4. Step 4: Collect and Synthesize Results

At this point, it is necessary to calculate the coverage for each of the defined quality characteristics. Following the example, Equation (1) allows calculating the coverage value for Fault-Tolerance which gives $OC_{q2} = p_{q2}/r_{q2} = 5/5 = 1$, being $i = 2$ as we are considering the second iteration of the application. It is worth mentioning that within Usability, each QAR was answered by ten respondents, who gave their perspective of the functioning and design of the web and mobile version of the second iteration of HeartCare.

Equation (6) and (7) allowed the unification of the Usability answers, obtaining a single value to represent the result per each Usability QAR. Once the results from Equation (6) and p_{qi} were obtained; Equations (1) to (4) were calculated for all of the characteristics, and therefore completing Table 2, obtaining by means of Equation (5) a TOC_2 value of 0.90; same TOC_3 value for the third iteration as can been on Table 3.

Table 2. Summary of results from the application of PQEM to the second iteration of the web and mobile applications.

Quality characteristic	**r_{q2}**	**p_{q2}**	**OC_{q2}**	**EC_{q2}**	**O_vC_{q2}**
Availability	12	10	0.83	0.005	0.04
Fault-Tolerance	5	5	1	0.02	0.02
Recoverability	7	5	0.71	0.03	0.02
Functional Suitability	59	56	0.95	0.23	0.22
Interoperability	6	4	0.67	0.02	0.02
Modifiability	59	59	1	0.23	0.23
Security	17	15	0.88	0.07	0.06
Usability	64	58	0.91	0.25	0.22
Portability	10	8	0.80	0.04	0.03
Total	**258**	**233**		**$TEC_2 = 1$**	**$TOC_2 = 0.90$**

Table 3. Summary of results from the application of PQEM to the third iteration of the web and mobile applications.

Quality characteristic	r_{q3}	p_{q3}	OC_{q3}	EC_{q3}	O_vC_{q3}
Availability	14	14	1.00	0.05	0.05
Fault-Tolerance	4	4	1.00	0.01	0.01
Recoverability	7	6	0.86	0.02	0.02
Functional Suitability	57	50	0.88	0.19	0.17
Interoperability	22	12	0.55	0.08	0.04
Modifiability	67	60	0.90	0.23	0.20
Performance Efficiency	17	16	0.94	0.06	0.05
Security	30	27	0.90	0.10	0.09
Usability	64	64	1.00	0.22	0.22
Portability	11	11	1.00	0.04	0.04
Total	**293**	**264**		**TEC_3 = 1**	**TOC_3 = 0.90**

5.1.5. Step 5: Assessment of the Product Quality Level

The assessment itself addresses the analysis of the value obtained by the Equation (5) based on the previous calculation of the coverage of all quality characteristics. In this case, an acceptance criteria was defined in 0.70; and following Table 2, the quality level TOC_2 for the second iteration was 0.90. As can be seen, the quality level TOC_2 not only reached but also exceeded the defined acceptance criteria (0.70). As such, when compared with the TOC_1 value obtained for the previous iteration, the application reached the acceptance criteria without an outstanding difference (only with 0.075) and with a bigger technical debt [21,26].

With respect to the third iteration, there was a newer quality characteristic (Performance Efficiency) as well as 35 more QARs to analyse (where the 50% belonged to the added quality characteristic). Based on Table 3, the quality level obtained for this third iteration was TOC_3 = 0.90, which were bigger to the 0.85 acceptance criteria defined for the iteration, and so, it was possible to accept the iteration.

When analysing each of the quality characteristics shown in Tables 2 and 3, it is possible to understand that none of them achieve a huge difference between the expected and the obtained coverage.

5.1.5.1. Step 5.1: Collect Measurement

Based on Equation (5), it is possible to obtain that Equation (7) gives the following result: $TOC_{product}$ = {0.775; 0.90; 0.90}, with the values TOC_i (i = 1,2,3) from the three iterations.

5.1.5.2. Step 5.2: Decision and Control

In this context, and due to the fact that the TOC_i value was bigger than the EQL_i defined, then it is possible to finish the iterations, due to the fact that all three iterations achieved (first iteration) or surpassed (second and third iterations) the acceptance criteria previously defined.

5.2. Trends analysis

Figure 2 shows the trend in the quality values achieved in each of the three iterations. It also allows to perform a visual comparison between the Expected Quality Value (EQL) and the quality value obtained (Total Obtained Coverage or TOC): for the first iteration (i = 1), it is possible to see $EQL_1 = 0.7$ and $TOC_1 = 0.775$; for i = 2, $EQL_2 = 0.8$ and $TOC_2 = 0.90$; and for i = 3, $EQL_3 = 0.85$ and $TOC_3 = 0.90$.

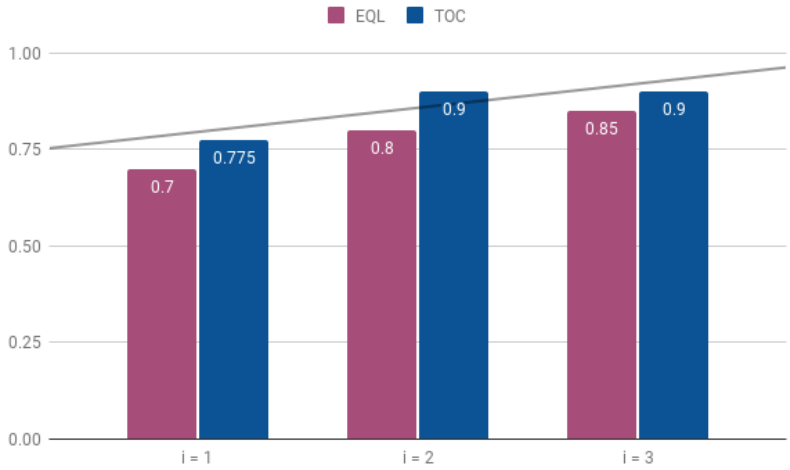

Figure 2. Trend analysis from the three iterations.

From Figure 2, it is possible to understand that as the iteration progressed, the acceptance criteria (or that expected quality value) was gradually increasing, which translates into an increase in the expected quality level for each iteration. In other words, both stakeholders and quality leaders agreed that, given a rate of variability of the functionalities and the requirements of the domain of each iteration, the acceptance criteria had to grow to adapt to those needs.

Fortunately, the TOC_i quality values obtained in the first measurement of each iteration did not require a new measurement, but it was possible to continue to the next iteration. This does not imply that there weren't points to improve, on the contrary, those differences obtained between EQL and TOC (technical debt) are points that should have been worked on between each iteration as part of the revision of each defined QAR, and in a fourth, if the stakeholders decide a new development. That review of the entire set of QARs between iteration and iteration after the measurement, gives the ability to the product manager or the quality leader to be able to further detail those quality needs, which will then be contrasted with the deliverable.

If the quantity of QARs per iteration (138, 258, 293) and the expected quality level (0.7, 0.8, 0.85) is considered, it is possible to understand that as there were adjustments and/or functional requirements that impacted on the architecture or the BD model, along with the need of the stakeholders to achieve a higher level of quality due to the demands of the domain, the quantity of QARs per iteration was increased, also reflected in the increase in the expected level of quality (EQLi).

The following corollaries, between iteration and iteration, were extracted:

- The requirements were adjusted and new functionalities were added.
- The required quality level (EQLi) was increased by stakeholder request.

- Increased the quantity of quality attribute requirements (QARs) based on the increase in quality required.
- Quality characteristics were added to measure.

Figure 2 also reinforces the idea that the application of PQEM allows to achieve this analysis and comparative in a simple and intuitive way, with a single multidimensional value representing quality; and from which, it is possible to make the decision to advance or not to iteration, based on the coverage values obtained.

6. DISCUSSION

Recent literature shows that some authors have studied of analysing a software product, its architecture and its quality attributes, allowing also the measurement of quality measures. However, the main issue is if the practitioners are able to properly identify the quality level of each defined iteration from a software product. In this context, PQEM is a five-step method that can be used to measure the quality level of each iteration within the life cycle of a software product, providing a multidimensional single value (TOC_i) that shows how well the set of quality characteristics are represented within the product. PQEM embedded ISO/IEC 25010 [2], the Goal-Question-Metric approach to perform the elicitation process, and the extension of the testing coverage for a set of quality characteristics.

The latter is another highlight, because the extension of the testing coverage allowed defining the coverage for each quality attribute in each iteration. These coverage values have been calculated for TOC_1, the first iteration of the application, which gave a value of 0.775. The acceptance criteria was set on 0.70, so it is possible to say that the quality level was achieved. Now, the TOC_2 value, that it was defined to be equal to or exceeds 0.80; and after applying PQEM it was obtained a coverage of 0.90, from which it can be understood that the iteration can still improve by about 10 percent, considered as technical debt [26]. And, the TOC_3 value was equal to 0.90, surpassing the 0.85 previously defined as acceptance criteria.

From these values it is feasible to understand the evolution of the product, where the rise of the TOC value is due to a quality increase, by adding QARs and desegregating even more the chosen quality characteristics and sub-characteristics. The first iteration of the application measured 7 quality characteristics with 138 QARs in total, the second iteration analysed and measured 10 quality characteristics with 258 QARs, and the third iteration measured 11 quality characteristics with 293 QARs.

There was a need to add quality characteristics Fault-Tolerance, Recoverability, and Portability in the second iteration and Performance Efficiency in the third iteration, in order to increase quality. The set of QARs was updated based on changes in the architecture and functional requirements. Subsequently, it is important to monitor the changes from one iteration to the other in order to produce a full quality analysis.

With respect to validity threats [27], within construct validity, it is necessary to ask whether the quality level really represents the quality of the product. In response, the quality level is an aggregated value based on the full set of QARs, where the selection of each QAR were validated with stakeholders, so we considered that is not necessary to validate the value obtained per se. PQEM presents the evaluation of a system in one number. By doing so, it assumes that all quality requirements are equally important. However, it may be the case (and in reality it is often so) that the violation of a single requirement may result in an unusable product.

This drawback will be contained in future work which includes the generalization of PQEM where it will contain the definition of a set of weights which will allow to pondering each quality characteristic. This generalization will be included on a software tool that represents the automated version of the PQEM method; and it will also provide an interface to connect to another existing quality measurement tools like SonarQube and Jenkins [28]. Likewise, not only will the importance of quality characteristics be included in addition to weights, also addressing the mandatory nature of certain characteristics in the software product under evaluation. Even though it might seem small the amount of selected quality characteristics, we believe that the community is well aware of the goodness and scientific reachability of the ISO/IEC 25010 [2].
Also, the initial definition of QARs as well as whether to include all of them or just a few may distort the evolution of the product quality level, due to the fact that the TOC value is an aggregated value obtained as the sum of the quality coverages of each quality characteristic. It must not be forgotten that the QARs are almost always related to the application domain. For example, the second and third iteration of the web and mobile application [5] are embedded within the healthcare sector, while helping patients ensure their cardiac recovery.

This explicit relationship with the domain has an impact on the definition and selection of the QARs because some quality characteristics can be more important than others, regarding the viability of the product and the stakeholder's perspective. Within healthcare, if an integration to existing healthcare records is necessary, then the set of QARs for Interoperability might be larger than other non-health related application.

As part of internal validity, it is possible to say that all of those QARs belonging to Usability have a reduced subjectivity due to the number of people involved in the Usability test carried out. Also, subjectivity included in the evaluation of the QARs, when we decided to accept or reject them. But it is worth mentioning that all of them were defined in order for them to be easily verifiable, testable or measurable.

Later on, some parts of PQEM might seem to be extremely dependent on the stakeholders. In its current form without tool support, PQEM it is just a very abstract (albeit very systematic) process that only becomes concrete when it gets to metric aggregation at the end. These conditions will be improved by the creation of a catalogue that includes quality measures and questions in order to decrease the dependency of stakeholders, and increase the practical applicability of PQEM. The latter will be supported by the development of the automated tool based on the method.

Another question point is whether the TOC value is representative and useful for the stakeholders; where that value arises from the entire previous breakdown for all quality characteristics, when synthesizing the coverage of the quality characteristics. Therefore, if you need to understand that multidimensional value it is possible to go through the different levels of aggregation to understand that number in depth.

With respect to external validity, it might look like that the validation of the method is performed on a relatively small case where the product might seem small and with no applicability to industrial practice. But, actually, the application possesses several actual characteristics such as concurrency, web and mobile version, use of sensors, healthcare domain, need for high availability, among others. Also, mobile applications are becoming complex software systems that must be developed quickly and evolve continuously to fit new user requirements and execution contexts [29].

All of these characteristics realize the need to applied the PQEM method in order to analyse the quality level of the second and third iterations of HeartCare, due to the need of understanding how well was designed and implemented the application. The future implementation of a software tool to make PQEM accessible as a web will allow to replicate the method more easily, and even increase the external validation.

6.1. Implications for Research and Practice

Regarding the implications of putting the PQEM method into practice, it is feasible to mention that PQEM itself takes time to apply due to the definition and specification of Steps 2 and 3. These steps require in-depth knowledge of the software product to be developed. Therefore, it is time and cost that it is required for its applicability. A product owner will need to understand this scope to map the resources, time and associated costs in order to achieve an effective implementation for each defined iteration; regardless of the type of project.

But considering that, by itself, any process of definition and measurement of quality requires the same considerations, it is therefore necessary to approach it as part of development and not as an aggregate. The quality must go hand in hand with the development of the iterations.

Considering the size of the projects and the teams, the illustrative case shows the feasibility of the application in a small project with two iterations (one prior to this article [5], and the second and third, described above) which was implemented with a team of five/six developers, a technical leader and a project manager. Therefore, in the example we showed that measurement with PQEM is feasible in small projects for small co-located teams when there is a need for the domain that justifies the addition of time and cost to applied the method. Likewise, the application of the method may not be necessary to justify the cost, if it is considered a complex product or domain or of which it is necessary to ensure a certain level of quality.

Inside an agile environment, the use of PQEM might require more documentation and analysis to the delivery cycle due to the fact that each iteration requires a quality measurement and evaluation. In case a new standard is needed for example the European DGDR standard for privacy [30], what is important to understand is that if it is possible to extract QARs, defining goals and requirements then it is possible to apply PQEM with a different standard; achieving adaptation and flexibility.

In this article, a method was introduced that facilitates the elicitation, measurement and monitoring of QARs, and therefore its application is justified when this represents a company policy, is a requirement of the complexity of the product or domain, or is has assumed to obtain a certain level of quality.

7. CONCLUSIONS

Software engineering principles and quality goals are necessary but not sufficient for the needs of today's marketplace; because exists the necessity of shorter and iterative cycle times, and completed with fewer resources. Establishing the proper metrics to monitor the software project is essential, as well as the requirement that project managers and leaders view the entire and big picture of the development process [27-31]. Therefore, project leaders and product owners need to understand the level and quality of a software product, intuitively; which facilitates the decision to accept or reject a product.

In this context, the PQEM method [5] is introduced which assess the quality of a product by a single numeric value between 0 and 1. To calculate this value, it uses a GQM-motivated quality model that refines quality goals to quality attribute requirements (question along with a metric and acceptance criteria). The quality evaluation derived from the rate of passed quality attribute requirements.

Also, we presented a two-fold illustrative example from the healthcare sector 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 [1].

It is possible to visualize the contribution of PQEM as it obviously helped an organization to refine and concretize their (often abstract) quality requirements down to hard, measurable criteria. Consequently, with PQEM the manager can know if the project has quality problems or if the quality level is below the expected; the same as the developer who can know what the points of failure are. In the same way, the output of PQEM, that is to say that unique and multidimensional number, allows us to understand how the quality of the product evolves between iteration and iteration. As future work, the authors will develop an automated tool of PQEM, and a catalogue that include a set of suggested questions and metrics for the stakeholder to use. to Understand the Evolution of Quality through the iterations of a Software Product

ACKNOWLEDGEMENTS

This work is supported by a research grant from Engineering School, Universidad Austral, Argentina.

REFERENCES

[1] K. Mordal, N. Anquetil, J. Laval, A. Serebrenik, B. Vasilescu, and S. Ducasse. "Software quality metrics aggregation in industry." Journal of Software: Evolution and Process 25, no. 10, pp. 1117-1135, 2013.

[2] ISO/IEC 25010 (n.d), https://iso25000.com/index.php/en/iso-25000-standards/iso-25010, Last access: 1/7/2021.

[3] V. R. Basili, Software modeling and measurement: the Goal/Question/Metric paradigm. 1992.

[4] J. A. McCall, P. K. Richards, and G. F. Walters. Factors in software quality. Volume i. concepts and definitions of software quality. General Electronic co. Sunnyvale, CA, 1977.

[5] M. Falco, and G. Robiolo. "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, 2019.

[6] L. Bass, P. Clements, and R. Kazman. Software architecture in practice. Addison-Wesley Professional, 2003.

[7] J. J. Chilenski, and S. P. Miller. "Applicability of modified condition/decision coverage to software testing." Software Engineering Journal 9, no. 5 (1994): 193-200.

[8] R. Kazman, M. Klein, and P. Clements. ATAM: Method for architecture evaluation. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2000.

[9] E. Woods. "Industrial architectural assessment using TARA", Journal of Systems and Software 85, no. 9, pp. 2034-2047, 2012.

[10] H. Koziolek, D. Domis, T. Goldschmidt, P. Vorst, and R. J. Weiss. "MORPHOSIS: A lightweight method facilitating sustainable software architectures." In 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, pp. 253-257. IEEE, 2012.

[11] P. Bengtsson, N. Lassing, J. Bosch, and H. van Vliet. "Architecture-level modifiability analysis (ALMA)", Journal of Systems and Software 69, no. 1-2, pp. 129-147, 2004.
[12] S. Sarkar, G. M. Rama, and Avinash C. Kak. "API-based and information-theoretic metrics for measuring the quality of software modularization." IEEE Transactions on Software Engineering 33, no. 1 (2006): 14-32.
[13] A. R. Hevner, S. T. March, J. Park, and S. Ram. "Design science in information systems research." MIS quarterly, pp. 75-105, 2004.
[14] P. Runeson, E. Engström, and M-A Storey. "The design science paradigm as a frame for empirical software engineering." In Contemporary empirical methods in software engineering, pp. 127-147. Springer, Cham, 2020.
[15] P. Jain, A. Sharma, and L. Ahuja. "The Impact of Agile Software Development Process on the Quality of Software Product." In 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 812-815. IEEE, 2018.
[16] H. R. Neri, and G. Horta Travassos. "Measuresoftgram: a future vision of software product quality." In Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 1-4. 2018.
[17] V. R. Caldiera, G. Basili, and H. Dieter Rombach. "The goal question metric approach." Encyclopedia of software engineering, pp. 528-532, 1994.
[18] A. S. Nuñez-Varela, H. G. Pérez-Gonzalez, F. E. Martínez-Perez, and C. Soubervielle-Montalvo. "Source code metrics: A systematic mapping study." Journal of Systems and Software 128, pp. 164-197, 2017.
[19] Segue Technologies, September 3, 2015, What Characteristics Make Good Agile Acceptance Criteria? https://www.seguetech.com/what-characteristics-make-good-agile-acceptance-criteria/. Last access: 1/7/2021.
[20] J. Dick, E. Hull, and K. Jackson. Requirements engineering. Springer, 2017.
[21] J. Estdale, and E. Georgiadou. "Applying the ISO/IEC 25010 quality models to software product." In European Conference on Software Process Improvement, pp. 492-503. Springer, Cham, 2018.
[22] S. Jiménez-Fernández, P. De Toledo, and F. Del Pozo. "Usability and interoperability in wireless sensor networks for patient telemonitoring in chronic disease management." IEEE Transactions on Biomedical Engineering 60, no. 12 (2013): 3331-3339.
[23] R. van Solingen, D.M. Rini, and E. W. Berghout. The Goal/Question/Metric Method: a practical guide for quality improvement of software development. McGraw-Hill, 1999.
[24] V. Gay, P. Leijdekkers, and E. Barin. "A mobile rehabilitation application for the remote monitoring of cardiac patients after a heart attack or a coronary bypass surgery." In Proceedings of the 2nd international conference on pervasive technologies related to assistive environments, pp. 1-7, 2009.
[25] P. Kakria, N. K. Tripathi, and P. Kitipawang. "A real-time health monitoring system for remote cardiac patients using smartphone and wearable sensors." International journal of telemedicine and applications, 2015.
[26] Z. Li, P. Avgeriou, and P. Liang. "A systematic mapping study on technical debt and its management." Journal of Systems and Software 101 (2015): 193-220.
[27] V. R. Basili, R. W. Selby, and D. H. Hutchens. "Experimentation in software engineering." IEEE Transactions on software engineering 7 (1986): 733-743.
[28] M. Falco, E. Scott, G. Robiolo, “Overview of an Automated Framework to Measure and Track the Quality Level of a Product”, In IEEE ARGENCON 2020, V Biennal Congress of IEEE Argentina Section.
[29] G. Hecht, R. Rouvoy, N. Moha, and L. Duchien. "Detecting antipatterns in android apps." In 2015 2nd ACM international conference on mobile software engineering and systems, pp. 148-149, IEEE, 2015.
[30] European Union, Regulartion 2016/679 of the European Parliament and of the Council. General Data Protection Regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN, Last access: 1/7/2021.
[31] R. T. Futrell, L. I. Shafer, and D. F. Shafer. Quality software project management. Prentice Hall PTR, 2001.
[32] M. Falco, and G. Robiolo, “Product Quality Evaluation Method (PQEM): A Comprehensive Approach for the Software Product Life Cycle”, 7th International Conference on Software Engineering (SOFT) 2021.

AUTHORS

Mariana Falco. PhD in Engineering student. Information System Engineer. She is a teacher within the Engineer School at Universidad Austral (UA), where she also does her research on Software Engineering, Quality Measurement, and Quality Metrics. She collaborates with fellow colleagues on researches related to information technologies applied to different domains. She is part of the LIDTUA Lab (UA) as well as the Research & Development Lab (UA), as a researcher and project manager.

Gabriela Robiolo. PhD in Information Sciences. She is currently a full-time professor at Universidad Austral, within the Engineering School. She is a researcher on Software Engineering, and Quality Measurement, and also she collaborates with other researchers on related topics. She is part of the LIDTUA Lab (UA) as well as the Research & Development Lab (UA), as one of the principal researchers.